Basic Research in Computer Science

BRICS RS-03-31 Lack & Sobociński: Adhesive Categories

Adhesive Categories

Stephen Lack
Paweł Sobociński

BRICS Report Series **RS-03-31**

ISSN 0909-0878 **October 2003**

**Copyright © 2003, Stephen Lack & Paweł Sobociński.
BRICS, Department of Computer Science
University of Aarhus. All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

**See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:**

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/03/31/`

Adhesive Categories

Stephen Lack
School of Quantitative Methods and Mathematical Sciences,
University of Western Sydney

Paweł Sobociński
BRICS,* University of Aarhus

Abstract

We introduce adhesive categories, which are categories with structure ensuring that pushouts along monomorphisms are well-behaved. Many types of graphical structures used in computer science are shown to be examples of adhesive categories. Double-pushout graph rewriting generalises well to rewriting on arbitrary adhesive categories.

Introduction

Recently there has been renewed interest in reasoning using graphical methods, particularly within the fields of mobility and distributed computing [13, 23] as well as applications of semantic techniques in molecular biology [7, 4, 15]. Research has also progressed on specific graphical models of computation [21, 16]. As the number of various models grows, it is important to understand the basic underlying principles of computation on graphical structures. Indeed, a solid understanding of the foundations of a general class of models (provided by *adhesive categories*, introduced in this paper), together with a collection of general semantic techniques (for example [26, 25]) will provide practitioners and theoreticians alike with a toolbox of standard techniques with which to construct the models, define the semantics and derive proof-methods for reasoning about these.

*Basic Research in Computer Science (`www.brics.dk`), funded by the Danish National Research Foundation.

Category theory provides uniform proofs and constructions across a wide range of models. The usual approach is to find a natural class of categories with the right structure to support the range of constructions particular to the application area. A well-known example is the class of cartesian-closed categories, which provides models for simply typed lambda calculi [19].

One important general construction which appears in much of the literature on graphical structures in computer science is the pushout construction. Sometimes referred to as generalised union [9], it can often be thought of as the construction of a larger structure from two smaller structures by gluing them together along a shared substructure. Recently, pushouts have been used in semantics in order to derive well-behaved labelled transition systems [20].

In this paper we introduce adhesive categories, which should be thought of as categories in which pushouts along monomorphisms are "well-behaved", where the paradigm for behaviour is given by the category of sets. An example of the good behaviour of these pushouts is that they are stable under pullback (the dual notion to pushout, which intuitively can often be thought of as a "generalised intersection"). The idea is analogous to that of extensive categories [3], which have well-behaved coproducts in a similar sense. Since coproducts can be obtained with pushouts and an initial object, and an initial object is "well-behaved" if it is strict, one might expect that adhesive categories with a strict initial object would be extensive, and this indeed turns out to be the case.

Various notions of graphical structures used in computer science form adhesive categories. This includes ordinary directed graphs, typed graphs [1] and ranked graphs [13], amongst others. The structure of adhesive category allows us to derive useful properties. For instance, the union of two subobjects is calculated as the pushout over their intersection, which corresponds well with the intuition of pushout as generalised union.

The properties of adhesive categories allow the development of a general notion of rewriting on such categories. Term rewriting has been a cornerstone of computation since the lambda-calculus; and indeed term rewriting can still be considered as an underlying computational mechanism in modern foundational process calculi for concurrency and mobility [22, 5]. Thus it is reasonable to expect a suitable notion of *graph* rewriting if one replaces syntax by directed graphs. One notion, that of double-pushout (d-p) graph rewriting, was introduced in the early seventies [12] and the field is now mature [6]. D-p graph rewriting is especially interesting since it works well together with a contextual/modular view of graphs as arrows in a cospan bicategory [14].

In D-p rewriting, a rewrite rule is given as a span $L \leftarrow K \rightarrow R$. Roughly,

the intuition is that L forms the left-hand side of the rewrite rule, R forms the right-hand side and K, common to both L and R, is the sub-structure to be unchanged as the rule is applied. To apply the rule to a structure C, one first needs to find a match $L \to C$ of L within C. The rule is then applied by constructing the missing parts (E, D and arrows) of the following diagram

$$\begin{array}{ccccc} L & \leftarrow & K & \rightarrow & R \\ \downarrow & & \downarrow & & \downarrow \\ C & \leftarrow & E & \rightarrow & D \end{array}$$

in a way which ensures that the two squares are pushout diagrams. Once such a diagram is constructed we may deduce that $C \Longrightarrow D$, that is, C rewrites to D.

D-p rewriting is formulated in categorical terms and is therefore portable to structures other than directed graphs. There have been several attempts [11, 9] to isolate classes of categories in which one can perform d-p rewriting and in which one can develop the rewriting theory to a satisfactory level. In particular, several axioms were put forward in [11] in order to prove a local Church-Rosser theorem for such general rewrite systems. Additional axioms were needed to prove a general version of the so-called concurrency theorem [17].

