Canonical Derivations with Negative Application Conditions

Andrea Corradini[1] and Reiko Heckel[2]

[1] Dipartimento di Informatica, Università di Pisa, Italy
andrea@di.unipi.it
[2] University of Leicester, UK
reiko@mcs.le.ac.uk

Abstract. Using graph transformations to specify the dynamics of distributed systems and networks, we require a precise understanding of concurrency. Negative application conditions (NACs) are an essential means for controlling the application of rules, extending our ability to model complex systems. A classical notion of concurrency in graph transformation is based on shift equivalence and its representation by canonical derivations, i.e., normal forms of the shift operation anticipating independent steps. These concepts are lifted to graph transformation systems with NACs and it is shown that canonical derivations exist for so-called incremental NACs.

Keywords: graph transformation, canonical derivation, incremental NACs.

1 Introduction

Graph Transformation Systems (GTS) provide a visual formal specification technique and computational model for concurrent and distributed systems, where graphs modelling system states evolve through the application of rules with local effects. A significant body of literature is dedicated to the study of parallelism and concurrency of graph transformation systems [7,14,5,1,2].

The classical theory includes, among others, the definition of the parallel composition of several rules (by coproduct, i.e. disjoint union) and its application to a graph, and of sequential independence between two consecutive rule applications [7]. These notions are exploited in the Church-Rosser theorem that shows that sequentially independent rule applications can be switched obtaining the same resulting graph. Furthermore, as stated by the Parallelism theorem, the same effect can be obtained by applying the parallel composition of the two rules to the start graph. This leads to the definition of a natural equivalence on the set of parallel derivations, i.e. on sequences of possibly parallel rule applications, called the *shift equivalence*: parallel derivations that differ only by the order in which independent rule applications appear are considered to be equivalent.

Kreowski showed in [14] that shift-equivalence classes of parallel derivations have canonical representatives, obtained by anticipating as much as possible the rule applications. Such representatives are called *canonical derivations*, and they feature an "early maximal parallelism": each rule application depends on at least one rule application in the preceding parallel rule application, if one exists.

Using graph transformations to specify the dynamics of distributed systems and networks, we require a precise understanding of concurrency. The notions of shift equivalence and canonical derivation are the original expression of such an understanding. They have been fundamental to more recent research on concurrency of graph transformation systems (including graph processes [5] and unfolding [3]), as well as their generalisation to transformation systems based on ($\mathcal{M}$-)adhesive categories [6,2].

Negative application conditions (NACs) [9] are an essential means for controlling the application of rules, extending our ability to model complex systems. However, a corresponding notion of concurrency has been missing so far. This paper addresses the generalisation of the results on canonical derivations to rules with NACs.

A NAC allows one to describe a "forbidden context", whose presence around a match inhibits the application of the rule. NACs introduce new kinds of dependencies among rule applications, as stressed in [11,4], due to the fact that a forbidden context for a rule can be generated by two sequentially independent rule applications. As a consequence, unlike the case without NACs, sequential independence of rule applications with NACs (as defined in [16]) is not stable under switching independent rule applications: as recalled in Sect. 3 two independent consecutive rule applications may become dependent if both are switched with a third rule application, independent of both. As shown in [4], this problem does not occur if the NACs are *incremental*, that is, if each morphism embedding the left-hand side of a rule into a forbidden context can be decomposed in an essentially unique way.

After presenting basic definitions in Sect. 2, Sect. 3 recalls concepts and results on independence of rule applications, including the definition of incremental NACs and some of their properties. The original contribution starts with Sect. 4 where we first introduce an operational semantics for transformation systems. Given a set of pairwise independent rule matches in a graph, we compose them via an *amalgamation* construction obtaining a *step*, whose application to a graph has the same effect of the parallel application of the given rules. In the presence of NACs, such a step is not always serializable. We call it *safe* if any interleaving of its constituent rule applications satisfies all the NACs, that is, if it is serializable in all possible ways. This might be expensive to check, but we show that if NACs are incremental then safety of a step made of several rules can be checked by a pairwise analysis of the rules.

Next in Sect. 5 we discuss existence of canonical derivations made of steps. We show that they do not exist in general for rules with arbitrary NACs, but are guaranteed to exist if all NACs are incremental. Finally, Sect. 6 provides a conclusion and sketches future developments.

2 Basic Definitions

In this paper we use the double-pushout approach [7] to graph transformation with negative application conditions [9]. However, we will state all definitions and results at the level of adhesive categories [15]. We recall that a category is *adhesive* if (1) it has pullbacks, (2) it has pushouts along monomorphisms (hereafter *monos*), and (3) all pushouts along a mono are *Van Kampen squares.* That means, when such a pushout is the bottom face of a commutative cube such as in the diagram to the right, whose rear faces are pullbacks, the top face is a pushout if and only if the front faces are pullbacks. In any adhesive category pushouts along monos are also pullbacks, and if a pushout complement of two composable arrows $K \xrightarrow{l} L \xrightarrow{m} G$ with l mono exists, then it is unique (up to iso).

$$\begin{tikzcd}
 & & A' \arrow[dll] \arrow[dr] \arrow[dd, dotted] & \\
B' \arrow[dr] \arrow[dd] & & & C' \arrow[dll] \arrow[dd] \\
 & D' \arrow[dd] & A \arrow[dll, dotted] \arrow[dr, dotted] & \\
B \arrow[dr] & & & C \arrow[dll] \\
 & D & &
\end{tikzcd}$$

As an example, the category of typed graphs for a fixed type graph TG, defined as the slice category $(\mathbf{Graph} \downarrow TG)$, is adhesive. In the rest of the paper, all objects and arrows live in an arbitrary but fixed adhesive category $\mathbf{C}$.

A *rule* $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ consists of a span of two monos l and r. A *redex* in a graph G is a pair (p, m), where p is a rule and $m : L \to G$ is a mono, called a *match.* Given a redex (p, m), a *transformation* $G \overset{p,m}{\Longrightarrow} H$ from G to H exists if the *double-pushout (DPO) diagram* to the right can be constructed, where (1) and (2) are both pushouts.[1]

$$\begin{tikzcd}
L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] \arrow[dr, phantom, "(2)"] & R \arrow[d, "m^*"] \\
G & D \arrow[l, "l^*"] \arrow[r, "r^*"'] & H
\end{tikzcd}$$

The applicability of rules can be restricted by imposing some negative constraints, defined as follows. A *(negative) constraint* over an object L is a mono $n : L \to N$. A mono $m : L \to G$ satisfies n (written $m \models n$) iff there is no mono $q : N \to G$ such that $n; q = m$. A negative application condition (NAC) is a set of constraints. A mono $m : L \to G$ *satisfies* a NAC ϕ over L (written $m \models \phi$) if and only if m satisfies every constraint, i.e., $\forall n \in \phi, m \models n$. In this paper we shall consider only monic matches and monic constraints.

