Deriving Bisimulation Congruences in the Presence of Negative Application Conditions*

Guilherme Rangel[1], Barbara König[2], and Hartmut Ehrig[1]

[1] Institut für Softwaretechnik und Theoretische Informatik,
Technische Universität Berlin, Germany
{rangel,ehrig}@cs.tu-berlin.de
[2] Abteilung für Informatik und Angewandte Kognitionswissenschaft,
Universität Duisburg-Essen, Germany
barbara_koenig@uni-due.de

Abstract. In recent years there have been several approaches for the automatic derivation of labels from an unlabeled reactive system. This can be done in such a way that the resulting bisimilarity is automatically a congruence. One important aspect that has not been studied so far is the treatment of reduction rules with negative application conditions. That is, a rule may only be applied if certain patterns are absent in the vicinity of a left-hand side. Our goal in this paper is to extend the borrowed context framework to label derivation with negative application conditions and to show that bisimilarity remains a congruence. An important application area is graph transformation and we will present a small example in order to illustrate the theory.

1 Introduction

Bisimilarity is an equivalence relation on states of transition systems, associating states that can match each other's moves. In this sense, bisimilar states can not be distinguished by an external observer. Bisimilarity provides a powerful proof technique to analyze the properties of systems and has been extensively studied in the field of process calculi since the early 80's. Especially for CCS [1] and the π-calculus [2, 3] an extensive theory of bisimulation is now available.

Congruence is a very desirable property that a bisimilarity may have, since it allows the exchange of bisimilar systems in larger systems without effect on the observable behavior. Unfortunately, a bisimulation defined on unlabeled reaction rules is in general not a congruence. Hence, Leifer and Milner [4, 5] proposed a method that uses so-called idem pushouts (IPOs) to derive a labeled transition system from unlabeled reaction rules such that the resulting bisimilarity is a congruence. Motivated by this work, two of the authors proposed in [6, 7] an extension to the double pushout approach (DPO, for short) called DPO with borrowed contexts (DPO-BC), which provides the means to derive labeled transitions from rewriting rules in such a way that the bisimilarity is automatically

* Research partially supported by the DFG project SANDS and DAAD (German Academic Exchange Service).

a congruence. This has turned out to be equivalent to a technique by Sassone and Sobociński [8, 9] which derives labels via groupoidal idem pushouts. In all approaches the basic idea is the one suggested by Leifer and Milner: the labels should be the minimal contexts that an observer has to provide in order to trigger a reduction.

The DPO with borrowed contexts works with productions consisting of two arrows $L \leftarrow I \rightarrow R$ where the arrows are either graph morphisms, or—more generally—arrows in an adhesive category. Even though the generative power of the DPO approach is sufficient to generate any recursively enumerable set of graphs, very often extra application conditions are a required feature of non-trivial specifications. Negative application conditions (NACs) [10] for a graph production are conditions such as the non-existence of nodes, edges, or certain subgraphs in the graph G being rewritten, as well as embedding restrictions concerning the match $L \rightarrow G$. Similar restrictions can also be achieved in Petri nets with inhibitor arcs, where these arcs impose an extra requirement to transition firing, i.e., a transition can only be fired if some specific places are currently unmarked.

Graph transformation systems, which are our main focus, are often used for specification purposes, where—in contrast to programming—it is quite convenient and often necessary to constrain the applicability of rules by negative application conditions. We believe that this is a general feature of specification languages, which means that the problem of deriving behavioural equivalences in the presence of NACs may occur in many different settings.

In this work we extend the borrowed context framework to handle productions with negative application conditions. The extension, which is carried out for adhesive categories, requires an enrichment of the labels which now do not only indicate the context that is provided by the observer, but also constrain further additional contexts that may satisfy the negative application condition. That is, we do not only specify what must be borrowed, but also what must not be borrowed. We prove that the main result of [7] (bisimilarity is a congruence) still holds for our extension. Moreover, we further develop an up-to context technique in order to cope with NACs and apply it to an example.

The current paper is structured as follows. Section 2 briefly reviews the DPO approach with borrowed contexts. In Section 3 we discuss the problems which arise due to productions with NACs and how they can be overcome in order to guarantee that the derived bisimilarities are congruences. Section 4 presents the up-to proof method for our extension and finally an example in terms of graph transformation is shown in Section 5.

An extended example and the full proof with all lemmas can be found in a technical report [11].

2 Double-Pushout with Borrowed Contexts

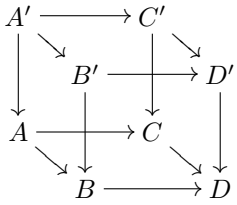
In this section we recall the DPO approach with borrowed contexts [6, 7]. In standard DPO [12], productions rewrite graphs with no interaction with any

other entity than the graph itself and the production. In the DPO with borrowed contexts [7] graphs have interfaces and may borrow missing parts of left-hand sides from the environment via the interface. This leads to open systems which take into account interaction with the outside world.

The DPO-BC framework was originally defined for the category of graph structures, but, as already stated in [6, 7], its results can be automatically lifted to adhesive categories since the corresponding proofs only use pushout and pullback constructions which are compliant with adhesive categories. In the following we present the DPO-BC setting for adhesive categories [13] to which we first give a short introduction.

Definition 1 (Adhesive Category). *A category* $\mathbf{C}$ *is called* adhesive *if*

1. $\mathbf{C}$ *has pushouts along monos;*
2. $\mathbf{C}$ *has pullbacks;*
3. *Given a cube diagram as shown on the right with: (i)* $A \to C$ *mono, (ii) the bottom square a pushout and (iii) the left and back squares pullbacks, we have that the top square is a pushout iff the front and right squares are pullbacks.*



Pullbacks preserve monos and pushouts preserve epis in any category. Furthermore, for adhesive categories it is known that monos are preserved by pushouts. For the DPO-BC extension to productions with negative application conditions, defined in Section 3, we need one further requirement, namely that pullbacks preserve epis. This means that if the square (A', B', A, B) above is a pullback and $A \to B$ is epi, we can conclude that $A' \to B'$ is epi as well.

