Under consideration for publication in Math. Struct. in Comp. Science

Analysis of Permutation Equivalence in $\mathcal{M}$-adhesive Transformation Systems with Negative Application Conditions

FRANK HERMANN[1,2], ANDREA CORRADINI[3],

AND HARTMUT EHRIG[1]

[1] [frank,ehrig]@cs.tu-berlin.de, *Institut für Softwaretechnik und Theoretische Informatik, Technische Universität Berlin, Germany*
[2] *Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg*
[3] andrea(at)di.unipi.it, *Dipartimento di Informatica, Università di Pisa, Italy*

Received February 2011

$\mathcal{M}$-adhesive categories provide an abstract framework for a large variety of specification frameworks for modelling distributed and concurrent systems. They extend the well-known frameworks of adhesive and weak adhesive HLR categories and integrate high-level constructs like attribution as in the case of typed attributed graphs.
This article presents M-adhesive transformation systems including negative application conditions (NACs) for transformation rules, which are often used in applied scenarios. For this purpose, the classical notion of switch equivalence is generalised to the notion of permutation equivalence, because there are intuitively equivalent NAC-consistent transformation sequences which cannot be derived by switching consecutive NAC-independent transformation steps. Furthermore, this article presents a general construction of processes of $\mathcal{M}$-adhesive transformation systems based on subobject transformation systems and permutation equivalence. As main results we show that a constructed process of a transformation sequence specifies its equivalence class of permutation-equivalent transformation sequences. Moreover, we show how the analysis of this process can be reduced to the analysis of the reachability graph of a generated Place/Transition Petri net. This net encodes the dependencies among rule applications of the transformation sequence, including the inhibiting effects of the NACs.

Contents

1 Introduction 2
2 Transformation Systems and Permutation Equivalence 4
 2.1 $\mathcal{M}$-adhesive Categories and General Assumption 4
 2.2 $\mathcal{M}$-adhesive Transformation Systems with NACs 6
 2.3 Permutation Equivalence of Transformation Sequences 11
3 From Subobject Transformation Systems to Processes of $\mathcal{M}$-adhesive Transformation Systems 14

3.1 $\mathcal{M}$-Subobject Transformation Systems 14
3.2 Processes of $\mathcal{M}$-adhesive Transformation Systems 17
4 Analysis of Permutation Equivalence Based on Petri Nets 22
5 Related Work 27
6 Conclusion 28
References 29
Appendix A Category of Typed Attributed Graphs 30
Appendix B Proofs of Technical Results 31

1. Introduction

The notion of $\mathcal{M}$-adhesive transformation systems provides an abstract framework for transformation systems based on the double pushout (DPO) approach originally developed for graphs (Ehrig et al.1973) and extended to typed attributed graphs and a large variety of Petri nets based on the slightly more specific framework of weak adhesive transformation systems with suitable classes $\mathcal{M}$ of monomorphisms (Ehrig et al.2006). While several analysis techniques for the crucial properties of termination and local confluence have been provided for the general setting, this paper presents general techniques for the analysis of processes of such systems, i.e. of equivalence classes of executions differing only for the interleaving of the same transformation steps.

The main problem in this context is to analyse whether a sequence of transformation steps can be rearranged in order to generate all possible equivalent executions, or some specific and possibly better ones. If the system is modelled by a Petri net these questions can be fairly easily answered: processes (or occurrence nets) incorporate a notion of concurrency that can be exploited to rearrange the tasks, while still respecting causality. We are here considering models with two further dimensions, which considerably complicate the problem: first, we work in the general setting of $\mathcal{M}$-adhesive categories where we can model systems with an evolving topology, such as graph transformation systems, in contrast to systems with a static structure like classical Petri nets. Second, we take into account Negative Application Conditions (NACs) that are used to ensure the "absence" of forbidden structures when executing a transformation step. It is well-known that NACs significantly improve the specification formalisms based on transformation rules leading to more compact and concise models as well as increased usability, and they are widely used in non-trivial applications.

In the case of systems with NACs, we generalise the classical notion of switch equivalence to the notion of permutation equivalence, because there are intuitively equivalent transformation sequences which cannot be derived by switching independent neighbouring steps. As defined in (Hermann2009), two transformation sequences are called permutation-equivalent, if they respect the NACs and disregarding the NACs they are switch-equivalent. The notion of permutation equivalence on transformation sequences with NACs is coarser and more adequate than the classical switch equivalence based on the local Church-Rosser theorem in the DPO approach including NACs (Lambers2009).

For the sake of generality, and also motivated by our case study based on typed attributed graph transformation systems, we consider transformation sequences with general (i.e. possibly non-monic) matches, and we introduce a new kind of NACs called *NAC-schemata*, which allows us to reduce the number of classical NACs significantly. Interestingly, we show in our first main result (Thm. 1) that permutation equivalence of transformation sequences using general matches and NAC-schemata can be reduced to permutation equivalence of sequences using only matches in $\mathcal{M}$ (called $\mathcal{M}$-matches) and classical NACs. This allows us to reduce also the analysis of permutation equivalence to the case of transformation sequences with $\mathcal{M}$-matches and classical NACs.

The main practical analysis problem for permutation equivalence is to construct for a given transformation sequence the set of all permutation-equivalent transformation sequences. The brute-force method would be to construct all switch-equivalent sequences disregarding NACs and then to filter out the NAC-consistent ones. However, our case study shows that this brute-force approach is in general very inefficient. In this paper, we show how to analyse permutation equivalence using subobject transformation systems (STSs) and Petri nets leading to much more efficient solutions.

For this purpose, we exploit the notion of process of a transformation sequence, which consists of an STS with an embedding into the original transformation system: this construction is based on and generalises results in (Corradini et al.2008; Hermann2009) for STSs over adhesive transformation systems with NACs. Our second main result (Thm. 2) shows that the constructed process of a given transformation sequence exactly characterizes the equivalence class of permutation-equivalent transformation sequences.

For improving the efficiency of the analysis of permutation equivalence we provide the construction of a dependency net for a given process of a transformation sequence with NACs. This net is given by a standard P/T Petri net which includes a complete account of the causal dependencies and NAC-dependencies among transformation steps. Our further main results (Thms. 3 and 4) show that complete firing sequences of the dependency net are one-to-one with transformation sequences that are permutation-equivalent to the given one. This allows us to derive the complete set of permutation-equivalent transformation sequences from a simple analysis of a Petri net. Furthermore, the constructed P/T Petri net can be used to derive specific permutations without generating the complete set first.

Concepts and results of this paper generalize those presented in (Hermann et al.2010) for graph transformation to the more abstract and general framework of $\mathcal{M}$-adhesive transformation systems with general matches. Sec. 2 reviews $\mathcal{M}$-adhesive categories and presents the main concepts of transformation systems with NACs and of permutation equivalence. Thereafter, Sec. 3 introduces the framework of Subobject Transformation Systems (STSs) with NACs and the process construction for $\mathcal{M}$-adhesive transformation systems. The analysis of the process via the construction of the dependency net given by a Petri net is presented in Sec. 4. Thereafter, Secs. 5 and 6 discuss related work and provide a conclusion. Finally, App. A recalls the technical details of the $\mathcal{M}$-adhesive category of typed attributed graphs and App. B provides the proofs of some auxiliary facts, while the proofs of the main results in Thms. 1-4 are given in the main part of the paper.

2. Transformation Systems and Permutation Equivalence

Most definitions and results of the Double Pushout (DPO) approach to transformation systems have been generalized to *adhesive categories* (Lack and Sobocinski2005), (weak) adhesive HLR categories (Ehrig et al.2006), partial map adhesive categories (Heindel2010), and $\mathcal{M}$-adhesive categories (Ehrig et al.2010) being the most general among them. These frameworks require that pushouts along monos (or along a distinguished subclass of monos, called $\mathcal{M}$-morphisms) "behave well" with respect to pullbacks. Because of this, it is quite natural to present our contribution at this level of generality, by referring all definitions and results to an arbitrary but fixed $\mathcal{M}$-adhesive category $\mathbf{C}$.

In this section we review $\mathcal{M}$-adhesive categories together with some additional properties in Sec. 2.1, $\mathcal{M}$-adhesive transformation systems with Negative Application Conditions (NACs) in Sec. 2.2, and the notion of *permutation equivalence* on transformation sequences of such systems in Sec. 2.3. Most of the definitions are illustrated with a running case study based on typed attributed graph transformation systems.

2.1. *$\mathcal{M}$-adhesive Categories and General Assumption*

The abstract framework of $\mathcal{M}$-adhesive categories unifies several important modelling techniques for parallel and distributed systems. $\mathcal{M}$-adhesive categories are slightly more general than weak adhesive HLR categories (Ehrig et al.2006) and thus include different kinds of graphs and Petri nets.

Definition 2.1 ($\mathcal{M}$-adhesive category). A pair $(\mathbf{C}, \mathcal{M})$ consisting of a category $\mathbf{C}$ and a class of morphism $\mathcal{M}$ is called an *$\mathcal{M}$-adhesive category* if:

1 $\mathcal{M}$ is a class of monomorphisms of $\mathbf{C}$ closed under isomorphisms, composition, and decomposition ($g \circ f \in \mathcal{M}, g \in \mathcal{M} \Rightarrow f \in \mathcal{M}$).
2 $\mathbf{C}$ has pushouts and pullbacks along $\mathcal{M}$-morphisms, and $\mathcal{M}$-morphisms are closed under pushouts and pullbacks.
3 Pushouts in $\mathbf{C}$ along $\mathcal{M}$-morphisms are *"$\mathcal{M}$-Van Kampen" ($\mathcal{M}$-VK) squares*, i.e. for any commutative cube like (2) below where the bottom face (1) is a pushout along $m \in \mathcal{M}$, the back faces are pullbacks, and $b, c, d \in \mathcal{M}$, we have: The top face is a pushout if and only if the front faces are pullbacks.

$$
\begin{tikzcd}
 & A \arrow[ld, "f"'] \arrow[rd, hook, "m"] \arrow[dd, phantom, "(1)"] & \\
C \arrow[rd, "n"'] & & B \arrow[ld, "g"] \\
 & D &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & & A' \arrow[lld, "f'"'] \arrow[rrd, "m'"] \arrow[dd, dashed, "a"] & & \\
C' \arrow[rd, "n'"] \arrow[dd, "c"'] & & & & B' \arrow[llld, "g'"'] \arrow[dd, "b"] \\
 & D' \arrow[dd, "d"] & A \arrow[lld, dotted, "f"'] \arrow[rrd, dotted, "m"] \arrow[d, phantom, "(1)"] & & \\
C \arrow[rd, "n"'] & & {} & & B \arrow[llld, "g"] \\
 & D & & &
\end{tikzcd}
\quad (2)
$$

As mentioned above, starting from (Lack and Sobociński2004) adhesivity as been defined in several variants and sometimes in subtly different ways: For a recollection of such notions and comparisons among them the reader is referred to (Ehrig et al.2010).

Example 2.2 (The category of typed attributed graphs). The $\mathcal{M}$-adhesive category of our case study is the category of typed attributed graphs $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$

which is given by the slice category ($\mathbf{AGraph}\!\downarrow\!ATG, \mathcal{M}$) of directed attributed graphs over a type graph ATG. The distinguished class $\mathcal{M}$ contains all monomorphisms that are isomorphisms on the data part. According to (Ehrig et al.2006), an attributed graph consists of an extended directed graph for the structural part, called E-graph, together with an algebra for the specification of the carrier sets of the value nodes (see App. A). The objects of ($\mathbf{AGraphs}_{ATG}, \mathcal{M}$) are attributed graphs with a *typing morphism* to a fixed attributed graph ATG (called the *type graph*), and as arrows all attributed graph morphisms preserving the typing. It follows from the results in (Ehrig et al.2006) that this category is $\mathcal{M}$-adhesive.

Several $\mathcal{M}$-adhesive categories and results for $\mathcal{M}$-adhesive transformation systems require the existence of epi-mono factorizations or more general $\mathcal{E}$-$\mathcal{M}$ pair factorizations. Similarly, we require in this paper that the underlying $\mathcal{M}$-adhesive categories provide extremal $\mathcal{E}$-$\mathcal{M}$ factorizations. This allows us to analyse transformation systems with general matches, i.e. matches that are possibly not in $\mathcal{M}$.

Definition 2.3 (Extremal $\mathcal{E}$-$\mathcal{M}$ factorization). Given an $\mathcal{M}$-adhesive category $(\mathbf{C}, \mathcal{M})$, the class $\mathcal{E}$ of all extremal morphisms with respect to $\mathcal{M}$ is defined by $\mathcal{E} := \{e \in \mathbf{C} \mid \text{for all } m, f \text{ in } \mathbf{C} \text{ with } m \circ f = e\colon m \in \mathcal{M} \text{ implies } m \text{ isomorphism}\}$. For a morphism $f : A \to B$ in $\mathbf{C}$ an extremal $\mathcal{E}$-$\mathcal{M}$ factorization of f is given by an object $\overline{B}$ and morphisms $(e : A \twoheadrightarrow \overline{B}) \in \mathcal{E}$ and $(m : \overline{B} \rightarrowtail B) \in \mathcal{M}$, such that $m \circ e = f$.

Remark 2.4 (Uniqueness of Extremal $\mathcal{E}$-$\mathcal{M}$ Factorizations). As shown by Prop. 3 in (Braatz et al.2010), extremal $\mathcal{E}$-$\mathcal{M}$ factorizations are unique up to isomorphism.

In the case of finitary $\mathcal{M}$-adhesive categories, the extremal factorization can be performed by constructing all factorizations and stepwise pullbacks of them as shown by Prop. 4 in (Braatz et al.2010). An $\mathcal{M}$-adhesive category is finitary, if each object A is finite in the sense that there are finitely many $\mathcal{M}$-subobjects $[b : B \rightarrowtail A]$, i.e. finitely many $\mathcal{M}$-morphisms up to isomorphism with target A. A typed attributed graph $AG = ((G, D), t)$ in ($\mathbf{AGraphs}_{ATG}, \mathcal{M}$) with typing $t : (G, D) \to ATG$ is finite if the graph part of G, i.e., all vertex and edge sets except the set V_D of data vertices generated from D, is finite, while the attributed type graph ATG or the data type part D may be infinite, because $\mathcal{M}$-morphisms are isomorphisms on the data type part. The restriction of ($\mathbf{AGraphs}_{ATG}, \mathcal{M}$) to finite objects forms a finitary category.

Example 2.5 (Extremal $\mathcal{E}$-$\mathcal{M}$ factorization). Given a morphism $f : G \to H$ in the finitary category of typed attributed graphs ($\mathbf{AGraphs}_{ATG}, \mathcal{M}$), the factorization $f = m \circ e$ is constructed by performing the epi-mono-factorization on the graph part, i.e. on all nodes and edges except the data value nodes V_D, while for the data part f_D we derive $e_D : A_G \to A_H$ with $e_D(x) = f_D(x)$ and $m_D = id : A_H \to A_H$.

In order to efficiently analyse permutation equivalence in Secs. 3 and 4, we require effective unions for the underlying category $\mathbf{C}$, i.e. that the join of two subobjects can be constructed as pushout in $\mathbf{C}$.

