

# Composition and Decomposition of DPO Transformations with Borrowed Context<sup>\*</sup>

Paolo Baldan<sup>1</sup>, Hartmut Ehrig<sup>2</sup>, and Barbara König<sup>3</sup>

<sup>1</sup> Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy

<sup>2</sup> Institut für Softwaretechnik und Theoretische Informatik,  
Technische Universität Berlin, Germany

<sup>3</sup> Institut für Informatik und interaktive Systeme, Universität Duisburg-Essen,  
Germany

**Abstract.** Double-pushout (DPO) transformations with borrowed context extend the standard DPO approach by allowing part of the graph needed in a transformation to be borrowed from the environment. The bisimilarity based on the observation of borrowed contexts is a congruence, thus facilitating system analysis. In this paper, focusing on the situation in which the states of a global system are built out of local components, we show that DPO transformations with borrowed context defined on a global system state can be decomposed into corresponding transformations on the local states and vice versa. Such composition and decomposition theorems, developed in the framework of adhesive categories, can be seen as a first step towards an inductive definition, in SOS style, of the labelled transition system associated to a graph transformation system. As a special case we show how an ordinary DPO transformation on a global system state can be decomposed into local DPO transformations with borrowed context using the same production.

## 1 Introduction

Graph transformations [7] have been applied successfully to several areas of software and system engineering, including syntax and semantics of visual languages, visual modelling of behaviour and programming, metamodeling and model transformation, refactoring of models and programs. Almost invariably the underlying idea is the same: the states of a system are modelled by suitable graphs and state changes are represented by graph transformations. Consequently, the behaviour of the system is expressed by a transition system, where states are reachable graphs and transitions are induced by graph transformations. The transition system can be the basis for defining various notions of abstract behavioural equivalences, e.g., trace, failures and bisimulation equivalence. These, in turn, can be used to provide a solid theoretical justification for various constructions and techniques in the above mentioned areas of system

---

<sup>\*</sup> Research partially supported by the EC RTN 2-2001-00346 Project SEGRAVIS, the MIUR Project ART, the DFG project SANDS and CRUI/DAAD VIGONI “Models based on Graph Transformation Systems: Analysis and Verification”.

engineering, e.g., for the formalisation of behavioural refinement, or to show semantical correctness of refactoring and model transformation.

The applicability of these techniques generally requires the considered behavioural equivalence to be a congruence: two systems—seen as equivalent from the point of view of an external observer—must be equivalent also in all possible contexts or environments.

Unfortunately, behavioural equivalences defined over unlabelled transition systems naively generated by using transformation rules often fail to be congruences. The same problem arises for several other computational formalisms which can be naturally endowed with an operational semantics based on unlabelled reductions, such as the  $\lambda$ -calculus [2] or many process calculi with mobility or name passing, e.g., the  $\pi$ -calculus [11] or the ambient calculus [4].

In order to overcome this problem recently there has been a lot of interest in the automatic derivation of labelled transition systems where bisimilarity is a congruence for reactive systems endowed with an (unlabelled) reduction semantics (see, e.g., [10,8,6,12]). In particular, in the case of double-pushout (DPO) graph rewriting this has led to an extension of the approach, called DPO approach with borrowed contexts [6]. Intuitively a label  $C$  of a transition represents the (minimal) context that must be “added” to the current state in order to allow the transformation or reduction step to be performed.

In this paper, we focus on the situation in which the states of a global system are built out of local states of the components of the systems. Then we show that DPO transformations with borrowed context defined on a global system state can be decomposed into corresponding transformations on the local states. Vice versa we study the conditions under which local transformations can be composed to yield global ones. The main results of this paper are composition and decomposition theorems for DPO transformations with borrowed context in the framework of rewriting systems over adhesive categories [9]. As a special case we show how an ordinary DPO transformation on a global system state can be decomposed into local DPO transformations with borrowed context using the same production.

These composition and decomposition results can be seen as a first step towards a structural operational semantics for adhesive rewriting systems, i.e., towards a framework where the transition system associated to a graph transformation system can be defined inductively, in SOS style. Compare for instance the inductive CCS rule stating that from  $P \xrightarrow{a} P'$  and  $Q \xrightarrow{\bar{a}} Q'$  (where  $a$  is an action and  $\bar{a}$  its corresponding coaction) one can derive  $P \mid Q \xrightarrow{\tau} P' \mid Q'$  (where the label  $\tau$  stands for a silent transition). Intuitively  $P \xrightarrow{a} P'$  means that  $P$  can move to  $P'$  if the environment performs an output on channel  $a$  and, similarly,  $Q$  can move if the environment performs an input on  $a$ . The two local moves can be combined leading to a transition for  $P \mid Q$  where nothing is “borrowed” from the environment (as expressed by the  $\tau$ -label).

Having an inductive way of specifying the behaviour of a graph can lead to a new understanding of system semantics and new proof techniques. E.g., inductive definitions can be quite useful when comparing the semantics of two calculi, as in [3].

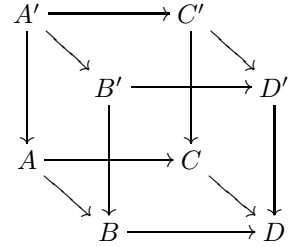
The rest of the paper is structured as follows. In Section 2 we introduce the basics of adhesive categories and of the DPO approach with borrowed contexts. In Section 3 we introduce a category of transformations with borrowed contexts, which is the basis for the formalisation of the composition and decomposition theorems for transformations given in Sections 4 and 5, respectively. Finally, in Section 6 we conclude and outline directions of future research. Proofs of all theorems, propositions and lemmas can be found in [1].

