Scientific Annals of Computer Science vol. 22 (1), 2012, pp. 105–145
DOI: 10.7561/SACS.2012.1.105

Structured Operational Semantics for Graph Rewriting[1]

Andrei DORMAN[2], Tobias HEINDEL[3], Barbara KÖNIG[4]

Abstract

Process calculi and graph transformation systems provide models of reactive systems with labelled transition semantics (LTS). While the semantics for process calculi is compositional, this is not the case for graph transformation systems, in general. Hence, the goal of this article is to obtain a compositional semantics for graph transformation system in analogy to the structural operational semantics (SOS) for Milner's Calculus of Communicating Systems (CCS).

The paper introduces an SOS style axiomatization of the standard labelled transition semantics for graph transformation systems that is based on the idea of minimal reaction contexts as labels, due to Leifer and Milner. In comparison to previous work on inductive definitions of similarly derived LTSs, the main feature of the proposed axiomatization is a composition rule that captures the communication of sub-systems so that it can feature as a counterpart to the communication rule of CCS.

Keywords: process calculi, graph transformation, structural operational semantics, compositional methods

[1]This work was partially supported by grants from Agence Nationale de la Recherche, ref. ANR-08-BLANC-0211-01 (COMPLICE project) and ref. ANR-09-BLAN-0169 (PANDA project).

[2] LIPN – UMR 7030, Université Paris 13, 99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France. Email: andrei.dorman@lipn.univ-paris13.fr

[3] CEA LIST, Institut CARNOT CEA LIST, DILS/LMEASI, Point Courrier 174, 91191 Gif-sur-Yvette CEDEX, France. Email: tobias.heindel@cea.fr

[4] Abteilung für Informatik und Angewandte Kognitionswissenschaft, Fachbereich Theoretische Informatik, Universität Duisburg-Essen, Campus Duisburg, Fakultät für Ingenieurwissenschaften, D-47048 Duisburg, Germany. Email: barbara_koenig@uni-due.de

1 Introduction

Process calculi remain one of the central tools for the description of interactive systems. The archetypal examples of process calculi are Milner's π-calculus and the even more basic calculus of communication systems (CCS). The semantics of these calculi is given by labelled transition systems (LTS), which can be given as structural operational semantics (SOS). An advantage of SOS is their potential for combination with compositional methods for the verification of systems (see e.g. [27]).

Fruitful inspiration for the development of LTS semantics for other "non-standard" process calculi originates from the area of graph transformation where techniques for the derivation of LTS semantics from "reaction rules" have been developed [24, 10]. The strongest point of these techniques is the context independence of the resulting behavioural equivalences, i.e. they are congruences. Moreover, label derivation techniques lead to original LTS-semantics for the ambient calculus [22, 3], which are also given as SOS systems. Already in the special case of ambients, the SOS-style presentation goes beyond the standard techniques of label derivation in [24, 10]. An open research challenge is the development of a general technique for the canonical derivation of SOS-style LTS-semantics. We shall address the problem of the "monolithic" character of the standard LTS for graph transformation systems.

In the present paper, we provide the basic results for a solution to the problem. In particular we describe a CCS-*like* labelled transition semantics for graph transformation systems. The main guiding idea is a "formal" analogy to CCS and the crucial point is the quest for a suitable counterpart of the *communication rule* of CCS.

The focus on the communication rule of CCS is the main feature that distinguishes the present paper from previous work on graph transformation systems that has been inspired by structural operational semantics. As a further delineation, we only consider labelled transition semantics for graph transformation systems that use the idea of "minimal" reaction contexts as labels. The idea of a "minimal" context has been formalized by Leifer and Milner as relative pushout in the seminal work [18]; for the concrete case of hypergraph transformation, the Borrowed Context technique [10] is a more direct, equivalent approach. Within this research field on system behaviour that is based on minimal contexts as actions, the focus on a counterpart of the communication rule of CCS is the main novelty. It is the distinguishing feature in comparison to the graphical encoding of the ambient calculus that has been studied in [3]. Both the latter work and the present paper work

with graph transformation systems instead of the term based approach that has been used in [22]. The exploration of the subtle differences between term based and graphical techniques in general is however beyond the scope of the present paper.

The general direction of our work is towards formal results that support the slogan that graph transformation and process calculi have essentially the same descriptive power (as witnessed by several encodings of process calculi as graph transformation systems). This slogan is well-established for reduction semantics of closed systems and calls for an extension to open systems. On the way, we also expect to gain insight on the subtle differences between the "flat" world of graphs, and the term based world of process calculi. The potential advantage of graph transformation over process calculi is their inherent generality, as one will seldom study a particular graph transformation system for its own sake; the results hold for graph transformation systems in general. In this paper, we provide basic results on interaction in graph transformation systems that are meant to facilitate the "importation" of the usual structural operational semantics from the world of process calculi.

Structure and contents of the paper We first recall the basic definitions and concepts for the concrete case of hypergraphs in Section 2; in particular we give a brief review of the Borrowed Context technique. In Section 3, we provide a reformulation of the Borrowed Context technique in analogy to Milner's CCS; however, this analogy is imperfect as there is no need for a counterpart of the communication rule. This issue is addressed in Section 4, where we present our main results, which allow to define a graph transformation counterpart of a communication rule. These results are applied in Section 5 to obtain a satisfactory SOS like reformulation of the Borrowed Context technique. We conclude with a discussion of future and related work.

2 Preliminaries

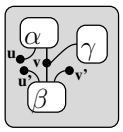
We first recall the standard definition of (hyper-)graphs and a formalism of transformation of hypergraphs (following the double pushout approach). We also present the labelled transition semantics for hypergraph transformation systems that has been proposed in [10]. In the present paper, the more general case of categories of graph-like structures is not of central importance.

We avoid category theoretical jargon and present all necessary concepts concretely for hypergraphs.

2.1 Hypergraphs

Definition 1 (Hypergraph)**.** *Let* Λ *be a set of* labels *with associated* arity *function* $\mathrm{ar}\colon \Lambda \to \mathbb{N}$*. A* ($\Lambda$-labelled hyper-) graph *is a tuple* $G = (E, V, \ell, \mathrm{cnct})$ *where* E *is a set of* (hyper-) edges*,* V *is a set of* vertices *or* nodes*,* $\ell\colon E \to \Lambda$ *is the* labelling function*, and* cnct *is the* connection *function, which assigns to each edge* $e \in E$ *a finite sequence of vertices* $\mathrm{cnct}(e) = v_1 \cdots v_n$ *where* $\mathrm{ar}(\ell(e)) = n$*, i.e.* $\mathrm{cnct}(e)$ *is a function from* $\{i \mid 0 < i \leq \mathrm{ar}(\ell(e))\}$ *to* V*. For a given edge* $e \in E$*, the set of all* e-adjacent vertices *is* $\mathrm{adj}(e) = \{\mathrm{cnct}(e)(i) \mid 0 < i \leq \mathrm{ar}(\ell(e))\}$ *and for a given node* $v \in V$*, the set of edges incident to it is* $\mathrm{inc}(v) = \{e \in E \mid v \in \mathrm{adj}(e)\}$*. The* degree *of a node* $v \in V$*, written* $\deg(v)$*, is the number of edges incident to it, i.e.* $\deg(v) = |\mathrm{inc}(v)|$ *(where for any finite set* M*, the number of elements of* M *is* $|M|$*). We also write* $v \in G$ *and* $e \in G$ *if* $v \in V$ *and* $e \in E$ *to avoid clutter.*

Example 2.1 (Hypergraph)**.** *An example of an* $\{\alpha, \beta, \gamma\}$*-labelled hypergraph is illustrated in the grey box in the display below.*



The arities of α*,* β *and* γ *are* 2*,* 3 *and* 1*, respectively. The graph has hyperedges of each "type", which are depicted as rounded boxes with the respective label inside; moreover, the graph has four nodes. The order of the nodes that are connected to an hyperedge is usually not important (but could be fixed easily by counting counter-clockwise from the bottom left corner of the edge).*

We usually do not discern isomorphic graphs (which roughly corresponds to the practice to consider terms of process calculi up to structural congruence); thus, we usually do not mention the "names" of nodes and edges. The full details of the above graph would be $(\{e, e', d\}, \{u, v, u', v'\}, \ell, \mathrm{cnct})$ *where* $\ell = \{e \mapsto \alpha, e' \mapsto \beta, d \mapsto \gamma\}$ *and* $\mathrm{cnct} = \{e \mapsto uv, e' \mapsto v'vu', d \mapsto v\}$*.*

For completeness' sake, we recall the standard definitions of hypergraph morphism, sub-graph and isomorphism. However, usually, inclusions of

sub-graphs in illustrations are the obvious ones that preserve the (relative) positioning of nodes and edges.

Definition 2 (Hypergraph morphisms, inclusions, isomorphisms)**.** *Let* $G_i = (E_i, V_i, \ell_i, \mathrm{cnct}_i)$ *(*$i \in \{1, 2\}$*) be hypergraphs; a* hypergraph morphism *from* G_1 *to* G_2*, written* $f\colon G_1 \to G_2$*, is a pair of functions* $f = (f_E\colon E_1 \to E_2, f_V\colon V_1 \to V_2)$ *that preserves labels and connectivity of edges: The equality* $\ell_2 \circ f_E = \ell_1$ *holds and we have* $f_V(\mathrm{cnct}_1(e)(i)) = \mathrm{cnct}_2(f_E(e))(i)$ *for each edge* $e \in E_1$ *and every* $i \in \mathbb{N}$ *such that* $0 < i \leq \mathrm{ar}(\ell(e))$*.*

A hypergraph morphism $f = (f_E, f_V)\colon G_1 \to G_2$ *is* injective (an isomorphism) *if both* f_E *and* f_V *are injective (bijective); it is* the inclusion (of G_1 into G_2) *if both* $f_E(e) = e$ *and* $f_V(v) = v$ *hold for all* $e \in E_1$ *and* $v \in V_1$ *and then* G_1 *is a* sub-graph *of* G_2*. As usual, we write* $G_1 \cong G_2$ *if there is an isomorphism* $f\colon G_1 \to G_2$*. We write* $G_1 \hookrightarrow G_2$ *or* $G_2 \hookleftarrow G_1$ *(and very often just* $G_1 \to G_2$ *or* $G_2 \leftarrow G_1$*) if* G_1 *is a sub-graph of* G_2*.*

In this subsection we have recalled the basic terminology for hypergraphs; next we shall review a standard approach to graph transformation.

2.2 Standard Graph Transformation

The most established approach to graph transformation is double pushout rewriting. It is most succinctly defined using the basic category theoretical notion of pushout, which makes it a uniform approach for arbitrary graph-like structures. However, for the particular case that we are interested in, pushouts can be understood as a variation of the disjoint union of hypergraphs. In the next definition, we also cover the particular case of pullbacks that we shall need to properly present the Borrowed Context technique [10] in Section 2.3 without a purely category theoretical perspective.