Definition 2.6 (Effective Unions). Given an $\mathcal{M}$-adhesive category $(\mathbf{C}, \mathcal{M})$ and two $\mathcal{M}$-morphisms $b : B \to Z$ and $c : C \to Z$, let (f, g) be obtained as the pullback of (b, c) as depicted, and (f', g') be obtained as the pushout (1) of (f, g), with induced mediating morphism $d : D \to Z$. Pushout (1) is called *effective*, if $d \in \mathcal{M}$. The $\mathcal{M}$-adhesive category $(\mathbf{C}, \mathcal{M})$ has *effective unions*, if for all pairs b, c of $\mathcal{M}$-morphisms pushout (1) is effective.

$$\begin{tikzcd}
 & B \arrow[dr, tail, "g'"] \arrow[drr, bend left, tail, "b"] & & \\
A \arrow[ur, tail, "f"] \arrow[dr, tail, "g"'] & (1) & D \arrow[r, "d"] & Z \\
 & C \arrow[ur, tail, "f'"'] \arrow[urr, bend right, tail, "c"'] & &
\end{tikzcd}$$

Remark 2.7 (Effective Unions in $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$). The $\mathcal{M}$-adhesive category $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$ has effective unions, because by commutativity of the diagram in Def. 2.6, the morphism d is an isomorphism on the data part.

General Assumption: In order to analyse transformation systems based on an $\mathcal{M}$-adhesive category $(\mathbf{C}, \mathcal{M})$ we base all our further constructions in this paper on the general assumption that $(\mathbf{C}, \mathcal{M})$ provides an extremal $\mathcal{E}$-$\mathcal{M}$ factorization (Def. 2.3) and effective unions (Def. 2.6).

2.2. *$\mathcal{M}$-adhesive Transformation Systems with NACs*

In the first part of this section we review basic notions of transformation steps and transformation systems. A transformation rule specifies how a given object G can be transformed into a resulting object H. Given a match $m : L \to G$ of the left hand side of rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ into the object G such that p is applicable, the resulting object H is intuitively derived by removing the parts that are in L but not in K and by adding those that are in R but not in K. Negative application conditions (NACs) extend a transformation rule in order to restrict the applicability of the rule by specifying forbidden contexts in which the rule shall not be applied. Intuitively, a match $m\colon L \to G$ satisfies a NAC $n : L \to N$ for a rule p if the image of the left hand side L in G cannot be extended to an image of the "forbidden context" N.

In the present paper we do not consider nested application conditions (Habel and Pennemann2009), but we plan to extend our results to this more general kind of application conditions.

Definition 2.8 (NAC-consistent Transformation Steps for $\mathcal{M}$-matches). Given an $\mathcal{M}$-adhesive category $(\mathbf{C}, \mathcal{M})$, a *(transformation) rule* $p = (L \xleftarrow{l} K \xrightarrow{r} R)$, also called *production*, is a pair of $\mathcal{M}$-morphisms with the same source in $|\mathbf{C}|$. A *Negative Application Condition (NAC)* for a rule p is an $\mathcal{M}$-morphism $n : L \rightarrowtail N$, having the left-hand side of p as source and a *rule with NACs* is a pair $(p, \mathbf{N})$ where p is a rule and $\mathbf{N}$ is a set of NACs for p. Given an $\mathcal{M}$-morphism $m : L \rightarrowtail G$ into an object $G \in \mathbf{C}$, called *match*, m *satisfies the NAC $n : L_p \rightarrowtail N$ for p*, written $m \models n$, if there is no $\mathcal{M}$-morphism $q : N \rightarrowtail G$ such that $q \circ n = m$. We say that there is a *NAC-consistent transformation step* from an object G to H using a rule with NACs $(p, \mathbf{N})$ and a match $m : L_p \to G$, if (a) there are two pushouts (1) and (2) in $\mathbf{C}$, as depicted; and (b) $m \models n$ for each NAC $(n : L_p \rightarrowtail N) \in \mathbf{N}$.

$$\begin{tikzcd}
N \arrow[drr, tail, "q"'] & L \arrow[l, tail, "n"'] \arrow[d, tail, "m"] \arrow[r, phantom, "(1)"] & K \arrow[l, tail, "l"'] \arrow[r, tail, "r"] \arrow[d] \arrow[r, phantom, "(2)"] & R \arrow[d] \\
 & G & D \arrow[l] \arrow[r] & H
\end{tikzcd}$$

If condition (a) above is satisfied (and (b) possibly not, thus NACs are ignored) we say that there is a *transformation step disregarding NACs* from G to H. In both cases we write $G \xRightarrow{p,m} H$.

The last definition considers transformation steps for $\mathcal{M}$-matches only, but as we will discuss now this is too restrictive for transformations in our sample category of typed attributed graphs, and therefore in $\mathcal{M}$-adhesive categories in general.

Remark 2.9 (Discussion on matches and NACs in $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$). Requiring that a match $m : L \to G$ is in $\mathcal{M}$ implies that the data part of L is isomorphic to that of G. But this is much too restrictive because usually (see e.g. Ex. 2.12) the data algebra of L is given by a term algebra with variables $T_{OP}(X)$, while the data algebra of G is an arbitrary OP-algebra: in this situation the match m is determined, on the data part, by an assignment $ass : X \to A_G$, and it might be neither injective (e.g. two variables could be mapped to the same element of A_G) nor surjective.

Therefore in this general setting we have to consider transformation steps with respect to arbitrary matches. But this requires to revisit the basic definitions of NACs and their satisfaction. Indeed, if match $m : L \to G$ does not belong to $\mathcal{M}$, from Def. 2.8 it follows that m satisfies trivially n for any NAC $n : L \rightarrowtail N$: in fact, $n \in \mathcal{M}$ by definition, and if there were a $q \in \mathcal{M}$ such that $q \circ n = m$ then $m \in \mathcal{M}$ as well, leading to a contradiction.

For a meaningful notion of NAC satisfaction in presence of arbitrary matches several options are possible. Firstly, one may drop the requirement on q being in $\mathcal{M}$, saying that $m \models n$ if there is no morphism $q : N \to G$ such that $q \circ n = m$. As discussed in (Habel et al.1996) for the case of graph transformation, this notion of satisfaction has serious limitations in the expressive power, because it cannot express natural constraints like those involving cardinality (e.g., *"there must be at least two A-labelled nodes in G"*) or injectivity (e.g., *"the match cannot identify two given nodes of L"*); thus we prefer to avoid this solution.

Alternatively, one may drop the requirement that NAC $n : L \to N$ has to be in $\mathcal{M}$, still requiring any $q : N \to G$ being in $\mathcal{M}$. This is indeed the approach taken for example in (Habel et al.1996), but we don't consider it very satisfactory because it can lead to a combinatorial explosion of the number of NACs. In fact, suppose for example that L is a graph consisting of three B-labeled nodes, and that we want to forbid matches from L to any graph G which contains an additional node labelled with A; thus node A is a "forbidden context". It is easy to see that we need five distinct NACs, one for each possible different way of identifying subsets of the nodes of L with a match. Similarly, consider again the category of typed attributed graphs, a match $(m : L \to G) \notin \mathcal{M}$ and a NAC $(n : L \to N) \notin \mathcal{M}$. If the data algebra A_N of N is not isomorphic to the data algebra A_G of G there cannot exist any $q : N \to G$ in $\mathcal{M}$ making the triangle commute and thus $m \models n$ trivially holds. This means that we need at least one different NAC for each distinct algebra (up to isomorphism) that could be the data algebra of an attributed graph to which the rule should not be applicable.

Motivated by this discussion, we introduce now *NAC-schemata*, a new notion of NACs and NAC-consistency inspired by (Kastenberg et al.2006), that at the same time is meaningful for general matches and avoids the combinatorial explosion in the number of NACs. A NAC-schema is simply an $\mathcal{M}$-morphism $n : L \rightarrowtail N$, but NAC-satisfaction does not require the absence of an $\mathcal{M}$-morphism $q : N \rightarrowtail G$, but of an $\mathcal{M}$-morphism $q : N' \rightarrowtail G$ with N' being obtained from N, intuitively, by performing the same identifications as in the match $f : L \to G$. This condition is formalized by a pushout over an extremal $\mathcal{E}$-$\mathcal{M}$-factorization $L \xrightarrow{e} L' \xrightarrow{m} G$ of the match f (see Def. 2.3).

Definition 2.10 (NAC-schemata and Satisfaction). Let $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ be a rule, a *NAC-schema* for p is an $\mathcal{M}$-morphism $n : L \rightarrowtail N$. Let $f : L \to G$ be a general match of p, $f = m \circ e$ be its extremal $\mathcal{E}$-$\mathcal{M}$-factorization and diagram (1) be constructed as pushout. Then f satisfies the NAC-schema $n : L \rightarrowtail N$, written $f \models n$, if there is no $q \in \mathcal{M}$ with $q \circ n' = m$. In this case, the match f is called NAC-consistent. If $p' = (p, \mathbf{N})$ is a rule with a set of NAC-schemata $\mathbf{N}$, a match satisfies $\mathbf{N}$ if it satisfies all $n \in \mathbf{N}$.

$$\begin{tikzcd} N \arrow[d] & L \arrow[l, tail, "n"'] \arrow[d, two heads, "e"] \arrow[l, phantom, shift left=0, ""] \arrow[dd, bend left=60, "f"] \arrow[dl, phantom, "(1)"] \\ N' \arrow[dr, tail, bend right, "q"'] & L' \arrow[l, tail, "n'"] \arrow[d, tail, "m"] \\ & G \end{tikzcd}$$

It is worth noting that if match $f : L \to G$ is an $\mathcal{M}$-morphism, then satisfaction of a NAC-schema $n : L \to N$ coincides with classical satisfaction, because the factorization is trivially $f = f \circ id$.

A set of named transformation rules forms a transformation system and the naming is specified by a mapping $\pi : P \to \mathit{RULES}(\mathbf{C}, \mathcal{M})$ from the set of rule names P to the set of rules in an $\mathcal{M}$-adhesive category $(\mathbf{C}, \mathcal{M})$.

Definition 2.11 ($\mathcal{M}$-adhesive Transformation System). An *$\mathcal{M}$-adhesive transformation system (TS) over* $(\mathbf{C}, \mathcal{M})$ *for general matches* is a pair $TS = (P, \pi)$ where P is a set of rule names, and π maps each name $p \in P$ to a rule $\pi(p) = ((L \xleftarrow{l} K \xrightarrow{r} R), \mathbf{N}_S)$ with NAC-schemata $\mathbf{N}_S$. A *NAC-consistent transformation sequence* of TS is a sequence $G_0 \xRightarrow{p_1, m_1} G_1 \cdots \xRightarrow{p_n, m_n} G_n$, where $p_1, \ldots, p_n \in P$ and $d_i = G_{i-1} \xRightarrow{\pi(p_i), m_i} G_i$ is a transformation step with NAC-consistent match (see Def. 2.10) for $i \in 1, \ldots, n$. Sometimes, we denote a transformation sequence as $d = (d_1; \ldots; d_n)$, where each d_i denotes a single transformation step.

An *$\mathcal{M}$-adhesive transformation system (TS) over* $(\mathbf{C}, \mathcal{M})$ *for $\mathcal{M}$-matches* is defined as above, where, however, the set $\mathbf{N}_S$ of NAC-schemata is replaced by a set of NACs $\mathbf{N}$ with NAC-consistency according to Def. 2.8.

Example 2.12 (Typed Attributed Graph Transformation System). The $\mathcal{M}$-adhesive transformation system for general matches of our case study is the typed attributed graph transformation system *GTS* in Fig. 1. The *type graph ATG* specifies persons and tasks: a task is active if it has a ":started" loop, and it can be assigned to a person with a ":worksOn" edge. Moreover, the attribute "accessLevel" specifies the required access level of tasks and the allowed maximal access level of persons. Rule "startTask" is used to start a task, where the access level of the task can be at most equal to the access level of the considered person and the NAC-schema ensures that the task is not started already. Rules "stopTask" and "finishTask" removes the assignment of a person, where "finishTask" additionally deletes the marker ":started" to specify that

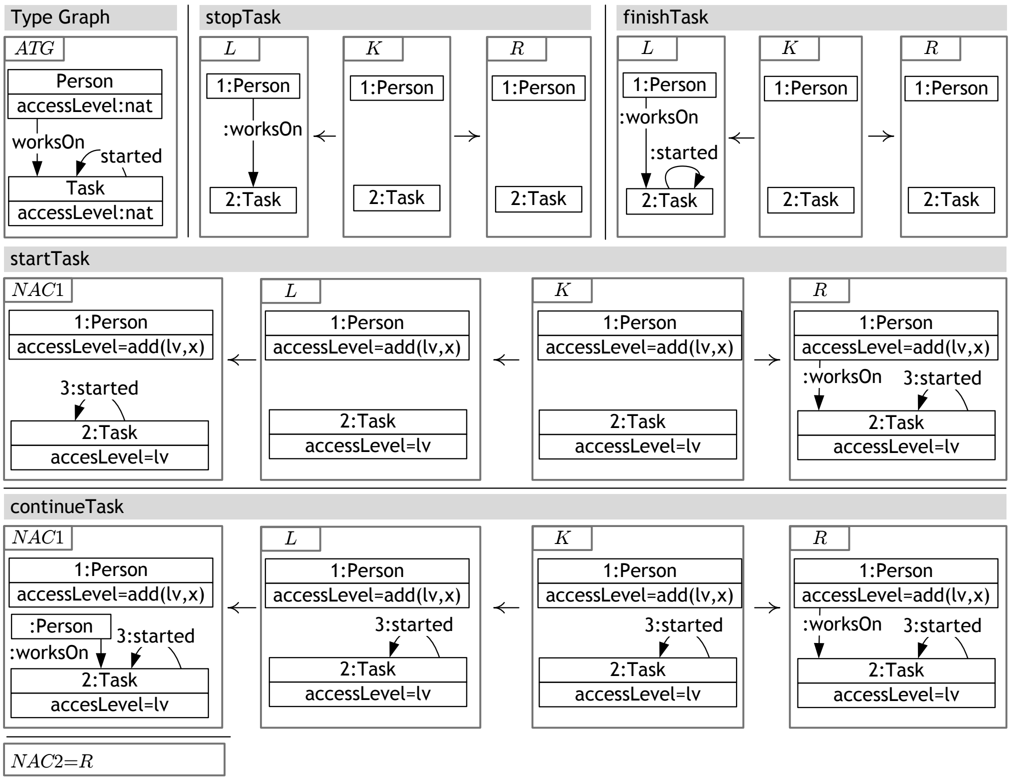

Fig. 1. Typed attributed graph transformation system *GTS*

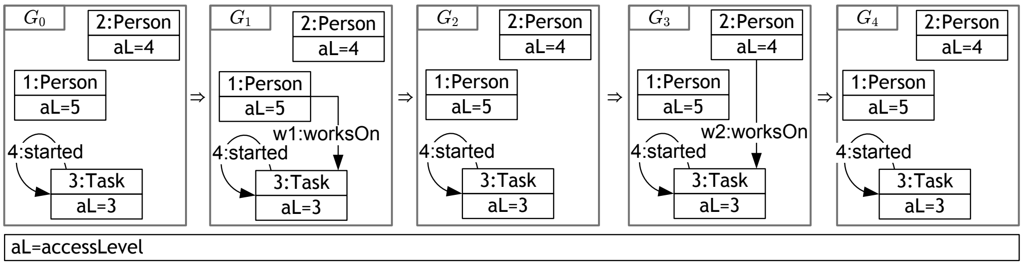

Fig. 2. Transformation sequence d of GTS

the task has been completed. Finally, rule "continueTask" assigns an already started task to a person. This rule contains two NAC-schemata which forbid the assignment of persons to already assigned tasks – either if another person is already assigned to that task ("NAC1") or the person itself is already assigned ("NAC2"). Fig. 2 shows a NAC-consistent transformation sequence $d = (G_0 \xRightarrow{continueTask,f_1} G_1 \xRightarrow{stopTask,f_2} G_2 \xRightarrow{continueTask,f_3} G_3 \xRightarrow{stopTask,f_4} G_4)$ of GTS. The first graph of the transformation sequence contains exactly one task which is first assigned to node "1:Person", and then, after being stopped, to node "2:Person". The NAC-schemata of rule "*continueTask*" are checked at graphs G_0 and G_2. The constructed pushouts according to Def. 2.10 yield instantiated NACs $n' : L \rightarrowtail N'$ with N' containing an edge of type *worksOn*. Since G_0 and G_2 do not contain an edge of this type there is no embedding q from N' into these graphs, such that the NAC-schemata are satisfied by the matches. Therefore, the transformation sequence is NAC-consistent, because the remaining steps do not involve NACs. Note that the use of NAC-schemata and general matches is essential for our case study. If we would use $\mathcal{M}$-matches respectively classical NACs we would have to provide specific rules and NACs for each possible variable assignment concerning persons with different actual access levels (see also Rem. 2.9).