## 2 DPO Transformation with Borrowed Contexts

Adhesive categories have been introduced in [9], as categories where pushouts along monomorphisms are so-called Van-Kampen squares (see Condition 3 in the definition below). We will only briefly sketch the theory of adhesive categories.

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

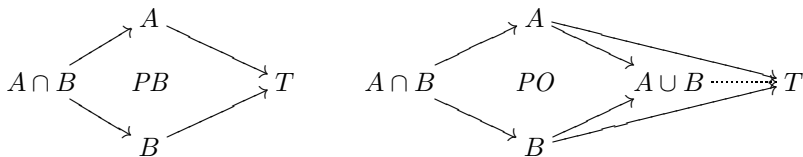
1.  $\mathbf{C}$  has pushouts along monos;
2.  $\mathbf{C}$  has pullbacks;
3. *Given a cube diagram as shown on the right with: (i)  $A \rightarrow C$  mono, (ii) the bottom square a pushout and (iii) the left and back squares pullbacks, we have that the top square is a pushout iff the front and right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



The category **Set** of sets and functions is adhesive. Adhesive categories enjoy closure properties, for instance if  $\mathbf{C}$  is adhesive then so is any functor category  $\mathbf{C}^{\mathbf{X}}$ , any slice category  $\mathbf{C} \downarrow C$  and any co-slice category  $C \downarrow \mathbf{C}$ . Therefore, since the category of graphs and graph morphisms is a functor category  $\mathbf{Graph} \cong \mathbf{Set}^{\bullet \leftarrow \bullet}$ , it is adhesive.

A subobject of a given object  $T$  is an isomorphism class of monomorphisms to  $T$ . Binary intersections of subobjects exist in any category with pullbacks. In adhesive categories also binary unions of subobjects exist and can be obtained by taking the pushout over their intersection. Moreover, the lattice of subobjects is distributive.

**Theorem 2 ([9]).** *For an object  $T$  of an adhesive category  $\mathbf{C}$ , the partially ordered set  $Sub(T)$  of subobjects of  $T$  is a distributive lattice. Given two subobjects  $A, B \in Sub(T)$ , the meet  $A \cap B$  is (the isomorphism class of) their pullback, while the join  $A \cup B$  is (the isomorphism class of) their pushout in  $\mathbf{C}$  over their intersection.*



The following lemma will be useful in the future where we have to show that certain squares in adhesive categories are pullbacks or pushouts. It follows directly from Theorem 2.

**Lemma 3.** *Consider the following diagram where all arrows are mono. The square below is a pullback if and only if  $A = B \cap C$  (all objects are seen as subobjects of  $E$ ). Furthermore the square is a pushout if and only if  $A = B \cap C$  and  $D = B \cup C$ .*

$$\begin{array}{ccccc} A & \longrightarrow & B & & \\ \downarrow & & \downarrow & & \\ C & \longrightarrow & D & \longrightarrow & E \end{array}$$

We next define rewriting with borrowed contexts on objects (e.g., over graphs) with interfaces, as introduced in [6]. Intuitively, the borrowed context is the smallest extra context which must be added to the object being rewritten in order to obtain an occurrence of the left-hand side. The extra context can be added only using the interface.

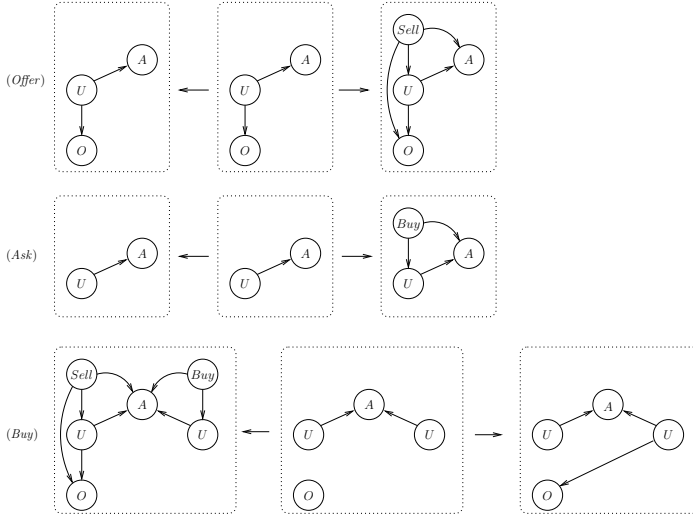
**Definition 4 (Borrowed contexts, transformations).** *Let  $\mathbf{C}$  be a fixed adhesive category and let  $r = (L \leftarrow I \rightarrow R)$  be a rewriting rule. A DPO transformation with borrowed context—short transformation— $t$  (of  $r$ ) is a diagram in  $\mathbf{C}$  of the following form, where all arrows are mono:*

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

In this case we write  $(J \rightarrow G) \xRightarrow{r, m} (K \rightarrow H)$  where  $m = G \leftarrow D \rightarrow L$  is the partial match. If instead we want to focus on the interaction with the environment we say that  $J \rightarrow G$  makes a transition with borrowed context  $J \rightarrow F \leftarrow K$  and becomes  $K \rightarrow H$  (written:  $(J \rightarrow G) \xrightarrow{J \rightarrow F \leftarrow K} (K \rightarrow H)$ ).