Definition 3 (Pullbacks & pushouts of inclusions)**.** *Let* $G_i = (E_i, V_i, \ell_i, \mathrm{cnct}_i)$ *(*$i \in \{0, 1, 2, 3\}$*) be hypergraphs and let* $G_1 \to G_3 \leftarrow G_2$ *be inclusions. The* intersection of G_1 and G_2 *is the hypergraph* $G' = (E_1 \cap E_2, V_1 \cap V_2, \ell', \mathrm{cnct}')$ *where* $\ell'(e) = \ell_1(e)$ *and* $\mathrm{cnct}'(e) = \mathrm{cnct}_2(e)$ *for all* $e \in E_1 \cap E_2$*. The* pullback of $G_1 \to G_3 \leftarrow G_2$ *is the pair of inclusions* $G_1 \leftarrow G' \to G_2$ *and the resulting square is a pullback square (see Figure 1).*

Let $G_1 \leftarrow G_0 \to G_2$ *be inclusions; they are* non-overlapping *if both* $E_1 \cap E_2 \subseteq E_0$ *and* $V_1 \cap V_2 \subseteq V_0$ *hold. The* pushout of non-overlapping inclusions $G_1 \leftarrow G_0 \to G_2$ *is the pair of inclusions* $G_1 \to (G_1 +_{G_0} G_2) \leftarrow G_2$

where $(G_1 +_{G_0} G_2) = (E_1 \cup E_2, V_1 \cup V_2, \ell'', \mathrm{cnct}'')$ *is the hypergraph such that*

$$\ell''(e) = \begin{cases} \ell_1(e) & \text{if } e \in E_1 \\ \ell_2(e) & \text{if } e \in E_2 \end{cases} \text{ and } \mathrm{cnct}''(e) = \begin{cases} \mathrm{cnct}_1(e) & \text{if } e \in E_1 \\ \mathrm{cnct}_2(e) & \text{if } e \in E_2 \end{cases}$$

hold for all $e \in E_1 \cup E_2$*; often,* $(G_1 +_{G_0} G_2)$ *is referred to as* the pushout of G_1 and G_2 over G_0.

Without loss of generality, we shall assume that pairs of inclusions $G_1 \leftarrow G_0 \rightarrow G_2$ are always non-overlapping.

$$\begin{array}{ccc} G_3 & \longleftarrow & G_2 \\ \uparrow & & \uparrow \\ G_1 & \longleftarrow & G' \end{array} \qquad \begin{array}{ccc} G_0 & \longrightarrow & G_2 \\ \downarrow & & \downarrow \\ G_1 & \longrightarrow & G'' \end{array}$$

Figure 1: Pullback and pushout square

We are finally ready to introduce graph transformation systems and their "reduction" semantics.

Definition 4 (Rules and graph transformation systems)**.** *A* rule *is a pair of inclusions of hypergraphs* $\rho = (L \leftarrow I \rightarrow R)$*. Let* A, B *be hypergraphs. Now,* ρ transforms A to B *if there exists a diagram as shown to the right in which the two squares are pushouts,* $A' \leftarrow I \rightarrow R$ *is non-overlapping, and* $A \cong A'$ *and* $B' \cong B$*. A* graph transformation system (GTS) *is a pair* $\mathcal{S} = (\Lambda, \mathcal{R})$ *where* Λ *is a set of labels and* $\mathcal{R}$ *is a set of rules (over* Λ*-labelled hypergraphs).*

$$\begin{array}{ccccc} L & \longleftarrow & I & \longrightarrow & R \\ \downarrow & & \downarrow & & \downarrow \\ A' & \longleftarrow & D & \longrightarrow & B' \end{array}$$

A graph transformation rule can be understood as follows. Whenever the left hand side L is (isomorphic to) a sub-graph of some graph A then this sub-graph can be "removed" from A, yielding the graph D. The vacant place in D is then "replaced" by the right hand side R of the rule. The middleman I is the memory of the connections that L had with the rest of the graph in order for R to be attached in exactly the same place. Also note, that if we remove a node in A, we have also to remove all incident edges explicitly, i.e. the deleted node has "the same" incident edges in L and A. Each graph transformation step can also be thought of as a chemical reaction according to the rule, which features as the reaction law. An example of a rewriting step is shown in Figure 2. As mentioned before, inclusions are

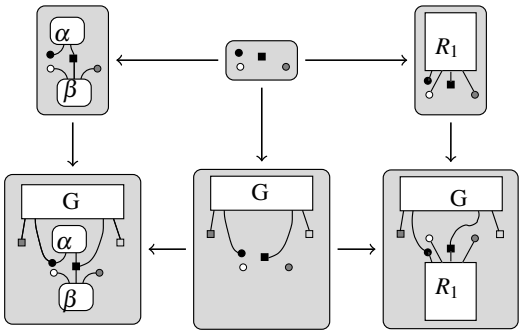


Figure 2: A rewriting step with rule "α/β".

given implicitly by the spatial arrangement of nodes and edges to keep the graphical representations clear.

As one might expect, the result of each transformation step is unique (up to isomorphism). This is a consequence of the following fact.

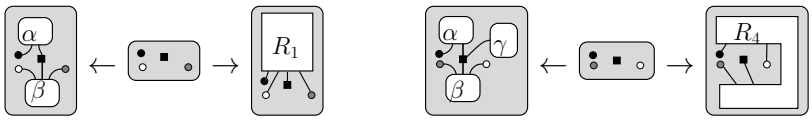
Fact 1 (Pushout complements)**.** *Let $G_2 \leftarrow G_1 \leftarrow G_0$ be a pair of hypergraph inclusions where $G_i = (E_i, V_i, \ell_i, \mathrm{cnct}_i)$ ($i \in \{0,1,2\}$) and assume that they satisfy the* dangling edge condition*: For all $v \in V_1 \setminus V_0$ there does not exist any edge $e \in E_2 \setminus E_1$ such that e is incident to v. Then there exists a unique sub-graph $G_2 \leftarrow D$ such that* (1) *is a pushout square.*

$$\begin{array}{ccc} G_1 & \longleftarrow & G_0 \\ \downarrow & & \downarrow \\ G_2 & \longleftarrow & D \end{array} \qquad (1)$$

Definition 5 (Pushout Complement)**.** *Let $G_2 \leftarrow G_1 \leftarrow G_0$ be a pair of hypergraph inclusions that satisfy the conditions of Fact 1; the unique completion $G_2 \leftarrow D \leftarrow G_0$ that yields the pushout square* (1) *is the* pushout complement *of $G_2 \leftarrow G_1 \leftarrow G_0$.*

We now introduce the example graph transformation system that we shall use throughout the paper to illustrate the basic ideas. The rule in the "reaction" in Figure 2 is part of this system.

Example 2.2 (Running Example)**.** *The system $\mathcal{S}_{ex} = (\Lambda, \mathcal{R})$ will be the following one in the sequel: The set of edge-labels is $\Lambda = \{\alpha, \beta, \gamma, \dots\}$ where $\mathtt{ar}(\alpha) = 2$, $\mathtt{ar}(\beta) = 3$ and $\mathtt{ar}(\gamma) = 1$; moreover, $\mathcal{R}$ is the set of rules given in Figure 3 where the R_i represent different graphs (e.g. edges with labels R_i).*



(a) Rule "α/β" (b) Rule "$\alpha/\beta/\gamma$"

Figure 3: Reaction rules of $\mathcal{S}_{ex}$.

In this subsection, we have presented the double pushout approach as a model for "reactions" that occur in a system about which one has complete knowledge. Thus rewriting is similar to the reaction semantics for CCS. Now, we come to the more recent and central idea that graph transformation systems "automatically" have an interactive nature, which endows each graph with a behaviour. This is similar to process calculi where process terms cannot only react but also exhibit behaviour that depends on possible interactions with other processes.

2.3 Behaviour as Interaction With the Environment

Finally, we use the Borrowed Context technique [10] to equip each graph transformation system with a labelled transition semantics that models the interactive behaviour of systems that are specified as graphs with graph transformation rules. Each labelled transition will model an interaction of a graph with an "external" environment; in the world of process calculi, this environment is formalized as an arbitrary (reactive) context.

Intuitively, one might want to "hide" parts of a graph from the environment while some portion of the state is directly exposed. Thus, each state of the labelled transition system (LTS) will be a graph with an interface, which makes part of the graph directly accessible whereas the remainder is "hidden"; more technically, the interface is also needed to have a meaningful way to consider the graph within an "external" context. To avoid confusion, we emphasize here that the labels of transitions in the LTS will depend directly on the rules of the GTS (and thus only indirectly on the edge-labels Λ). We use the standard definition of labelled transition systems, which we recall here to fix notation.

Definition 6 (Labelled transition system)**.** *A* labelled transition system *(*LTS*) is a tuple* (S, Ξ, R) *where* S *is a set of* states*,* Ξ *is a set of* labels *and*

$R \subseteq S \times \Xi \times S$ *is the* transition relation. *We write*

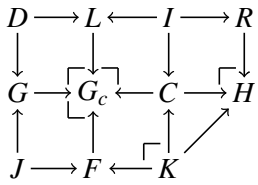
$$s \xrightarrow{a} s'$$

if $(s, a, s') \in R$ *and say that* s *can evolve to* s' *by performing* a.

Before we delve into the technical details of the LTS semantics for graphs, we first discuss the main ideas informally. The states will be graphs with interface $J \to G$. The "larger" part G models the whole "internal" state of the system while the "smaller" part, the interface J, models the part that is directly accessible to the environment and allows for (non-trivial) interaction. As a particularly simple example, one could have a Petri net where the set of places (with markings) is the complete state and some of the places are "open" to the environment such that interaction takes place by exchange of tokens.

More generally, the addition of agents/resources from the environment to (the interface of) a state might result in "new" reactions, which have not been possible before. In the Petri net scenario, extra tokens might enable transitions that could not be fired before. The idea of the LTS semantics for graph transformation is to consider as labels "minimal" contexts that trigger "new" reactions (by providing extra agents/resources). For a more formal treatment of this intuition, see [25]. A direct definition of the LTS semantics for graph transformation can be given in terms of so-called borrowed context diagrams.

Definition 7 (DPOBC)**.** *Let* $\mathcal{S} = (\Lambda, \mathcal{R})$ *be a graph transformation system. Its* LTS *has all inclusions of hypergraphs* $J \to G$ *as* states *where* J *is called the* interface. *Labels are pairs of inclusions* $J \to F \leftarrow K$. *A state* $J \to G$ *evolves to another one* $K \to H$ *if there is a diagram as shown to the right, which is called a* DPOBC-diagram *or just a* BC-diagram*: All morphisms are injective and the squares are pullbacks or pushouts as marked. In such a* BC*-diagram,* $G \leftarrow D \to L$ *is called the* partial match of L (in G).



We often speak of the graph D as the partial match for a transition; in fact, the whole BC-diagram is determined by the rule and the partial match (up to isomorphism).

Example 2.3. *An example of a* DPOBC*-diagram is given in Figure 4. The original state* $J \to G$ *contains a hyperedge* α. *The partial match* D *is exactly*

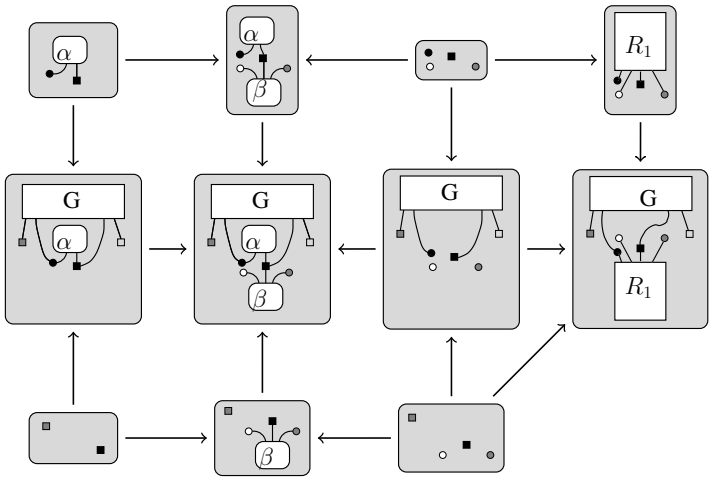


Figure 4: An example of a BC-diagram that uses the rule "α/β".

this edge. The state can therefore evolve using rule α/β, provided that it borrows β from the environment.

The bigger graph G_c contains G but is also completed with what is missing such that the left-hand side of the rule can be embedded into it. It can thus be rewritten, as shown in Figure 2. The last row shows the "evolution of the interface during the rewriting": First, we have the original interface; then the missing part is added; finally, some elements from the interface have to be removed if they have been deleted by application of the rule.