Our prototypical instance of an adhesive category, which will be used for the examples in the paper are the categories of node-labeled and edge-labeled graphs, where arrows are graph morphisms. In this category pullbacks preserve epis.

We will now define the notion of objects with interfaces and contexts, followed by the definition of a rewriting step with borrowed contexts as defined in [7] and extended in [9].

Definition 2 (Objects with Interfaces and Contexts). *An* object G with interface J *is an arrow* $J \to G$ *and a context consists of two arrows* $J \to E \leftarrow \overline{J}$. *The* embedding[3] *of* $J \to G$ *into a context* $J \to E \leftarrow \overline{J}$ *is an object with interface* $\overline{J} \to \overline{G}$ *which is obtained by constructing* $\overline{G}$ *as the pushout of* $J \to G$ *and* $J \to E$.

$$\begin{array}{ccccc} J & \longrightarrow & E & \longleftarrow & \overline{J} \\ \downarrow & PO & \downarrow & \swarrow & \\ G & \longrightarrow & \overline{G} & & \end{array}$$

[3] The embedding is defined up to iso since the pushout object is unique up to iso. Embedding/insertion into a context and contextualization are used as synonyms.

Definition 3 (Rewriting with Borrowed Contexts). *Given an object with interface $J \to G$ and a production $p\colon L \leftarrow I \to R$, we say that $J \to G$ reduces to $K \to H$ with transition label*[4] *$J \to F \leftarrow K$ if there are objects D, G^+, C and additional arrows such that the diagram below commutes and the squares are either pushouts (PO) or pullbacks (PB) with monos. In this case a* rewriting step with borrowed context *(BC step) is called feasible:* $(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$.

$$\begin{array}{ccccc} D & \rightarrowtail & L & \longleftarrow & I & \longrightarrow & R \\ \downarrow & PO & \downarrow & PO & \downarrow & PO & \downarrow \\ G & \rightarrowtail & G^+ & \longleftarrow & C & \longrightarrow & H \\ \uparrow & PO & \uparrow & PB & \uparrow & \nearrow & \\ J & \rightarrowtail & F & \longleftarrow & K & & \end{array}$$

In the diagram above the upper left-hand square merges L and the object G to be rewritten according to a partial match $G \leftarrow D \to L$. The resulting object G^+ contains a total match of L and can be rewritten as in the standard DPO approach, producing the two remaining squares in the upper row. The pushout in the lower row gives us the borrowed (or minimal) context F, along with an arrow $J \to F$ indicating how F should be pasted to G. Finally, we need an interface for the resulting object H, which can be obtained by "intersecting" the borrowed context F and the object C via a pullback. Note that the two pushout complements that are needed in Definition 3, namely C and F, may not exist. In this case, the rewriting step is not feasible. The arrows depicted as $\to$ in the diagram above can also be non-mono (see [8]).

Note that with the procedure described above we may derive infinitely many labels of the form $J \to F \leftarrow K$. However, note that there are only finitely many up to iso and hence they can be represented in a finite way.

A bisimulation is an equivalence relation between states of transition systems, associating states which can simulate each other.

Definition 4 (Bisimulation and Bisimilarity). *Let $\mathcal{P}$ be a set of productions and $\mathcal{R}$ a symmetric relation containing pairs of objects with interfaces $(J \to G, J \to G')$. The relation $\mathcal{R}$ is called a* bisimulation *if, whenever we have $(J \to G)\,\mathcal{R}\,(J \to G')$ and a transition $(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$, then there exists an object with interface $K \to H'$ and a transition $(J \to G') \xrightarrow{J \to F \leftarrow K} (K \to H')$ such that $(K \to H)\,\mathcal{R}\,(K \to H')$.*

We write $(J \to G) \sim (J \to G')$ whenever there exists a bisimulation $\mathcal{R}$ that relates the two objects with interface. The relation $\sim$ is called bisimilarity.

Theorem 1 (Bisimilarity is a Congruence [7]). *The bisimilarity relation $\sim$ is a congruence, i.e., it is preserved by contextualization as described in Definition 2.*

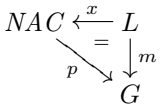
[4] Transition labels, derived labels and labels are synonyms in this work.

3 Borrowed Contexts with NACs

Here we will extend the DPO-BC framework of [7] to productions with negative application conditions. In order to simplify the theory and the presentation we will from now on require that productions and objects with interfaces consist of monos, which implies that all arrows in the diagram in Definition 3 are monos.

Prior to the extension we will investigate in Section 3.1 why such an extension is not trivial. It is worth emphasizing that the extension will be carried out for adhesive categories with an additional requirement that pullbacks preserve epis, but the examples will be given in the category of labeled directed graphs. First we define negative application conditions for productions.

Definition 5 (Negative Application Condition). *A negative application condition $NAC(x)$ on L is a mono $x\colon L \to NAC$. A mono $m\colon L \to G$ satisfies $NAC(x)$ on L if and only if there is no mono $p\colon NAC \to G$ with $p \circ x = m$.*

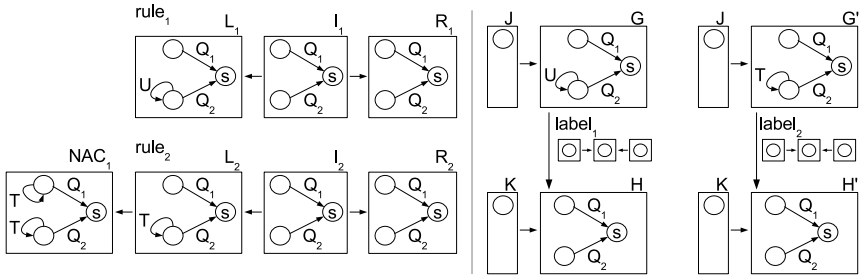


A rule $L \leftarrow I \rightarrow R$ with NACs is equipped with a finite set of negative application conditions $\{L \to NAC_y\}_{y \in Y}$ and is applicable to a match $m\colon L \to G$ only if all NACs are satisfied. If we add NACs to the rules in Definition 3, we have two ways to check their satisfiability: before (on G) or after the borrowing (on G^+), but the latter is more suitable since the first one does not take into account any borrowed structure.