For a given transformation  $t_i$  we will denote the objects occurring in the corresponding diagram by  $D_i, G_i, J_i, G_i^+, C_i, H_i, F_i, K_i$ .

The squares in the diagram above have the following meaning: the upper left-hand square merges the left-hand side  $L$  and the object  $G$  to be rewritten according to a partial match  $G \leftarrow D \rightarrow L$  of the left-hand side in  $G$ . The resulting object  $G^+$  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  $J \rightarrow F$  indicating how  $F$  should be attached to  $G$ . Finally, the interface for the



**Fig. 1.** Rewriting system **Market**

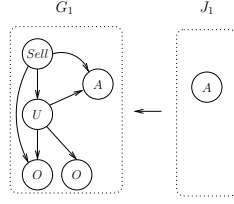
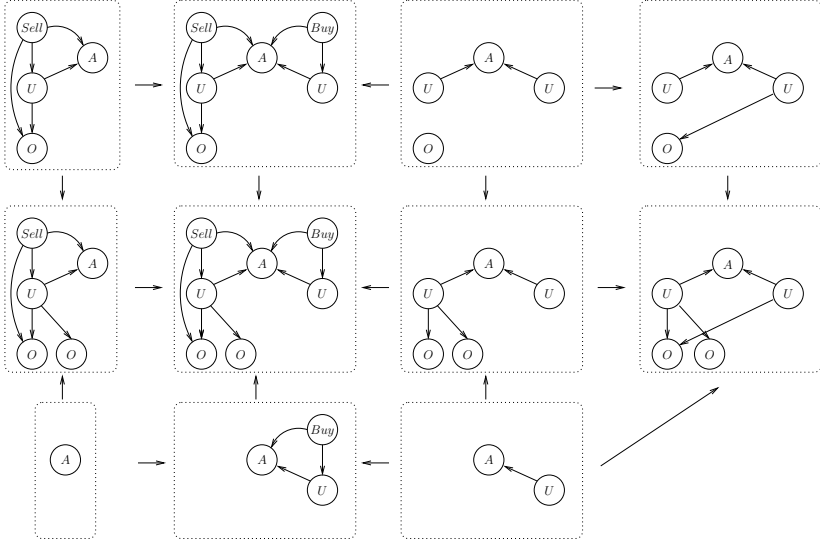
resulting object  $H$  is obtained by “intersecting” the borrowed context  $F$  and the object  $C$  via a pullback. Roughly, the new interface includes what is preserved of the old interface and of the context borrowed from the environment. The two pushout complements that are constructed in Definition 4 may not exist. In this case no rewriting step is possible.

It has been shown in [6] that bisimilarity on the transition system labelled with borrowed contexts is a congruence with respect to cospan composition.

*Example.* Consider the category **Graph** of labelled graphs and label-preserving morphisms. Take the rewriting system **Market** in **Graph** depicted in Fig. 1, which can be interpreted as a very high-level description of the interactions between users of an electronic market place. Graph nodes are represented as circles, with their label inside. Edges are directed and unlabelled. Users, represented as  $U$ -labelled nodes, can possess objects, denoted by  $O$ -labelled nodes, and they can be connected to one (or more) market places, represented by  $A$ -labelled nodes.

A user possessing some objects can autonomously decide to offer one of them to other users, on a market place, expressed by rule *(Offer)*. A user can also ask for something to buy on a market he is connected to, expressed by rule *(Ask)*. A request and an offer, after some negotiation which is not modelled, can meet, the object is sold and moved from the seller to the buyer, modelled by rule *(Buy)*.

An example of a transformation with borrowed context using production *(Buy)* can be found in Fig. 3. It is applied to the graph with interface  $J_1 \rightarrow G_1$  in Fig. 2. The graph  $G_1$  includes a market place  $A$ , with a user  $U$ , possessing two objects and trying to sell one of them. Note that the borrowed context consists of an additional user playing the role of a buyer. In other words, the existence of

**Fig. 2.** The graph with interface  $J_1 \rightarrow G_1$ **Fig. 3.** A transformation with borrowed context  $t_1$  over  $J_1 \rightarrow G_1$ , using rule *(Buy)*

the transformation expresses the fact that rule *(Buy)*, can be applied assuming that the context provides a user which buys the object sold by the user in  $G_1$ .

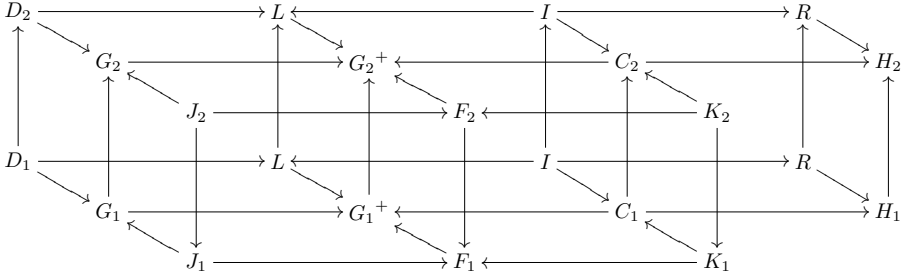
*Remark:* Note that we obtain the well-known case of DPO transformations if we consider total matches  $L \rightarrow G$  instead of partial matches  $G \leftarrow D \rightarrow L$ , which implies  $G = G^+$ . In this case we can take any interface object  $J$ , for instance the initial object—if it exists in the category—which implies that  $F$  and  $K$  are also initial objects.

