Parallel High-Level Replacement Systems

Gabriele Taentzer

Computer Science Department, Technical University of Berlin,
Franklinstr. 28/29, Sekr. FR 6-1, D-10587 Berlin
gabi@cs.tu-berlin.de

Abstract

High-level replacement systems are an axiomatic categorical framework based on double-pushouts in order to unify replacement systems like grammars for different kinds of graphs and relational structures or other types of structures like algebraic specifications. Parallel high-level replacement systems are introduced to formalize parallel rewriting of these high-level structures. On one hand this concept generalizes and extends parallel graph grammars presented so far in the algebraic approach by allowing other structures than graphs, on the other hand the kinds of replacement introduced for high-level replacement systems are extended by different types of parallel replacement which are compared to each other in different parallel replacement theorems.

An abstract version of a window-based graph editor and movement of objects in configuration spaces are presented as examples of parallel high-level replacement systems.

Contents

1 Introduction

Parallel graph grammars as described by Ehrig and Kreowski in [11] introduce the concept of parallel replacement of graphs in an algebraic framework. Other approaches to parallel replacement of graphs are given by Nakamura and Aizawa [27], Nagl [26], Kreowski [20], Bailey et.al.[3], etc. Parallel graph grammars can be seen as an extension of sequential graph grammars (see Ehrig [8]) on one hand, on the other hand parallel graph grammars generalize Lindenmayer–systems. L–systems describe a kind of context–free parallel replacement on strings and were introduced to model the development of organisms. A large survey over different kinds of L–systems is given by Rozenberg and Salomaa in [36] and [35].

Until now, the concept of parallel rewriting is introduced only for the basic type of algebraic graph grammars which uses directed graphs with colored nodes and edges and color–preserving graph morphisms. In order to extend the field of applications for parallel rewriting the concept of parallel graph grammars is generalized to parallel high–level replacement systems (PHLR-systems). This concept allows other structures than this kind of graphs like graphs with partially ordered color alphabets, relational structures, hypergraphs, different kinds of Petri nets, algebraic specifications, etc. For example, in [38] Schween and Taentzer defined parallel replacement for graphs with a particular node color set which is partially ordered to model the movement of objects in configuration spaces.

High–level replacement systems (HLR-systems) generalize the algebraic approach to graph grammars to other types of replacement systems. A lot of properties like Church–Rosser–Properties, Parallelism Theorem and Concurrency Theorem, as they are known for the graph case are shown for HLR–systems by Ehrig et al. in [10].

In this paper different kinds of parallel rewriting are introduced for HLR–systems. Before we start to describe the principles of parallel rewriting formally a short introduction to derivation concepts in high–level replacement systems is given in figure 1. The arrows between different types of derivations are concerned to the generative power of high-level replacement systems defined on these types.

The basic kind of rewriting is the *direct derivation* where one production is applied to an object. It is introduced in various approaches to graph grammars (by Ehrig [8], Löwe [22], Janssens and Rozenberg [25], Nagl [26],etc.) and in high–level replacement systems ([10]).

Direct derivations via productions which are independent of each other can be combined to *derivations with parallel productions*. The parallel production combines several productions to a new one. Applying it in the usual sequential way the result shows the effect of applying all corresponding productions in parallel. This type is also formulated for graphs and high–level replacement systems (in [8], [22] and [10]).

In [11] a basic version of parallel rewriting is introduced where the whole graph has to be covered by occurrences of left hand sides of productions. Interfaces between the different occurrences have to remain constant during parallel replacement. This type of parallel rewriting is formulated as a special kind of *star-parallel derivations* and is generalized in the framework of HLR–systems.

If the interfaces between different occurrences of productions do not need to be constant during the replacement we have general star-parallel derivations. Here all interfaces

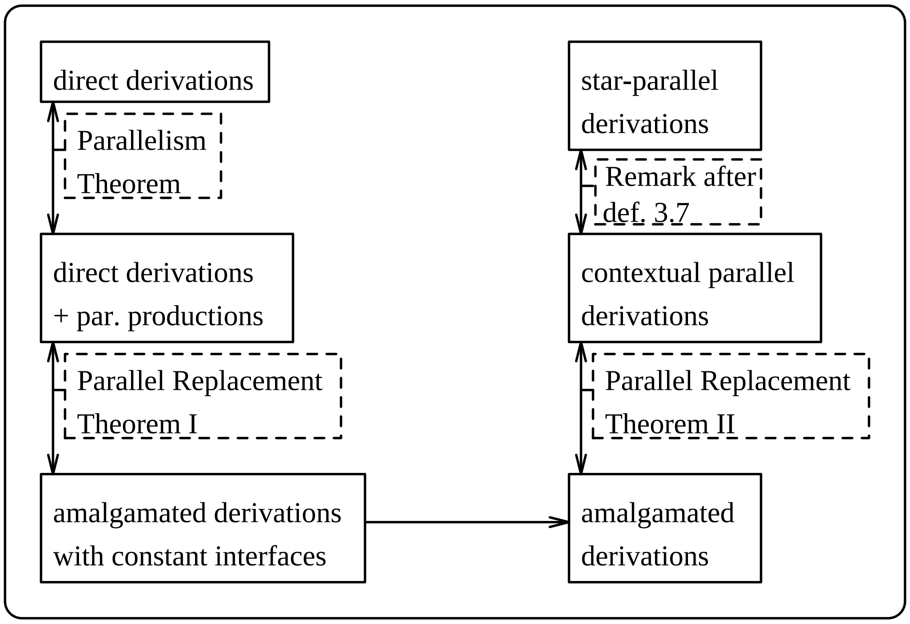


Figure 1: Overview of derivation types in high-level replacement systems

are specified by subproductions which implies that existing interfaces can be dissolved and new ones established.

In most cases only a part of an object should be replaced. There, the context has to be determined and joined unchanged to the object derived in parallel. This is done in a special kind of star-parallel derivation, the so-called *contextual parallel derivations*. This kind of derivation corresponds to the type of *amalgamated derivations* where first a new (amalgamated) production is generated from a set of elementary productions and then this production is applied to the given object by a direct derivation. The very close relationship between contextual parallel derivations and amalgamated derivations is shown in the Parallel Replacement Theorem II.

Amalgamated derivations combine the concepts of parallel and sequential rewriting. On the first level an amalgamated production is constructed doing a complete parallel derivation. On the second level this production is applied in a sequential way. (See also mixed rewriting in [26].) This type of derivation is already examined for special graphs with partial ordered node labels (in [38]). In the following it is also generalize to high–level replacement systems.

Amalgamated derivations with constant interfaces are related to direct derivations with parallel productions which are introduced for HLR–structures in [10]. This relationship is shown in the Parallel Replacement Theorem I. It turns out that overlappings of occurrences which are explicitly given by interfaces in amalgamated derivations are handled implicitly in direct derivations with parallel productions. Especially this result overcomes the gap between the sequential and parallel graph grammars.

These correspondences between different derivation types open the possibility to carry over the theory of sequential rewriting – like Church-Rosser-Properties, analysis and synthesis constructions stated in the Parallelism Theorem and concepts of concurrency, embedding and distribution – to parallel derivations under suitable conditions.

Examples of PHLR-Systems can be given, for example, on the basis of graphs, hypergraphs, relational structures, Petri nets or algebraic specifications. An abstract version of a window-based graph editor and movement of objects in configuration spaces are presented as examples of PHLR-systems. They are based on graphs which are probably the most intuitive and easy to visualize structures. Moreover, the theory of graph grammars set going a theoretical approach to high-level replacement.

An appendix presents all concepts from category theory needed in this paper.

A short version of this paper is presented by Ehrig and Taentzer in [17] while [40] presents PHLR-systems more detailed.

2 Basic concepts of high-level replacement systems

In this section basic concepts of HLR–systems like productions and derivations introduced in [10] are reviewed. They are formulated for an arbitrary category and follow the double-pushout approach. This theory about HLR–systems is extended in [10] by concepts for independence and parallelism. Properties like Church–Rosser and the Parallelism Theorem are shown under suitable conditions. HLR–systems which are formulated in a category that satisfies these conditions are called HLR1–systems.

Let **CAT** be a category where the objects can be regarded as high-level structures and the morphisms as structure preserving functions between high-level structures.

Additionally, we want to distinguish a class M of morphisms which are used in the productions while general morphisms in **CAT** are used to define applications of productions to high–level structures.

2.1 Definition *A* production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ *in* **CAT** *consists of objects L, I and R, called left hand side, interface or gluing object and right hand side respectively, and two morphisms $I \xrightarrow{l} L$ and $I \xrightarrow{r} R$.*

- *p is called* M-production *if both morphisms l and r belong to class M.*
- *A production $p = (I \xleftarrow{id_I} I \xrightarrow{id_I} I)$ of two identities is called* identical production*.*
- *The production $p^{-1} = (R \xleftarrow{r} I \xrightarrow{r} L)$ is called* inverse production *of p.*
- *The production $j = (J \longleftarrow J \longrightarrow J)$ is called* empty *with J being the initial object of* **CAT**.

2.2 Definition *Given a production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ and an object C, called context object, together with a morphism $I \xrightarrow{c} C$ a* direct derivation $S \Longrightarrow_d S'$ *via p, short $S \overset{p}{\Longrightarrow}_d S'$, from an object S to an object S' is given by the following two pushout diagrams (1) and (2) in the category* **CAT***:*

$$
\begin{tikzcd}
L \arrow[d, "o"'] \arrow[dr, phantom, "(1)"] & I \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "c"] \arrow[dr, phantom, "(2)"] & R \arrow[d, "q"] \\
S & C \arrow[l, "s"] \arrow[r, "s'"'] & S'
\end{tikzcd}
$$

Morphisms $L \xrightarrow{o} S$ *and* $R \xrightarrow{q} S'$ *are called* occurrences *of* L *in* S *and* R *in* S'*, respectively. By occurrences of the production* p *in a structure* S *we mean occurrences of the left hand side* L *in* S.

The idea of a direct derivation is that the left hand side L mapped in S is replaced by the right hand side R mapped in S' where the interface I together with its mappings to L, R and to the context C designates corresponding interfaces in L, R and C.

Production p is called *applicable* to object S via occurrence $L \xrightarrow{o} S$ if there is a pushout–complement (def. 7.16) of $I \xrightarrow{l} L$ and $L \xrightarrow{o} S$, called C, together with morphisms $I \xrightarrow{c} C$ and $C \xrightarrow{s} S$ and diagram (2) in the figure above is a pushout.

An HLR–system $HLRS = (S, P, T)$ in a category **CAT** is given by a start object S, a production set P and a class T of objects in **CAT**, called terminal objects. The language $\mathcal{L}(HLRS)$ of such an HLR–system is then defined by the set of all terminal colored objects in **CAT** derivable from S by a sequence of direct derivations via productions of P.

Adding the concept of sequential independence the Parallelism Theorem is formulated under additional conditions, the HLR1–conditions which are presented in [10]. An HLR–system that satisfies these conditions is called HLR1-category.

2.3 Definition

Given two productions $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ *and* $p' = (L' \xleftarrow{l'} I' \xrightarrow{r'} R')$*, a derivation sequence* $S \overset{p}{\Longrightarrow}_d S' \overset{p'}{\Longrightarrow}_d S''$ *given by the following pair of (solidly drawn) double-pushouts is called* sequentially independent*, if there are morphisms* $L' \xrightarrow{d} C$ *and* $R \xrightarrow{e} C'$ *(dotted arrows) such that* $s' \circ d = o'$ *and* $s'' \circ e = q$.

$$\begin{tikzcd}
L \arrow[d] & I \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] & R \arrow[dr, "q"] \arrow[drrr, dotted, "e"] & & L' \arrow[dl, "o'"'] \arrow[dlll, dotted, "d"'] & I' \arrow[l, "l'"'] \arrow[r, "r'"] \arrow[d] & R' \arrow[d] \\
S & C \arrow[l] \arrow[rr, "s'"] & & S' & & C' \arrow[ll, "s''"'] \arrow[r] & S''
\end{tikzcd}$$

2.4 Definition

Given productions $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ *and* $p' = (L' \xleftarrow{l'} I' \xrightarrow{r'} R')$ *the production* $p + p' = (L + L' \xleftarrow{l+l'} I + I' \xrightarrow{r+r'} R + R')$ *is called the* parallel production *of* p *and* p' *where "+" indicates the binary coproduct.*

Morphisms $l + l'$ and $r + r'$ are uniquely determined according to the universal property of coproducts.

A parallel production $p + p'$ is an M-production if p and p' are M-productions, **CAT** has binary coproducts $L + L'$, $I + I'$ and $R + R'$ and M is closed under binary coproducts. (See [10] for details.) The construction of parallel productions can be iterated leading to productions $p1 + p2 + \ldots + pn$ with $n \geq 1$ which are also called parallel productions, written $\sum_1^n p_i = (\sum_1^n L_i \xleftarrow{\sum l} \sum_1^n I_i \xrightarrow{\sum r} \sum_1^n R_i)$.

Direct derivations via parallel productions show the application of several productions in parallel where interfaces are not modeled explicitly but implicitly by overlappings of corresponding occurrences of productions in the given object.

2.5 Parallelism Theorem *Let* **CAT** *be an HLR1-category and p and p' M-productions then we have:*

1. Synthesis*: Given a sequentially independent derivation sequence $S \overset{p}{\Longrightarrow}_d S' \overset{p'}{\Longrightarrow}_d S''$ there is a synthesis construction leading to a derivation $S \overset{p+p'}{\Longrightarrow}_d S''$.*

2. Analysis*: Given a derivation $S \overset{p+p'}{\Longrightarrow}_d S''$ there is an analysis construction leading to two sequentially independent derivation sequences $S \overset{p}{\Longrightarrow}_d S' \overset{p'}{\Longrightarrow}_d S''$ and $S \overset{p'}{\Longrightarrow}_d \bar{S}' \overset{p}{\Longrightarrow}_d S''$.*

3. Bijective Correspondence*: The constructions "synthesis" and "analysis" are inverse to each other in the following sense: Given p, p' and $p + p'$ there is a bijective correspondence (up to isomorphism) between sequentially independent derivation sequences $S \overset{p}{\Longrightarrow}_d S' \overset{p'}{\Longrightarrow}_d S''$ and derivations $S \overset{p+p'}{\Longrightarrow}_d S''$.*

The proof of this theorem is given in [10]. This result is related to amalgamated derivations with constant interfaces later on.

$HLRS+$ denotes an HLR-system $(S, P+, T)$ where the set $P+$ of productions is the closure of P wrt. parallel productions. On this basis we can formulate the following result.

2.6 Corollary *Let* **CAT** *be an HLR1-category and $HLRS$ an HLR-system with M-productions only. $\mathcal{L}(HLRS) = \mathcal{L}(HLRS+)$*

Proof: Direct consequence of the Parallelism Theorem 2.5. □

Furthermore, additional concepts are developed for HLR-systems. Concepts of concurrency are described in [10] while concepts of embedding and distribution are investigated by Padberg and Taentzer in [33] and [29].

3 Parallel high-level replacement systems

In the last section, we had derivations via parallel productions which do not show interfaces explicitly. Now we want to study parallel rewriting where the interfaces are modeled explicitly by interface objects showing overlappings of occurrences of productions in the given object. Moreover, productions are applied which are not independent of each other.

All the following types of parallel replacement are formulated for arbitrary categories.

Given a set of so-called component productions with occurrences in a given object there may be interfaces which show overlappings of the productions in the object. If the interfaces are preserved during the derivation we have parallel rewriting with constant interfaces modeled by interface objects showing the same overlappings between the left and right hand sides of the component productions.

Otherwise the interfaces can be dissolved or extended during the derivations. In this case, an interface is not modeled by only one object but by a production which has to be a subproduction of all component productions that use it as interface. These interface productions are used to amalgamate parts of the component productions in order to avoid deletion or addition of subobjects inside the whole object several times.

If an object is not covered completely by occurrences the context has to be determined and joined unchanged to the transformed object.

In this section three kinds of parallel replacement are presented which cover the aspects mentioned above. These are star-parallel derivations where the whole object given is replaced in parallel. The amalgamated derivation consists of the generation of a new production that amalgamates the component productions according to their interface productions. Afterwards, this new production can be applied to the whole object by a direct derivation. The advantage of these amalgamated derivations is the efficient possibility of frequent reusage of the production generated if the covered part occurs in different contexts.

For the last kind of parallel replacement presented, the contextual parallel derivation, the context is determined and joined unchanged to the object derived in parallel.

First, we have to define how an object is covered by different component productions and what kind of interface productions are allowed.

3.1 Definition *Given a production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ a production $s = (L_s \xleftarrow{l_s} I_s \xrightarrow{r_s} R_s)$ is called* subproduction *of p if there are morphisms $L_s \xrightarrow{e} L$, $I_s \xrightarrow{f} I$ and $R_s \xrightarrow{g} R$ such that $e \circ l_s = l \circ f$ and $g \circ r_s = r \circ f$, i.e. the following diagram commutes. The triple $t = (e, f, g)$ from s to p (short $s \xrightarrow{t} p$) is called* subproduction embedding.

$$\begin{tikzcd}
L_s \arrow[d, "e"] \arrow[dr, phantom, "(=)"] & I_s \arrow[l, "l_s"'] \arrow[r, "r_s"] \arrow[d, "f"] \arrow[dr, phantom, "(=)"] & R_s \arrow[d, "g"] \\
L & I \arrow[l, "l"'] \arrow[r, "r"] & R
\end{tikzcd}$$

3.2 Definition *Let $T = (\mathcal{P}, \mathcal{SP}, E)$ be a triple consisting of sets of productions $\mathcal{P} = \{p_i | p_i = (L_i \xleftarrow{l_i} I_i \xrightarrow{r_i} R_i),\ 1 \le i \le n\}$, $\mathcal{SP} = \{s_{ij} | s_{ij} = (L_{ij} \xleftarrow{l_{ij}} I_{ij} \xrightarrow{r_{ij}} R_{ij}),\ 1 \le i < j \le n\}$ and subproduction embeddings $E = \{t_{ij} | 1 \le i \ne j \le n\}$ where t_{ij} are embeddings of s_{ij} into p_i and t_{ji} are embeddings of s_{ij} into p_j for all $1 \le i < j \le n$. T is called* component production set.

