Compositionality in Graph Transformation

Arend Rensink

Department of Computer Science, Universiteit Twente
rensink@cs.utwente.nl

Abstract. Graph transformation works under a whole-world assumption. In modelling realistic systems, this typically makes for large graphs and sometimes also large, hard to understand rules. From process algebra, on the other hand, we know the principle of reactivity, meaning that the system being modelled is embedded in an environment with which it continually interacts. This has the advantage of allowing modular system specifications and correspondingly smaller descriptions of individual components. Reactivity can alternatively be understood as enabling *compositionality*: the specification of components and subsystems are composed to obtain the complete model.

In this work we show a way to ingest graph transformation with compositionality, reaping the same benefits from modularity as enjoyed by process algebra. In particular, using the existing concept of graph interface, we show under what circumstances rules can be decomposed into smaller subrules, each working on a subgraph of the complete, whole-world graph, in such a way that the effect of the original rule is precisely captured by the synchronisation of subrules.

1 Introduction

Graph transformation has shown to be a very useful specification formalism, enabling the rapid modelling of systems of all kinds, ranging from physical systems to protocols and concrete software. However, one drawback of graph transformation systems is that they require a "whole-world" view of the system to be modelled: the model at hand always describes the entire system, and thus can grow very large. There is no support for component submodels which can be analysed individually and composed later.

This is a consequence of the fact that graph transformation, like all rewriting techniques, has a *reductive* semantics: applying a graph transformation rule involves finding a match in the current model (the host graph) and making local changes in the model without any reference to an external environment (a step which in some contexts is called a reduction). This is in contrast to the *reactive* semantics enjoyed by, for instance, process algebra: there the application of a rule typically involves communication with an environment that is unknown at the moment (often called a reaction); the overall system behaviour results from synchronisation of different models, in which one plays the role of environment for the other.

In this paper we study a notion of reactivity for graph transformation. We compose individual models by "gluing them together" along a predefined interface; the result is like union, where the interface identifies the parts that are are merged. (Technically, this comes down to constructing pushouts in an appropriate category.) Transformation rules

S. Abramsky et al. (Eds.): ICALP 2010, Part II, LNCS 6199, pp. 309–320, 2010.
© Springer-Verlag Berlin Heidelberg 2010

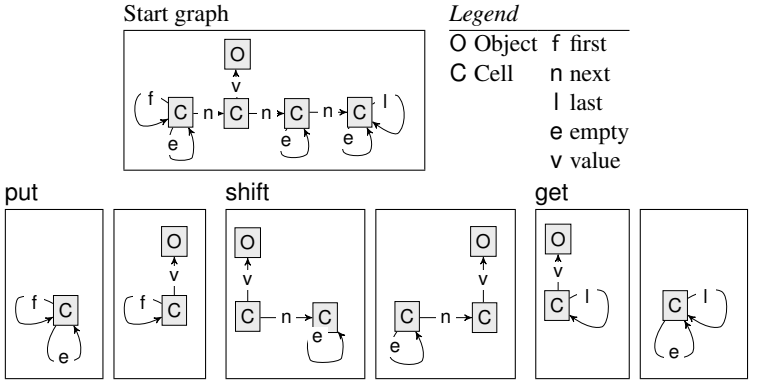


Fig. 1. Start graph and transformation rules for a simple, 4-cell buffer

are glued together in a similar fashion. Let us use the symbol '$+$' to represent composition (which is partial since only objects with compatible interfaces can be composed). We then compare transitions of individual components, $G_i \xrightarrow{p_i} H_i$ for $i = 1, 2$, where the G_i and H_i are graphs and the p_i transformation rules, to transitions $G_1 + G_2 \xrightarrow{p} H$ of the composed system. We then investigate the following crucial properties (or actually a slightly more involved version that also takes matches into account):

Soundness. Compatible local transitions always give rise to global transitions: when the G_i and p_i are compatible, then $G_i \xrightarrow{p_i} H_i$ for $i = 1, 2$ implies that the H_i are compatible and that $G_1 + G_2 \xrightarrow{p_1+p_2} H_1 + H_2$.

Completeness. All global transitions can be obtained by composing compatible local transitions: $G_1 + G_2 \xrightarrow{p} H$ implies that there are compatible p_i for $i = 1, 2$ such that $G_i \xrightarrow{p_i} H_i$, $p = p_1 + p_2$ and $H = H_1 + H_2$.