While general matches for $\mathcal{M}$-adhesive transformation systems leads to extended concepts for NACs and NAC satisfaction, we now show that we can reduce the analysis of a concrete given transformation sequence to the case of $\mathcal{M}$-matches by instantiating the rules and transformation diagrams along the given matches. Note in particular, that for transformation steps along $\mathcal{M}$-matches, the instantiated transformation steps coincide with the given ones.

Definition 2.13 (Instantiated Rules and Transformation Sequences). Let $G \xRightarrow{p,f} H$ be a NAC-consistent transformation step via a rule $p = ((L \leftarrowtail K \rightarrowtail R), \mathbf{N}_S)$ with NAC-schemata $\mathbf{N}_S$. Let $f = m \circ e$ be the extremal $\mathcal{E}$-$\mathcal{M}$ factorization of match f. The *instantiated transformation step* is given by $G \xRightarrow{p',m} H$ with *instantiated rule* p' *derived via* e and constructed as follows.

$$
\begin{tikzcd}
N \arrow[d] & L \arrow[l, tail, "n"'] \arrow[d, "e"'] \arrow[dd, bend left, "f"] \arrow[dr, phantom, "(3)"] & K \arrow[l, tail] \arrow[r, tail] \arrow[d, "k_e"'] \arrow[dd, bend left, "k"] \arrow[dr, phantom, "(4)"] & R \arrow[d, "e^*"'] \arrow[dd, bend left, "f^*"] \\
N' \arrow[ur, phantom, "(7)"] & L' \arrow[l, tail, "n'"] \arrow[d, tail, "m"'] \arrow[dr, phantom, "(5)"] & K' \arrow[l, tail] \arrow[r, tail] \arrow[d, tail] \arrow[dr, phantom, "(6)"] & R' \arrow[d, tail, "m^*"'] \\
 & G & D \arrow[l, tail] \arrow[r, tail] & H
\end{tikzcd}
$$

Construct pullback (PB) (5) leading to pushouts (POs) (3) and (5) by PB splitting and $\mathcal{M}$-PO-PB decomposition lemma (item 2 of Thm. 4.26 in (Ehrig et al.2006)). Construct PO (4) leading to PO (6) by PO splitting. Instantiate each NAC-schema $n : L \rightarrowtail N$ in $\mathbf{N}_S$ along morphism e (Def. 2.10) leading to a new NAC $n' : L' \rightarrowtail N'$. Let $\mathbf{N}'$ be the new set of NACs consisting of all NACs $n' : L' \rightarrowtail N'$ obtained from all $n \in \mathbf{N}_S$. The *instantiated rule* is given by $p' = ((L' \leftarrowtail K' \rightarrowtail R'), \mathbf{N}')$ and the *instantiated transformation step* is defined by $G \xRightarrow{p',m} H$ with $m \in \mathcal{M}$ via DPO diagram $((5) + (6))$.

Let d be a transformation sequence, then the instantiated transformation sequence d_I is derived by instantiating each transformation step as defined above.

The instantiation of rules ensures that transformation steps of the instantiated rule are in one-to-one correspondence to those of the original rule.

Fact 2.14 (Compatibility of Applicability and NAC-consistency with Instantiation). Let $G_1 \xRightarrow{p,f_1} H_1$ be a NAC-consistent transformation step, let $G_1 \xRightarrow{p',m_1} H_1$ be the instantiated step with extremal $\mathcal{E}$-$\mathcal{M}$-factorization $f_1 = m_1 \circ e$ according to Def. 2.13 and let $m_2 : L' \to G_2$ be a match with $m_2 \in \mathcal{M}$. Then, there is a NAC-consistent transformation step $G_2 \xRightarrow{p',m_2} H_2$ via p' if and only if there is a NAC-consistent transformation step $G_2 \xRightarrow{p,f_2} H_2$ via p with $f_2 = m_2 \circ e$.

Example 2.15 (Instantiation of Transformation Sequence). In the case of typed attributed graphs the instantiated rules are attributed via the algebra A of the transformed objects $G_0 \ldots G_n$. As in most cases the algebra A in our case study is different from the term algebra $T_{OP}(X)$. The instantiation of the transformation sequence d in Fig. 2 via rules of Fig. 1 is performed according to Def. 2.13. We derive an instantiated transformation sequence d_I. By definition, the lower line of the DPO diagrams coincides with the one of d in Fig. 2. The instantiated rules for the four steps are depicted in Figs 5 and 6 in Sec. 3.2 (rules "stop1", "stop2", "cont1", and "cont2") and they are used in the following sections for the analysis of permutation-equivalence.

2.3. *Permutation Equivalence of Transformation Sequences*

The classical theory of the DPO approach introduces an equivalence among transformation sequences which relates those sequences that differ only in the order in which independent transformation steps are performed leading to the notion of *switch equivalence.* Given a transformation sequence d without NACs, then switch equivalence provides the full set of all possible linearisations of the transformation steps of d (Baldan et al.2006; Baldan et al.1999). As we will show in this section, there are transformation sequences with NACs, for which some linerisations cannot be obtained via the notion of switch equivalence with NACs. For this reason, we introduce the more general notion of *permutation equivalence* and show that this notion generates the full set of all (NAC-consistent) linearisations for systems with NACs.

The *switch equivalence* is based on the notion of *sequential independence* and on the *local Church-Rosser theorem* and closed under isomorphism "$\cong$" of transformation sequences. Informally, transformation sequences d and d' are switch equivalent ($d \cong d'$) if they have the same length and there are isomorphisms between the corresponding objects of d and d' compatible with the involved morphisms.

Definition 2.16 (Sequential independence (disregarding NACs)). Let $d_1 = G_0 \xRightarrow{p_1,f_1} G_1$ and $d_2 = G_1 \xRightarrow{p_2,f_2} G_2$ be two transformation steps disregarding NACs. Then they are *sequentially independent* if there exist arrows $i : R_1 \to D_2$ and $j : L_2 \to D_1$ such that $g_2 \circ i = f_1^*$ and $h_1 \circ j = f_2$ (see the diagram on the right, which shows a part of the transformation diagrams).

$$
\begin{tikzcd}
K_1 \arrow[r, tail] \arrow[d] & R_1 \arrow[dr, "f_1^*"'] \arrow[drrr, dotted, "i"] & & L_2 \arrow[dl, "f_2"] \arrow[dlll, dotted, "j"'] & K_2 \arrow[l, tail] \arrow[d] \\
D_1 \arrow[rr, tail, "h_1"] & & G_1 & & D_2 \arrow[ll, tail, "g_2"']
\end{tikzcd}
$$

If d_1 and d_2 are sequentially independent, then according to the local Church-Rosser theorem (Thm. 5.12 in (Ehrig et al.2006)) they can be "switched" obtaining transformation steps $d_2' = G_0 \xRightarrow{p_2,\overline{f_2}} G_1'$ and $d_1' = G_1' \xRightarrow{p_1,\overline{f_1}} G_2$, which apply the two rules in the opposite order.

Definition 2.17 (Switch Equivalence for Transformation Sequences). Let $d = (d_1; \ldots; d_k;\ d_{k+1}; \ldots; d_n)$ be a transformation sequence, where d_k and d_{k+1} are two sequentially independent transformation steps, and let d' be obtained from d by switching them according to the Local Church-Rosser Theorem. Then, d' is a *switching of* d, written

$d \overset{sw}{\sim} d'$. The *switch equivalence*, denoted $\overset{sw}{\approx}$, is the smallest equivalence on transformation sequences containing both $\overset{sw}{\sim}$ and the relation $\cong$ for isomorphic transformation sequences.

Corresponding notions of parallel and sequential independence have been proposed for graph transformation systems with NACs (Habel et al.1996; Lambers2009). However, the derived notion of switch equivalence does not identify all intuitively equivalent transformation sequences with NACs. The reason is that, in presence of NACs, there might be an equivalent permutation of the transformation steps that cannot be derived by switch equivalence, which is indeed the case for our case study (see Ex. 2.19). This brings us to the following notion of permutation equivalence of NAC-consistent transformation sequences, first proposed in (Hermann2009). Note that for permutation-equivalent transformation sequences $d \overset{\pi}{\approx} d'$ the sequence of rules used in d' is a permutation of those used in d.

Definition 2.18 (Permutation Equivalence of Transformation Sequences). Two NAC-consistent transformation sequences d and d' are *permutation-equivalent*, written $d \overset{\pi}{\approx} d'$ if, disregarding the NACs, they are switch-equivalent as for Def. 2.17. The equivalence class of all permutation equivalent transformation sequences π-$Equ(d)$ of d is given by π-$Equ(d) = \{d' \mid d' \overset{\pi}{\approx} d\}$.

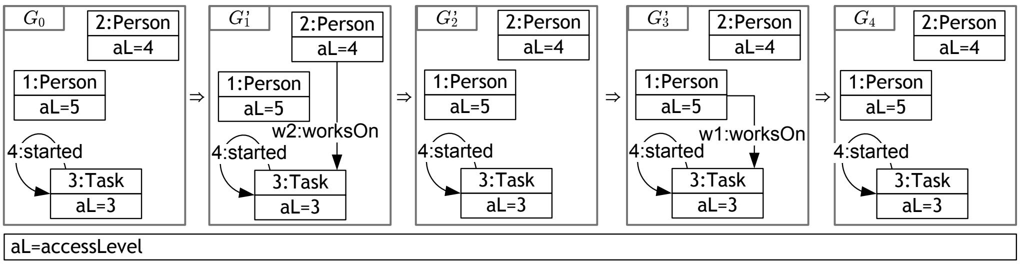

Fig. 3. Permutation-equivalent transformation sequence d' of GTS

Example 2.19 (Permutation Equivalence). Fig. 3 shows a NAC-consistent transformation sequence $d' = (G_0 \xRightarrow{continueTask,f'_1} G'_1 \xRightarrow{stopTask,f'_2} G'_2 \xRightarrow{continueTask,f'_3} G'_3 \xRightarrow{stopTask,f'_4} G_4)$, which is permutation-equivalent to the transformation sequence d of Fig. 2, by performing the following switchings of steps disregarding NACs: $(d_2; d_3), (d_1; d'_3), (d'_2; d_4), (d'_1; d'_4)$. The equivalent transformation sequences are not switch equivalent with NACs, because there is no pair of independent neighbouring transformation steps in any of the transformation sequences.

Remark 2.20 (Complexity of the Analysis). The brute-force method for generating all permutation-equivalent sequences would be to construct first all switch-equivalent ones disregarding NACs and then filtering out the NAC-consistent ones. But as discussed in (Hermann et al.2010), this is far too inefficient for realistic examples: given the transformation sequence d of Fig. 2, the sequence $d^3 = (d; d; d)$ consisting of twelve steps would lead to 7.484.400 switch-equivalent sequences disregarding NACs out of which only 720 are NAC-consistent and therefore permutation-equivalent. For this reason, we provide in Sec. 4 a more efficient approach by generating directly the permutation-equivalent ones. As shown in (Hermann2009) and (Hermann et al.2010), the construction of the derived Petri net has polynomial time complexity.

Given a transformation sequence d via general matches, we now show in Thm. 1 that we can reduce the analysis of permutation equivalence to $\mathcal{M}$-matches. For this purpose we first show by the following fact that there is a one-to-one correspondence between sequential independence disregarding NACs for the instantiated steps and for the corresponding original steps.

Fact 2.21 (Sequential Independence disregarding NACs for Instantiated Steps). Let $(d_1; d_2) = (G_0 \xrightarrow{p_1,f_1} G_1 \xrightarrow{p_2,f_2} G_2)$ be two transformation steps disregarding NACs and let $(d_{1,I}; d_{2,I}) = (G_0 \xrightarrow{p'_1,m_1} G_1 \xrightarrow{p'_2,m_2} G_2)$ be their instantiated steps according to Def. 2.13. Then, d_1 and d_2 are sequentially independent disregarding NACs iff $d_{1,I}$ and $d_{2,I}$ are sequentially independent disregarding NACs.

Theorem 1 (Reduction of Permutation Equivalence for General Matches to $\mathcal{M}$-matches). Two transformation sequences d and d' with general matches are permutation-equivalent if and only if their instantiated transformation sequences d_I and d'_I with $\mathcal{M}$-matches are permutation-equivalent, i.e. $d \overset{\pi}{\approx} d' \Leftrightarrow d_I \overset{\pi}{\approx} d'_I$.

Proof. First of all, we have by Fact 2.21 and Def. 2.17 that switch equivalence disregarding NACs is implied for both directions. By Fact 2.14 we have that the transformation steps and hence, also the transformation sequences, are additionally NAC consistent. Therefore, $d \overset{\pi}{\approx} d' \Leftrightarrow d_I \overset{\pi}{\approx} d'_I$. □

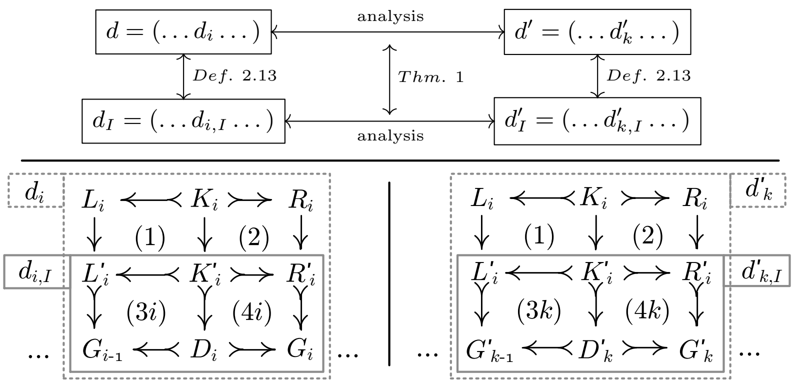

Fig. 4. Correspondence between transformation sequences and their instantiations

Remark 2.22 (Permutation Equivalence for General Matches). By the above theorem we can base our analysis techniques in the following sections on the derived transformation sequences with $\mathcal{M}$-morphisms only as visualized in Fig. 4. Given a transformation sequence d, we first instantiate d according to Def. 2.13, such that the lower transformation diagrams form a new transformation sequence d_I with $\mathcal{M}$-matches only. Thereafter, we can analyse permutation equivalence for d_I and derive the analysis results for d via Thm. 1. In particular, the derived permutation-equivalent transformation sequences d'_I of d_I can be composed with the upper DPO diagrams of the instantiation leading to permutation-equivalent transformation sequences d' of d.

General Assumption: As a consequence of the above remark, in the following sections we will consider transformation sequences with $\mathcal{M}$-matches only. In fact, for analysing transformation sequences with general matches it is sufficient to analyse their instantiated sequences, lifting back the results to the original sequences using Thm. 1.