A *graph transformation system (GTS)* $\mathcal{G} = (P, \pi)$ consists of a set of rule names P and a function π assigning to each name p a rule $\pi(p) = L \xleftarrow{l} K \xrightarrow{r} R$. A *conditional GTS* (P, π, Φ) consists of an *underlying GTS* (P, π) and a function Φ providing for each $p \in P$ a NAC $\Phi(p)$ over $\pi(p)$'s left-hand side. A *derivation* in a GTS $\mathcal{G} = (P, \pi)$ is a finite sequence of transformations $s = (G_0 \overset{p_1,m_1}{\Longrightarrow} G_1 \overset{p_2,m_2}{\Longrightarrow} \cdots \overset{p_n,m_n}{\Longrightarrow} G_n)$ with $p_i \in P$. A *conditional derivation* in a conditional GTS (P, π, Φ) is a derivation in its underlying GTS such that each $G_{i-1} \overset{p_i,m_i}{\Longrightarrow} G_i$ is a *conditional transformation*, that is, it is a DPO diagram where match m_i satisfies the NAC $\Phi(p_i)$.

We write $s : G_0 \overset{*}{\Longrightarrow} G_n$ for a generic derivation and, given $s' : G_k \overset{*}{\Longrightarrow} G_m$ with $G_n = G_k$, denote their sequential composition by $s; s' : G_0 \overset{*}{\Longrightarrow} G_m$.

[1] Note that we stick to DPO rewriting with monic matches only (see [10]).

3 Independence and Switch Equivalence

This section recalls the notions of parallel and sequential independence and switch equivalence, and illustrates the problem that sequential independence with NACs is not stable under switching. In the DPO approach, two transformations from the same graph $G_1 \overset{p_1,m_1}{\Longleftarrow} G_0 \overset{p_2,m_2}{\Longrightarrow} G_2$ as in the left of the diagram below are *parallel independent* iff there exist morphisms $i : L_1 \to D_2$ and $j : L_2 \to D_1$ such that $j;l_1^* = m_2$ and $i;l_2^* = m_1$.

$$\begin{tikzcd}
R_1 \arrow[d, "m_1^*"] & K_1 \arrow[l, "r_1"'] \arrow[r, "l_1"] \arrow[d] & L_1 \arrow[dr, "m_1"] \arrow[drrr, dotted, "i"] & & L_2 \arrow[dl, "m_2"'] \arrow[dlll, dotted, "j"'] & K_2 \arrow[l, "l_2"'] \arrow[r, "r_2"] \arrow[d] & R_2 \arrow[d, "m_2^*"] \\
G_1 & D_1 \arrow[l, "r_1^*"] \arrow[rr, "l_1^*"'] & & G & & D_2 \arrow[ll, "l_2^*"] \arrow[r, "r_2^*"'] & G_2
\end{tikzcd}
\qquad
\begin{tikzcd}
L_1 \arrow[d, "m_1"] & K_1 \arrow[l, "l_1"'] \arrow[r, "r_1"] \arrow[d] & R_1 \arrow[dr, "m_1^*"] \arrow[drrr, dotted, "i"] & & L_2 \arrow[dl, "m_2"'] \arrow[dlll, dotted, "j"'] & K_2 \arrow[l, "l_2"'] \arrow[r, "r_2"] \arrow[d] & R_2 \arrow[d, "m_2^*"] \\
G_0 & D_1 \arrow[l, "l_1^*"] \arrow[rr, "r_1^*"'] & & G_1 & & D_2 \arrow[ll, "l_2^*"] \arrow[r, "r_2^*"'] & G_2
\end{tikzcd}$$

Two consecutive transformations $G_0 \overset{p_1,m_1}{\Longrightarrow} G_1 \overset{p_2,m_2}{\Longrightarrow} G_2$ as in the right of the diagram above are *sequentially independent* iff there exist morphisms $i : R_1 \to D_2$ and $j : L_2 \to D_1$ such that $j;r_1^* = m_2$ and $i;l_2^* = m_1^*$.

The Local Church-Rosser theorem (Thm. 3.20 of [6], called "LCR" in the following) states that (1) given the two parallel independent transformations, there are transformations $G_1 \overset{p_2,m_2'}{\Longrightarrow} X \overset{p_1,m_1'}{\Longleftarrow} G_2$ such that $G_0 \overset{p_1,m_1}{\Longrightarrow} G_1 \overset{p_2,m_2'}{\Longrightarrow} X$ (and $G_0 \overset{p_2,m_2}{\Longrightarrow} G_2 \overset{p_1,m_1'}{\Longrightarrow} X$) are sequentially independent and that (2) given sequentially independent transformations $\sigma = G_0 \overset{p_1,m_1}{\Longrightarrow} G_1 \overset{p_2,m_2}{\Longrightarrow} G_2$, there exists $\sigma' = G_0 \overset{p_2,m_2'}{\Longrightarrow} G_2' \overset{p_1,m_1'}{\Longrightarrow} G_2$ such that $G_1 \overset{p_1,m_1}{\Longleftarrow} G_0 \overset{p_2,m_2'}{\Longrightarrow} G_2'$ are parallel independent. In the last case we write $\sigma \sim_{sw} \sigma'$ to denote that derivations σ and σ' are in the *switch relation*. This relation extends to an equivalence on derivations of arbitrary finite length, called *switch equivalence*, by setting $s\, ;\, s_1\, ;\, s' \sim_{sw} s\, ;\, s_2\, ;\, s'$ if $s_1 \sim_{sw} s_2$, for arbitrary derivations s, s', and closing under transitivity.

The definition of parallel independence carries over to conditional transformations [17] by requiring that the matches $j;r_1^*$ for p_2 in G_1 and $i;r_2^*$ for p_1 in G_2 satisfy their respective NACs. Similarly, for sequential independence, the match for p_2 in G_0 given by $m_2' = j;l_1^*$ must satisfy the NAC of p_2 and the induced match of p_1 into graph G_1' obtained by $G_0 \overset{p_2,m_2'}{\Longrightarrow} G_1'$ must satisfy the NAC of p_1. With these definitions, the statements of LCR carry over verbatim to the conditional case.

However, as mentioned in the introduction and as shown in the next example, in the conditional case the sequential independence of two transformations is not preserved by switch equivalence, a property that is known to hold for derivations without NACs by the results in [14]. The same problem also holds if the coarser *permutation equivalence* is considered on conditional derivations, as defined in [12].

Example 1 (independence with NACs). Along the paper we use the rules of Fig. 1 to show the problems that arise using rules with arbitrary NACs. Rule p_1 has a negative constraint $n_1 : L_1 \to N_1$: it creates a new node only if the node it is

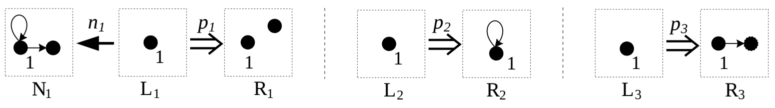


Fig. 1. Three simple graph transformation rules

applied to does not have both a loop and an edge to another node; p_2 adds a loop to a node; p_3 adds to a node an edge connecting a new node. Recall that rules are spans: the mid graph is omitted as it coincides with the left-hand side.