That the label $J \to F \leftarrow K$ in Definition 7 is "minimal" is captured by the two leftmost squares in the BC diagram above: The "addition" $J \to F$ is "just enough" to complete the partial match of the left hand side of the rule $L \leftarrow I \to R$. That the interaction with the environment involves a reaction is captured by the other two squares in the upper row in the BC-diagram. During this reaction, some agents might disappear or some resources might be used (depending on the preferred metaphor) and new ones might come into play. Finally the bottom left pullback square in the BC-diagram restricts the changes to obtain the new interface into the resulting state.

Different rules might result in different deletion effects that are "visible"

to the environment. Thus, the full label of each such "new" reaction is the "trigger" $J \to F$ together with the "observable" change $F \leftarrow K$ (with state $K \to H$ after interaction). Note that more recent process calculi also have several reaction rules (as for example the ambient calculus) while CCS has only a single one.

3 A Process Calculus Perspective on Borrowed Contexts

We begin with a reformulation of the Borrowed Context technique that breaks the "monolithic" BC-steps into axioms and rules. We only assume familiarity with process calculi and in particular do not require knowledge of the Borrowed Context technique (beyond the definition in the previous section). This section should also clarify the purpose and relevance of the main results in Section 4 and Section 5.

We begin with an informal motivation by developing an analogy with the axioms and rules of CCS. The axioms will provide *basic actions*, which can be seen as a generalization of transitions of the form $\alpha.P \xrightarrow{\alpha} P$ in CCS. After a quick review of how contexts are formalized in graph transformation, we shall give rules that allow to "embed" transitions into contexts; these rules are similar to the rule that allows to infer $P \parallel Q \xrightarrow{\alpha} P' \parallel Q$ from $P \xrightarrow{\alpha} P'$ in CCS because $[\cdot] \parallel Q$ is a "non-interfering" context. Finally, we show formally that axioms for basic actions with two "contextualization" rules exactly capture the Borrowed Context technique.

3.1 The analogy with CCS

The axioms of our system will be similar to the axioms of CCS, where the process $\alpha.P$ can perform the action α and then behaves as P; this is usually written $\alpha.P \xrightarrow{\alpha} P$ where α ranges over the actions $a, \overline{a}$ (and τ). In particular, a and $\overline{a}$ are co-actions of each other, which are consumed during the reaction of $a.P$ and $\overline{a}.P$ in the combined process $a.P$ and $\overline{a}.P$.

In the case of graphs, each rule $L \leftarrow I \to R$ gives rise to a whole family of such actions – one for each subgraph of L. More precisely, each subgraph D of L can be seen as an "action"; each such action has a co-action $\widehat{D}^L \to L$ such that L is the union of D and $\widehat{D}^L$ (and $\widehat{D}^L$ is the minimal sub-graph with this property). For example, in the rule α/β, both edges α and β yield (complementary) basic actions. Indeed, to make the analogy closer, the

common node between the two edges in the left hand side of the rule α/β is the analogue of a channel; one of the edges performs the input and the other the output "on" the common node.

Formally, in Table 1, we have the family of *Basic Action* axioms. It essentially represents all the possible uses of a transformation rule. In (an encoding of) CCS, the left hand side would be a pair of unary edges a and $\overline{a}$, which both disappear during reaction. Now, if only a is present "within" the system, it needs $\overline{a}$ to perform a reaction; thus, the part a of the left hand side induces the (inter-)action that consists in "borrowing" $\overline{a}$ and deleting both edges (and similarly for $\overline{a}$). In general, e.g. in the rule $\alpha/\beta/\gamma$, there might be more than two edges that are involved in a reaction and thus we have a whole family of actions. More precisely, each portion of a left hand side induces the action that consists in borrowing the missing part to perform the reaction (thus obtaining the complete left hand side), followed by applying the changes that are described by the rule.

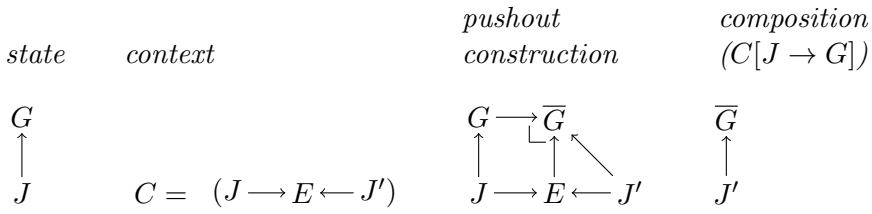
Next, we shall describe counterparts for the two CCS-rules that allow to perform a given action in parallel to another process and under a restriction; the respective forms of contexts in which actions can be performed are "parallel contexts" $[\cdot] \parallel Q$ and "restriction contexts" $(\nu b)[\cdot]$. More precisely, whenever we have the transition $P \xrightarrow{\alpha} P'$ in CCS and another process Q, then there is also a transition $P \parallel Q \xrightarrow{\alpha} P' \parallel Q$; similarly, we also have $(\nu b)P \xrightarrow{\alpha} (\nu b)P'$ whenever $\alpha \notin \{\overline{b}, b\}$. More abstractly, actions are preserved by certain contexts and not by others; for example $a.[\cdot]$ does block all actions.

In the case of graph transformation, there is a natural counterpart for process contexts $C[\cdot]$ such as $P \parallel [\cdot]$ and $(\nu b)[\cdot]$. The only complication is that graphs have arbitrary interfaces $J \to G$ (see also Definition 7) while processes have a sacrosanct "interface", viz. their free names. Thus, graph contexts have a "type", which is an interface graph J; only states with interface J can be put into a context of this "type". The result is called the composition[5] of the state with the context.

Definition 8 (Context and composition)**.** *Let $J \to G$ be a state. A* context (of type J) *is a pair of inclusions $C = J \to E \leftarrow J'$. The* composition *of $J \to G$ with the context C, written $C[J \to G]$, is the inclusion of J' into the pushout of $E \leftarrow J \to G$ as illustrated in the following display (with the*

[5]The reason for this is that the construction in Definition 8 is essentially the composition of co-spans.

assumption that C is free for $J \to G$).



The left inclusion of the context, i.e. $J \to E$ in the definition, can also be seen as a state with the same interface. The pushout then gives the result of "gluing" E to the original G at its interface J; the second inclusion $J' \to E$ models a new interface, which possibly contains part of J and additional "new" entities in E.

The idea of our stratified presentation of the Borrowed Context technique is based on the observation that each BC-transition

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

which might be a basic action or not, remains roughly unchanged in contexts of a certain form; in other words, some contexts C allow $C[J \to G]$ to perform "the same" action as $J \to G$.

With this observation in mind, we shall first characterize two classes of contexts that are *non-interfering* in the described sense. These two classes roughly correspond to CCS contexts of the form $P \parallel [\cdot]$ and $(\nu b)[\cdot]$. However, even though "non-interfering" contexts have no substantial influence on actions, we will have to keep track on what they add and how they change interfaces. Finally, at the end of this section, we show that axioms for basic actions together with the two natural contextualization rules for non-interfering contexts yield a sound a complete description of the Borrowed Context technique.

3.2 Borrowed Contexts in Three Layers

In this subsection we shall provide the formal details of a process calculus like presentation of the Borrowed Context technique. We have already discussed the idea of basic actions and non-interfering contexts. We fix now a graph transformation system $\mathcal{S} = (\Lambda, \mathcal{R})$ and – relative to this parameter – define the system 3L.

Basic Actions We start with the axioms of the system. They derive the basic actions as discussed above; and example of a basic action is given in Figure 5.

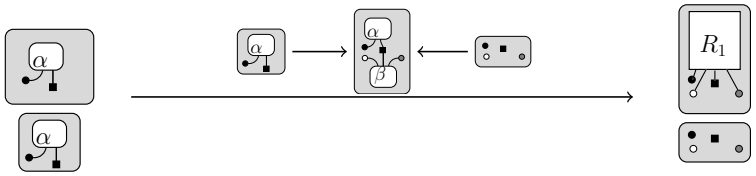


Figure 5: An example of a basic action.

Definition 9 (Basic Action axioms)**.** *Let* $(L \leftarrow I \rightarrow R) \in \mathcal{R}$ *be a rule and let* $D \rightarrow L$ *be a sub-graph. Then*

$$\frac{}{(D \rightarrow D) \xrightarrow{D \rightarrow L \leftarrow I} (I \rightarrow R)}$$

is a Basic Action Axiom *of* 3L*.*

Interface Narrowing Next, we address the counterpart of name restriction. This means, we first define the counterpart of CCS-contexts of the form $(\nu a)[\cdot]$; these are just contexts of the form $C = J \rightarrow J \leftarrow J'$, which will be called *narrowing contexts*. Intuitively, such a context does not interfere with a transition with label $J \rightarrow F \leftarrow K$ if J' is still big enough to glue all new entities in F. This is in direct analogy to CCS, where the restriction (νa) preserves only those actions that do not involve a. While we do not have to adjust labels in CCS, even non-interfering narrowing context "narrows" the label of the transition while the "proper" action remains untouched. This is made formal in the following definition and the analogy to CCS is made more precise afterwards.

Definition 10 (Narrowing)**.** *A* narrowing context *is a pair of inclusions* $C = J \rightarrow J \leftarrow J'$ *in which only the right inclusion might be proper. Let* $J \rightarrow F \leftarrow K$ *be a label. The narrowing context* $C = J \rightarrow J \leftarrow J'$ *does* not interfere *with the label if the pushout complement of* $F \leftarrow J \leftarrow J'$ *exists. If* C *is non-interfering, then the* C-narrowing *of the label, written* $C\langle J \rightarrow F \leftarrow K\rangle$*, is the lower row in the following display*

$$C\langle J \rightarrow F \leftarrow K\rangle := \begin{array}{ccccc} J & \longrightarrow & F & \longleftarrow & K \\ \uparrow & & \uparrow & & \uparrow \\ J' & \longrightarrow & F' & \longleftarrow & K' \end{array} \qquad \text{where } C = J \rightarrow J \leftarrow J'$$

where the left square is a pushout and the right one a pullback. Whenever we write $C\langle J \to F \leftarrow K\rangle$, we assume that the relevant pushout complement exists.

If we think of the interface as the set of free names of a process, then restricting a name means removing it from the interface. Thus, J' plays the role of the set of all remaining free names. If the pushout complement F' exists, it represents F with the restricted names erased. Finally, since a pullback here can be seen as an intersection, K' is K without the restricted names. So we finally obtain the "same" label where "irrelevant" names are not mentioned. It is of course not always possible to narrow the interface. For instance, one cannot restrict the names that are involved in labelled transitions of CCS-like process calculi. This impossibility is captured by the non-existence of the pushout complement.

Narrowing contexts just make interfaces smaller; the remainder of the involved states is left unchanged. An example of interface narrowing is given

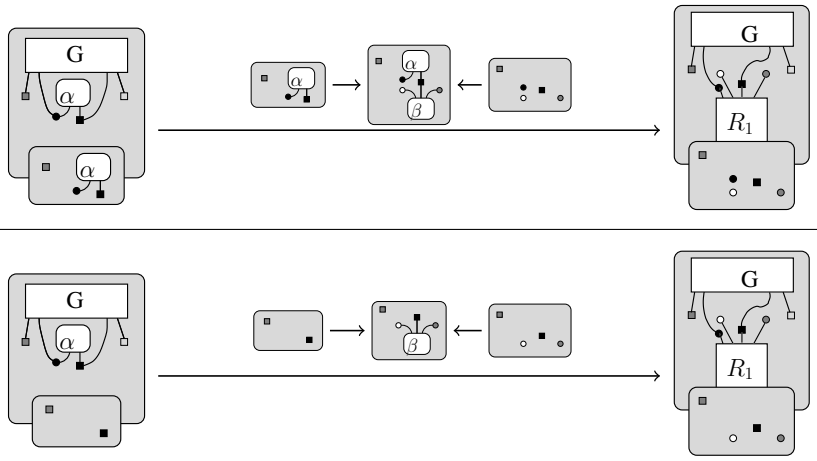


Figure 6: An example of interface narrowing

in Figure 6 where the narrowing context removes the α-labelled edge and the first of its nodes from the interfaces. Interface narrowing yields the first rule scheme in the system 3L.