3. From Subobject Transformation Systems to Processes of $\mathcal{M}$-adhesive Transformation Systems

In the theory of Petri nets (Reisig1985), from a given firing sequence one can build a *deterministic process*, which is a net which records all the transitions fired in the sequence, together with their causal dependencies. Similar constructions have been proposed for graph transformation (Corradini et al.1996) and for transformation systems based on adhesive categories (Baldan et al.2006; Corradini et al.2008). In particular, in (Corradini et al.2008) it is shown that starting with a transformation sequence (without NACs) in an adhesive transformation system one can build a *Subobject Transformation System (STS)*, i.e. a system where the sequence can be simulated and where it is possible to analyse the independence among steps of the sequence. In this section we generalize these results to transformation systems with NACs, and we will consider the more general framework of $\mathcal{M}$-adhesive categories.

3.1. *$\mathcal{M}$-Subobject Transformation Systems*

Subobject transformation systems are essentially double-pushout transformation systems over the lattice of subobjects $\mathbf{Sub}(T)$ of a given object T of an adhesive category $\mathbf{C}$. We revisit here the main definitions of (Corradini et al.2008) in the case of $\mathcal{M}$-adhesive categories, starting with the notion of $\mathcal{M}$-subobject. In the following we assume that $\mathbf{C}$ is an arbitrary but fixed $\mathcal{M}$-adhesive category, unless specified differently, and by $|\mathbf{C}|$ we denote the class of objects of $\mathbf{C}$.

Definition 3.1 (Category of $\mathcal{M}$-Subobjects). Let T be an object of an $\mathcal{M}$-adhesive category $\mathbf{C}$. Given two $\mathcal{M}$-morphisms $a : A \rightarrowtail T$ and $a' : A' \rightarrowtail T$, they are *equivalent* if there exists an isomorphism $\phi : A \to A'$ such that $a = a' \circ \phi$. An *$\mathcal{M}$-subobject* $[a : A \rightarrowtail T]$ of T is an equivalence class of $\mathcal{M}$-morphisms with target T. The *category of $\mathcal{M}$-subobjects of* T, denoted $\mathbf{Sub}_{\mathcal{M}}(T)$, has the $\mathcal{M}$-subobjects of T as objects. Furthermore, there is an arrow from $[a : A \rightarrowtail T]$ to $[b : B \rightarrowtail T]$ if there exists a morphism $f : A \to B$ such that $a = b \circ f$; in this case f is an $\mathcal{M}$-morphism and it is unique (therefore $\mathbf{Sub}_{\mathcal{M}}(T)$ is a partial order), and we write $[a : A \rightarrowtail T] \subseteq [b : B \rightarrowtail T]$.

Usually we will denote an $\mathcal{M}$-subobject $[a : A \rightarrowtail T]$ simply by A, leaving the $\mathcal{M}$-morphism a implicit, and correspondingly we write $A \subseteq B$ if $[a : A \rightarrowtail T] \subseteq [b : B \rightarrowtail T]$ and denote the corresponding embedding by $f : A \hookrightarrow B$.

If $\mathcal{M}$ is the class of all monomorphism of $\mathbf{C}$, as for adhesive categories, then $\mathbf{Sub}_{\mathcal{M}}(T)$ for $T \in |\mathbf{C}|$ is the standard category of subobjects of T. The following notions of "intersection" and "union" will be used in the definition of direct derivations of an STS.

Definition 3.2 (Intersection and Union in $\mathbf{Sub}_{\mathcal{M}}(T)$). Let $A, B \in |\mathbf{Sub}_{\mathcal{M}}(T)|$ be two $\mathcal{M}$-subobjects, with $T \in |\mathbf{C}|$. The product of A and B in $\mathbf{Sub}_{\mathcal{M}}(T)$ will be called their *intersection*, denoted $A \cap B$. The coproduct of A and B in $\mathbf{Sub}_{\mathcal{M}}(T)$ will be called *union*, denoted $A \cup B$.

In the case of adhesive categories, as shown in (Lack and Sobocinski2005), intersections and unions exist, unions are effective, and $\mathbf{Sub}(T)$ is a distributive lattice for any $T \in \mathbf{C}$. We show that also for $\mathcal{M}$-adhesive categories $\mathbf{Sub}_{\mathcal{M}}(T)$ is a distributive lattice if unions are effective. Since unions are not effective in general, we require this property by our general assumption in Sec. 2.1.

Fact 3.3 (Intersection in $\mathbf{Sub}_{\mathcal{M}}(T)$). Let $T \in |\mathbf{C}|$ and $A, B \in \mathbf{Sub}_{\mathcal{M}}(T)$. The intersection $A \cap B$ exists and it is given by the pullback (1) in $\mathbf{C}$ with the $\mathcal{M}$-morphism $i : A \cap B \xrightarrow{a \,\circ\, p_A} T$.

$$\begin{tikzcd} A \cap B \arrow[r, hook, "p_A"] \arrow[d, hook, "p_B"'] \arrow[dr, phantom, "(1)"] & A \arrow[d, hook, "a"] \\ B \arrow[r, hook, "b"'] & T \end{tikzcd}$$

Remark 3.4 (Unions in $\mathbf{Sub}_{\mathcal{M}}(T)$ for $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$). According to Rem. 2.7 in Sec. 2.1 the category of typed attributed graphs $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$ has effective unions, i.e. the union $A \cup B$ of two $\mathcal{M}$-subobjects A and B can be constructed as the pushout over the intersection $A \cap B$ in $\mathbf{C}$.

In contrast to $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$, the category of simple graphs provides an example of an $\mathcal{M}$-adhesive category which has unions, but where unions are not effective. A simple graph is a pair (A, N) where N is a set of nodes and $A \subseteq N \times N$ is a set of arcs. A morphism $f : (N, A) \to (N', A')$ is a function $f : N \to N'$ such that $(n_1, n_2) \in A \Rightarrow (f(n_1), f(n_2)) \in A'$. Such a morphism is *regular* if it is injective and also the opposite implication holds.

The category of simple graphs with the class $\mathcal{M}$ of all regular monomorphism is shown to be a partial-map adhesive category in (Heindel2010), and therefore it is $\mathcal{M}$-adhesive by the results in (Ehrig et al.2010). But it is well-known that unions are not effective in this category: given the graph $G = (\{n, n'\}, \{(n, n')\})$, the pushout built over the regular subobjects $(\{n\}, \emptyset)$ and $(\{n'\}, \emptyset)$ is $(\{n, n'\}, \emptyset)$, which is not regular.

Fact 3.5 (Distributivity). Let $\mathbf{C}$ be an $\mathcal{M}$-adhesive category with effective unions and T be an object of $\mathbf{C}$, then the union and intersection constructions in $\mathbf{Sub}_{\mathcal{M}}(T)$ are distributive, i.e.

$$(i): \quad A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad \text{and}$$
$$(ii): \quad A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$

Based on the notion of $\mathcal{M}$-subobjects and the distributivity law for intersection and union we now present subobject transformation systems (STSs) as a formal framework for the concurrent semantics of $\mathcal{M}$-adhesive transformation systems. This concept generalises the notion of elementary nets, which form the category of process nets for P/T Petri nets, in the way that STSs form the category of process transformation systems for $\mathcal{M}$-adhesive transformation systems. The typical effect occurring in elementary nets – namely the situation of contact – also appears in the setting of STSs and forms an additional application condition for the transformation rules. Thus, we first introduce the general setting of STSs on which we base the construction of the process of a transformation sequence thereafter.

Definition 3.6 (STS with NACs). A *Subobject Transformation System with NACs* $\mathcal{S} = (T, P, \pi)$ over an $\mathcal{M}$-adhesive category $\mathbf{C}$ with effective unions consists of a super object $T \in \mathbf{C}$, a set of rule names P – also called productions – and a function π, which maps a rule name $q \in P$ to a *rule with negative application conditions (NACs)* $((L, K, R), \mathbf{N})$, where L, K, and R are objects in $\mathbf{Sub}_{\mathcal{M}}(T)$, $K \subseteq L$, $K \subseteq R$ and its NACs $\mathbf{N}$ are given by $\mathbf{N} = (N, \nu)$ consisting of a set N of names for the NACs together with a function ν mapping each NAC name $i \in N$ to a *NAC* $\nu(i)$, which is given by a subobject $\nu(i) = N_i \in \mathbf{Sub}_{\mathcal{M}}(T)$ with $L \subseteq N_i \subseteq T$. The short notation $\mathbf{N}[i]$ refers to a NAC N_i of rule p with $\nu(i) = N_i$.

Direct derivations ($G \overset{q}{\Longrightarrow} G'$) with NACs in an STS correspond to transformation steps with NACs in $\mathcal{M}$-adhesive TS, but the construction is simplified, because morphisms between two subobjects are unique. There is no need for pattern matching and for this reason, we use the notion of derivations within an STS in contrast to transformation sequences in an $\mathcal{M}$-adhesive TS and we use names $\{p_1, \ldots, p_n\}$ for rules in an $\mathcal{M}$-adhesive TS and $\{q_1, \ldots, q_n\}$ for rules in an STS.

Definition 3.7 (Direct Derivations with NACs in an STS). Let $\mathcal{S} = (T, P, \pi)$ be a Subobject Transformation System with NACs, $\pi(q) = ((L, K, R), \mathbf{N})$ be a production with NACs, and let $G \in |\mathbf{Sub}_{\mathcal{M}}(T)|$. Then there is a *direct derivation with NACs* from G to G' using q, written $G \overset{q}{\Longrightarrow} G'$, if $G' \in |\mathbf{Sub}_{\mathcal{M}}(T)|$, for each $\mathbf{N}[i]$ in $\mathbf{N}$: $\mathbf{N}[i] \not\subseteq G$, and there is an object $D \in \mathbf{Sub}_{\mathcal{M}}(T)$ such that:

$$\begin{array}{llll} (i) & L \cup D = G; & (ii) & L \cap D = K; \\ (iii) & D \cup R = G', \text{ and} & (iv) & D \cap R = K. \end{array}$$

It is instructive to consider the relationship between a direct derivation in an STS and the usual notion of a DPO transformation step in an $\mathcal{M}$-adhesive category. It is possible to make this comparison, since one can consider a rule $L_q \supseteq K_q \subseteq R_q$ as the underlying span of $\mathcal{M}$-morphisms in $\mathbf{C}$. However, given an $\mathcal{M}$-subobject $G \in \mathbf{Sub}_{\mathcal{M}}(T)$ such that $L \subseteq G$, an additional condition has to be satisfied in order to guarantee that the result of a double-pushout transformation in $\mathbf{C}$ using rule $L_q \supseteq K_q \subseteq R_q$ and match $L \subseteq G$ is again an object in $\mathbf{Sub}_{\mathcal{M}}(T)$.

In fact, suppose that $G \cap R \not\subseteq L$. Intuitively, this means that part of the $\mathcal{M}$-subobject G is created but not deleted by the rule: if we were allowed to apply the rule at this match via a DPO transformation step, the resulting object would contain the common part twice and consequently the resulting morphism to T would not be an $\mathcal{M}$-morphism; i.e., the result would not be an $\mathcal{M}$-subobject of T.

By analogy with a similar concept for elementary Petri nets, we shall say that there is a *contact situation* for a rule (L, K, R) at an $\mathcal{M}$-subobject $G \supseteq L \in \mathbf{Sub}_{\mathcal{M}}(T)$ if $G \cap R \nsubseteq L$: as stated by the next result STS direct derivations and DPO transformation steps coincide if there is no contact.

Fact 3.8 (STS Derivations are Contact-Free Double Pushouts). Let $\mathcal{S} = (T, P, \pi)$ be an STS over an $\mathcal{M}$-adhesive category $\mathbf{C}$ with effective unions, $\pi(q) = (L, K, R)$ be a rule, and $G \in |\mathbf{Sub}_{\mathcal{M}}(T)|$. Then $G \overset{q}{\Longrightarrow} G'$ iff $L \subseteq G$, $G \cap R \subseteq L$, and there is an object D in $\mathbf{C}$ such that diagrams (1) and (2) are pushouts in $\mathbf{C}$.

$$\begin{tikzcd} L \arrow[d, tail, "m"'] \arrow[dr, phantom, "(1)"] & K \arrow[l, tail, "l"'] \arrow[r, tail, "r"] \arrow[d, tail, "k"] \arrow[dr, phantom, "(2)"] & R \arrow[d, tail, "n"] \\ G & D \arrow[l, tail, "f"] \arrow[r, tail, "g"'] & G' \end{tikzcd}$$

Proof. See the proof of Prop. 6 in (Corradini et al.2008). $\square$

3.2. *Processes of $\mathcal{M}$-adhesive Transformation Systems*

Based on the notion and construction of processes for adhesive transformation systems without NACs in (Baldan et al.2006) and (Corradini et al.2008), this section presents the construction of processes for a transformation sequence of an $\mathcal{M}$-adhesive transformation systems with NACs. The first step is to construct the STS for a given transformation sequence d with matches in $\mathcal{M}$ due to the general assumption in Sec. 2.3 based on Thm. 1.

Definition 3.9 (STS of a Transformation Sequence with $\mathcal{M}$-matches). Let $d = (G_0 \xRightarrow{p_1, m_1} \dots \xRightarrow{p_n, m_n} G_n)$ be a NAC-consistent transformation sequence in an $\mathcal{M}$-adhesive TS with matches in $\mathcal{M}$. The *STS with NACs generated by* d is given by $STS(d) = (T, P, \pi)$ and its components are constructed as follows. T is the colimit of the DPO-diagrams given by d, $P = \{i \mid 0 < i \leq n\}$ is a set that contains a rule occurrence name for each rule occurrence in d. For each $k \in P$ let $\pi(k) = ((L_k \supseteq K_k \subseteq R_k), \mathbf{N}_k)$ using the embeddings $in_T(X)$ into T for $X \in \{L_k, K_k, R_k\}$ of the rules in d and the NACs $\mathbf{N}_k = (N_k, \nu)$ are constructed as follows. For each NAC $n : L_k \rightarrowtail N$ of p_k construct the possible subobjects $[j : N \rightarrowtail T] \in \mathbf{Sub}_{\mathcal{M}}(T)$ such that $j \circ n = in_T(L_k)$ building up the set $J_{\mathbf{N}_k}$ of constructed subobjects $[j]$. The NAC names N_k are given by $N_k = \{i \mid 0 < i \leq |J_{\mathbf{N}_k}|\}$ and the function ν is an arbitrary but fixed bijective function $\nu : N_k \to J_{\mathbf{N}_k}$ mapping NAC names to corresponding subobjects.

When analysing permutation equivalence in concrete case studies we consider only transformation sequences such that the colimit object T is finite, i.e. has finitely many $\mathcal{M}$-subobjects, in order to ensure termination. Finiteness is guaranteed if each rule of TS has finite left- and right-hand sides, and if the start object of the transformation sequence is finite. For typed attributed graphs, this means that T is finite on the structural part, but the carrier sets of the data algebra for the attribution component may by infinite ($\mathcal{M}$-morphisms in $\mathbf{AGraphs}_{ATG}$ are isomorphisms on the data part).

Remark 3.10. Note that during the construction of $STS(d)$ the set of instantiated NACs for a NAC of a rule p applied in d may be empty, which means that the NAC n cannot be found within T. Furthermore, if we require T to be finite, the lists of NACs in $STS(d)$ are finite.