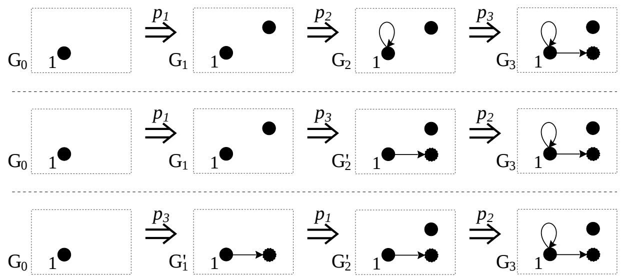


Fig. 2. Independence of p_1 and p_2 is not preserved by switching with p_3

Fig. 2 shows three conditional derivations from G_0 applying p_1, p_2 and p_3 in different orders. All rules are applied to node labeled 1. In the first derivation $s = G_0 \xRightarrow{p_1} G_1 \xRightarrow{p_2} G_2 \xRightarrow{p_3} G_3 = (s_1; s_2; s_3)$ (top), s_1 and s_2 are sequentially independent, and so are s_2 and s_3. After switching s_2 and s_3 by point (2) of LCR we obtain $s' = (s_1; s'_3; s'_2)$ (middle of Fig. 2), thus we have $s \sim_{sw} s'$. Since s_1, s'_3 are independent, we can perform a further switch obtaining $s'' = (s''_3; s'_1; s'_2)$ (bottom). However, s'_1 and s'_2 are sequentially dependent in s'', because the match for p_1 into G''_2, obtained as $G'_1 \xRightarrow{p_2} G''_2$, does not satisfy its NAC. Hence, sequential independence may not be preserved in equivalent derivations.

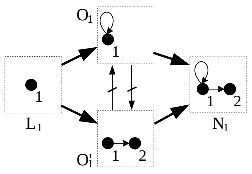
Essentially, the problem identified in Ex. 1 is due to the fact that the NAC of rule p_1 is made of two parts that can be created, in any order and independently, by p_2 and p_3. In fact, as shown in [4], the problem disappears if all NACs are *incremental* in the following sense.

Definition 1 (incremental monos and NACs). *A mono $f\colon A \to B$ is called* incremental*, if for any pair of decompositions $g_1; g_2 = f = h_1; h_2$ as in the diagram to the right where all morphisms are monos, there is either a mediating morphism $o : O \to O'$ or $o' : O' \to O$, such that the resulting triangles commute. A monic NAC ϕ over L is* incremental *if each constraint $(n : L \to N) \in \phi$ is incremental.*

$$\begin{tikzcd}
 & O \arrow[dr, "g_2"] \arrow[d, dotted, shift right, "o"'] & \\
A \arrow[ur, "g_1"] \arrow[rr, "f"] \arrow[dr, "h_1"'] & & B \\
 & O' \arrow[ur, "h_2"'] \arrow[u, dotted, shift right, "o'"'] &
\end{tikzcd}$$

Equivalently, f above is incremental if the pullback of arrows g_2 and h_2 has a projection that is an iso.[2]

Example 2 (Incremental NACs). The diagram to the right shows that the negative constraint $n_1 : L_1 \to N_1$ of rule p_1 of Ex. 1 is not incremental, because N_1 extends L_1 in two independent ways: by the loop on 1 in O_1, and by the outgoing edge with an additional node 2 in O'_1. Indeed, there is no mediating arrow from O_1 to O'_1 or vice versa relating these two decompositions.



Incremental monos in the category of directed graphs can only represent forbidden contexts made of a single edge or of a single node (possibly with an incident edge). Indeed, every incremental mono between graphs can be decomposed in at most two non-isomorphic monos: for example, $\{\bullet\} \to \{\bullet\!\to\!\bullet\} = \{\bullet\} \to \{\bullet \quad \bullet\} ; \{\bullet \quad \bullet\} \to \{\bullet\!\to\!\bullet\}$. Despite that, as discussed in [4] incremental NACs are sufficiently expressive for several applications of GTSs. For example, [13] describes a significant model transformation problem addressed with a graph transformation system made of almost 500 rules. NACs are used extensively, and only a few rules have non-incremental NACs, but could be converted to rules with incremental NAC only with a small increase in execution time.

The next result shows that independence is stable under switching in the case of incremental NACs. It follows easily from Thm. 1 of [4], where only the sequential case was considered.

Theorem 1 (indep. stable under switch). *In the cube on the right, edges represent conditional transformations $t_i^j = (p_i, m_i^j)$ for $1 \leq i, j \leq 3$, where all rules have incremental NACs only. Furthermore, each face $A \overset{a}{\Leftarrow} B \overset{b}{\Leftarrow} C \overset{c}{\Rightarrow} D \overset{d}{\Rightarrow} A$ is a* switch operation *iff either b and c are parallel independent and a, d are obtained by point (1) of LCR, or $b\,;a$ (resp. $c\,;d$) are sequentially independent, and $c\,;d$ (resp. $b\,;a$) are obtained by point (2) of LCR. Then we have:*

$$\begin{tikzcd}
& \bullet \arrow[rr, Rightarrow, "t_1^2"] \arrow[dd, Rightarrow, "t_2^2"] & & \bullet \arrow[dd, Rightarrow, "t_2^3"] \\
\bullet \arrow[ur, Rightarrow, "t_3"] \arrow[rr, Rightarrow, "t_1"] \arrow[dd, Rightarrow, "t_2"] & & \bullet \arrow[ur, Rightarrow, "t_3^1"] \arrow[dd, Rightarrow, "t_2^1"] & \\
& \bullet \arrow[rr, Rightarrow, "t_1^3"] & & \bullet \\
\bullet \arrow[ur, Rightarrow, "t_3^2"] \arrow[rr, Rightarrow, "t_1^1"] & & \bullet \arrow[ur, Rightarrow, "t_3^3"] &
\end{tikzcd}$$

1. *If the top and right faces are switch operations, the front face is a switch operation iff the back face is.*
2. *If the top and front faces are switch operations, the right face is a switch operation iff the left face is.*

Another way of reading item 1 above is that $t_1; t_2^1; t_3^3 \sim_{sw} t_3; t_1^2; t_2^3$ implies that t_1 and t_2^1 are sequentially independent iff t_1^2 and t_2^3 are, i.e., as shown in [4] sequential independence is stable under switching (the property that did not hold for Ex. 1 because of the non-incremental NAC of p_1). By symmetry, this implies analogous statements assuming that the faces in the top and back, front and bottom, front and right, left and back, left and bottom, respectively, represent switch operations.

[2] We are grateful to an anonymous referee for this observation.

Item 2 states stability of parallel independence under switch, so statements symmetrical to 2 hold by assuming that the faces in the front and left, left and top, respectively, represent switch operations. This can be shown by exploiting the duality between parallel and sequential independency, i.e., t_1 and t_2 are sequentially independent iff the inverse t_1^{-1} of t_1 and t_2 are parallel independent.