Given T and an object S a set $O = \{o_i | 1 \le i \le n\}$ of occurrences $L_i \xrightarrow{o_i} S$ is called parallel occurrence *if $o_i \circ e_{ij} = o_j \circ e_{ji}$, $\forall\, 1 \le i < j \le n$.*

T and O together are called partial covering *$COV = (T, O)$ of S.*

$$\begin{tikzcd}
 & L_i \arrow[dl, "o_i"'] & I_i \arrow[l, "l_i"'] \arrow[r, "r_i"] & R_i \\
S \arrow[r, phantom, "(=)"] & L_{ij} \arrow[u, "e_{ij}"] \arrow[d, "e_{ji}"'] \arrow[ur, phantom, "(=)"] \arrow[dr, phantom, "(=)"] & I_{ij} \arrow[l, "l_{ij}"'] \arrow[r, "r_{ij}"] \arrow[u, "f_{ij}"'] \arrow[d, "f_{ji}"] \arrow[ur, phantom, "(=)"] \arrow[dr, phantom, "(=)"] & R_{ij} \arrow[u, "g_{ij}"'] \arrow[d, "g_{ji}"] \\
 & L_j \arrow[ul, "o_j"] & I_j \arrow[l, "l_j"'] \arrow[r, "r_j"] & R_j
\end{tikzcd}$$

In the case that all subproductions s_{ij} are identical productions, T is called a component production set with constant interfaces.

A covering COV of S is called complete *if the pushout-star object (see definition 7.14) of morphism star $(L_i \xleftarrow{e_{ij}} L_{ij} \xrightarrow{e_{ji}} L_j)_{1\leq i<j\leq n}$ exists and is isomorphic to S.*

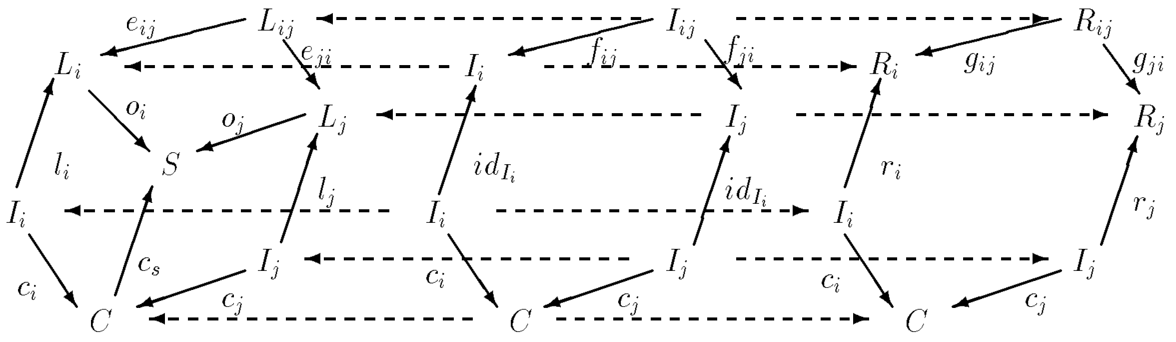
A covering which usually covers just partially can be completed if the context is determined and explicitly mentioned in the covering.

3.3 Definition *Let $COV = (T, O)$ be a partial covering of object S as defined in 3.2 and let object C, called the context object, with morphisms $I_i \xrightarrow{c_i} C$, $\forall\, 1 \leq i \leq n$, and $C \xrightarrow{c_s} S$ be a POS–complement (see def. 7.16) of morphism–star $(L_i \xleftarrow{e_{ij}} L_{ij} \xrightarrow{e_{ji}} L_j)_{1\leq i<j\leq n}$, its parallel occurrence $L_i \xrightarrow{o_i} S$ and morphisms l_i, $\forall\, 1 \leq i \leq n$. Additionally, we have $c_i \circ f_{ij} = c_j \circ f_{ji}$, $\forall 1 \leq i < j \leq n$. Then a* completed covering $COV' = (T', O')$ *of COV is defined consisting of $T' = (\mathcal{P}', \mathcal{SP}', E')$ with $\mathcal{P}' = \mathcal{P} \cup \{p_{n+1} \mid p_{n+1} = (C \xleftarrow{id_C} C \xrightarrow{id_C} C)\}$,*

$$\mathcal{SP}' = \mathcal{SP} \cup \{s_{i(n+1)} \mid s_{i(n+1)} = (I_i \xleftarrow{id_{I_i}} I_i \xrightarrow{id_{I_i}} I_i),\ 1 \leq i \leq n\} \ \textit{and}$$

$$E' = E \cup \{t_{i(n+1)} | t_{i(n+1)} = (l_i, id_{I_i}, r_i),\ 1 \leq i \leq n\} \cup \{t_{(n+1)i} | t_{(n+1)i} = (c_i, c_i, c_i),\ 1 \leq i \leq n\}$$

and $O' = O \cup \{c_s\}$.



Completed coverings may not exist since, even if the underlying category is finitely cocomplete. POS-complements do exist only under certain conditions called gluing conditions. Moreover, POS-complements are not unique, in general. (See [40] for conditions for the existence of unique POS-complements.) The completed covering is unique up to isomorphism if the POS-complement C with all c_i and c_s is so, too.

To get comparability between completed coverings and applicability of productions in direct derivations we need the condition $c_i \circ f_{ij} = c_j \circ f_{ji}$, $\forall 1 \leq i < j \leq n$, for completed coverings. This condition is used in the proof of the following Parallel Replacement Theorem II (4.4) to make sure that the pushout–star of $(I_i \xleftarrow{f_{ij}} I_{ij} \xrightarrow{f_{ji}} I_j)_{1\leq i<j\leq n+1}$ with morphisms $f_{i(n+1)} = id_{I_i}$ and $f_{(n+1)i} = c_s$, $\forall\, 1 \leq i \leq n$ is C. (See also the Context–Extending–Lemma 4.7. For more details concerning the relation between POS-complements and corresponding PO-complements see lemma 5.14 in [40].)

3.4 Lemma *If covering COV' is defined as in def. 3.3 it is a complete covering of object S.*

Proof:

1. Since C is POS-complement we have $POS(L_i \xleftarrow{e_{ij}} L_{ij} \xrightarrow{e_{ji}} L_j)_{1 \le i < j \le n+1} = S$ with $L_{i(n+1)} \xrightarrow{e_{(n+1)i}} L_{n+1} = I_i \xrightarrow{c_i} C$ and $L_{n+1} \xrightarrow{o_{n+1}} S = C \xrightarrow{c_s} S, \ \forall\, 1 \le i \le n$.
2. (a) Each production $s_{ij} \in \mathcal{SP}$ is subproduction of p_i and $p_j \in \mathcal{P}, \ \forall\, 1 \le i < j \le n$, because COV is a covering.

   (b) Each production $s_{i(n+1)} = (I_i \xleftarrow{id_{I_i}} I_i \xrightarrow{id_{I_i}} I_i) \in \mathcal{SP}'$ is subproduction of $p_i = (L_i \xleftarrow{l_i} I_i \xrightarrow{r_i} R_i) \in \mathcal{P}', \ \forall\, 1 \le i \le n$.

   (c) Each production $s_{i(n+1)}$ is subproduction of $p_{n+1} = (C \xleftarrow{id_C} C \xrightarrow{id_C} C) \in \mathcal{P}'$, $\forall\, 1 \le i \le n$, since we have $c_i \circ id_{I_i} = c_i = c_i \circ id_C, \ \forall\, 1 \le i \le n$.

□

3.5 Definition *Given a complete covering COV of an object S a* star-parallel derivation *$S \Longrightarrow_{sp} S'$ over COV is given by the following diagram where the pushout–star construction of $(R_i \xleftarrow{g_{ij}} R_{ij} \xrightarrow{g_{ji}} R_j)_{1 \le i < j \le n}$ yields object S' and morphisms $R_i \xrightarrow{g_i} S'$, $1 \le i \le n$.*

$$\begin{tikzcd}
& L_i \arrow[dl] & I_i \arrow[l] \arrow[r] & R_i \arrow[dr, "g_i"] & \\
S \arrow[r, phantom, "(POS)"] & L_{ij} \arrow[u] \arrow[d] & I_{ij} \arrow[l] \arrow[r] \arrow[u] \arrow[d] \arrow[ul, phantom, "(=)"] \arrow[ur, phantom, "(=)"] \arrow[dl, phantom, "(=)"] \arrow[dr, phantom, "(=)"] & R_{ij} \arrow[u, "g_{ij}"'] \arrow[d, "g_{ji}"] \arrow[r, phantom, "(POS)"] & S' \\
& L_j \arrow[ul] & I_j \arrow[l] \arrow[r] & R_j \arrow[ur, "g_j"'] &
\end{tikzcd}$$

The component production set T of COV provides also a complete covering for S' and the POS–object of $(R_i \xleftarrow{g_{ij}} R_{ij} \xrightarrow{g_{ji}} R_j)_{1 \le i < j \le n}$ is S'. Thus, there is also star-parallel derivation $S' \Longrightarrow_{sp} S$ via $COV^{-1} = (T^{-1}, \{g_i \mid 1 \le i \le n\})$ where T^{-1} contains all inverse productions with the same embedding morphisms of T.

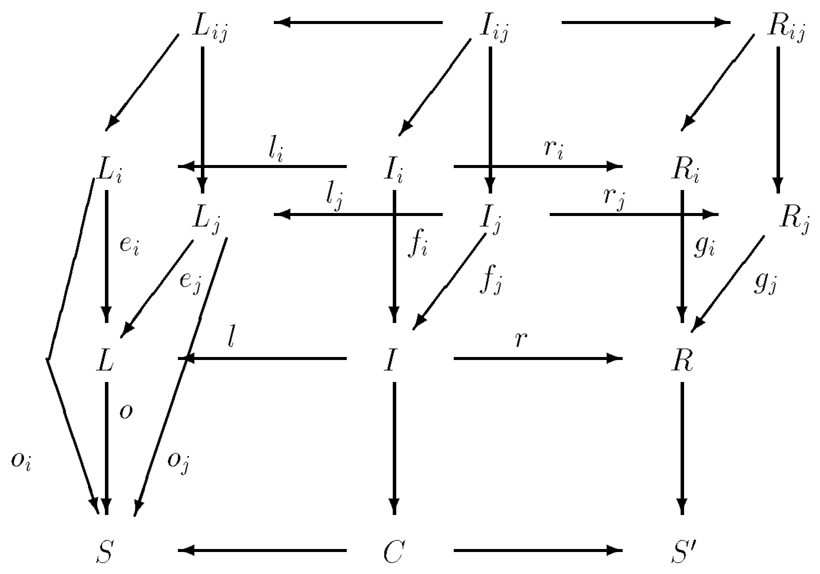
The star-parallel derivation over a covering with constant interfaces was first defined as parallel replacement on graphs in [11].

If only a part of object S has to be covered by occurrences of productions we define an amalgamated derivation and a contextual parallel derivation.

3.6 Definition *Given a covering COV of an object S as defined in 3.2, an* amalgamated derivation *$S \Longrightarrow_a S'$ over COV consists of the following star-parallel derivation $L \Longrightarrow_{sp} R$ and direct derivation $S \Longrightarrow_d S'$:*

1. *Construction of an amalgamated production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ by the star-parallel derivation $L \Longrightarrow_{sp} R$ via $COV' = (T, \{L_i \xrightarrow{e_i} L \mid 1 \le i \le n\})$ where all e_i are the pushout-star of $(L_i \xleftarrow{e_{ij}} L_{ij} \xrightarrow{e_{ji}} L_j)_{1 \le i < j \le n}$. The pushout–star construction of the interface morphism star $(I_i \xleftarrow{f_{ij}} I_{ij} \xrightarrow{f_{ji}} I_j)_{1 \le i < j \le n}$ yields the interface object I with morphisms $I_i \xrightarrow{f_i} I$. The morphism l (r) is the induced morphism such that $l \circ f_i = e_i \circ l_i$ ($r \circ f_i = g_i \circ r_i$), $\forall\, 1 \le i \le n$. The morphism $L \xrightarrow{o} S$ is the induced morphism such that $o \circ e_i = o_i, \ \forall\, 1 \le i \le n$.*

2. *Application of the amalgamated production p to the object S with the occurrence o by a direct derivation $S \Longrightarrow_d S'$.*



The r–amalgamated derivation defined with additional application conditions concerning the subproduction relation in [6] is a special case of the amalgamated derivation if $n = 2$.

While for each amalgamated derivation a direct derivation can uniquely be related a direct derivation can be the second part of different amalgamated derivations.

Amalgamated derivations can be compared with a special kind of star-parallel derivations where the given coverings are completed.

3.7 Definition *Given a covering COV of an object S such that COV can be completed to a covering COV' as in definition 3.3* a contextual parallel derivation *$S \Longrightarrow_{con} S'$ is given by the star-parallel derivation $S \Longrightarrow_{sp} S'$ via COV'.*

Remark: By definition each contextual parallel derivation $S \Longrightarrow_{con} S'$ is a star-parallel derivation $S \Longrightarrow_{sp} S'$. The other way around, for each $S \Longrightarrow_{sp} S'$ over a covering COV there is also a contextual parallel derivation $S \Longrightarrow_{con} S'$ via completed covering COV' of COV if C being the POS-object of $(I_i \xleftarrow{f_{ij}} I_{ij} \xrightarrow{f_{ji}} I_j)_{1 \leq i < j \leq n}$ exists and all c_i's being POS-morphisms, c_s is obtained then as induced morphism from C to S. It is easy to proof that COV' is a completed covering of COV.

A special case of a contextual parallel derivation is the direct derivation. For $n = 1$ the covering consists of only one object L_1 and the context completion is the context object C together with morphisms $I_1 \xrightarrow{c_1} C$ and $C \xrightarrow{c_s} S$ such that the pushout construction over $(L_1 \xleftarrow{l_1} I_1 \xrightarrow{c_1} C)$ yields S. The sets of productions $\mathcal{P}$ and $\mathcal{S}$ are extended by the identical productions $(C \xleftarrow{id_C} C \xrightarrow{id_C} C)$ and $(I_1 \xleftarrow{id_{I_1}} I_1 \xrightarrow{id_{I_1}} I_1)$ and result S' of the derivation is obtained by the pushout construction over $(R_1 \xleftarrow{r_1} I_1 \xrightarrow{c_1} C)$.

The different kinds of parallel replacement just defined are compared with each other in the next section.

A system where objects can be derived via parallel replacement shall form a PHLR-system. The effects of parallel replacement are specified very concretely by component production sets where each production is applied exactly once. Comparing this with the L-systems way of parallel replacement productions can be applied more than once in one step. Generalizing this for PHLR-systems the notion of synchronization rules is introduced. Such a rule is the basis for the construction of coverings dependent of the given object.

There are a lot of possibilities to construct a covering from such a synchronization rule even if the object that has to be covered is fixed. In the following different covering constructions are presented.

For the description of a complex operation, the elementary rules which can be applied in principle are summarized in synchronization rules. Moreover, all interaction possibilities with other elementary rules are described by specifying the subrule embeddings allowed. One of the interaction possibilities has to be used for synchronizing the elementary rules. If interaction is not required the empty rule has to be subrule of the corresponding rules.

3.8 Definition *Let $\mathcal{E}$ be a set of productions, called* elementary productions *and $\mathcal{S}$ a set of productions, called subproductions. Let SE be a set of subproduction embeddings of the form $se : s \to e$ where $s \in \mathcal{S}$ and $e \in \mathcal{E}$. A* synchronization rule *$\mathcal{SR}$ is then a triple $\mathcal{SR} = (\mathcal{E}, \mathcal{S}, SE)$. For each $e \in \mathcal{E} : I(e) = \{se \in SE | se : s \to e \text{ for some } s \in \mathcal{S}\}$ describes its* interface.

If $\forall e \in \mathcal{E}$: the cardinality of $I(e)$, $|I(e)|$, is less or equal to 1, $\mathcal{SR}$ is called simple.

If the subproductions of all $se \in \bigcup_{e \in \mathcal{E}} I(e)$ are identical, $\mathcal{SR}$ is called constant.

A synchronization rule $(\{e\}, \emptyset, \emptyset)$ is called basic.

A synchronization rule can also be seen as a graph where the nodes are labeled by elementary and subproductions. The arcs are labeled by subproduction embeddings. Thus, a synchronization rule can be described by a functor $G \to \mathbf{PROD(CAT)}$ where G is graph and $\mathbf{PROD(CAT)}$ is the category of productions in the category $\mathbf{CAT}$. Morphisms in this category are subproduction embeddings as defined in 3.1. Composition and identities are defined componentwise via that of morphisms in $\mathbf{CAT}$. Hence, an isomorphism in $\mathbf{PROD(CAT)}$ is defined via three isomorphisms in $\mathbf{CAT}$.

Morphisms f and f' in $\mathbf{PROD(CAT)}$ are isomorphic if there are isomorphisms $i : dom(f) \to dom(f')$ and $i' : cod(f) \to cod(f')$ such that $f' \circ i = i' \circ f$.

If we want to describe coverings by a functor similar to synchronization rules we consider the category $\mathbf{APAIR}_S\mathbf{(CAT)}$ of application pairs. Each application pair consists of a production with its occurrence in object S. Morphisms in $\mathbf{APAIR}_S\mathbf{(CAT)}$ are described by subproduction embeddings which have to commute with the corresponding occurrences. A covering can be described then also by a functor $CG \to \mathbf{APAIR_S(CAT)}$ from a special graph CG. This kind of description is used in the following to define a special covering construction.

Now, we describe all coverings over a given synchronization rule.

3.9 Definition *Given a synchronization rule $\mathcal{SR} = (\mathcal{E}, \mathcal{S}, SE)$ and an object S the* covering set *$\mathcal{COV}(\mathcal{SR}, S)$ of S by $\mathcal{SR}$ consists of all partial coverings $COV = ((\mathcal{P}, \mathcal{SP}, E), O)$ such that there are mappings*

- $m_p : \mathcal{P} \rightarrow \mathcal{E}$ *with* $m_p(p_i) \cong p_i$, $1 \leq i \leq n$,
- $m_{sp} : \mathcal{SP} \rightarrow \mathcal{S}$ *with* $m_{sp}(s_{ij}) \cong s_{ij}$, $1 \leq i < j \leq n$ *and*
- $m_{se} : E \rightarrow SE$ *with* $m_{se}(se_{ij}) \cong se_{ij}$, $1 \leq i \neq j \leq n$

such that $dom(m_{se}(se_{ij})) = m_p(dom(se_{ij}))$ *and* $cod(m_{se}(se_{ij})) = m_{sp}(cod(se_{ij}))$, $\forall 1 \leq i \neq j \leq n$.