We illustrate the results on a simple example of a buffer. The whole-world model with corresponding rules (consisting of a left hand side and a right hand side graph) is given in Fig. 1. The intuitive meaning of the rules is that a match is found for the left hand side, which is then replaced by the right hand side. A modular specification will allow us to specify the behaviour per cell (with sufficient context information to glue the cells together). For instance, Fig. 2 shows how the original graph can be decomposed into four individual cells, and also shows the decomposition of the rules.

The structure of the paper is as follows: Sect. 2 defines concrete graphs and their composition; subsequently, Sect. 3 generalises this to the more abstract setting of adhesive categories. We prove soundness and state sufficient conditions for completeness. Sect. 4 concludes the paper with an overview of related work and open questions.

A somewhat extended version of the paper, including all the proofs, can be found in the report [15].

2 Graph Composition

In this section we introduce a concrete definition of graphs with interfaces, called *marked graphs*, as well as the rules to transform them.

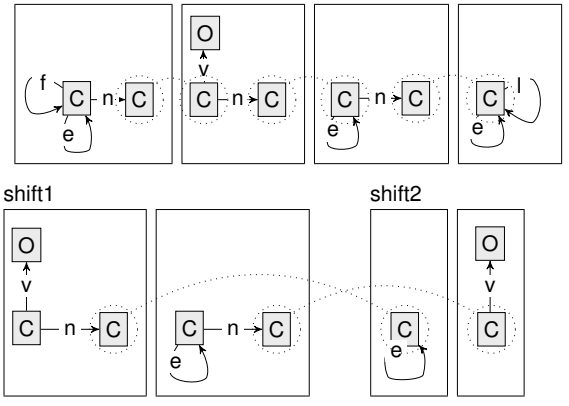


Fig. 2. Decomposed start graph and shift-rule. Dotted lines indicate sharing

2.1 Graphs and Their Transformation

Throughout this paper we assume global (countable) disjoint universes N of nodes, E of edges, and L of labels, with (also globally defined) functions $src, tgt\colon \mathsf{E} \to \mathsf{N}$ and $lab\colon \mathsf{E} \to \mathsf{L}$. In particular, for every combination of $v, w \in \mathsf{N}$ and $a \in \mathsf{L}$, there are assumed to be countably many $e \in \mathsf{E}$ such that $src(e) = v$, $tgt(e) = w$ and $lab(e) = a$.

Furthermore, we will use *structure-preserving functions* over $\mathsf{N} \cup \mathsf{E}$, which are functions $f = f_V \cup f_E$ with $f_V\colon V \to \mathsf{N}$ for some $V \subseteq \mathsf{N}$ and $f_E\colon E \to \mathsf{E}$ for some $E \subseteq \mathsf{E}$ such that $src \circ f_E = f_V \circ src \restriction E$, $tgt \circ f_E = f_V \circ tgt \restriction E$ and $lab \circ f_E = lab \restriction E$. Note that this implies $src(E) \cup tgt(E) \subseteq V$.

Definition 1 (graph)

- *A graph is a finite set $G \subseteq \mathsf{N} \cup \mathsf{E}$, such that $src(G \cap \mathsf{E}) \cup tgt(G \cap \mathsf{E}) \subseteq G$. We often write V_G for $G \cap \mathsf{N}$ and E_G for $G \cap \mathsf{E}$, or just V and E if the index G is clear from the context.*
- *Given graphs G, H, a* graph morphism *$f\colon G \to H$ is a structure-preserving function. If f is bijective we also call it an* isomorphism *and G and H* isomorphic.

We often use the pointwise extension of morphisms to sets of elements. Due to the use of globally defined sets of nodes and edges, graphs are closed under union and intersection, but not under set difference: $G \setminus H$ may contain "dangling edges". Moreover, for a morphism $f\colon G \to H$ and a subgraph $H' \subseteq H$, $f^{-1}(H')$ is also a graph.

Definition 2 (rule). *A graph transformation rule is a tuple $p = \langle L, R \rangle$, consisting of a left hand side (LHS) graph L and a right hand side (RHS) graph R. The intersection $I = L \cap R$ is often called the* interface *of p.*

Let $p = \langle L, R \rangle$ be a transformation rule. p is *applicable* to a graph G (often called the *host graph*) if there exists a *match* $m\colon L \to G$, which is a graph morphism satisfying

No dangling edges: For all $e \in E_G$, $src(e) \in m(L \setminus R)$ or $tgt(e) \in m(L \setminus R)$ implies $e \in m(L \setminus R)$;
No delete conflicts: $m(L \setminus I) \cap m(I) = \emptyset$.