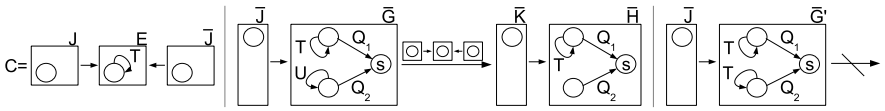
3.1 Bisimulation and NACs – Is Bisimilarity still a Congruence?

Let us assume that borrowed context rewriting works as in Definition 3 (with monos) if the total match $L \to G^+$ satisfies all NACs of a production, i.e., G^+ does not contain any prohibited structure (specified by a NAC) at the match of L. With the following example in terms of labeled directed graphs we will show that such a definition is unsuitable.

Below on the right we depict two servers as graphs with interfaces: $\mathsf{J} \to \mathsf{G}$ and $\mathsf{J} \to \mathsf{G}'$. An s-node represents a server. Each server has two queues Q_1 and Q_2 where it receives tasks to be processed. Tasks are modelled as loops and may either be standard (T) or urgent (U). In real world applications, standard tasks may come from regular users while urgent ones come from administrators. On the left we depict how the servers work. Rule_1 says that an urgent task in Q_2 must be immediately executed, whereas Rule_2 specifies how a standard task T in Q_2 is executed. The negative application condition NAC_1 allows rule_2 to be fired only when there is no other T-task waiting in the high priority queue Q_1. We consider that a processed task is consumed by the server (see R_1 and R_2).



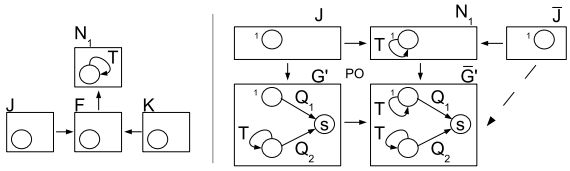
From the servers $\mathsf{J} \to \mathsf{G}$ and $\mathsf{J} \to \mathsf{G}'$ above we derive the labeled transition system (LTS) on the right w.r.t. rule_1 and rule_2. No further label can be derived from $\mathsf{K} \to \mathsf{H}$ and $\mathsf{K} \to \mathsf{H}'$ and the labels leading to these graphs are equal. By Definition 4 we can conclude that $(\mathsf{J} \to \mathsf{G}) \sim (\mathsf{J} \to \mathsf{G}')$. Since bisimilarity is a congruence (at least for rules without NACs), the insertion of $\mathsf{J} \to \mathsf{G}$ and $\mathsf{J} \to \mathsf{G}'$ into a context C, as in Definition 2, produces graphs $\overline{\mathsf{J}} \to \overline{\mathsf{G}}$ and $\overline{\mathsf{J}} \to \overline{\mathsf{G}}'$ respectively, which should be bisimilar. Below we show a context C with a standard task, the resulting graphs $\overline{\mathsf{J}} \to \overline{\mathsf{G}}$ and $\overline{\mathsf{J}} \to \overline{\mathsf{G}}'$ which received the T-task in queue Q_1 via the interface J, and their LTS. The server $\overline{\mathsf{J}} \to \overline{\mathsf{G}}'$ cannot perform any transition since NAC_1 of rule_2 forbids the BC step, i.e., the T-task in Q_2 cannot be executed because there is another standard task in the high priority queue Q_1. However, $\overline{\mathsf{J}} \to \overline{\mathsf{G}}$ is still able to perform a transition and evolve to $\overline{\mathsf{K}} \to \overline{\mathsf{H}}$. Thus, bisimilarity is no longer a congruence when productions have NACs.



The LTS for $\mathsf{J} \to \mathsf{G}$ and $\mathsf{J} \to \mathsf{G}'$ shows that label_1, which is derived from rule_1 (without NAC) is matched by label_2, which is generated by rule_2 (with NAC). These matches between labels obtained from rules with and without NACs are the reason why the congruence property does no longer hold. In fact, the actual definitions of bisimulation and borrowed context step are too coarse to handle NACs.

Our idea is to enrich the transition labels $J \to F \leftarrow K$ with some information provided by the NACs in order to define a finer bisimulation based on these labels. A label must not only know which structures (borrowed context) are needed to perform it, but also which forbidden structures (defined by the NACs) cannot be additionally present in order to guarantee its execution. These forbidden structures will be called *negative borrowed contexts* and are represented by objects N_i attached to the label via monomorphisms from the borrowed context F (see example below). In our server example, label_1 would remain without any negative borrowed context since rule_1 has no NAC. However, label_2 would be the label below on the left, where the negative borrowed context $\mathsf{F} \to \mathsf{N}_1$ specifies that if a T-task was in Q_1, then NAC_1 would have forbidden the BC step of $\mathsf{J} \to \mathsf{G}'$ via rule_2. That is, with the new form of labels the two graphs

are no longer bisimilar and hence we no longer have a counterexample to the congruence property.



The intuition of negative borrowed contexts is the following: given $J \to G$, whenever it is possible to derive a label $J \to F \leftarrow K$ with negative borrowed context $F \to N_i$ via a production p with NACs, then if $J \to G$ is inserted into a context[5] $J \to N_i \leftarrow J$ no further label can be derived from $J \to \overline{G}$ via p since some of its NACs will forbid the rule application (see example above on the right). Put differently the label says that a transition can be executed if the environment "lends" F as minimal context. Furthermore the environment can observe that a production is only executable under certain constraints on the context. Finally, it is not executable at all if the object G^+ with borrowed context already contains the NAC.

3.2 DPO with Borrowed Contexts – Extension to Rules with NACs

Now we are ready to extend the DPO-BC framework to deal with productions with NACs. First we define when a BC step is executable.

