UNIVERSITÄT
DUISBURG
ESSEN

ABTEILUNG FÜR INFORMATIK UND
ANGEWANDTE KOGNITIONSWISSENSCHAFT

FAKULTÄT FÜR INGENIEURWISSENSCHAFTEN

INFORMATIK

Technischer Bericht Nr. 2008-01

Deriving Bisimulation Congruences in the Presence of Negative Application Conditions

Guilherme Rangel Barbara König Hartmut Ehrig

21.2.2008

IMPRESSUM:

Technische Berichte der Abteilung für Informatik und Angewandte Kognitionswissenschaft, Universität Duisburg-Essen

Herausgeber:

Abteilung für Informatik und Angewandte Kognitionswissenschaft
Fakultät für Ingenieurwissenschaften
Universität Duisburg-Essen
Campus Duisburg
47048 Duisburg

http://duepublico.uni-duisburg-essen.de/informatik/berichte.xml

Deriving Bisimulation Congruences in the Presence of Negative Application Conditions (Long Version)*

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 an example in terms of blade server systems 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 [2] and the π-calculus [3, 4] 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 [5, 6] 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 [7, 8] an

* This technical report is the full version of [1]. Research partially supported by the DFG project SANDS and DAAD (German Academic Exchange Service).

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 a congruence. This has turned out to be equivalent to a technique by Sassone and Sobociński [9, 10] 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) [11] 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 certain 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 behavioral 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 (not) 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 [8] (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 examples.

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 proof techniques for our extension. Finally, we present two examples in terms of graph transformation: a small one in Section 5 and a more elaborate one in Section 6, where blade server systems are presented. The appendices contain additional proofs and further information about the examples.

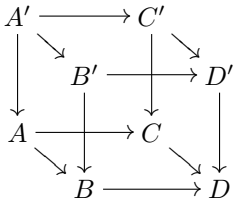
2 Double-Pushout with Borrowed Contexts

In this section we recall the DPO approach with borrowed contexts [7, 8]. 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 [8] 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 [7, 8], 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.*



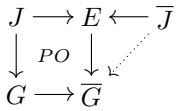
Pullbacks preserve monos and pushouts preserve epis in any category. Furthermore, for adhesive categories it is known that monos are preserved by pushouts. For the DPO-BC extension to productions with negative application conditions, defined in Section 3, we need one further requirement, namely that 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.

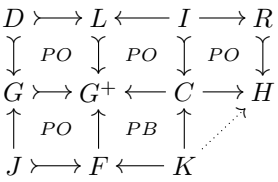
We will now define the notion of objects with interfaces and contexts, followed by the definition of a rewriting step with borrowed contexts as defined in [8] and extended in [10].

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* $J \to E$ *(see diagram below).*

[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:* $(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$.



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 [9]).

Note that with the procedure described above we may derive infinitely many labels of the form $J \to F \leftarrow K$. However, observe 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

$$(J \to G') \xrightarrow{J \to F \leftarrow K} (K \to H')$$

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

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.

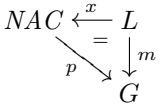
Theorem 1 (Bisimilarity is a Congruence [8]). *The bisimilarity relation* $\sim$ *is a congruence, i.e., it is preserved by contextualization as described in Definition 2.*

3 Borrowed Contexts with NACs

Here we will extend the DPO-BC framework of [8] 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* $p \circ x = m$.



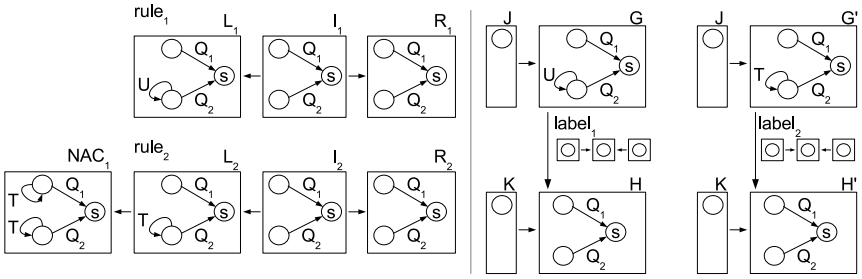
A rule $L \leftarrow I \to R$ with NACs is equipped with a finite set of negative application conditions $\{L \to NAC_y\}_{y \in Y}$ and is applicable to a match $m\colon L \to G$ only if all NACs are satisfied. If we add NACs to the rules in Definition 3, we have two ways to check their satisfiability: 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?

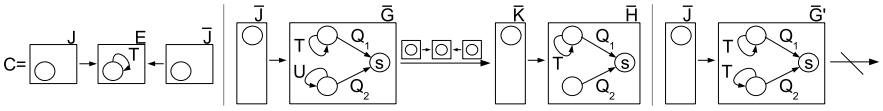
Let us assume that borrowed context rewriting works as in Definition 3 (with monos) if the total match $L \to G^+$ satisfies all NACs of a production, i.e., G^+ does not contain any prohibited structure (specified by a NAC) at the match of L. With the following example in terms of labeled directed graphs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

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 applied only when there is no other T-task waiting in the high priority queue Q_1. We assume that a processed task is consumed by the server (see R_1 and R_2).



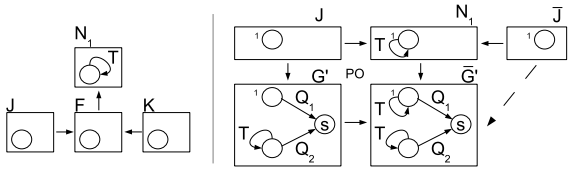
From the servers $\mathsf{J} \to \mathsf{G}$ and $\mathsf{J} \to \mathsf{G}'$ above we derive the labeled transition system (LTS) on the right w.r.t. rule_1 and rule_2. No further label can be derived from $\mathsf{K} \to \mathsf{H}$ and $\mathsf{K} \to \mathsf{H}'$ and the labels leading to these graphs are equal. By Definition 4 we could 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 monos 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.

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

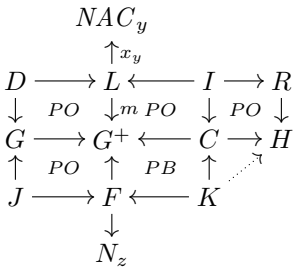
Definition 7 (Jointly Epi Arrows). *Two arrows $f\colon A \to B$ and $g\colon C \to B$ are* jointly epi *whenever for every pair of arrows $a, b\colon B \to D$ such that $a \circ f = b \circ f$ and $a \circ g = b \circ g$ it holds that $a = b$.*

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

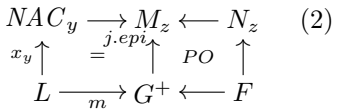
Definition 8 (Borrowed Context Rewriting for Rules with NACs). *Given $J \to G$, a production $L \leftarrow I \to R; \{L \to NAC_y\}_{y \in Y}$ and a partial match $G \leftarrow D \to L$, we say that $J \to G$ reduces to $K \to H$ with transition label $J \to F \leftarrow K; \{F \to N_z\}_{z \in Z}$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

(i) the BC step is executable (as in Definition 6);
(ii) there are objects G^+, 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.)