The intuition is that the elements of G that are in $m(L)$ but not in $m(I)$ are scheduled to be deleted by the production. If a node is deleted, then so must its incident edges, or the result would not be a graph. Note that, due to the absence of delete conflicts, $m(L \setminus R) = m(L \setminus I) = m(L) \setminus m(I)$. Given such a match m, the *application* of p to G is defined by extending m to a morphism $m' \colon L \cup R \to H'$, where $H' \supseteq G$ and all elements of $R \setminus L$ have distinct, fresh images under m', and defining

$$H = (G \setminus m(L)) \cup m'(R) \ .$$

H is called the *target* of the production; we write $G \xrightarrow{p,m} H$ to denote that m is a valid match on host graph G, giving rise to target graph H, and $G \xrightarrow{p} H$ to denote that there is a match m such that $G \xrightarrow{p,m} H$. Note that H is not uniquely defined for a given p and m, due to the freedom in choosing the fresh images for $R \setminus L$; however, it is well-defined modulo isomorphism. From now on we assume that the fresh images are chosen in some deterministic fashion, depending on the element of $R \setminus L$ that triggers their introduction as well as the elements that have been generated "so far". (This is a kind of requirement that is quite hard to formalise but easy to realise in any implementation.)

2.2 Marked Graphs and Their Composition

We now define the notion of graphs and rules with interfaces, discussed above.

Definition 3 (marked graphs and rules)

- *A* marked graph G *is a pair of graphs* $(\underline{G}, \overline{G})$ *with* $\underline{G} \subseteq \overline{G}$. $\underline{G}$ *is called the* inner graph *or* subgraph*, and* $\overline{G}$ *the* outer graph*. Two marked graphs* G, H *are called* compatible *if* $\underline{G} = \underline{H} = \overline{G} \cap \overline{H}$*; in that case, we use* $G + H$ *to denote the graph* $\overline{G} \cup \overline{H}$.
- *A* marked morphism *between two marked graphs* G, H *is a pair of morphisms* $m = (\underline{m} \colon \underline{G} \to \underline{H}, \overline{m} \colon \overline{G} \to \overline{H})$ *such that* $\underline{m} = \overline{m} \restriction \underline{G}$*. We write* $m \colon G \to H$.
- *A* marked rule p *is a pair of marked graphs* (L, R)*, such that* $\underline{L} \cap \overline{R} = \overline{L} \cap \underline{R}$*. This can alternatively be interpreted as a pair of rules* $(\underline{p}, \overline{p})$ *with* $\underline{p} = (\underline{L}, \underline{R})$ *and* $\overline{p} = (\overline{L}, \overline{R})$*. As before,* $\underline{p}$ *is called the* inner rule *or* subrule *and* $\overline{p}$ *the* outer rule*. Two marked rules* p, q *are compatible if* L_p *and* L_q *as well as* R_p *and* R_q *are compatible; the composition is denoted* $p + q$ *(*$= (L_p + L_q, R_p + R_q)$*).*

Obviously, the intention is that a marked rule should act upon a marked graph by applying the outer rule to the outer graph and the inner rule to the inner graph. The outcome should be a new marked graph. For this purpose, we define a (marked) match of a marked rule p into a marked graph G as a marked morphism $m \colon L \to G$ such that $\overline{m}$ is a match of $\overline{p}$ into $\overline{G}$ (meaning that there are no dangling edges or delete conflicts) and in addition the following condition is satisfied:

Delete consistency: $\overline{m}(\overline{L}) \cap \underline{G} \subseteq \overline{m}(\underline{L} \cup \overline{I})$

Delete consistency essentially states that elements deleted from the outer graph (by the outer rule) should either not occur in the inner graph in the first place, or be explicitly deleted from the inner graph (by the inner rule). The following states that delete consistency is a necessary and sufficient conditions for marked rule application to yield a new marked graph.

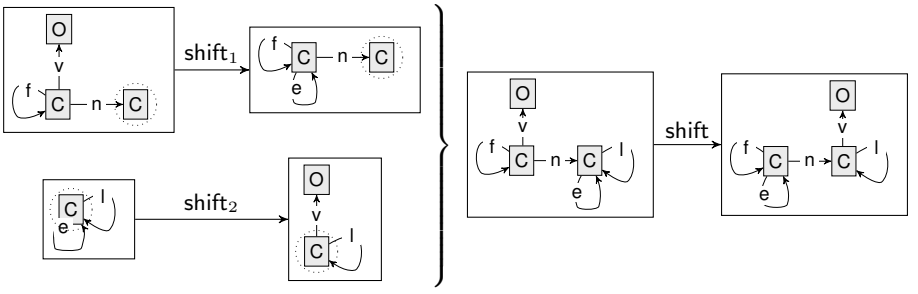


Fig. 3. Composition of two transformations

Proposition 4. *Let p be a marked rule, G a marked graph, and $m\colon L \to G$ a marked morphism. m is a marked match if and only if the pair $H = (\underline{H}, \overline{H})$ defined by $\underline{G} \xrightarrow{\underline{p},\underline{m}} \underline{H}$ and $\overline{G} \xrightarrow{\overline{p},\overline{m}} \overline{H}$ is a marked graph. We write $G \xrightarrow{p,m} H$.*