Definition 11 (Narrowing rule)**.** *Let $J \to F \leftarrow K$ be a label, let $C = J \to J \leftarrow J'$ be a non-interfering narrowing context, and let $J' \to F' \leftarrow K' = C\langle J \to F \leftarrow K\rangle$ be the C-narrowing of the label; moreover let $J \to G$ and*

$K \to H$ be inclusions. Then

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

is an instance of the narrowing rule of 3L.

Compatible contexts It remains to define contexts that correspond to parallel composition with another process P in CCS. In the case of graph transformation, this case is slightly more involved than one might expect. The problem is that even the "pure" addition of context potentially interferes with transitions. For example, if an interaction involves the deletion of an (isolated) node in the interface, the addition of an edge to this node blocks the reaction. Thus a context that only adds new entities, which will be called *monotone*, interferes if it creates dangling edges. Non-interfering, monotone contexts are intuitively similar to CCS-contexts of the form $P \parallel [\cdot]$; they are called *compatible*.

Definition 12 (Compatible contexts)**.** *Let $C = J \to E \leftarrow \overline{J}$ be a context; it is* monotone *if $J \to \overline{J}$. Let $J \to F \leftarrow K$ be a label; now C does not* interfere with *$J \to F \leftarrow K$ if it is possible to construct the diagram in* (2) *where both squares are pushouts. Finally, a context $J \to E \leftarrow \overline{J}$ is* compatible *with the label $J \to F \leftarrow K$ if it is monotone and does not interfere with the label.*

$$\begin{array}{ccccc} J & \longrightarrow & F & \longleftarrow & K \\ \downarrow & & \downarrow & & \downarrow \\ E & \longrightarrow & E_1 & \longleftarrow & E' \end{array} \quad (2)$$

In a label $J \to F \leftarrow K$, the left inclusion represents the addition of new entities that "trigger" a certain reaction. A compatible context is simply a context that preserves the old interface and adds new entities that do not block the reaction, i.e. it does not add new edges to nodes that disappear during the interaction. An illustration of how to embed of a whole transition into a monotone context is given in Figure 7.

To properly define a rule for monotone contexts, we introduce a partial operation for the *combination* of co-spans with a common interface, which generalizes the narrowing construction.

Definition 13 (Cospan combination)**.** *Let $C = (J \to F \leftarrow K)$ and $\overline{C} = (J \to E \leftarrow \overline{J})$ be two co-spans. They are* combinable *if there exists a diagram of the following form.*

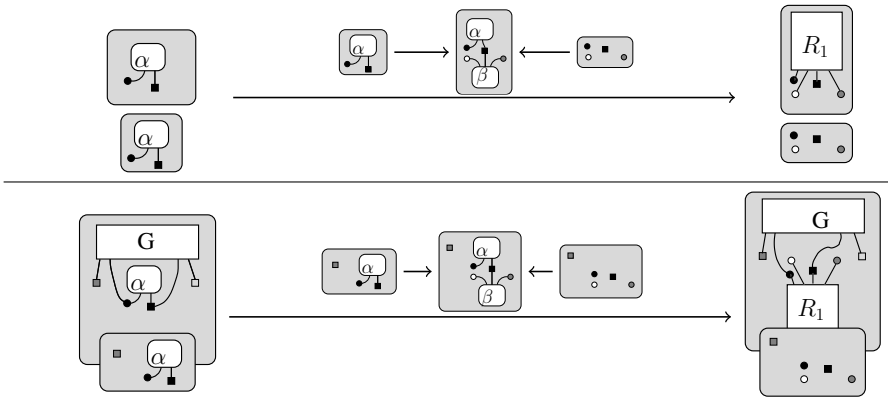


Figure 7: A transition in a monotone context.

$$\begin{array}{ccccc} J & \longrightarrow & F & \longleftarrow & K \\ \downarrow & & \downarrow & & \downarrow \\ E & \longrightarrow & E_1 & \longleftarrow & E' \\ \uparrow & & \uparrow & & \uparrow \\ \overline{J} & \longrightarrow & \overline{F} & \longleftarrow & \overline{K} \end{array} =: \overline{C}\langle J \to F \leftarrow K \rangle$$

The label $\overline{J} \to \overline{F} \leftarrow \overline{K}$ *is the* combination of C with $\overline{C}$*, and is denoted by* $\overline{C}\langle J \to F \leftarrow K \rangle$*.*

In fact, it is easy to show that compatible contexts are combinable with their label.

Lemma 1. *Given a label* $J \to F \leftarrow K$ *and a compatible context* $J \to E \leftarrow \overline{J}$ *, we can split the diagram in* (2) *to obtain the following diagram.*

$$\begin{array}{ccccc} J & \longrightarrow & F & \longleftarrow & K \\ \downarrow & & \downarrow & & \downarrow \\ \overline{J} & \longrightarrow & \overline{F} & \longleftarrow & \overline{K} \\ \downarrow & & \downarrow & & \downarrow \\ E & \longrightarrow & E_1 & \longleftarrow & E' \end{array} \qquad \textit{and therefore} \qquad \begin{array}{ccccc} J & \longrightarrow & F & \longleftarrow & K \\ \downarrow & & \downarrow & & \downarrow \\ E & \longrightarrow & E_1 & \longleftarrow & E' \\ \uparrow & & \uparrow & & \uparrow \\ \overline{J} & \longrightarrow & \overline{F} & \longleftarrow & \overline{K} \end{array}$$

With this lemma we can easily define the rule that corresponds to "parallel composition" of an action with another "process".

Definition 14 (Compatible Contexts rule)**.** *Let $J \to F \leftarrow K$ be a label, let $C = J \to E \leftarrow \overline{J}$ be a context that is compatible with it, and let $\overline{C} = (J \to F \leftarrow K)\langle C \rangle$ be the combination of C with the label; moreover let $J \to G$ and $K \to H$ be inclusions. Then*

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{C[J \to G] \xrightarrow{C\langle J \to F \leftarrow K \rangle} \overline{C}[K \to H]}$$

is an instance of the combination rule in 3L*.*

Soundness and Completeness The 3L-system, which consists of basic actions, the narrowing rule and the combination rule is summarized in Table 1. It does not only give an analogy to the standard SOS-semantics for CCS. In fact, we shall see that the labels that are derived by the standard BC technique are exactly those labels that can be obtained from the basic actions by compatible contextualization and interface narrowing. In technical terms, the 3L-system is sound and complete.

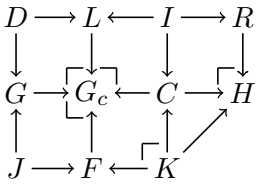
Theorem 2 (Soundness and completeness)**.** *Let $\mathcal{S}$ be a graph transformation system. Then there is a* BC*-transition*

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

if and only if it is derivable in the 3L*-system.*

Sketch of the proof: It is easy to build a DPOBC-diagram to justify the Basic Action axioms. We have seen while defining the Narrowing and Compatible Context rules that they are derivable with the BC technique. Let us now show completeness.

Let d be a DPOBC-diagram using the rule $\rho = L \leftarrow I \to R$; the resulting transition is $t = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ and the partial match is D.



We have that the transition $t_0 = (D \to D) \xrightarrow{D \to L \leftarrow I} (I \to R)$ is a basic action, i.e. derivable by an axiom of the system. Let $C = D \to G \leftarrow G$ be a

- **Basic Actions**

$$\frac{}{(D \to D) \xrightarrow{D \to L \leftarrow I} (I \to R)} \quad \text{where } (L \leftarrow I \to R) \in \mathcal{R} \text{ and } D \to L$$

- **Interface Narrowing**

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{(J' \to G) \xrightarrow{J' \to F' \leftarrow K'} (K' \to H)} \quad \text{where } C = J \to J \leftarrow J' \text{ and } J' \to F' \leftarrow K' = C[J \to F \leftarrow K]$$

- **Compatible Contextualization**

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{C[J \to G] \xrightarrow{C[J \to F \leftarrow K]} \overline{C}[K \to H]} \quad \text{where } C = J \to E \leftarrow \overline{J} \text{ is compatible with } J \to F \leftarrow K \text{ and } \overline{C} = (J \to F \leftarrow K)[C]$$

Table 1: Axioms and rules of the 3L-semantics.

monotone context. It is clearly non-inhibiting w.r.t. $D \to L \leftarrow I$. Thus it is compatible with it and $\overline{C}\langle D \to L \leftarrow I\rangle = G \to G_c \leftarrow C$. So one can use the contextualization rule, and obtain, from t_0, the transition

$$t'' = (G \to G) \xrightarrow{G \to G_c \leftarrow C} (C \to H).$$

Let $C' = G \to G \leftarrow J$. By uniqueness of pushout complement and pullback, the C-narrowing of $G \to G_c \leftarrow C$ is exactly $J \to F \leftarrow K$. Therefore the narrowing rule applied to t'' yields the original transition t.

As a result, for any DPOBC-diagram that justifies a transition t (where the names of the graphs are the usual ones, as in Definition 7), the following

three step derivation in 3L justifies t:

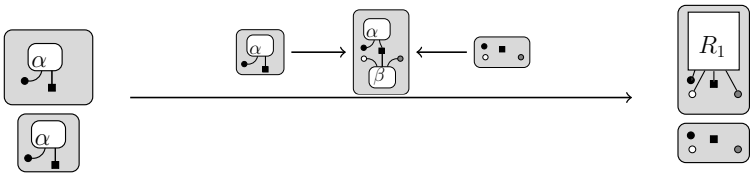
$$\dfrac{\dfrac{\dfrac{}{(D \to D) \xrightarrow{D \to L \leftarrow I} (I \to R)}\,\texttt{ax.}}{(G \to G) \xrightarrow{G \to G_c \leftarrow C} (C \to H)}\,\texttt{ctx.}}{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}\,\texttt{narr.}$$

Example 3.1. *An example of a derivation of the example transition is shown in Figure 8. We can see the basic action first, using the partial match of the diagram. Using compatible contextualization, we "add" to this transition, all that is in the original state. The interface is everything that is necessary; in this example, we just add an extra vertex (and we could have put some more objects in the interface of the monotone context). Finally, we remove from the interface everything that is not needed, to get the desired interface.*

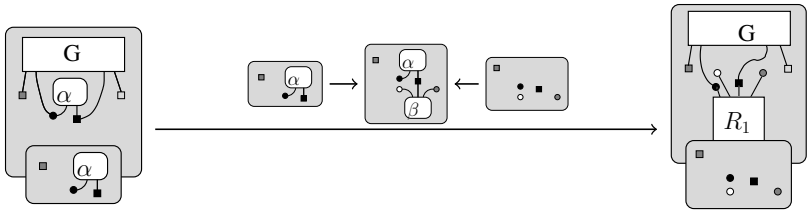
The main role of soundness and completeness is not its technical "backbone", which is similar to many other theorems on the Borrowed Context technique. The main insight to be gained is the absence of any "real" communication between sub-systems; roughly, every reaction of a state can be "localized" and then derived from a basic action (followed by contextualization and narrowing). In particular, we do not have any counterpart to the communication-rule in CCS, which has complementary actions $P \xrightarrow{a} P'$ and $Q \xrightarrow{\bar{a}} Q'$ as premises and allows to infer communication of the processes P and Q, i.e. a silent "internal" transition $P \parallel Q \xrightarrow{\tau} P' \parallel Q'$. This absence of communication in the "monolithic" BC-labels is the main motivation for our study of composition of transitions.