### 3 Transformation Morphisms

A first step towards the composition of transformations is the formalisation of the intuitive idea of embedding of a transformation into another. This is done by introducing a suitable notion of transformation morphism.

**Definition 5 (Transformation morphisms).** Let  $t_1, t_2$  be two transformations for a fixed rewriting rule  $L \leftarrow I \rightarrow R$ . A transformation morphism  $\theta: t_1 \rightarrow t_2$  consists of arrows  $D_1 \rightarrow D_2, G_1 \rightarrow G_2, G_1^+ \rightarrow G_2^+, C_1 \rightarrow C_2, H_1 \rightarrow H_2, J_2 \rightarrow J_1, F_2 \rightarrow F_1$  and  $K_2 \rightarrow K_1$  such that the diagram below commutes. (The arrows  $L \rightarrow L, I \rightarrow I, R \rightarrow R$  in the diagram are the identities.)

A transformation morphism is called *componentwise mono* if it is composed of monos only.

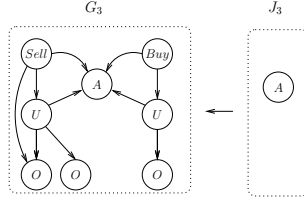
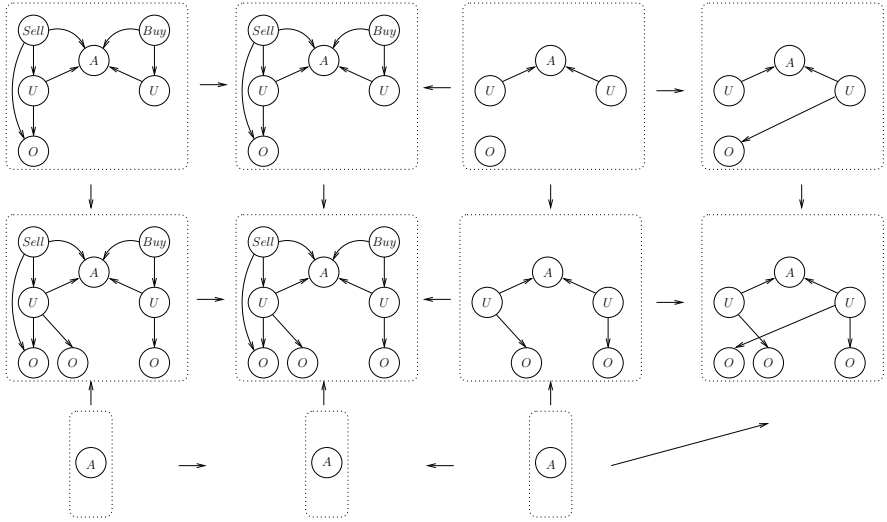


The intuition—at least if all arrows are mono—is that a morphism “embeds” transformation  $t_1$  into  $t_2$ . Thus,  $G_1$  (the object being rewritten) is mapped into  $G_2$  and the same holds for  $D_1$  (the partial match),  $G_1^+, C_1$  and  $H_1$ . Furthermore, since  $G_1$  is contained in  $G_2$ , it might be necessary to borrow more context from the environment. Hence  $F_1$  can be larger than  $F_2$  and the same holds for the inner and outer interfaces of  $F_1$  (denoted by  $J_1$  and  $K_1$ ). For instance  $J_1$  might have to be larger than  $J_2$  since more context has to be attached. Hence the “squares”  $J_2, J_1, G_1, G_2$  and  $F_2, F_1, G_1^+, G_2^+$  and  $K_2, K_1, C_1, C_2$  are not real squares, but will be called *horseshoes* in the following.

The complexity of our proofs stems from the fact that these horseshoes have to be taken into account. Otherwise it would be possible to simply work in a functor category.

**Definition 6 (Category of transformations).** The category having as objects transformations and as arrows transformation morphisms is denoted by **Trafo**. Composition of transformation morphisms is defined componentwise.

*Example.* Consider the graph with interface  $J_3 \rightarrow G_3$  in Fig. 4. The graph  $G_3$  includes a market place with two users. The first one possesses two objects and is trying to sell one of them. The second user is looking for an object to buy. A transformation for  $J_3 \rightarrow G_3$ , using rule (*Buy*), can be found in Fig. 5. Observe that in this case the given graph already includes all what is needed for applying rule (*Buy*) and thus nothing is actually borrowed from the context. Thus only the interface is exposed in the label, i.e., the graph  $F_3 = J_3$ . It is not difficult to see that there is an obvious transformation morphism  $\theta_1: t_1 \rightarrow t_3$ , where  $t_1$  is the transformation in Fig. 3.

Fig. 4. The graph with interface  $J_3 \rightarrow G_3$ .Fig. 5. A transformation with borrowed context  $t_3$  over  $J_3 \rightarrow G_3$ , using rule  $(Buy)$ 

Although the definition of transformation morphisms does not impose any condition on the vertical squares or horseshoes, we can infer some properties by taking into account that all horizontal squares are either pullbacks or pushouts (along monos, and thus also pullbacks).