Definition 6 (Executable Borrowed Context Step). *Assume that all arrows are mono. Given $J \to G$, a production $L \leftarrow I \to R; \{x_y \colon L \to NAC_y\}_{y \in Y}$ and a partial match $G \leftarrow D \to L$, we say that the BC step is* executable *on $J \to G$ if for the pushout G^+ in the diagram below there is no $p_y \colon NAC_y \to G^+$ with $m = p_y \circ x_y$ for every $y \in Y$.*

$$\begin{array}{ccccc} D & \longrightarrow & L & \xrightarrow{x_y} & NAC_y \\ \downarrow & PO & \downarrow m & \swarrow_{p_y} & \\ J & \longrightarrow & G & \longrightarrow & G^+ \end{array}$$

In the following we need the concept of a pair of jointly epi arrows in order to "cover" an object with two other objects. That is needed to find possible overlaps between the NACs and the object G^+ which includes the borrowed context.

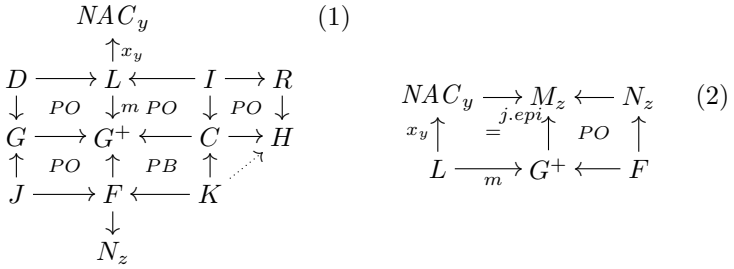
Definition 7 (Jointly Epi Arrows). *Two arrows $f \colon A \to B$ and $g \colon C \to B$ are* jointly epi *whenever for every pair of arrows $a, b \colon B \to D$ such that $a \circ f = b \circ f$ and $a \circ g = b \circ g$ it holds that $a = b$.*

In a pushout square the generated arrows are always jointly epi. This is a straightforward consequence of the uniqueness of the mediating arrow.

[5] $J \to N_i$ is the composition of $J \to F \to N_i$.

Definition 8 (Borrowed Context Rewriting for Rules with NACs). *Given $J \to G$, a production $L \leftarrow I \to R; \{L \to NAC_y\}_{y \in Y}$ and a partial match $G \leftarrow D \to L$, we say that $J \to G$ reduces to $K \to H$ with transition label $J \to F \leftarrow K; \{F \to N_z\}_{z \in Z}$ if the following holds:*

(i) the BC step is executable (as in Definition 6);
(ii) there is an object C and additional arrows such that Diagram (1) below commutes and the squares are either pushouts (PO) or pullbacks (PB) with monos;
(iii) the set $\{F \to N_z\}_{z \in Z}$ contains exactly the arrows constructed via Diagram (2) (where all arrows are mono). (That is, there exists an object M_z such that all squares commute and are pushouts or arrows are jointly epi as indicated.)



In this case a borrowed context step *is feasible and we write:* $(J \to G) \xrightarrow{J \to F \leftarrow K; \{F \to N_z\}_{z \in Z}} (K \to H)$.

Observe that Definition 8 coincides with Definition 3 when no NACs are present (cf. Condition (ii)). By taking NACs into account, a BC step can only be executed when G^+ contains no forbidden structure of any NAC_y at the match of L (Condition (i)). Additionally, enriched labels are generated (Condition (iii)).

In Condition (iii) the arrows $F \to N_z$ are also called *negative borrowed contexts* and each N_z represents the structures that should not be in G^+ in order to enable the BC step. This extra information in the label is of fundamental importance for the bisimulation game with NACs (Definition 9), where two objects with interfaces must not only agree on the borrowed context which enables a transition but also on what should not be present in order to perform the transition. The negative borrowed contexts $F \to N_z$ are obtained from $NAC_y \overset{x_y}{\leftarrow} L \overset{m}{\to} G^+ \leftarrow F$ of Diagram (1) via Diagram (2), where we create all possible overlaps M_z of G^+ and NAC_y in order to check which structures the environment should not provide in order to guarantee the execution of a BC step. To consider all possible overlaps is necessary in order to take into account that parts of the NAC might already be present in the object which is being rewritten.

Whenever the pushout complement in Diagram (2) exists, the object G^+ with borrowed context can be extended to M_z by attaching the negative borrowed context N_z via F. When the pushout complement does not exist, some

parts of G^+ which are needed to perform the extension are not visible from the environment and no negative borrowed context is generated.

Due to the non-uniqueness of the jointly-epi square one single negative application condition NAC_y may produce more than one negative borrowed context. Furthermore, the set $\{F \to N_z\}_{z \in Z}$ is in general infinite, but if we consider finite objects L, NAC_y and G^+ (i.e., objects which have only finitely many subobjects) there exist only finitely many overlaps M_z up to iso. Hence the set $\{F \to N_z\}_{z \in Z}$ can be finitely represented by forming appropriate isomorphism classes of arrows.

A concrete instance of Diagram (2) is discussed in Section 5 in relation with our running example.

Definition 9 (Bisimulation and Bisimilarity with NACs). *Let $\mathcal{P}$ be a set of productions with NACs and $\mathcal{R}$ a symmetric relation containing pairs of objects with interfaces $(J \to G, J \to G')$. The relation $\mathcal{R}$ is called a* bisimulation *if, for every $(J \to G)\,\mathcal{R}\,(J \to G')$ and a transition $(J \to G) \xrightarrow{J \to F \leftarrow K; \{F \to N_z\}_{z \in Z}} (K \to H)$, there exists an object with interface $K \to H'$ and a transition $(J \to G') \xrightarrow{J \to F \leftarrow K; \{F \to N_z\}_{z \in Z}} (K \to H')$ such that $(K \to H)\,\mathcal{R}\,(K \to H')$.*

We write $(J \to G) \sim (J \to G')$ whenever there exists a bisimulation $\mathcal{R}$ that relates the two objects with interface. The relation $\sim$ is called bisimilarity.

The difference between the bisimilarity of Definition 4 and the one above is the transition label, which in the latter case is enriched with negative borrowed contexts. Thus, Definition 9 yields in general a finer bisimulation.

