Deriving Bisimulation Congruences in the DPO Approach to Graph Rewriting*

Hartmut Ehrig[1] and Barbara König[2]

[1] Technische Universität Berlin, Germany
[2] Universität Stuttgart, Germany

ehrig@cs.tu-berlin.de koenigba@fmi.uni-stuttgart.de

Abstract. Motivated by recent work on the derivation of labelled transitions and bisimulation congruences from unlabelled reaction rules, we show how to solve this problem in the DPO (double-pushout) approach to graph rewriting. Unlike in previous approaches, we consider graphs as objects, instead of arrows, of the category under consideration. This allows us to present a very simple way of deriving labelled transitions (called rewriting steps with borrowed context) which smoothly integrates with the DPO approach, has a very constructive nature and requires only a minimum of category theory. The core part of this paper is the proof sketch that the bisimilarity based on rewriting with borrowed contexts is a congruence relation.

1 Introduction

In the last few years the problem of deriving labelled transitions and bisimulation congruences from unlabelled reaction or rewriting rules has received great attention. This line of research was motivated by the theory of bisimulation congruences for process calculi, such as the π-calculus [22]. A bisimilarity defined on unlabelled reduction rules is usually not a congruence, that is, it is not closed under the operators of the process calculus. Congruence is a very desirable property since it allows us to replace a subsystem with an equivalent one without changing the behaviour of the overall system and futhermore helps to make bisimilarity proofs modular.

Previous solutions have been to either require that two processes are related if and only if they are bisimilar under all possible contexts (see [17]) or to derive a labelled transition system manually. Since the first solution needs quantification over all possible contexts, proofs of bisimilarity can be very complicated. In the second solution, proofs tend to be much easier, but it is necessary to show that the labelled variant of the transition system is equivalent to the unlabelled variant.

So the idea which was formulated in the papers of Leifer/Milner [13, 14], Sewell [24] and Sassone/Sobocinski [23] is to automatically derive a labelled

* Research partially supported by the BMBF project DACHIA, the TMR network SEGRAVIS and EPSRC grant R93346/01.

transition system such that the resulting bisimilarity is a congruence. A central concept of this approach is to formalize the notion of minimal context which enables a process to reduce. Consider, for example, the CCS process $a.P$. It reduces when put into the contexts $_ \mid \bar{a}.Q$ and $_ \mid \bar{a}.Q \mid b.R$, but one is interested only in the first context, since it is in some sense smaller than the second one. This yields the labelled transition

$$a.P \xrightarrow{_ \mid \bar{a}.Q} P \mid Q,$$

saying that $a.P$ put into this contexts reacts and reduces to $P \mid Q$. Using all possible contexts as labels would also result in a bisimulation congruence, but we do not gain anything compared to quantification over all contexts.

In [13, 14] the notion of "minimal context" is formalized as the categorical concept of relative pushout respectively idem pushout. This notion has also been applied to bigraphs [10]. However, the theory is complicated by the fact that one can not work with isomorphism classes of graphs, since in this case the category under consideration would not possess all necessary relative pushouts. Thus one is forced to give unique names to all edges and nodes in a graph and to either work in a precategory or to construct a suitable category starting from such a precategory. Another approach, given by Sassone and Sobocinski [23], is to work with cells inside a 2-category.

It is our aim to achieve similar results in the context of graph rewriting [20], a framework which allows to model dynamic and concurrent systems consisting of interconnected components in a natural and intuitive way. Many process calculi such as the π-calculus [9, 19, 11] and the ambient calculus [8] can be translated into graph rewriting. We are specifically interested in the double-pushout (DPO) approach [2], one of the standard approaches to graph rewriting. So far, there is not yet a uniform theory of bisimulation for graph transformation systems. Using the concepts explained earlier would be possible in theory, but contradicts the philosophy behind graph rewriting where graphs are considered only up to isomorphism. Furthermore, deriving labels via relative pushouts is entirely non-trivial and can be rather complicated.

The approach which is presented in this paper is motivated by the work of Leifer/Milner and other contributions to this area, but does not directly rely on their theory. Instead we present a very simple way of deriving minimal contexts—we call them borrowed contexts—which smoothly integrates with the DPO approach and which has a very constructive nature. The only categorical concepts that are needed are standard pushouts and pullbacks. The main difference to previous approaches is that in our case graphs are objects and not arrows of the category under consideration. Our arrows instead are graph morphisms which provide the necessary tracking information for nodes and edges which, in the case of graphs as arrows, can only be provided by adding support to a category. This work is based on ideas presented in [3], a paper which points out similarities and differences between Milner's bigraphs [10, 16] and the DPO approach to graph rewriting.

Our main result states that bisimilarity defined on graph rewriting with borrowed contexts is indeed a congruence relation (see Theorem 7).

The paper is structured as follows: In Section 2 we will give a short introduction to the DPO approach, followed by the definition of rewriting with borrowed contexts (Section 3). Section 4 provides the proof ideas that the resulting bisimilarity is a congruence. After having introduced a proof technique we continue with an example showing borrowed contexts at work in Section 5.

This paper requires only basic knowledge of category theory [15]. In fact, we only need pushouts and pullbacks, including some general as well as specific preservation, composition and decomposition properties. The general properties hold in any category and the specific ones at least in the category of sets and, as needed in the paper, in the category of graphs. The specific properties are presented in our technical report [6], which also contains the full proof of our main result.

2 The DPO Approach to Graph Rewriting

We will first define a family of categories of graphs and graph morphisms, being as general as possible by defining graph structures [5], which include different forms of graphs such as directed graphs and hypergraphs.

Definition 1 (Graph Structures). *A* graph structure signature $GS = (S, OP, \Sigma)$ *consists of a set of sorts* S*, a family* $(OP_{s,s'})_{s,s' \in S}$ *of* unary *operator symbols and a family* $(\Sigma_s)_{s \in S}$ *of labelling alphabets.*

A graph structure A *over* GS *is a sort-indexed family* $(A_s)_{s \in S}$ *of carrier sets together with a sort-indexed family of labelling functions* $(l_s^A)_{s \in S}$ *such that* $l_s^A \colon A_s \to \Sigma_s$ *and an* OP*-indexed family of mappings* $(op^A)_{op \in OP}$ *such that* $op^A \colon A_s \to A_{s'}$ *if* $op \in OP_{s,s'}$.