Some coverings which are interesting with regard to the examples that will be investigated later on are described in the following definition.

3.10 Definition *We distinguish the following* special sets of coverings *which are subsets of* $\mathcal{COV}(\mathcal{SR}, S)$ *and defined as in 3.2:*

1. Basic coverings: *All* $COV \in \mathcal{COV}^{basic}(\mathcal{SR}, S)$ *satisfy the condition:* $n = 1$.
2. Complete coverings*: All* $COV \in \mathcal{COV}^{comp}(\mathcal{SR}, S)$ *are complete.*
3. Fully synchronized coverings*: All* $COV \in \mathcal{COV}^{sync}(\mathcal{SR}, S)$ *satisfy the condition:* $\forall 1 \leq i < j \leq n : (L_i \longleftarrow L_{ij} \longrightarrow L_j)$ *is the pullback of* $(L_i \longrightarrow S \longleftarrow L_j)$.
4. Region-coverings*: All* $COV \in \mathcal{COV}^{reg}(\mathcal{SR}, S)$ *satisfy the condition: Let* $COV' = ((\mathcal{P}, \mathcal{SP}', E'), O)$ *with* $\mathcal{SP}' = \mathcal{SP} - \{s \in \mathcal{SP} | s \ncong j\}$ *where* j *denotes the empty production and* $E' = \{se \in E | se : s \rightarrow p \text{ and } s \in \mathcal{SP}'\}$*, i.e.* E *restricted to* $\mathcal{SP}'$*.* $COV \in \mathcal{COV}^{reg}(\mathcal{SR}, S)$ *if* COV' *can be described by a functor* $CG \stackrel{COV'}{\rightarrow}$ $\mathbf{APAIR_S(CAT)}$ *where* CG *is a connected graph.*
5. Different-occurrence-coverings*: All* $COV \in \mathcal{COV}^{diff}(\mathcal{SR}, S)$ *satisfy the condition:* $\neg\exists 1 \leq i < j \leq n : o_i \cong o_j$ *if* $p_i \cong p_j$.
6. All-occurrence-coverings*: Let* $\mathcal{COV}^{all}(\mathcal{SR}, S) \subseteq \mathcal{COV}^{diff}(\mathcal{SR}, S)$ *such that* $\forall COV = (T, O) \in \mathcal{COV}^{all}(\mathcal{SR}, S) : \neg\exists COV' = (T', O') \in \mathcal{COV}^{diff}(\mathcal{SR}, S)$ *such that* $\forall o \in O : o \in O'$ *and* $\exists o' \in O' : o' \notin O$*, i.e.* $O' \subset O$.
7. Local-different-occurrence-coverings*: Let* $\mathcal{COV}^{locdiff}(\mathcal{SR}, S) \subseteq \mathcal{COV}^{diff}(\mathcal{SR}, S)$ *such that for all* $COV \in \mathcal{COV}^{diff}(\mathcal{SR}, S)$ $\forall 1 \leq i < j_k, j_l \leq n : se_{ij_k} = se_{ij_l}$
8. Local-all-occurrence-coverings*: Let* $\mathcal{COV}^{locall}(\mathcal{SR}, S) \subseteq \mathcal{COV}^{locdiff}(\mathcal{SR}, S)$ *such that* $\forall COV = (T, O) \in \mathcal{COV}^{locall}(\mathcal{SR}, S) : \neg\exists COV' = (T', O') \in \mathcal{COV}^{locdiff}(\mathcal{SR}, S)$ *such that* $\forall o \in O : o \in O'$ *and* $\exists o' \in O' : o' \notin O$.

A basic covering is exactly one production and its occurrence to an object S.
Complete coverings cover the whole object S, thus, they are *real* coverings.
In coverings which are fully synchronized, the images of all subproductions in S describe exactly the overlapping of images of corresponding elementary productions.
Region-coverings describe coverings where a whole region, i.e. a connected part of object S, is covered.
Different-match-coverings are not allowed to contain isomorphic occurrence for isomorphic elementary productions.

The algorithmic way of constructing all-occurrence-coverings for a given synchronization rule $\mathcal{SR}$ and an object S is the following:

Look first for all occurrences of all elementary productions of $\mathcal{E}$ in S. For each two distinct occurrences of elementary productions p_1 and p_2 find then a common subproduction s in $\mathcal{SP}$ with subproduction embeddings into p_1 and p_2 such that the left hand sides of p_1 and p_2 overlap in the mapping of the left hand side of s in S which has to be covered. The covering of S consists of as many copies of all elementary productions as different occurrences have been found together with their occurrences in S. Furthermore, for each two copies of them a copy of the right subproduction is joined. All elementary productions with occurrences in S that do not have common subproductions do not belong to the covering of S.

All subproductions of local-all-occurrence-coverings have to be the same for fullfilling the corresponding condition, i.e. $|\mathcal{SP}| = 1$.

Considering somehow simplified synchronization rules, uniqueness can be stated for some covering constructions.

For example, $\mathcal{COV}^{all}(\mathcal{SR}, S)$ usually has cardinality greater than 1. If we assume that we have two different occurrences o_1 and o_2 of the elementary production e and the set $I(e)$ has cardinality greater than 1. Then we can choose each subproduction embedding $se : s \to e \in I(e)$ if occurrences o_1 and o_2 overlap in the image of s in S.

3.11 Lemma *Let $\mathcal{SR} = (\mathcal{E}, \mathcal{S}, SE)$ be a synchronization rule and S an object.*

1. *If $\mathcal{SR}$ is basic $\mathcal{COV}^{sync}(\mathcal{SR}, S) = \mathcal{COV}^{reg}(\mathcal{SR}, S) = \mathcal{COV}^{diff}(\mathcal{SR}, S) = \mathcal{COV}^{all}(\mathcal{SR}, S) = \mathcal{COV}^{locdiff}(\mathcal{SR}, S) = \mathcal{COV}^{locall}(\mathcal{SR}, S) = \mathcal{COV}^{basic}(\mathcal{SR}, S)$*

2. *If $\mathcal{SR}$ is simple $|\mathcal{COV}^{all}(\mathcal{SR}, S)| = 1$.*

3. *If $\forall e \in \mathcal{E}$: no two subproductions in $I(e)$ have isomorphic left hand sides , $|\mathcal{COV}^{sync}(\mathcal{SR}, S) \cap \mathcal{COV}^{all}(\mathcal{SR}, S)| = 1$.*

Proof: These results follow directly from the construction chosen above. □

Note that the coverings constructed with one of the constructions given above are empty if $n = 0$, i.e. no occurrences of productions of $\mathcal{P}$ can be found.

3.12 Definition

1. *A* parallel high–level replacement system *$PHLRS_D = (S, \mathcal{R}, T)_D$ of type D based on a category* **CAT** *is given by a start object S in* **CAT**, *a set $\mathcal{R}$ of synchronization rules and a class T of objects in* **CAT**, *called terminal colored objects. Type $D = \{sp, a, con\}$ indicates the type of derivations that are used for parallel replacement.*

2. *Given a type D an object S' is* D-derivable *from S via $\mathcal{R}$ if there is a sequence of derivations $S_{i-1} \Longrightarrow_D S_i$ via COV_i and $COV_i \in \mathcal{COV}(\mathcal{SR}_i, S_{i-1})$ (see definition 3.9 with $\mathcal{SR}_i \in \mathcal{R}$ for $1 \leq i \leq m$ such that $S = S_0$, $S' = S_m$.*

3. *The* language *$\mathcal{L}(PHLRS_D)$ of an PHLR-system $PHLRS_D = (S, \mathcal{R}, T)$ is given by the set of all terminal colored objects in* **CAT** *D-derivable from S. $\mathcal{L}^{basic}(PHLRS_D)$, $\mathcal{L}^{comp}(PHLRS_D)$, $\mathcal{L}^{sync}(PHLRS_D)$, $\mathcal{L}^{reg}(PHLRS_D)$, $\mathcal{L}^{diff}(PHLRS_D)$, $\mathcal{L}^{all}(PHLRS_D)$,*

$\mathcal{L}^{locdiff}(PHLRS_D)$ and $\mathcal{L}^{locall}(PHLRS_D)$ denote the sublanguages of $\mathcal{L}(PHLRS_D)$ which are generated using basic, complete, fully synchronized, region-, different-occurrence-, all-occurrence-, local-different-occurrence- or local-all-occurrence-coverings, only.

4 Relations between different parallel replacement concepts

In this section we want to relate the different kinds of parallel replacement given in the last sections. If all interfaces are constant amalgamated derivations can be compared with direct derivations via parallel productions. In theorem 4.1 we show that there is a correspondence between these two kinds of derivation and it turns out that the explicitly given interfaces in amalgamated derivations are handled implicitly in direct derivations via parallel productions.

This theorem gives information when an amalgamated derivation can be decomposed using Parallelism Theorem 2.5. This means that for each given amalgamated derivation with constant interfaces there is a direct derivation via a parallel production which can be sequentialized by the 'analysis'-construction of the Parallelism Theorem.

Amalgamated derivations and contextual parallel derivations correspond to each other bijectively. This is shown in theorem 4.4.

4.1 Parallel Replacement Theorem I *Let* **CAT** *be a finitely cocomplete category and $COV = (T, O)$ and $COV' = (T', O')$ partial coverings as in definition 3.2. $T = (\mathcal{P}, \mathcal{SP}, E)$ and $T' = (\mathcal{P}, \mathcal{SP}', E')$ are component production sets with constant interfaces.*

1. Implicate*: Given an amalgamated derivation $S \Longrightarrow_a S'$ via COV there is a direct derivation $S \Longrightarrow_d S'$ via parallel production $\sum_1^n p_i = (\sum_1^n L_i \xleftarrow{\sum l} \sum_1^n I_i \xrightarrow{\sum r} \sum_1^n R_i)$ with $p_i \in \mathcal{P}$, $\forall 1 \leq i \leq n$.*

2. Explicate*: Given a direct derivation $S \Longrightarrow_d S'$ via parallel production $\sum_1^n p_i$ there is an amalgamated derivation $S \Longrightarrow_a S'$ via COV'.*

3. E/I-Identity*: Let $S \Longrightarrow_d S'$ be a direct derivation via parallel production $\sum_1^n p_i$. Doing first 'explicate' and afterwards 'implicate' the derivation $S \Longrightarrow_d S'$ via $\sum_1^n p_i$ is constructed again.*

We cannot state I/E-Identity because for the construction 'explicate' there are different possibilities of choosing interfaces I_{ij} between the I_i's. Among others there are two distinguished ways to choose interfaces I_{ij}:

1. All I_{ij} are equal to the initial object J which means purely implicit handling of overlapping of I_i and I_j.

2. All I_{ij} are pullback objects of $(I_i \xrightarrow{c_i} C \xleftarrow{c_j} I_j)$ for $1 \leq i < j \leq n$ which means explicit declaration of the overlapping of I_i and I_j.

Proof:

1. *Explicate*:

   Given a direct derivation $S \Longrightarrow_d S'$ via production $\sum_1^n p_i$:

$$
\begin{tikzcd}
L_i \arrow[dd, "e'_i"] \arrow[ddd, dotted, "o_i"'] & & I_i \arrow[ll, dashed, "l_i"'] \arrow[rr, dashed, "r_i"] \arrow[dd, "f'_i"] \arrow[ddd, dotted, "c_i"'] & & R_i \arrow[dd, "g'_i"] \arrow[ddd, dotted, "q_i"'] & \\
 & L_j \arrow[dl, "e'_j"'] \arrow[ddl, dotted, "o_j"] & & I_j \arrow[ll, dashed, "l_j"'] \arrow[rr, dashed, "r_j"] \arrow[dl, "f'_j"'] \arrow[ddl, dotted, "c_j"] & & R_j \arrow[dl, "g'_j"'] \arrow[ddl, dotted, "q_j"] \\
\sum_1^n L_i \arrow[d, "\sum o"] \arrow[drr, phantom, "(1)"] & & \sum_1^n I_i \arrow[ll, "\sum l"'] \arrow[rr, "\sum r"] \arrow[d, "\sum c"] \arrow[drr, phantom, "(2)"] & & \sum_1^n R_i \arrow[d, "\sum q"] & \\
S & & C \arrow[ll] \arrow[rr] & & S' &
\end{tikzcd}
$$

   Since finite coproducts are pushout-stars over the initial object (compare for example [23] or the appendix) the coproducts over families (L_i), (I_i) and (R_i), $\forall\, 1 \le i \le n$, are equal (up to isomorphism) to pushout-stars over the morphism stars $(L_i \longleftarrow J \longrightarrow L_j)_{1\le i<j\le n}$, $(I_i \longleftarrow J \longrightarrow I_j)_{1\le i<j\le n}$ and $(R_i \longleftarrow J \longrightarrow R_j)_{1\le i<j\le n}$. All newly created diagram squares commute because of uniqueness of initial morphisms. Thus, there exists an amalgamated derivation $S \Longrightarrow_a S'$ via COV' with $T' = (\mathcal{P}, \mathcal{SP}', E')$ with constant interfaces $\mathcal{SP}' = \{s_{ij} \mid s_{ij} = (J \longleftarrow J \longrightarrow J),\ \forall\, 1 \le i < j \le n\}$ where $I \xrightarrow{l} L = \sum l$ and $I \xrightarrow{r} R = \sum r$ consisting of the diagram in definition 3.6 where all $L_{ij} = I_{ij} = R_{ij} = J$.

2. *Implicate*:

   Let $S \Longrightarrow_a S'$ be an amalgamated derivation via COV. Since **CAT** is finitely cocomplete, we can construct first the finite coproducts $\sum_1^n L_i$, $\sum_1^n I_i$ and $\sum_1^n R_i$.

   The morphism $\sum l$ ($\sum r$) is the induced morphism such that

$$I_i \xrightarrow{l_i} L_i \xrightarrow{e'_i} \sum_1^n L_i = I_i \xrightarrow{f'_i} \sum_1^n I_i \xrightarrow{\sum l} \sum_1^n L_i$$

$$(I_i \xrightarrow{r_i} R_i \xrightarrow{g'_i} \sum_1^n R_i = I_i \xrightarrow{f'_i} \sum_1^n I_i \xrightarrow{\sum r} \sum_1^n R_i).$$

   Moreover, there are induced morphisms $\sum_1^n L_i \xrightarrow{\sum o} S$, $\sum_1^n I_i \xrightarrow{\sum c} C$ and $\sum_1^n R_i \xrightarrow{\sum q} S'$ such that

   - $L_i \xrightarrow{e'_i} \sum_1^n L_i \xrightarrow{\sum o} S = L_i \xrightarrow{o_i} S = L_i \xrightarrow{e_i} L \xrightarrow{o} S$
   - $I_i \xrightarrow{f'_i} \sum_1^n I_i \xrightarrow{\sum c} C = I_i \xrightarrow{c_i} C = I_i \xrightarrow{f_i} I \xrightarrow{c} C$ and
   - $R_i \xrightarrow{g'_i} \sum_1^n R_i \xrightarrow{\sum q} S' = R_i \xrightarrow{q_i} S' = R_i \xrightarrow{g_i} R \xrightarrow{q} S'$, $\forall\, 1 \le i \le n$.

   If both squares (1) and (2) in the diagram above are pushouts the direct derivation $S \Longrightarrow_d S'$ via parallel production $\sum_1^n p_i = (\sum_1^n L_i \xleftarrow{\sum l} \sum_1^n I_i \xrightarrow{\sum r} \sum_1^n R_i)$ is constructed.

Commutativity can be stated for (1) and (2) since the universal property of $\sum_1^n I_i$ holds for S and S'.

Proving the universal property of (1) we can state the following:

Given an object X with morphisms $\sum_1^n L_i \xrightarrow{ox'} X$ and $C \xrightarrow{cx} X$ satisfying $ox' \circ \sum l = cx \circ \sum c$ we can state $ox_i \circ e_{ij} = ox_j \circ e_{ji}$, $\forall 1 \leq i < j \leq n$.

$ox_i \circ e_{ij} = ox_i \circ e_{ij} \circ l_{ij} = ox_i \circ l_i \circ f_{ij} = ox' \circ e'_i \circ l_i \circ f_{ij} = ox' \circ \sum l \circ f'_i \circ f_{ij} = cx \circ \sum c \circ f'_i \circ f_{ij} = cx \circ c_i \circ f_{ij} = cx \circ c_j \circ f_{ji} = cx \circ \sum c \circ f'_j \circ f_{ji} = ox' \circ \sum l \circ f'_j \circ f_{ji} = ox' \circ e'_j \circ l_j \circ f_{ji} = ox_j \circ l_j \circ f_{ji} = ox_j \circ e_{ji} \circ l_{ij} = ox_j \circ e_{ji}$

Now, there exists a unique morphism $L \xrightarrow{ox} X$ with

$$L_i \xrightarrow{e_i} L \xrightarrow{ox} X = L_i \xrightarrow{ox_i} X = L_i \xrightarrow{e'_i} \sum_1^n L_i \xrightarrow{ox'} X,\ \forall\, 1 \leq i \leq n.$$

Since S is PO-object of $(L \xleftarrow{l} I \xrightarrow{c} C)$ there is a unique morphism $S \xrightarrow{sx} X$ with $L \xrightarrow{o} S \xrightarrow{sx} X = L \xrightarrow{ox} X$ and $C \xrightarrow{s} S \xrightarrow{sx} X = C \xrightarrow{cx} X$. Consequently, it follows

$$\sum_1^n L_i \xrightarrow{ox'} X = \sum_1^n L_i \xrightarrow{\sum ll} L \xrightarrow{ox} X = \sum_1^n L_i \xrightarrow{\sum ll} L \xrightarrow{o} S \xrightarrow{sx} X = \sum_1^n L_i \xrightarrow{\sum o} S \xrightarrow{sx} X$$

The universal property of (2) can be proven analogously.