We define *adhesive grammars* which are d-p rewrite systems on adhesive categories. We show that the resulting rewriting theory is satisfactory by proving the local Church-Rosser theorem and the concurrency theorem without the need for extra axioms. Indeed, we shall argue that adhesive categories provide a natural general setting for d-p rewriting. We also examine how adhesive categories fit within the previously conceived general frameworks for rewriting [11, 9]. Many of the axioms put forward in [11] follow elegantly as lemmas from the axioms of adhesive categories.

Adhesive categories, therefore, provide a satisfactory model in which to define a theory of rewriting on "graph-like" structures. They are mathematically elegant and arguably less ad-hoc than previous approaches. We firmly believe that they will prove useful in the development of further theory in the area of semantics of graph-based computation, and in particular, in the development of a contextual theory of graph rewriting.

Structure of the paper. In §1 we recall the definition of extensive categories. The notion of van Kampen (VK) square is given in §2. VK squares are central in the definition of adhesive categories which are introduced in §3. In §4 we state and prove some basic lemmas which hold in any adhesive category. We also show that the subobjects of an object in an adhesive category form a distributive lattice, with the union of two subobjects constructed as the pushout

over their intersection. We develop double-pushout rewriting theory in adhesive categories in §5 and offer a comparison with High-Level Replacement Systems in §6. We conclude in §7 with directions for future research.

1 Extensive categories

Throughout the paper we assume that the reader is familiar with basic concepts of category theory. In this section we recall briefly the notion of extensive category [3].

Definition 1.1. A category **C** is said to be *extensive* when

(i) it has finite coproducts

(ii) it has pullbacks along coproduct injections

(iii) given a diagram where the bottom row is a coproduct diagram

$$\begin{array}{ccccc} X & \xrightarrow{m} & Z & \xleftarrow{n} & Y \\ {\scriptstyle r}\downarrow & & \downarrow{\scriptstyle h} & & \downarrow{\scriptstyle s} \\ A & \xrightarrow[i]{} & A+B & \xleftarrow[j]{} & B \end{array}$$

the two squares are pullbacks if and only if the top row is a coproduct.

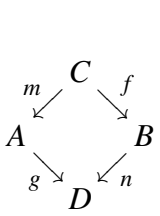
The third axiom states what we mean when we say that the coproduct $A+B$ is "well-behaved": it includes the fact that coproducts are stable under pullback, and it implies that coproducts are disjoint (the pullback of the coproduct injections is initial) and that initial objects are strict. It also implies a cancellativity property of coproducts: given an isomorphism $A+B \cong A+C$ compatible with the injections, one can construct an isomorphism $B \cong C$. For an object Z of an extensive category, the lattice $\mathrm{Sub}(Z)$ of coproduct summands of Z is a Boolean algebra.

2 Van Kampen squares

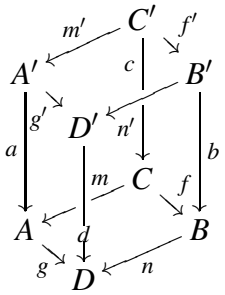
The definition of adhesive category is stated in terms of something called a *van Kampen square*, which can be thought of as a "well-behaved pushout", in a similar way to which coproducts can be thought of as "well-behaved" in an extensive category; essentially this means that they behave as they do in the category of sets.

The name van Kampen derives from the relationship between these squares and the van Kampen theorem in topology, in its "coverings version", as presented for example in [2]. This relationship is described in detail in [18].

Definition 2.1 (van Kampen square). A *van Kampen (VK) square (i)* is a pushout which satisfies the following condition: given a commutative cube *(ii)* of which *(i)* forms the bottom face and the back faces are pullbacks,



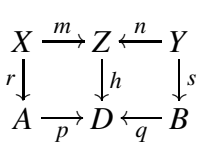
(i)



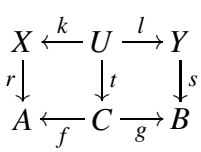
(ii)

the front faces are pullbacks if and only if the top face is a pushout

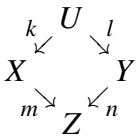
Another, equivalent, way of defining a VK square in a category with pullbacks is as follows. A VK square *(i)* is a pushout which satisfies the property that given a commutative diagram *(iii)*, the two squares are pullbacks if and only if there exists an object U and diagrams



(iii)



(iv)



(v)

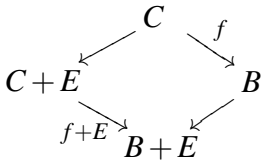
in which the squares in *(iv)* are pullbacks and *(v)* is a pushout.

By a *pushout along a monomorphism* we mean a pushout, as in Diagram *(i)* above, in which m is a monomorphism. (Of course by the symmetric nature of pushouts, if instead f were a monomorphism this pushout square could also be viewed as a pushout along a monomorphism). Similarly, if m is a coproduct injection, we have a pushout along a coproduct injection.