A graph structure morphism $\varphi \colon A \to B$ *is a sort-indexed family of mappings* $\varphi = (\varphi_s \colon A_s \to B_s)_{s \in S}$ *such that* $l_s^A(x) = l_s^B(\varphi(x))$ *and* $op^B(\varphi(x)) = \varphi(op^A(x))$ *for all* $x \in A_s$*. A graph structure morphism* φ *is called* injective *if all its mappings are injective. It is an* isomorphism *if all mappings are bijective. An isomorphism of the form* $\varphi \colon A \to A$ *is called* automorphism.

The simplest graph structure signature has two sorts: *node* and *edge* and two operator symbols $s, t \in OP_{edge,node}$ standing for "source" and "target". Graph structures over this signature are ordinary labelled directed graphs and graph structure morphisms are standard graph morphisms. The sets Σ_{node} and Σ_{edge} contain node respectively edge labels. In the following we will say "graph" instead of "graph structure" and "graph morphism" or just "morphism" instead of "graph structure morphism".

A category of graphs and graph morphisms has all pushouts and pullbacks, which can be constructed componentwise in the category **Set**. Furthermore, constructing the pushout or pullback of two injective morphisms always gives us two injective morphisms. Working exclusively in the category of injective morphisms is not possible since this category does not have all pushouts and

pullbacks, which is due to missing non-injective mediating morphisms. So far we can obtain our main result (Theorem 7) only if we work with injective morphisms, which is, however, a natural requirement.

Definition 2 (Graph Transformation System). *A* rule *or* production *is a pair* $(\varphi_L\colon I \to L, \varphi_R\colon I \to R)$ *of injective graph morphisms. It can be applied to a graph* G*, resulting in a graph* H*, if there is an injective match morphism* $\varphi\colon L \to G$ *and we can find a graph* C *and morphisms such that the two squares in the following diagram are both pushouts.*

$$\begin{tikzcd} L \arrow[d, "\varphi"'] & I \arrow[l, "\varphi_L"'] \arrow[r, "\varphi_R"] \arrow[d] & R \arrow[d] \\ G & C \arrow[l] \arrow[r] & H \end{tikzcd}$$

A graph transformation system *is a set* $\mathcal{P}$ *of productions.*

The diagram above consisting of two pushouts has led to the name double-pushout or DPO approach. The intuition behind this approach is to find a left-hand side L in a graph G, remove L apart from the interface I and to attach R to the interface in the remaining graph C, resulting in H.

Note: Instead of writing $(\varphi_L\colon I \to L, \varphi_R\colon I \to R)$ we will in the following abbreviate a rule by $(L \xleftarrow{\varphi_L} I \xrightarrow{\varphi_R} R)$, or even $(L \leftarrow I \rightarrow R)$ if there is no danger of misunderstanding. This short form will be used for other morphisms as well.

We use a running example throughout the paper which is deliberately kept very simple. Figure 1 shows three spans $L \leftarrow I \rightarrow R$ which form the rule set $\mathcal{P}$ of our example graph transformation system. The graphs are directed graphs with edge labels where nodes are unlabelled (or are labelled with a dummy label). We give rules for a simplex connection S and a duplex connection D over both of which messages M—represented by a loop—are sent. A duplex connection can be used both ways, whereas a simplex connection has a fixed direction. The connections themselves are preserved and are therefore in the interfaces of the rules. An alternative choice, which is also covered by the concept of graph structures, would have been to model this situation by hypergraphs with unary edges (for messages) and binary edges (for connections).

In order to state congruence results, we first need a notion of contexts and contexualization.

Definition 3 (Graphs with interfaces and graph contexts). *A graph* G *with* interface J *is an injective morphism* $J \to G$*. Furthermore a* context *or* cospan *consists of two injective morphisms* $J \to E \leftarrow \overline{J}$*.*

The composition of a graph with interface $J \to G$ *and a context* $J \to E \leftarrow \overline{J}$ *is a graph with interface* $\overline{J} \to \overline{G}$ *which is obtained by constructing* $\overline{G}$ *as the pushout of* $J \to G$ *and* $J \to E$*.*

$$\begin{tikzcd} J \arrow[r] \arrow[d] & E \arrow[d] & \overline{J} \arrow[l] \arrow[dl, dotted] \\ G \arrow[r] & \overline{G} & \end{tikzcd}$$

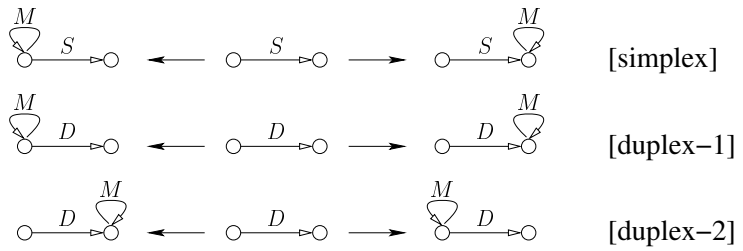

Fig. 1. Rules of a graph transformation system.

Note that composition is defined only up to isomorphism, since the pushout object is unique only up to isomorphism.

This notion of interfaces, contexts and composition is within the spirit of the DPO approach where the pushouts for G and H in Definition 2 can be interpreted as composition of L with C respectively R with C along interface I. In the context of this paper however it is important to consider also the graph G with interface J leading to $\overline{G}$ with interface $\overline{J}$, which requires a context E with two interfaces J and $\overline{J}$. Discrete interfaces, which are a special case, have already been used, see for instance [7].

3 Rewriting with Borrowed Contexts

We are now ready for the central definition of this paper: graph rewriting with borrowed contexts on graphs with interfaces. The underlying idea is to allow not only total, but also partial matches of a left-hand side. The missing part of the left-hand side is then displayed as the label of the resulting transition.