(1)



(2)



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

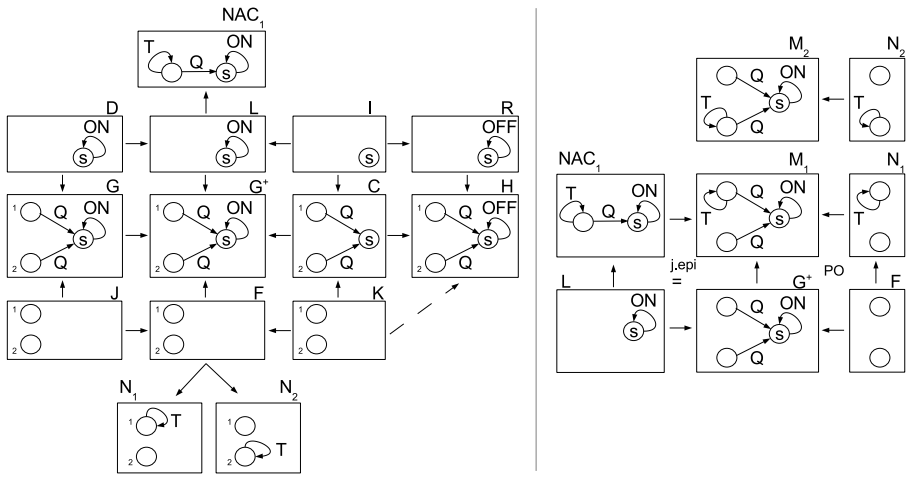
Observe that Definition 8 coincides with Definition 3 when no NACs are present (cf. Condition (ii)). By taking NACs into account, a BC step can only be executed when G^+ contains no forbidden structure of any negative application condition NAC_y at the match of L (Condition (i)). Additionally, enriched 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 as depicted in the example below. The rule used in the BC step on the left shows that an online server (marked with an ON-loop) can be turned off only if there is no standard task in any of its queues. Note that there are two possible overlaps between NAC_1 and G^+. On the right we show the two corresponding negative borrowed contexts $\{\mathsf{F} \to \mathsf{N_z}\}_{\mathsf{z}\in\{1,2\}}$. We depict in detail the construction of $\mathsf{F} \to \mathsf{N}_1$ as described in Definition 8.



Furthermore, in Definition 8 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.

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.

All lemmas mentioned in the proof can be found in Appendix A. Furthermore, some steps in the proof can be conveniently illustrated by Venn-like diagrams, which can be found in Appendix B.

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. In [8] 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 [8] also hold for any adhesive category. Here we will extend the proof of [8] to handle productions with NACs in adhesive categories. All constructions used in this current proof are compliant with adhesive categories, except for parts of Lemma 2 (in Appendix A), which requires 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. With the following argument we can infer that $\hat{\sim} \subseteq \sim$ and that $\sim$ is a congruence: Whenever $(\overline{J} \to \overline{G}) \,\hat{\sim}\, (\overline{J} \to \overline{G}')$, there exists a bisimulation $\mathcal{R}$ such that $(\overline{J} \to \overline{G})\ \hat{\mathcal{R}}\ (\overline{J} \to \overline{G}')$. Since, as we will show, $\hat{\mathcal{R}}$ is a bisimulation, it follows that $(\overline{J} \to \overline{G}) \sim (\overline{J} \to \overline{G}')$.

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

$$(\overline{J} \to \overline{G}') \xrightarrow{\overline{J} \to \overline{F} \leftarrow \overline{K}; \{\overline{F} \to \overline{N}_x\}_{x \in X}} (\overline{K} \to \overline{H'})$$

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

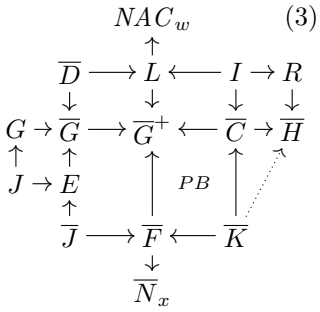
$$(J \to G) \xrightarrow{J \to F \leftarrow K; \{F \to N_y\}_{y \in Y \cup Z}} (K \to H)$$

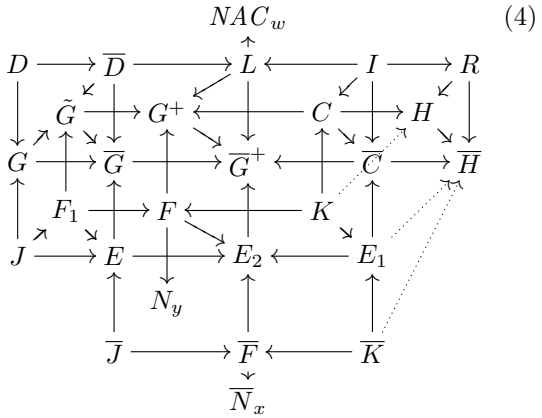
which implies a transition

$$(J \to G') \xrightarrow{J \to F \leftarrow K; \{F \to N_y\}_{y \in Y \cup Z}} (K \to H')$$

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 [8], 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.





From Diagram (3) we construct Diagram (4) according to [8], 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 is shown in Lemma 6 (NAC compatibility) that such a diagram can be "decomposed" 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 borrowed context of the smaller object G^+. Note that the transformation of one negative borrowed context into the other 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 $\overline{J} \to E$, $\overline{J} \to \overline{F}$. This independence of G will allow us to use this construction for $J \to G'$ in *Step B* (see Appendix B for this construction as a Venn diagram).

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) due to Lemma 6.

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. So by Lemma 7 there does not exist any iso $\overline{F} \to \overline{N}_x$, which by Lemma 5 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). Then by Lemma 7 we conclude 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 $J \to E \leftarrow \overline{J}$.

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

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

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

$$\begin{array}{ccccccccc} & & & & NAC_w & & & & \\ & & & & \uparrow & & & & \\ D' & \longrightarrow & \overline{D}' & \longrightarrow & L & \longleftarrow & I & \longrightarrow & R \\ \downarrow & & \downarrow & \swarrow & \downarrow & \swarrow & \downarrow & \swarrow & \downarrow \\ & & \tilde{G}' & \longrightarrow & G'^+ & \longleftarrow & C' & \longrightarrow & H' \\ & \nearrow & \uparrow & \searrow & \uparrow & \searrow & \uparrow & \searrow & \downarrow \\ G' & \longrightarrow & \overline{G}' & \longrightarrow & \overline{G}'^+ & \longleftarrow & \overline{C}' & \longrightarrow & \overline{H}' \\ \uparrow & & \uparrow & \searrow & \uparrow & & \uparrow & & \\ & & F_1 & \longrightarrow & F & \longleftarrow & K & \searrow & \\ & \nearrow & \uparrow & & \downarrow & \searrow & \uparrow & & \\ J & \longrightarrow & E & & E_2 & \longleftarrow & E_1 & & \\ & & \uparrow & & \downarrow & & \uparrow & & \\ & & \overline{J} & \longrightarrow & \overline{F} & \longleftarrow & \overline{K} & & \\ & & & & \downarrow & & & & \\ & & & & \overline{N}_x & & & & \end{array} \quad (8)$$

According to [8] 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) due to Lemma 6 (NAC compatibility). 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 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' (see Appendix B). 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 Lemma 7 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 Proof 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. In order to introduce this technique we first need to define the notion of 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.

It can be shown that bisimilarity up to context implies bisimilarity.

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 [8]). □

For productions of the form $L \leftarrow I \rightarrow R$ (without NACs) a bisimulation checking technique is proposed in [8] and it takes into account only certain labels. A label is considered superfluous and called *independent* if we can add two arrows $D \rightarrow J$ and $D \rightarrow I$ to the BC step diagram in Definition 3 such that $D \rightarrow I \rightarrow L = D \rightarrow L$ and $D \rightarrow J \rightarrow G = D \rightarrow G$. That is, intuitively, the object G to be rewritten and the left-hand side L overlap only in their interfaces. Any move made by an independent label need not be matched in the bisimulation game, since a matching transition is always possible. Hence, only *dependent* labels have to be checked. Dependent labels are called *engaged* in Milner's approach [15]. In the following we will investigate whether this technique can be carried over to productions with NACs.

First, we repeat the relevant definition for productions without NACs.