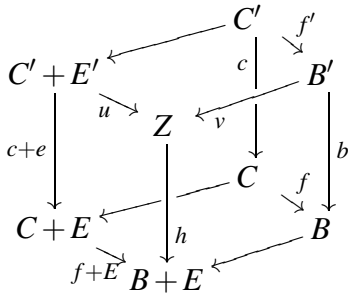
A crucial class of examples of VK squares is provided by:

Theorem 2.2. *In an extensive category, pushouts along coproduct injections are VK squares.*

Proof. If $m : C \to A$ is a coproduct injection, say $C \to C+E$, then the diagrams *(i)* and *(ii)* have the form



(i)



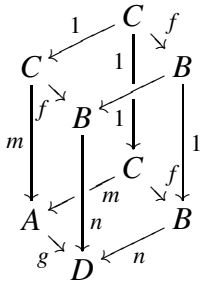
(ii)

where the unnamed arrows are coproduct injections; it is a straightforward exercise in the theory of extensive categories to show that if either the top face is a pushout or the front faces are pullbacks, then Z has the form $B'+E'$ and so to deduce the result. □

We have the following important properties of VK squares:

Lemma 2.3. In a VK square as in *(i)*, if m is a monomorphism then n is a monomorphism and the square is also a pullback.

Proof. Suppose that the bottom face of the cube



is VK. Then the top and bottom squares are pushouts, while the back squares are pullbacks if m is a monomorphism. Thus the front faces will be pullbacks: the front right face being a pullback means that n is a monomorphism, and the front left face being a pullback means that the original square is a pullback. □

3 Adhesive categories

We shall now proceed to define the notion of adhesive category, and provide various examples and counterexamples.

Definition 3.1 (Adhesive category). A category **C** is said to be *adhesive* if

(i) **C** has pushouts along monomorphisms;

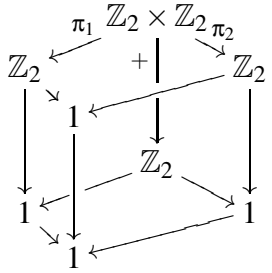
(ii) **C** has pullbacks;

(iii) pushouts along monomorphisms are VK-squares.

Since every monomorphism in **Set** is a coproduct injection, and **Set** is extensive, we immediately have:

Example 3.2. Set is adhesive.

Observe that the restriction to pushouts along monomorphisms is necessary: there are pushouts in **Set** which are not VK squares. Consider the 2 element abelian group $\mathbb{Z}_2$ (the following argument works for any non-trivial group). In the diagram



both the bottom and the top faces are easily verified to be pushouts and the rear faces are both pullbacks. However, the front two faces are not pullbacks.

Even with the restriction to pushouts along a monomorphism, many well-known categories fail to be adhesive.

Counterexample 3.3. The categories **Pos**, **Top**, **Gpd** and **Cat** are not adhesive.

Proof. For the case of the category of posets, write $[n]$ for the ordered set $\{0 \leq 1 \leq \ldots n-1\}$. The pushout square

$$\begin{array}{ccc} [1] & \xrightarrow{0} & [2] \\ {\scriptstyle 1}\downarrow & & \downarrow \\ [2] & \longrightarrow & [3] \end{array}$$

is not van Kampen, since it is not stable under pullback along the map $[2] \to [3]$ sending 0 to 0 and sending 1 to 2. Thus **Pos** is not adhesive. The same pushout square, regarded as a pushout of categories, shows that **Cat** is not adhesive. For the case of **Gpd**, one simply replaces the poset $[n]$ by the groupoid with n objects and a unique isomorphism between each pair of objects.

Finally consider the category **Top** of topological spaces. A finite poset induces a finite topological space on the same underlying set: the topology is determined by specifying that y is in the closure of x if and only if $x \leq y$. Applying this process to the previous example yields an example showing that **Top** is not adhesive. □

Since the definition of adhesive category only uses pullbacks, pushouts, and relationships between these, we have the following constructions involving adhesive categories:

Proposition 3.4.

(i) If $\mathbf{C}$ and $\mathbf{D}$ are adhesive categories then so is $\mathbf{C} \times \mathbf{D}$;

(ii) If $\mathbf{C}$ is adhesive then so are $\mathbf{C}/C$ and $C/\mathbf{C}$ for any object C of $\mathbf{C}$;

(iii) If $\mathbf{C}$ is adhesive then so is any functor category $[\mathbf{X}, \mathbf{C}]$.

Since **Set** is adhesive, part (iii) of the proposition implies that any presheaf topos $[\mathbf{X}, \mathbf{Set}]$ is adhesive. In particular, the category **Graph** of directed graphs is adhesive. Indeed, if $\mathbf{C}$ is adhesive, then so is the category

$$Graph(\mathbf{C}) = [\cdot \rightrightarrows \cdot, \mathbf{C}]$$

of internal graphs in $\mathbf{C}$ [24].

Part (ii) implies that categories of typed graphs [1], coloured (or labelled) graphs [6], ranked graphs [13] and hypergraphs [11], considered in the literature on graph grammars, are adhesive.

As a consequence, all proof techniques and constructions in adhesive categories can be readily applied to any of the aforementioned categories of graphs. In fact, more generally, we have:

Proposition 3.5. *Any elementary topos is adhesive.*

This is somewhat harder to prove than the result for presheaf toposes; the proof can be found in [18].