We are now ready to show the congruence result. Recall that we are working in the framework of adhesive categories. Our main result below needs one extra requirement, namely that pullbacks preserve epis. The full proof of the following theorem with all lemmas is contained in the technical report [11].

Theorem 2 (Bisimilarity based on Productions with NACs is a Congruence). *The bisimilarity $\sim$ of Definition 9 is a congruence, i.e., it is preserved by contextualization as in Definition 2.*

Proof (Sketch). In [7] it was shown for the category of graph structures that bisimilarity derived from graph productions of the form $L \leftarrow I \rightarrow R$ with monos is a congruence. The pushout and pullback properties employed in [7] also hold for any adhesive category. Here we will extend the proof of [7] to handle productions with NACs in adhesive categories. All constructions used in this current proof are compliant with adhesive categories, except for some steps which require that pullbacks preserve epis.

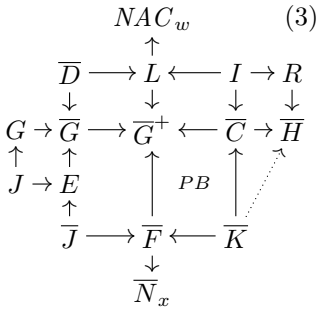
We will show that whenever $\mathcal{R}$ is a bisimulation, then $\hat{\mathcal{R}}$, which is the contextualization of $\mathcal{R}$ as in Definition 2, is also a bisimulation.

Let $\mathcal{R}$ be a bisimulation and let $(\overline{J} \to \overline{G})\ \hat{\mathcal{R}}\ (\overline{J} \to \overline{G}')$. That is, there is a pair $(J \to G)\ \mathcal{R}\ (J \to G')$ and a context $J \to E \leftarrow \overline{J}$ such that $\overline{J} \to \overline{G}$ and $\overline{J} \to \overline{G}'$ are obtained by inserting $J \to G$ and $J \to G'$ into this context.

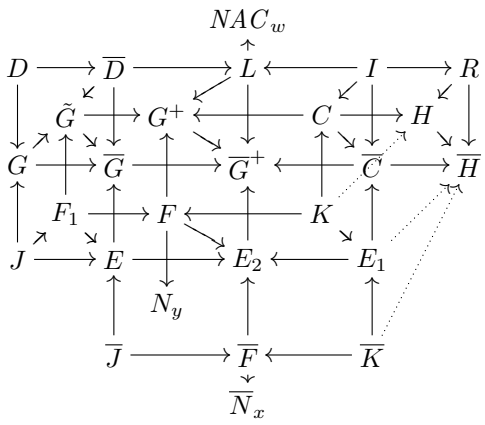
Let us also assume that $(\overline{J} \to \overline{G}) \xrightarrow{\overline{J}\to\overline{F}\leftarrow\overline{K};\{\overline{F}\to\overline{N}_x\}_{x\in X}} (\overline{K} \to \overline{H})$. Our goal is to show that there exists a transition label $(\overline{J} \to \overline{G}') \xrightarrow{\overline{J}\to\overline{F}\leftarrow\overline{K};\{\overline{F}\to\overline{N}_x\}_{x\in X}} (\overline{K} \to \overline{H'})$ with $(\overline{K} \to \overline{H}) \;\hat{\mathcal{R}}\; (\overline{K} \to \overline{H'})$, which implies that $\hat{\mathcal{R}}$ is a bisimulation. In *Step A* we construct a transition $(J \to G) \xrightarrow{J\to F\leftarrow K;\{F\to N_y\}_{y\in Y\cup Z}} (K \to H)$ which implies a transition $(J \to G') \xrightarrow{J\to F\leftarrow K;\{F\to N_y\}_{y\in Y\cup Z}} (K \to H')$ with $(K \to H) \;\mathcal{R}\; (K \to H')$, since $\mathcal{R}$ is a bisimulation. In *Step B* we extend the second transition to obtain the transition stated in our goal above. This argument is basically the same as in [7], except for the fact that here we are dealing with a bisimulation definition involving transition labels with negative borrowed contexts.

Step A: From transition $(\overline{J} \to \overline{G}) \xrightarrow{\overline{J}\to\overline{F}\leftarrow\overline{K};\{\overline{F}\to\overline{N}_x\}_{x\in X}} (\overline{K} \to \overline{H})$ we can derive Diagram (3), where the decomposition of $\overline{J} \to \overline{G}$ is shown explicitly, all arrows are mono and all squares are pushouts, except for the indicated pullback.

(3)



(4)



From Diagram (3) we construct Diagram (4) according to [7], i.e., we project the borrowed context diagram of $\overline{J} \to \overline{G}$ to a borrowed context diagram of $J \to G$, first without taking into account NACs. The square $(K, H, E_1, \overline{H})$ is a pushout.

Observe that all negative borrowed contexts $\overline{N}_x$ of the transition are obtained via Diagram (7). It can be shown that such a diagram can be "decomposed"[6] into two Diagrams (5) and (6), where the former shows the derivation of negative borrowed contexts for G^+. That is, every negative borrowed context of the larger object $\overline{G}^+$ is associated with at least one negative borrowed context of the smaller object G^+. Note that the transformation of one negative borrowed context into the other via Diagram (6) is only dependent on the context $J \to E \leftarrow \overline{J}$, into which $J \to G$ is inserted, but not on G itself, since E_2 is the pushout of $\overline{J} \to E$,

[6] We will use this "decomposition" result throughout the proof. Note that from Diagram (7) we can construct Diagrams (5) and (6) and vice versa. The proof of this result requires that pullbacks preserve epis.

$\overline{J} \to \overline{F}$. This independence of G will allow us to use this construction for $J \to G'$ in *Step B*.

In addition there might be further negative borrowed contexts $F \to N_y$ with indices $y \in Z$, where Y and Z are disjoint index sets. These are exactly the negative borrowed contexts for which Diagram (6) can not be completed since the pushout complement does not exist. If we could complete Diagram (6) we would be able to reconstruct Diagram (7).