2.3 Soundness and Completeness

We can now formulate the notion of composition of graph transformations that is central to this paper. Composition is defined only for compatible marked transformations, where two marked transformations $G_i \xrightarrow{p_i,m_i} H_i$ for $i = 1, 2$ are called compatible if (G_1, G_2), (p_1, p_2) and (m_1, m_2) are compatible pairs. The assumption of determinism then implies that H_1 and H_2 are also compatible.

As discussed in the introduction, the crucial properties that we are after are soundness and completeness of subgraph transformations with respect to transformation of the complete graph. Soundness holds for all marked matches: the following theorem is a consequence of the more general Th. 14 proved in the next section.

Theorem 5 (soundness for graphs). *Any compatible pair of marked transformations is composable; i.e., if $G_i \xrightarrow{p_i,m_i} H_i$ for $1 = 1, 2$ are compatible marked transformations, then $G_1 + G_2 \xrightarrow{p_1+p_2,m_1+m_2} H_1 + H_2$.*

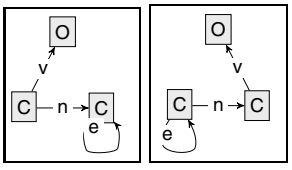
For instance, Fig. 3 shows how transformations on two marked fragments of a buffer, using the shift_i-rules of Fig. 2, give rise to a global transformation using the composed rule shift of Fig. 1. As before, the common subgraphs are indicated by dotted lines.

Completeness, on the other hand, only holds under an additional constraint. This is due to the fact that rules cannot be decomposed arbitrarily. We give a sufficient condition for decomposability, based on the concept of *accommodation*.

Definition 6 (accommodation of graphs). *Let p be a rule, and $m\colon L \to G$ a match of p in G. A subgraph $G' \subseteq G$* accommodates *a subgraph $R' \subseteq R$ under m if $L \cap R' = R \cap m^{-1}(G')$.*

Intuitively, G' accommodates R' if for all edges created according to R' (with respect to L), the end nodes are either themselves created by p or matched in G'. This ensures that a subrule p' of p exists with RHS R' and LHS $m^{-1}(G')$ (being the largest subgraph of L

that matches into G'). For instance, in a match of the rule shift of Fig. 1 in a graph G, the subgraph $G' \subseteq G$ consisting only of the image of the right hand C-node accommodates the subgraph of R_{shift} consisting of the v-edge and its end nodes, since the other end node (the O-node) is itself created. On the other hand, consider the following variation:



(1)

Here the O-node is not created by the rule: hence, in order to accommodate the RHS' v-edge, a subgraph must include an image of the O-node.

The completeness result for the concrete graphs studied in this section is given by the following theorem, the proof of which follows from the more general Th. 17 below.

Theorem 7 (completeness for graphs). *Let G_i be compatible marked graphs for $i = 1, 2$. A transformation $G_1 + G_2 \xrightarrow{p,m} H$ can be decomposed into marked transformations $G_i \xrightarrow{p_i,m_i} H_i$ if there exist graphs $\overline{R}_i$ for $i = 1, 2$ such that (i) $R = \overline{R}_1 \cup \overline{R}_2$ and (ii) $\overline{G}_i$ accommodates $\overline{R}_i$ under m.*

For our running example, the rules in Fig. 1 are such that all created edges have precisely one pre-existing end node; for that reason, the conditions for decomposability are always satisfied, meaning that we can soundly and completely decompose all transformations under arbitrary decompositions of the graphs. Indeed, the following (fixed) rule decomposition turns out to be sufficient:

- put is decomposed into put_1, which is identical to put, and an empty rule put_2;
- shift is decomposed into shift_1 and shift_2;
- get is decomposed into get_1, which is empty, and get_2, which is identical to get.

An example is shown in Fig. 4. To the left is the transition system T resulting from the repeated application of the composed rules to an initial graph consisting of two empty cells. To the right are the transition systems T_1 and T_2 generated from subgraphs consisting of one cell each, and the subrules discussed above. The matches are not included. The correctness of the (de)composition can be seen from the fact that T is the product of T_1 and T_2 in the automata sense, where x_i-transitions ($x = \mathsf{put}, \mathsf{shift}, \mathsf{get}$ and $i = 1, 2$) are synchronised and relabelled to x.