Part *(ii)* of Proposition 3.4 also allows us to construct examples of adhesive categories which are not toposes.

Example 3.6. The category $\mathbf{Set}_* = 1/\mathbf{Set}$ of pointed sets (or equivalently, sets and partial functions) is adhesive, but is not extensive, and therefore, is not a topos.

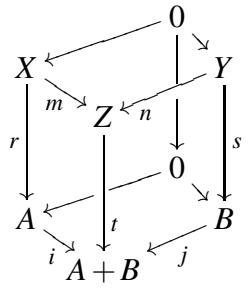
Proof. In the category of pointed sets, the initial object is the one-point set 1. Since every non-initial object has a map into 1, the initial object is not strict, and so the category is not extensive [3, Proposition 2.8]. □

4 Basic properties of adhesive categories

Here we provide several simple lemmas which hold in any adhesive category. Lemma 4.1 demonstrates why adhesive categories can be considered as a generalisation of extensive categories. Lemmas 4.2, 4.3, 4.5 and 4.6 shed some light on pushouts in adhesive categories.

Lemma 4.1. An adhesive category is extensive if and only if it has a strict initial object.

Proof. In an extensive category the initial object is strict [3, Proposition 2.8]. On the other hand, in an adhesive category with strict initial object, any arrow with domain 0 is mono. Consider the cube



in which the bottom square is a pushout along a monomorphism, while the back squares are pullbacks since the initial object is strict. By adhesiveness, front squares are pullbacks if and only if the top squares is a pushout; but this says that the front squares are pullbacks if and only if the top row of these squares is a coproduct (Z=X+Y). □

The conclusions of the following two lemmas are used extensively in literature on algebraic graph rewriting. Indeed, they are usually assumed as axioms (see [9] and §6 below) in attempts at generalising graph rewriting. They hold in any adhesive category by Lemma 2.3:

Lemma 4.2. Monomorphisms are stable under pushout.

Lemma 4.3. Pushouts along monomorphisms are also pullbacks.

The notion of pushout complement [12] is vital in algebraic approaches to graph rewriting.

Definition 4.4. Let $m : C \to A$ and $g : A \to B$ be arrows in an arbitrary category (m is not assumed to be monic). A *pushout complement* of the pair (m, g) consists of arrows $f : C \to B$ and $n : B \to D$ for which the resulting square commutes and is a pushout.

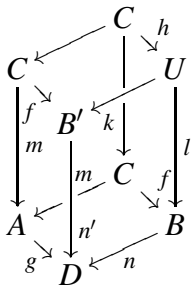
The conclusion of the following lemma is a crucial ingredient in many applications of graph rewriting. It has also been assumed as an axiom [11] in order to prove the concurrency theorem (cf. Theorem 5.9). It is important mainly because it assures that once an occurrence of a left hand side of a rewrite rule is found within a structure, then the application of the rewrite rule results in a structure which is unique up to isomorphism (cf. §5). In other words, rewrite rule application is functional up to isomorphism.

Lemma 4.5. Pushout complements of monos (if they exist) are unique up to isomorphism.

Proof. Suppose that

$$\begin{array}{ccc} & C & \\ {\scriptstyle m}\swarrow & & \searrow{\scriptstyle f} \\ A & & B \\ {\scriptstyle g}\searrow & & \swarrow{\scriptstyle n} \\ & D & \end{array} \qquad \begin{array}{ccc} & C & \\ {\scriptstyle m}\swarrow & & \searrow{\scriptstyle f'} \\ A & & B' \\ {\scriptstyle g}\searrow & & \swarrow{\scriptstyle n'} \\ & D & \end{array}$$

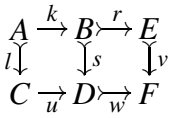
are pushouts and that m is mono. Consider the cube



in which the front right face is a pullback, and $h : C \to U$ is the map induced by g and g'. Then the front faces and the back left face are pullbacks, hence the back right face is also a pullback; and the bottom face is a pushout, hence the top face is a pushout. But this implies that k is invertible; by symmetry, so too is l. The induced isomorphism $j = lk^{-1} : C \to C'$ satisfies $n'j = n$ and $jf = f'$. □

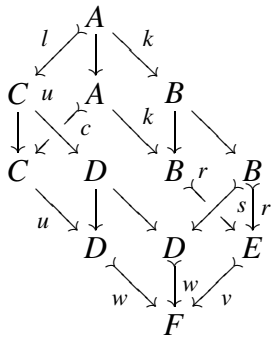
The final lemma of this section will be used in Section 6 to show that adhesive categories are high-level replacement systems:

Lemma 4.6. Consider a diagram



in which the marked morphisms are mono, the exterior is a pushout and the right square is a pullback. Then the left square is a pushout, and so all squares are both pullbacks and pushouts.

Proof. This amounts to stability of the exterior pushout under pullback along $w : D \to F$. The reader can verify this by examining the diagram below.