Hence we obtain a transition from $J \to G$ which satisfies Conditions (ii) and (iii) of Definition 8. We still have to show that the BC step for G^+ is executable (Condition (i)). By assumption, the BC step from $\overline{J} \to \overline{G}$ of Diagram (4) is executable. One can show that a transition is executable if and only if none of the derived negative borrowed contexts is an iso. This means that there does not exist any iso $\overline{F} \to \overline{N}_x$, which in turn implies that no $F \to N_y$, $y \in Y$ is an iso. Furthermore no $F \to N_y$ with $y \in Z$ can be an iso, since otherwise we could complete Diagram (6). Finally we conclude with the observation above that the BC step from $J \to G$ is executable.

Since all conditions of Definition 8 are satisfied, we can derive the transition $(J \to G) \xrightarrow{J \to F \leftarrow K; \{F \to N_y\}_{y \in Y \cup Z}} (K \to H)$ from Diagram (4) using Definition 9. Since $\mathcal{R}$ is a bisimulation, this implies $(J \to G') \xrightarrow{J \to F \leftarrow K; \{F \to N_y\}_{y \in Y \cup Z}} (K \to H')$ with $(K \to H)\ \mathcal{R}\ (K \to H')$. Additionally, we can infer from Diagram (4) that $\overline{K} \to \overline{H}$ is the insertion of $K \to H$ into the context $K \to E_1 \leftarrow \overline{K}$.

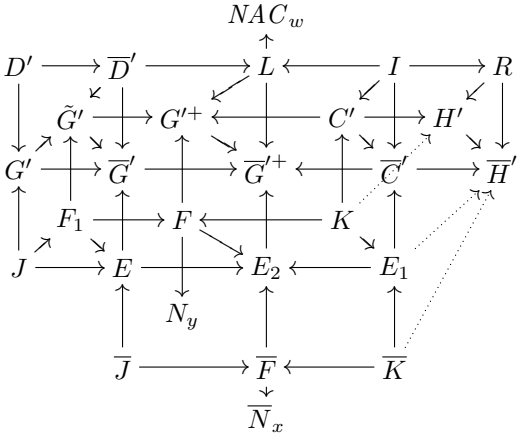
Step B: In *Step A* we have shown that $J \to G'$ can mimic $J \to G$ due to the bisimulation $\mathcal{R}$. Here we will show that $(\overline{J} \to \overline{G}')$ can also mimic $(\overline{J} \to \overline{G})$ since $\mathcal{R}$ is a bisimulation and both objects with interface are derived from the insertion of $J \to G$ and $J \to G'$ into the context $J \to E \leftarrow \overline{J}$.

We take the transition from $J \to G'$ to $K \to H'$ with $(K \to H)\ \mathcal{R}\ (K \to H')$ from *Step A* and construct a transition from $(\overline{J} \to \overline{G}')$ to $(\overline{K} \to \overline{H}')$ with $(\overline{K} \to \overline{H})\ \hat{\mathcal{R}}\ (\overline{K} \to \overline{H}')$. Recall that $\overline{J} \to \overline{G}'$ is $J \to G'$ in the context $J \to E \leftarrow \overline{J}$.

$$\begin{array}{ccccc} NAC_w & \to & M_y & \leftarrow & N_y \\ \uparrow & & \overset{j.epi}{=} \uparrow & PO & \uparrow \\ L & \to & G^+ & \leftarrow & F \end{array} \quad (5)$$

$$\begin{array}{ccccc} N_y & \to & M'_x & \leftarrow & \overline{N}_x \\ \uparrow & & \overset{j.epi}{=} \uparrow & PO & \uparrow \\ F & \to & E_2 & \leftarrow & \overline{F} \end{array} \quad (6)$$

$$\begin{array}{ccccc} NAC_w & \to & \overline{M}_x & \leftarrow & \overline{N}_x \\ \uparrow & & \overset{j.epi}{=} \uparrow & PO & \uparrow \\ L & \to & \overline{G}^+ & \leftarrow & \overline{F} \end{array} \quad (7)$$



(8)

According to [7] we obtain Diagram (8), first without considering the NACs. The square $(K, H', E_1, \overline{H}')$ is a pushout. Then we construct $\{\overline{F} \to \overline{N}_x\}_{x \in X}$ as shown in Diagram (6). The arrows $F \to E_2 \leftarrow \overline{F}$ and $\{F \to N_y\}_{y \in Y}$ are already present in Diagram (8) and so we build M'_x and $\overline{N}_x$ by considering all jointly epi squares. Each $\overline{F} \to \overline{N}_x$ constructed in this way can be also derived as a negative borrowed context with Diagram (7) (where $\overline{G}^+$ is replaced by $\overline{G}'^+$) due to the fact that we can construct Diagram (7) based on (5) and (6). Furthermore we will not derive additional negative borrowed contexts because the arrows $F \to N_y$ with $y \in Z$ can not be extended to negative borrowed contexts of the full object $\overline{G}'^+$ since an appropriate Diagram (6) does not exist. Hence we obtain a transition label from $\overline{J} \to \overline{G}'$ which satisfies Conditions (ii) and (iii) of Definition 8. We still have to show that the BC step for $\overline{G}'^+$ is executable (Condition (i)).

Observe that $F \to E_2 \leftarrow \overline{F}$ of Diagram (6) are equal in *Step A* and *Step B* and do not contain any information about G or G'. Hence we can conclude that Diagram (6) generates the same negative borrowed contexts in both steps. Since in Diagram (4) there is no negative borrowed context which is an iso, the same holds for Diagram (8). By the observation concerning isos we conclude that the BC step from $\overline{J} \to \overline{G}'$ is also executable.

Finally, by Definition 9 we infer that $(\overline{J} \to \overline{G}') \xrightarrow{\overline{J} \to \overline{F} \leftarrow \overline{K}; \{\overline{F} \to \overline{N}_x\}_{x \in X}} (\overline{K} \to \overline{H}')$, and since the square $(K, H', E_1, \overline{H}')$ is a pushout, $\overline{K} \to \overline{H}'$ is $K \to H'$ inserted into the context $K \to E_1 \leftarrow \overline{K}$. From earlier considerations we know that $\overline{K} \to \overline{H}$ is obtained by inserting $K \to H$ into $K \to E_1 \leftarrow \overline{K}$. Hence, we can conclude that $(\overline{K} \to \overline{H})\ \hat{\mathcal{R}}\ (\overline{K} \to \overline{H}')$ and we have achieved our goal stated at the beginning of the proof, which implies that $\hat{\mathcal{R}}$ is a bisimulation and $\sim$ is a congruence.