Definition 4 (Rewriting with borrowed contexts). *Let $\mathcal{P}$ be a set of graph productions of the form $(L \leftarrow I \rightarrow R)$ and let $J \rightarrow G$ be a graph with interface. We say that $J \rightarrow G$ reduces to $K \rightarrow H$ with transition label $(J \rightarrow F \leftarrow K)$ whenever there is a production $(L \leftarrow I \rightarrow R) \in \mathcal{P}$ and there are graphs D, G^+, C and additional morphisms such that the following diagram commutes and the squares are either pushouts (PO) or pullbacks (PB) with injective morphisms.*

$$\begin{tikzcd}
D \arrow[r] \arrow[d] \arrow[dr, phantom, "PO"] & L \arrow[d] \arrow[dr, phantom, "PO"] & I \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "PO"] & R \arrow[d] \\
G \arrow[r] & G^+ & C \arrow[l] \arrow[r] & H \\
J \arrow[r] \arrow[u] \arrow[ur, phantom, "PO"] & F \arrow[u] \arrow[ur, phantom, "PB"] & K \arrow[l] \arrow[u] \arrow[ur, dotted] &
\end{tikzcd}$$

Symbolically this is denoted by the transition $(J \to G) \stackrel{J \to F \leftarrow K}{\longrightarrow} (K \to H)$*, which is also called* rewriting step with borrowed context.

The squares in the diagram above have the following meaning: the upper left-hand square merges the left-hand side L and the graph G to be rewritten according to a partial match $G \leftarrow D \to L$ of the left-hand side in G. The resulting graph G^+ contains a total match of L and can be rewritten as in the standard DPO approach, which produces the two remaining squares in the upper row. The pushout in the lower row gives us the borrowed (or minimal) context F which is missing in order to obtain a total match of L, along with a morphism $J \to F$ indicating how F should be attached to G. Finally, we need an interface for the resulting graph H, which can be obtained by "intersecting" the borrowed context F and the graph C via a pullback.

The two pushout complements that are constructed in Definition 4 may not exist. The middle square in the upper row can only be completed if the dangling edge condition is satisfied, i.e., if the left-hand side L is connected to the rest of the graph G^+ exclusively via its interface I and no edges would be left "dangling" by removing it. The left square in the lower row can only be completed if there is a way to extend the partial match to a left-hand side L by attaching some context $J \to F$ to $J \to G$. In other words, the dangling edge condition is required also for the morphism $G \to G^+$ with respect to the interface morphism $J \to G$.

In this case the borrowed context F is minimal in the following sense: Given the partial match $G \leftarrow D \to L$, the pushout G^+ is the minimal graph containing both G and L attached according to the partial match. The borrowed context F is a pushout complement of the injective morphisms $J \to G \to G^+$, leading to the injective morphisms $J \to F \to G^+$. This implies that F is the unique graph (up to isomorphism) that is needed to extend G to the minimal graph G^+.

From the properties of the category of graph structures we can infer that all morphisms in the diagram above are injective. It is thus possible to draw a schematic representation of the four left-hand side squares of Definition 4 (see Figure 2). This figure also illustrates that the new interface K is the "union" of the interfaces I and J, minus the graph components that are internal in either G or L.

In order to illustrate Definition 4, we regard rule [simplex] of Figure 1 and an example graph G consisting of two S-edges for which we find a partial match of the left-hand side. This results in the derivation shown in Figure 3. Note that the image of a node under a morphism is implicitly given by its position, i.e., the left-hand node is always mapped to a left-hand node, analogously for the right-hand node.

4 Bisimilarity is a Congruence

We now arrive at the main theorem of this paper: We will show that the bisimilarity defined on labelled graph transition systems is a congruence. Before that we need two more definitions.

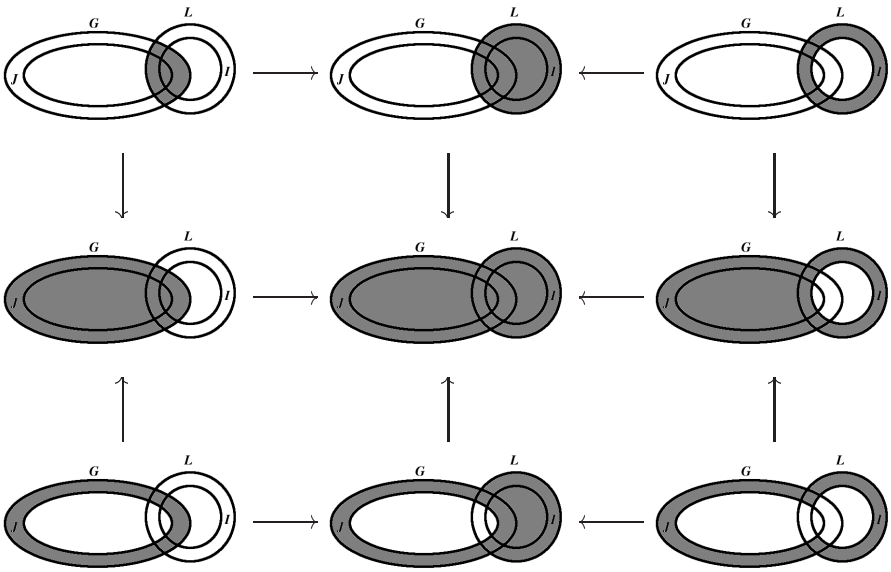

Fig. 2. Graphical represenation of rewriting with borrowed contexts.

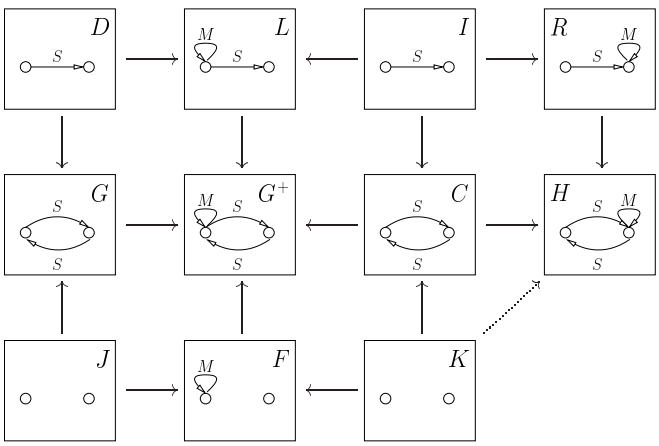

Fig. 3. Rewriting with borrowed contexts in the example graph transformation system.