Definition 12 ((In)Dependent Transition Labels of Productions without NACs). *Let* $(J \rightarrow G) \xrightarrow{J \rightarrow F \leftarrow K} (K \rightarrow H)$ *be a transition of* $(J \rightarrow G)$. *We say that this transition is* independent *whenever we can add two arrows* $D \rightarrow J$ *and* $D \rightarrow I$ *to the diagram in Definition 3 such that the diagram below commutes, i.e.,* $D \rightarrow I \rightarrow L = D \rightarrow L$ *and* $D \rightarrow J \rightarrow G = D \rightarrow G$. *We write* $(J \rightarrow G) \xrightarrow{J \rightarrow F \leftarrow K}_d (K \rightarrow H)$ *if the transition is not independent and we call it* dependent.

$$\begin{array}{ccccccc} D & \longrightarrow & L & \longleftarrow & I & \longrightarrow & R \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ G & \longrightarrow & G^+ & \longleftarrow & C & \longrightarrow & H \\ \uparrow & & \uparrow & & \uparrow & & \\ J & \longrightarrow & F & \longleftarrow & K & & \end{array} \qquad (9)$$

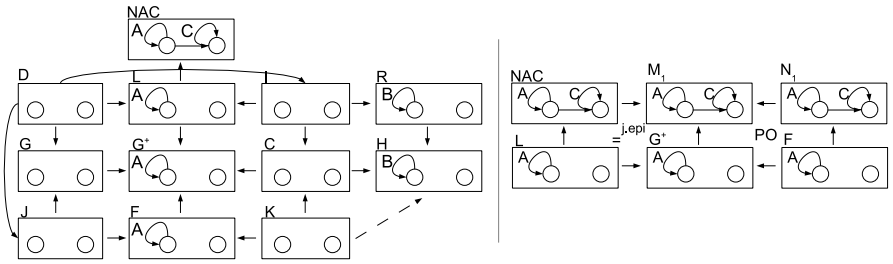
Let $\mathcal{R}$, $\mathcal{S}$ *be relations containing pairs of objects with interfaces of the form* $(J \rightarrow G, J \rightarrow G')$, *where* $\mathcal{R}$ *is symmetric. We say that* $\mathcal{R}$ *d-progresses to* $\mathcal{S}$, *abbreviated by* $\mathcal{R} \rightarrowtail_d \mathcal{S}$, *if whenever* $(J \rightarrow G)\,\mathcal{R}\,(J \rightarrow G')$ *and* $(J \rightarrow G) \xrightarrow{J \rightarrow F \leftarrow K}_d (K \rightarrow H)$, *there exists an object with interface* $K \rightarrow H'$ *such that*[6] $(J \rightarrow G') \xrightarrow{J \rightarrow F \leftarrow K} (K \rightarrow H')$ *and* $(K \rightarrow H)\,\mathcal{S}\,(K \rightarrow H')$.

If no NACs are present, the proof technique works according to the following proposition.

Proposition 2. *Let* $\mathcal{R}$ *be symmetric and let* $\mathcal{R} \rightarrowtail_d \hat{\mathcal{R}}$. *This implies that* $\mathcal{R}$ *is contained in* $\sim$.

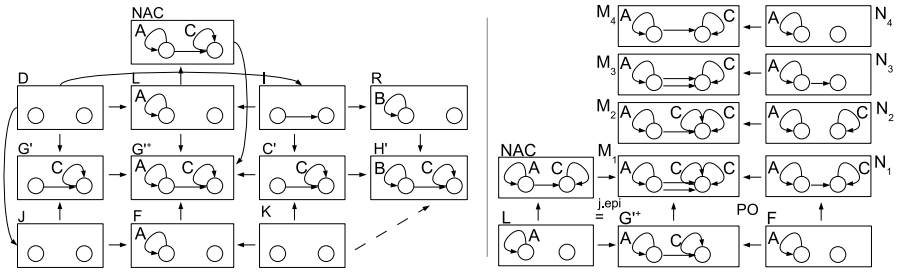
Unfortunately, as we will show in the following counterexample, the proof technique based on (in)dependent labels does not carry over straightforwardly into the setting with NACs. We will give an example for a transition with an independent label for one graph that can not be simulated by its partner due to the fact that the negative application condition is satisfied for the first graph, but not for the second.

[6] Note that $J \rightarrow G'$ may answer with an independent transition label.



Consider two graphs with interface: $\mathsf{J} \to \mathsf{G}$ above and $\mathsf{J} \to \mathsf{G}'$ below. Above we depict how the independent label $l_i = \mathsf{J} \to \mathsf{F} \leftarrow \mathsf{K}; \{\mathsf{F} \to \mathsf{N}_1\}$ is derived from $\mathsf{J} \to \mathsf{G}$ via a graph production $\mathsf{p} = \mathsf{L} \leftarrow \mathsf{I} \to \mathsf{R}; \{\mathsf{L} \to \mathsf{NAC}\}$ using Definition 8.

Recall that if p was a rule without NAC then the same transition label l_i would be induced for $\mathsf{J} \to \mathsf{G}'$ and it would not be necessary to consider these labels in the bisimulation checking procedure [8]. However, the presence of a NAC in p restricts the applicability of this proof technique since the transition with label l_i of $\mathsf{J} \to \mathsf{G}'$ is only feasible if the NAC allows the execution of the BC step. Below we illustrate the induced BC step from $\mathsf{J} \to \mathsf{G}'$ via p. This BC step is not executable since G'^{+} contains NAC. However, for completeness we still list all negative borrowed contexts (below on the right).



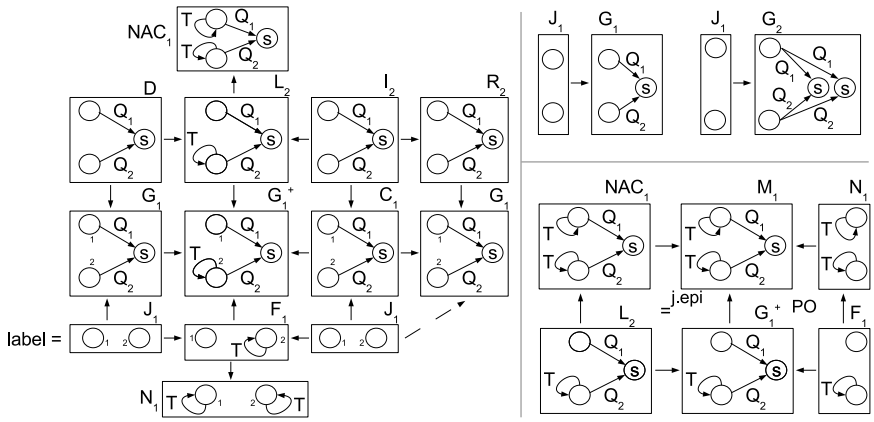
Note that in other cases the BC step of the second graph could be feasible, but with different negative borrowed contexts. Again, in this case the two steps would not properly match each other and the two graphs would not be bisimilar.

As we have shown, the proof technique based on independent labels does not work in general for rules with NACs. On the other hand, we can still use these labels to improve efficiency of the bisimulation checking procedure. This works as follows: whenever we derive an independent label from $\mathsf{J} \to \mathsf{G}$ via a production with NACs, we can construct the same borrowed context diagram for $\mathsf{J} \to \mathsf{G}'$ and it only remains to show that in both cases the same set of negative borrowed contexts is produced. Then for sure the BC step from $\mathsf{J} \to \mathsf{G}'$ is executable and both independent labels are suitable matched. Furthermore, in this case it is not necessary to check whether the pair of successors is contained in the bisimulation relation since both can be obtained by inserting $\mathsf{J} \to \mathsf{G}$, $\mathsf{J} \to \mathsf{G}'$ into the same context (this is analogous to the corresponding proof in [8]). This simplification will often lead to smaller bisimulations.

In Appendix C we illustrate this efficiency improvement for our next example.

5 Example 1: 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 $J_1 \to G_1$ and $J_1 \to G_2$ (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 servers which may work in parallel.



Above on the left we show a transition derivation for $J_1 \to G_1$ (which contains only one server) via rule_2 according to Definition 8. There is no mono $\mathsf{NAC}_1 \to G_1^+$ 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 $F_1 \to N_1$ from $\mathsf{NAC}_1 \leftarrow L_2 \to G_1^+ \leftarrow F_1$, 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 $F_1 \to N_1 \to M_1$ 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 $\mathsf{label} = J_1 \to F_1 \leftarrow J_1; \{F_1 \to N_1\}$ 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). It can be shown that $J_1 \to G_2$ can do a matching step with the same label.