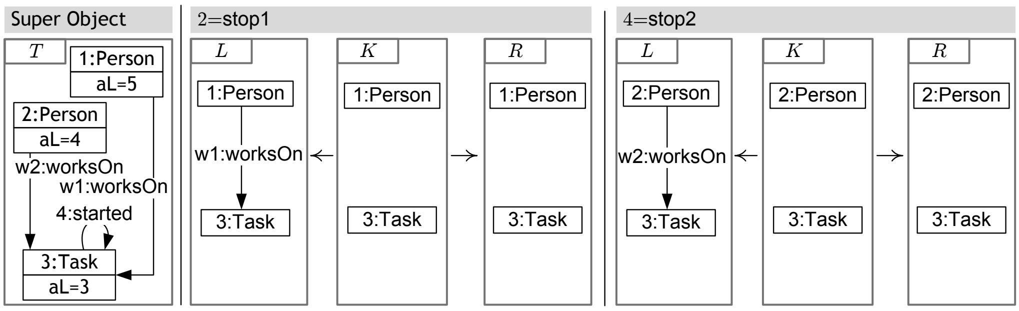

Fig. 5. Super object T and two rules of process $Prc(d)$

Fig. 6. Further rules of STS $STS(d)$

Example 3.11 (Derived STS $STS(d)$). For the transformation sequence in Fig. 2 the construction of the STS leads to the STS as shown in Figs. 5 and 6. The transformation sequence d involves the rules “continueTask” and “stopTask” and thus, the derived STS contains the rule occurrences “cont1”, “cont2”, “stop1” and “stop2”.

The process of a transformation sequence d consists of the STS derived from d according to Def. 3.9 together with an embedding v relating the STS with the TS of the given transformation sequence. A process of d induces the complete equivalence class of transformation sequences with respect to permutation equivalence, which we show by Thm. 2 below.

Definition 3.12 (Process of a Transformation Sequence with NACs). Let $d = (G_0 \xRightarrow{q_1,m_1} \ldots \xRightarrow{q_n,m_n} G_n)$ be a NAC-consistent transformation sequence in an $\mathcal{M}$-adhesive transformation system $TS = (P_{TS}, \pi_{TS})$. The *process* $Prc(d) = (STS(d), \mu)$ of d consists of the derived STS $STS(d) = (T, P, \pi)$ of d together with the mapping $\mu : STS(d) \to TS$ given by $\mu : P \to P_{TS}, \mu(i) = q_i$ for each step i of d.

Note that the mapping μ induces a function $\mu_\pi : \pi(P) \to \pi_{TS}(P_{TS})$ mapping each rule in $STS(d)$ to the corresponding rule in TS, where $\mu_\pi(\pi(q)) = \pi_{TS}(\mu(q))$.

The notion of processes for transformation sequences corresponds to the notion of processes for Petri nets given by an occurrence net together with a Petri net morphism into the system Petri net. Moreover, as shown in (Corradini et al.2008) the process construction yields a special kind of STS, called *pure* STS meaning that no rule deletes and produces again the same part of a subobject, i.e. $L \cap R = K$. This terminology is adapted from the theory of Elementary Net Systems, where a system which does not contain transitions with a self-loop is called "pure". Therefore, the class of pure STSs can be seen as a generalisation of elementary nets to the setting of $\mathcal{M}$-adhesive transformation systems and thus, as a generalisation of the Petri net class of occurrence nets.

According to Fact 3.8, a direct derivation in an STS induces a DPO diagram in the underlying $\mathcal{M}$-adhesive category $(\mathbf{C}, \mathcal{M})$. Therefore, a derivation in an STS gives rise to a transformation sequence in $(\mathbf{C}, \mathcal{M})$ according to Def. 3.13 below, where we denote by $TRAFO(\mathbf{C}, \mathcal{M})$ the set of all transformation sequences in the $\mathcal{M}$-adhesive category $(\mathbf{C}, \mathcal{M})$.

Definition 3.13 (Transformation Sequence of an STS-sequence). Let $d_S = (G_0 \xRightarrow{q_1} G_1 \xRightarrow{q_2} \ldots \xRightarrow{q_n} G_n)$ be a derivation in the STS $\mathcal{S}$ of a process $Prc(d) = (\mathcal{S}, \mu)$ of a NAC-consistent transformation sequence d in an $\mathcal{M}$-adhesive TS with $\mathcal{S} = (T, P, \pi)$ and let $s = \langle q_1; \ldots; q_n \rangle$ denote the sequence of the rule occurrences according to d_S. Then, the partial mapping *trafo* $: P^* \to TRAFO(\mathbf{C}, \mathcal{M})$ maps the sequence s to the sequence of DPO diagrams (transformation steps) in $\mathbf{C}$ for each derivation step $G_{i-1} \xRightarrow{q_i} G_i$ in $\mathcal{S}$. Moreover, the partial mapping *seq* $: TRAFO(\mathbf{C}, \mathcal{M}) \to P^*$ maps the transformation sequence *trafo*(s) to the sequence s of rule names in $\mathcal{S}$.

The partial mappings *seq* and *trafo* are well defined and are inverse to each other according to Fact 3.14 below.

Fact 3.14 (Correspondence between Transformation and Rule Sequences). The partial mappings *trafo* and *seq* are well defined, i.e. they are partial functions. Moreover, let $d_S = (G_0 \xRightarrow{q_1} G_1 \xRightarrow{q_2} \ldots \xRightarrow{q_n} G_n)$ be a derivation in a process $Prc(d)$ of a transformation sequence, then $d = trafo(seq(d))$ and $s = seq(trafo(s))$.

The following relations between the rules of an STS with NACs specify the possible dependencies among them: the first four relations are discussed in (Corradini et al.2008), while the last two are introduced in (Hermann2009).

Table 1. Relations on rules in an STS

Name	Notation	Condition		
Read Causality	$q_1 <_{rc} q_2$	$R_1 \cap K_2$	$\not\subseteq$	K_1
Write Causality	$q_1 <_{wc} q_2$	$R_1 \cap L_2$	$\not\subseteq$	$K_1 \cup K_2$
Deactivation	$q_1 <_{d} q_2$	$K_1 \cap L_2$	$\not\subseteq$	K_2
Independence	$q_1 \lozenge q_2$	$(L_1 \cup R_1) \cap (L_2 \cup R_2)$	$\subseteq$	$K_1 \cap K_2$
Weak NAC Enabling	$q_1 <_{wen[i]} q_2$	$0 < i \leq \lvert\mathbf{N}_2\rvert \ \wedge \ L_1 \cap \mathbf{N}_2[i]$	$\not\subseteq$	$K_1 \cup L_2$
Weak NAC Disabling	$q_1 <_{wdn[i]} q_2$	$0 < i \leq \lvert\mathbf{N}_1\rvert \ \wedge \ \mathbf{N}_1[i] \cap R_2$	$\not\subseteq$	$L_1 \cup K_2$

Definition 3.15 (Relations on Rules). Let q_1 and q_2 be two rules in an STS $\mathcal{S} = (T, P, \pi)$ with $\pi(q_i) = ((L_i, K_i, R_i), \mathbf{N}_i)$ for $i \in \{1, 2\}$. The relations on rules are defined on P as shown in Tab. 1.

In words, $q_1 <_{wen[i]} q_2$ (read: "q_1 weakly enables q_2 at i") if q_1 deletes a piece of the NAC $\mathbf{N}[i]$ of q_2; instead $q_1 <_{wdn[i]} q_2$ ("q_2 weakly disables q_1 at i") if q_2 produces a piece of the NAC $\mathbf{N}[i]$ of q_1. It is worth stressing that the relations introduced above are not transitive in general.

Example 3.16 (Relations of an STS). The rules of $STS(d)$ in Ex. 3.11 are related by the following dependencies. For write causality we have "cont1 $<_{wc}$ stop1" and "cont2 $<_{wc}$ stop2". The further dependencies are shown below:

Weak Enabling		Weak Disabling	
stop1 $<_{wen[1]}$ cont1	stop2 $<_{wen[2]}$ cont1	cont1 $<_{wdn[1]}$ cont1	cont2 $<_{wdn[2]}$ cont2
stop1 $<_{wen[1]}$ cont2	stop2 $<_{wen[2]}$ cont2	cont2 $<_{wdn[1]}$ cont1	cont1 $<_{wdn[2]}$ cont2

Definition 3.17 (STS-Switch Equivalence of Sequences disregarding NACs). Let $\mathcal{S} = (T, P, \pi)$ be an *STS*, let d be a derivation in $\mathcal{S}$ disregarding NACs and let $s = \langle q_1, \ldots, q_n \rangle$ be its corresponding sequence of rule occurrence names. Let $q_k \lozenge q_{k+1}$, then the sequence $s' = \langle q_1, \ldots, q_{k+1}, q_k, \ldots, q_n \rangle$ is *STS-switch-equivalent* to the sequence s, written $s \overset{sw}{\sim}_{\mathcal{S}} s'$. Switch equivalence $\overset{sw}{\approx}_{\mathcal{S}}$ of rule sequences is the transitive closure of $\overset{sw}{\sim}_{\mathcal{S}}$.

In order to characterise the set of possible permutations of transformation steps of a given transformation sequence, we now define suitable conditions for permutations of rule occurrences. We call rule sequences of a derived STS $STS(d)$ *legal sequences*, if they are switch-equivalent without NACs to the sequence of rules $seq(d)$ of d and if the following condition concerning NACs holds. For every NAC $\mathbf{N}[i]$ of a rule q_k, either there is a rule which *deletes* part of $\mathbf{N}[i]$ and is applied *before* q_k, or there is a rule which *produces* part of $\mathbf{N}[i]$ and is applied *after* q_{k-1}. In both cases $\mathbf{N}[i]$ cannot be present when applying q_k, because the STS $STS(d)$ is a sort of "unfolding" of the transformation sequence, and every subobject is created at most once and deleted at most once (see (Corradini et al.2008)).

Definition 3.18 (Legal Sequence). Let $d = (d_1; \ldots; d_n)$ be a NAC-consistent transformation sequence in an $\mathcal{M}$-adhesive TS, and let $STS(d) = (T, P, \pi_N)$ be its derived STS. A sequence $s = \langle q_1; \ldots; q_n \rangle$ of rule names of P is *locally legal at position* $k \in \{1, \ldots, n\}$ *with respect to* d, if each rule name in P occurs exactly once in s and the following conditions hold:

1 $s \overset{sw}{\approx}_{STS(d)} seq(d)$

2 $\forall$ NAC $\mathbf{N}_k[i]$ of q_k : $\begin{pmatrix} \exists \ e \in \{1, \ldots, k-1\} : & q_e <_{wen[i]} q_k \text{ or} \\ \exists \ l \in \{k, \ldots, n\} : & q_k <_{wdn[i]} q_l. \end{pmatrix}$

A sequence s of rule names is *legal with respect to* d, if it is locally legal at all positions $k \in \{1, ..., \text{n}\}$ with respect to d.

Definition 3.19 (STS-Equivalence of Rule Sequences). Let d be a NAC-consistent transformation sequence with NACs of an $\mathcal{M}$-adhesive TS and let $Prc(d) = (STS(d), \mu)$ be its derived process. Two sequences s, s' of rule names in $STS(d)$ are STS-equivalent, written $s \approx_{STS(d)} s'$, if they are legal sequences with respect to d. The set of all STS-equivalent sequences of $Prc(d)$ is given by $Seq(d) = \{s \mid s \approx_{STS(d)} seq(d)\}$. Moreover, the specified class of transformation sequences of $Seq(d)$ is given by $Trafo(s) = [trafo(s)]_{\cong}$ for single sequences and $Trafo(Seq(d)) = \bigcup_{s \in Seq(d)} Trafo(s)$ for the complete set.

Theorem 2 (Characterization of Permutation Equivalence Based on STSs). Given the process $Prc(d)$ of a NAC-consistent transformation sequence d.

1 The class of permutation-equivalent transformation sequences of d coincides with the set of derived transformation sequences of the process $Prc(d)$ of d:
$\pi\text{-}Equ(d) = Trafo(Seq(d))$

2 The mapping *Trafo* defines a bijective correspondence between STS-equivalent sequences of rule names and permutation-equivalent transformation sequences:
$Trafo : Seq(d) \xrightarrow{\sim} (\pi\text{-}Equ(d))/_{\cong}$

Proof. Let d be a NAC-consistent transformation sequence in an $\mathcal{M}$-adhesive TS and let $Prc(d) = (\mathcal{S}, \mu)$ be the process of d with $\mathcal{S} = (T, P, \pi)$. We have to show that each STS-equivalent rule sequence s' of $seq(d)$ in $\mathcal{S}$ defines a permutation-equivalent transformation sequence $trafo(s')$ of d and vice versa, each permutation-equivalent transformation sequence d' of d defines an STS-equivalent rule sequence $seq(d')$ of $seq(d)$ in $\mathcal{S}$:

$$\forall\, s' \in P^* : \quad s' \approx_{STS(d)} seq(d) \quad \Rightarrow \quad trafo(s') \overset{\pi}{\approx} d \quad (1)$$

$$\forall\, d' \in TRAFO(\mathbf{C}, \mathcal{M}) : \quad d' \overset{\pi}{\approx} d \quad \Rightarrow \quad seq(d') \approx_{STS(d)} seq(d) \quad (2)$$

The proof of Thm. 1 in (Hermann2009) shows the results (1) and (2) for the case of adhesive transformation systems with NACs and monomorphic matches using the operations intersection and union on subobjects and distributivity. The operations are available for $\mathcal{M}$-adhesive transformation systems with effective unions, which we require by our general assumption in Sec. 2.1, intersection is given by Fact 3.3 and distributivity is shown by Fact. 3.5. Thus, (1) and (2) hold for $\mathcal{M}$-adhesive transformation systems with $\mathcal{M}$-matches.

Finally, by Def. 3.19 we have that $d' \in \mathit{Trafo}(\mathit{Prc}(d))$ is equivalent to $d' = \mathit{trafo}(s')$ and $s' \approx_{STS(d)} \mathit{seq}(d)$. Using (1) and (2) above together with Def. 2.18 we derive $\pi\text{-}\mathit{Equ}(d) = \mathit{Trafo}(\mathit{Prc}(d))$. $\square$

According to Thm. 2, the construction of the process $\mathit{Prc}(d)$ of a transformation sequence d specifies the equivalence class of all transformation sequences which are permutation-equivalent to d. In the next section, we present an efficient analysis technique for processes based on Petri nets.

4. Analysis of Permutation Equivalence Based on Petri Nets

Based on the process of a transformation sequence given by an STS, we now present the construction of its *dependency net*, given by a P/T Petri net which specifies only the dependencies between the transformation steps. All details about the internal structure of the objects and the transformation rules are excluded, allowing us to increase the efficiency of the analysis of permutation equivalence (see Rem. 2.20). The names of the generated placed of the dependency net are composed of constant symbols and numbers, where constant symbols s are denoted by $\mathtt{s}$.