3. All colimit constructions are unique up to isomorphism. Applying the construction 'explicate' does not result into any object or morphism newly constructed. By using construction 'implicate' the coproducts $\sum_1^n L_i$, $\sum_1^n I_i$ and $\sum_1^n R_i$ are reconstructed.

□

The 'explicate'-construction is possible if the corresponding covering does not have to fulfill special properties. The following example shows a situation where 'explicate' is not possible if it has to yield a special covering.

4.2 Example

1. *Consider an HLR-system $HLRS = (S, P, T)$ where $S = L$ and $P = \{p = (L \xleftarrow{l} J \xrightarrow{r} R)\}$. Then the language $\mathcal{L}(HLRS+) = \{L\} \cup \{\sum_1^n R | n \in N\}$ is mainly the set of all coproducts over R. This result is due to the fact that matches of L can overlap in S arbitrarily often.*

   On the other side, the languages of $PHLRS_a = (S, \mathcal{R}, T)_a$ with $\mathcal{R} = (P, \{j\}, \{j \to p\})$ where the sets of coverings are restricted are subsets of $\mathcal{L}(HLRS+)$.

   $\mathcal{L}^{basic}(PHLRS_a) = \mathcal{L}^{sync}(PHLRS_a) = \mathcal{L}^{reg}(PHLRS_a) = \mathcal{L}^{diff}(PHLRS_a) =$

   $\mathcal{L}^{all}(PHLRS_a) = \mathcal{L}^{locdiff}(PHLRS_a) = \mathcal{L}^{locall}(PHLRS_a) = \{L, R\}$. In all restricted coverings except of complete ones exactly one occurrence of L in S can be found because otherwise it is not

   - *basic,*

- *fully synchronized: all occurrences have to overlap in L but the only possible interface is the empty production,*
- *a region-covering: different occurrences can only be connected by empty subproductions,*
- *a different-occurrence- or an all-occurrence-covering: there is only one possible occurrence $L \xrightarrow{id_L} L$,*
- *a local-different-occurrence- or a local-all-occurrence-covering with the same reason as above.*

2. *Let $P' = \{p' = (J \xleftarrow{l} J \xrightarrow{r} R)\}$ we have $\mathcal{L}((S, P'+, T)) = \mathcal{L}(HLRS+)$. But $\mathcal{L}^{comp}((S, \mathcal{R}', T)_a) = \{L\}$ with $\mathcal{R}' = (P', \{j\}, \{j \to p'\})$ since even a parallel application of p' would not yield a complete covering.*

4.3 Corollary *Let* **CAT** *be finitely cocomplete, all $\mathcal{SR}_u = (\mathcal{E}_u, \mathcal{S}_u, SE_u) \in \mathcal{R}$ with $u \in U$ are constant and $P = \bigcup_{u \in U} \mathcal{E}_u$:*

1. *The generated languages of $PHLRS_a = (S, \mathcal{R}, T)_a$ and $HLRS+ = (S, P+, T)$ are equal, i.e. $\mathcal{L}(PHLRS_a) = \mathcal{L}(HLRS+)$*
2. *Restricting the generated language of $PHLRS_a$ by using special coverings only it is a proper sublanguage of $\mathcal{L}(HLRS+)$, i.e.*
   (a) $\mathcal{L}^x((S, \mathcal{R}, T)_a) \subseteq \mathcal{L}((S, P+, T))$ *for* $x \in \{basic, comp, sync, reg, diff, all, locdiff, locall\}$
   (b) $\mathcal{L}^x((S, \mathcal{R}, T)_a) \neq \mathcal{L}((S, P+, T))$ *for some PHLR-systems*

Proof: 1. Direct consequence of parallel replacement theorem I. 2.(a) is obvious because of 1. 2.(b) is shown by example 4.2. □

The following Parallel Replacement Theorem II shows the very close relationship between amalgamated derivations and contextual parallel derivations.

4.4 Parallel Replacement Theorem II

Let **CAT** *be a finitely cocomplete category and COV a partial covering.*

1. Decapsulate*: Given an amalgamated derivation $S \Longrightarrow_a S'$ via COV there is a contextual parallel derivation $S \Longrightarrow_{con} S'$ via COV.*
2. Encapsulate*: Given a contextual parallel derivation $S \Longrightarrow_{con} S'$ via COV there is an amalgamated derivation $S \Longrightarrow_a S'$ via COV.*
3. Bijective Correspondence*: The constructions 'decapsulate' and 'encapsulate' are inverse to each other which means that there is a bijective correspondence between amalgamated derivations $S \Longrightarrow_a S'$ and contextual parallel derivations $S \Longrightarrow_{con} S'$ via COV.*

The proof of theorem 4.4 makes use of commutativity properties of colimits (7.20, [23]) which are also used, for example, in the framework of module specifications in [14]. In our case we have to apply it to pushouts and pushout–stars which are special colimits.
Proof:

1. *Decapsulate*:

   Let $S \Longrightarrow_a S'$ be an amalgamated derivation via COV as depicted in definition 3.6. Morphism stars $(L_i \xleftarrow{e_{ij}} L_{ij} \xrightarrow{e_{ji}} L_j)_{1\leq i<j\leq n}$,$(I_i \xleftarrow{f_{ij}} I_{ij} \xrightarrow{f_{ji}} I_j)_{1\leq i<j\leq n}$ and $(R_i \xleftarrow{g_{ij}} R_{ij} \xrightarrow{g_{ji}} R_j)_{1\leq i<j\leq n}$ can be extended to size $n+1$ such that the pushout-star objects of them and morphisms to these objects are the same if $L_{i(n+1)} = L_{n+1} = I_{i(n+1)} = I_{n+1} = R_{i(n+1)} = R_{n+1} = J,\ \forall\, 1 \leq i \leq n$. (See lemma 4.6.) Thus, also

   - L is the POS-object of $LS = (L_i \xleftarrow{e_{ij}} L_{ij} \xrightarrow{e_{ji}} L_j)_{1\leq i<j\leq n+1}$
   - I is the POS-object of $IS = (I_i \xleftarrow{f_{ij}} I_{ij} \xrightarrow{f_{ji}} I_j)_{1\leq i<j\leq n+1}$ and
   - R is the POS-object of $RS = (R_i \xleftarrow{g_{ij}} R_{ij} \xrightarrow{g_{ji}} R_j)_{1\leq i<j\leq n+1}$.

   Morphism $J \longrightarrow J$ is unique. Moreover, we have $J \longrightarrow J \longrightarrow L_i = J \longrightarrow L_i = J \longrightarrow I_i \xrightarrow{l_i} L_i$ because $J \longrightarrow L_i$ is unique and $J \longrightarrow J \longrightarrow R_i = J \longrightarrow R_i = J \longrightarrow I_i \xrightarrow{r_i} R_i$ because $J \longrightarrow R_i$ is unique.

   There is a morphism star $CS = (I_i \xleftarrow{f_{ij}} I_{ij} \xrightarrow{f_{ji}} I_j)_{1\leq i<j\leq n+1}$ with $I_{i(n+1)} \xrightarrow{f_{i(n+1)}} I_i = I_i \xrightarrow{id_{I_i}} I_i$ and $I_{i(n+1)} \xrightarrow{f_{(n+1)i}} I_{n+1} = I_i \xrightarrow{c_i} C = I_i \xrightarrow{f_i} I \xrightarrow{c} C,\ \forall\, 1 \leq i \leq n$, such that C is its POS-object. (See lemma 4.7.)

   All squares between IS and CS commute because of the uniqueness of morphisms $J \longrightarrow I_i,\ \forall\, 1 \leq i \leq n$, and $J \longrightarrow C$ or identities.

   Now the left part of the diagram just constructed consists of the solid part and the pushout over $L \xleftarrow{l} I \xrightarrow{c} C$ which is drawn dotted in the figure below. The right part of the diagram looks similarly.

   Considering morphism stars LS, IS and CS with all morphisms running in between it is easy to see that this diagram is a functor from diagram scheme $PO \times POS_{n+1}$ to **CAT**. (See definition 7.7.) This applies in the same way for RS, IS and CS with corresponding morphisms. Thus, theorem 7.20 about commutativity of colimits implies first that the pushout of diagram $(L \xleftarrow{l} I \xrightarrow{c} C)$ and the pushout-star of diagram $ELS = (L_i \xleftarrow{e_{ij}} L_{ij} \xrightarrow{e_{ji}} L_j)_{1\leq i<j\leq n+1}$ with $L_{i(n+1)} \xrightarrow{e_{i(n+!)}} L_i = I_i \xrightarrow{l_i} L_i$ and $L_{i(n+1)} \xrightarrow{e_{(n+1)i}} L_{n+1} = I_i \xrightarrow{c_i} C,\ \forall\ 1 \leq i \leq n$, are isomorphic and secondly that the pushout of diagram $(R \xleftarrow{r} I \xrightarrow{c} C)$ and the pushout-star of diagram $ERS = (R_i \xleftarrow{g_{ij}} R_{ij} \xrightarrow{g_{ji}} R_j)_{1\leq i<j\leq n+1}$ with $R_{i(n+1)} \xrightarrow{g_{i(n+1)}} R_i = I_i \xrightarrow{r_i} R_i$ and $R_{i(n+1)} \xrightarrow{g_{(n+1)i}} R_{n+1} = I_i \xrightarrow{c_i} C,\ \forall\, 1 \leq i \leq n$, are isomorphic.

   So there exists a contextual parallel derivation $S \Longrightarrow_{con} S'$ via COV consisting of ELS, CS, ERS, the rule morphisms in between and pushout-stars of ELS and ERS.

2. *Encapsulate:*

   Given a contextual parallel derivation $S \Longrightarrow_{con} S'$ over COV and T as in definition 3.7.

Restricting the morphism stars of S and S' and the interface-star to size n yields the morphism stars $LS = (L_i \xleftarrow{e_{ij}} L_{ij} \xrightarrow{e_{ji}} L_j)_{1 \leq i < j \leq n}$, $IS = (I_i \xleftarrow{f_{ij}} I_{ij} \xrightarrow{f_{ji}} I_j)_{1 \leq i < j \leq n}$ and $RS = (R_i \xleftarrow{g_{ij}} R_{ij} \xrightarrow{g_{ji}} R_j)_{1 \leq i < j \leq n}$ with L being the POS-object of LS, I being the POS-object of IS and R being the POS-object of RS.

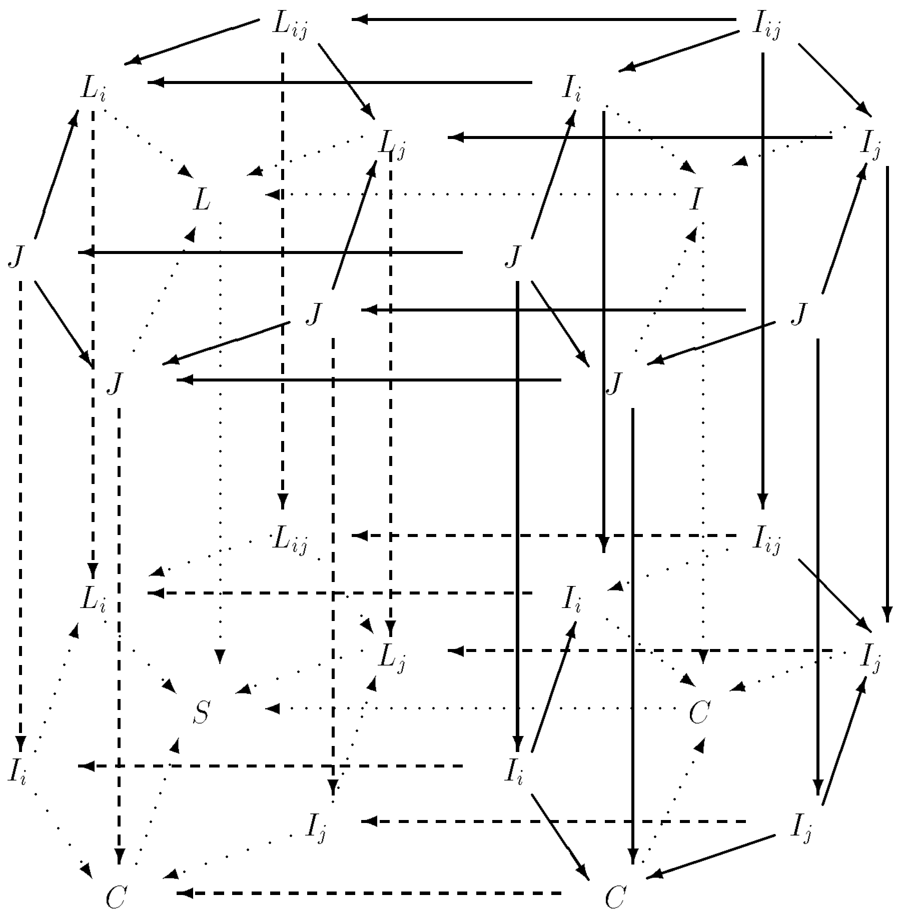
Using lemma 4.6 we extend the stars again to size $n+1$ such that the pushout-star objects remain the same if we have $L_{i(n+1)} = L_{n+1} = I_{i(n+1)} = I_{n+1} = R_{i(n+1)} = R_{n+1} = J,\ \forall\, 1 \leq i \leq n$.

All morphisms between the stars of L and I and R and I are either given by assumption, identities or initial morphisms. Thus, all squares between the stars commute by assumption or because of the uniqueness of the corresponding morphisms.

Since the POS-object of IS is I and due to the fact that we have

$$I_{ij} \xrightarrow{f_{ij}} I_i \xrightarrow{c_i} C = I_{ij} \xrightarrow{f_{ji}} I_j \xrightarrow{c_j} C,\ \forall\, 1 \leq i < j \leq n$$

because a contextual parallel derivation is a star-parallel derivation over a completed covering there is an induced morphism $I \xrightarrow{c} C$. According to the Context-Extending Lemma 4.7, C is the POS-object of $(I_i \xleftarrow{f_{ij}} I_{ij} \xrightarrow{f_{ji}} I_j)_{1 \leq i < j \leq n+1}$ with $I_{i(n+1)} \xrightarrow{f_{i(n+1)}} I_i = I_i \xrightarrow{id_{I_i}} I_i$ and $I_{i(n+1)} \xrightarrow{f_{(n+1)i}} I_{n+1} = I_i \xrightarrow{c_i} C,\ \forall\, 1 \leq i \leq n$.



Now the left part of the diagram just constructed consists of the whole diagram in the figure above except of morphisms $L \xrightarrow{o} S$ and $C \xrightarrow{c_s} S$. The right side of the diagram looks similarly.

Similarly to the "decapsulate"-step we have the morphism stars of L, I, R and C and all morphisms in between which lead to diagrams over diagram scheme $PO \times POS_{n+1}$. Thus, theorem 7.20 implies first that the pushout of diagram $(L \xleftarrow{l} I \xrightarrow{c} C)$ and the pushout-star of diagram$(L_i \xleftarrow{e_{ij}} L_{ij} \xrightarrow{e_{ji}} L_j)_{1 \leq i < j \leq n+1}$ with $L_{i(n+1)} \xrightarrow{e_{i(n+1)}} L_i = I_i \xrightarrow{l_i} L_i$ and $L_{i(n+1)} \xrightarrow{e_{(n+1)i}} L_{n+1} = I_i \xrightarrow{c_i} C, \ \forall\ 1 \leq i \leq n$, are equal and secondly that the pushout of diagram $(R \xleftarrow{r} I \xrightarrow{c} C)$ and the pushout-star of diagram $(R_i \xleftarrow{g_{ij}} R_{ij} \xrightarrow{g_{ji}} R_j)_{1 \leq i < j \leq n+1}$ with $R_{i(n+1)} \xrightarrow{g_{i(n+1)}} R_i = I_i \xrightarrow{r_i} R_i$ and $R_{i(n+1)} \xrightarrow{g_{(n+1)i}} R_{n+1} = I_i \xrightarrow{c_i} C, \ \ \forall\ 1 \leq i \leq n$, are equal. Hence, there is an amalgamated derivation $S \Longrightarrow_a S'$ via COV as depicted in definition 3.6.

3. (Bijective Correspondence)

   All pushout-star constructions are unique up to isomorphism. Applying construction 'decapsulate' there is not any object or morphism that is newly constructed. By doing construction 'encapsulate' the pushout-stars of LS, IS, RS and CS with POS-objects L, I, R and C are constructed and induce unique morphisms $I \xrightarrow{l} L$, $I \xrightarrow{r} R$ and $I \xrightarrow{c} C$.