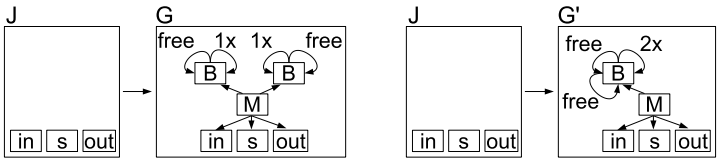
Analogously, we can derive other transitions from $J_1 \to G_1$ and $J_1 \to G_2$, where the labels generated via rule_1 (without NAC) do not have any negative borrowed context. Below we depict schematically the resulting labeled transition systems (LTS), for which we have already shown the derivation of label for $J_1 \to G_1$. In each LTS the labels on the left are derived via rule_1 and the labels on the right via rule_2. The label l' results from rule_1 with a maximal overlap of the graph (G_1 or G_2) and the left-hand side (similar to label). Both LTSs have several transitions pointing downwards (labelled l'',l''', etc.), which are derived with partial matches smaller than the matches of the loops. In fact, the labels in both LTSs are the same and the resulting states can be matched. So $J_1 \to G_1$, $J_1 \to G_2$

and all their successors can be matched via a bisimulation and we conclude that $(J_1 \to G_1) \sim (J_1 \to G_2)$.

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

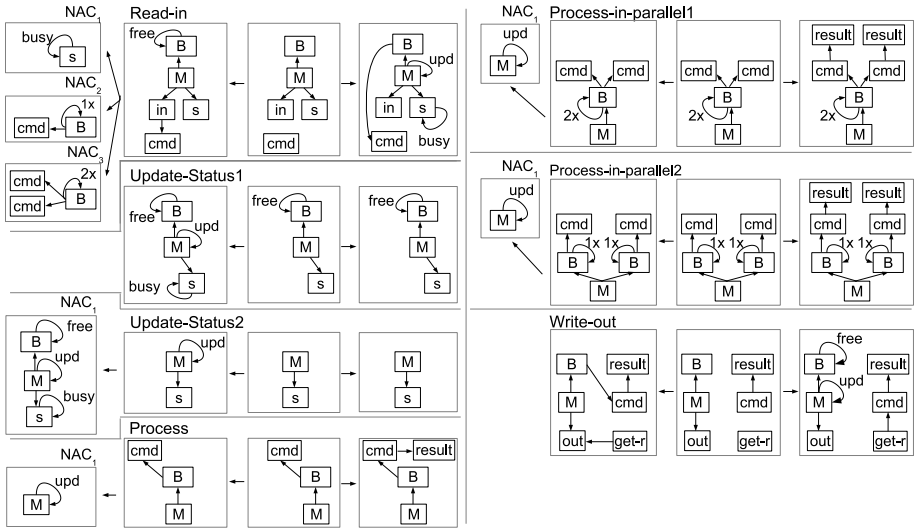
6 Example 2: Blade Servers

Here we present a more elaborate example in terms of blade server systems. A blade server is a server chassis which has multiple circuit boards, known as blade units. Each blade unit is a server in its own right with components such as processors, memory and local disk storage. These systems are flexible and modular since their processing power can be extended by just adding blade units.



Above we depict two blade servers. Each server chassis (M-labeled node) has three ports: input (in-node), output (out-node) and status (s-node). The input receives commands from external systems (not modeled here) which are executed by a blade. The output makes the result of command executions available to external systems. The status indicates if a blade server is busy and cannot handle any further request. The external systems have only access to these three ports (see interfaces J). Each blade (B-node) performs command executions independently from other blades. Single-processed blades (marked with a 1x-loop) perform one command execution at a time, while double processed blades (depicted with a 2x-loop) perform up to two commands.

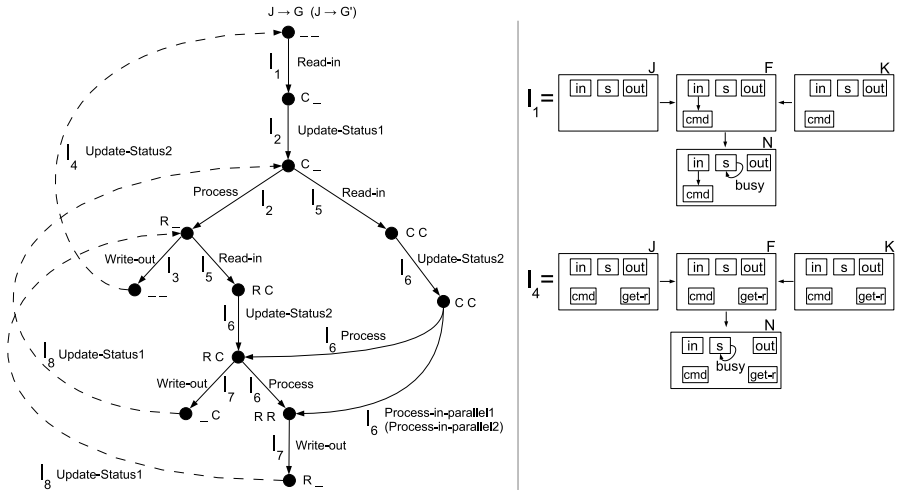
The operational semantics for our blade servers, which is not intended to be comprehensive, is given by the rules below. Read−in shows how a command (cmd-node) is read by an available blade (indicated by a free-loop). The free-loops on each blade specify its processing power currently available. To improve efficiency, each blade handles incoming command requests simultaneously. The NACs of Read−in restrict a command to be read only when the blade server is not busy (NAC_1) and not fully loaded (NAC_2 and NAC_3).



Update−Status1 and Update−Status2 update the current status of a server. When a command is read we set the status to busy and force updating. The production Update−Status1 checks if there is a free blade so that the busy-flag can be removed. Update−Status2 checks if the server is not fully loaded. Process executes commands if the server is not currently updating (see NAC_1). Process−in−parallel1 allows a double capacity blade to execute two commands in one single step. Process−in−parallel2 specifies that two single capacity blades can execute their commands in parallel. Finally, Write−out returns the result, sets the blade as available (free-loop) and flags the blade server to update its status.

From $\mathsf{J} \to \mathsf{G}$ and $\mathsf{J} \to \mathsf{G}'$ and the operational semantics we derive the labeled transition system (LTS) below w.r.t. to Definition 8. Black circles are states and arrows are annotated with labels and rule names. In our example, for each state the processing capability of a free blade is represented as "_". In the following C abbreviates "command" and R stands for "result". So when a blade reads a command, then "_" becomes "C" and when the command is executed, then "C" becomes "R". Since graphs are considered up to isomorphism "C_" ("R_") represents the same system state as "_C" ("_R"). If the start state is $\mathsf{J} \to \mathsf{G}$ ($\mathsf{J} \to \mathsf{G}'$) then Process−in−parallel1 (Process−in−parallel2) is applied to derive the transition between the "CC" and "RR" states. Labels with the same index (e.g. l_2) are equal. Examples of label derivations and the other labels of the LTS below can be found in Appendix D.

The LTS below is a simplification since the dashed arrows point to graphs that are basically $\mathsf{J} \to \mathsf{G}$ ($\mathsf{J} \to \mathsf{G}'$) plus some extra elements to represent the results of processed commands (which are the same in both cases). This issue is automatically handled by the up-to context technique given in Definition 11.



Blade servers are very flexible in terms of adding and removing blades to cover non-constant demands in terms of computational power. In our example even though each blade processes commands independently from other blades, we had to make this explicit for two single-processor blades inside the same blade server. If we had not defined Process−in−parallel2, it would not have been possible to show $(J \to G) \sim (J \to G')$. This problem might be solvable by using parallel rule applications, but this might lead to problems with non-injective matches. Furthermore we would get a different notion of behavioral equivalence that is reminiscent of step bisimilarity.