4 Communication in Composed States

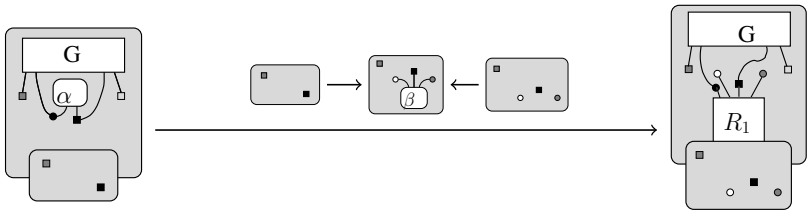
The formal analogy between CCS and the Borrowed Context technique that we have established in the previous section is imperfect: we have no counterpart to the communication rule of CCS, which allows to derive the reaction $P \parallel Q \xrightarrow{\tau} P' \parallel Q'$ of the "composed" state $P \parallel Q$ from the two interactions $P \xrightarrow{a} P'$ and $Q \xrightarrow{\bar{a}} Q'$ of the "constituents" P and Q. Thus, we shall now analyze when and how two labelled transitions from two different states in a GTS give rise to a "smaller" labelled transition of the composition of the two states. This analysis will lead to a counterpart of the communication rule of CCS that is admissible in the system 3L; moreover, as we shall exploit in Section 5, we can reduce the number of axioms.



(a) First, make the partial match evolve (Basic Action).



(b) Then, add what is missing from the original graph, thanks to Compatible Context.



(c) Finally, remove the interface you don't need, using Interface Narrowing.

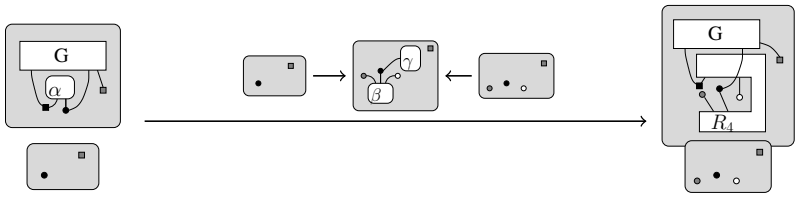
Figure 8: An example of a derivation in the 3L-system.

4.1 The idea of composition of transitions

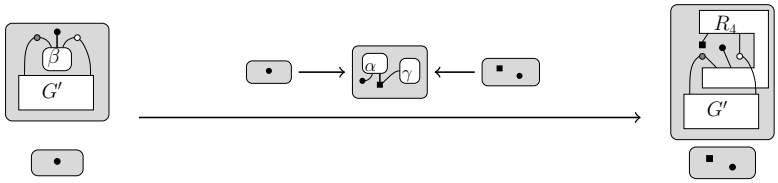
Communication within a composed state is based on the following idea: (sub-)states may provide resources for each other that they (used to) borrow from the environment; as a consequence, the composed state needs to borrow less from the environment. This idea is illustrated in the following example.

Example 4.1 (Composition of transitions)**.** *Let $s = J \to G$ be a state of $\mathcal{S}_{ex}$ that contains an edge α with its second connected node in the interface as shown in Figure 9(a). Further, let $s' = J' \to G'$ be a state that contains an edge β with its second connected node in the interface as shown in Figure 9(b). Both graphs can perform transitions t and t', using the rule $\alpha/\beta/\gamma$. Let X be the graph that consist of the black round vertex only, which is both a subgraph of J and J'. Now we can compose t and t' along $J \leftarrow X \to J'$ to*

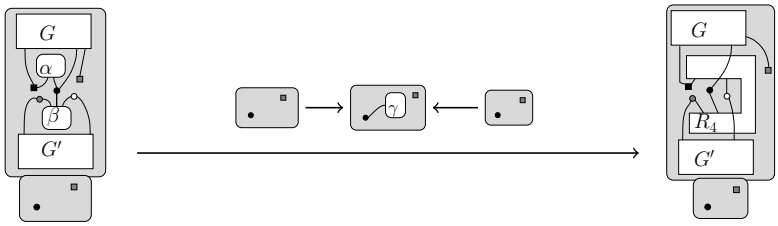
obtain the transition in Figure 9(c).



(a) A first transition for a state $s = J \to G$.



(b) A second transition for a state $s' = J' \to G'$.



(c) The composition of s and s' along $J \leftarrow X \to J'$.

Figure 9: An example of composition of two transitions that use the rule $\alpha/\beta/\gamma$.

The crucial problem of a general composition rule for the system 3L is due to the inherent, "incomplete" information of labels. The transitions that we derive with the Borrowed Context technique only indicate what a state needs from the environment to perform *some* reaction that the state cannot perform on its own; in particular the transition label abstracts away from the complete BC diagram. Roughly, labels indicate what needs to "be around" to make something happen but do not inform about what exactly is happening "inside" the interacting state. For instance, a state can react after "borrowing" some small graph F; however the graph F could be used in several ways – possibly even applying different rules.

Thus, in general, one can neither determine what part of a state is actually reacting, i.e. what partial match has been used to derive the label, nor what rule is used. Thus, it is non-trivial to generalize the communication

rule of CCS to a composition rule for "opposite labels" – simply, because *the* "opposite" of a derived label does not exist. The following example illustrates the problem and also suggests that it is not due to the use of graphs as system models but is rather a consequence of the use of minimal contexts as labels.

Example 4.2 (Failure of Composition)**.** *Consider the following microscopic (process) calculus. The terms are given by*

$$P ::= ?a \mid !a \mid \flat a \mid \natural a \mid 0 \mid P \parallel P$$

where a is element of a set of names and we have $\parallel$ as an associative operator. Moreover, we have the following reaction axioms and rules for contextuatlization.

$$?a \parallel !a \to 0 \qquad \flat a \parallel !a \to 0 \qquad ?a \parallel \natural a \to 0$$

$$\frac{P \to Q}{P \parallel R \to Q \parallel R} \qquad \frac{P \to Q}{R \parallel P \to R \parallel Q}$$

With the intuitive idea of minimal contexts as labels, we have labelled transitions $?a \xrightarrow{[\cdot \parallel !a]} 0$ and $!a \xrightarrow{[?a \parallel \cdot]} 0$; they can be composed, leading to the "silent" transition $?a \parallel !a \xrightarrow{[\cdot]} 0$. However, we also have $\flat a \xrightarrow{[\cdot \parallel !a]} 0$ and $\natural a \xrightarrow{[?a \parallel \cdot]} 0$ but not $\flat a \parallel \natural a \xrightarrow{[\cdot]} 0$. Thus, two transitions with labels $[\cdot \parallel !a]$ and $[?a \parallel \cdot]$ are not always composable. This means that composability depends on the states. Nevertheless, there is at most one way to compose "opposite" labels.

In the case of graphs, the use of the Borrow Context technique implies the existence of some "substate" that is "responsible" for an interaction with the environment. Moreover, in the composition of the two labelled transitions in Example 4.1, we can still locate the "responsible" substates of the constituents, namely the edges with labels α and β. Finally, we see that the new "responsible" substate is the union of the old ones. Thus, the interaction of the composed state is not only based on the same rule, but it actually reuses the same parts of the original states that triggered the transition.

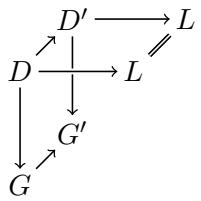
4.2 Composition results for Borrowed Context diagrams

We now formalize the idea that labelled transitions of "composed" states can be "restricted" to interactions of their "constituents". We start by defining

superstates of states (which in turn will be substates) and formally describe how transitions of substates can be extended to superstates.

Definition 15 (Superstate and homogeneous transitions)**.** *Let $s = J \to G$ and $s' = J' \to G'$ be two states. Now s' is a* superstate *of s and s is a* substate *of s' if $s' = C[s]$ for some monotone context $C = J \to F \leftarrow J'$ (see Definition 12).*

Let $\rho = L \leftarrow I \to R$ be a rule, let $J' \to G'$ be a superstate of $J \to G$ and let $t = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ and $t' = (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H')$ be two transitions that are derived with respective partial matches D and D'. Now, t' is homogeneous *with respect to t and t* extends to t' (relative to D and D') *if $D \to D'$.*



The extension of a transition yields the illustrated diagram (as part of the complete BC diagrams).

Now we are ready to formulate our composition results: in Proposition 1, we describe how two transitions that are derived using partial matches into a common rule can be composed such that the original states have a "minimal" overlap in the composed state, namely the intersection of the partial matches; further, Theorem 3 gives sufficient conditions for the composition of two transitions such that the resulting transition is homogeneous w.r.t. to both of them (relative to their partial matches).

Composition with minimal overlap Homogeneity expresses the fact that the rule is applied in the superstate exactly where we expect it to be, i.e. its partial match "reuses" the elements that were already in the partial match in the original graph. In the opposite direction, for any transition $\overline{t}$ from a state $\overline{s} = \overline{J} \to \overline{G}$ that is derived using a partial match $\overline{D}$, any substate $s = J \to G$ of $\overline{s}$ can evolve in "the same" way by "restricting" $\overline{t}$ to a transition t from s such that $\overline{t}$ is homogeneous w.r.t. t: we simply take the intersection of G and $\overline{D}$ as partial match for t. In general, s might miss some parts that were in $\overline{s}$ to evolve and thus the superstate $\overline{s}$ will need to borrow less from the environment.

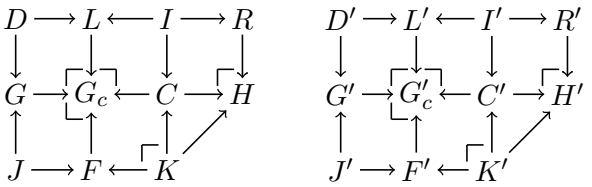
Now consider states s and s' with respective transitions t and t' that are derived with respective partial matches D and D' into a common rule $\rho = L \leftarrow I \to R$. Suppose we want to extend t and t' to a composed transition $\overline{t}$ such that it needs to borrow as little as possible from the

environment (relative to D and D'). In fact, the solution uses the union of D and D' as partial match and yields a minimal overlap of s and s' in the composed state $\overline{s}$.

Proposition 1 (Composition with minimal overlap)**.** *Let $s = J \to G$ and $s' = J' \to G'$ be states, let $\rho = L \leftarrow I \to R$ be a rule, and let $t = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ and $t' = (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H')$ be two transitions that are derived using respective partial matches D and D' into L.*

Then there exists a transition $\overline{t} = (\overline{J} \to \overline{G}) \xrightarrow{\overline{J} \to \overline{F} \leftarrow \overline{K}} (\overline{K} \to \overline{H})$ that is homogeneous w.r.t. both t and t' (relative to D and D') where $G \to \overline{G} \leftarrow G'$ is the pushout of $G \leftarrow (D \cap D') \to G'$ and $\overline{J} = J \cup J'$.

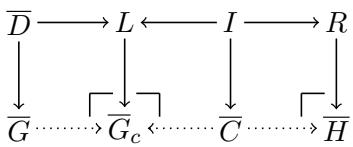
Proof: Let d and d' be the DPOBC-diagrams yielding t and t' using rule ρ, with partial matches D and D' where we use the usual names for objects in DPOBC diagrams (as in Definition 7).