□

4.5 Corollary *Let* **CAT** *be finitely cocomplete and* $x \in \{basic, comp, sync, reg, diff, all, locdiff, locall\}$. *The generated languages of PHLR-systems using amalgamated or contextual parallel derivations only are equal, i.e.*

- $\mathcal{L}(PHLRS_a) = \mathcal{L}(PHLRS_{con})$
- $\mathcal{L}^x(PHLRS_a) = \mathcal{L}^x(PHLRS_{con})$

Proof: Direct consequence of parallel replacement theorem II. □

For proving the theorems we need the following lemmata:

4.6 Initial-Extending-Lemma *Given a finitely cocomplete category* **CAT** *there is a pushout-star* $(S_i \xleftarrow{s_{ij}} S_{ij} \xrightarrow{s_{ji}} S_j)_{1 \leq i < j \leq n+1}$ *with* $S_{i(n+1)} = S_{n+1} = J, \ \forall\ 1 \leq i \leq n$, *pushout-star object* S *and morphisms* $S_i \xrightarrow{s_i} S, \ \ \forall\ 1 \leq i \leq n+1$, *if and only if there is a pushout-star* $(S_i \xleftarrow{s_{ij}} S_{ij} \xrightarrow{s_{ji}} S_j)_{1 \leq i < j \leq n}$ *with the same pushout-star object* S *and morphisms* $s_i \ \forall\, 1 \leq i \leq n$. *(See the left diagram below.)*

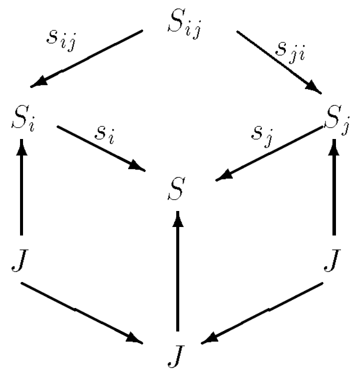
Proof:

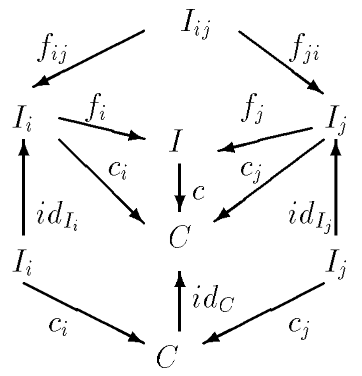
1. Given an object S being the POS-object of the morphism star $(S_i \xleftarrow{s_{ij}} S_{ij} \xrightarrow{s_{ji}} S_j)_{1 \leq i < j \leq n+1}$, we have $s_i \circ s_{ij} = s_j \circ s_{ji}, \ \ \forall\ 1 \leq i < j \leq n$. There exists a unique morphism $S \xrightarrow{u} S'$ for all objects S' with $S_{ij} \xrightarrow{s_{ij}} S_i \xrightarrow{s'_i} S' = S_{ij} \xrightarrow{s_{ji}} S_j \xrightarrow{s'_j} S'$, $\forall\, 1 \leq i < j \leq n+1$, and we have $u \circ s_i = s'_i, \ \forall\, 1 \leq i \leq n$. This is exactly each S' with $s'_i \circ s_{ij} = s'_j \circ s_{ji}, \ \ \forall\ 1 \leq i < j \leq n$, because $s'_i \circ s_{i(n+1)} = J \longrightarrow S' = s'_{n+1} \circ s_{(n+1)i}$, $\forall\, 1 \leq i \leq n$, is always satisfied. Hence S is POS-object of $(S_i \xleftarrow{s_{ij}} S_{ij} \xrightarrow{s_{ji}} S_j)_{1 \leq i < j \leq n}$.

2. Given an object S being the POS-object of the morphism star $(S_i \xleftarrow{s_{ij}} S_{ij} \xrightarrow{s_{ji}} S_j)_{1\leq i<j\leq n}$, we have $s_i \circ s_{ij} = s_j \circ s_{ji}, \ \ \forall\ 1 \leq i < j \leq n$, and $J \longrightarrow S_i \xrightarrow{s_i} S = J \longrightarrow S = J \longrightarrow J \longrightarrow S, \ \forall\, 1 \leq i \leq n$, because morphism $J \longrightarrow S$ is unique.

   For all objects S' with $s'_i \circ s_{ij} = s'_j \circ s_{ji}, \ \ \forall\ 1 \leq i < j \leq n$, and unique u as above we have $u \circ s_i = s'_i, \ \ \forall\ 1 \leq i \leq n+1$ because the uniqueness of $J \longrightarrow S'$ implies $J \longrightarrow S_i \xrightarrow{s'_i} S' = J \longrightarrow J \longrightarrow S', \ \ \forall\ 1 \leq i \leq n$. Consequently, there is a unique morphism u with $u \circ s_i = s'_i, \ \forall\ 1 \leq i \leq n$. Additionally, we have $J \longrightarrow S \xrightarrow{u} S' = J \longrightarrow S'$ because of the uniqueness of initial morphisms. Hence S is also POS-object of $(S_i \xleftarrow{s_{ij}} S_{ij} \xrightarrow{s_{ji}} S_j)_{1\leq i<j\leq n+1}$.

□





4.7 Context-Extending-Lemma *Given an object I as POS-object of a morphism star $(I_i \xleftarrow{f_{ij}} I_{ij} \xrightarrow{f_{ji}} I_j)_{1\leq i<j\leq n}$ and a morphism $I \xrightarrow{c} C$ in a finitely cocomplete category* **CAT***, object C with morphisms $I_i \xrightarrow{c_i} C = I_i \xrightarrow{f_i} I \xrightarrow{c} C, \ \ \forall\ 1 \leq i \leq n$, and id_C C is then POS-object of morphism star $(I_i \xleftarrow{f_{ij}} I_{ij} \xrightarrow{f_{ji}} I_j)_{1\leq i<j\leq n+1}$ with morphisms $I_{i(n+1)} \xrightarrow{f_{i(n+1)}} I_i = I_i \xrightarrow{id_{I_i}} I_i$ and $I_{i(n+1)} \xrightarrow{f_{(n+1)i}} C = I_i \xrightarrow{c_i} C, \ \ \forall\ 1 \leq i \leq n$. (See the right diagram above.)*

Proof: We can state $c_i \circ f_{ij} = c_j \circ f_{ji}, \ \forall\, 1 \leq i < j \leq n+1$, since

1. $c_i \circ f_{ij} = c \circ f_i \circ f_{ij} = c \circ f_j \circ f_{ji} = c_j \circ f_{ji}, \ \forall\, 1 \leq i < j \leq n$ and
2. $c_i \circ id_{I_i} = c_i = id_C \circ c_i \ \ \forall\, 1 \leq i \leq n$

For all objects C' with $I_{ij} \xrightarrow{f_{ij}} I_i \xrightarrow{c'_i} C' = I_{ij} \xrightarrow{f_{ji}} I_j \xrightarrow{c'_j} C', \ \forall\, 1 \leq i < j \leq n+1$, exists a morphism $C \xrightarrow{x} C'$ by assumption. We have $c'_i \circ id_{I_i} = c'_i = x \circ c_i, \ \forall\, 1 \leq i \leq n$ and $c \circ id = c$. Assume that there are morphisms $\alpha, \beta : C \longrightarrow C'$ with $I_i \xrightarrow{c_i} C \xrightarrow{\alpha,\beta} C' = I_i \xrightarrow{c'_i} C', \ \forall\, 1 \leq i \leq n$, it follows $\alpha = \alpha \circ id_C = x = \beta \circ id_C = \beta$.

□

5 Examples of PHLR-systems

In this section we show some examples of underlying categories for PHLR-systems and two explicit examples of PHLR-systems: the high level specification of a simple window-based graph editor based on category **GRAPHS** which is described in [8] and in example 7.2. Moreover, the movement of objects in configuration spaces based on category **CPOGRAPHS** of graphs with completely partially ordered label alphabet investigated in [30].

5.1 Examples of underlying categories for PHLR-systems

Comparing the literature concerning category theory there are a lot of finitely cocomplete categories. In the following paragraph some cocomplete categories are listed which are already presented in the framework of HLR-systems in [10] and [31].

1. the category **SETS** of sets
2. the category **GRAPHS** of colored graphs with color preserving graph morphisms (Ehrig et al. [16], [8])
3. the category **HYP-GRAPHS** of hypergraphs (Habel [18])
4. the category **STRUCT** of relational structures (Ehrig et al. [12])
5. the category **SPEC** of algebraic specifications and strict specification morphisms (Parisi-Presicce [32])
6. the category **CPOGRAPHS** of graphs with completely partially ordered label alphabet (Parisi-Presicce et al. [30])
7. the category **P-T-NETS** of place-transition nets (Hummert [19], Meseguer and Montanari [24])
8. the category **AHL-NETS** of algebraic high-level nets (Padberg et al. [31])

5.2 A simple window-based graph editor

The first example comprises an abstract specification of a simplified window-based editor which manipulates graphs following a prototype graph editor for algebraic graph transformation designed and implemented by Löwe and Beyer on SUN4-workstations under X-windows. (See [5] and [21].)

This editor supports several views on graphs which can be considered in different windows. Nodes and arcs inserted or deleted in the current window are also inserted or deleted in all other windows that show the current graph. This graph can be saved or deleted and a new graph can be loaded into the current window.

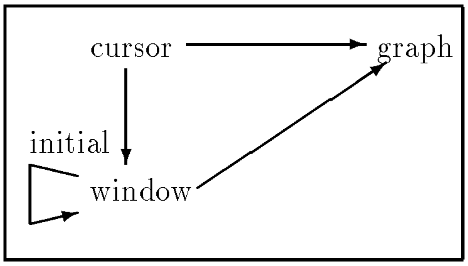
The user interface operations comprise the insertion, deletion, selection and deselection of nodes and arcs, movement and copy operations, creation and deletion of windows, change of the current window, creation and deletion of graphs, change to another graph and many more. The operations just mentioned belong mostly to the editor operations which are described below.

The abstract specification of a graph editor which does not care of the layout and position of graphs and windows but only of the relation structure of objects taking part is presented as a PHLR-system based on the category **GRAPHS**. This means that the internal high-level states of the editor are modeled by graphs where objects like windows, graphs, nodes, edges, etc. are presented by nodes and their relations to each other by arcs. The editor operations introduced are described by synchronization rules that induce parallel replacement on graphs showing internal states as just described.

5.2.1 The initial state

The *initial state* G_0 of the graph editor consists of an empty graph viewed in one initial window which is the current window because the cursor points to it. The cursor points always to the current graph and the current window and a window points always to that graph which it is showing.

G_0:

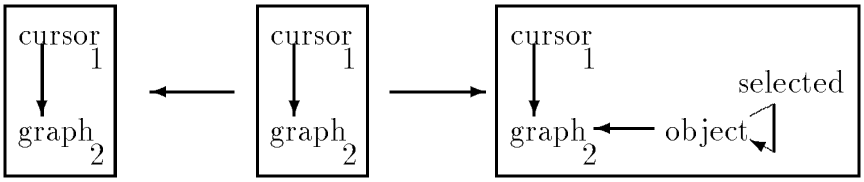


Arcs which describe relations between different items are usually not labeled except for loops. Here we have one loop labeled by "initial" describing the status of windows. Loops describe attributes of the object to which they belong. A window can be "initial" or "non-initial" if it is created later on.

5.2.2 Insertion and deletion of objects

Now a node (which is simply an object) can be inserted into the current graph. It is automatically selected, as indicated by the loop with label "selected".

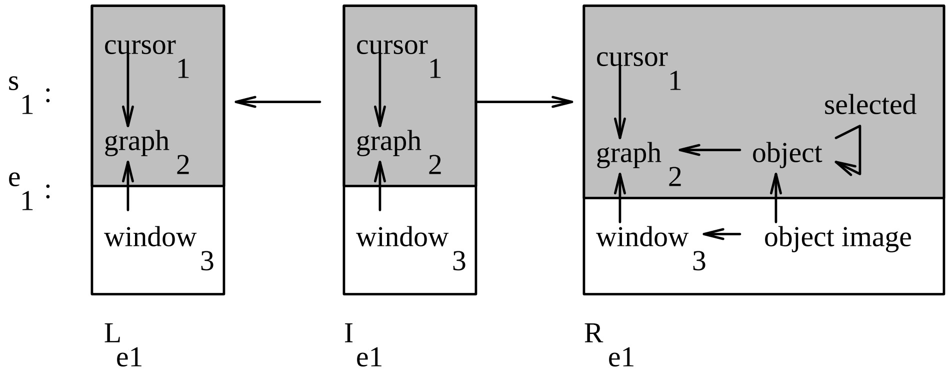
s_1:



L_{s1} I_{s1} R_{s1}

Numbers at the above nodes indicate which items are mapped to each other via graph morphisms $I_{s1} \longrightarrow L_{s1}$ and $I_{s1} \longrightarrow R_{s1}$. For this production and all following the indication is restricted to nodes.

A node insertion as described above implies that for each window that shows this graph a node image (object image) is created belonging to that window. Such an object image stores layout and position of the object in this particular window. However, this low level information is not shown on this level of abstraction.



Note that graphs L_{e1}, I_{e1} and R_{e1} of e_1 have L_{s1}, I_{s1} and R_{s1} in s_1 as subgraphs such that s_1 is a subproduction of e_1. The subgraphs are shown with a grey background in the above diagram. They are indicated in the same way in all following diagrams.

As shown above node insertion is described by subproduction s_1 and elementary production e_1. Thus, the corresponding synchronization rule, called $\mathcal{SR}_{INSNODE}$, is defined by

$$\mathcal{SR}_{INSNODE} = (\{e_1\}, \{s_1\}, \{inc_1 : s_1 \to e_1\}).$$

Since s_1 should be applied once and e_1 as often as there are windows we are interested in local-all-occurrence-coverings.

For each node insertion the object and all its object images are newly created. A node which is selected can be deleted by using the above productions in the inverse direction. This implies that the objects and object images are destroyed.

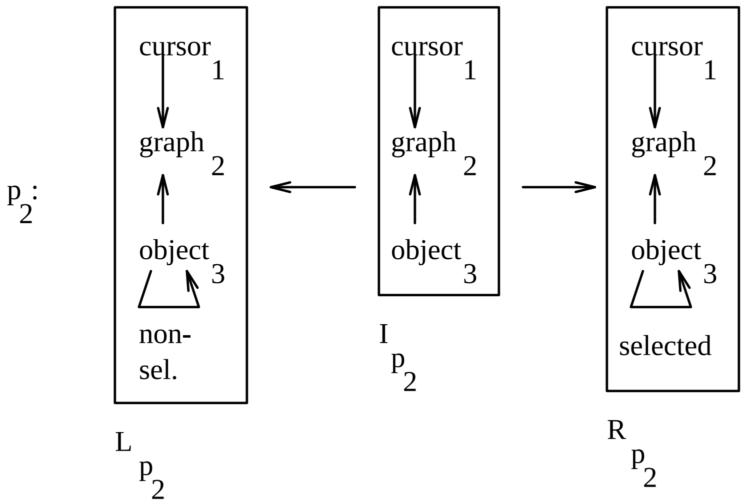
$$\mathcal{SR}_{DELNODE} = (\{e_1^{-1}\}, \{s_1^{-1}\}, \{inc_1^{-1} : s_1^{-1} \to e_1^{-1}\})$$

Insertion of arcs is similar to the insertion of nodes. The newly created arc (object) is related to that graph which comprises the two adjacent objects. For the distinction of source and target of the newly created arc corresponding objects have to be selected differently. Note that the selected objects can be nodes or arcs. Thus, this operation allows higher-order arcs. After insertion the new arc is selected as well as a new node after insertion.

Similarly to node insertion, an arc image (object image) has to be created for each window that shows this graph.

5.2.3 Selection and deselection of objects

Selection and deselection of an object is modeled by the following production p and the inverse production p^{-1}:



The corresponding basic synchronization rules are the following:

$$\mathcal{SR}_{SELECT} = (\{p_2\}, \emptyset, \emptyset)$$
$$\mathcal{SR}_{DESELECT} = (\{p_2^{-1}\}, \emptyset, \emptyset)$$

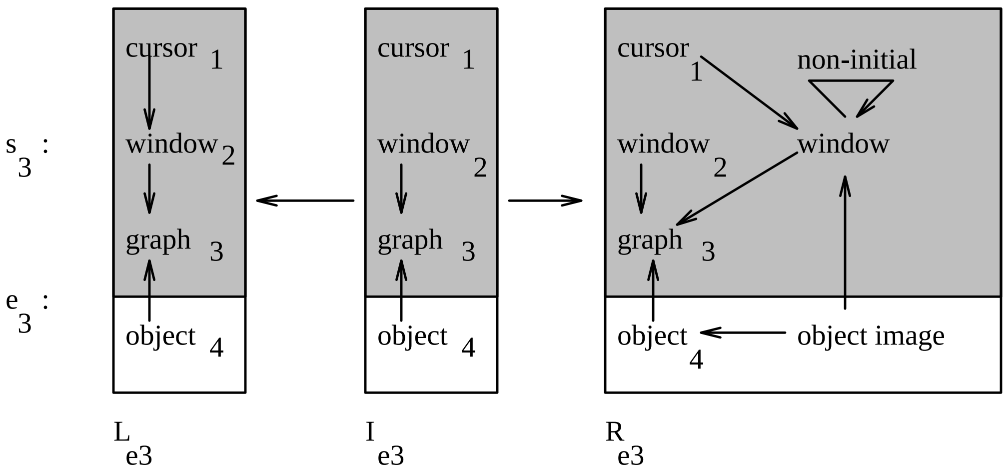
5.2.4 Creation of windows

The contents of the current window can be copied to a new non-initial window which is then the current one.

Such a window creation means that a new object image belonging to the new window is created for all objects (nodes and arcs) of the current graph.

Again s_3 is subproduction of e_3. The corresponding synchronization rule is the following:

$$\mathcal{SR}_{CREATEWIN} = (\{e_3, s_3\}, \{s_3\}, \{inc_3 : s_3 \to e_3, id_{s_3}\})$$



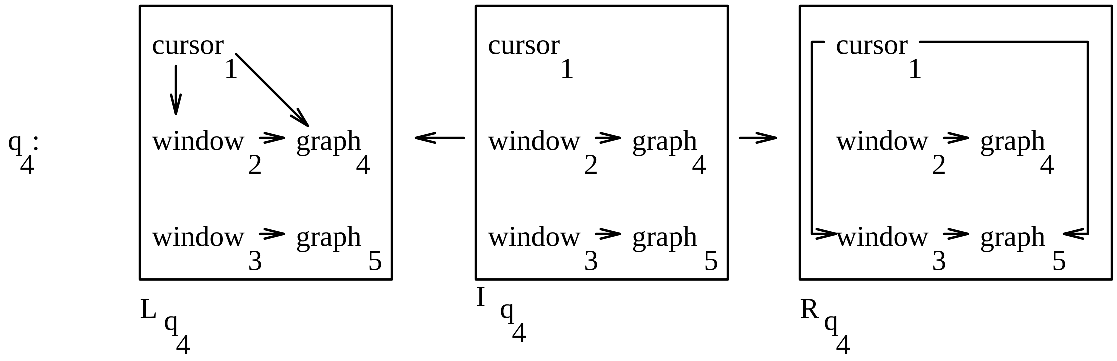
If there does not exist any occurrence of elementary production e_3 we look at least for an occurrence of the subproduction s_3. This is the case for example if a new window is created for a graph which is empty. Thus, similarly to insertion of objects local-all-occurrence-coverings are interesting.

Copy operations via selection are performed in a similar way. Deletion of a current window is only permitted if it is non-initial and implies the deletion of all object images belonging to it. The cursor is set to the initial window. This kind of operation is not explicitly shown here.