3 Composition for Marked Objects

We now lift the framework for concrete graphs presented above to a more abstract, categorical level, and we prove the soundness and completeness results on this level.

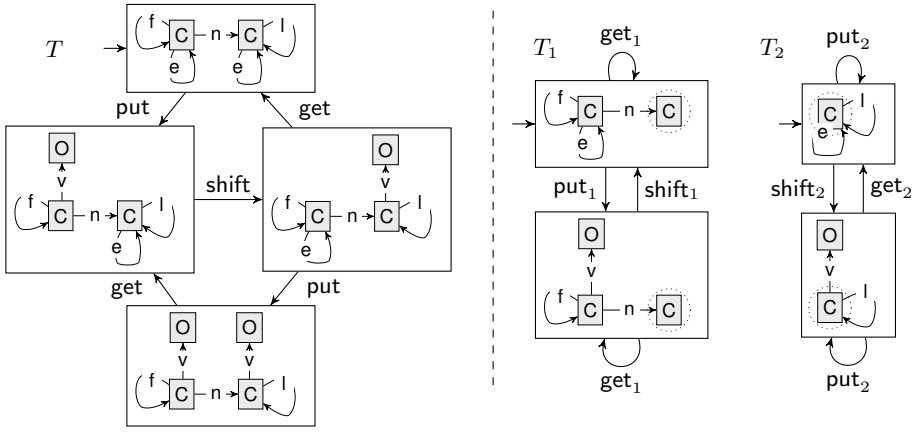


Fig. 4. Transformations of a complete and decomposed 2-cell buffer

3.1 Transformation in Adhesive Categories

Unfortunately, there is no space to recall the concepts from category theory used in this section (but see the report version [15]). We now recall the definitions of adhesive and quasiadhesive categories from [11,10]:

Definition 8 ((quasi-)adhesive category). *A category C is* quasiadhesive *if it satisfies the following properties:*

1. *C has pushouts along regular monomorphisms;*
2. *C has pullbacks;*
3. *Pushouts along regular monomorphisms are Van Kampen squares.*

A quasiadhesive category is called adhesive *if all monos are regular.*

For those that are not familiar with this theory, the following intuitions may be helpful:

- A regular mono $f\colon A \to B$ identifies a subobject of B that is isomorphic to A;
- The pushout of $B \xleftarrow{f} A \xrightarrow{g} C$ may be thought of as the union of B and C, where the shared subset is given by A and its "embedding" in B and C;
- The pullback of $B \xrightarrow{h} D \xleftarrow{k} C$ may be thought of as the intersection of B and C, where their "embedding" in D determines which elements they have in common.

It has been shown in [11,10] that (quasi-)adhesive categories form a nice, general framework in which properties of graph transformation systems can be proved abstractly; they generalise in some part the High-Level Replacement systems studied in, e.g., [4]. The graphs used in the previous section fall into this framework.

Proposition 9. *Graphs with graph morphisms form an adhesive category* Graph.

The notion of transformation rule used in the previous section generalises to categories in the standard way: the pair (L, R) turns into a span $L \hookleftarrow I \hookrightarrow R$, where I acts as the interface between L and R — which in the setting of the previous section corresponds to the intersection $L \cap R$. We also introduce morphisms over rules.

Definition 10 (rule). *Let C be a quasiadhesive category.*

- *A rule p is a span of regular monos $L \xleftarrow{l} I \xrightarrow{r} R$.*
- *A rule p is applicable to a given graph G if there exists a morphism $m\colon L \to G$ such that $I \hookrightarrow L \xrightarrow{m} G$ has a pushout complement. In that case, the application of p to G is defined by the diagram*

$$\begin{array}{ccccc} L & \xleftarrow{\ l\ } & I & \xrightarrow{\ r\ } & R \\ {\scriptstyle m}\downarrow & \mathit{PO} & {\scriptstyle k}\downarrow & \mathit{PO} & \downarrow{\scriptstyle m'} \\ G & \xleftarrow[\ g\ ]{} & K & \xrightarrow[\ h\ ]{} & H \end{array}$$

- *A rule morphism $f\colon p \to q$ is a triple of regular monomorphisms $\langle f_L, f_I, f_R\rangle$ such that the following diagram commutes and both squares are pullbacks:*

$$\begin{array}{ccccc} L_p & \xleftarrow{\ l_p\ } & I_p & \xrightarrow{\ r_p\ } & R_p \\ {\scriptstyle f_L}\uparrow & \mathit{PB} & {\scriptstyle f_I}\uparrow & \mathit{PB} & \uparrow{\scriptstyle f_R} \\ L_q & \xleftarrow[\ l_q\ ]{} & I_q & \xrightarrow[\ r_q\ ]{} & R_q \end{array}$$

The purpose of insisting that the squares of a rule morphism are pullbacks is to ensure that I_p is essentially the intersection of L_p and I_q, or alternatively, of R_p and I_q. This implies that elements preserved by the target rule (q) will also be preserved by the source rule (p). For an arbitrary (quasiadhesive) category C, the rules with rule morphisms form a category $\mathsf{Rule}(C)$.