4 Up-to Techniques for DPO-BC with NACs

Bisimulation proofs often need infinite relations. Up-to techniques [14] relieve the onerous task of bisimulation proofs by reducing the size of the relation needed to define a bisimulation. It is also possible to check bisimilarity with finite up-to relations in some cases where any bisimulation is infinite. We first need to define progression (see also [14]).

Definition 10 (Progression with NACs). *Let $\mathcal{R}$, $\mathcal{S}$ be relations containing pairs of objects with interfaces of the form $(J \to G, J \to G')$, where $\mathcal{R}$ is symmetric. We say that $\mathcal{R}$ progresses to $\mathcal{S}$, abbreviated by $\mathcal{R} \rightarrowtail \mathcal{S}$, if whenever $(J \to G)\,\mathcal{R}\,(J \to G')$ and $(J \to G) \xrightarrow{J \to F \leftarrow K; \{F \to N_z\}_{z \in Z}} (K \to H)$, there exists an object with interface $K \to H'$ such that $(J \to G') \xrightarrow{J \to F \leftarrow K; \{F \to N_z\}_{z \in Z}} (K \to H')$ with $(K \to H)\,\mathcal{S}\,(K \to H')$.*

According to Definition 9, a relation $\mathcal{R}$ is a bisimulation if and only if $\mathcal{R} \rightarrowtail \mathcal{R}$.

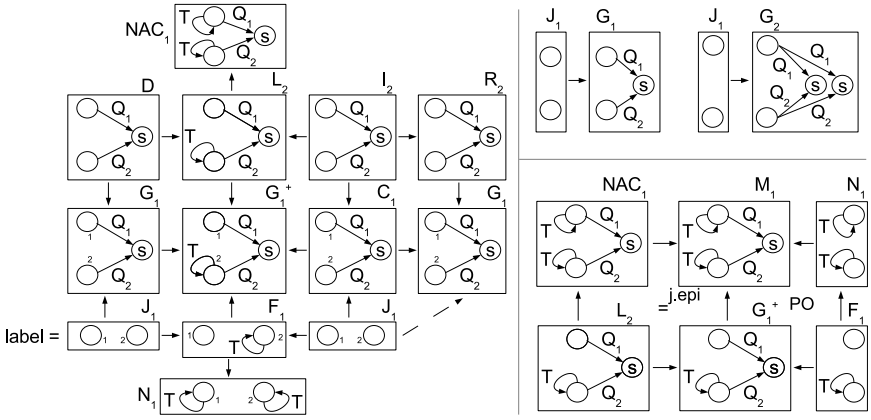
Definition 11 (Bisimulation up to Context with NACs). *Let $\mathcal{R}$ be a symmetric relation containing pairs of objects with interfaces of the form $(J \to G, J \to G')$. If $\mathcal{R} \rightarrowtail \hat{\mathcal{R}}$, where $\hat{\mathcal{R}}$ is the closure of $\mathcal{R}$ under contextualization, then $\mathcal{R}$ is called* bisimulation up to context.

Proposition 1 (Bisimulation up to Context with NACs implies Bisimilarity). *Let $\mathcal{R}$ be a bisimulation up to context. Then it holds that $\mathcal{R} \subseteq \sim$.*

Proof. Follows quite easily from the proof of Theorem 2 (see also [7]).

5 Example: Servers as Graphs with Interfaces

Here we apply the DPO-BC extension to NACs in order to check the bisimilarity of two graphs with interfaces $\mathsf{J}_1 \to \mathsf{G}_1$ and $\mathsf{J}_1 \to \mathsf{G}_2$ (shown below on the right) with respect to rule_1 and rule_2 of Section 3.1. Here G_1 contains only one server, whereas G_2 contains two severs which may work in parallel.



Above on the left we show a transition derivation for $\mathsf{J}_1 \to \mathsf{G}_1$ (which contains only one server) via rule_2 according to Definition 8. There is no mono $\mathsf{NAC}_1 \to \mathsf{G}_1^+$ forbidding the BC rewriting (Condition (i)) and the step is executable. The graph C_1 and additional monos lead to the BC step (Condition (ii)). The construction of the negative borrowed context $\mathsf{F}_1 \to \mathsf{N}_1$ from $\mathsf{NAC}_1 \leftarrow \mathsf{L}_2 \to \mathsf{G}_1^+ \leftarrow \mathsf{F}_1$, as specified in Condition (iii), is shown on the right. Here the graph M_1 is the only possible overlap of NAC_1 and G_1^+ such that the square with indicated jointly epi monos commutes. Since the pushout complement $\mathsf{F}_1 \to \mathsf{N}_1 \to \mathsf{M}_1$ exists, G_1^+ can be indeed extended to M_1 by gluing N_1 via F_1. All three conditions of Definition 8 are satisfied and so the BC step above with $\mathsf{label} = \mathsf{J}_1 \to \mathsf{F}_1 \leftarrow \mathsf{J}_1; \{\mathsf{F}_1 \to \mathsf{N}_1\}$ is feasible. This transition can be interpreted as follows: the environment provides G_1 with a T-task in Q_2 (see borrowed context F_1) in order to enable the BC step, but the rewriting is only possible if no T-task is waiting in queue Q_1 (see N_1).

Analogously we can derive other transitions from $\mathsf{J}_1 \to \mathsf{G}_1$ and $\mathsf{J}_1 \to \mathsf{G}_2$, where the labels generated via rule_1 (without NAC) do not have any negative borrowed context. So $\mathsf{J}_1 \to \mathsf{G}_1$, $\mathsf{J}_1 \to \mathsf{G}_2$ and all their successors can be matched via a bisimulation and we conclude that $(\mathsf{J}_1 \to \mathsf{G}_1) \sim (\mathsf{J}_1 \to \mathsf{G}_2)$.