5.2.5 Change of the current window

The change of the current window can be performed by one production that changes the cursor position.

In order to use this production for changing between windows viewing the same graph both "graph"-nodes of the following left hand side of q have to be identified in the application of q.

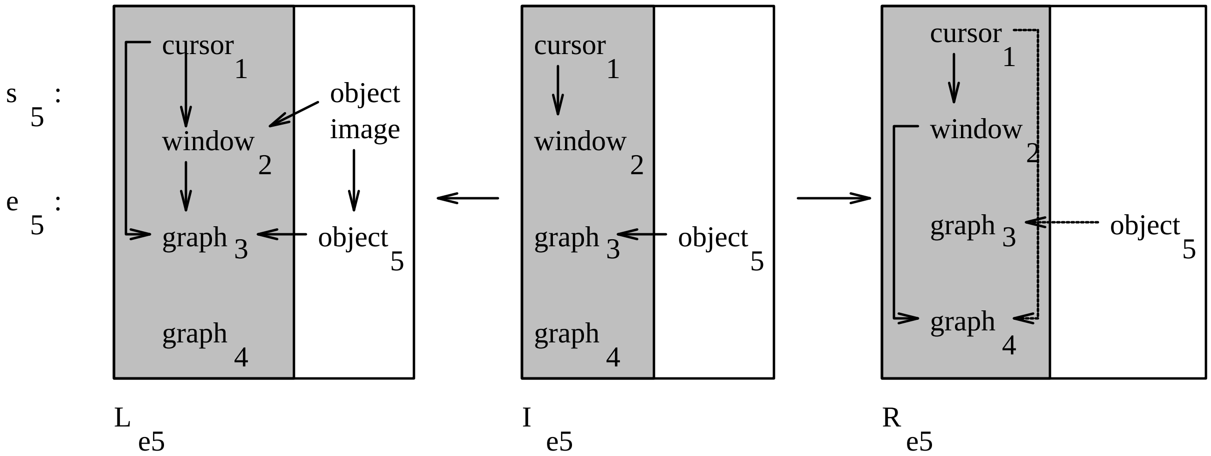


$$\mathcal{SR}_{CHANGECURWIN} = (\{q_4\}, \emptyset, \emptyset)$$

5.2.6 Change to another graph

The change of the current to another graph means that it is replaced in the current window by the other one.

This means that all object images are deleted which are images of objects of the current graph.



Additionally, all objects of the new graph have to be imaged in the current window. This is modeled by inverse production e_5^{-1}.

The production s_5 is subproduction of e_5 and e_5^{-1}. Both, the deletion of old object images and the creation of new object images have to be done, thus the synchronization rule $\mathcal{SR}_{CHANGEGRAPH}$ looks slightly more complex than the other ones.

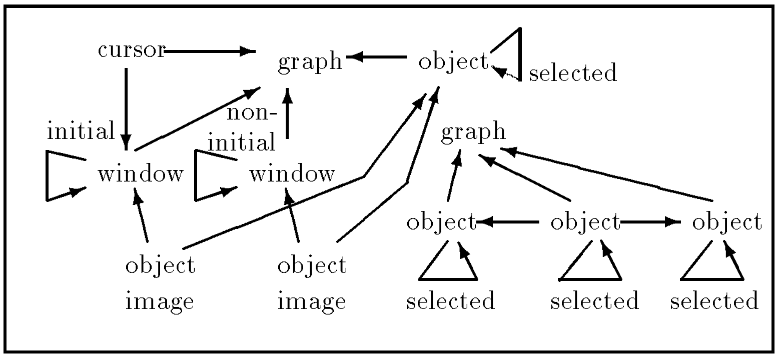
$$\mathcal{SR}_{CHANGEGRAPH} = (\{e_5, e_5^{-1}, s_5\}, \{s_5\}, \{inc_5 : s_5 \to e_5, inc_5^{-1} : s_5 \to e_5^{-1}, id_{s_5}\})$$

As before, local-all-occurrence-coverings specify the right operation.

5.2.7 Example: Deletion of a node – a contextual parallel derivation

Starting with the initial state G_0 of the editor, inserting two nodes with an arc in between them, creating a new graph with one node and a new window that shows this graph we get the following state G_1 of the editor.

G_1:

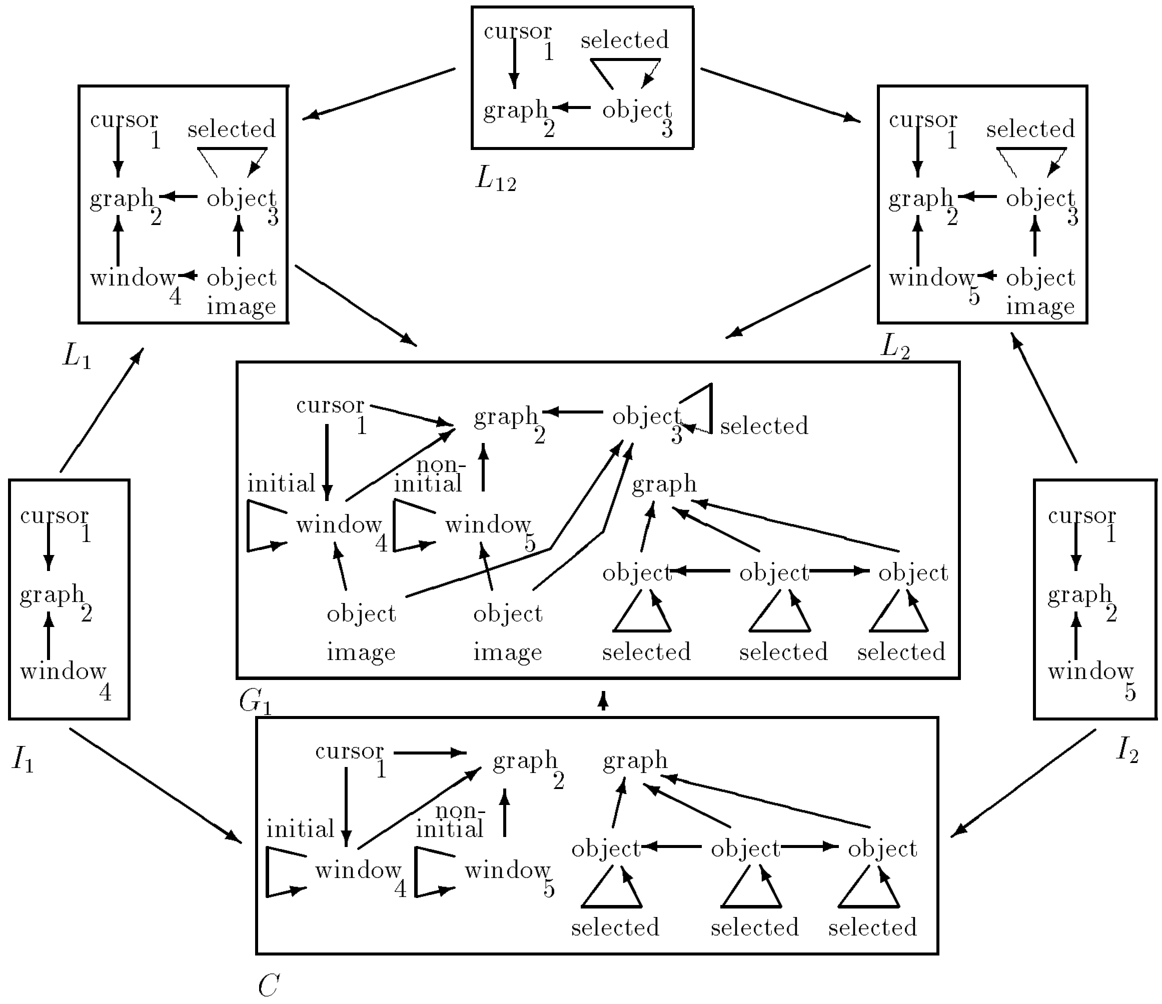


The deletion of the only node in the current graph of G_1 is now done with a contextual parallel derivation. The current graph is shown in both windows so there are two different occurrences of e_1^{-1} which overlap in an occurrence of their subproduction s_1^{-1}. The context graph C is generated by the POS-complement construction described in 3.3. Thus we get the pushout-star of G_1 depicted below.

The construction of a pushout-star and a POS-complement in category **GRAPHS** is outlined in the appendix. The context graph C is the corresponding POS-complement graph. Its construction is described in the appendix. Here, it is that subgraph of G_1 where the only object of the current graph is left out together with all its object images in the windows showing the current graph. The new state G_2 is obtained by the second pushout-star shown below.

Note that here a local-all-occurrence-covering is used. In this special case an all-occurrence-covering would yield the same result since there is only one occurrence of s_1^{-1} in G_1. For example, for the insertion of an arc between selected objects different results can be obtained. A derivation via a local-all-occurrence-covering would model the insertion of exactly one arc while that via an all-occurrence-covering would describe the insertion of several arcs.

Other operations mentioned above can also be modeled by contextual parallel derivations using local-all-occurrence-coverings, only.

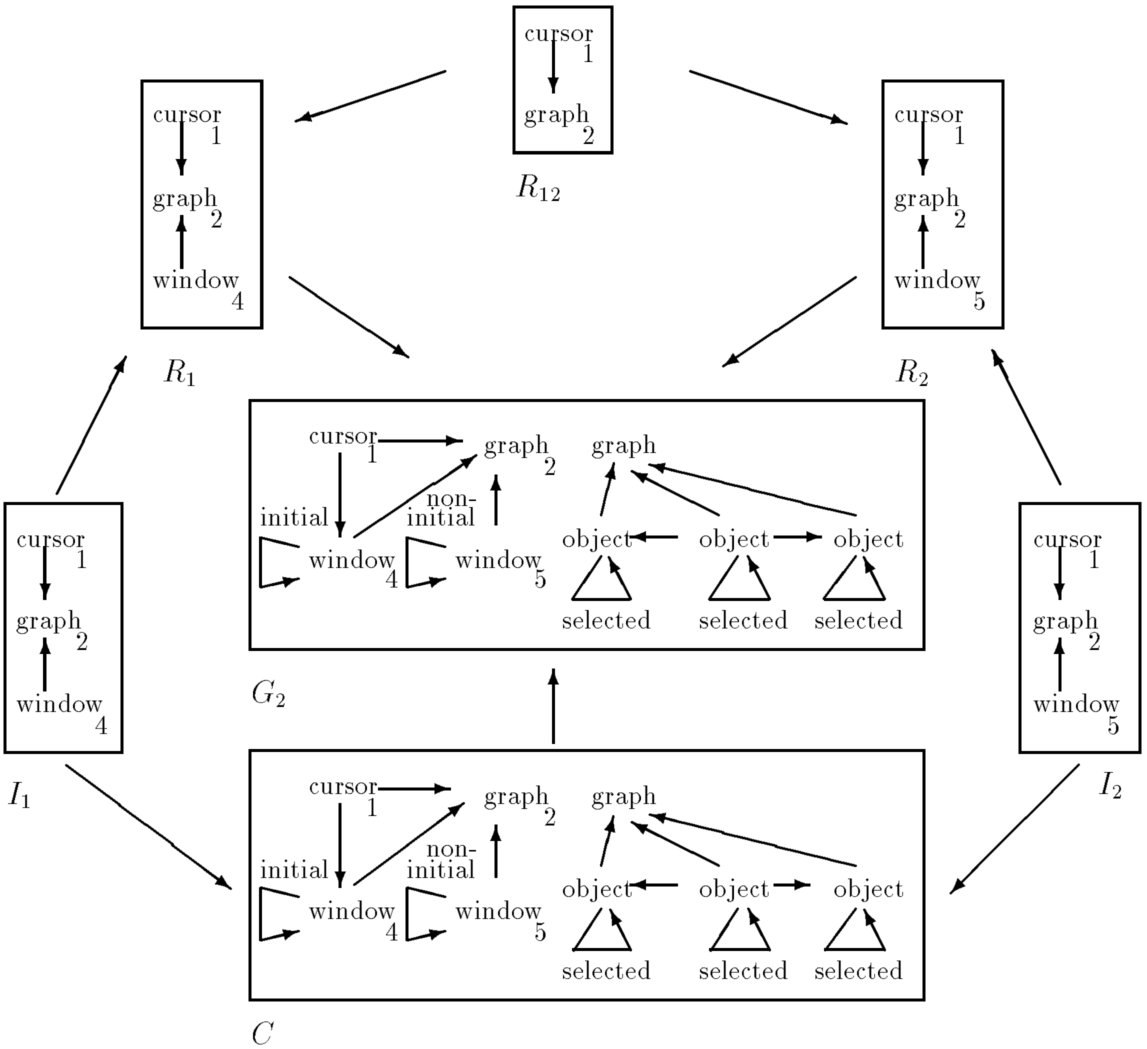


According to the Parallel Replacement Theorem II these operations can be described equivalently by amalgamated derivations and contextual parallel derivations. But there is only little advantage of using amalgamated derivations in this example since the amalgamated production constructed cannot be reused frequently.

Initial state G_0 of the editor together with the set of all synchronization rules introduced above form a PHLR-system using contextual parallel derivations. All objects $Obj(\mathbf{GRAPHS})$ of the category **GRAPHS** are allowed as terminal objects.

$\mathcal{L}^{locall}((G_0, \mathcal{R}, Obj(GRAPHS))_{con})$ with $\mathcal{R}$ being the set of all synchronization rules introduced above describes all possible editor states.

In [39] Taentzer and Beyer specified an abstraction concept on the basis given in this example. In that context, amalgamated derivations are used to describe corresponding editor operations.



Movement of objects would not cause any changes in the internal state of the editor introduced so far because it is only related to the layout of a graph. This is treated in the next example.

5.3 Movement of objects in configuration spaces

In this example the shape and position of arbitrary objects in two-dimensional configuration spaces are modeled (see also [38]). The objects are allowed to move via translations within this space if there is no collision with each other. Amalgamated derivations are used to model basic motions via a finite set of elementary productions. The example is based on category **CPOGRAPHS** of graphs with completely partially ordered label alphabet and label compatible graph morphisms. In [37] and [38] also rotation of objects based on parallel derivations is considered.

To model the layout of configuration spaces (which are, for example, windows that show graphs) by graphs a Cartesian grid is laid in the space and this space which is made discrete is mapped to a graph in the following way. Each grid square is modeled by a node. If there is a part of an object in the square then the node is black otherwise it is white. Horizontically or vertically neighboring squares are mapped to nodes which are connected by “h”-colored (horizontal) or “v”-colored (vertical) arcs, respectively.

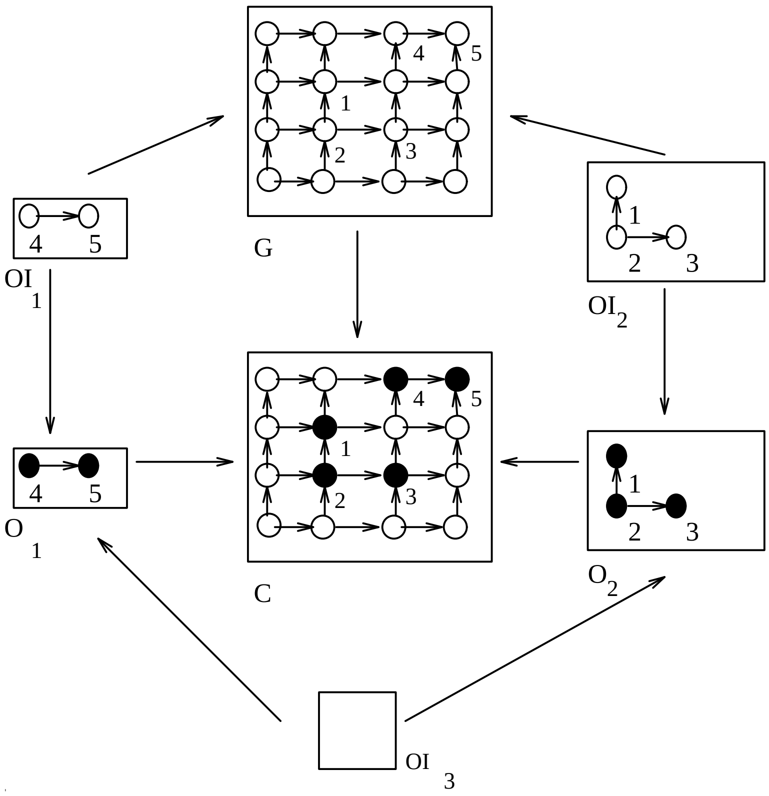
Thus, the node label alphabet consists of black ($\bullet$) and white ($\circ$) with the complete partial order

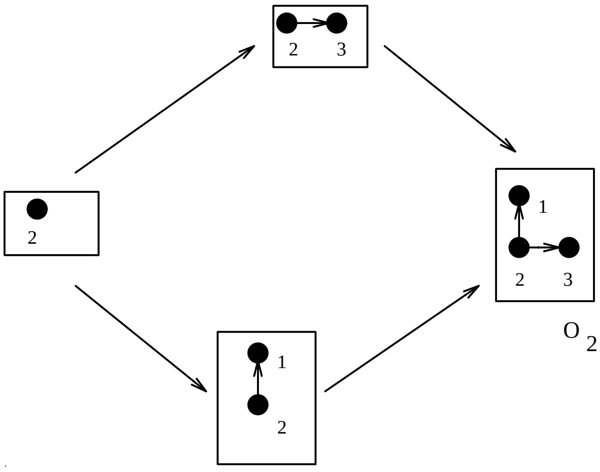
$$\circ \leq_N \circ \leq_N \bullet \leq_N \bullet$$

The arc label alphabet is $SC_A = \{h, v\}$. Actually, there is not any complete partial order of SC_A, however, it can be extended to $SC'_A = \{\lambda, h, v, \{h, v\}\}$ with $\lambda >_A h >_A \{h, v\}$ and $\lambda >_A v >_A \{h, v\}$. But label λ and $\{h, v\}$ will not be needed because each graph morphism used is color preserving on arc labels. On the node labels, the graph morphisms are allowed to change labels according to the partial order given above.