□

4.1 Algebra of subobjects

We can put a preorder on monomorphisms into an object Z of an arbitrary category by defining a monomorphism $a : A \to Z$ to be less than or equal to a monomorphism $b : B \to Z$ precisely when there exists an arrow $c : A \to B$ such that $bc = a$. A subobject (of Z) is an equivalence class with respect to the equivalence generated by this preorder. For example, subobjects in **Set** are subsets while subobjects in **Graph** are subgraphs.

Here we shall demonstrate that unions of subobjects in adhesive categories can be formed by pushout over their intersection This provides further evidence of how pushouts behave in adhesive categories as well as making more precise the intuition that the pushout operation "glues together" two structures along a common substructure. As a corollary, it follows that in an adhesive category the lattices of subobjects are distributive.

Let **C** be an adhesive category, and Z a fixed object of **C**. We write Sub(Z) for the category of subobjects of Z in **C**; it has products (=intersections), given by pullback in **C**. It has a top object, given by Z itself. If **C** has a strict initial object 0, then the unique map $0 \to Z$ is a monomorphism, and is the bottom object of Sub(Z).

Theorem 4.7. *For an object Z of an adhesive category* **C***, the category Sub(Z) of subobjects of Z has binary coproducts: the coproduct of two subobjects is the pushout in* **C** *of their intersection.*

Proof. We shall show how to form binary coproducts (=unions) in Sub(Z). Let $a : A \to Z$ and $b : B \to Z$ be subobjects of Z, and form the intersection $A \cap B \to Z$, with projections $p : A \cap B \to A$ and $q : A \cap B \to B$; and now the pushout

$$\begin{array}{ccc} A \cap B & \xrightarrow{q} & B \\ {\scriptstyle p}\downarrow & & \downarrow{\scriptstyle v} \\ A & \xrightarrow[u]{} & C \end{array}$$

in **C**. Let $c : C \to Z$ be the unique map satisfying $cu = a$ and $cv = b$. We shall show that c is a monomorphism, and so that C is the coproduct $A \cup B$ in Sub(Z) of A and B. Suppose then that $f, g : K \to C$ satisfy $cf = cg$. Form the following pullbacks

$$\begin{array}{ccccc} L_1 & \xrightarrow{l_1} & K & \xleftarrow{l_2} & L_2 \\ {\scriptstyle f_1}\downarrow & & \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle f_2} \\ A & \xrightarrow[u]{} & C & \xleftarrow[v]{} & B \end{array} \qquad \begin{array}{ccccc} M_1 & \xrightarrow{m_1} & K & \xleftarrow{m_2} & M_2 \\ {\scriptstyle g_1}\downarrow & & \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle g_2} \\ A & \xrightarrow[u]{} & C & \xleftarrow[v]{} & B \end{array} \qquad \begin{array}{ccccc} N_{11} & \xrightarrow{m_{11}} & M_1 & \xleftarrow{m_{12}} & N_{12} \\ {\scriptstyle l_{11}}\downarrow & & \downarrow{\scriptstyle m_1} & & \downarrow{\scriptstyle l_{12}} \\ L_1 & \xrightarrow{l_1} & K & \xleftarrow{l_2} & L_2 \\ {\scriptstyle l_{21}}\uparrow & & \uparrow{\scriptstyle m_2} & & \uparrow{\scriptstyle l_{22}} \\ N_{21} & \xrightarrow[m_{21}]{} & M_2 & \xleftarrow[m_{22}]{} & N_{22} \end{array}$$

and note that each of the following pairs are the coprojections of a pushout, hence each pair is jointly epimorphic: (l_1, l_2), (m_1, m_2), (m_{11}, m_{12}), and (m_{21}, m_{22}). We are to show that $f = g$; to do this, it will suffice to show that $fm_1 = gm_1$ and $fm_2 = gm_2$; we shall prove only the former, leaving the latter to the reader. To show that $fm_1 = gm_1$ it will in turn suffice to show that $fm_1m_{11} = gm_1m_{11}$ and $fm_1m_{12} = gm_1m_{12}$.

First note that $af_1l_{11} = cuf_1l_{11} = cfl_1l_{11} = cgl_1l_{11} = cgm_1m_{11} = cug_1m_{11} = ag_1m_{11}$, so that $f_1l_{11} = g_1m_{11}$ since a is monic; thus $fm_1m_{11} = fl_1l_{11} = uf_1l_{11} = ug_1m_{11} = gm_1m_{11}$ as required.

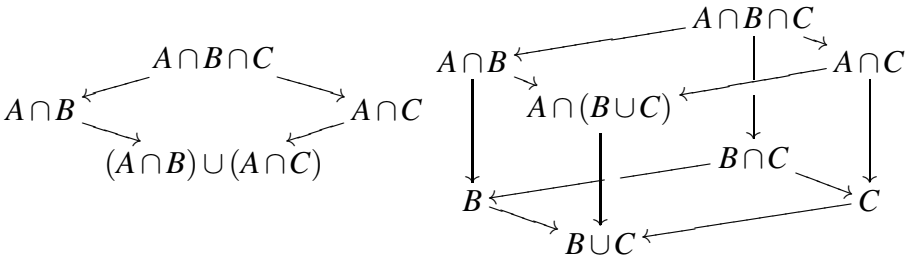
On the other hand, $bf_2l_{12} = cvf_2l_{12} = cfl_2l_{12} = cgl_2l_{12} = cgm_1m_{12} = cug_1m_{12} = ag_1m_{12}$, so by the universal property of the pullback $A \cap B$, there is a unique map $h : N_{12} \to A \cap B$ satisfying $ph = g_1m_{12}$ and $qh = f_2l_{12}$. Now