**Lemma 7 (Properties of transformation morphisms).** *For a transformation morphism  $\theta$  as defined in Definition 5 it holds that:*

- The squares  $I, I, L, L$  and  $I, I, R, R$  and  $C_1, C_2, G_1^+, G_2^+$  and  $C_1, C_2, H_1, H_2$  are pushouts.
- If the arrows  $G_1^+ \rightarrow G_2^+$ ,  $C_1 \rightarrow C_2$  and  $H_1 \rightarrow H_2$  are mono, the squares  $L, L, G_1^+, G_2^+$  and  $I, I, C_1, C_2$  and  $R, R, H_1, H_2$  and  $K_2, K_1, F_2, F_1$  and  $D_1, D_2, G_1, G_2$  are pullbacks.



## 4 Composition of Transformations

In this section we study a composition mechanism for transformations. More precisely we show that given two transformations  $t_1, t_2$ , using the same production, with a common subtransformation  $t_0$ , the two transformations can be combined via a pushout. We will give sufficient conditions for the existence of this pushout and show how it can be constructed.

We first consider a simpler category where objects are pushouts and we show how to construct pushouts in this setting.

**Lemma 8 (Pushouts in the category of pushouts).** *Let  $\mathbf{C}$  be a fixed adhesive category. Consider the category of pushouts in  $\mathbf{C}$ , where objects are pushouts  $p_i$  of the form*

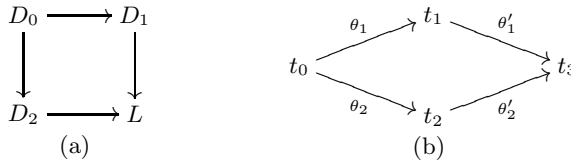
$$\begin{array}{ccc} A_0^i & \longrightarrow & A_2^i \\ \downarrow & & \downarrow \\ A_1^i & \longrightarrow & A_3^i \end{array}$$

and an arrow  $\varphi: p_1 \rightarrow p_2$  consists of four arrows  $(\varphi_0, \varphi_1, \varphi_2, \varphi_3)$  (with  $\varphi_i: A_i^1 \rightarrow A_i^2$ ) which connect the corners of the squares such that the full diagram (which is a cube) commutes.

Given two arrows  $\varphi^1: p_0 \rightarrow p_1, \varphi^2: p_0 \rightarrow p_2$  in this category, a pushout  $\psi^1: p_1 \rightarrow p_3, \psi^2: p_2 \rightarrow p_3$  can be computed by constructing four pushouts of the arrows  $\varphi_i, \psi_i$ , provided that these pushouts exist. Then the resulting (pushout) square is composed of the four mediating arrows.

Note that even if the pushouts  $p_0, p_1, p_2$  consist only of monos, the resulting pushout square  $p_3$  does not necessarily consist of monos. Hence in our case this property has to be shown by different means.

We next introduce a property ensuring the composability of transformations.



**Fig. 6.** Composition of transformations

**Definition 9 (Composable transformation morphisms).** *Let  $\theta_i: t_0 \rightarrow t_i$  with  $i \in \{1, 2\}$  be transformation morphisms. We say that  $\theta_1$  and  $\theta_2$  are composable if*

1.  $\theta_1, \theta_2$  are componentwise mono and
2. the square in the underlying category  $\mathbf{C}$  in Fig. 6(a) (where the top and right arrows appear in  $\theta_1$  and the left and bottom arrows appear in  $\theta_2$ ) is a pullback.

Intuitively the second condition in the definition above requires that the partial match for  $t_0$  is the intersection of the partial matches for  $t_1, t_2$ .

**Theorem 10 (Composition of transformations).** *Let  $\theta_i: t_0 \rightarrow t_i$  with  $i \in \{1, 2\}$  be two composable transformation morphisms. Then the pushout of  $\theta_1, \theta_2$  exists (see Fig. 6(b)) and can be obtained in the following way:*

- Construct  $D_3, G_3, G_3^+, C_3, H_3$  by taking pushouts and  $J_3, F_3, K_3$  by taking pullbacks. For instance  $D_3$  is constructed by taking the pushout of  $D_0 \rightarrow D_1, D_0 \rightarrow D_2$ , where these two arrows are taken from  $\theta_1$  respectively  $\theta_2$ . This produces the transformation morphisms  $\theta'_i: t_i \rightarrow t_3$ .
- In order to construct the arrows in  $t_3$  we proceed as follows:
  - Most arrows can be immediately obtained as mediating arrows. This is the case for  $D_3 \rightarrow G_3, D_3 \rightarrow L, G_3 \rightarrow G_3^+, L \rightarrow G_3^+, C_3 \rightarrow G_3^+, C_3 \rightarrow H_3, J_3 \rightarrow F_3, K_3 \rightarrow F_3, I \rightarrow C_3, R \rightarrow H_3$ .
  - Furthermore construct  $J_3 \rightarrow G_3$  by composing  $J_3 \rightarrow J_1 \rightarrow G_1 \rightarrow G_3$ . Similarly for  $F_3 \rightarrow G_3^+$  and  $K_3 \rightarrow C_3$ .