7 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 an important role. They are also frequently used when specifying model transformation, such as transformations of UML models. Behavior 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 [16], 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 [17, 18], 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. We discussed in Section 4 that the proof technique eliminating independent labels as in [7, 8] (or non-engaged labels as they are called in [15]) does not carry over straightforwardly from the case without NACs, but that it can still be useful. This needs to be studied further.

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].

Finally, 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

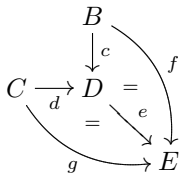
1. Rangel, G., König, B., Ehrig, H.: Deriving bisimulation congruences in the presence of negative application conditions. In: Proc. of FOSSACS '08, Springer (2008) LNCS, to appear.
2. Milner, R.: Communication and concurrency. Prentice-Hall (1989)
3. Milner, R., Parrow, J., Walker, D.: A calculus for mobile process I. Information and Computation **vol. 100**(1) (1992) pp. 1–40
4. Milner, R., Parrow, J., Walker, D.: A calculus for mobile process II. Information and Computation **vol. 100**(1) (1992) pp. 41–77
5. Leifer, J.J.: Operational Congruences for Reactive Systems. PhD thesis, University of Cambridge Computer Laboratory (2001)
6. 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
7. 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
8. 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

9. Sassone, V., Sobociński, P.: Reactive systems over cospans. In: Proc. of LICS '05, IEEE (2005) pp. 311–320
10. Sobociński, P.: Deriving process congruences from reaction rules. PhD thesis, Department of Computer Science, University of Aarhus (2004)
11. Habel, A., Heckel, R., Taentzer, G.: Graph grammars with negative application conditions. Fundam. Inf. **26**(3-4) (1996) pp. 287–313
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. Milner, R.: Pure bigraphs: structure and dynamics. Inf. Comput. **204**(1) (2006) pp. 60–122
16. Rensink, A.: Representing first-order logic using graphs. In: Proc. of ICGT '04. Volume 3256 of LNCS., Springer (2004) pp. 319–335
17. 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)
18. Hirschkoff, D.: Bisimulation verification using the up-to techniques. International Journal on Software Tools for Technology Transfer **3**(3) (August 2001) pp. 271–285
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

A Additional Lemmas

In this section we present the lemmas required by the proofs in this paper. Lemmas 2, 4, 5, 6 and 7 require the category to be adhesive. Lemma 2 (and Lemma 6, which is based on Lemma 2) additionally requires that pullbacks preserve epis.

Lemma 1. *Given the commuting diagram below, where the arrows c and d are jointly epi, the arrow e is an epi if and only if (f, g) is jointly epi.*



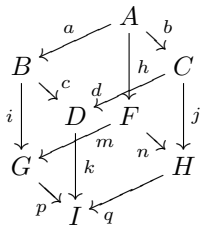
Proof. We consider both directions. Take two arrows $x, y\colon E \to F$.

("$\Leftarrow$") Assume that (f, g) is jointly epi and $x \circ e = y \circ e$. Then we have: $x \circ e \circ c = y \circ e \circ c \Rightarrow x \circ f = y \circ f$ and $x \circ e \circ d = y \circ e \circ d \Rightarrow x \circ g = y \circ g$. Since

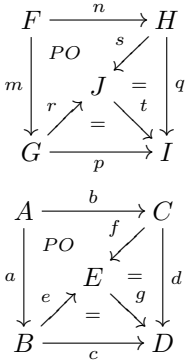
(f, g) is jointly epi, $x \circ f = y \circ f$ and $x \circ g = y \circ g$ we have $x = y$. Therefore we can conclude that e is an epi.

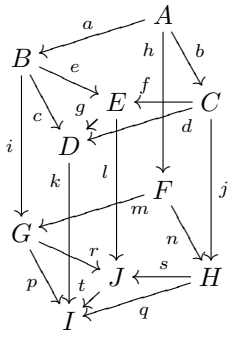
("⇒") Let e be an epi, $x \circ f = y \circ f$ and $x \circ g = y \circ g$. Then we have: $x \circ e \circ c = x \circ f = y \circ f = y \circ e \circ c$ and $x \circ e \circ d = x \circ g = y \circ g = y \circ e \circ d$. Since (c, d) is jointly epi, $x \circ e \circ c = y \circ e \circ c$ and $x \circ e \circ d = y \circ e \circ d$ we have $x \circ e = y \circ e$. Our assumption that e is an epi implies $x = y$. Thus, we conclude that (f, g) is jointly epi. □

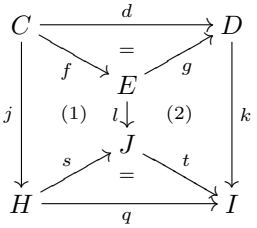
Lemma 2. *Given a commuting cube with all arrows mono and all lateral sides pushouts, then the pair of arrows (c, d) is jointly epi if and only if (p, q) is jointly epi.*



Proof. We construct the pushout (r, s, J) of m and n and obtain t as an induced arrow such that $p = t \circ r$ and $q = t \circ s$ (see the upper left diagram below). We proceed analogously for (A, C, B, D) (see the lower left diagram below). Gluing these new monos to the cube gives rise to the cube below, except for $l \colon E \to J$.







Note that $r \circ i \circ a = r \circ m \circ h = s \circ n \circ h = s \circ j \circ b$ since (A, F, B, G), (F, G, H, J) and (A, F, C, H) commute. So (A, B, C, E) as a pushout and $r \circ i \circ a = s \circ j \circ b$ imply a unique arrow $l \colon E \to J$ such that $l \circ e = r \circ i$ and $l \circ f = s \circ j$. From the inner cube we can infer that $(A, F, C, H) + (F, H, G, J)$ is a pushout and by the commutativity it implies $(A, C, B, E) + (B, E, G, J)$ as a pushout. By pushout decomposition, (B, E, G, J) is a pushout since (A, C, B, E) is a pushout. Analogously, (C, E, H, J) is also a pushout. The rightmost diagram is extracted from the cube: the outer square and (1) are pushouts. By pushout decomposition (2) is also a pushout.

Here we will show: (c, d) jointly epi $\Rightarrow$ (p, q) jointly epi. Since (A, B, C, E) is a pushout and (c, d) is jointly epi (assumption), then by Lemma 1 the arrow g is epi. The arrow t is also epi since (2) is a pushout. Finally, by Lemma 1 we obtain that p and q are jointly epi.

The other direction ("$\Leftarrow$") is shown as follows. Since (F, G, H, J) is a pushout and (p, q) is jointly epi (assumption), then by Lemma 1 the arrow t is epi. Since (2) is a pushout and l is mono, then (2) is also a pullback. The pullback (2) and t as epi imply that g is epi as well. So by Lemma 1 (c, d) is jointly epi. □

Lemma 3 (Composition of Jointly Epi Arrows). *Whenever* (1) *is a commuting diagram and* (f, g) *and* (i, j) *are pairs of jointly epi arrows, then the composition* $(f, g \circ j)$ *is also jointly epi.*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle x}\uparrow & (1) & \uparrow{\scriptstyle g} \\ C & \xrightarrow[i]{} & D \\ & & \uparrow{\scriptstyle j} \\ & & E \end{array}$$

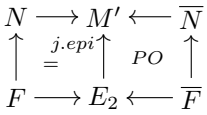
Proof. By Definition 7 (f, g) and (i, j) jointly epi means: *(i)* $\forall a, b\colon B \to F\colon\ a \circ f = b \circ f \ \wedge\ a \circ g = b \circ g \ \Rightarrow\ a = b$ and *(ii)* $\forall\ c, d\colon D \to G\colon\ c \circ i = d \circ i \ \wedge\ c \circ j = d \circ j \ \Rightarrow\ c = d$. We assume that $\forall a, b\colon B \to F : \ a \circ f = b \circ f \ \wedge\ a \circ (g \circ j) = b \circ (g \circ j)$ and we will show that this implies $a = b$. Observe that $a \circ g \circ i = a \circ f \circ x$ ((1) commutes) $= b \circ f \circ x$ ($a \circ f = b \circ f$ by assumption) $= b \circ g \circ i$ ((1) commutes). Then we have $a \circ g \circ i = b \circ g \circ i$ (previous calculation) and $a \circ g \circ j = b \circ g \circ j$ (assumption), which implies $a \circ g = b \circ g$ *(iii)*. By using *(i)* and *(iii)* ($a \circ f = b \circ f$ and $a \circ g = b \circ g$, respectively) together we have: $\forall a, b\colon B \to F : a \circ f = b \circ f \ \wedge\ a \circ g = b \circ g$, which by *(i)* implies that $a = b$. □