Definition 5 (Bisimulation and Bisimilarity). *Let $\mathcal{P}$ be a set of productions. Let $\mathcal{R}$ be a symmetric relation containing pairs of graphs with interfaces of the form $(J \to G, J \to G')$, also written $(J \to G)\,\mathcal{R}\,(J \to G')$.*

The relation $\mathcal{R}$ is a bisimulation *if whenever we have $(J \to G)\,\mathcal{R}\,(J \to G')$ and a transition $(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ (in words: $J \to G$ reduces to $K \to H$ with transition label $J \to F \leftarrow K$) can be derived from $\mathcal{P}$, then there exists a morphism $K \to H'$ and a transition $(J \to G') \xrightarrow{J \to F \leftarrow K} (K \to H')$ with the same transition label $J \to F \leftarrow K$ 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 morphisms. The relation $\sim$ is called bisimilarity.

In order to state Theorem 7, we have to be able to close a bisimulation or simply a relation under all possible contexts.

Definition 6 (Closure under Contextualization). *Let $\mathcal{R}$ be a relation on graphs with interfaces as in Definition 5. By $\hat{\mathcal{R}}$ we denote the closure of $\mathcal{R}$ under contextualization, i.e., $\hat{\mathcal{R}}$ is the smallest relation that contains, for every pair $(J \to G, J \to G') \in \mathcal{R}$ and for every context of the form $J \to E \leftarrow \overline{J}$, the pair of morphisms $(\overline{J} \to \overline{G}, \overline{J} \to \overline{G}')$ which results from the composition of $J \to G$ and $J \to E \leftarrow \overline{J}$ respectively $J \to G'$ and $J \to E \leftarrow \overline{J}$.*

A relation $\mathcal{R}$ is a congruence, i.e., closed under contexts whenever $\hat{\mathcal{R}} = \mathcal{R}$. Since obviously $\mathcal{R}$ is contained in $\hat{\mathcal{R}}$, it suffices to show $\hat{\mathcal{R}} \subseteq \mathcal{R}$. We only give a proof sketch, the full proof can be found in [6].

Theorem 7 (Bisimilarity is a Congruence). *Whenever $\mathcal{R}$ is a bisimulation, then $\hat{\mathcal{R}}$ is a bisimulation as well. This implies that the bisimilarity relation $\sim$ is a congruence.*

Proof (Proof Sketch).

Remark: In this proof we are using properties of the category of graph structures, such as pushout complement splitting and special decomposition properties, that do not necessarily hold in other categories (cf. the remarks at the end of the introduction).

We will show that whenever $\mathcal{R}$ is a bisimulation, then $\hat{\mathcal{R}}$ is a bisimulation as well. With the following argument we can then 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 $(\overline{J} \to \overline{G}) \sim (\overline{J} \to \overline{G}')$.

So let $\mathcal{R}$ be a bisimulation and let $(\overline{J} \to \overline{G})\,\hat{\mathcal{R}}\,(\overline{J} \to \overline{G}')$. We assume that

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

Our goal is to show that there exists a transition

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

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 1 we will construct a transition $(J \to G) \overset{J \to F \leftarrow K}{\longrightarrow} (K \to H)$ which implies a transition $(J \to G') \overset{J \to F \leftarrow K}{\longrightarrow} (K \to H')$ with $(K \to H)\,\mathcal{R}\,(K \to H')$, since $\mathcal{R}$ is a bisimulation. In Step 2 we will extend the second transition to obtain the transition stated in our goal above.

Step 1: Our first assumption $(\overline{J} \to \overline{G})\,\hat{\mathcal{R}}\,(\overline{J} \to \overline{G}')$ means that there is some 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 $\overline{J} \to \overline{G}'$ can be obtained by composing $J \to G$ and $J \to G'$ with this context. The second assumption is the transition $(\overline{J} \to \overline{G}) \overset{\overline{J} \to \overline{F} \leftarrow \overline{K}}{\longrightarrow} (\overline{K} \to \overline{H})$ which leads to the situation depicted in Diagram (1), where the decomposition of $\overline{J} \to \overline{G}$ is shown explicitly and all morphisms are injective and all (basic) squares are pushouts, apart from the square $\overline{K}$, $\overline{C}$, $\overline{F}$, $\overline{G}^+$, which is a pullback.

$$
\begin{tikzcd}
 & \overline{D} \arrow[r] \arrow[d] & L \arrow[d] & I \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\
G \arrow[r] & \overline{G} \arrow[r] & \overline{G}^+ & \overline{C} \arrow[l] \arrow[r] & \overline{H} \\
J \arrow[r] \arrow[u] & E \arrow[u] & & & \\
 & \overline{J} \arrow[r] \arrow[u] & \overline{F} \arrow[uu] & \overline{K} \arrow[l] \arrow[uu] \arrow[uur, dotted] &
\end{tikzcd}
\quad (1)
$$

$$
\begin{tikzcd}
 & \overline{D} \arrow[r] \arrow[d] & L \arrow[d] & I \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\
G \arrow[r] & \overline{G} \arrow[r] & \overline{G}^+ & \overline{C} \arrow[l] \arrow[r] & \overline{H} \\
J \arrow[r] \arrow[u] & E \arrow[r] \arrow[u] & E_2 \arrow[u] & E_1 \arrow[l] \arrow[u] \arrow[ur, dotted] & \\
 & \overline{J} \arrow[r] \arrow[u] & \overline{F} \arrow[u] & \overline{K} \arrow[l] \arrow[u] \arrow[uur, dotted] &
\end{tikzcd}
\quad (2)
$$

We can now split the lower pushout and the lower pullback along E (see Diagram (2)).