The configuration space is not modeled by only one graph but by a graph morphism star and its parallel gluing which yields the configuration space. Such a graph morphism star consists of a graph which models the configuration space without any object, a set of object graphs modeling arbitrary objects and interface graphs with corresponding graph morphisms used to determine the position of objects in the configuration space. All graphs have Cartesian grid like structures as described above. The parallel gluing of these graphs is described by a pushout-star in category **CPOGRAPHS** which differs from a pushout-star in **GRAPHS** in the construction of node labels obtained by greatest lower bounds.

The following configuration C consists of two objects which are non-overlapping. Object graphs O_1 and O_2 are connected graphs with more than one node where all nodes are black. The empty space is modeled by graph G where all nodes are white. Each configuration space where no two objects overlap is called consistent.

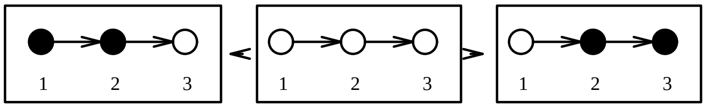




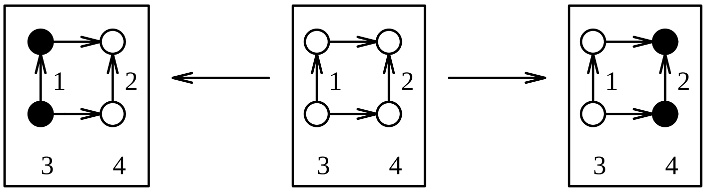
All "h"-colored ("v"-colored) arcs are drawn horizontically (vertically). Object interfaces OI_1 and OI_2 have the same structure than their object graphs O_1 and O_2 but all their nodes are colored white. Interface OI_3 is empty. The morphisms between these interface graphs and graph G determine the positions of the objects. Each object graph consisting of more than one node can also be described as a configuration consisting of all its handles (i.e. two nodes with one arc between them) and single nodes or empty graphs as interfaces. (See figure above.)

Basic motions like translations about one step to the right dependent on the fineness of the grid used are modeled by amalgamated derivations using all-occurrence coverings that are fully synchronized for a certain region with the following underlying synchronization rule: $\mathcal{SR}_{TRANS} = (\{e_1, e_2, s_1\}, \{s_1, s_2\}, \{inc_{11} : s_1 \to e_1, inc'_{11} : s_1 \to e_1, inc_{12} : s_1 \to e_2, inc'_{12} : s_1 \to e_2, inc_{21} : s_2 \to e_1, inc_{22} : s_2 \to e_2\})$ with

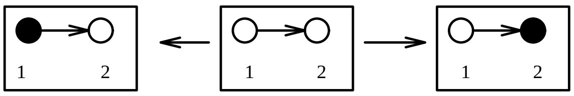
e_1 :



e_2 :



s_1 :

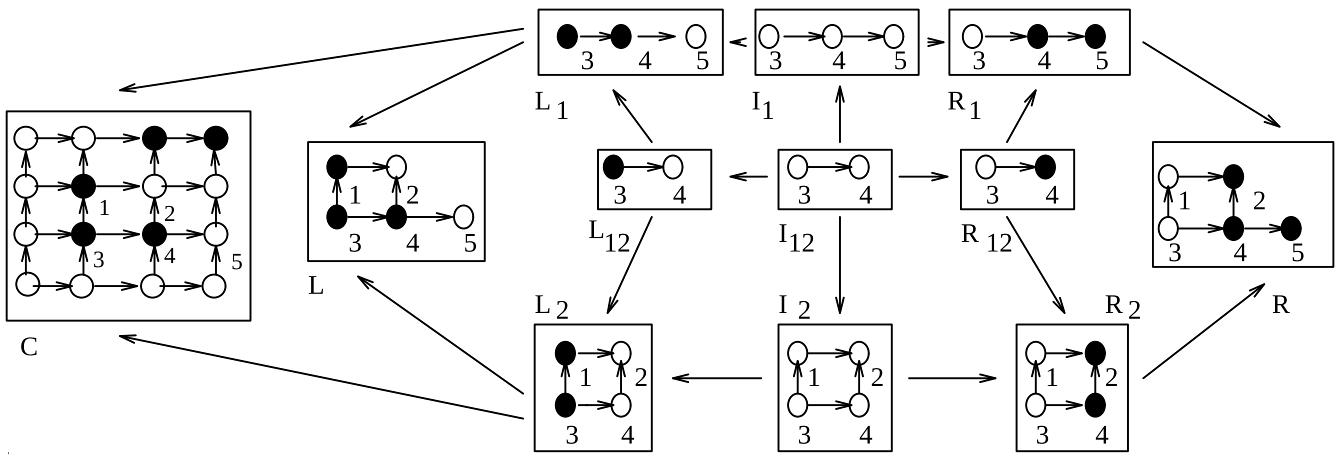


and $s_2 = (\emptyset \longleftarrow \emptyset \longrightarrow \emptyset)$. It is easy to check that s_1 and s_2 are subproductions of e_1 and e_2. Note that s_1 is subproduction of e_1 (e_2) in two different ways.

Translations into other directions can be described in a similar way.

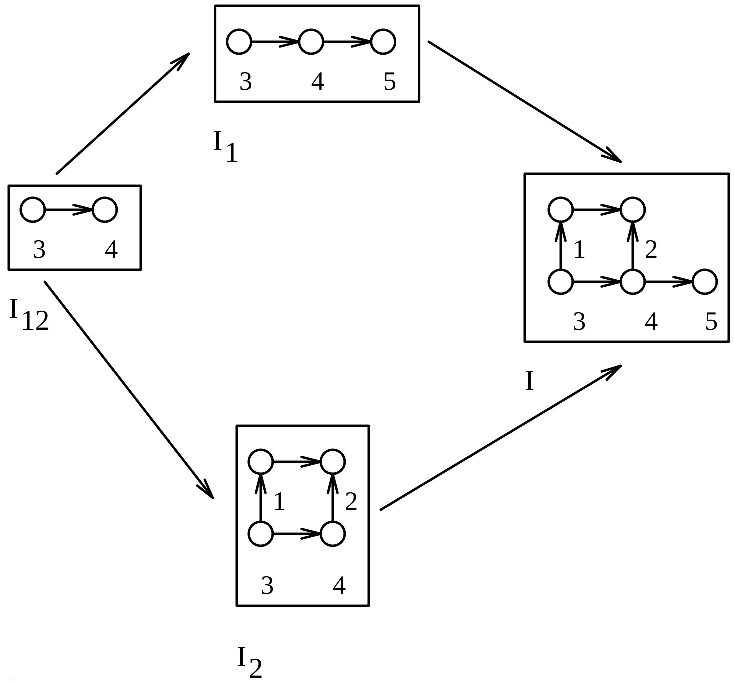
The partial covering is constructed by looking for all different occurrences of e_1 and e_2 for handles which belong to that object which has to be moved. If the occurrences of e_1 and e_2 overlap in a node their common subproduction is s_1 otherwise it is s_2. A partial covering constructed in such a way is in $\mathcal{COV}^{all}(C, \mathcal{SR}_{TRANS}) \cap \mathcal{COV}^{sync}(C, \mathcal{SR}_{TRANS}) \cap \mathcal{COV}^{reg}(C, \mathcal{SR}_{TRANS})$ where C is an arbitrary configuration space. Since s_1 and s_2 have non-isomorphic left hand sides all coverings constructed are unique according to lemma 3.11.

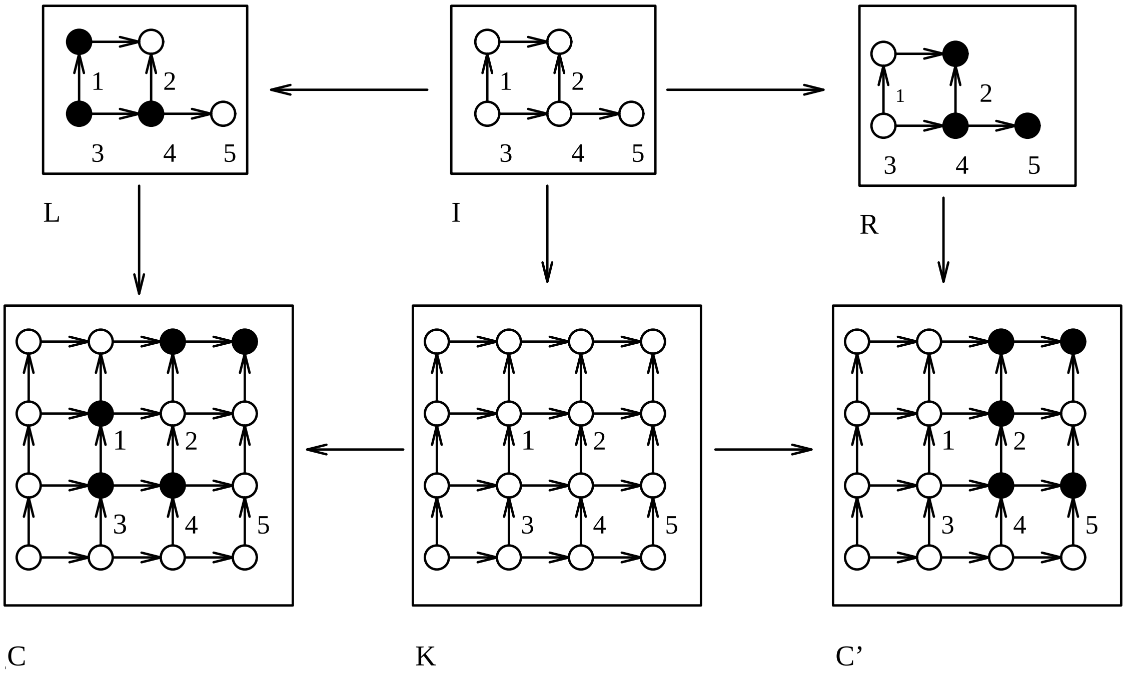
For example, object O_2 in configuration space C is moved one step to the right. Here, the following component production set $T = (\mathcal{P}, \mathcal{S}, E)$ with $\mathcal{P} = \{p_i | \ 1 \leq i \leq 2, p_1 = e_1, p_2 = e_2\}$ and $\mathcal{S} = \{s_{ij} | \ 1 \leq i < j \leq 2, s_{12} = s_1\}$ is constructed. Set E of subproduction embeddings consists of (e_{12}, f_{12}, g_{12}) and (e_{21}, f_{21}, g_{21}) which are the obvious embeddings of s_{12} into p_1 and p_2 such that there two occurrences $o_i : L_i \rightarrow C, \ \ 1 \leq i \leq 2$ with $o_i \circ e_{ij} = o_j \circ e_{ji}, \ \forall\, 1 \leq i < j \leq 2$.



The amalgamated production $p = (L \longleftarrow I \longrightarrow R)$ with $I = \mathrm{POS}(I_1 \longleftarrow I_{12} \longrightarrow I_2)_{1 \leq i < j \leq 2}$ models now the translation for an object like O_2, i.e. for an object with the particular shape of O_2.

A direct derivation via production p performs the translation of O_2 in configuration space C. This is possible if an occurrence of L in C can be found which is color preserving. Otherwise this would model a collision between objects.





In general, it is useful to describe the movement of objects by amalgamated derivations because it is expectable that the amalgamated production is frequently reused for moving the same object within the configuration space. But of course, due to the Parallel Replacement Theorem II also contextual parallel derivations can be used everywhere where amalgamated derivations perform movement of objects.

For translations about more than one step elementary productions or amalgamated productions have to be sequentially composed by constructing concurrent productions ([15], [10]).

Each configuration space together with synchronization rule $\mathcal{SR}_{TRANS}$ form a PHLR-system based on category **CPOGRAPH**. All objects T =Obj(**CPOGRAPHS**) are allowed as terminal objects. Coverings are constructed in the way described above. Let $PHLRS_a = (C, \{\mathcal{SR}_{TRANS}\}, T)_a$ be the corresponding PHLR-system. If the initial configuration space C is consistent all derivable configuration spaces in $\mathcal{L}^{all}(PHLRS_a) \cap \mathcal{L}^{sync}(PHLRS_a) \cap \mathcal{L}^{reg}(PHLRS_a)$ are consistent, too. (See [38] and [37] for the proof and more details.)

6 Conclusions

In this paper we have introduced parallel rewriting for HLR-systems. According to the Parallel Replacement Theorem I amalgamated derivations with constant interfaces can also be performed by direct derivations via parallel productions. This fact gives us a hint how such amalgamated derivations with constant interfaces can be sequentialized using the analysis construction of the Parallelism Theorem.

Moreover, amalgamated derivations combine concepts of parallel and sequential rewriting. Here, first a new production is generated by a complete parallel derivation and then this production is applied using a direct derivation. So, on one hand we have the advantage of using parallel replacements speeding up the derivation process, on the other hand all results which are known for direct derivations (like Church–Rosser–Properties, Parallelism Theorem and Concurrency Theorem, etc.) can be used also for this type of derivation if the corresponding conditions for HLR–systems (see [10]) are satisfied.

These properties give information about when derivations can be used in arbitrary order and moreover, in parallel. Thus, future work would be to describe the composition and decomposition of derivations in PHLR-systems under suitable conditions.

An amalgamated production can be generated from a set of elementary productions. So, the amalgamated derivations can also be used to describe derivations over production families. Dependent on the actual structure a suitable amalgamated production is generated from a given set of productions described by synchronization rules specifying an operation independent of particular structures.

The examples introduced use special constructions to get partial coverings. It remains open to consider other constructions in detail to get more information about uniqueness of covering constructions.

The different kinds of parallel replacement introduced in this paper can also be regarded as concepts to model some ideas of distribution. A global state object can be split into local structures which are transformed in parallel, each independent of the others. Each pair of local objects is combined by an interface which can be transformed, too. After the local transformations all objects obtained are joined again to a global state according to the interfaces obtained. All local transformations run concurrently without any synchronization. So, the only way to synchronize the local transformations is to switch to the global state. (See also [4], [13].) It would be an interesting task to combine the concepts of parallel rewriting in this paper with those of distribution ([29]) to get a nice abstract framework for the description of distributed systems. This work has been started by Taentzer in [41] for graph grammars in the single-pushout approach.

Acknowledgements

I thank Hans-Jörg Kreowski, Hartmut Ehrig, Ralph Betschko, Andrea Corradini, Reiko Heckel, Martin Korff and the referee for valuable comments on this and a preliminary version of this paper.

References

[1] J. Adamek, H. Herrlich, G. Strecker: Abstract and Concrete Categories, Wiley, 1990

[2] M.A. Arbib, E.G. Manes: Arrows, Structures and Functors, Academic Press, New York – San Francisco – London,1975

[3] D. A. Bailey, J. E. Cuny, C. D. Fisher: Programming with very large Graphs in Proc. 4th Int. Workshop on Graph Grammars and Their Application to Computer Science, LNCS 532, Springer Verlag, 1991

[4] H. Ehrig, P. Böhm, U. Hummert, M. Löwe: Distributed Parallelism of Graph Transformations, 13th Int. Workshop on Graph Theoretic Concepts in Computer Science, LNCS 314, Springer, 1988

[5] M. Beyer: Ein graphischer Editor für algebraische Graphgrammatiken - Konzeption, Objektorientierter Entwurf und Implementierung, Diplomarbeit, Fachbereich Informatik, TU Berlin, 1991

[6] P. Boehm, H.-R. Fonio, A. Habel: Amalgamation of Graph Transformations: A synchronization Mechanism in JCSS,34:377-408 (1987)

[7] V. Claus, H. Ehrig, G. Rozenberg: Graph Grammars and Their Applications to Computer Science and Biology in LNCS 73, Springer Verlag, 1979

[8] H. Ehrig: Introduction to the Algebraic Theory of Graph Grammars in LNCS 73, Springer Verlag, Berlin, 1979

[10] H. Ehrig, A. Habel, H.-J. Kreowski, F. Parisi-Presicce: Parallelism and Concurrency in High-Level Replacement Systems, Mathematical Structures in Comp. Sci. 1992, vol. 1, pp. 361 - 404

[11] H. Ehrig, H.-J. Kreowski: Parallel Graph Grammars in A. Lindenmayer, G. Rozenberg: Automata, Languages, Development, 425-447, Amsterdam: North Holland 1976

[12] H. Ehrig, H.-J. Kreowski, A. Maggiolo-Schettini, B. Rosen, J. Winkowski: Transformation of Structures: An Algebraic Approach, Math. Syst. Theory, 14 (1981)

[13] H. Ehrig, M. Loewe: Parallel and Distributed Derivations in the Single Pushout Approach, TCS (1993)

[14] H. Ehrig, B. Mahr: Fundamentals of Algebraic Specification 2: Module Specifications and Constraints, Springer New York – Heidelberg – Berlin 1990

[15] H. Ehrig, B. Rosen: Parallelism and Concurrency of Graph Manipulations, TCS 11 (1980), p. 247 - 275

[16] H. Ehrig, M. Pfender, H.-J. Schneider: Graph Grammars: An Algebraic Approach, Proc. IEEE Conf. SWAT73, Iowa City (1973), p. 167 - 180

[17] H. Ehrig, G. Taentzer: From Parallel Graph Grammars to Parallel High-Level Replacement Systems, Lindenmayer Systems, G. Rozenberg, 1992, Springer Verlag

[18] A. Habel: Hyperedge Replacement: Grammars and Languages, PhD Thesis, Univ. Bremen, 1989, LNCS 643, Springer, 1992

[19] U. Hummert: Algebraische Theorie von High-Level Netzen, PhD Thesis, TU Berlin, 1989

[20] H.-J. Kreowski: Parallel Hyperedge Replacement, in [34]

[21] M. Löwe, M. Beyer: AGG –An Implementation of Algebraic Graph Rewriting, LNCS 690, Springer, 1993, RTA'93

[22] M. Löwe: Extended Algebraic Graph Transformation, PhD thesis, Berlin, 1991

[23] S. MacLane: Categories for the working mathematician, Springer New York – Heidelberg – Berlin 1972

[24] J. Meseguer, U. Montanari: Petri nets are monoids: A New Algebraic Foundation for Net Theory, Proc. Logics in Comp.Sci., 1988

[25] D. Janssens, G. Rozenberg: On the structure of node–label controlled graph grammars in Information Science, 20:191 – 216,1980