3.2 Marked Objects

We now define the general notion of a marked object. As in the concrete case of graphs, a marked object is a monomorphism from an inner object to an outer object. The inner object is an interface used to glue marked objects together: gluing two marked objects with the same interface comes down to taking the pushout of the corresponding span.

Definition 11 (marked object). *Let C be an arbitrary category.*

- *A* marked object *X is a monomorphism $e_X\colon \underline{X} \hookrightarrow \overline{X}$. $\underline{X}$ is called the inner object and $\overline{X}$ the outer object.*
- *Given two marked objects X, Y, a marked morphism $f\colon X \to Y$ is a pair of morphisms $\underline{f}\colon \underline{X} \to \underline{Y}$ and $\overline{f}\colon \overline{X} \to \overline{Y}$ such that the resulting square commutes:*

$$\begin{array}{ccc} \underline{X} & \xrightarrow{\ \underline{f}\ } & \underline{Y} \\ {\scriptstyle e_X}\downarrow & & \downarrow{\scriptstyle e_Y} \\ \overline{X} & \xrightarrow{\ \overline{f}\ } & \overline{Y} \end{array}$$

- *Two marked objects X, Y are* compatible *if $\underline{X} = \underline{Y}$. If this is the case, we will use $X + Y$ to refer to the pushout object in the diagram*

$$\begin{array}{ccc} \underline{Y} = \underline{X} & \xrightarrow{\ e_X\ } & \overline{X} \\ {\scriptstyle e_Y}\downarrow & \mathit{PO} & \downarrow \\ \overline{Y} & \longrightarrow & X + Y \end{array}$$

For a category C, we use C^{M} to denote the category consisting of marked C-objects and marked C-morphisms. This construction is a special case of the *Artin gluing* studied in [10]. From that paper we may infer

Corollary 12 ([10, Th. 24]). *If C is an adhesive category, then C^{M} is a quasiadhesive category in which the regular monos are pullbacks of monos in C.*

This means that transformation of marked objects is well-defined, and that rules in C^{M} correspond to arrows in $\mathsf{Rule}(C)$ (cf. Def. 11). In the remainder of this section, unless stated otherwise we implicitly restrict ourselves to adhesive categories C. The following proposition states how the transformation of marked objects works: it essentially consists of the individual transformation of the inner and outer objects.

Proposition 13. *For any marked rule p, marked object G and marking consistent morphism $m\colon L \to G$, $G \xRightarrow{p,m} H$ if and only if $\underline{G} \xrightarrow{\underline{p},\underline{m}} \underline{H}$ and $\overline{G} \xrightarrow{\overline{p},\overline{m}} \overline{H}$.*

In the full paper [15], we also take a look at the existence of pushout complements for marked objects. We give a sufficient criterion for their existence, which in the case of Graph is implied by the delete consistency defined in Sect. 2.2.

3.3 Soundness and Completeness

We now arrive at the main results of this paper, which establish the relation between marked transformations of marked subgraphs and global transformations of the composed graph. The first main result of this paper states that we can compose arbitrary (compatible) marked transformations.

Theorem 14 (soundness). *For $i = 1, 2$, let G_i be compatible marked objects and p_i compatible marked rules. If $G_i \xRightarrow{p_i,m_i} H_i$ for $i = 1, 2$ such that $\underline{m}_1 = \underline{m}_2$, then $G_1 + G_2 \xRightarrow{p_1+p_2,m_1+m_2} H_1 + H_2$.*

The second main result states that, under some circumstances, we can also decompose global transformations into transformations of subgraphs. The circumstances involve that the decomposition into subgraphs allows an analogous decomposition of the rule's RHS. To formulate this precisely, we lift the notion of accommodation, defined in Def. 6, to the categorical level.