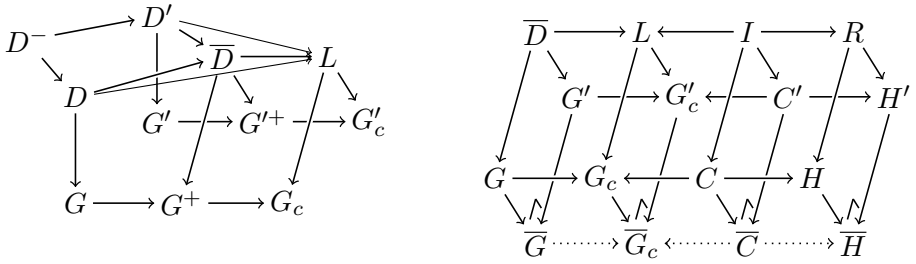


We shall build a DPOBC-diagrams $\overline{d}$ using rule ρ and yielding a transition $\overline{t}$ where $\overline{J} \to \overline{G}$ is as in the statement of the proposition. The outline of the proof is as follows: we start by constructing the graph $\overline{G}$ and the first row of the BC diagram $\overline{d}$, then we build the interface $\overline{J}$, and finally we show that there exists a pushout complement for $\overline{J} \to \overline{G} \to \overline{G}_c$.

Let D^- be the pullback of $D \to L \leftarrow D'$ and let $\overline{D}$ be the union of D and D' over L, i.e. $D \to \overline{D} \leftarrow D'$ is the pushout of $D \leftarrow D^- \to D'$; now we use $\overline{D}$ to split the upper left pushout squares of d and d' as shown in Figure 10(a).

Next, take all pushouts and mediating morphisms as shown in Figure 10(b). It is straightforward to verify that all faces are pushouts: by composition of pushout squares, the vertical "diagonal" squares are pushouts. We have thus constructed the first row of $\overline{d}$.



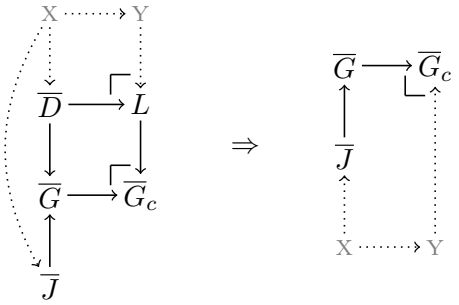


(a) (b)

Figure 10

As $\overline{J}$, let us take the union of J and J' in $\overline{G}$. It is clear that $\overline{J} \to \overline{G}$ is a superstate of $J \to G$ and $\overline{J}' \to \overline{G}'$.

Let us show now that a pushout complement for $\overline{J} \to \overline{G} \to \overline{G}_c$ exists. It is sufficient to show that there exist graphs X and Y such that Y is a pushout complement of $X \to \overline{D} \to L$ and $X \to \overline{J}$, as shown in the diagram below.



In d and d' we already have pushout complements for D and D' in L as shown in Figure 11(a). By splitting the back pushout squares through $\overline{D}$ and constructing the pullbacks in Figures 11(b) and 11(c), we obtain a pushout square where Y is the pushout complement of $X \to \overline{D} \to L$. It is now sufficient to show that $X \to \overline{J}$, which is a consequence of the following reasoning in the distributive lattice of subgraphs.

$$X = X \cap \overline{D} = X \cap (D \cup D') = (X \cap D) \cup (X \cap D') \subseteq E \cup E' \subseteq J \cup J' = \overline{J} \quad (3)$$

□ This proposition already generalizes the preliminary results that have been described in [9]. However, it is still too specialized to obtain

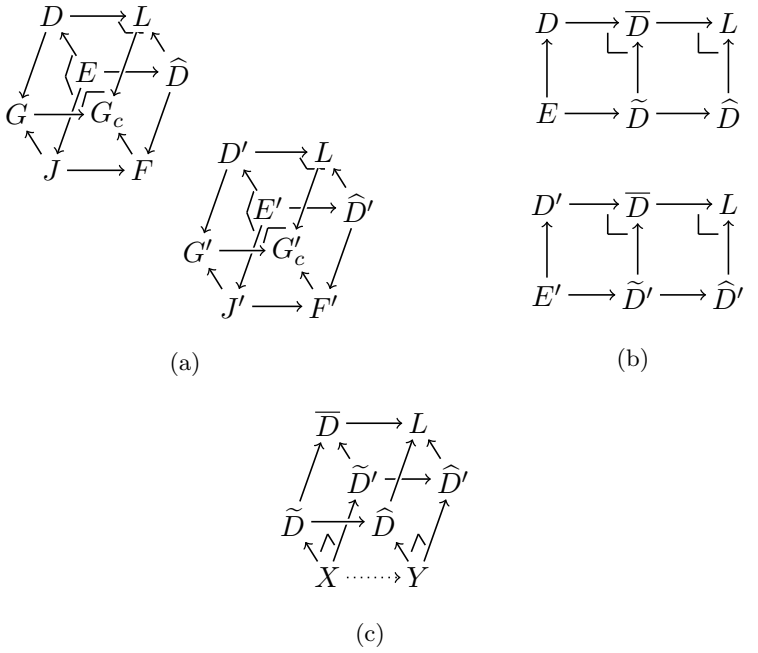


Figure 11

a meaningful counterpart to the communication rule of CCS; nevertheless, it can be "re-used" to obtain the main theorem about the composition of DPOBC-diagrams.

Composition with arbitrary overlap The composition for states that is used in Proposition 1 uses as default a minimal overlap of the substates in the composed super-state; in particular, their overlap is part of the left hand side of the rule that is used to derive the involved transitions. In general, in the composition of two states, we might want to have a bigger overlap than strictly necessary. To illustrate this point, the processes $\overline{a}.0 \parallel P$ and $a.0 \parallel Q$, can communicate. It suffices that they communicate over the name a; however, in the parallel composition $\overline{a}.0 \parallel P \parallel a.0 \parallel Q$, the process P and Q share all free names and not only the name a. Thus, while the minimal overlap is just the channel name a, we in fact want also to share *all* common free names of P and Q in $\overline{a}.0 \parallel P \parallel a.0 \parallel Q$.

Thus, we want to extend Proposition 1 to "arbitrary" overlaps of two given states $J \to G$ and $J' \to G'$; more detailed, we start with two DPOBC-

diagrams d and d' and an "admissible" overlap $G \leftarrow X \rightarrow G'$. The idea of an "admissible" is simple: the graph X should just be an extension of the minimal overlap of G and G' that is necessary for communication.

As proof technique, we want to reuse Proposition 1 by "artificially" enlarging the rule that we use to obtain the desired composition to a suitable *extended rule*.

Definition 16 (Extended rule)**.** *For any rule $\rho = L \leftarrow I \rightarrow R$ and any supergraph $L \rightarrow L^+$ of L, the L^+-*instance *of ρ, denoted $\rho(L^+)$, is the lower-span of the following double-pushout diagram (if it exists).*

$$\begin{array}{ccccc} L & \longleftarrow & I & \longrightarrow & R \\ \downarrow & \urcorner & \downarrow & \ulcorner & \downarrow \\ L^+ & \longleftarrow & I^+ & \longrightarrow & R^+ \end{array}$$

Now, the main idea of the theorem is to extend the rule that is used in the composition just enough to obtain the "desired" overlap X as the minimal overlap of Proposition 1.

Theorem 3 (Composition of transitions)**.** *Let $s = J \rightarrow G$ and $s' = J' \rightarrow G'$ be states; moreover let $t = (J \rightarrow G) \xrightarrow{J \rightarrow F \leftarrow K} (K \rightarrow H)$ and $t' = (J' \rightarrow G') \xrightarrow{J' \rightarrow F' \leftarrow K'} (K' \rightarrow H')$ be two transition using respective partial matches D and D' into a common rule $\rho = L \leftarrow I \rightarrow R$. Let X be a common subgraph of G and G' and let $\overline{G} = G +_X G'$ be the pushout of G and G' over X.*

Now, there exists a transition $\overline{t} = (\overline{J} \rightarrow \overline{G}) \xrightarrow{\overline{J} \rightarrow \overline{F} \leftarrow \overline{K}} (\overline{K} \rightarrow \overline{H})$ that is homogeneous with respect to both t and t' (where $\overline{J}$ is the union of J and J') if there exists E such that (the black part of) the diagram in Figure 12 consists of three pullback squares.

Proof: We start by building the pushout of $L \leftarrow E \rightarrow X$, and call $\widetilde{L}$ the pushout graph. Because EXG_cL (resp. EXG'_cL) is a composition of two pullbacks, it is itself a pullback square. Therefore, there is a unique monomorphism $\widetilde{L} \rightarrow G_c$ (resp. $\widetilde{L} \rightarrow G'_c$) making the diagram with the square commute, in other words, $\widetilde{L}$ splits $L \rightarrow G_c$ (resp. $L \rightarrow G'_c$) into two monomorphisms. By classical pushout splitting, we can split the square $EL\tilde{L}X$ into two pushouts. Since the square $EDGX$ is a pullback, there is a unique monomorphism $D^+ \rightarrow G$ such that the diagram in Figure 13(a) commutes. Because of pushout decomposition properties, the square $D^+\tilde{L}G_cG$ is a pushout.

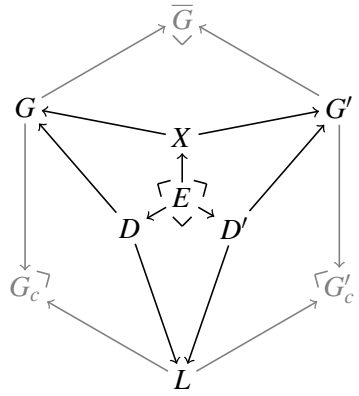
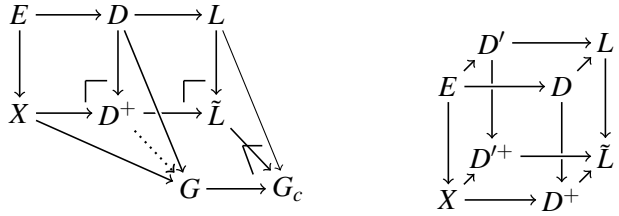


Figure 12: The condition for ρ-composability (in black) and some pushouts from the proof (in grey)



(a) (b)

Figure 13

A symmetric construction yields a match D'^+ for G' and $\tilde{L}$. Since $EDLD'$ is a pullback, and all vertical squares of the cube of Figure 13(b) are pushouts, the lower square is a pullback.

We can now construct the diagrams $d(\widetilde{L})$ and $d'(\widetilde{L})$ by the construction mentioned after Definition 16, with D^+ and D'^+ as matches. We compose them by Proposition 1 and obtain the DPOBC-diagram $\widetilde{d}$ shown in Figure 14(a).

By pushout complement splitting of the upper-left square of the construction of $d(\widetilde{L})$ into two pushouts, as shown in Figure 14(b), one obtains a match for the completion of $\widetilde{d}$ into a DPOBC-diagram with rule ρ. Thanks to the uniqueness properties of pushouts and pullbacks, it is easy to show that the same match would be constructed from d'.