## 5 Decomposition of Transformations

In the previous section we have shown how to compose larger transformations out of smaller ones. Here we are going into the opposite direction and show under which conditions transformations can be split into smaller ones. That is, given a transformation of  $J \rightarrow G$  and a decomposition of  $G$  into subobjects  $G_1, G_2$ , is it possible to find transformations for these subobjects, such that the composition of these transformations yields the original transformation?

### 5.1 Projecting Transformations

In order to be able to formulate the decomposition of transformations, we will first show how to project a transformation to a subobject of  $G$ , i.e., to a subobject of the object to be rewritten. We identify some conditions which ensure that a transformation can be projected over a subobject of the rewritten object. Roughly, the interface of the subobject must be sufficiently large to guarantee that the needed context can be actually borrowed.

**Definition 11 (Extensibility).** *Let  $t_2$  be a transformation and let  $J_2 \rightarrow J_1 \rightarrow G_1 \rightarrow G_2$  be a factorisation of the arrow  $J_2 \rightarrow G_2$ . Then the transformation is called extensible with respect to this factorisation, whenever there exists a subobject  $F_1$  of  $U_2$  (the pushout of  $G_2^+ \leftarrow C_2 \rightarrow H_2$ ) such that*

$$G_1 \cup L = G_1 \cup F_1 \quad G_1 \cap F_1 = J_1.$$

The definition above basically requires (in lattice-theoretic terms) that the pushout complement  $F_1$  of  $J_1 \rightarrow G_1 \rightarrow G_1^+$  exists, where  $G_1^+ = G_1 \cup L$ . Note that in adhesive categories the pushout complement of monos is unique (if it exists).

The extensibility condition given in Definition 11 can be difficult to work with. Below we give an alternative handier condition, sufficient for extensibility.

**Lemma 12 (Sufficient condition for extensibility).** *Let  $t_3$  be a transformation and let  $J_3 \rightarrow J_1 \rightarrow G_1 \rightarrow G_3$  be a factorisation of the arrow  $J_3 \rightarrow G_3$ . Then  $t_3$  is extensible with respect to this factorisation if the pushout complement  $X_{13}$  of  $J_1 \rightarrow G_1 \rightarrow G_3$  exists, i.e., there exists an object  $X_{13}$  and morphisms such that the square below is a pushout.*

$$\begin{array}{ccc} J_1 & \longrightarrow & G_1 \\ \downarrow & & \downarrow \\ X_{13} & \longrightarrow & G_3 \end{array}$$

In this case set  $F_1 = (X_{13} \cup F_3) \cap (G_1 \cup L)$ .

Essentially, the sufficient condition requires that the interface of the smaller object  $G_1$  is sufficiently large to allow to get the larger object  $G_3$  by extending  $G_1$  along its interface.

Now let  $t_i$  be a transformation over an object with interface  $J_i \rightarrow G_i$  ( $i \in \{1, 2\}$ ) and let  $J_2 \rightarrow J_1 \rightarrow G_1 \rightarrow G_2$  be a factorisation of  $J_2 \rightarrow G_2$ . We say that a transformation morphism  $\theta : t_1 \rightarrow t_2$  is *consistent* with the factorisation if it has the arrows  $J_2 \rightarrow J_1$  and  $G_1 \rightarrow G_2$  as components.

**Proposition 13 (Projection of transformations).** *Let  $t_2$  be a transformation and let  $J_2 \rightarrow J_1 \rightarrow G_1 \rightarrow G_2$  be a (mono) factorisation of the morphism  $J_2 \rightarrow G_2$  such that  $t_2$  is extensible with respect to this factorisation. Then there exists a unique transformation  $t_1$  of  $J_1 \rightarrow G_1$ , with a componentwise mono transformation morphism  $\theta : t_1 \rightarrow t_2$ , consistent with the factorisation.*

The objects of this transformation can be constructed as follows:

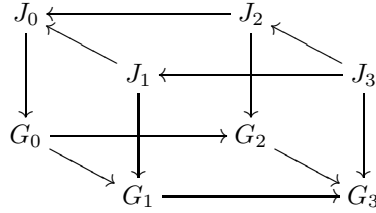
1. Construct  $U_2$  as the pushout of  $C_2 \rightarrow G_2^+$  and  $C_2 \rightarrow H_2$ . Now all objects can be considered as subobjects of  $U_2$ .
2. The object  $F_1$  is given by the extensibility property above, which requires that  $G_1 \cup L = G_1 \cup F_1$  and  $G_1 \cap F_1 = J_1$ . Set  $D_1 = G_1 \cap D_2$ ,  $G_1^+ = G_1 \cup L$ ,  $C_1 = G_1^+ \cap C_2$ ,  $H_1 = C_1 \cup R$ ,  $K_1 = F_1 \cap C_1$ .

## 5.2 Decomposing Transformations

As a first step towards the decomposition of a transformation, we introduce a suitable decomposition for an object with interface.

**Definition 14 (Proper decomposition).** *Let  $J_3 \rightarrow G_3$  be an object with interface. Then a proper decomposition of  $J_3 \rightarrow G_3$  is a cube as shown below*

where all arrows are mono, the square  $G_0, G_1, G_2, G_3$  is a pushout and and the square  $J_0, J_1, J_2, J_3$  is a pullback. (Note that the four remaining “squares” are horseshoes.)