As a next step we construct D as the pullback of $G \to \overline{G}$ and $\overline{D} \to \overline{G}$, followed by the construction of $\tilde{G}$ as the pushout of the resulting morphisms. In Diagram (2) we can now split the upper row of pushouts and the pushout to the very left, obtaining the graphs F_1, G^+, C and H. We then construct F as the pullback of $G^+ \to \overline{G}^+$ and $E_2 \to \overline{G}^+$ and K as pullback of the morphisms $C \to \overline{C}$ and $E_1 \to \overline{C}$. This results in Diagram (3), with two commuting cubes in the middle of the diagram.

All morphisms are injective, all squares commute and we can infer that the squares D, G, L, G^+ and I, L, C, G^+ and J, G, F, G^+ and I, R, C, H are pushouts and the square K, C, F, G^+ is a pullback, as in Definition 4. Hence, from Diagram (3) we can derive the following transition:

$$(J \to G) \overset{J \to F \leftarrow K}{\longrightarrow} (K \to H),$$

using the notation of Definition 4. Since $\mathcal{R}$ is a bisimulation, this implies

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

with $(K \to H)\,\mathcal{R}\,(K \to H')$. Furthermore we can infer from Diagram (3) that $\overline{K} \to \overline{H}$ can be obtained by composing $K \to H$ with the context $K \to E_1 \leftarrow \overline{K}$, since the square $K, H, E_1, \overline{H}$ is a pushout.

$$
\begin{tikzcd}
D \arrow[rr] \arrow[dd] & & \overline{D} \arrow[rr] \arrow[dl] \arrow[dd] & & L \arrow[dl] \arrow[dd] & & I \arrow[ll] \arrow[rr] \arrow[dl] \arrow[dd] & & R \arrow[dl] \arrow[dd] \\
& \tilde{G} \arrow[rr] \arrow[dr] & & G^{+} \arrow[ll] \arrow[dr] & & C \arrow[rr] \arrow[dr] & & H \arrow[dr] & \\
G \arrow[rr] \arrow[ur] & & \overline{G} \arrow[rr] \arrow[ur] & & \overline{G}^{+} \arrow[ll] & & \overline{C} \arrow[ll] \arrow[rr] \arrow[ur, dotted] & & \overline{H} \\
& F_1 \arrow[rr] \arrow[dr] & & F \arrow[ll] \arrow[dr] & & K \arrow[ll] \arrow[rr] \arrow[dr] & & E_1 \arrow[uu, dotted] & \\
J \arrow[rr] \arrow[ur] \arrow[uu] & & E \arrow[rr] \arrow[uu] & & E_2 \arrow[ll] & & E_1 \arrow[ll] \arrow[uu] & & \\
& & \overline{J} \arrow[rr] \arrow[uu] & & \overline{F} \arrow[ll] \arrow[uu] & & \overline{K} \arrow[ll] \arrow[uu] \arrow[uuuuuu, dotted] & &
\end{tikzcd}
\tag{3}
$$

Step 2: We will now extend the transition from $J \to G'$ to $K \to H'$ with $(K \to H)\,\mathcal{R}\,(K \to H')$ obtained above to construct a transition from $(\overline{J} \to \overline{G}')$ to $(\overline{K} \to \overline{H}')$ with $(\overline{K} \to \overline{H})\,\hat{\mathcal{R}}\,(\overline{K} \to \overline{H}')$. We will construct $\overline{K} \to \overline{H}'$ in such a way that it is the composition of $K \to H'$ with the context $K \to E_1 \leftarrow \overline{K}$. Recall also that $\overline{J} \to \overline{G}'$ is the composition of $J \to G'$ and the context $J \to E \leftarrow \overline{J}$.

We now cut away the upper layer of Diagram (3) and we obtain Diagram (4) where all squares are pushouts, apart from the square $\overline{K}, \overline{F}, E_1, E_2$, which is a pullback.

$$
\begin{tikzcd}
& F_1 \arrow[rr] \arrow[dr] & & F \arrow[ll] \arrow[dr] & & K \arrow[ll] \arrow[dr] & \\
J \arrow[rr] \arrow[ur] & & E \arrow[rr] & & E_2 \arrow[ll] & & E_1 \arrow[ll] \\
& & \overline{J} \arrow[rr] \arrow[uu] & & \overline{F} \arrow[ll] \arrow[uu] & & \overline{K} \arrow[ll] \arrow[uu]
\end{tikzcd}
\tag{4}
$$

From the derivation step of $J \to G'$ given earlier one can derive Diagram (5) for some rule $L' \leftarrow I' \to R'$ where the lower right-hand square is a pullback and all other squares are pushouts. The morphism $J \to F$ is split by F_1 and therefore we can split the two left-hand side pushouts as shown in Diagram (6).

$$
\begin{tikzcd}
D' \arrow[r] \arrow[d] & L' \arrow[d] & I' \arrow[l] \arrow[r] \arrow[d] & R' \arrow[d] \\
G' \arrow[r] & G'^{+} & C' \arrow[l] \arrow[r] & H' \\
J \arrow[r] \arrow[u] & F \arrow[u] & K \arrow[l] \arrow[u] \arrow[ur, dotted] &
\end{tikzcd}
\tag{5}
$$

$$
\begin{tikzcd}
D' \arrow[r] \arrow[d] & \overline{D}' \arrow[r] \arrow[d] & L' \arrow[d] & I' \arrow[l] \arrow[r] \arrow[d] & R' \arrow[d] \\
G' \arrow[r] & \tilde{G}' \arrow[r] & G'^{+} & C' \arrow[l] \arrow[r] & H' \\
J \arrow[r] \arrow[u] & F_1 \arrow[r] \arrow[u] & F \arrow[u] & K \arrow[l] \arrow[u] \arrow[ur, dotted] &
\end{tikzcd}
\tag{6}
$$

We compose Diagrams (4) and (6) and construct the graph $\overline{G}'$ as the pushout of $F_1 \to \tilde{G}'$ and $F_1 \to E$, the graph $\overline{G}'^{+}$ as the pushout of $F \to G'^{+}$ and $F \to E_2$ and the graph $\overline{C}'$ as pushout of $K \to C'$ and $K \to E_1$. This results in Diagram (7), which is identical to Diagram (3) in structure.