Definition 15 (accommodation). *Let p be a rule, G an object and $m\colon L \to G$ a morphism. A subobject $G' \hookrightarrow G$* accommodates *a subobject $R' \hookrightarrow R$ if they give rise to the following diagram:*

$$\begin{array}{ccccc} G & \xleftarrow{m} & L & \hookleftarrow & I & \hookrightarrow & R \\ \uparrow & & \mathit{PB} & & \uparrow & \mathit{PB} & \uparrow \\ G' & \longleftarrow & & & I' & \hookrightarrow & R' \end{array}$$

This notion of accommodation generalises the one of Def. 6. The intuition is the same: the intersection of R' and I (given by the right hand pullback), which determines where to connect the elements to be created according to R', should coincide with the part of I that matches into G'.

Proposition 16. *Accommodation in* Graph *coincides with the property in Def. 6.*

We can now formulate the generalised completeness theorem:

Theorem 17 (completeness). *Let p be a rule and for $i = 1, 2$ let G_i be compatible marked objects. If $G_1 + G_2 \xrightarrow{p,m} H$, and $R = R_1 + R_2$ for marked graphs R_i ($i = 1, 2$) such that $\overline{G}_i$ accommodates $\overline{R}_i$ under m, then there are marked transformations $G_i \xrightarrow{p_i,m_i} H_i$ for $i = 1, 2$ such that $p = p_1 + p_2$, $m = m_1 + m_2$ and $H = H_1 + H_2$.*

The accommodation criterion in Def. 15 is sufficient but not necessary for a transformation to be decomposable. This can be seen by observing that the construction in the proof of Th. 17 always gives rise to marked matches m_i that are actually pullback squares in C; but composition also works for "ordinary" matches. In particular, it is quite possible to have component rules p_i of which the outer LHS $\overline{L}_i$ is not the pullback of $\overline{G}_i$ and L, but some subgraph of this pullback. The search for a more precise criterion is future work (see below).

4 Conclusion

We have defined a notion of composition for graph transformations that acts on graphs as well as rules. We have proved that composition is sound, and have given sufficient conditions under which it is complete. This is a first, and essential, ingredient for enabling graph transformation to move from a purely reductive, whole-world specification formalism to a reactive, compositional formalism.

4.1 Related Work

Though we believe our aims and approach to be completely original, there is a number of loosely related topics, which we review here.

Synchronised Hyperedge Replacement. This is a paradigm in which graph transformation rules (more specifically, hyperedge replacement rules) can be synchronised based one the adjacency of their occurrences within a graph; see [9,7]. The synchronised rules are not themselves understood as graph transformation rules, an consequently the work does not address the type of compositionality issues that we have studied here. Still, it is interesting to see whether SHR synchronisation can be understood as a special type of composition in our sense.

History-Dependent Automata. This is a behavioural model in which states are enriched with a set of *names* (see [13] for an overview). Transitions expose names to the environment, and can also record the deletion, creation and permutation of names. HD-automata can be composed while synchronising their transitions: this provides a model for name passing. Transition systems induced by graph transformation rules (such as the ones depicted in Fig. 4) can be understood as a variant of HD-automata where the states are enriched with graphs rather than just sets, and the information on the transitions is extended accordingly. We intend to investigate this connection in the future.

Rule amalgamation. Studied first in [3] and later, much more extensively, in [16], the principle of rule amalgamation provides a general mechanism for rule (de)composition.

This is a sub-problem of the one we have addressed here (we study composition of the graphs as well as the rules), and indeed for that sub-problem our approach is entirely based on amalgamation.

Borrowed contexts. Like our paper, the work on borrowed contexts [5,1] uses a setting where only part of a graph is available, and studies the application of rules to such subgraphs in a way that is compatible with the original, reductive semantics. In contrast to our approach, however, they do not decompose rules: instead, when a rule is applied to a graph in which some of the required structure ("context") for the match is missing, this is imported ("borrowed") as part of the transformation. As a result, in this paradigm the subgraphs grow while being transformed, incorporating ever more context information. This is quite different from the basic intuitions behind our approach.

Summarising, where only rules are (de)composed in rule amalgamation, and only graphs in borrowed contexts, in our approach both rules and graphs are subject to (de)composition.

Compositional model transformation. The recent [2] studies a notion of compositionality in model transformation. Though on the face of it this sounds similar, in fact they study a different question altogether, namely whether a transformation affects the *semantics* of a model (given as a separate mapping to a semantic domain) in a predictable (compositional) manner. This is in sharp contrast with our work, which rather addresses the compositionality of the graph transformation framework itself.

4.2 Open Issues

Adhesive HLR categories. Adhesive categories as a foundation for graph transformation have been generalised to classes of categories with weaker assumptions; for instance, adhesive HLR categories in [6] and weak adhesive HLR categories in [14]. A natural question is whether our results also carry over to this generalisation.