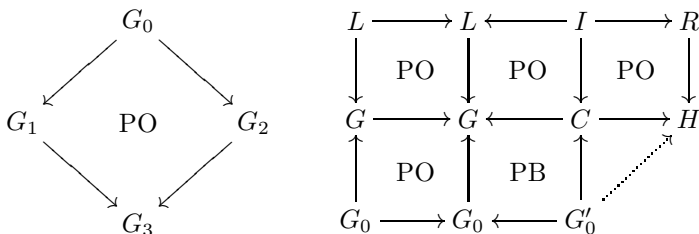
**Theorem 15 (Decomposition of transformations).** *Let  $t_3$  be a transformation of an object with interface  $J_3 \rightarrow G_3$ . Consider a proper decomposition of  $J_3 \rightarrow G_3$  as in Definition 14 and assume that the transformation  $t_3$  is extensible with respect to the factorisations  $J_3 \rightarrow J_1 \rightarrow G_1 \rightarrow G_3$  and  $J_3 \rightarrow J_2 \rightarrow G_2 \rightarrow G_3$ .*

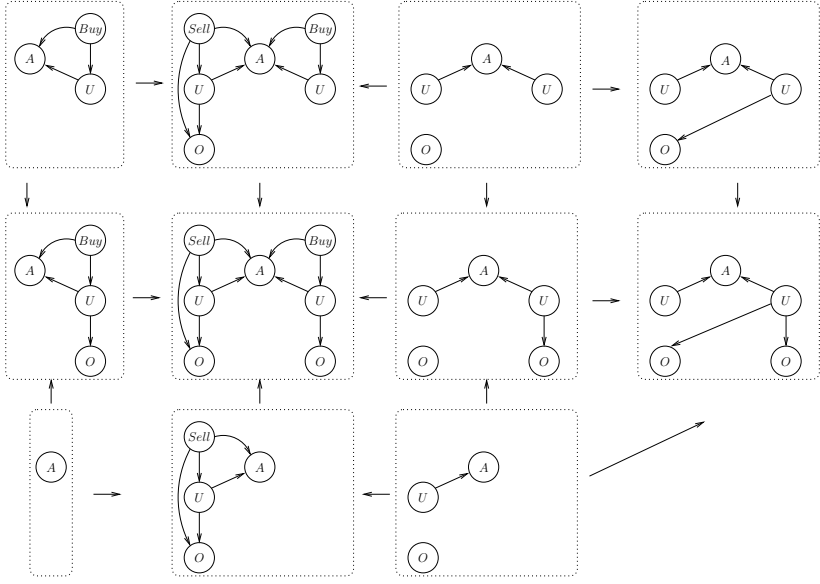
*Then there are transformations  $t_i$  for  $J_i \rightarrow G_i$  (where  $i \in \{0, 1, 2\}$ ) with componentwise mono transformation morphisms  $\theta_j: t_0 \rightarrow t_j$ ,  $\theta'_j: t_j \rightarrow t_3$  (where  $j \in \{1, 2\}$ ) forming a pushout in the category of transformations (see the diagram in Theorem 10). These transformation morphisms can be obtained via projections as described in Proposition 13.*

Observe that, if in the cube in Theorem 15 above we have the special (but very typical) case where  $J_0 = J_1 = J_2 = J_3 = G_0$  (and all arrows between these objects are the identities), the sufficient extensibility condition of Lemma 12 is satisfied: in the terminology of this lemma  $X_{13} = G_2$  and  $X_{23} = G_1$ .

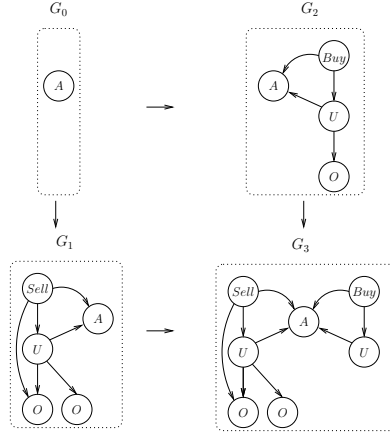
In a sense, composition and decomposition are inverse to each other up to isomorphism. The fact that composition is the inverse of decomposition has been shown directly in Theorem 15. On the other hand, since projections are unique (by Proposition 13), there is—up to isomorphism—only one way to decompose a transformation according to a proper decomposition of the rewritten object (see Definition 14). Hence, also decomposition is the inverse of composition.

Next we discuss the special case where a DPO rewriting step with trivial borrowed context is decomposed, leading to transformations with possibly non-empty borrowed contexts. Assume that  $G = G_3$  can be split into  $G_0, G_1, G_2$  as in the pushout diagram below on the left and consider a DPO rewriting step for  $G_3$ . Then this step can be extended to a transformation with borrowed context for  $G_3$  (with interface  $G_0$ ) with a total match of the left-hand side.





**Fig. 7.** A transformation with borrowed context  $t_2$  over  $J_2 \rightarrow G_2$ , using rule  $(Buy)$



**Fig. 8.** Decomposition of transformations

In this case we can set  $J_0 = J_1 = J_2 = J_3 = G_0$  and obtain a proper decomposition of  $J_3 \rightarrow G_3$  as in Definition 14 (the top square is trivially pullback and the bottom square is a pushout by assumption). Then, decomposing transformation  $t_3$  as described in Proposition 15 leads to three transformations  $t_0, t_1, t_2$ , with—in general—partial matches  $D_0, D_1, D_2$ .