$$
\begin{tikzcd}
D' \arrow[rr] \arrow[dd] & & \overline{D}' \arrow[rr] \arrow[dl] \arrow[dd] & & L' \arrow[dl] \arrow[dd] & & I' \arrow[ll] \arrow[rr] \arrow[dl] \arrow[dd] & & R' \arrow[dl] \arrow[dd] \\
& \tilde{G}' \arrow[rr] \arrow[dr] & & G'^{+} \arrow[ll] \arrow[dr] & & C' \arrow[rr] \arrow[dr] & & H' \arrow[dr] & \\
G' \arrow[rr] \arrow[ur] & & \overline{G}' \arrow[rr] & & \overline{G}'^{+} \arrow[ll] & & \overline{C}' \arrow[ll] \arrow[rr] & & \overline{H}' \\
& F_1 \arrow[rr] \arrow[dr] \arrow[uu] & & F \arrow[ll] \arrow[dr] \arrow[uu] & & K \arrow[ll] \arrow[dr] \arrow[uu] \arrow[urr, dotted] & & & \\
J \arrow[rr] \arrow[ur] \arrow[uu] & & E \arrow[rr] \arrow[uu] & & E_2 \arrow[ll] \arrow[uu] & & E_1 \arrow[ll] \arrow[uu] \arrow[uurr, dotted] & & \\
& & \overline{J} \arrow[rr] \arrow[uu] & & \overline{F} \arrow[ll] \arrow[uu] & & \overline{K} \arrow[ll] \arrow[uu] \arrow[uuurr, dotted] & &
\end{tikzcd}
\tag{7}
$$

In Diagram (7), the three right-hand squares in the upper row are all pushouts, the square $\overline{J}, \overline{F}, \overline{G}', \overline{G}'^{+}$ is a pushout and the square $\overline{K}, \overline{F}, \overline{C}', \overline{G}'^{+}$ is a pullback.

Hence, by Definition 4 we infer that

$$(\overline{J} \to \overline{G}') \overset{\overline{J} \to \overline{F} \leftarrow \overline{K}}{\longrightarrow} (\overline{K} \to \overline{H}'),$$

and since the square $K, H', E_1, \overline{H}'$ is also a pushout we can infer that $\overline{K} \to \overline{H}'$ can be obtained by composing $K \to H'$ and the context $K \to E_1 \leftarrow \overline{K}$. From earlier considerations we know that $\overline{K} \to \overline{H}$ is the composition of $K \to H$ with $K \to E_1 \leftarrow \overline{K}$ and hence $(\overline{K} \to \overline{H})\, \hat{\mathcal{R}}\, (\overline{K} \to \overline{H}')$. This means that we have achieved our goal stated at the beginning of the proof sketch, which implies that $\hat{\mathcal{R}}$ is a bisimulation and $\sim$ is a congruence.

5 Borrowed Contexts at Work: An Example

In order to further pursue the example we will first introduce a proof technique simplifying bisimilarity proofs. This technique is a straightforward instance of an up-to technique [21]. The underlying idea behind the technique is the observation that the relation $\mathcal{R}$ should be as small as possible, in order to obtain a compact proof. This goal can be reached by slightly extending the notion of bisimulation: We now demand that if a transition is matched by another, the pair of resulting graphs can be found in $\mathcal{R}$ after removal of identical contexts. Hence,

this extended notion of bisimulation is called "bisimulation up to context". We first need an auxiliary definition.

Definition 8 (Progression). *Let $\mathcal{R}, \mathcal{S}$ be relations containing pairs of graphs 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 $(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$, there exists a morphism $K \to H'$ such that $(J \to G') \xrightarrow{J \to F \leftarrow K} (K \to H')$ and $(K \to H)\,\mathcal{S}\,(K \to H')$.*

For example, $\mathcal{R}$ is a bisimulation if and only if $\mathcal{R} \rightarrowtail \mathcal{R}$.

Definition 9 (Bisimulation up to Context). *Let $\mathcal{R}$ be a symmetric relation containing pairs of graphs with interfaces of the form $(J \to G, J \to G')$.*

If $\mathcal{R} \rightarrowtail \hat{\mathcal{R}}$, then $\mathcal{R}$ is called bisimulation up to context.

We will show in Proposition 10 that every bisimulation up to context is contained in the bisimilarity $\sim$. The attractiveness of bisimulations up to context stems from the fact that such a relation can be much smaller than the least bisimulation that contains it and thus proofs can be compressed. This technique might even allow us to work with a finite relation instead of an infinite one.

Proposition 10 (Bisimulation up to Context implies Bisimilarity). *Let $\mathcal{R}$ be a bisimulation up to context. Then it holds that $\mathcal{R} \subseteq \sim$.*

Proof (Proof Sketch). By carefully examining the proof of Theorem 7 again we can see that some simple modifications give us the following (stronger) theorem:

If $\mathcal{R} \rightarrowtail \mathcal{S}$, then also $\hat{\mathcal{R}} \rightarrowtail \hat{\mathcal{S}}$.

Since $\mathcal{R}$ is a bisimulation up to context we have $\mathcal{R} \rightarrowtail \hat{\mathcal{R}}$. The stronger version of Theorem 7 now implies $\hat{\mathcal{R}} \rightarrowtail \widehat{(\hat{\mathcal{R}})}$. Since the composition of contexts is associative we have $\widehat{(\hat{\mathcal{R}})} = \hat{\mathcal{R}}$, which implies $\hat{\mathcal{R}} \rightarrowtail \hat{\mathcal{R}}$ and hence that $\hat{\mathcal{R}}$ is a bisimulation, i.e., $\hat{\mathcal{R}} \subseteq \sim$. This implies $\mathcal{R} \subseteq \hat{\mathcal{R}} \subseteq \sim$.

Since contextualization is defined only up to isomorphism, we can assume that $\hat{\mathcal{R}}$ is closed under isomorphism in the following sense: For every span $G \leftarrow J \to G'$, all isomorphic spans $\tilde{G} \leftarrow \tilde{J} \to \tilde{G}'$ are also contained in $\hat{\mathcal{R}}$.