4 Conditional Step Derivations

We investigate conditional step derivations as a computational model where, in each state, a number of rules with NACs can be applied in parallel as long as they are not in conflict, i.e., the same effect can be obtained by applying them sequentially. We face two challenges. First, to define a notion of parallel transformation the traditional approach is to construct a parallel rule as the disjoint union of all the rules to be applied. If the matches of these rules are not disjoint, this results in a non-monic match for the parallel rule. However, we have restricted our matches to monos because they work more naturally with the notion of satisfaction employed by NACs. That means, rather than parallel rules based on disjoint union, we consider amalgamated rules which merge individual rules wherever their matches overlap. We will call the application of such amalgamated rules to a graph a *step*.

The notion of amalgamated transformation and derivation has been introduced for graphs and HLR systems [18], and it is adapted here to adhesive categories and extended by NACs. A similar construction is also presented in [10], but for two rules only. Paper [8] considers amalgamation of several rules with *nested* application conditions, which generalize negative ones, in arbitrary ($\mathcal{M}$-)adhesive categories. However, rules are amalgamated along *a single* common subrule, thus this approach is not applicable to our setting.

The second challenge lies in ensuring that conditional steps do not contain conflicting application conditions. We call these conditional steps *safe* in the sense that they can be serialised in any order with the same effect, so that conditional derivations and conditional step derivations induce the same reachability relation on graphs. While for the unconditional case this follows from the Parallelism theorem [7,6], in the conditional case checking serializability of a step requires to check that the negative constraints of a rule cannot be produced by applying an arbitrary subset of the remaining rules, which is computationally very costly. We show that if all NACs are incremental, in order to check that a conditional step is safe a pairwise analysis of the involved rules is sufficient.

A different approach is taken in [17], where to ensure serialisation binary parallel rules are equipped with additional NACs, besides those of the component rules. The process can be iterated, but the high number of constraints generated would make the approach difficult to apply.

We will define steps with a two-level construction. For a multiset of redexes enabled individually, first an amalgamated rule is constructed, then it is applied at a match obtained by combining the individual matches. The amalgamated rule is obtained using a colimit construction called *pushout star*.

Proposition 1 (existence of pushout stars). *Given a family of spans $(O_i \leftarrow O_{ij} \rightarrow O_j)_{i<j\in I}$ connecting a finite collection of objects $(O_i)_{i\in I}$, its colimit, denoted $POS(O_i \rightarrow O \leftarrow O_j)_{i<j\in I}$, is called a* pushout star *(see [7,18]).*

Let $I = \{1, \dots, n\}$, $\{m_i : K_i \rightarrow G\}_{i\in I}$ be a set of monos, and $(K_i \leftarrow K_{ij} \rightarrow K_j)_{i<j\in I}$ be a family of spans in an adhesive category, where $K_{ij}K_iK_jG$ is a pullback for each $i < j \in I$. Then the colimit of this family, i.e. $POS(K_i \rightarrow K \leftarrow K_j)_{i<j\in I}$, exists.

Definition 2 ((conditional) steps and step derivations). *Given a graph transformation system (P, π), assume a graph G and a finite multiset of redexes $S = [(p_i, m_i)]_{i\in I}$ with $p_i \in P$, $\pi(p_i) = L_i \xleftarrow{l_i} K_i \xrightarrow{r_i} R_i$, and $m_i : L_i \rightarrow G$ for $i \in I = \{1, \dots, n\}$.*[3] *The multiset of redexes S is* enabled *in G if*

(1) there exist transformations $G \overset{p_i,m_i}{\Longrightarrow} H_i$ for all $i \in I$;
(2) for all $i \neq j \in I$, transformations $G \overset{p_i,m_i}{\Longrightarrow} H_i$ and $G \overset{p_j,m_j}{\Longrightarrow} H_j$ are parallel independent.

Then a step *$G \overset{S}{\Longrightarrow} H$ in (P, π) is obtained as shown in the left of Fig. 3 by*

1. *constructing pullbacks $K_i \leftarrow K_{ij} \rightarrow K_j$ of $K_i \xrightarrow{l_i} L_i \xrightarrow{m_i} G \xleftarrow{m_j} L_j \xrightarrow{l_j} K_j$ for all $i < j \in I$;*
2. *building (thanks to Prop. 1) the amalgamated rule p_S with $\pi(p_S) = L \xleftarrow{l} K \xrightarrow{r} R$ by*
   - *$L = POS(L_i \rightarrow L \leftarrow L_j)_{i<j\in I}$ of $(L_i \xleftarrow{l_i} K_i \leftarrow K_{ij} \rightarrow K_j \xrightarrow{l_j} L_j)_{i<j\in I}$*
   - *$K = POS(K_i \rightarrow K \leftarrow K_j)_{i<j\in I}$ of $(K_i \leftarrow K_{ij} \rightarrow K_j)_{i<j\in I}$*
   - *$R = POS(R_i \rightarrow R \leftarrow R_j)_{i<j\in I}$ of $(R_i \xleftarrow{r_i} K_i \leftarrow K_{ij} \rightarrow K_j \xrightarrow{r_j} R_j)_{i<j\in I}$*

   with l, r induced by the universal property of the POS forming K;
3. *applying p_S at the match $m_S : L \rightarrow G$ induced by the universal property of the POS forming L. This is possible because it can be shown that under the above hypotheses the pushout complement of l and m_S exists.*

A conditional step *in a conditional GTS (P, π, Φ) is a step $G \overset{S}{\Longrightarrow} H$ in (P, π) such that for each $i \in I$, match m_i satisfies $\Phi(p_i)$.* (Conditional) step derivations[4] *are sequences of (conditional) steps, defined as expected.*

If S is a singleton, step $G \overset{S}{\Longrightarrow} H$ specialises to a transformation. If S is empty, the amalgamated rule is empty and we assume $G = H$. Note that steps are based on *multi*sets of redexes. This allows auto-concurrency, i.e., applying a redex twice in the same step is possible if it is not in conflict with itself. In the

[3] We denote a finite multiset over a set X as $[x_1, \dots, x_n]$, where $x_i \in X$ for all $0 < i \leq n$ and the order is irrelevant. Multiset union is denoted by juxtaposition, and a singleton multiset $[x]$ is sometimes represented simply as x.