Definition 4.1 (Dependency Net *DNet* of a Transformation Sequence). Let d be a NAC-consistent transformation sequence of an $\mathcal{M}$-adhesive TS, let $STS(d) = (T, P, \pi)$ be the generated STS of d and let $s = \mathit{seq}(d) = \langle q_1, \dots, q_n \rangle$ be the sequence of rule names in $STS(d)$ according to the steps in d. The dependency net of d is given by the following marked Petri net $\mathit{DNet}(d) = (\mathit{Net}, M)$, $\mathit{Net} = (\mathit{PL}, \mathit{TR}, \mathit{pre}, \mathit{post})$:

— $\mathit{TR} = P = \{1 \dots n \mid n = |P|\}$

— $\mathit{PL} = \{\mathtt{p}(q) \mid q \in \mathit{TR}\} \cup \{\mathtt{p}(q' \mathtt{<}_x q) \mid q, q' \in \mathit{TR}, x \in \{rc, wc, d\}, q' <_x q\}$
$\cup \{\mathtt{p}(q\mathtt{,N[}i\mathtt{]}) \mid q \in \mathit{TR}, \pi(q) = ((L_q, K_q, R_q), \mathbf{N}), 0 < i \le |\mathbf{N}|, q \not<_{wdn[i]} q\}$

— $$\mathit{pre}(q) = \mathtt{p}(q) \oplus \sum_{\substack{q' <_x q \\ x \in \{rc, wc, d\}}} \mathtt{p}(q' \mathtt{<}_x q) \oplus \sum_{\substack{q' <_{wdn[i]} q \\ q' \neq q}} \mathtt{p}(q'\mathtt{,N[}i\mathtt{]}) \oplus \sum_{\mathtt{p}(q\mathtt{,N[}i\mathtt{]}) \in PL} \mathtt{p}(q\mathtt{,N[}i\mathtt{]})$$

— $$\mathit{post}(q) = \sum_{\substack{q <_x q' \\ x \in \{rc, wc, d\}}} \mathtt{p}(q \mathtt{<}_x q') \oplus \sum_{q <_{wen[i]} q'} \mathtt{p}(q'\mathtt{,N[}i\mathtt{]}) \oplus \sum_{\mathtt{p}(q\mathtt{,N[}i\mathtt{]}) \in PL} \mathtt{p}(q\mathtt{,N[}i\mathtt{]})$$

— $$M = \sum_{q \in TR} \mathtt{p}(q) \oplus \sum_{\substack{q' <_{wdn[i]} q \\ \mathtt{p}(q'\mathtt{,N[}i\mathtt{]}) \in PL}} \mathtt{p}(q'\mathtt{,N[}i\mathtt{]})$$

Figure 7 shows how the dependency net is constructed algorithmically. The construction steps are performed in the order they appear in the table. Each step is visualized as a rule, where gray line colour and plus-signs mark the elements to be inserted. The matched context that is preserved by a rule is marked by black line colour, e.g. in step 2 the new place "$p(q <_x q')$" is inserted between the already existing transitions q and q'.

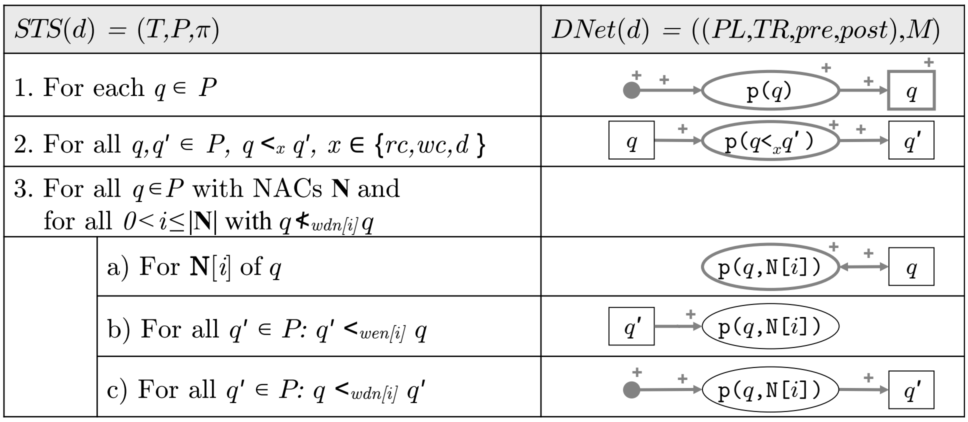

Fig. 7. Visualization of the construction of the Petri net

The tokens of the initial marking of the net are represented by bullets that are connected to their places via arcs. In the first step, each rule q of the STS is encoded as a transition and it is connected to a marked place, which prevents the transition to fire more than once. In step 2, between each pair of transitions in each of the relations $<_{rc}$, $<_{wc}$ and $<_d$, a new place is created in order to enforce the corresponding dependency. The rest of the construction is concerned with places which correspond to NACs and can contain several tokens in general. Each token in such a place represents *the absence* of a piece of the NAC; therefore if the place is empty, the NAC is complete.

In this case, by step (3a) the transition cannot fire. Consistently with this intuition, if $q' <_{wen[i]} q$, i.e. transition q' consumes part of the NAC $\mathbf{N}[i]$ of q, then by step (3b) q' produces a token in the place corresponding to $\mathbf{N}[i]$. Symmetrically, if $q <_{wdn[i]} q'$, i.e. q' produces part of NAC $\mathbf{N}[i]$ of q, then by step (3c) q' consumes a token from the place corresponding to $\mathbf{N}[i]$. Notice that each item of a NAC is either already in the start graph of the transformation sequence or produced by a single rule. If a rule generates part of one of its NACs, say $\mathbf{N}[i]$ ($q <_{wdn[i]} q$), then by the acyclicity of $Prc(d)$ the NAC $\mathbf{N}[i]$ cannot be completed before the firing of q: therefore we ignore it in the third step of the construction of the dependency net. Examples of such weakly self-disabling rules are rules ($1 = cont1$) and ($3 = cont2$) in Fig. 6, where the specific NACs coincide with the right hand sides of the rules ($NAC2 = R$).

Note that the constructed net in general is not a safe one, because the places for the NACs can contain several tokens. Nevertheless it is a bounded P/T net. The bound is the maximum of one and the maximal number of adjacent edges at a NAC place minus two.

Example 4.2 (Dependency Net). Consider the transformation sequence d in Fig. 2 from Ex. 2.12 and its derived STS in Ex. 3.11. The marked Petri net in Fig. 8 is the dependency net $DNet(d)$ according to Def. 4.1. The places encoding the write causality relation are "$p(1 <_{wc} 2)$" and "$p(3 <_{wc} 4)$". For the NAC-dependencies we have the places `p(1,N[2])` for the second instantiated NAC in the first transformation step of d

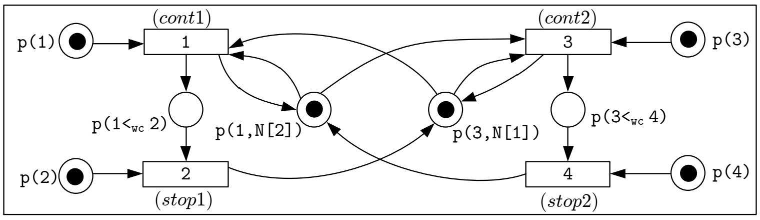

Fig. 8. Dependency Net *DNet*(d) as Petri Net

and `p(3,N[1])` for the third transformation step and its first instantiated NAC. The other two instantiated NACs are not considered, because the corresponding rules are weakly self-disabling ($q <_{wdn[i]} q$). At the beginning, Transitions `1` and `2` (*cont*1 and *cont*2) are enabled. The firing sequences according to the transformation sequences d and d' in Figs. 2 and 3 can be executed and they are the only complete firing sequences of this net. Thus, the net specifies exactly the transformation sequences which are permutation-equivalent to d.

We now show that we can exploit the constructed Petri net *DNet*(d) to characterize STS-equivalence of sequences of rule occurrences by Thm. 3. Note that according to Def. 4.1 each sequence s of rule names in the STS of *Prc*(d) can be interpreted as a sequence of transitions in the derived marked Petri net *DNet*(d), and vice versa. This correspondence allows us to transfer the results of the analysis of the dependency net back to the STS. Notice that the construction of the dependency net (Def. 4.1) ensures that each transition can fire at most once by construction.

Definition 4.3 (Transition Complete Firing Sequences). A firing sequence of a Petri net is called *transition complete*, if each transition of the net occurs exactly once. The set of transition complete firing sequences of a dependency net *DNet*(d) is denoted by *FSeq*(*DNet*(d)).

Theorem 3 (Characterization of STS-Equivalence Based on Petri nets). Given the process *Prc*(d) and the dependency net *DNet*(d) of a NAC-consistent transformation sequence d of an $\mathcal{M}$-adhesive transformation system with $\mathcal{M}$-matches. The class of STS-equivalent sequences of *seq*(d) coincides with the set of transition complete firing sequences in the dependency net *DNet*(d), i.e. $\mathit{Seq}(d) = \mathit{FSeq}(\mathit{DNet}(d))$.

Remark 4.4 (Bijective Correspondence). Analogous to Thm. 2, there is also a bijective correspondence between STS sequences and transition complete firing sequences, which is in this case directly given by the identity function $id : \mathit{Seq}(d) \xrightarrow{\sim} \mathit{FSeq}(\mathit{DNet}(d))$.

In order to prove Thm. 3 we first proof Fact 4.5, which shows that STS-switch equivalence disregarding NACs of rule sequences respects the partial order of the relations "$<_{rc}, <_{wc}$" and "$<_d$", and vice versa. This is important to show that the causal dependencies are correctly reflected within the dependency net, where firing sequences correspond to linearisations.

Fact 4.5 (Linearisation). Let d be a NAC-consistent transformation sequence of an $\mathcal{M}$-adhesive TS, let $\mathcal{S} = STS(d)$ be the generated STS of d, and let $s = \langle s_1, \ldots, s_n \rangle$ be a permutation of $seq(d)$. Then
$s \overset{sw}{\approx}_{\mathcal{S}} seq(d)$ if and only if $\forall\, i, j \in \{1, \ldots, n\},\ x \in \{rc, wc, d\} : s_i <_x s_j \Rightarrow i < j$.

Proof of Thm. 3 Let $Prc(d) = (STS(d), \mu)$ and $\mathcal{S} = STS(d)$, we have to show that $s \approx_{STS(d)} seq(d)$ **iff** s is a transition complete firing sequence of $DNet(d)$. Let $seq(d) = \langle q_1, \ldots, q_n \rangle$ and $s = \langle s_1, \ldots, s_n \rangle$.

Direction "⇒": By Def. 3.19 s is a legal sequence with respect to d in $STS(d)$. We show that s is a transition complete firing sequence of $DNet(d)$. Since s is a permutation of $seq(d)$ in $STS(d)$ we know $(*)$: each transition occurs exactly once in s. Consider the transition name $tr = s_m$ in s and the claimed firing step $M_m \xrightarrow{tr} M_{m+1}$. We check the activation of tr in M_m, i.e. $M_m \geq \mathrm{pre}(tr)$ according to Def. 4.1. Now, let $\mathrm{pre}(tr) = \sum_{pl \in PL} \lambda_{pl} \cdot pl$. For each pl we have:

— **case** $pl = \mathtt{p}(q)$**:** this implies that $tr = q$ and $\lambda_{pl} = 1$. By definition this place is initially marked with one token and there is no other transition connected to this place. Since each transition occurs exactly once in s $(*)$ this token is available in M_m.

— **case** $pl = \mathtt{p}(q\mathtt{<}_x q')$, $x \in \{rc, wc, d\}$**:** this implies that $tr = q'$ and $\lambda_{pl} = 1$. By Def. 4.1 we then have $\mathrm{post}(q) \geq pl$ and pl is not in the pre domain of any other transition than $tr = q'$. By Fact 4.5 we have that q occurs before q' in s and by $(*)$ we know that q' was not fired already. Thus, $M_m \geq pl$.

— **case** $pl = \mathtt{p}(q,\ \mathtt{N}[i])$**:** For the initial marking M we know by Def. 4.1 that $M \geq d \cdot pl$ with d being the amount of weak disabling causes, i.e. $d = |DC|, DC = \{q_l \mid q, q' \in P, q <_{wdn[i]} q_l\}$. Moreover, by Def. 4.1 we know that $q \not<_{wdn[i]} q$.

1. **case** $q \neq tr$**:** Let $q' = tr$. By Def. 4.1 we have that $\lambda_{pl} = 1$ and $q <_{wdn[i]} q'$. The only transition tr' in $TR \backslash DC$ with $\mathrm{pre}(tr') \geq pl$ is q and q consumes and produces one token. Each of the transitions in DC consumes exactly one token and in sum they consume exactly d tokens and each transition occurs exactly once in s $(*)$. Therefore, $M_m \geq pl$, because $tr = q'$ was not fired already according to $(*)$.
2. **case** $q = tr$**:** Thus, $\lambda_{pl} = 1$. Let $s_k = q$, i.e., q occurs in s at position k. By Def. 3.18 there is one preceding rule occurrence $q' = s_e$ in s with $q' = s_e <_{wen[i]} s_k = q$ or there is one subsequent rule occurrence $q' = s_l$ in s with $q = s_k <_{wdn[i]} s_l = q'$ (because $q \not<_{wdn[i]} q$). Using $(*)$, this means that for the first case: $M_m \geq d \cdot pl + 1 - d \cdot pl = pl$ and for the second case: $M_m \geq d \cdot pl - (d-1)pl = pl$.

Direction "⇐": Assume that s is a transition complete firing sequence of $DNet(d)$. We show that s is a legal sequence with respect to d in $STS(d)$. First of all, s is a transition complete firing sequence implies that each transition tr occurs exactly once. We show that the two conditions in Def. 3.18 hold:

— condition 1: $s \overset{sw}{\approx}_{\mathcal{S}} seq(d)$
By Fact 4.5 this condition is equivalent to
$(*)$: $\forall\, i, j \in \{1, \ldots, n\},\ x \in \{rc, wc, d\} : s_i <_x s_j \Rightarrow i < j$. According to Def. 4.1 there is exactly one initially unmarked place $pl = \mathtt{p}(q\mathtt{<}_x q')$ for each pair (q, q') with $q <_x q', x \in \{rc, wc, d\}$. This implies that for $s_i = q$ and $s_j = q'$ the transition

s_i produces exactly one token and s_j consumes exactly one token from this place and there is no other transition connected to this place. Therefore, the condition is ensured, because transition s_j is not activated before s_i has been fired.

— condition 2: $\forall$ *NAC*s $\mathbf{N}_k[i]$ of $s_m = q_k$: $\begin{pmatrix} \exists\ e \in \{1, \ldots, m-1\} : & s_e <_{wen[i]} s_m \text{ or} \\ \exists\ l \in \{m, \ldots, n\} : & s_m <_{wdn[i]} s_l. \end{pmatrix}$

Consider a NAC $\mathbf{N}_k[i]$ of $q_k = s_m$.

1 case $q_k <_{wdn[i]} q_k$: Thus, we have $l = m$ for the above condition.

2 case $q_k \not<_{wdn[i]} q_k$: Thus, there is the place $\texttt{p(}k\texttt{,N[}i\texttt{])}$, such that the transition $s_m = q_k$ consumes exactly one token from that place. Consider the firing step $M_m \xrightarrow{s_m} M_{m+1}$ according to s. Since $s_m = q_k$ has fired according to this step there was a token on $\texttt{p(}k\texttt{,N[}i\texttt{])}$ in the marking M_m. The initial marking contains d tokens for this place, where d is the amount of weak disabling causes, i.e. $d = |DC|, DC = \{q_{l'} \mid q_k <_{wdn[i]} q_{l'}\}$. Let $EC = \{q_{e'} \mid q_{e'} <_{wen[i]} q_k\}$ be the set of weak enabling causes of q_k for $\mathbf{N}_k[i]$. Assume that condition 2 of Def. 3.18 does not hold. We then have that all $q_{l'}$ in DC occur before q_k in s and there is no $q_{e'}$ in EC that occurs before q_k in s. This implies that each transition of DC has consumed a token from $\texttt{p(}k\texttt{,N[}i\texttt{])}$ and none of the transitions that precede q_k have produced a token on this place. Therefore, there is no token left on $\texttt{p(}k\texttt{,N[}i\texttt{])}$, which is a contradiction to the firing of $s_m = q_k$ and thus, condition 2 holds. □

In order to solve the challenge of computing the set of all permutation-equivalent transformation sequences for a given one, we can now combine the presented results leading to our forth main result by Thm. 4 below, where we show that the analysis of permutation equivalence can be completely performed on the dependency net $DNet(d)$.