Note that in order to obtain an extended example, we could add a rule modeling the processing of tasks waiting in queue Q_1.

6 Conclusions and Future Work

We have shown how rules with NACs should be handled in the DPO with borrowed contexts and proved that the derived bisimilarity relation is a congruence. This extension to NACs is relevant for the specification of several kinds of non-trivial systems, where complex conditions play a very important role. They are also frequently used when specifying model transformation, such as transformations of UML models. Behaviour preservation is an important issue for model transformation.

Here we have obtained a finer congruence than the usual one. Instead, if one would reduce the number of possible contexts (for instance by forbidding contexts that contain certain patterns or subobjects), we would obtain coarser congruences, i.e., more objects would be equivalent. Studying such congruences will be a direction of future work.

Furthermore, a natural question to ask is whether there are other extensions to the DPO approach that, when carried over to the DPO-BC framework, would require the modification of transition labels. One such candidate are generalized application conditions, so-called graph conditions [15], which are equivalent to first-order logic and of which NACs are a special case. Such conditions would lead to fairly complex labels.

Due to the fact that the bisimulation checking procedure is time consuming and error-prone when done by hand, we plan to extend the on-the-fly bisimulation checking algorithm, defined in [16, 17], for productions with NACs. In order to do this efficiently we need further speed-up techniques such as additional up-to techniques and methods for downsizing the transition system, such as the elimination of independent labels. Preliminary investigations have already determined that the proof technique eliminating independent labels as in [6, 7] (or non-engaged labels as they are called in [18]) does not carry over straightforwardly from the case without NACs.

Some open questions remain for the moment. First, in the categorical setting it would be good to know whether pullbacks always preserve epis in adhesive categories. This question is currently open, as far as we know. Second, it is unclear where the congruence is located in the lattice of congruences that respect rewriting steps with NACs. As for IPO bisimilarity it is probably not the coarsest such congruence, since saturated bisimilarity is in general coarser [19]. So it would be desirable to characterize such a congruence in terms of barbs [20].

Also, it is not clear to us at the moment how NACs could be integrated directly into reactive systems and how the corresponding notion of IPO would look like. In our opinion this would lead to fairly complex notions, for instance one would have to establish a concept similar to that of jointly epi arrows.

Acknowledgements: We would like to thank Tobias Heindel for helpful discussions on this topic.

References

1. Milner, R.: Communication and concurrency. Prentice-Hall (1989)
2. Milner, R., Parrow, J.: A calculus for mobile process I. Information and Computation **vol. 100** (1992) pp. 1–40
3. Milner, R., Parrow, J., Walker, D.: A calculus for mobile process II. Information and Computation **vol. 100** (1992) pp. 41–77
4. Leifer, J.J.: Operational Congruences for Reactive Systems. PhD thesis, University of Cambridge Computer Laboratory (2001)
5. Leifer, J.J., Milner, R.: Deriving bisimulation congruences for reactive systems. In: Proc. of CONCUR '00. Volume 1877 of LNCS., Springer-Verlag (2000) pp. 243–258
6. Ehrig, H., König, B.: Deriving bisimulation congruences in the DPO approach to graph rewriting. In: Proc. of FoSSaCS '04. Volume 2987 of LNCS. (2004) pp. 151–166
7. Ehrig, H., König, B.: Deriving bisimulation congruences in the DPO approach to graph rewriting with borrowed contexts. Mathematical Structures in Computer Science **16**(6) (2006) pp. 1133–1163
8. Sassone, V., Sobociński, P.: Reactive systems over cospans. In: Proc. of LICS '05, IEEE (2005) pp. 311–320
9. Sobociński, P.: Deriving process congruences from reaction rules. PhD thesis, Department of Computer Science, University of Aarhus (2004)
10. Habel, A., Heckel, R., Taentzer, G.: Graph grammars with negative application conditions. Fundam. Inf. **26**(3-4) (1996) pp. 287–313
11. Rangel, G., König, B., Ehrig, H.: Deriving bisimulation congruences in the presence of negative application conditions. Technical Report 2008-1, Abteilung für Informatik und Angewandte Kognitionswissenschaft, Universität Duisburg-Essen (2008) to appear.
12. Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., Loewe, M.: Algebraic approaches to graph transformation part I: Basic concepts and double pushout approach. In Rozenberg, G., ed.: Handbook of Graph Grammars and Computing by Graph transformation, Volume 1: Foundations, World Scientific (1997) pp. 163–246
13. Lack, S., Sobociński, P.: Adhesive and quasiadhesive categories. RAIRO - Theoretical Informatics and Applications **39**(2) (2005) pp. 522–546
14. Sangiorgi, D.: On the proof method for bisimulation. In Wiedermann, J., Hájek, P., eds.: Proc. of MFCS '95. Volume 969 of LNCS., Springer (1995) pp. 479–488
15. Rensink, A.: Representing first-order logic using graphs. In: Proc. of ICGT '04. Volume 3256 of LNCS., Springer (2004) pp. 319–335
16. Rangel, G., König, B., Ehrig, H.: Bisimulation verification for the DPO approach with borrowed contexts. In: Proc. of GT-VMT '07. Volume 6 of Electronic Communications of the EASST. (2007)
17. Hirschkoff, D.: Bisimulation verification using the up-to techniques. International Journal on Software Tools for Technology Transfer **3**(3) (August 2001) pp. 271–285
18. Milner, R.: Pure bigraphs: structure and dynamics. Inf. Comput. **204**(1) (2006) pp. 60–122
19. Bonchi, F., König, B., Montanari, U.: Saturated semantics for reactive systems. In: Proc. of LICS '06, IEEE (2006) pp. 69–80
20. Rathke, J., Sassone, V., Sobociński, P.: Semantic barbs and biorthogonality. In: Proc. of FoSSaCS '07. Volume 4423 of LNCS., Springer (2007) pp. 302–316