[4] Based on this, the title of this paper should really refer to Canonical Step Derivations with NACs. We stick to the more traditional title for consistency with the literature.

$$\begin{tikzcd}
 & K_{ij} \arrow[dl] \arrow[ddr] & & K_{ij} \arrow[ll] \arrow[rr] \arrow[dl] \arrow[ddr] & & K_{ij} \arrow[dl] \arrow[ddr] & \\
L_i \arrow[ddr] \arrow[dddr, bend right, "m_i"'] & & K_i \arrow[ll, "l_i"'] \arrow[rr, "r_i"] \arrow[d] \arrow[ddr] & & R_i \arrow[d] \arrow[ddr] & & \\
 & & L_j \arrow[ddl, "m_j"] & & K_j \arrow[ll, "l_j"'] \arrow[rr, "r_j"] \arrow[dl] & & R_j \arrow[dl] \\
 & L \arrow[d, "m_S"] & & K \arrow[ll, "l"'] \arrow[rr, "r"] \arrow[d] & & R \arrow[d] & \\
 & G & & D \arrow[ll, "l^*"'] \arrow[rr] & & H &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & & L_i \arrow[d, "m_i"] \arrow[dd, dotted] \arrow[ddd, bend left, "m'_i"] & K_i \arrow[l] \arrow[r] \arrow[d] \arrow[dd, dotted] & R_i \arrow[d] \arrow[dd, dotted] \\
L_j \arrow[r] \arrow[rr, bend left, "m_j"] \arrow[rrr, dotted, bend right] & L_- \arrow[r, "m_-"] & G & D_i \arrow[l] \arrow[r] & G_i \\
K_j \arrow[u] \arrow[d] \arrow[r] \arrow[rrr, dotted, bend right] & K_- \arrow[u] \arrow[d] \arrow[r] & D_- \arrow[u] \arrow[d] & D \arrow[l] \arrow[r] \arrow[u] \arrow[d] & D'_- \arrow[u] \arrow[d] \\
R_j \arrow[r] \arrow[rrr, dotted, bend right] & R_- \arrow[r] & H_- & D'_i \arrow[l] \arrow[r] & H
\end{tikzcd}$$

Fig. 3. A step $G \overset{S}{\Longrightarrow} H$ and a serialisation

unconditional case, a step is a special case of amalgamated transformation with constant interface rules [18].

A step is called *safe* if it can be serialised in any order.

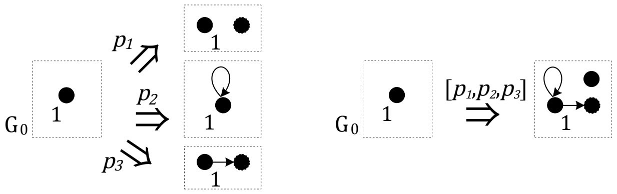
Definition 3 (serialisation, parallelisation, safe step, residual). *A (conditional) step $G \overset{S}{\Longrightarrow} H$* has a serialisation *$G \overset{S_-}{\Longrightarrow} H_- \overset{s'_i}{\Longrightarrow} H$ or $G \overset{s_i}{\Longrightarrow} G_i \overset{S'_-}{\Longrightarrow} H$ if such derivations follow the diagram on the right of Fig. 3, where*

- *$s_i = (p_i, m_i)$ and $S = [s_i]S_-$.*
- *$L_i \to D_-$ exists such that $L_i \to D_- \to G = m_i$, allowing to define m'_i as the* residual match *of m_i by $L_i \to D_- \to H_-$, and $s'_i = (p_i, m'_i)$*
- *for all $(p_j, m_j) \in S_-$, an $L_j \to D_i$ exists such that $L_j \to D_i \to G = m_j$ allowing to define $m'_j = L_j \to D_i \to G_i$ as the residual match of m_j, and S'_- from S_- by replacing, in each (p_j, m_j), match m_j by its residual m'_j.*

Vice versa, $G \overset{S}{\Longrightarrow} H$ is a parallelisation *of either of the two derivations. A step $G \overset{S}{\Longrightarrow} H$ is* safe *if either S is empty or a singleton, or for all $s_i \in S$ it has serialisations $G \overset{S_-}{\Longrightarrow} H_- \overset{s'_i}{\Longrightarrow} H$ and $G \overset{s_i}{\Longrightarrow} G_i \overset{S'_-}{\Longrightarrow} H$ where $G \overset{S_-}{\Longrightarrow} H_-$ and $G_i \overset{S'_-}{\Longrightarrow} H$ are safe.*

In the unconditional case, due to pairwise parallel independence of the redexes all steps are safe (see also [18]). Instead, spelling out the definition, a conditional step $G \overset{S}{\Longrightarrow} H$ is safe if for all $i \in I$, the residual match $m'_i : L_i \to H_-$ satisfies $\Phi(p_i)$ and for all $j \in I \setminus \{i\}$ the residual match $m'_j : L_j \to G_i$ satisfies $\Phi(p_j)$. That means, to decide if the step is safe each individual transformation has to be checked for parallel independence against the amalgamation of the rest of the redexes in the step. Because this applies recursively, there are $\mathcal{O}(n!)$ independence checks. Each of these requires the construction of the individual or amalgamated transformations in order to check if NACs are satisfied in the resulting graphs. That means, even if we assume an efficient implementation of individual transformations, an operational semantics of conditional GTSs based on conditional step derivations would not be efficient, as it requires to check safety of each step.

Example 3 (unsafe steps with general NACs). Consider the rules of Fig. 1 and their matches in graph G_0 below. Clearly all rules are applicable, and the three transformations below to the left are pairwise parallel independent. Therefore the conditional step $[p_1, p_2, p_3]$ depicted on the right is well-defined. However, it is not safe: the serialisation $p_2\,;\,p_3\,;\,p_1$ is not possible because if the (sub)step $[p_2, p_3]$ is fired first the NAC of p_1 is not satisfied in the resulting graph. Other serialisations are possible, though, including those depicted in Fig. 2.



Instead, using incremental NACs only, the situation is comparable to the unconditional case. In fact, as shown in Thm. 2, it is sufficient to consider pairwise parallel independence.

Theorem 2 (safety with incremental NACs). *A conditional step $G \overset{S}{\Longrightarrow} H$ with incremental NACs only is safe if for all redexes $s \neq t \in S$, the corresponding conditional transformations are parallel independent.*

Proof. Following Def. 3 we work by induction. For steps based on 1 or 0 redexes, the statement holds by definition. Assume that all conditional steps are safe if they are based on no more than n redexes, and consider $G \overset{S}{\Longrightarrow} H$ with $S = [t_1, \ldots, t_n, t_{n+1}]$. Since t_{n+1} is parallel independent of each of the $t_1, \ldots, t_n$ we can use the Local Church-Rosser theorem to form a derivation $t_{n+1}; [t'_1, \ldots, t'_n]$ such that t_{n+1} is sequentially independent of each $t_i, 1 \leq i \leq n$. This provides one of the two sequentialisations required, since item 2 of Thm. 1 implies that the $t'_1 \ldots t'_n$ are still pairwise parallel independent.