Lemma 4. *Given the diagram below, where all arrows are mono and* (c, d) *is jointly epi, whenever* b *is an iso then so is* c*.*

$$\begin{array}{ccc} C & \xrightarrow{d} & D \\ {\scriptstyle b}\uparrow & {\scriptstyle j.epi}\uparrow{\scriptstyle c} & \\ A & \xrightarrow[a]{} & B \end{array}$$

Proof. We have to check that c is an iso. Since c is mono and we are working in an adhesive category it is enough to show that c is epi (cf. [13]). Assume that there are arrows $x, y\colon D \to X$ with $x \circ c = y \circ c$. Composing with a gives us $x \circ c \circ a = y \circ c \circ a$ and hence $x \circ d \circ b = y \circ d \circ b$. Since b is an iso it follows that $x \circ d = y \circ d$. Finally, since c and d are jointly epi we obtain $x = y$. □

Lemma 5. *In the diagram below, where all arrows are mono, it holds: whenever* $\overline{F} \to \overline{N}$ *is not iso then* $F \to N$ *is not iso as well.*

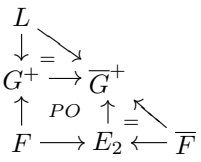


Proof. We show the contrapositive: $F \to N$ iso $\Rightarrow \overline{F} \to \overline{N}$ iso. The arrow $F \to N$ as an iso implies by Lemma 4 that $E_2 \to M'$ is also an iso. The pushout along monos is also a pullback and $E_2 \to M'$ iso implies $\overline{F} \to \overline{N}$ iso. □

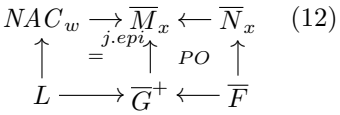
Lemma 6 (NAC Compatibility). *In the following let all arrows be mono and let Diagram (10) be given.*

If we have Diagram (12), then there exist objects M_y, N_y and M'_x such that Diagram (11)+(13) can be constructed as indicated. Furthermore, if we have Diagram (11)+(13), then there exists an object $\overline{M}_x$ such that Diagram (12) can be constructed as indicated.

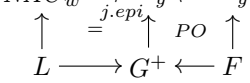
(10)



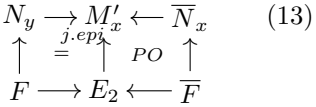
(12)



(11)



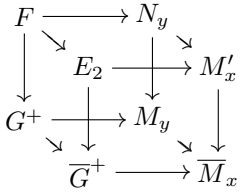
(13)

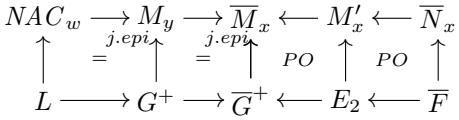


Proof. The proof is split into two steps.

Step 1 *(Diagram (11)+(13) ⇒ Diagram (12)).* We take the inner squares of Diagram (11)+(13) and build $\overline{G}^+$ as the pushout of $E_2 \leftarrow F \to G^+$ and $\overline{M}_x$ as the pushout of $M'_x \leftarrow N_y \to M_y$ (see Diagram (14)). Since *(left)* is a pushout and *(back)*, *(right)* and *(top)* commute, then there exists a unique arrow $\overline{G}^+ \to \overline{M}_x$ such that *(bottom)* and *(front)* commute. By pushout composition and then decomposition we find that *(front)* is a pushout. So $E_2 \to M'_x$ mono implies $\overline{G}^+ \to \overline{M}_x$ mono. The arrows $E_2 \to M'_x$ and $N_y \to M'_x$ are jointly epi which implies by Lemma 2 that $\overline{G}^+ \to \overline{M}_x$ and $M_y \to \overline{M}_x$ are also jointly epi.

(14)





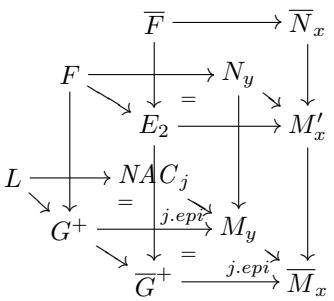
Gluing the leftmost and rightmost squares of Diagram (11)+(13) to the bottom and front faces of Diagram (14) produces the diagram above on the right, which by Lemma 3 and pushout composition is exactly Diagram (12). Note that

for each choice of M_y and M'_x in Diagram (11)+(13), there is a unique $\overline{M}_x$ (up to iso) leading to Diagram (12).

Step 2 *(Diagram (12) ⇒ Diagram (11)+(13)).* Combining Diagrams (10) and (12) gives rise to Diagram (15). We take all possible factorizations $NAC_w \to M_y \to \overline{M}_x$ of $NAC_w \to \overline{M}_x$ such that there exists an arrow $G^+ \to M_y$ (see Diagram (16)) with (1), (2) commuting and jointly epi and all arrows are monos. At least one such M_y—which can be obtained as the pushout of $L \to NAC_w$ and $L \to G^+$—exists. By pushout splitting we find M'_x and both squares are pushouts along monos.

$$\begin{array}{ccccc} NAC_w & \longrightarrow & \overline{M}_x & \longleftarrow & \overline{N_x} \\ \uparrow & = & \uparrow{\scriptstyle j.epi} & PO & \uparrow \\ L & \longrightarrow & G^+ \to \overline{G}^+ & \leftarrow & E_2 \leftarrow \overline{F} \end{array} \quad (15) \qquad \begin{array}{ccccccccc} NAC_w & \to & M_y & \to & \overline{M}_x & \leftarrow & M'_x & \leftarrow & \overline{N_x} \\ \uparrow & (1) & \uparrow & (2) & \uparrow & PO & \uparrow & PO & \uparrow \\ L & \longrightarrow & G^+ & \to & \overline{G}^+ & \leftarrow & E_2 & \leftarrow & \overline{F} \end{array} \quad (16)$$

Gluing the pushout of Diagram (10) to Diagram (16) produces Diagram (17), except for N_y and its arrows. So the fact that (*left*)+(*front*) is a pushout and the commutativity of *(bottom)* imply that the outer square over E_2 in Diagram (18) is a pushout. By pushout splitting we obtain the inner squares as pushouts along monos.