Similarly, we can restrict ourselves to abstract transitions when checking for bisimilarity: Assume that $(J \to G)\,\mathcal{R}\,(J \to G')$ and there are two transitions

$$(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H) \quad \text{and} \quad (J \to G) \xrightarrow{J \to \tilde{F} \leftarrow \tilde{K}} (\tilde{K} \to \tilde{H})$$

with isomorphisms from $\tilde{F}, \tilde{K}, \tilde{H}$ to F, K, H respectively such that the entire diagram commutes (see Diagram (8)). It is sufficient to show the existence of a transition $(J \to G') \xrightarrow{J \to F \leftarrow K} (K \to H')$ such that $(K \to H)\,\hat{\mathcal{R}}\,(K \to H')$. From this transition and Diagram (8) we can derive Diagram (9), where the arrows pointing upwards are isomorphisms and the diagram commutes. In such

a situation we can infer the existence of a transition $(J \to G') \overset{J \to \tilde{F} \leftarrow \tilde{K}}{\longrightarrow} (\tilde{K} \to \tilde{H}')$ such that $H \leftarrow K \to H'$ and $\tilde{H} \leftarrow \tilde{K} \to \tilde{H}'$ are isomorphic spans, from which it follows that $(\tilde{K} \to \tilde{H}) \,\hat{\mathcal{R}}\, (\tilde{K} \to \tilde{H}')$.

$$\begin{tikzcd}
 & F & K \arrow[l] \arrow[r] & H \\
G & J \arrow[l] \arrow[ur] \arrow[dr] \arrow[u, phantom, "\wr"] & \arrow[u, phantom, "\wr"] & \arrow[u, phantom, "\wr"] \\
 & \tilde{F} \arrow[uu] & \tilde{K} \arrow[l] \arrow[r] \arrow[uu] & \tilde{H} \arrow[uu]
\end{tikzcd} \quad (8)$$

$$\begin{tikzcd}
 & F & K \arrow[l] \arrow[r] & H' \\
G' & J \arrow[l] \arrow[ur] \arrow[dr] \arrow[u, phantom, "\wr"] & \arrow[u, phantom, "\wr"] & \arrow[u, phantom, "\wr"] \\
 & \tilde{F} \arrow[uu] & \tilde{K} \arrow[l] \arrow[r] \arrow[uu] & \tilde{H}' \arrow[uu]
\end{tikzcd} \quad (9)$$

We now show how to exploit this proof technique and prove that two graphs are bisimilar. We assume that the set $\mathcal{P}$ of rules depicted in Figure 1 is given and we consider the two graphs with interfaces of Figure 4.

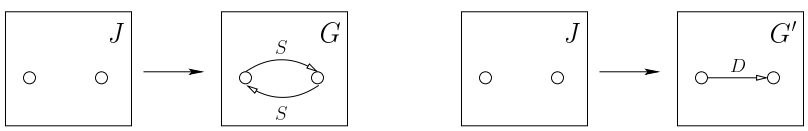

Fig. 4. Two graphs with interfaces which are bisimilar.

We consider the symmetric relation

$$\mathcal{R} = \{(J \to G, J \to G'), (J \to G', J \to G)\}$$

and we will show that it is a bisimulation up to context. For each of the three rules there are several partial matches for both G and G'. Most of these matches are not very interesting, since the graph to be rewritten and the left-hand side overlap only in their interfaces, but the corresponding transitions have to be checked nevertheless. (We discuss possible simplifications in the conclusion.)

In order to give a general idea, we consider only two transitions of $J \to G$ in detail, where both are instances of rule [simplex]. These two transitions will be written

$$(J \to G) \overset{J \to F_i \leftarrow K_i}{\longrightarrow} (K_i \to H_i),$$

where $i = 1, 2$. The graphs F_i, K_i, H_i and the corresponding morphisms are depicted in Figure 5. Note that we have already shown how to derive the first transition of $J \to G$ in Figure 3 (where $F_1 = F$, $K_1 = K$, $H_1 = H$).

In order to show that $\mathcal{R}$ is a bisimulation up to context, we have to find matching transitions

$$(J \to G') \overset{J \to F_i \leftarrow K_i}{\longrightarrow} (K_i \to H'_i)$$

for $i = 1, 2$, such that $(K_i \to H_i) \,\hat{\mathcal{R}}\, (K_i \to H'_i)$. Such transitions can be derived and the graphs H'_i with their corresponding morphisms are also depicted in

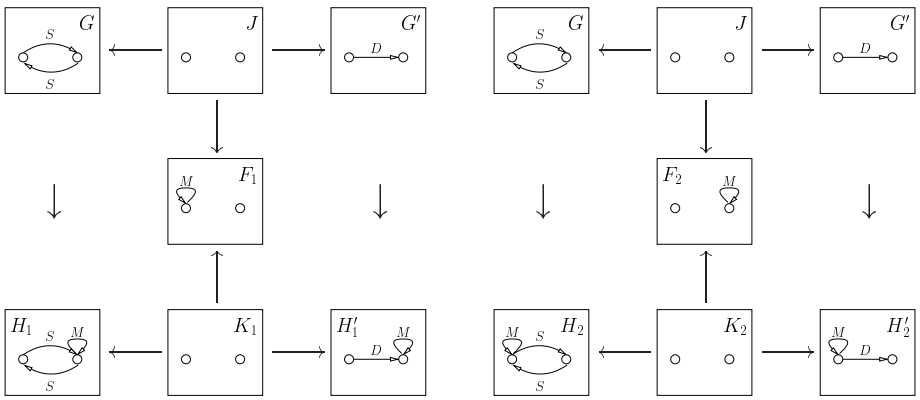

Fig. 5. Matching transitions of bisimilar graphs.

Figure 5. Note that the first transition is an instance of rule [duplex-1], while the second transition is an instance of rule [duplex-2].

Furthermore, it holds that $(K_i \to H_i)\,\hat{\mathcal{R}}\,(K_i \to H'_i)$ for $i = 1, 2$, since these graphs can be obtained by composing $J \to G$ respectively $J \to G'$ with a context consisting of two nodes and a looping M-edge. After checking also the remaining transitions we can conclude $(J \to G) \sim (J \to G')$ from Proposition 10.