Vice versa, we can transform $[t_1, \ldots, t_n, t_{n+1}]$ into $t_1; \ldots; t'_n; t'_{n+1}$ by successively delaying independent steps and using Local Church Rosser and item 2 of Thm. 1 to show that this operation preserves independence of the remaining steps. Finally, the resulting conditional derivation $t_1; \ldots; t'_n; t_{n+1}$ can be transformed into the conditional step derivation $[t_1, \ldots, t''_n]; t_{n+1}$ again preserving pairwise independence due to Local Church Rosser and Thm. 1. □

Given conditional transformations $G_0 \overset{p_1,m_1}{\Longrightarrow} G_1$ and $G_0 \overset{p_2,m_2}{\Longrightarrow} G_2$, as recalled in Sect. 3, they are parallel independent if the underlying unconditional transformations are, and the residual match $j; r_1^*$ for rule p_2 in G_1 satisfies the NACs $\Phi(p_2)$ (and symmetrically for match $i; r_2^*$). The next result shows that if NACs are incremental, then it is sufficient to check for each negative constraint $n : L_2 \to N$ of p_2, if the negative part $N \setminus n(L_2)$ of n already occurs in the right-hand side of p_1. More abstractly, this negative part of N is characterised by the initial pushout over constraint n, defined as follows.

Definition 4 (initial pushout). *The rectangle with vertices $ABNM$ to the right is an initial pushout over $f : M \to N$ if it is a pushout and for every pushout $IJNM$ there exist unique morphisms $A \to I$ and $B \to J$ such that the triangles commute and $ABJI$ is a pushout.*

$$\begin{tikzcd}
A \arrow[rr] \arrow[dd] \arrow[dr] & & B \arrow[dd] \arrow[dr] & \\
& I \arrow[rr] \arrow[dl] & & J \arrow[dl] \\
M \arrow[rr, "f"] & & N &
\end{tikzcd}$$

Theorem 3 (satisfaction of incremental NACs by residual match). *In an adhesive category* **C** *with initial pushouts, assume a GTS (P, π, Φ) with incremental NACs, and conditional transformations $s_i = (G_0 \overset{p_i,m_i}{\Longrightarrow} G_i)$ for $i = 1, 2$ as in the left of Fig. 4, such that there exists $j : L_2 \to D_1$ with $j; l_1^* = m_2$. Furthermore, let $(n_2 : L_2 \to N_2) \in \Phi(p_2)$ be a negative constraint of p_2, and let $L_2^0 N_2^0 L_2 N_2$ be an initial pushout over n_2. Then match $j; r_1^*$ satisfies n_2 if and only if there exists no monic $k : N_2^0 \to R_1$ such that $L_2^0 N_2^0 G_1 L_2$ is a pullback.*

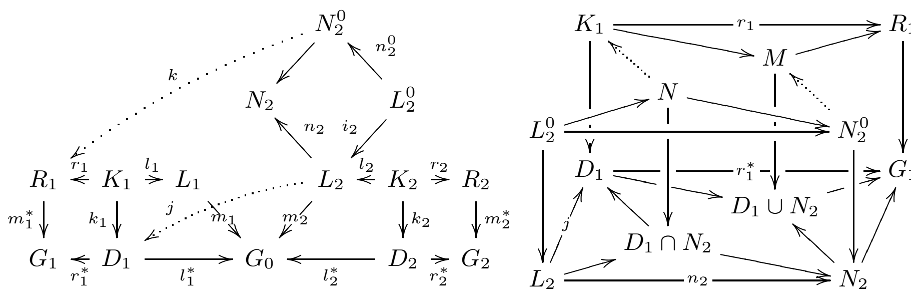


Fig. 4. Satisfaction of incremental NACs (left) and Proof of Thm. 3 (right)

Proof. "$\Rightarrow$": We show that, if k as above exists, then match $j; r_1^*$ does not satisfy constraint n_2. Indeed, given k and j as in the left diagram of Fig. 4, since $L_2^0 N_2^0 L_2 N_2$ is a pushout there exists a morphism $i : N_2 \to G_1$ commuting with n_2 and $j; r_1^*$. Since **C** is adhesive, $\mathbf{C} \downarrow G_1$ has *effective unions* [15], that is, the coproduct of two objects (arrows of **C** with target G_1) is computed as the pushout over their pullback. Since $L_2^0 N_2^0 G_1 L_2$ is a pullback, i is the coproduct of $k; m_1^*$ and $j; r_1^*$ in $\mathbf{C} \downarrow G_1$, and thus i is monic. Therefore constraint n_2 is violated.

"$\Leftarrow$": We show that, if mono $N_2 \to G_1$ exists commuting with n_2 and $j; r_1^*$, we can find k as required such that $L_2^0 N_2^0 G_1 L_2$ is a pullback. The back and bottom faces of the diagram in the right of Fig. 4 represent the second pushout of s_1 and the non-satisfaction of the constraint by the match $j; r_1^*$. The initial pushout over n_2 is shown in the front of the diagram.

Let $D_1 \leftarrow D_1 \cap N_2 \to N_2$ be a pullback of $D_1 \to G_1 \leftarrow N_2$ and $D_1 \to D_1 \cup N_2 \leftarrow N_2$ a pushout over $D_1 \leftarrow D_1 \cap N_2 \to N_2$, with $L_2 \to D_1 \cap N_2$ and $D_1 \cup N_2 \to G_1$ given by universality. Both arrows are monic, the second one because $\mathbf{C} \downarrow G_1$ has effective unions. All other arrows in the front, bottom, and back faces are also monic, so in particular the front initial pushout is also a pullback.

We can thus decompose it into pullbacks $L_2^0 N L_2 (D_1 \cap N_2)$ and $N N_2^0 (D_1 \cap N_2) N_2$ and then infer by pushout-pullback decomposition [15] that both diagrams are pushouts, too. It can be shown, by a general result about decomposition of initial pushouts, that $N N_2^0 (D_1 \cap N_2) N_2$ is an initial pushout over arrow $D_1 \cap N_2 \to N_2$. In particular, $D_1 \cap N_2 \to N_2$ is not iso because otherwise we would find $N_2 \to D_1 \to G_0$, i.e., m_2 would not satisfy n_2, contradicting the existence of s_2.

Like the front face, also the pushout in the back decomposes into pushouts along $D_1 \to D_1 \cup N_2 \to G_1$, in particular $K_1 M D_1 (D_1 \cup N_2)$. It is easy to check that the composition of an initial pushout and a pushout is again initial if the second is all monic. Hence, since $N N_2^0 (D_1 \cap N_2) N_2$ is initial pushout over $D_1 \cap N_2 \to N_2$ and $(D_1 \cap N_2) N_2 D_1 (D_1 \cup N_2)$ is a pushout, this implies that $N N_2^0 D_1 (D_1 \cup N_2)$ is initial pushout over $D_1 \to D_1 \cup N_2$. Therefore there exists a pushout $N N_2^0 K_1 M$, providing us with the desired $N_2^0 \to M \to R_1$. Finally, $L_2^0 N_2^0 G_1 L_2$ is a pullback because $L_2^0 N_2^0 L_2 N_2$ is a pushout over monos and therefore a pullback also, and $N_2 \to G_1$ is mono. $\square$

5 Canonical Derivations with Incremental NACs

As recalled in the Introduction, Kreowski showed in [14] that *shift*-equivalence classes of *parallel derivations* of a GTS have unique representatives (up to iso), called canonical derivations. Parallel derivations are sequences of parallel transformations, defined as follows. Given a multiset of redexes $S = [(p_i, m_i)]_{i \in I}$ enabled in a graph G (see Def. 2), a *parallel transformation* $G \overset{\Sigma S}{\Longrightarrow} H$ is defined as a transformation $G \overset{p,m}{\Longrightarrow} H$, where rule $p = L \xleftarrow{l} K \xrightarrow{r} R$ is obtained as the coproduct of rules $[p_i]_{i \in I}$, and match $m : L \to G$ (which is not mono, in general) is induced by the universal property of the coproduct forming L. The *shift* equivalence is the generalization of the *switch* equivalence on *sequential* derivations (see Sect. 3): two parallel derivations are shift equivalent if any of their serializations, which exist by the Parallelism theorem [6], are switch equivalent.