*Example.* Consider the graph with interface  $J_3 \rightarrow G_3$  in Fig. 4. Note that there is an obvious factorisation  $J_3 \rightarrow J_1 \rightarrow G_1 \rightarrow G_3$  of  $J_3 \rightarrow G_3$ . Furthermore, the transformation  $t_3$  in Fig. 5, which uses rule *(Buy)*, is extensible along such factorisation. In fact, the sufficient condition given by Lemma 12 is satisfied.

Therefore we can project the transformation  $t_3$  in Fig. 5 along such factorisation thus obtaining the transformation  $t_1$  over  $J_1 \rightarrow G_1$  depicted in Fig. 3. As already noted, as an effect of projecting the transformation over a smaller graph, the borrowed context becomes non-trivial (larger than the interface): the rule can be applied assuming that the context provides a user which buys the object sold by the user in  $G_1$ .

More generally, consider the diagram in Fig. 8, where morphisms are the inclusions suggested by the shapes of the graphs. This is a pushout in **Graph**. Moreover, we can imagine all graphs  $G_i$  to have an interface given by  $J_i = G_0$ . Then the conditions of Proposition 15 are satisfied: we can project the transformation  $t_3$  in Fig. 5 to transformations over  $J_i \rightarrow G_i$  ( $i \in \{0, 1, 2\}$ ). The projection over  $J_1 \rightarrow G_1$  leads to the transformation  $t_1$  in Fig. 3, while the projection over  $J_2 \rightarrow G_2$  leads to the transformation  $t_2$  in Fig. 7. Both  $t_1$  and  $t_2$  project to the same derivation  $t_0$  over  $J_0 \rightarrow G_0$ . The pushout of the obtained transformations can be computed, according to Theorem 10 to obtain  $t_3$  again.

## 6 Conclusion and Comparison to Related Work

In this paper, focusing on a setting in which a system is built out of smaller components, we discussed how derivations with borrowed context over the global state can be decomposed into transformations over the local state of each single component using the same rule. Vice versa, we showed that, under suitable consistency conditions, local transformations can be composed to give rise to a transformation over the global system state.

We remark that the form of composition described in this paper is quite different from amalgamation as described for instance in [5]. There two transformations for different rules are amalgamated producing a transformation for the amalgamated rule. In our case, instead, the rule is fixed and the transformations differ with respect to the context that has to be borrowed from the environment. By composing objects and hence transformations we obtain additional structure which might reduce the borrowed context.

The composition and decomposition results can be seen as a basic step towards the possibility of defining transformations only for “atomic objects” and assemble all possible transformations out of these atomic transformations, and thus, towards an inductive definition, in SOS style, of the transition system of a graph transformation system (more generally an adhesive rewriting system). In addition to the composition result we will also need the possibility to compose an evolving object with a passive context and to have rules for handling restrictions of the interface. This would correspond to the communication, parallel composition and restriction rules for process calculi. Additionally, composition would be even more natural and closer to process calculi if performed over so-called rewriting

steps, hiding the internal details, rather than on full transformations. That is—in the terminology of Definition 4—we would like to observe only the object with interface  $J \rightarrow G$ , the resulting object  $K \rightarrow H$  and the label or borrowed context  $J \rightarrow F \leftarrow K$ , but not the objects  $D$ ,  $G^+$ ,  $C$  which are only auxiliary or intermediate objects. We plan to extend our approach to this setting.

## References

1. P. Baldan, H. Ehrig, and B. König. Composition and decomposition of DPO transformations with borrowed contexts. Technical report, Universität Duisburg-Essen, 2006.
2. H. P. Barendregt. *The Lambda Calculus—its Syntax and Semantics*, volume 103 of *Studies in Logic and Foundations of Mathematics*. North-Holland, 1984.
3. F. Bonchi, F. Gadducci, and B. König. Process bisimulation via a graphical encoding. In *Proc. of ICGT '06*. Springer, 2006. LNCS, to appear.
4. L. Cardelli and A. D. Gordon. Mobile ambients. In *Proc. of FoSSaCS '98*, pages 140–155. Springer-Verlag, 1998. LNCS 1378.
5. 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.
6. H. Ehrig and B. König. Deriving bisimulation congruences in the DPO approach to graph rewriting. In *Proc. of FOSSACS '04*, pages 151–166. Springer, 2004. LNCS 2987.
7. H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.3: Concurrency, Parallelism, and Distribution*. World Scientific, 1999.
8. O. H. Jensen and R. Milner. Bigraphs and transitions. In *Proc. of POPL 2003*, pages 38–49. ACM, 2003.
9. S. Lack and P. Sobociński. Adhesive and quasiadhesive categories. *RAIRO – Theoretical Informatics and Applications*, 39(3), 2005.
10. J. J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In *Proc. of CONCUR 2000*, 2000. LNCS 1877.
11. R. Milner. The polyadic  $\pi$ -calculus: a tutorial. In *Logic and Algebra of Specification*. Springer-Verlag, Heidelberg, 1993.
12. V. Sassone and P. Sobociński. Reactive systems over cospans. In *Proc. of LICS '05*, pages 311–320. IEEE, 2005.