Theorem 4 (Analysis of Permutation Equivalence Based on Petri Nets). Given the process $Prc(d)$ and the dependency net $DNet(d)$ of a NAC-consistent transformation sequence d.

1 The class of permutation-equivalent transformation sequences of d coincides with the set of derived transformation sequences using $DNet(d)$:
$\pi\text{-}Equ(d) = Trafo(FSeq(DNet(d)))$

2 The mapping *Trafo* according to Def. 3.19 defines a bijective correspondence between transition complete firing sequences and permutation-equivalent transformation sequences:
$Trafo : FSeq(DNet(d)) \xrightarrow{\sim} (\pi\text{-}Equ(d))/_{\cong}$

Proof. By combining the characterisations of Thms. 2 and 3 we derive the equality $\pi\text{-}Equ(d) = Trafo(FSeq(DNet(d)))$ and the bijection $Trafo : FSeq(DNet(d)) \xrightarrow{\sim} (\pi\text{-}Equ(d))/_{\cong}$ is given by $Trafo : Seq(d) \xrightarrow{\sim} (\pi\text{-}Equ(d))/_{\cong}$ of Thm. 2 with $Seq(d) = FSeq(DNet(d))$ in Thm. 3. □

Remark 4.6 (Analysis of Permutation Equivalence). We now describe how the presented results can be used for an efficient analysis of permutation equivalence, i.e. for the generation of the complete set of permutation equivalent transformation sequences for a given one and for checking permutation equivalence of specific ones. Given a NAC-consistent transformation sequence with general matches and NAC-schemata we can first reduce the analysis problem to the derived instantiated transformation sequence with $\mathcal{M}$-matches and standard NACs according to Thm. 1 and Rem. 2.22. According to Thm. 4, we can perform the analysis of permutation equivalence based on Petri nets by first constructing the dependency net $DNet(d)$. For the generation of all permutation-equivalent sequences we construct the complete reachability graph of $DNet(d)$, where each path specifies one permutation-equivalent transformation sequence up to isomorphism. If only specific reorderings of the transformation steps shall be checked, then the corresponding firing sequences are checked to be executable in $DNet(d)$.

The dependency net $DNet(d)$ is a compact representation of the equivalence class $\pi\text{-}Equ(d)$ specified by the process of a transformation sequence d. Moreover, the analysis of permutation equivalence based on the dependency net shows significant advantages with respect to efficiency as shown in Rem. 2.20.

5. Related Work

Transformation systems based on the double pushout (DPO) approach (Rozenberg1997; Ehrig et al.1999) with negative application conditions (NACs) (Habel et al.1996; Ehrig et al.2006) are a suitable modelling framework for several application domains in the area of distributed and concurrent systems. The behaviour of these systems is formalized by an operational semantics given by a graph transformation system, where each transformation rule can simulate a step of the modelled system.

Processes of graph transformation systems based on the DPO approach are introduced in (Corradini et al.1996) and characterized as occurrence grammars in (Baldan2000). These concepts generalise the notion of processes available for the classical formal models in this context – namely Petri nets (Reisig1985) – which can be completely defined by restricted graph transformation systems (GTSs), while general GTSs are more expressive. In (Baldan et al.2006), processes of graph transformation systems are lifted to the abstract setting of adhesive rewriting systems in order to generalise the process construction and analysis. This way, the analysis techniques can be instantiated for transformation systems based on arbitrary adhesive categories (Lack and Sobocinski2005), such as typed graphs, graphs with scopes and graphs with second order edges.

The concept and analysis of processes of transformation systems with negative application conditions (NACs) is more complex. As presented by the case study, the notion of switch equivalence with NACs is in general too strict for analysing equivalence of transformation sequences in this context. The reason is that switch equivalence with NACs based on sequential independence of transformation sequences with NACs (Habel et al.1996; Lambers et al.2008; Lambers2009) does not relate all transformation sequences which are intuitively equivalent, i.e. those that are switch equivalent without considering the NACs and additionally satisfy the NACs of each applied rule. The reason is that several switchings of NAC-dependent steps have to be performed in order to derive a new transformation sequence that is again NAC-consistent, while the derived transformation sequences in between are not NAC-consistent. For this reason, we introduced the notion of permutation equivalence, which is by definition exactly the required equivalence relation as described above.

Moreover, we extended the construction and analysis of processes to the more general framework of $\mathcal{M}$-adhesive transformation systems, such that the important category of typed attributed graphs is included. Furthermore, the developed techniques are extended to transformation sequences with general match morphisms, i.e. matches do not have to be injective as in the cases before. The provided process construction is based on the notion of subobject transformation systems (STSs), which generalises the concept of elementary Petri nets (Rozenberg and Engelfriet1996) being the class of process models for P/T Petri net processes, such that STSs form the class of process models for arbitrary $\mathcal{M}$-adhesive transformation systems. As pointed out in Sec. 2.1, the concept of extremal $\mathcal{E}$-$\mathcal{M}$-factorization has been introduced in (Braatz et al.2010), which is important for NAC-consistency of our new concept of NAC-schemata for transformation rules. Furthermore, several results concerning finite objects and finitary categories in (Braatz et al.2010) will be useful for further application domains and application scenarios.

Some of the problems addressed in this paper are similar to those considered in the process semantics (Kleijn and Koutny2004) and unfolding (Baldan2000; Braatz et al.2004) of Petri nets with inhibitor arcs, and actually we could have used some sort of inhibitor arcs to model the inhibiting effect of NACs in the process skeleton of a derivation. However, we would have needed some kind of “generalised” inhibitor nets, where a transition is connected to several (inhibiting) places and can fire if at least one of them is unmarked. To avoid the burden of introducing yet another model of nets, we preferred to stick to a direct encoding of the process of a derivation into a standard marked P/T nets.

6. Conclusion

In this paper, we introduce the concept of permutation equivalence for transformation systems with negative application conditions (NACs) in $\mathcal{M}$-adhesive categories. Permutation equivalence generalises switch equivalence with NACs and has interesting applications in the area of business processes (Brandt et al.2009). Formally, we are able to define processes of $\mathcal{M}$-adhesive transformation systems based on subobject transformation systems inspired by processes for Petri nets (Rozenberg and Engelfriet1996) and adhesive rewriting systems (Baldan et al.2006).

In our main results we show that processes can be represented by equivalence classes of permutation-equivalent transformation sequences. Moreover, they can be analysed efficiently by complete firing sequences of a Petri net, which can be constructed effectively as a dependency net of a given transformation sequence. Most constructions and results are illustrated by a case study of a typed attributed graph transformation system using the new concept of NAC-schemata. Tool support for the analysis is available by the tool AGT-M (Hermann et al.2010; Brandt et al.2009) based on Wolfram Mathematica and provides the construction of the STS, the dependency net and the generation of the reachability graph for a given transformation sequence.

Future work will include the study of non-deterministic processes of transformation systems with NACs, which will be based on incomplete firings of the constructed P/T Petri net and suitable side conditions. Furthermore, the notion of permutation equivalence can be extended to the more general case of nested application conditions (Habel and Pennemann2009) leading probably to an extended concept for processes based on STSs including nested application conditions. Further improvements of efficiency could be obtained by observing the occurring symmetries in the P/T Petri net, and applying symmetry reduction techniques on it. Additionally, the space complexity of the analysis could be reduced by unfolding the net and then representing all permutation-equivalent derivations in a more compact, partially ordered structure.

References

Baldan, P. (2000). *Modelling Concurrent Computations: from Contextual Petri Nets to Graph Grammars.* PhD thesis, Computer Science Department - University of Pisa.

Baldan, P., Corradini, A., Heindel, T., König, B., and Sobocinski, P. (2006). Processes for Adhesive Rewriting Systems. In Aceto, L. and Ingólfsdóttir, A., editors, *Proc. FoSSaCS'06*, volume 3921 of *LNCS*, pages 202–216. Springer.

Baldan, P., Corradini, A., Montanari, U., Rossi, F., Ehrig, H., and Löwe, M. (1999). Concurrent Semantics of Algebraic Graph Transformations. In Rozenberg, G., editor, *The Handbook of Graph Grammars and Computing by Graph Transformations, Volume 3: Concurrency, Parallelism and Distribution*, pages 107–188. World Scientific.

Braatz, B., Ehrig, H., Gabriel, K., and Golas, U. (2010). Finitary $\mathcal{M}$-Adhesive Categories. In Ehrig, H., Rensink, A., Rozenberg, G., and Schürr, A., editors, *Proc. ICGT'10*, volume 6372 of *LNCS*, pages 234–249. Springer.

Braatz, B., Klein, M., and Schröter, G. (2004). Semantical Integration of Object-Oriented Viewpoint Specification Techniques. In *Integration of Software Specification Techniques forApplications in Engineering*, volume 3147 of *LNCS*, pages 602–626. Springer.

Brandt, C., Hermann, F., and Engel, T. (2009). Modeling and Reconfiguration of critical Business Processes for the purpose of a Business Continuity Management respecting Security, Risk and Compliance requirements at Credit Suisse using Algebraic Graph Transformation. In *Proc. Dynamic and Declarative Business Processes (DDBP 2009)*, pages 64–71. IEEE Xplore Digital Library.

Corradini, A., Hermann, F., and Sobociński, P. (2008). Subobject Transformation Systems. *Applied Categorical Structures*, 16(3):389–419.

Corradini, A., Montanari, U., and Rossi, F. (1996). Graph processes. *Fundamenta Informaticae*, 26(3/4):241–265.

Ehrig, H., Ehrig, K., Prange, U., and Taentzer, G. (2006). *Fundamentals of Algebraic Graph Transformation.* EATCS Monographs in Theor. Comp. Science. Springer.

Ehrig, H., Golas, U., and Hermann, F. (2010). Categorical Frameworks for Graph Transformation and HLR Systems based on the DPO Approach. *Bulletin of the EATCS*, 102:111–121.

Ehrig, H., Kreowski, H.-J., Montanari, U., and Rozenberg, G., editors (1999). *Handbook of Graph Grammars and Computing by Graph Transformation. Vol 3: Concurrency, Parallelism and Distribution.* World Scientific.

Ehrig, H., Pfender, M., and Schneider, H. (1973). Graph-grammars: an algebraic approach. In Book, R., editor, *Switching and Automata Theory*, pages 167–180. IEEE Computer Society Press.

Habel, A., Heckel, R., and Taentzer, G. (1996). Graph Grammars with Negative Application Conditions. *Fundamenta Informaticae*, 26(3,4):287–313.

Habel, A. and Pennemann, K.-H. (2009). Correctness of high-level transformation systems relative to nested conditions. *Mathematical Structures in Computer Science*, 19(2):245–296.

Heindel, T. (2010). Hereditary Pushouts Reconsidered. In Ehrig, H., Rensink, A., Rozenberg, G., and Schürr, A., editors, *Proc. ICGT'10*, volume 6372 of *LNCS*, pages 250–265. Springer.

Hermann, F. (2009). Permutation Equivalence of DPO Derivations with Negative Application Conditions based on Subobject Transformation Systems. *Electronic Communications of the EASST*, 16.

Hermann, F., Corradini, A., Ehrig, H., and König, B. (2010). Efficient Analysis of Permutation Equivalence of Graph Derivations Based on Petri Nets. *Electronic Communications of the EASST*, 29.

Kastenberg, H., Hermann, F., and Modica, T. (2006). Towards Translating Graph Transformation Systems by Model Transformation. *Electronic Communications of the EASST*, 4.

Kleijn, H. C. M. and Koutny, M. (2004). Process semantics of general inhibitor nets. *Information and Computation*, 190(1):18–69.

Lack, S. and Sobociński, P. (2004). Adhesive Categories. In *Proc. FOSSACS'04*, volume 2987 of *LNCS*, pages 273–288. Springer.

Lack, S. and Sobocinski, P. (2005). Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications*, 39(3):511–545.

Lambers, L. (2009). *Certifying Rule-Based Models using Graph Transformation.* PhD thesis, Technische Universität Berlin.

Lambers, L., Ehrig, H., Orejas, F., and Prange, U. (2008). Parallelism and Concurrency in Adhesive High-Level Replacement Systems with Negative Application Conditions. In *Proc. of the ACCAT workshop at ETAPS 2007*, volume 203/6 of *ENTCS*, pages 43–66. Elsevier.

Reisig, W. (1985). *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer.

Rozenberg, G. (1997). *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations.* World Scientific.

Rozenberg, G. and Engelfriet, J. (1996). Elementary Net Systems. In Reisig, W. and Rozenberg, G., editors, *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 12–121. Springer.

Appendix A. Category of Typed Attributed Graphs

In this appendix, we review the main constructions for the $\mathcal{M}$-adhesive category of typed attributed graphs ($\mathbf{AGraphs}_{ATG}, \mathcal{M}$) according to (Ehrig et al.2006). An attributed graph consists of an extended directed graph for the structural part – called E-graph – together with an algebra for the specification of the carrier sets of the value nodes. An E-graph extends a directed graph by additional attribute value nodes and edges for the attribution of structural nodes and edges.

Definition A.1 (E-graph and E-graph morphism). An *E-graph* G with $G = (V_G, V_D, E_G, E_{NA}, E_{EA}, (source_j, target_j)_{j \in \{G,NA,EA\}})$ consists of the sets

— V_G and V_D , called the graph and data nodes (or vertices), respectively;

— E_G, E_{NA} , and E_{EA} called the graph, node attribute, and edge attribute edges, respectively; and the source and target functions

— $source_G : E_G \to V_G, target_G : E_G \to V_G$ for graph edges;
— $source_{NA} : E_{NA} \to V_G, target_{NA} : E_{NA} \to V_D$ for node attribute edges; and
— $source_{EA} : E_{EA} \to E_G, target_{EA} : E_{EA} \to V_D$ for edge attribute edges:

$$\begin{tikzcd}
& E_G \arrow[rr, shift left, "source_G"] \arrow[rr, shift right, "target_G"'] & & V_G & \\
E_{EA} \arrow[ur, "source_{EA}"] \arrow[drr, "target_{EA}"'] & & & & E_{NA} \arrow[ul, "source_{NA}"'] \arrow[dll, "target_{NA}"] \\
& & V_D & &
\end{tikzcd}$$

Consider the E-graphs G^1 and G^2 with $G^k = (V_G^k, V_D^k, E_G^k, E_{NA}^k, E_{EA}^k, (source_j^k, target_j^k)_{j\in\{G,NA,EA\}})$ for $k = 1, 2$. An E-graph morphism $f : G^1 \to G^2$ is a tuple $(f_{V_G}, f_{V_D}, f_{E_G}, f_{E_{NA}}, f_{E_{EA}})$ with $f_{V_i} : V_i^1 \to V_i^2$ and $f_{E_j} : E_j^1 \to E_j^2$ for $i \in \{G, D\}$, $j \in \{G, NA, EA\}$ such that f commutes with all source and target functions, for example $f_{V_G} \circ source_G^1 = source_G^2 \circ f_{E_G}$.

The carrier sets of attribute values that form the single set V_D of an E-graph are defined by an additional data algebra D, which also specifies the operations for generating and manipulating data values. The carrier sets D_s of D contain the data elements for each sort $s \in S$ according to a data signature $DSIG = (S_D, OP_D)$. These carrier sets are combined by disjoint union and form the set V_D of data elements.