Existence of canonical representatives is shown by Kreowski by defining a *shift operation* on parallel derivations, that anticipates a redex from a parallel transformation to the previous one, if it is sequentially independent. This operation transforms a derivation into an equivalent one, and it is shown to be confluent and terminating, from which uniqueness of normal forms follows (up to iso, because so is the shift operation). The *canonical derivations* obtained this way feature maximal parallelism by applying each redex as early as possible.

We generalize here definitions and constructions by Kreowski to conditional step derivations, showing that canonical derivations exist if NACs are incremental. First we formalise sequential independence of redexes in steps and introduce the shift operation.

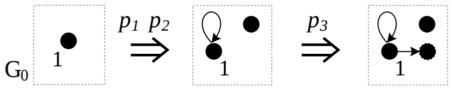
Definition 5 (sequential independence of redexes). *Given two consecutive safe steps $\sigma = (G_0 \overset{S_1}{\Longrightarrow} G_1 \overset{S_2}{\Longrightarrow} G_2)$ in a (conditional) GTS, redexes $s_1 \in S_1$ and $s_2 \in S_2$ are* sequentially independent *if for a serialisation* $(G_0 \overset{S_1^-}{\Longrightarrow} G_1' \overset{s_1'}{\Longrightarrow}$

$G_1 \overset{s_2}{\Longrightarrow} G_2' \overset{S_2^-}{\Longrightarrow} G_2$) *of* σ*, transformations* $G_1' \overset{s_1'}{\Longrightarrow} G_1 \overset{s_2}{\Longrightarrow} G_2'$ *are independent. Steps* $G_0 \overset{S_1}{\Longrightarrow} G_1 \overset{S_2}{\Longrightarrow} G_2$ *are* sequentially independent *if for all* $s_1 \in S_1$ *and* $s_2 \in S_2$*,* s_1 *and* s_2 *are.*

Definition 6 (shift operation). *Given two consecutive safe steps* $\sigma = (G_0 \overset{S_1}{\Longrightarrow} G_1 \overset{S_2}{\Longrightarrow} G_2)$ *in* (P, π, Φ)*, let* $s_2 = (p_2, m_2) \in S_2$ *be such that for all* $s_1 = (p_1, m_1) \in S_1$*,* s_1 *and* s_2 *are sequentially independent, and let* $G_1 \overset{s_2}{\Longrightarrow} G_2' \overset{S_2'}{\Longrightarrow} G_2$ *be a serialization of* $G_1 \overset{S_2}{\Longrightarrow} G_2$*. Furthermore, let* $\sigma' = (G_0 \overset{S_1'}{\Longrightarrow} G_2' \overset{S_2'}{\Longrightarrow} G_2)$ *be the step derivation where* S_1' *is the parallelisation of* S_1 *and* s_2*. Then we say that* σ' *is a* shift *of* σ*, denoted* $\sigma \overset{sh(s_2)}{\longrightarrow} \sigma'$*. The shift relation over step derivations in a (conditional) GTS is defined as the smallest relation including* $\overset{sh(\_)}{\longrightarrow}$ *and such that* $\sigma \overset{sh(s)}{\longrightarrow} \sigma'$ *implies* $\sigma_1; \sigma; \sigma_2 \overset{sh(s)}{\longrightarrow} \sigma_1; \sigma'; \sigma_2$ *for any derivations* σ_1, σ_2*. Shift equivalence* $\equiv_{sh}$ *is the smallest equivalence containing* $\overset{sh(\_)}{\longrightarrow}$*.*

It is easy to see that if $\sigma \overset{sh(s_2)}{\longrightarrow} \sigma'$ according to the above definition, then in σ' all redexes satisfy their NACs and that redexes are pairwise parallel independent in both steps. Therefore the result is a legal step derivation. It remains to analyse under which conditions it is safe. The following example provides some intuition.

Example 4 (unsafe shift). Using again the rules of Fig. 1, we can build the following safe step derivation. Notice that p_3 is independent of p_1 and p_2 in isolation, but if it is shifted using the construction of Def. 6 this would result in the unsafe step of Ex. 3.



As expected, this problem can be avoided by restricting to incremental NACs, as shown in the theorem below. First we need the following technical result.

Proposition 2 (parallelisation). *Given two consecutive safe steps* $G_0 \overset{S_1}{\Longrightarrow} G_1 \overset{s_2}{\Longrightarrow} G_2$ *using incremental NACs only, they are sequential independent if and only if there is a parallelisation* $G_0 \overset{S_1[s_2']}{\Longrightarrow} G_2$ *that is safe (see Def. 3).*

Theorem 4 (safe and confluent shift with incremental NACs). *Let* σ *be a safe step derivation over a GTS with incremental NACs only, and let* $\sigma \overset{sh(s)}{\longrightarrow} \sigma'$*. Then* σ' *is a safe step derivation. Moreover, shift is confluent, that is, given* $\sigma \overset{sh(s_1)}{\longrightarrow} \sigma_1$ *and* $\sigma \overset{sh(s_2)}{\longrightarrow} \sigma_2$ *we also have* $\sigma_1 \overset{sh(s_2'')}{\longrightarrow} \sigma_3$ *and* $\sigma_2 \overset{sh(s_1'')}{\longrightarrow} \sigma_3$ *where* s_1'', s_2'' *are the residuals of* s_1 *and* s_2*, respectively.*

Proof. The first point follows easily from Prop. 2. To show confluence of shift, assume a two step derivation $\sigma = (G_0 \overset{S_1}{\Longrightarrow} G_1 \overset{S_2}{\Longrightarrow} G_2)$ and let $s_1, s_2 \in S_2$ such that

$\sigma \overset{sh(s_1)}{\longrightarrow} \sigma_1$ and $\sigma \overset{sh(s_2)}{\longrightarrow} \sigma_2$ exist. Therefore, σ is safe and s_1, s_2 are independent of all $s \in S_1$. Then, let $\sigma_1 = (G_0 \overset{S_1 s_1'}{\Longrightarrow} G_1^1 \overset{S_2^1}{\Longrightarrow} G_2)$ and $\sigma_2 = (G_0 \overset{S_1 s_2'}{\Longrightarrow} G_1^2 \overset{S_2^2}{\Longrightarrow} G_2)$.