It is now left to show that $\overline{G} \cong \widetilde{G}$. In fact, we not only want to

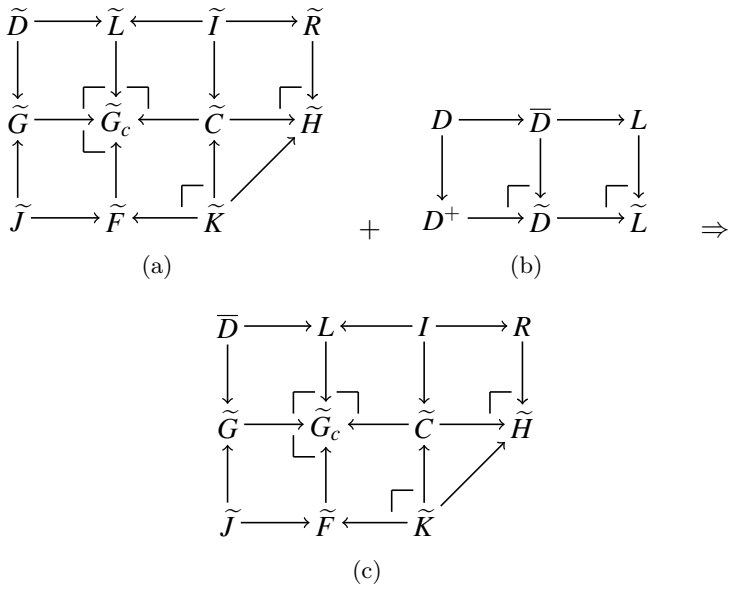


Figure 14: Recomposition of a DPOBC-diagram with the correct rule.

prove that there exists a homogeneous transition, but also that there is one involving the given graph $\overline{G}$.

Figure 15(a), where all "faces" are pushout squares, shows how $\widetilde{G}$ is constructed, as well as some other parts of the composition diagram. One part of it, namely the left face of the cube, the top square and the two diamond-shaped faces are shown flattened out in Figure 15(b).

By uniqueness of pushouts, the outer square of Figure 15(b) being one, we have $\overline{G} \cong \widetilde{G}$. □

The intuition behind the condition of Theorem 3 is as follows. Whatever G and G' had in common with L, namely the partial matches D and D' respectively, are glued in $\overline{G}$ as they should be in L, i.e. by identifying the same elements, namely E. Technically, the condition ensures that the pullback of $D \to \overline{G} \leftarrow D'$ is the same as the one of $D \to L \leftarrow D'$. For instance, if D and D' contain both more than half the rule L, the gluing in $\overline{G}$ has to identify everything they have "in common" to form L, and not something "bigger". Of course, if this gluing is bigger than L (if not enough is identified), it still makes $\overline{G}$ able to evolve using this rule. But then, it is not clear how to obtain the resulting graph $\overline{H}$ from the two resulting graphs H and H', so we will not understand it as composition of transitions.

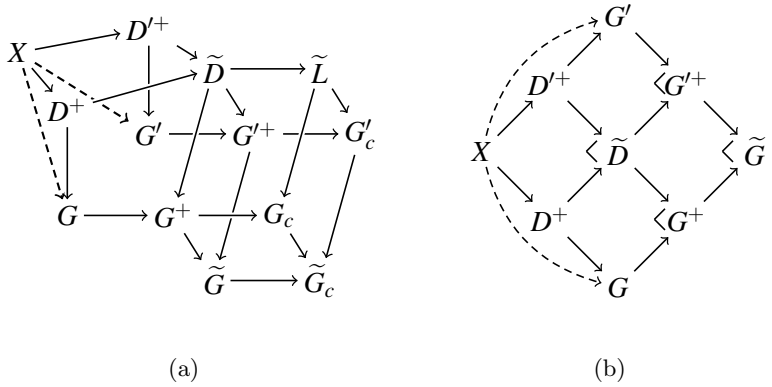


Figure 15

We will make use of the condition of the theorem in the next section to succinctly describe the premises of the counterpart to the communication rule of CCS.

Definition 17. *With the assumptions of Theorem 3, the transitions t and t' are* ρ-composable *if there exists E such that (the black part of) the diagram in Figure 12 consists of three pullback squares; the transition $\bar{t}$ of the theorem is the* composition through $\overline{G}$ *of t and t' (relative to D and D').*

Remark 1 (The "Meaning" of Compositionality)**.** *The word* compositionality *is used differently in different contexts. We understand compositionality as the principle that the semantics of the whole is determined by the semantics of the parts, where the (possibly composed) entities are states with interfaces. To illustrate the basic idea, take all transitions of a state as semantics; in this situation, using Theorem 3, we have that this "all transitions" semantics is compositional: each transition of the composed state can be (re-)constructed from the transitions of its components. It is an open problem how this approach to compositionality can be incorporated into the work presented in [23] where compositionality is considered on the level of rules and rewriting diagrams.*

In fact, we shall only make use of the special case of Theorem 3 where X is actually a common sub-interface of the states $J \to G$ and $J' \to G'$, i.e. X is a common subgraph of J and J'. This is in analogy to process calculi where it is crucial that parallel composition lets the composed processes only share some of their free names.

5 SOS semantics

We now make use of the composition result of Theorem 3 in two ways: first, as already discussed at length, we obtain a counterpart of CCS-style communication for graph rewriting; second, we can dispense with a number of "superfluous axioms" for basic actions.

Looking at CCS again, we can notice a difference with 3L in the definition of axioms. In fact, in 3L, the application of the rewriting rules themselves appear as basic actions, which is not the case in CCS where we cannot have both actions a and $\overline{a}$ in a single label. The communication rule covers the case where both are performed complementary, which corresponds exactly to the application of a reduction rule. Conversely, in 3L, any match for a left-hand side of a rule yields a basic action, in particular the empty graph. Now that we have an equivalent of the CCS-composition, we can remove from the set of basic actions certain "superfluous" ones, which can then be reconstructed by addition of a composition rule.

As "good" matches for a rule in graphs we shall take the irreducible elements in the lattice of subgraphs of the left hand side of the rule. Using irreducible graphs, we define *atomic* actions, which use irreducible graphs as partial matches. However, we shall show that we can obtain all basic actions by applying the composition rule to the atomic ones. This allows us to define a system that is exactly as expressive as 3L (and hence DPOBC) with one extra rule, but a smaller number of axioms.

Definition 18 (Irreducible graph)**.** *Let G be a hypergraph. A* (non-trivial) decomposition *of G is a pair of inclusions $A \to G \leftarrow B$ such that G is the union of A and B and $G \neq A$ or $G \neq B$. A hypergraph G is* irreducible *if it has no non-trivial decomposition.*

Thus, an irreducible graph cannot be decomposed into strictly smaller graphs. Clearly, a single node is an irreducible graph, but a graph composed of two single nodes is not. This formal definition fits the intuition of "atomic" hypergraphs.

Fact 4. *The only irreducible hypergraphs are the single vertex graph and all graphs that consist of a single hyperedge that is incident to every node.*

We now have all concepts to inductively define a transition system for graph rewriting in analogy to CCS. The system is summarized in Table 2. We call the system SOSBC, since it is an SOS-like definition for borrowed contexts (as we shall show below).

Formally, the first rule describes a family of *Atomic Actions*, which are the basic actions in which the partial match is an irreducible graph. The second and third rule are taken from 3L and have already been discussed in Section 3. Finally, the last rule is the one justified by the composition theorem, i.e. Theorem 3.

We now show that this system is exactly as expressive as 3L. We have already seen that it is included in it. Indeed, every transition derivable in SOSBC is also in 3L (trivially for the first three rules, and by Section 4 for the last one). It then suffices to show that every transition derivable in 3L is also in SOSBC. It is trivially true for interface narrowing and compatible contextualization. It is left to show that every basic action can be derived in SOSBC.

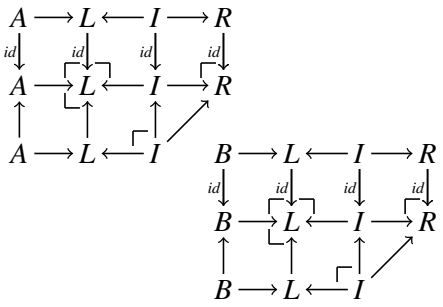
Lemma 2 (Basic action decomposition)**.** *Let $\rho = L \leftarrow I \rightarrow R$ be a rule and D a partial match for L. Let $A \rightarrow D \leftarrow B$ be a decomposition of D. Then $A \rightarrow A$ and $B \rightarrow B$ are ρ-composable through D and the result of the composition is exactly the axiom transition $(D \rightarrow D) \xrightarrow{D \rightarrow L \leftarrow I} (I \rightarrow R)$.*

Proof: We first show that the condition of Theorem 3 holds. As $A \rightarrow D \leftarrow B$ is a decomposition of D, the morphisms D is the union of A and B. Thus, let $A \leftarrow O \rightarrow B$ be the pullback of $A \rightarrow D \leftarrow B$ to obtain a pushout square. Since D is a partial match for L, then so are A and B, and therefore their corresponding basic action transitions exist. They are justified by the DPOBC-diagrams in Figure 16(a), which lead to the construction of the diagram 16(b).

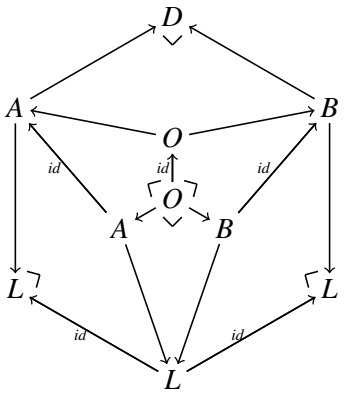
After applying the construction of Theorem 3, it is easy to check that the resulting composition of the two transitions is the transition $(D \rightarrow D) \xrightarrow{D \rightarrow L \leftarrow I} (I \rightarrow R)$. □

Theorem 5 (Basic Completeness)**.** *Any basic action of the* 3L*-system can be obtained from atomic actions and the composition rule of the* SOSBC*-system.*

Proof: Let $t = (D \rightarrow D) \xrightarrow{D \rightarrow L \leftarrow I} (I \rightarrow R)$ be a basic action and l be the size of D, i.e. the number of hyperedges. By applying the decomposition lemma (Lemma 2) repeatedly until all the subgraphs of D are decomposed into irreducible graphs, one obtains a binary tree whose root is t and nodes are basic actions. Since l is finite, and the decomposition lemma decreases the size of graphs, this process is finite, and the leaves of the tree are atomic actions. □



(a)



(b)

Figure 16

Theorem 6 (Soundness and completeness)**.** *Let $\mathcal{S}$ be a graph transformation system. Then there is a* BC*-transition*

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

if and only if it is derivable in the SOSBC*-system.*

Proof: Atomic actions, interface narrowing and contextualization are part of the 3L-system, therefore they are derivable as BC-transitions. By construction, the composition yields derivable transitions in BC too.

Conversely, every BC-transition is derivable in 3L, and by Theorem 5, is derivable in SOSBC. □

- **Atomic Actions**

$$\frac{}{(D \to D) \xrightarrow{D \to L \leftarrow I} (I \to R)} \quad \text{where } (L \leftarrow I \to R) \in \mathcal{S} \text{ and } D \text{ irreducible with } D \to L$$

- **Interface Narrowing**

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{(J' \to G) \xrightarrow{J' \to F' \leftarrow K'} (K' \to H)} \quad \text{where } C = J \to J \leftarrow J' \text{ and } J' \to F' \leftarrow K' = C[J \to F \leftarrow K]$$

- **Compatible Contextualization**

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{C[J \to G] \xrightarrow{C[J \to F \leftarrow K]} \overline{C}[K \to H]} \quad \text{where } C = J \to E \leftarrow \overline{J} \text{ compatible with } J \to F \leftarrow K \text{ and } \overline{C} = (J \to F \leftarrow K)[C]$$

- **Composition**

$$\frac{t = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H) \qquad t' = (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H') \qquad J \leftarrow X \to J'}{\bar{t} = ((J +_X J') \to (G +_X G')) \xrightarrow{(J +_X J') \to \overline{F} \leftarrow \overline{K}} (\overline{K} \to \overline{H})}$$

where t and $t'$are ρ-composable and $\bar{t}$ is their composition through $G +_X G'$

Table 2: Axioms and rules of the SOSBC-system.

In fact, every derivation tree obtained by the process that is described in the theorem will have the same structure: a certain number n of Atomic Actions, followed by $n-1$ applications of the composition rule. Then a Compatible Contextualization and an Interface Narrowing, as in the case of the 3L system. Consider the following example of a derivation tree.