$fm_1m_{12} = fl_2l_{12} = vf_2l_{12} = vqh = uph = ug_1m_{12} = gm_1m_{12}$, and so $fm_1 = gm_1$ as claimed. As promised, we leave the proof that $fm_2 = gm_2$ to the reader, and deduce that $f = g$, so that c is monic. □

Since pushouts are stable it follows that intersections distribute over unions:

Corollary 4.8. The lattice Sub(Z) is distributive.

Proof. For subobjects $a : A \to Z, b : B \to Z$ and $c : C \to Z$, construct the pushout below left



to obtain $(A \cap B) \cup (A \cap C)$. It is an easy exercise to show that the side faces of the cube above right are all pullbacks, and therefore, that the top face is a pushout. By the universal property of pushouts we have that

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

as subobjects. □

5 Double-Pushout Rewriting

Here we shall recall the basic notions of double-pushout rewriting [12, 6] and show that it can be defined within an arbitrary adhesive category.

Henceforth we shall assume that **C** is an adhesive category.

Definition 5.1 (Production). A production p is a span

$$L \xleftarrow{l} K \xrightarrow{r} R \tag{1}$$

in **C**. We shall say that p is *left-linear* when l is mono, and *linear* when both l and r are mono. We shall let $\mathcal{P}$ denote an arbitrary set of productions and let p range over $\mathcal{P}$.

In order to develop an intuition of why a production is defined as a span, we shall restrict our attention to linear production rules. One may then consider K as a substructure of both L and R. We think of L as the left-hand side of the rewrite rule p, and therefore, in order to perform the rewrite, we need to match L as a substructure of a redex C. The structure K, thought of as a substructure of L, is exactly the part of L which is to remain invariant as we apply the rule to C. Finally, parts of R which are not in K should be added to produce the final result of the rewrite.

Thus, an application of a rewrite rule consists of three steps. First we must match L as a substructure of the redex C; secondly, we delete all of parts of the redex matched by L which are not included in K. Thirdly, we add all of R which is not contained in K, thereby producing a new structure D. The deletion and addition of structure is handled, respectively, by finding a pushout complement and constructing a pushout.

We shall first clarify what is meant by matching a left-hand side of a rewrite rule within a redex.

Definition 5.2 (Gluing Conditions). Given a production p as in (1), a *match* in C is a morphism $f : L \to C$. A match f satisfies the *gluing conditions* with respect to p precisely when there exists an object E and morphisms $g : K \to E$ and $v : E \to C$ such that

$$\begin{array}{ccc} L & \xleftarrow{l} & K \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ C & \xleftarrow[v]{} & E \end{array}$$

is a pushout diagram. (In other words, there exists a pushout complement of (f,l) in the sense of Definition 4.4.)

Definition 5.3 (Derivation). Given an object $C \in \mathbf{C}$ and a set of productions $\mathcal{P}$, we write $C \Longrightarrow_{p,f} D$ for a production $p \in \mathcal{P}$ and a morphism $f : L \to C$ if (a) f satisfies the gluing conditions with respect to l, and (b) there is a diagram

$$\begin{array}{ccccc} L & \xleftarrow{l} & K & \xrightarrow{r} & R \\ {\scriptstyle f}\downarrow & & {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xleftarrow[v]{} & E & \xrightarrow[w]{} & D \end{array}$$

in which both squares are pushouts.

The object E in the above diagram can be thought of as a temporary state in the middle of the rewrite process. Returning briefly to our informal description, it is the structure obtained from C by deleting all the parts of L not contained in K. Recall from Lemma 4.5 that if l is mono (that is, if p is

left-linear) then E is unique up to isomorphism. Indeed, if p is a left-linear production, $C \longrightarrow\!\!\triangleright_{p,f} D$ and $C \longrightarrow\!\!\triangleright_{p,f} D'$ then we must have $D \cong D'$. This is a consequence of Lemma 4.5 and the fact that pushouts are unique up to isomorphism.

Definition 5.4 (Adhesive Grammar). An adhesive grammar $\mathbf{G}$ is a pair $\langle \mathbf{C}, \mathbf{P} \rangle$ where $\mathbf{C}$ is an adhesive category and $\mathbf{P}$ is a set of linear productions.

Assuming that all the productions are linear allows us to derive a rich rewriting theory on adhesive categories. Henceforward we assume that we are working over an adhesive grammar $\mathbf{G}$.

5.1 Local Church-Rosser theorem

As shall be explained in section 6, adhesive categories with coproducts are high-level replacement systems. In particular, we get the local Church-Rosser theorem [17, 9].

Before presenting this theorem we need to recall briefly the notions of parallel-independent derivation and sequential-independent derivation. The reader may wish to consult [6] for a more complete presentation.