By Prop. 2 all steps are safe, so in order to have shifts $\sigma_1 \overset{sh(s_2'')}{\longrightarrow} \sigma_3$ and $\sigma_2 \overset{sh(s_1'')}{\longrightarrow} \sigma_3$ we have to show that $s_2'' \in S_2^1$ (resp. $s_1'' \in S_2^2$) is independent of all transformations in $S_1 s_1'$ (resp. in $S_1 s_2'$). Then, by confluence of shift on the underlying step derivations, which can be proved essentially as in [14] for parallel derivations, we know that $sh(s_1'')$ and $sh(s_2'')$ lead to isomorphic results.

This is similar, and in fact a consequence, of the stability of independence under switching in item 1 of Thm. 1. More generally we show that the shift operation does not affect the independence of the remaining transformations in S_1 and S_2, i.e., for any $s \in S_1$ and $t \in S_2 - s_1$ with residual $t' \in S_2'$, s, t are independent iff s, t' are. Consider the cube in Thm. 1, and let s be t_1 and t be t_2^1, with t_3^1 being the transformation s_1 anticipated by the shift to $t_3 = s_1'$ and t_2^3 playing the role of t'. By item 1 of Thm. 1, t_1, t_2^1 are independent iff t_1^2, t_2^3 are, while the latter is equivalent to independence of s, t' by Def. 5. □

Thus the shift relation on safe step derivations with incremental NACs is confluent. Since it is also terminating, as it can be proved along the lines of [14], it has normal forms characterising shift equivalence, called *canonical derivations.* That is, two safe step derivations with incremental NACs are shift equivalent if and only if their canonical derivations are isomorphic.

6 Conclusion

We have investigated the computational model of safe step derivations for conditional graph transformation systems. We have proved that, unlike the general case, if NACs are restricted to be incremental then the safety of a step, which ensures its serializability, follows from the pairwise independence of the component match-rule pairs. In turn, Thm. 3 showed that parallel independence of conditional transformations can be checked efficiently if NACs are incremental: we think that this result can be exploited for the efficient computation of critical pairs. Finally Thm. 4 showed the existence of canonical representatives for the shift-equivalence classes of step derivations with incremental NACs. This result holds for any system satisfying Thm. 1: the stability of independence under switching provides a semantic characterisation for well-behaved concurrency, for which incremental NACs are a sufficient structural condition on rules.

There are several possible developments of the present work, that we intend to explore in the future. First, we aim at investigating graph processes with incremental NACs as a more compact representation of shift equivalence classes of derivations, and how far this can be generalised to arbitrary NACs. Next we intend to look for weaker structural conditions able to ensure that a GTS, even if including non-incremental NACs, still satisfies Thm. 1 (and therefore has canonical derivations). Finally, the extension of the proposed step-based computational model to non-monic matches and non-monic NACs looks like an interesting challenge to face.

References

1. Baldan, P., Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Löwe, M.: Concurrent Semantics of Algebraic Graph Transformations. In: Rozenberg, G. (ed.) The Handbook of Graph Grammars and Computing by Graph Transformations, Concurrency, Parallelism and Distribution, vol. 3, pp. 107–188. World Scientific (1999)
2. Baldan, P., Corradini, A., Heindel, T., König, B., Sobociński, P.: Unfolding grammars in adhesive categories. In: Kurz, A., Lenisa, M., Tarlecki, A. (eds.) CALCO 2009. LNCS, vol. 5728, pp. 350–366. Springer, Heidelberg (2009)
3. Baldan, P., Corradini, A., Montanari, U., Ribeiro, L.: Unfolding semantics of graph transformation. Inf. Comput. 205(5), 733–782 (2007)
4. Corradini, A., Heckel, R., Hermann, F., Gottmann, S., Nachtigall, N.: Transformation systems with incremental negative application conditions. In: Martí-Oliet, N., Palomino, M. (eds.) WADT 2012. LNCS, vol. 7841, pp. 127–142. Springer, Heidelberg (2013)
5. Corradini, A., Montanari, U., Rossi, F.: Graph processes. Fundamenta Informaticae 26(3/4), 241–265 (1996)
6. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs in Theor. Comp. Science. Springer (2006)
7. Ehrig, H.: Introduction to the algebraic theory of graph grammars (a survey). In: Claus, V., Ehrig, H., Rozenberg, G. (eds.) Graph Grammars 1978. LNCS, vol. 73, pp. 1–69. Springer, Heidelberg (1978)
8. Golas, U., Ehrig, H., Habel, A.: Multi-amalgamation in adhesive categories. In: Ehrig, H., Rensink, A., Rozenberg, G., Schürr, A. (eds.) ICGT 2010. LNCS, vol. 6372, pp. 346–361. Springer, Heidelberg (2010)
9. Habel, A., Heckel, R., Taentzer, G.: Graph Grammars with Negative Application Conditions. Fundamenta Informaticae 26(3,4), 287–313 (1996)
10. Habel, A., Müller, J., Plump, D.: Double-pushout graph transformation revisited. Mathematical Structures in Computer Science 11(5), 637–688 (2001)
11. Heckel, R.: DPO Transformation with Open Maps. In: Ehrig, H., Engels, G., Kreowski, H.-J., Rozenberg, G. (eds.) ICGT 2012. LNCS, vol. 7562, pp. 203–217. Springer, Heidelberg (2012)
12. Hermann, F.: Permutation equivalence of DPO derivations with negative application conditions based on Subobject Transformation Systems. ECEASST 16 (2008)
13. Hermann, F., Gottmann, S., Nachtigall, N., Braatz, B., Morelli, G., Pierre, A., Engel, T.: On an automated translation of satellite procedures using triple graph grammars. In: Duddy, K., Kappel, G. (eds.) ICMT 2013. LNCS, vol. 7909, pp. 50–51. Springer, Heidelberg (2013)
14. Kreowski, H.J.: Is parallelism already concurrency? Part 1: Derivations in graph grammars. In: Ehrig, H., Nagl, M., Rosenfeld, A., Rozenberg, G. (eds.) Graph Grammars 1986. LNCS, vol. 291, pp. 343–360. Springer, Heidelberg (1987)
15. Lack, S., Sobocinski, P.: Adhesive and quasiadhesive categories. ITA 39(3) (2005)
16. Lambers, L., Ehrig, H., Orejas, F., Prange, U.: Parallelism and Concurrency in Adhesive High-Level Replacement Systems with Negative Application Conditions. In: Proceedings of the ACCAT workshop at ETAPS 2007. ENTCS, vol. 203 / 6, pp. 43–66. Elsevier (2008)
17. Lambers, L.: Certifying Rule-Based Models using Graph Transformation. Ph.D. thesis. Technische Universität, Berlin (2009)
18. Taentzer, G.: Parallel high-level replacement systems. Theor. Comput. Sci. 186(1-2), 43–81 (1997)