[26] M. Nagl: Set Theoretic Approaches to Graph Grammars in Proc. 3rd Int. Workshop on Graph Grammars, Springer LNCS 291(1987),p.41–54

[27] A. Nakamura and K. Aizawa: On a Relationship between Graph L-systems and Picture Languages, TCS 24 (1983), 161 -177

[28] J. Padberg: Kolimeskonstruktionen in algebraischen Spezifikationssprachen, Studienarbeit, TU Berlin, 1991

[29] J. Padberg: Survey of High-Level Replacement Systems, FB 20,Technical Report 93/8, TU Berlin, 1993

[30] F. Parisi-Presicce, H. Ehrig, U. Montanari: Graph Rewriting with Unification and Composition, Proc. 3rd Int. Workshop on Graph Grammars, Springer LNCS 291 (1987), p. 496 - 511

[31] J. Padberg, H. Ehrig, L. Ribeiro: Algebraic High-Level Net-Transformation Systems, TU Berlin, Technical Report 93/12, accepted for MSCS

[32] F. Parisi-Presicce: Foundations of Rule-Based Design of Modular Systems, TCS 83, No. 1 (1991)

[33] J. Padberg, G. Taentzer: Embedding of Derivations in High-Level Replacement Systems, FB20, Technical Report 93/9, TU Berlin, 1993

[34] G. Rozenberg: An introduction to the NLC way of rewriting graphs, Proc. 3rd Int. Workshop on Graph Grammars and Their Applications to Computer Science, LNCS 291, 1987

[35] G. Rozenberg: Lindenmayer Systems, Springer Verlag, 1992

[36] G. Rozenberg, A. Salomaa: The Book of L, Springer Verlag, 1986

[37] H. Schween, G. Taentzer: Anwendungen und Erweiterungen der Theorie von Graphgrammatiken auf Bewegungsprobleme in der Robotik, Diplomarbeit, TU Berlin, 1990

[38] H. Schween, G. Taentzer: Movement of Objects in Configuration Spaces modelled by Graph Grammars in 4rd Int. Workshop on Graph Grammars and Their Applications to Computer Science, LNCS 532, Springer Verlag, 1991

[39] G. Taentzer, M. Beyer: Amalgamated Graph Transformations and Their Use for Specifying AGG – an Algebraic Graph Grammar System, LNCS 776, Springer, 1994

[40] G. Taentzer: Parallel High-Level Replacement Systems, Technical Report, TU Berlin, FB 20, 92/10, 1992

[41] G. Taentzer: Synchronous and Asynchronous Graph Transformations, submitted for a special issue of Fundamenta Informaticae devoted to graph transformations, 1994

7 Appendix: Concepts of category theory

7.1 Definition *A* **category** $\mathcal{C}$ *consists of a class Obj($\mathcal{C}$) of objects and a set of morphisms* $\mathcal{MOR}(A, B)$ *for each pair A, B of Obj($\mathcal{C}$) with* $dom(f) = A$ *and* $cod(f) = B$ *for* $f \in \mathcal{MOR}(A, B)$ *such that*

- *there is a composition morphism* $g \circ f : A \longrightarrow C$ *for each pair of morphisms* $f : A \longrightarrow B$ *and* $g : B \longrightarrow C$ *if A, B and C are objects of* $\mathcal{C}$.
- *the composition is associative, i.e. for any three morphisms* $f : A \longrightarrow B$, $g : B \longrightarrow C$ *and* $h : C \longrightarrow D$ *we have* $h \circ (g \circ f) = (h \circ g) \circ f$ *if A, B, C and D are objects of* $\mathcal{C}$.
- *For each object A of* $\mathcal{C}$ *there is a morphism* $id_A : A \longrightarrow A$, *called the identity of A. The identity is neutral, i.e. for any object B of* $\mathcal{C}$*and any morphisms* $f : A \longrightarrow B$ *and* $g : B \longrightarrow A$ *we have* $f \circ id_A = f$ *and* $id_a \circ g = g$.

A category $\mathcal{C}$ is called small if the object class of $\mathcal{C}$ is a set.

7.2 Example

1. *Let* C_A *and* C_N *be fixed color alphabets for arcs and nodes. A graph*

$$G = (G_A, G_N, s_G, t_G, m_{GA}, m_{GN})$$

   consists of the sets of arcs and nodes, G_A *and* G_N, *mappings* $s_G, t_G : G_A \longrightarrow G_N$, *called source resp. target map, and the coloring maps for arcs and nodes,* $m_{GA} : G_A \longrightarrow C_A$ *and* $m_{GN} : G_N \longrightarrow C_N$.

2. *Given two graphs G and G′ a graph morphism* $f : G \longrightarrow G'$ *is a pair of maps* $(f_A : G_A \longrightarrow G'_A, f_N : G_N \longrightarrow G'_N)$ *such that*
   - $f_N \circ s_G = s_{G'} \circ f_A$, $f_N \circ t_G = t_{G'} \circ f_A$ *(structure preserving)*
   - $m_{G'A} \circ f_A = m_{GA}, m_{G'N} \circ f_N = m_{GN}$ *(label preserving)*

3. *Category* **GRAPHS** *is obtained by taking graphs as objects and graph morphisms as arrows. The identity and composition of morphisms is defined componentwise for nodes and arcs.*

7.3 Definition *A morphism* $f : A \to B$ *is called* **isomorphism** *if there is a morphism* $f^{-1} : B \to A$ *such that* $f^{-1} \circ f = id_A$ *and* $f \circ f^{-1} = id_B$. *a and B are called* isomorphic, *written* $A \cong B$, *if there is an isomorphism* $f : A \to B$.

7.4 Definition *A* **functor** $F : \mathcal{A} \longrightarrow \mathcal{B}$ *is a function between two categories* $\mathcal{A}$ *and* $\mathcal{B}$ *which maps each object A of* $\mathcal{A}$ *onto an object F(A) of* $\mathcal{B}$ *and each morphism* $f : A \longrightarrow A'$ *of* $\mathcal{A}$ *onto a morphism* $F(f) : F(A) \longrightarrow F(A')$ *of* $\mathcal{B}$ *such that*

- $F(id_A) = id_{F(A)}$ *for each object A od* $\mathcal{A}$ *and*
- $F(g \circ f) = F(g) \circ F(f)$ *whenever* $g \circ f$ *is defined in in* $\mathcal{A}$.

7.5 Definition *A* **diagram** *in a category $\mathcal{C}$ is a functor $\triangle : \mathcal{I} \longrightarrow \mathcal{C}$ from a small category $\mathcal{I}$ which is called* **diagramscheme** *to $\mathcal{C}$.*

7.6 Definition *Given categories $\mathcal{C}_1, ..., \mathcal{C}_n$ the* **product category** *$\mathcal{C}_1 \times ... \times \mathcal{C}_n$ is a category whose objects $(C_1, ..., C_n)$ are tupel of objects $C_1, ..., C_n$ and where the morphisms $(f_1, ..., f_n)$ are tupel of morphisms $f_1, ..., f_n$ of the given categories.*

7.7 Definition *Given the following diagramschemes PO and POS_n:*

$$PO: \quad \begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ b & & c \end{tikzcd} \qquad\qquad POS_n: \quad \begin{tikzcd} & ij \arrow[dl] \arrow[dr] & \\ i & & j \end{tikzcd} \qquad \forall\, 1 \leq i < j \leq n$$

$PO \times POS_n$ is the product of these diagramschemes.

7.8 Definition *Given two functors $F, G : \mathcal{A} \longrightarrow \mathcal{B}$ a* **natural transformation** *is a tripel (F, η, G) such that*

- *there exists a morphism $\eta(A) : F(A) \longrightarrow G(A)$ in $\mathcal{B}$ for all objects A in $\mathcal{A}$ and*
- *the following diagram commutes.*

$$\begin{tikzcd} F(A) \arrow[r, "\eta(A)"] \arrow[d, "F(f)"'] \arrow[dr, phantom, "(=)"] & G(A) \arrow[d, "G(f)"] \\ F(A') \arrow[r, "\eta(A')"'] & G(A') \end{tikzcd}$$

7.9 Definition *Let $\mathcal{A}$ and $\mathcal{B}$ be categories. The* **functor category** *$\mathcal{B}^{\mathcal{A}}$ is a category whose objects are functors $F : \mathcal{A} \longrightarrow \mathcal{B}$ and whose morphisms are natural transformations (F_1, η, F_2) for $F1, F2 : \mathcal{A} \longrightarrow \mathcal{B}$.*

7.10 Definition *An object J of $Obj(C)$ is* **initial** *if for any object $A \in Obj(C)$ there is a unique morphism $J \longrightarrow A$.*

7.11 Definition *The* **colimit** *(CO, c) of a diagram $\triangle : \mathcal{I} \longrightarrow \mathcal{C}$ consists of an object CO in $\mathcal{C}$ and a family $c = (c_I)$ of morphisms $c_I : \triangle(I) \longrightarrow CO$ for all I in $\mathcal{I}$ such that the following conditions are satisfied:*

- *Commutativity: $c_{I2} \circ \triangle(i) = c_{I1}$ for all $i : I1 \longrightarrow I2$ of $\mathcal{I}$.*
- *Universal property: For any object CO' in $\mathcal{C}$ and family $c' = (c'_I)$ of morphisms $c'_I : \triangle(I) \longrightarrow CO'$ for all I in $\mathcal{I}$ with $c'_{I2} \circ \triangle(i) = c'_{I1}$ for all $i : I1 \longrightarrow I2$ of $\mathcal{I}$ there is a unique morphism $f : CO \longrightarrow CO'$ in $\mathcal{C}$ such that we have $f \circ c_I = c'_I$.*

7.12 Definition *If there exists a colimit in the category $\mathcal{C}$ for all diagrams $\triangle : \mathcal{I} \longrightarrow \mathcal{C}$ the category is called $\mathcal{I}$-cocomplete (or the categroy $\mathcal{C}$ has $\mathcal{I}$-colimits). If $\mathcal{C}$ is $\mathcal{I}$-cocomplete for all small categories $\mathcal{I}$ it is called cocomplete (or it has colimits).*

7.13 Theorem (Cocompletness)
If a category **CAT** *has an initial object and pushouts it is* **finitely cocomplete**.

(See [2] or [1] for the proof of this theorem.)

Special colimits are, for example, finite coproducts, pushouts or pushoutstars:

7.14 Definition *A morphism star of degree $n \geq 1$ is a diagram $MS = (S_i \xleftarrow{s_{ij}} S_{ij} \xrightarrow{s_{ji}} S_j)_{1 \leq i < j \leq n}$ consisting of objects S_i,S_j,I_{ij} and morphisms s_{ij} resp. s_{ji}. In the case $n = 1$ we only have the single object S_1. An object S together with morphisms $s_i : S_i \longrightarrow S$ for $1 \leq i \leq n$ is called* **pushout–star** *of MS if we have:*

1. *commutativity: $s_i \circ s_{ij} = s_j \circ s_{ji}$ for all $1 \leq i < j \leq n$*
2. *universal property: For all objects S' and morphisms $s'_i : S_i \longrightarrow S'$ satisfying $s'_i \circ s_{ij} = s'_j \circ s_{ji}$ for all $i < j$ there is a unique morphism $\psi : S \longrightarrow S'$ such that $\psi \circ s_i = s'_i$ for all $1 \leq i \leq n$.*

$$\begin{tikzcd}
 & S_i \arrow[dr, "s_i"] \arrow[drr, "s'_i"] & & \\
I_{ij} \arrow[ur, "s_{ij}"] \arrow[dr, "s_{ji}"'] & (POS) & S \arrow[r, dashed, "u"] & S' \\
 & S_j \arrow[ur, "s_j"'] \arrow[urr, "s'_j"'] & &
\end{tikzcd}$$

Remarks:

1. In the definition above MS can be denoted as the morphism star of S: $MS = \mathrm{STAR}(S)$.
2. Object S without morphisms is called pushout-star object, short POS-object, of MS, written as $S = POS(MS)$.
3. If $I_{ij} = J$, $\forall\, 1 \leq i < j \leq n$, the pushout-star is called finite coproduct. Its object is denoted by $\sum_1^n S_i$.
4. In the case $n = 2$ the pushout–star is called pushout. So, all properties of pushout–stars are also valid for pushouts.

7.15 Example *A pushout-star in the category* **GRAPHS** *is obtained by the following construction: Let $\bar{\mathcal{I}}_S$ be the equivalence relation generated by the relation $\mathcal{I}_S = \{(s_{ij}(x), s_{ji}(x)) \mid x \in S_{ij},\ 1 \leq i < j \leq n\}$. Then POS-graph S is defined by $S = (\coprod_{1 \leq i \leq n} S_i)_{/\bar{\mathcal{I}}_S}$, the quotient set of the disjoint union of all S_i. Functions $s_i : S_i \longrightarrow S$ for all $1 \leq i \leq n$ in the diagram send each element of S_i to its equivalence class in S. (See [8] and [11] for details of pushout-star constructions.)*

7.16 Definition *Given a morphism star* $(S_i \xleftarrow{s_{ij}} S_{ij} \xrightarrow{s_{ji}} S_j)_{1 \leq i < j \leq n}$ *and morphisms* $s_{i(n+1)} : S_{i(n+1)} \longrightarrow S_i$ *and* $s_i : S_i \longrightarrow S$, $\forall\, 1 \leq i \leq n$, *object* S_{n+1} *together with morphisms* $s_{(n+1)i} : S_{i(n+1)} \longrightarrow S_{n+1}$, $\forall\, 1 \leq i \leq n$, *and* $s_{n+1} : S_{n+1} \longrightarrow S$ *is called* **pushout-star complement**, *short POS-complement, if* S *is POS-object of* $(S_i \xleftarrow{s_{ij}} S_{ij} \xrightarrow{s_{ji}} S_j)_{1 \leq i < j \leq n+1}$.

In the case $n = 1$ a pushout-star complement is called pushout complement.

7.17 Example *A pushout-star complement in category* **GRAPHS** *is, roughly described, a graph* $S_{n+1} = S - \bigcup_{1 \leq i \leq n} s_i(S_i) + \bigcup_{1 \leq i \leq n} s_i(s_{i(n+1)}(S_{i(n+1)}))$ *where* $s_{(n+1)i}$ *is the restriction of* $s_i \circ s_{(n+1)i}$ *to* S_{n+1}, $\forall\, 1 \leq i \leq n$.

7.18 Definition *Given a category* $\mathcal{C}$ *which has colimits the* **colimit functor** $COLIM_{\mathcal{I}} : \mathcal{C}_{\mathcal{I}} \longrightarrow \mathcal{C}$ *maps each diagram* $\triangle : \mathcal{I} \longrightarrow \mathcal{C}$ *of the functor category* $\mathcal{C}_{\mathcal{I}}$ *to its corresponding colimit* $(CO^{\triangle}, c^{\triangle})$ *in* $\mathcal{C}$ *such that*

- $COLIM_I(\triangle) = CO^{\triangle}$ *for all diagrams* $\triangle$ *of* $\mathcal{C}_{\mathcal{I}}$ *and*
- *the following diagram commutes for all natural transformations* $(\triangle, \eta, \triangle')$ *with* $\eta = (\eta_I)$ *for all objects* I *of* $\mathcal{I}$.

$$\begin{tikzcd}
\Delta(I) \arrow[r, "c_I^{\Delta}"] \arrow[d, "\eta_I"'] \arrow[dr, phantom, "(=)"] & COLIM_I(\Delta(I)) \arrow[d, "COLIM_I(\eta)"] \\
\Delta'(I) \arrow[r, "c_I^{\Delta'}"'] & COLIM_I(\Delta'(I))
\end{tikzcd}$$

7.19 Definition *Given a functor* $F : \mathcal{A} \times \mathcal{B} \longrightarrow \mathcal{C}$ *between a product category* $\mathcal{A} \times \mathcal{B}$ *and a category* $\mathcal{C}$ *we have*

- *a* **right associated functor** $F(A, -) : \mathcal{B} \longrightarrow \mathcal{C}$ *for each object* A *of* $\mathcal{A}$ *such that* $F(A,-)(B) = F(A,B)$ *for each object* B *of* $\mathcal{B}$ *and* $F(A,-)(h) = F(id_A, h)$ *for each morphism* h *of* $\mathcal{B}$.
- *a* **left associated functor** $F(-, B) : \mathcal{A} \longrightarrow \mathcal{C}$ *for each object* B *of* $\mathcal{B}$ *such that* $F(-,B)(A) = F(A,B)$ *for each object* A *of* $\mathcal{A}$ *and* $F(-,B)(g) = F(g, id_B)$ *for each morphism* g *of* $\mathcal{B}$.
- $F^{\mathcal{A}} : \mathcal{A} \longrightarrow \mathcal{C}^{\mathcal{B}}$ *is a functor which maps each object* A *of* $\mathcal{A}$ *to the right associated functor* $F(A,-) : \mathcal{B} \longrightarrow \mathcal{C}$ *and each morphism* $f : A \longrightarrow A'$ *of* $\mathcal{A}$ *to the natural transformation* $(F(A,-), F(f,-), F(A',-))$.
- $F^{\mathcal{B}} : \mathcal{A} \longrightarrow \mathcal{C}^{\mathcal{B}}$ *is defined analogously to* $F^{\mathcal{A}}$.

These functors have the same functionality so we have for all objects A of $\mathcal{A}$ and B of $\mathcal{B}$

$$F^{\mathcal{A}}(A)(B) = F(A,-)(B) = F(A,B) = F(-,B)(A) = F^{\mathcal{B}}(B)(A)$$

7.20 Theorem (Commutativity of colimits)

Given a $\mathcal{I}$- and $\mathcal{J}$-cocomplete category $\mathcal{C}$ it is also $\mathcal{I} \times \mathcal{J}$-cocomplete and we have

$$COLIM_{\mathcal{I}}(COLIM_{\mathcal{J}}(\triangle^{\mathcal{I}})) = COLIM_{\mathcal{I}\times\mathcal{J}}(\triangle) = COLIM_{\mathcal{J}}(COLIM_{\mathcal{I}}(\triangle^{\mathcal{J}}))$$

for any diagram $\triangle : \mathcal{I} \times \mathcal{J} \longrightarrow \mathcal{C}$.

See [28] for an explicit proof of this theorem.