Example 5.1 (Composition of transitions)**.** *As example, consider the derivation tree of* SOSBC *for the final transition of Example 4.1, which is shown in Figure 17.*

6 Conclusion

We have made a first step towards a proper SOS semantics for graph transformation systems, by introducing SOS rules not only for a specific graph transformation system, but by presenting a method for synthesizing such rules for arbitrary systems. This will hopefully lead to a better understanding of the nature of SOS semantics in general.

While earlier work on automatic derivation of labelled transition systems, pioneered by [18], focused on the derivation of labelled transitions such that the resulting bisimilarity is a congruence, we here concentrated on synthesizing SOS rules, obtaining a compositional operational semantics.

Composition rules for process calculi, such as CCS or the π-calculus, where at most two processes interact in a well-defined manner, can usually be stated quite concisely, whereas our composition rule is surprisingly complex. In future work we want to investigate under which conditions the label of the composed step can be determined from the labels of the interacting graphs by a simpler procedure. We regard this work as a first step towards a SOS semantics for graph transformation, however in order to obtain a simpler and more straightforward presentation it might be necessary to impose certain constraints on the rules, such as restrictions on the left-hand or right-hand side.

A natural question to ask is how the SOS semantics for a specific calculus or graph transformation system would look like, for instance for the encoding of CSS into graph transformation studied in [2]. One of the synthesized rules would certainly correspond to the CCS communication rule, which states that $P \xrightarrow{a} P'$ and $Q \xrightarrow{\overline{a}} Q'$ imply $P \mid Q \xrightarrow{\tau} P' \mid Q'$. On the other hand we would also generate many other rules, for instance a rule where P in addition borrows the output prefix of Q and Q borrows the input prefix of P. Our SOS rules allow the composition of such labelled transitions, since we accept

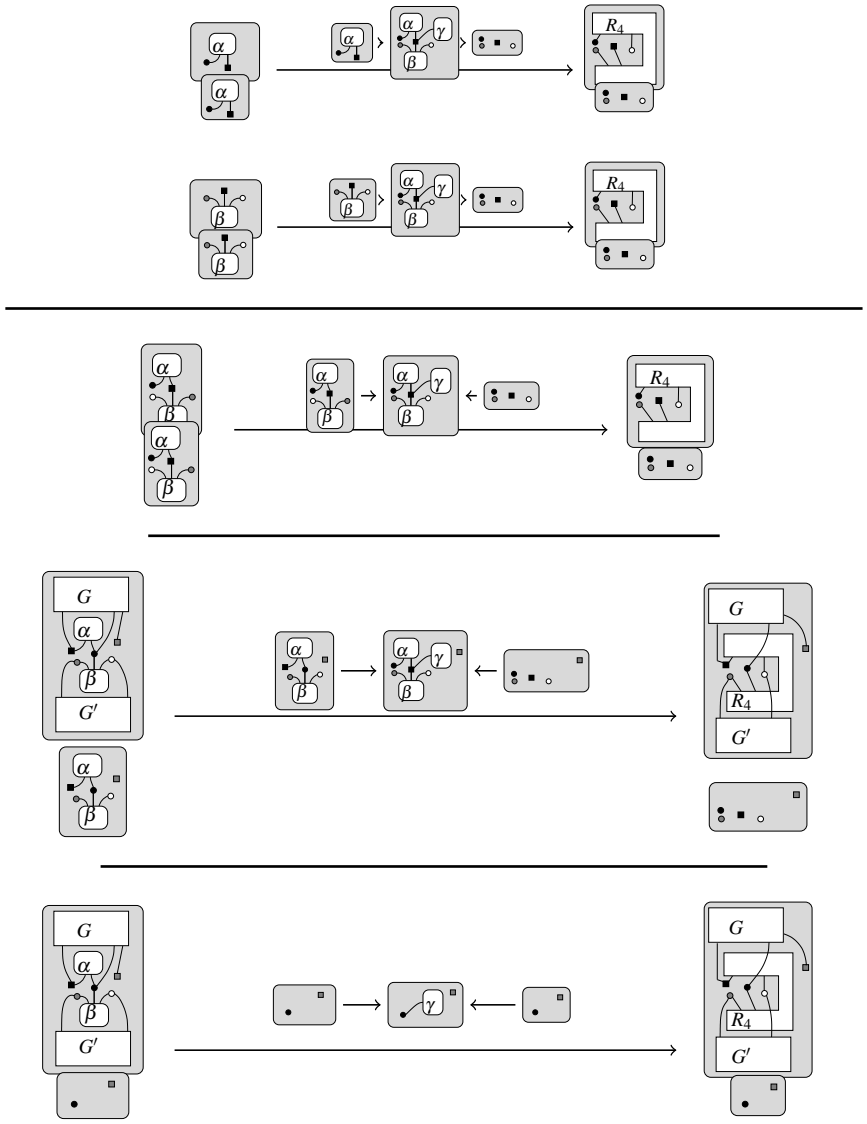


Figure 17: An example of a derivation in the SOSBC-system.

arbitrary overlaps of the left-hand side and the two graphs to be composed. Hence the full SOS semantics would be somewhat large and unwieldy and difficult to represent. Pruning the synthesized rules to a minimal set is a direction of future work.

Our focus is quite different than that of work on process calculi defining syntactic conditions on the rules in such a way that the resulting bisimilarity is a congruence (compare with the de Simone format [8] and the `tyft/tyxt`-format [14]). Instead, bisimilarity on labelled transition systems derived via the Borrowed Context technique is automatically a congruence [10]; thus we work in the "opposite" direction and synthesize the SOS rules, which is in contrast to work on rule formats where the SOS rules are given in advance.

Earlier work has established graph transformation as a formalism into which many process calculi can be encoded or "compiled". This becomes apparent in Milner's work on bigraphs [19, 20] and related work (e.g. [13]); also there are several approaches of graphical syntax for processes, such as [5, 11] or the well-known interaction nets [17]. Even closer to our work are papers that discuss SOS semantics for graphs such as [7, 15]. However, in these two papers, SOS rules are given a priori, similar to the work on rule formats cited above, while they are synthesized in the present paper; a further development of the ideas of [15] is presented in [16], which in turn has inspired the work described in [4] where so-called SOS rewriting is performed on term representations of graphs.

A predecessor paper of the present one is [1], which addresses the composition and decomposition of derivations without the emphasis on SOS rules and instead focuses on the combination of entire Borrowed Context diagrams. In [23] compositionality for graph transformation is obtained by decomposing rules into subrules, a view that is somehow dual to ours: we do not decompose rules, but the graph to be rewritten. Finally [6] shows how label derivation can be addressed in double categories and explores connections to the tile model [12]. There are similarities to our approach, however [6] works in a different categorical setting and does not investigate composition of LTS steps.

This diversity of work on compositionality in semantics of interactive systems reflects the richness of the subject. Attracted by the simplicity of structural operational semantics, we have developed basic results for a simplified account of the Borrowed Context technique. Even though we have missed the mark of a proper SOS format since composition of labels depends on the involved states, we are convinced that our results are the core of a fully fledged structural operational semantics for graph transformation.

Acknowledgements

We would like to thank Paolo Baldan and Filippo Bonchi for many interesting discussions on this topic.

References

[1] P. Baldan, H. Ehrig, and B. König. Composition and decomposition of DPO transformations with borrowed context. In *Proceedings of ICGT '06*, volume 4178 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2006.

[2] F. Bonchi, F. Gadducci, and B. König. Synthesising CCS bisimulation using graph rewriting. *Information and Computation*, 207(1):14–40, 2009.

[3] F. Bonchi, F. Gadducci, and G. V. Monreale. Labelled Transitions for Mobile Ambients (as Synthesized via a Graphical Encoding). *Electronic Notes in Theoretical Computer Science*, 242(1):73–98, 2009.

[4] R. Bruni and E. Tuosto A. Lluch Lafuente, U. Montanari. Style-based architectural reconfigurations. *Bulletin of the EATCS*, 94:161–180, 2008.

[5] R. Bruni, F. Gadducci, and A. Lluch Lafuente. A graph syntax for processes and services. *Web Services and Formal Methods*, pages 46–60, 2010.

[6] R. Bruni, F. Gadducci, U. Montanari, and P. Sobociński. Deriving weak bisimulation congruences from reduction systems. In *Proceedings of CONCUR '05*, volume 3653 of *Lecture Notes in Computer Science*, pages 293–307. Springer, 2005.

[7] A. Corradini, R. Heckel, and U. Montanari. Graphical operational semantics. In *Proceedings of GT-VMT '00*, pages 411–418. Carleton Scientific, Ottawa, 2000.

[8] R. de Simone. Higher level synchronizing devices in MEIJE-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[9] A. Dorman and T. Heindel. Structured operational semantics for graph rewriting. In Silva et al. [26], pages 37–51.

[10] H. Ehrig and B. König. Deriving Bisimulation Congruences in the DPO Approach to Graph Rewriting with Borrowed Contexts. *Mathematical Structures in Computer Science*, 16(6):1133–1163, 2006.

[11] F. Gadducci. Graph rewriting for the pi-calculus. *Mathematical Structures in Computer Science*, 17(3):407–437, 2007.

[12] F. Gadducci and U. Montanari. The tile model. In Plotkin et al. [21], pages 133–166.

[13] D. Grohmann and M. Miculan. Graph algebras for bigraphs. *Electronic Communications of the EASST*, 29, 2010.

[14] J.F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100:202–260, 1992.

[15] D. Hirsch and U. Montanari. Synchronized hyperedge replacement with name mobility (a graphical calculus for mobile systems). In *Proceedings of CONCUR '01*, volume 2154 of *Lecture Notes in Computer Science*, pages 121–136. Springer, 2001.

[16] D. Hirsch and U. Montanari. Shaped hierarchical architectural design. *Electronic Notes in Theoretical Computer Science*, 109:97–109, 2004.

[17] Y. Lafont. Interaction nets. In *Proceedings of POPL '90*, pages 95–108, New York, NY, USA, 1990. ACM.

[18] J. J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In *Proceedings of CONCUR '00*, volume 1877 of *Lecture Notes in Computer Science*, pages 243–258, 2000.

[19] R. Milner. Bigraphical reactive systems. In *Proceedings of CONCUR '01*, volume 2154 of *Lecture Notes in Computer Science*, pages 16–35. Springer, 2001.

[20] R. Milner. Pure bigraphs: Structure and dynamics. *Information and Computation*, 204(1):60–122, 2006.

[21] G. D. Plotkin, C. Stirling, and M. Tofte, editors. *Proof, Language, and Interaction, Essays in Honour of Robin Milner*. The MIT Press, 2000.

[22] J. Rathke and P. Sobociński. Deriving structural labelled transitions for mobile ambients. *Information and Computation*, 208:1221–1242, 2010.

[23] A. Rensink. Compositionality in graph transformation. In *Proceedings of ICALP '10*, volume 6199 of *Lecture Notes in Computer Science*, pages 309–320. Springer, 2010.

[24] V. Sassone and P. Sobociński. Deriving bisimulation congruences using 2-categories. *Nordic Journal of Computing*, 10(2):163–183, 2003.

[25] V. Sassone and P. Sobociński. A Congruence for Petri Nets. *Electronic Notes in Theoretical Computer Science*, 127(2):107–120, 2005.

[26] A. Silva, S. Bliudze, R. Bruni, and M. Carbone, editors. *Proceedings Fourth Interaction and Concurrency Experience*, volume 59 of *Electronic Proceedings in Theoretical Computer Science*, 2011.

[27] A. Simpson. Sequent Calculi for Process Verification: Hennessy-Milner Logic for an Arbitrary GSOS. *Journal of Logic and Algebraic Programming*, 60–61:287 – 322, 2004.