Definition A.2 (Attributed Graph and Attributed Graph Morphism). Let $DSIG = (S_D, OP_D)$ be a data signature with attribute value sorts $S'_D \subseteq S_D$. An attributed graph $AG = (G, D)$ consists of an E-graph G together with a $DSIG$-algebra D such that $\uplus_{s\in S'_D} D_S = V_D$. For two attributed graphs $AG^1 = (G^1, D^1)$ and $AG^2 = (G^2, D^2)$, an attributed graph morphism $f : AG^1 \to AG^2$ is a pair $f = (f_G, f_D)$ with an E-graph morphism $f_G : G^1 \to G^2$ and an algebra homomorphism $f_D : D^1 \to D^2$ such that (1) commutes for all $s \in S'_D$, where the vertical arrows are inclusions.

$$\begin{tikzcd}
D_s^1 \arrow[r, "f_{D,s}"] \arrow[d, hook] \arrow[dr, phantom, "(1)"] & D_s^2 \arrow[d, hook] \\
V_D^1 \arrow[r, "f_{G,V_D}"] & V_D^2
\end{tikzcd}$$

The category of typed attributed graphs $\mathbf{AGraphs}_{ATG}$ has as objects all attributed graphs with a *typing morphism* to the attributed graph ATG (type graph), and as arrows all attributed graph morphisms preserving the typing. The category $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$ is shown in (Ehrig et al.2006) to be an adhesive HLR category, where the distinguished class of monomorphisms $\mathcal{M}$ contains all monomorphisms that are isomorphisms on the data part. For this reason, all results for adhesive HLR transformation systems presented in (Ehrig et al.2006) are valid. Since $\mathcal{M}$-adhesive categories (Ehrig et al.2010) are a slight generalisation of weak adhesive and adhesive HLR categories the category $(\mathbf{AGraphs}_{ATG}, \mathcal{M})$ is an $\mathcal{M}$-adhesive category.

Appendix B. Proofs of Technical Results

In this section we provide proofs for Facts 2.14, 2.21, 3.3, 3.5, 3.14 and 4.5.

Fact 2.14 (Compatibility of Applicability and NAC-consistency with Instantiation (see Sec. 2.2)**).**

Proof. Without considering the NACs we have that the transformation step via p' can be composed with the diagrams (3) and (4) acc. to Def. 2.13 leading to a transformation step via p and match f_2. Vice versa, for a transformation step via p and match m_2 we can conclude that K' is isomorphic to the pullback of $(L' \rightarrowtail G_2 \leftarrowtail D_2)$ using the $\mathcal{M}$ pushout-pullback lemma (item 2 of Thm. 4.26 in (Ehrig et al.2006)) and uniqueness of pushout complements for rules in $\mathcal{M}$-adhesive transformation systems and we derive pushouts (3) and (5). The comatch m_2^* of the instantiated rule is induced by pushout (4). Finally, (6) is a pushout by pushout decomposition. We now consider the NACs and a transformation diagram with step $G_2 \xRightarrow{p',m_2} H_2$. For a NAC-schema $n \in \mathbf{N}_S$ we have by Def. 2.10 for the satisfaction of NAC-schemata that a NAC occurrence $q' : N' \rightarrowtail G_2$ of the instantiated rule p' defines a NAC occurrence of $n \in \mathbf{N}_S$ and vice versa, a violation of $n \in \mathbf{N}_S$ induces a NAC occurrence $q' : N' \rightarrowtail G_2$ of the instantiated rule p'. $\square$

$$\begin{tikzcd}
L \arrow[d, "e"'] \arrow[dd, bend left, "f_2"] \arrow[dr, phantom, "(3)"] & K \arrow[l, tail] \arrow[r, tail] \arrow[d, "k_e"'] \arrow[dd, bend left] \arrow[dr, phantom, "(4)"] & R \arrow[d, "e^*"'] \arrow[dd, bend left, "f_2^*"] \\
L' \arrow[d, tail, "m_2"'] \arrow[dr, phantom, "(5)"] & K' \arrow[l, tail] \arrow[r, tail] \arrow[d, tail] \arrow[dr, phantom, "(6)"] & R' \arrow[d, tail, "m_2^*"'] \\
G_2 & D_2 \arrow[l, tail] \arrow[r, tail] & H_2
\end{tikzcd}$$

Fact 2.21 (Sequential Independence disregarding NACs for Instantiated Steps (see Sec. 2.3)**).**

Proof. First of all, a mediating morphism $j' : L' \to D_1$ of the instantiated DPO diagrams directly induces a mediating morphism $j : L_2 \to D_1$ for the original DPO diagrams by $j = j' \circ e_2$. The case of $i' : R' \to D_2$ is dual. Now, given a mediating morphism $j : L_2 \to D_1$ we show that there is a mediating morphism $j' : L' \to D_1$ for the instantiated DPO diagram. The dual case with morphism $i : R_1 \to D_2$ is again analogous. By Def. 2.13 we have the extremal $\mathcal{E}$-$\mathcal{M}$ factorization $f_2 = e_2 \circ m_2$. Now, we construct the extremal $\mathcal{E}$-$\mathcal{M}$ factorization $j = j_1 \circ j_e : L_2 \twoheadrightarrow L'' \rightarrowtail D_1$. By uniqueness of extremal $\mathcal{E}$-$\mathcal{M}$ factorizations and commutativity $g_1 \circ j = f_2$ we have that $L'' \cong L'$ via iso and $m_2 = g_1 \circ j_1 \circ iso$. Therefore, $j_1 \circ iso : L' \to D_1$ is compatible with m_2, i.e. $m_2 = g_1 \circ j_1 \circ iso$. $\square$

$$\begin{tikzcd}
K_1 \arrow[r, tail] \arrow[dd] & R_1 & & L_2 \arrow[dll, two heads, "j_e"'] \arrow[dl, two heads, "e_2"'] \arrow[ddl, "f_2"] \arrow[ddlll, dotted, bend right, "j"'] & K_2 \arrow[l, tail] \arrow[dd] \\
& L'' \arrow[dl, tail, "j_1"] & L' \arrow[l, "\sim"'] \arrow[d, tail, "m_2"'] & & \\
D_1 \arrow[rr, tail, "g_1"] & & G_1 & & D_2 \arrow[ll, tail]
\end{tikzcd}$$

Fact 3.3 (Intersection in $\mathbf{Sub}_{\mathcal{M}}(T)$ (see Sec. 3.1)**).**

Proof. Let $A, B \in |\mathbf{Sub}_{\mathcal{M}}(T)|$ and construct pullback (1) in $\mathbf{C}$ using $a \in \mathcal{M}$ leading to $\mathcal{M}$-morphisms p_A and p_B, because $a, b \in \mathcal{M}$. Furthermore, p_A, p_B are morphisms in $\mathbf{Sub}_{\mathcal{M}}(T)$ by commutativity of the pullback. Now, a comparison object X for the product $A \cap B$ in $\mathbf{Sub}_{\mathcal{M}}(T)$ is also a comparison object for the pullback $A \cap B$ in $\mathbf{C}$. Thus, there is a unique morphism h satisfying the universal property. Furthermore, $h \in \mathcal{M}$ by decomposition of x_1 and h is a morphism in $\mathbf{Sub}_{\mathcal{M}}(T)$ by the commutativity of the diagram on the right. $\square$

$$\begin{tikzcd}
& X \arrow[dl, "x_1"'] \arrow[d, dotted, "h"] \arrow[dr, "x_2"] \arrow[dd, bend left=60, "x"] & \\
A \arrow[dr, hook, "a"'] & A \cap B \arrow[l, "p_A"'] \arrow[r, "p_B"] \arrow[d, "i"] & B \arrow[dl, hook, "b"] \\
& T &
\end{tikzcd}$$

Fact 3.5 (Distributivity(see Sec. 3.1)**).**

Proof. Property (i) : The proof is analogous to the one for Cor. 5.2 in (Lack and Sobocinski2005) concerning adhesive categories and we lift it to $\mathcal{M}$-adhesive categories. Let $A, B, C \in |\mathbf{Sub}_{\mathcal{M}}(T)|$, then (1) is pushout in $\mathbf{C}$ by the general assumption that $\mathbf{C}$ has effective unions. The cube is commutative, because all diagrams in $\mathbf{Sub}_{\mathcal{M}}(T)$ commute and $A \cap C \subseteq A \cap (B \cup C)$, because $C \subseteq B \cup C$. The bottom face is a pushout in $\mathbf{C}$ along an $\mathcal{M}$-morphism, because $\mathbf{C}$ has effective unions. The back faces are pullbacks in $\mathbf{C}$ according to Fact 3.3. The front left face of the cube is a pullback by pullback decomposition of the pullback $(2+3)$. For the analogous reason, the front right face of the cube is a pullback. By the VK-property of $\mathcal{M}$-adhesive categories we derive that the top face of the cube is a pushout and by uniqueness of pushouts we deduce property (i) and by duality in lattices we also have property (ii). $\square$

$$
\begin{tikzcd}
 & & A \cap B \cap C \arrow[dll] \arrow[dr] & \\
A \cap B \arrow[dr] & & (1) & A \cap C \arrow[dll] \\
 & (A \cap B) \cup (A \cap C) & &
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & & A \cap B \cap C \arrow[dll] \arrow[dr] \arrow[ddd, dotted] & \\
A \cap B \arrow[dr] \arrow[ddd] & & & A \cap C \arrow[dll] \arrow[ddd] \\
 & A \cap (B \cup C) \arrow[ddd] & & \\
 & & B \cap C \arrow[dll, dotted] \arrow[dr, dotted] & \\
B \arrow[dr] & & & C \arrow[dll] \\
 & B \cup C & &
\end{tikzcd}
$$

$$
\begin{tikzcd}
A \cap B \arrow[r] \arrow[d] \arrow[dr, phantom, "(2)"] & A \cap (B \cup C) \arrow[r] \arrow[d] \arrow[dr, phantom, "(3)"] & A \arrow[d] \\
B \arrow[r] & B \cup C \arrow[r] & A \cup B \cup C
\end{tikzcd}
$$

Fact 3.14 (Correspondence between Transformation and Rule Sequences (see Sec. 3.2)**).**

Proof. By Prop. 6 in (Corradini et al.2008) we have that *trafo* is well-defined for adhesive transformation systems. The proof is based on the characterisation of intersection and union by pullback and pushout in the underlying category and distributivity, which we are ensured for $\mathcal{M}$-adhesive transformation systems with effective unions, which we require by our general assumption in Sec. 2.1. By Fact 2.14 we know that the derived DPO diagrams can be composed with the upper DPO diagrams of the instantiation (Def. 2.13). The equations hold, because of the above results and each step i in d is specified by a distinguished rule $q_i = i$ in $Prc(d)$, i.e. *trafo* and *seq* are left unique relations. $\square$

Fact 4.5 (Linearisation (see Sec. 4)**).**

Proof. Let $(*): \ \forall\, i, j \in \{1, \ldots, n\},\ x \in \{rc, wc, d\} : s_i <_x s_j \Rightarrow i < j$.

Direction "⇒": Let $s \overset{sw}{\approx}_{\mathcal{S}} seq(d)$ and $seq(d) = \langle q_1, \ldots, q_n \rangle$. We show that $(*)$ holds.

— We first show the property for $s = seq(d)$, i.e.
 $(**): \ \forall\, i, j \in \{1, \ldots, n\}, x \in \{rc, wc, d\} : q_i <_x q_j \Rightarrow i < j$.
 $\Leftrightarrow \ \forall\, i, j \in \{1, \ldots, n\}, x \in \{rc, wc, d\} : i \geq j \Rightarrow q_i \not<_x q_j$.
 Let $\pi(q_i) = (\langle L_i, K_i, R_i \rangle, \mathbf{N}_i)$ and $\pi(q_j) = (\langle L_j, K_j, R_j \rangle, \mathbf{N}_j)$.
 For $i = j$ the condition is fulfilled directly, because $\forall\, k \in \{1, \ldots, n\} : L_k \cap R_k = K_k$ according to Prop. 30 in (Corradini et al.2008), where the proof can be directly lifted to the case of $\mathcal{M}$-adhesive categories via the provided results (constructions

intersection and union as well as distributivity and VK-property for the case that all morphisms are in $\mathcal{M}$).
Now, consider $i > j$.

- Case $x = rc$:
  By definition we have that $q_i \not<_{rc} q_j \Leftrightarrow R_i \cap K_j \subseteq K_i$.
  We can build up the colimit of the instantiated transformation sequence d_I of d (see Def. 2.13) by stepwise pushouts. Let T_{i-1} be the colimit of the steps $d_1, \ldots, d_{i-1}$. Then we have that $(1) : K_j \subseteq T_{i-1}$. Let T'_i be the colimit of transformation step d_i, and therefore, T'_i is given by the pushout (2) of $G_{i-1} \leftarrow D_i \rightarrow G_i$. We perform a pushout (3) of T_{i-1} and T'_i and obtain T_i. We compose the pushouts (2) and (3) with the pushout $(4) : D_i \leftarrow K_i \rightarrow R_i \rightarrow G_i$ of the transformation step d_i. This is also a pullback and thus, $R_i \cap T_{i-1} \cong K_i$. Using (1) this implies $R_i \cap K_j \subseteq K_i$.
- Case $x = wc$:
  By definition we have that $q_i \not<_{wc} q_j \Leftrightarrow R_i \cap L_j \subseteq K_i \cup K_j$.
  Considering the construction from before, we additionally derive $L_j \subseteq T_{i-1}$ and thus, the equation holds.
- Case $x = d$:
  By definition we have that $q_i \not<_{wc} q_j \Leftrightarrow K_i \cap L_j \subseteq K_j$.
  Considering the construction from before, we can additionally compose the pushout $(5) : D_j \leftarrow K_j \rightarrow L_j \rightarrow G_{j-1}$ of the transformation step d_j with the pushouts of the stepwise construction of T_{i-1} and finally derive $L_j \cap T_{i-1} \cong K_j$. Furthermore, we have $K_i \subseteq T_{i-1}$ from (1) and thus, the above equation holds.

— We now show that the condition $(*)$ holds for every sequence s that is STS-switch-equivalent to $seq(d)$ disregarding NACs. By $(**)$ we know that the condition holds for $seq(d)$. Furthermore, each sequence s is derived from $seq(d)$ by switchings according to $\stackrel{sw}{\approx}_{\mathcal{S}}$. It remains to show that each switching preserves the condition $(*)$. Now, STS-switch equivalence of sequences $\stackrel{sw}{\approx}_{\mathcal{S}}$ is based on $(q_i \diamondsuit q_j)$, which is equivalent to $(q_i \not<_{rc} q_j \wedge q_i \not<_{wc} q_j \wedge q_i \not<_{d} q_j)$ according to Thm. 32.2 in (Corradini et al.2008). Thus, the condition is not affected by any switching.

Direction "$\Leftarrow$": By contraposition we show $\neg(s \stackrel{sw}{\approx}_{\mathcal{S}} seq(d)) \Rightarrow \neg(*)$. Since s is a permutation of $seq(d)$ the condition $\neg(s \stackrel{sw}{\approx}_{\mathcal{S}} seq(d))$ means that s can be derived by switching neighbouring steps of $seq(d)$, where at least on switching is performed on a pair $(q_i; q_j)$ of steps that is dependent, i.e. $\neg(q_i \diamondsuit q_j)$, which is equivalent to $(q_i <_x q_j)$ for one or more $x \in \{rc, wc, d\}$ according to Thm. 32.2 in (Corradini et al.2008) as above. Thus, this pair would violate the condition $(*)$ in the new order. Since s is assumed to be not STS-switch equivalent to $seq(d)$ there is at least one such pair, where the final position of q_j is in front of q_i in s. □