(17)

$$\begin{array}{ccccc} & & E_2 & & \\ & & = & & \\ F & \longrightarrow & N_y & \longrightarrow & M'_x \\ \downarrow & PO & \downarrow & PO & \downarrow \\ G^+ & \longrightarrow & M_y & \longrightarrow & \overline{M}_x \end{array} \quad (18)$$

$$\begin{array}{ccccccccc} NAC_j & \longrightarrow & M_y & \longleftarrow & N_y & \longrightarrow & M'_x & \longleftarrow & \overline{N}_x \\ \uparrow & = & \uparrow{\scriptstyle j.epi} & PO & \uparrow & = & \uparrow{\scriptstyle j.epi} & PO & \uparrow \\ L & \longrightarrow & G^+ & \longleftarrow & F & \longrightarrow & E_2 & \longleftarrow & \overline{F} \end{array} \quad (19)$$

Since all lateral sides of the cube are pushouts, the bottom and top faces commute and $\overline{G}^+ \to \overline{M}_x$ and $M_y \to \overline{M}_x$ are jointly epi we can infer by Lemma 2 that $N_y \to M'_x$ and $E_2 \to M'_x$ are jointly epi as well. By taking the squares we are interested in, we obtain Diagram (19), which is Diagram (11)+(13). Observe that for each choice of $\overline{M}_x$ in Diagram (12) and each factor M_y there is a unique N_y leading to Diagram (11)+(13). □

Lemma 7. *A borrowed context step (as in Definition 8) is not executable whenever there exists a mono $p_y\colon NAC_y \to G^+$ such that $m = p_y \circ x_y$ (see Definition 6). This is equivalent to the situation, in which there exists a negative borrowed context $F \to N_z$ which is an iso.*

Proof. We have to show: $\exists p_y\colon NAC_y \to G^+$ mono with $m = p_y \circ x_y \Leftrightarrow \exists$ negative borrowed context $(F \to N)$ which is iso.

$$\begin{array}{ccccc} NAC_y & \xrightarrow{p_y} & G^+ & \longleftarrow & F \\ {\scriptstyle x_y}\uparrow & = & \uparrow{\scriptstyle j.epi} & PO & \uparrow\wr \\ L & \xrightarrow[m]{} & G^+ & \longleftarrow & F \end{array}$$

("⇒"). Assume there exists a mono $p_y \colon NAC_y \to G^+$ with $m = p_y \circ x_y$ (left square of the diagram below). The arrows p_y and id_{G^+} are jointly epi. A pushout along monos is also a pullback. Thus, we can infer that $F \to N$ is also an iso (id_F).
("⇐"). Suppose that $F \to N$ is an iso, i.e., the arrow coincides with $F \xrightarrow{\sim} F$. The pushout implies $G^+ \xrightarrow{\sim} G^+$. That is, we have $id_{G^+} \circ m = p_y \circ x_y$ which can be simplified to $m = p_y \circ x_y$. Observe that p_y and $G^+ \xrightarrow{\sim} G^+$ are clearly jointly epi. □

B Objects with Interfaces as Venn Diagrams

We depict some diagrams of Theorem 2 as Venn diagrams. The left-hand side L of a production, its interface I, the object G and its interface J are shown as circles. The NAC is the circle L together with the "boomerang"-shaped area. Figure 1 shows typical overlaps between an object $J \to G$ and the left-hand side of a production that occur when a BC step takes place. Depending on the situation the NAC might have a bigger overlapping structure with G, which—in the picture—means that the NAC is rotated counterclockwise. On the right we show an overlapping when $J \to G$ is inserted into a context $J \to E \leftarrow \overline{J}$.

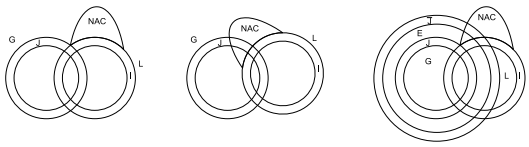


Fig. 1. Overlaps between $J \to G$ and a production $NAC \leftarrow L \leftarrow I \to R$

Figure 2 shows graphical representations of Diagrams (5),(6) and (7) of Theorem 2. Non-empty areas, i.e., areas where items are present, are shaded gray.

The diagram at the top shows the construction of negative borrowed contexts of the form $F \to N$ for the BC step of $J \to G$. The objects G^+ and NAC overlap, giving rise to M. The object G^+ is G plus the borrowed context F. So N represents exactly what should not be further provided by the environment in order to guarantee the feasibility of the BC step. Note that N does not contain any information about G which is not "visible" from the interface J.

The diagram at the bottom depicts how the negative borrowed context $\overline{F} \to \overline{N}$ is built for the BC step of $\overline{J} \to \overline{G}$. Also note that $\overline{N}$ does not contain any information about $\overline{G}$ which is not already present in the interface $\overline{J}$.

Finally, the diagram in the middle represents the translation of a negative borrowed context with respect to a new context that is added. Observe that the

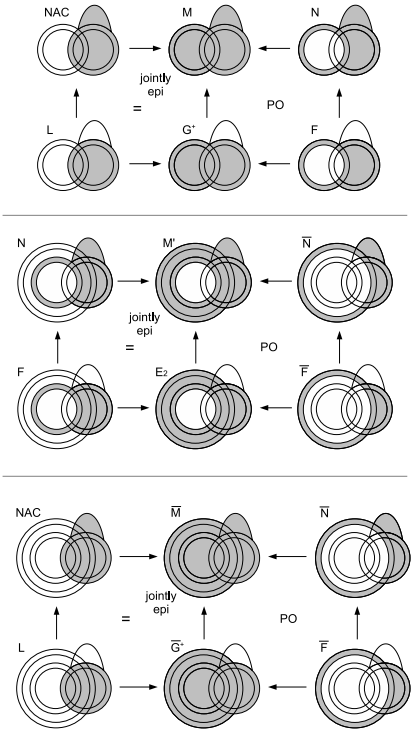


Fig. 2. Graphical representations of Diagrams (5), (6) and (7) of Theorem 2.

translation is based on the context E_2, which does not depend on the contents of G, and therefore can be used for both parts of the proof of Theorem 2.

C Example 1 (Servers) – Independent Label Derivation

Here we illustrate how independent labels can be used to improve the efficiency of the bisimulation checking procedure.

Consider the servers $\mathsf{J}_1 \to \mathsf{G}_1$ and $\mathsf{J}_1 \to \mathsf{G}_2$ from the example of Section 5. In Figure 3 we derive the transition label $\mathsf{J}_1 \to \mathsf{F}_1 \leftarrow \mathsf{K}_1; \{\mathsf{F}_1 \to \mathsf{N}_1\}$ via rule_2 according to Definition 8. This transition label is independent w.r.t. Definition 12 since there exist $\mathsf{D} \to \mathsf{J}_1$ and $\mathsf{D} \to \mathsf{I}_2$ such that $\mathsf{D} \to \mathsf{G}_1 = \mathsf{D} \to \mathsf{J}_1 \to \mathsf{G}_1$ and $\mathsf{D} \to \mathsf{L}_2 = \mathsf{D} \to \mathsf{I}_2 \to \mathsf{L}_2$. In this case the environment provides G with the entire left-hand side of the rule (see F_1).

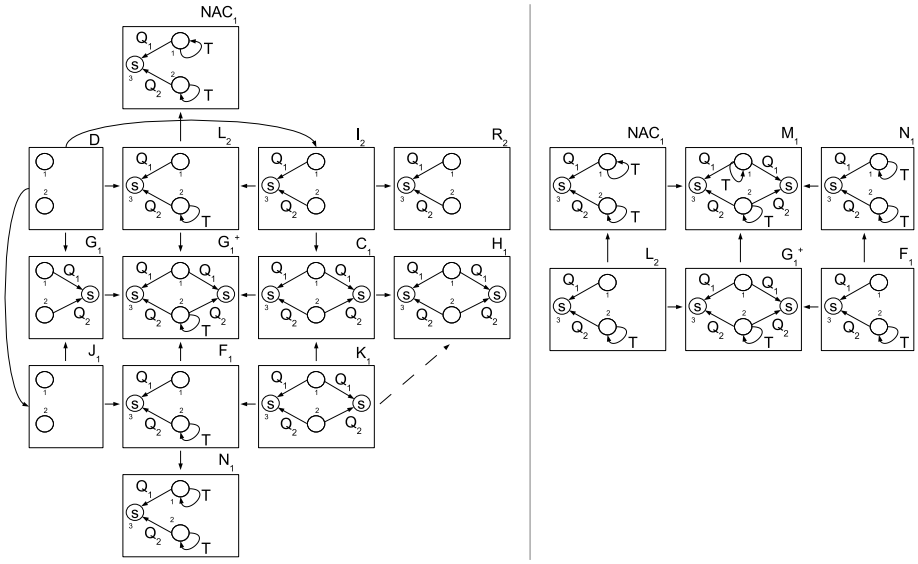
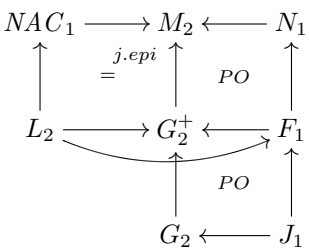
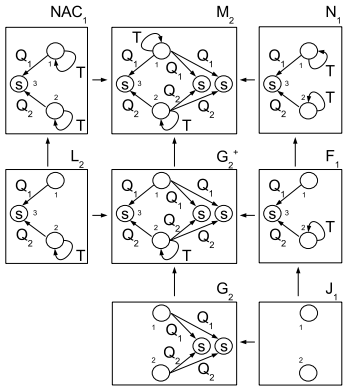