A *parallel-independent derivation* is a pair of derivations

$$C \longrightarrow\!\!\triangleright_{p_1,f_1} D_1 \qquad \text{and} \qquad C \longrightarrow\!\!\triangleright_{p_2,f_2} D_2$$

as illustrated in diagram (2) which satisfy an additional requirement, namely the existence of morphisms $r : L_1 \to E_2$ and $s : L_2 \to E_1$ which render the diagram commutative, in the sense that $v_2 r = f_1$ and $v_1 s = f_2$.

$$\begin{array}{ccccccccccc}
R_1 & \xleftarrow{r_1} & K_1 & \xrightarrow{l_1} & L_1 & \overset{r}{\searrow} \quad \overset{s}{\swarrow} & L_2 & \xleftarrow{l_2} & K_2 & \xrightarrow{r_2} & R_2 \\
\downarrow{\scriptstyle h_1} & & \downarrow{\scriptstyle g_1} & & & \overset{f_1}{\searrow} \quad \overset{f_2}{\swarrow} & & & \downarrow{\scriptstyle g_2} & & \downarrow{\scriptstyle h_2} \\
D_1 & \xleftarrow[w_1]{} & E_1 & & & \xrightarrow[v_1]{} C \xleftarrow[v_2]{} & & & E_2 & \xrightarrow[w_2]{} & D_2
\end{array} \tag{2}$$

Similarly, a *sequential-independent derivation*, illustrated in diagram (3), is a derivation

$$C \longrightarrow\!\!\triangleright_{p_1,f_1} D_1 \longrightarrow\!\!\triangleright_{p_2,f_2'} D$$

where there additionally exist arrows $r' : R_1 \to E_3$ and $s' : L_2 \to E_1$ such that $w_1 s' = f_2'$ and $v_3 r' = h_1$.

$$\begin{array}{ccccccccccc}
L_1 & \xleftarrow{l_1} & K_1 & \xrightarrow{r_1} & R_1 & \overset{r'}{\searrow} \quad \overset{s'}{\swarrow} & L_2 & \xleftarrow{l_2} & K_2 & \xrightarrow{r_2} & R_2 \\
\downarrow{\scriptstyle f_1} & & \downarrow{\scriptstyle g_1} & & & \overset{h_1}{\searrow} \quad \overset{f_2'}{\swarrow} & & & \downarrow{\scriptstyle g_2'} & & \downarrow{\scriptstyle h_2'} \\
C & \xleftarrow[v_1]{} & E_1 & & & \xrightarrow[w_1]{} D_1 \xleftarrow[v_3]{} & & & E_3 & \xrightarrow[w_3]{} & D
\end{array} \tag{3}$$

The statement of the theorem below differs from those previously published in the literature in that we do not need coproducts to establish the equivalence of the first 3 items.

Theorem 5.5 (Local Church-Rosser). *The following are equivalent*

1. $C \Longrightarrow_{p_1,f_1} D_1$ *and* $C \Longrightarrow_{p_2,f_2} D_2$ *are parallel-independent derivations*
2. $C \Longrightarrow_{p_1,f_1} D_1$ *and* $D_1 \Longrightarrow_{p_2,f_2'} D$ *are sequential-independent derivations*
3. $C \Longrightarrow_{p_2,f_2} D_2$ *and* $D_2 \Longrightarrow_{p_1,f_1'} D$ *are sequential-independent derivations.*

If moreover **C** *is extensive then we may add*

4. $C \Longrightarrow_{p_1+p_2,[f_1,f_2]} D$ *is a derivation.*

Proof. (1)⇒(2): Suppose that $C \Longrightarrow_{p_1,f_1} D_1$ and $C \Longrightarrow_{p_2,f_2} D_2$ are parallel-independent derivations. Form the pullback *(i)* below.

$$\begin{array}{ccc} E_1 & \xleftarrow{e_1} & G \\ {\scriptstyle v_1}\downarrow & & \downarrow{\scriptstyle e_2} \\ C & \xleftarrow[v_2]{} & E_2 \end{array} \qquad \begin{array}{ccccc} L_1 & \xleftarrow{l_1} & K_1 & & \\ {\scriptstyle r}\downarrow & & {\scriptstyle g_1}\downarrow & & \\ E_2 & \xleftarrow{} & \cdot & \xleftarrow{g_2} & K_2 \\ {\scriptstyle v_2}\downarrow & & \downarrow & & \downarrow{\scriptstyle l_2} \\ C & \xleftarrow[v_1]{} & E_1 & \xleftarrow[s]{} & L_2 \end{array} \qquad \begin{array}{ccccc} L_1 & \xleftarrow{l_1} & K_1 & & \\ {\scriptstyle r}\downarrow & (\dagger) & {\scriptstyle k_1}\downarrow & & \\ E_2 & \xleftarrow{e_2} & G & \xleftarrow{k_2} & K_2 \\ {\scriptstyle v_2}\downarrow & (\star) & {\scriptstyle e_1}\downarrow & (\ddagger) & \downarrow{\scriptstyle l_2} \\ C & \xleftarrow[v_1]{} & E_1 & \xleftarrow[s]{} & L_2 \end{array} \qquad \begin{array}{ccccc} & & K_1 & \xrightarrow{r_1} & R_1 \\ & & {\scriptstyle k_1}\downarrow & & \downarrow{\scriptstyle t} \\ K_2 & \xrightarrow{k_2} & G & \xrightarrow{e_3} & E_3 \\ {\scriptstyle r_2}\downarrow & & {\scriptstyle e_4}\downarrow & & \downarrow{\scriptstyle w_3} \\ R_2 & \xrightarrow[u]{} & E_4 & \xrightarrow[w_4]{} & D \end{array}$$