This means that in every context we can replace a duplex connection by two simplex connections and vice versa. Even this small example shows us that in order to obtain a bisimilarity result, proof techniques are needed in order to keep $\mathcal{R}$ finite. Otherwise we would have to deal with a relation containing infinitely many elements.

6 Conclusion

We have presented a way to derive labelled transitions and bisimulation congruences for graph transformation systems. It is our hope that this work will be helpful for the transfer of concepts from the world of process algebras to the world of graph rewriting and vice versa. We believe that having graphs as objects (and graph morphisms as arrows) instead of having graphs as arrows is useful for tracking graph components and thus enables us to easily state which components are associated with each other in different graphs. Hence we need not consider explicit names for graph components.

We have made some investigations concerning the adaptation of the concept of relative pushouts for cospans of graphs. However, there are fundamental problems, mainly caused by graphs having non-trivial automorphisms (see, e.g., the counterexample in [13] on pages 80/81, which can be directly transferred into our framework). We believe, however, that our construction is very close in spirit to the notion of relative pushouts introduced by Leifer and Milner and that it

should be possible to show the equivalence of these two notions in a suitable graph category with support.

Our results do not only hold in graph structure categories, but also in other categories which satisfy certain properties typical for the categories of sets and high-level replacement systems [4]. In this context it is also interesting to point out that most of the categorical properties we need hold already in adhesive categories [12].

In the future we also plan to address the following two questions: How should weak bisimilarity be defined and is it a congruence? Do our results still hold if we allow for non-injective morphisms? Furthermore we plan to introduce more proof techniques in order to simplify bisimulation proofs. One such technique is clearly suggested by the example in Section 5. Whenever a graph and a left-hand side overlap only in their interfaces, another graph with the same interface will certainly be able to match the corresponding rewriting step with borrowed context, since this step only changes the interface itself. Hence it should be possible to remove some superfluous transitions without changing the underlying bisimilarity.

Another interesting question would be to find out which bisimulation congruences are produced by the various encodings of π-calculus into graph rewriting and to see in what way they are related to existing congruences for this calculus. It also remains to determine in what way our bisimilarity is related to dynamic bisimulation as presented in [1, 18].

Acknowledgements: We would like to thank the anonymous referees for their helpful comments. We are especially grateful to Robin Milner for suggesting this research topic and for sharing his ideas with us.

References

1. Roberto Bruni, Ugo Montanari, and Vladimiro Sassone. Open ended systems, dynamic bisimulation and tile logic. In *Proc. of IFIP TCS 2000*. Springer, 2000. LNCS 1872.
2. A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic approaches to graph transformation—part I: Basic concepts and double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, chapter 3. World Scientific, 1997.
3. H. Ehrig. Bigraphs meet double pushouts. *EATCS Bulletin*, 78:72–85, October 2002.
4. H. Ehrig, M. Gajewsky, and F. Parisi-Presicce. High-level replacement systems with applications to algebraic specifications and Petri nets. In H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.3: Concurrency, Parallellism, and Distribution*, chapter 6, pages 341–400. World Scientific, 1999.
5. H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, and A. Corradini. Algebraic approaches to graph transformation—part II: Single pushout approach and comparison with double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, chapter 4. World Scientific, 1997.

6. Hartmut Ehrig and Barbara König. Deriving bisimulation congruences in the DPO approach to graph rewriting. Technical report, Universität Stuttgart, 2004. to appear.
7. F. Gadducci and R. Heckel. An inductive view of graph transformation. In *Recent Trends in Algebraic Development Techniques, 12th International Workshop, WADT '97*, pages 223–237. Springer-Verlag, 1997. LNCS 1376.
8. F. Gadducci and U. Montanari. A concurrent graph semantics for mobile ambients. In S. Brookes and M. Mislove, editors, *Proceedings of the 17th MFPS*, volume 45 of *Electronic Notes in Computer Science*. Elsevier Science, 2001.
9. F. Gadducci and U. Montanari. Comparing logics for rewriting: Rewriting logic, action calculi and tile logic. *Theoretical Computer Science*, 285(2):319–358, 2002.
10. Ole Høgh Jensen and Robin Milner. Bigraphs and transitions. In *Proc. of POPL 2003*, pages 38–49. ACM, 2003.
11. Barbara König. A graph rewriting semantics for the polyadic π-calculus. In *Workshop on Graph Transformation and Visual Modeling Techniques (Geneva, Switzerland), ICALP Workshops '00*, pages 451–458. Carleton Scientific, 2000.
12. Stephen Lack and Pawel Sobocinski. Adhesive categories. In *Proc. of FOSSACS '04*, LNCS. Springer, 2004. to appear.
13. James J. Leifer. *Operational congruences for reactive systems*. PhD thesis, University of Cambridge Computer Laboratory, September 2001.
14. James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In *Proc. of CONCUR 2000*, 2000. LNCS 1877.
15. Saunders Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.
16. Robin Milner. Bigraphical reactive systems. In *Proc. of CONCUR '01*, pages 16–35. Springer-Verlag, 2001. LNCS 2154.
17. Robin Milner and Davide Sangiorgi. Barbed bisimulation. In *Proc. of ICALP '92*. Springer-Verlag, 1992. LNCS 623.
18. U. Montanari and V. Sassone. Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta Informaticae*, 16:171–196, 1992.
19. Ugo Montanari and Marco Pistore. Concurrent semantics for the π-calculus. *Electronic Notes in Theoretical Computer Science*, 1, 1995.
20. Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, volume 1. World Scientific, 1997.
21. Davide Sangiorgi. On the proof method for bisimulation. In *Proc. of MFCS '95 (Mathematical Foundations of Computer Science)*, pages 479–488. Springer, 1995. LNCS 969.
22. Davide Sangiorgi and David Walker. *The π-calculus—A Theory of Mobile Processes*. Cambridge University Press, 2001.
23. Vladimiro Sassone and Pawel Sobocinski. Deriving bisimulation congruences: 2-categories vs precategories. In *Proc. of FoSSaCS 2003*, pages 409–424, 2003.
24. Peter Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science*, 274(1–2):183–230, 2002.