Fig. 3. Example of independent label derivation from $\mathsf{J}_1 \to \mathsf{G}_1$

Since the label above is independent it induces the derivation of an independent label $\mathsf{J}_1 \to \mathsf{F}_1 \leftarrow \mathsf{K}_1$ (without negative borrowed contexts) for the transition of $\mathsf{J}_1 \to \mathsf{G}_2$ via rule_2 (compare with the proof technique for productions without NACs in [8]). However, rule_2 has a NAC and so we have to check an additional requirement, namely that the negative borrowed contexts are the same for both independent transition labels. Whenever this extra requirement is satisfied the independent transition label $\mathsf{J}_1 \to \mathsf{F}_1 \leftarrow \mathsf{K}_1; \{\mathsf{F}_1 \to \mathsf{N}_1\}$ is the same for both BC steps and the BC step of $\mathsf{J}_1 \to \mathsf{G}_2$ is executable.

Below we depict how this condition is checked. In the diagram on the left the morphisms $NAC_1 \leftarrow L_2 \rightarrow F_1 \leftarrow J_1$ and $J_1 \rightarrow G_2$ are already known. The morphisms $L_2 \rightarrow F_1 \leftarrow J_1$ stems from the BC step of $\mathsf{J_1} \rightarrow \mathsf{G_1}$. We then construct the pushout square in the second row and $L_2 \rightarrow G_2^+$ is the composition of $L_2 \rightarrow F_1$ and $F_1 \rightarrow G_2^+$. The construction of the upper part of the diagram proceeds as in Definition 8 (BC rewriting with NACs). On the right we depict this diagram for our current example. Note that $\mathsf{NAC_1}$ is not present in $\mathsf{G_2^+}$ and the negative borrowed context $\mathsf{F_1} \rightarrow \mathsf{N_1}$ obtained below is the same as for the independent label of $\mathsf{J_1} \rightarrow \mathsf{G_1}$. In this case, the induced BC step of $\mathsf{J_1} \rightarrow \mathsf{G_2}$ is executable and provides a matching label for the transition of $\mathsf{J_1} \rightarrow \mathsf{G_1}$. Furthermore, we do not have to check whether the pair of successors is contained in the bisimulation relation since both can be obtained by inserting $\mathsf{J} \rightarrow \mathsf{G_1}$, $\mathsf{J} \rightarrow \mathsf{G_2}$ into the same context.





D Example 2 (Blade Servers) – Label Derivations

Here we show additional label derivations for the blade server example. Figure 4 depicts the BC step for the blade server $\mathsf{J} \rightarrow \mathsf{G}$, which produces label $\mathsf{l_1}$ of the LTS in Section 6. On the right we show how the negative borrowed context is generated. Observe that $\mathsf{L} \rightarrow \mathsf{NAC_2}$ and $\mathsf{L} \rightarrow \mathsf{NAC_3}$ of $\mathsf{Read{-}in}$ do not yield any negative borrowed contexts.

Figure 5 shows another case of label generation, namely the BC step via $\mathsf{Update{-}Status2}$ from the leftmost state ("__").

Finally, in Figure 6 we depict the remaining labels ($\mathsf{l_2}, \mathsf{l_3}, \mathsf{l_5}, \mathsf{l_6}, \mathsf{l_7}$ and $\mathsf{l_8}$) obtained for the blade servers of Section 6. These labels are associated with empty sets of negative borrowed contexts.

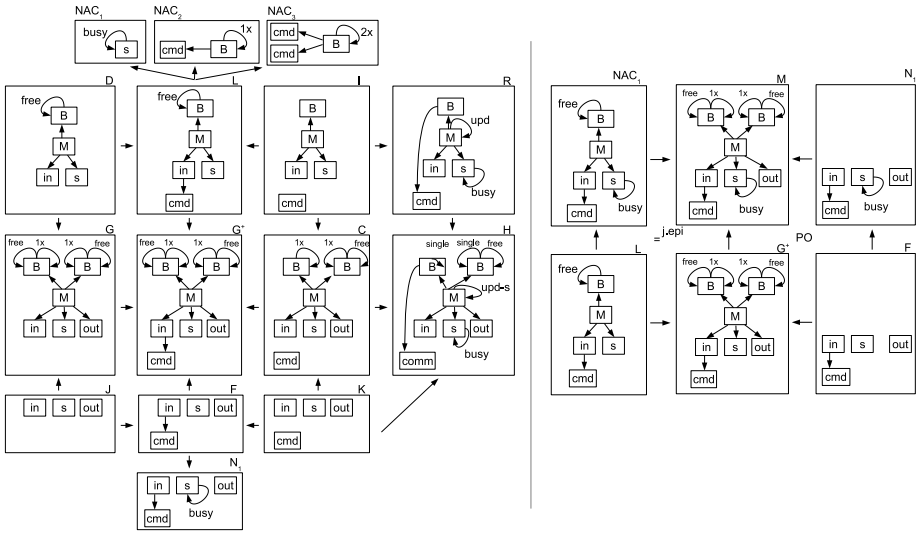


Fig. 4. Label derivation – label l_1

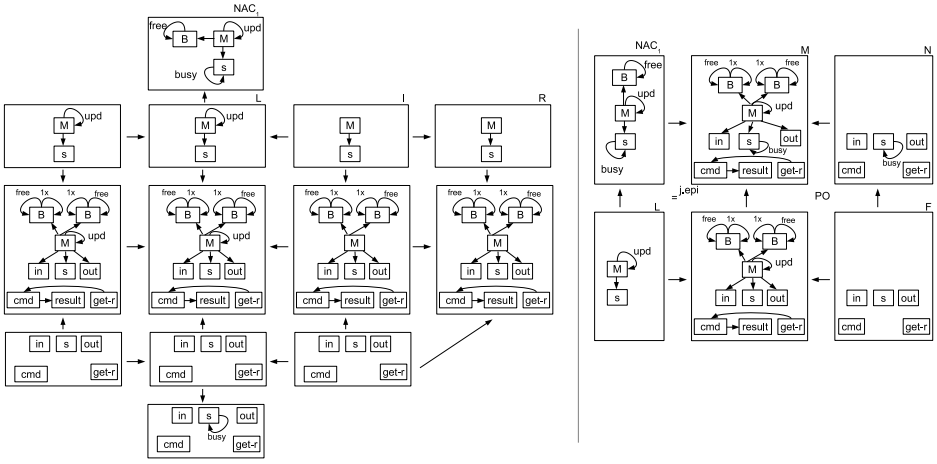


Fig. 5. Label derivation – label l_4

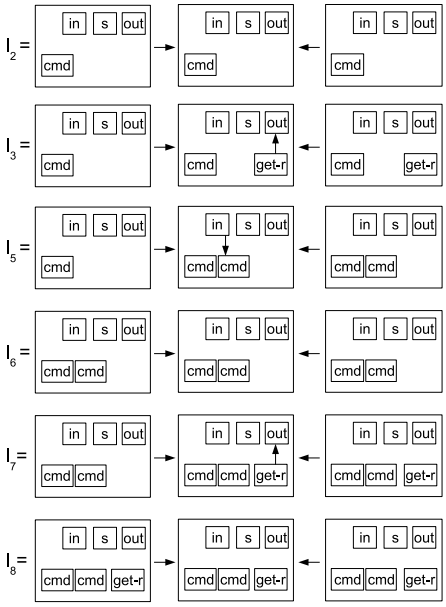


Fig. 6. Remaining transition labels for the blade server example