Negative application conditions. For the use of graph transformation in practice, negative application conditions (NACs) as introduced in [8] have shown to be extremely useful. We plan to investigate the extension of our results to a setting with NACs.

Improved criteria for completeness. Th. 17 only gives a sufficient criterion for decomposing a transformation. It is not clear if this is weak enough to be usable in practice. From a theoretical point of view, in any case it would be interesting to have a more precise, necessary and sufficient criterion.

More general types of synchronisation. Our notion of composition requires marked rules to be compatible, meaning that they have a common subrule. This implies that joint nodes can only be created and deleted from subgraphs simultaneously. In other words, it is not possible to *hand over* a node from one subgraph to another. This is illustrated by the non-decomposable rule in (1) (page 314). Since such hand-over is definitely a desirable feature (used, e.g., in mobile calculi [12] to communicate channel names), we intend to study a generalisation of our framework that does not synchronise on subrules.

The larger picture. We have studied decomposition for individual rules. We want to extend this to rule systems; ideally, it should be possible to give a fixed decomposition of a

rule system which is guaranteed to remain correct for arbitrary sequences of transitions. At that point, also the connection with SOS semantics for process algebra (which was our starting point, see Sect. 1) should be strengthened.

References

1. Baldan, P., Ehrig, H., König, B.: Composition and decomposition of DPO transformations with borrowed context. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G. (eds.) ICGT 2006. LNCS, vol. 4178, pp. 153–167. Springer, Heidelberg (2006)
2. Bisztray, D., Heckel, R., Ehrig, H.: Compositionality of model transformations. In: Aldini, A., ter Beek, M., Gadducci, F. (eds.) 3rd International Workshop on Views On Designing Complex Architectures (VODCA). ENTCS, vol. 236, pp. 5–19 (2009)
3. Boehm, P., Fonio, H.R., Habel, A.: Amalgamation of graph transformations: A synchronization mechanism. J. Comput. Syst. Sci. 34(2/3), 377–408 (1987)
4. Ehrig, H., Habel, A., Kreowski, H.J., Parisi-Presicce, F.: From graph grammars to high level replacement systems. In: Ehrig, H., Kreowski, H.-J., Rozenberg, G. (eds.) Graph Grammars 1990. LNCS, vol. 532, pp. 269–291. Springer, Heidelberg (1991)
5. Ehrig, H., König, B.: Deriving bisimulation congruences in the DPO approach to graph rewriting with borrowed contexts. Mathematical Structures in Computer Science 16(6), 1133–1163 (2006)
6. Ehrig, H., Padberg, J., Prange, U., Habel, A.: Adhesive high-level replacement systems: A new categorical framework for graph transformation. Fundam. Inform. 74(1), 1–29 (2006)
7. Ferrari, G.L., Hirsch, D., Lanese, I., Montanari, U., Tuosto, E.: Synchronised hyperedge replacement as a model for service oriented computing. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2005. LNCS, vol. 4111, pp. 22–43. Springer, Heidelberg (2006)
8. Habel, A., Heckel, R., Taentzer, G.: Graph grammars with negative application conditions. Fundam. Inform. 26(3/4), 287–313 (1996)
9. Hirsch, D., Montanari, U.: Synchronized hyperedge replacement with name mobility. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 121–136. Springer, Heidelberg (2001)
10. Johnstone, P.T., Lack, S., Sobocinski, P.: Quasitoposes, quasiadhesive categories and Artin glueing. In: Mossakowski, T., Montanari, U., Haveraaen, M. (eds.) CALCO 2007. LNCS, vol. 4624, pp. 312–326. Springer, Heidelberg (2007)
11. Lack, S., Sobocinski, P.: Adhesive categories. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 273–288. Springer, Heidelberg (2004)
12. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, I. Inf. Comput. 100(1), 1–40 (1992)
13. Montanari, U., Pistore, M.: History-dependent automata: An introduction. In: Bernardo, M., Bogliolo, A. (eds.) SFM-Moby 2005. LNCS, vol. 3465, pp. 1–28. Springer, Heidelberg (2005)
14. Prange, U., Ehrig, H.: From algebraic graph transformation to adhesive HLR categories and systems. In: Bozapalidis, S., Rahonis, G. (eds.) CAI 2007. LNCS, vol. 4728, pp. 122–146. Springer, Heidelberg (2007)
15. Rensink, A.: A first study of compositionality in graph transformation. Technical Report TR-CTIT-10-08, Centre for Telematics and Information Technology, University of Twente (2010)
16. Taentzer, G.: Parallel high-level replacement systems. TCS 186(1-2), 43–81 (1997)