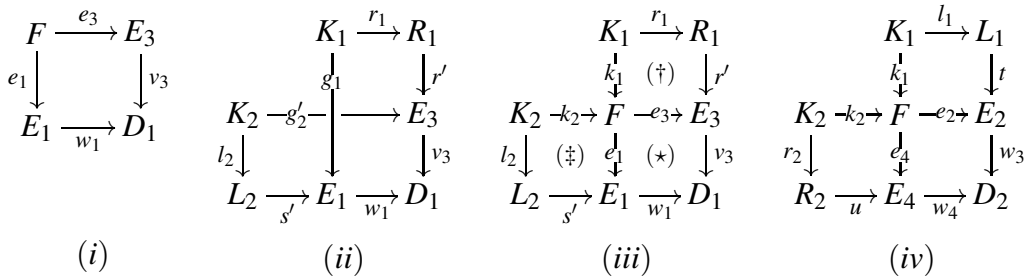
(i) (ii) (iii) (iv)

The two regions in *(ii)* are pushouts [cf. diagram (3)]. Combining the two diagrams gives *(iii)*, with k_1 and k_2 obtained by the universal property of *(i)* and satisfying $e_2k_1 = g_1$ and $e_1k_2 = g_2$. Regions (†), (‡), and (⋆) are pushouts by Lemma 4.6, and one now goes on to construct *(iv)* by taking successive pushouts.

The sequential-independent derivation $C \Longrightarrow_{p_1,f_1} D_1 \Longrightarrow_{p_2,f_2'} D$ may now be constructed with the pushout squares below.

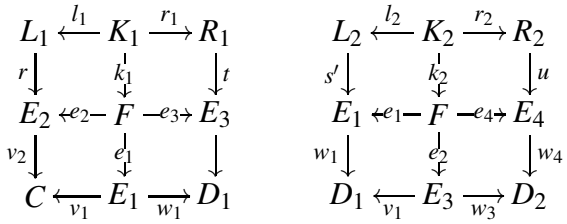
$$\begin{array}{ccccc} L_1 & \xleftarrow{l_1} & K_1 & \xrightarrow{r_1} & R_1 \\ {\scriptstyle r}\downarrow & & {\scriptstyle k_1}\downarrow & & \downarrow{\scriptstyle t} \\ E_2 & \xleftarrow{e_2} & G & \xrightarrow{e_3} & E_3 \\ {\scriptstyle v_2}\downarrow & & {\scriptstyle e_1}\downarrow & & \downarrow{\scriptstyle v_3} \\ C & \xleftarrow[v_1]{} & E_1 & \xrightarrow[w_1]{} & D_1 \end{array} \qquad \begin{array}{ccccc} L_2 & \xleftarrow{l_2} & K_2 & \xrightarrow{r_2} & R_2 \\ {\scriptstyle s}\downarrow & & {\scriptstyle k_2}\downarrow & & \downarrow{\scriptstyle u} \\ E_1 & \xleftarrow{e_1} & G & \xrightarrow{e_4} & E_4 \\ {\scriptstyle w_1}\downarrow & & {\scriptstyle e_3}\downarrow & & \downarrow{\scriptstyle w_4} \\ D_1 & \xleftarrow[v_1]{} & E_3 & \xrightarrow[w_3]{} & D \end{array}$$

(2)⇒(1): Suppose that $C \Longrightarrow_{p_1,f_1} D_1$ and $D_1 \Longrightarrow_{p_2,f_2'} D$ are sequential-independent derivations. Form the pullback *(i)* below.



The two regions in *(ii)* are pushouts [cf. diagram (3)]. Combining the two diagrams gives *(iii)*, with k_1 and k_2 obtained by the universal property of *(i)* and satisfying $e_3k_2 = g_2'$ and $e_1k_1 = g_1$. Regions (†), (‡), and (⋆) are pushouts by Lemma 4.6, and one now goes on to construct *(iv)* by taking successive pushouts.

The parallel independent derivations $C \Longrightarrow_{p_1,f_1} D_1$ and $C \Longrightarrow_{p_2,f_2} D_2$ may now be constructed with the pushout squares below.



The proof of (1)⇔(3) is similar; the proof of (1)⇔(4) is a straightforward exercise in the theory of extensive categories. □

In fact, the proof that (1)⇒(2) remains valid more generally in the context of left-linear productions, but the proof of the converse requires linearity.

5.2 Concurrency Theorem

The original concurrency theorems were proved for graph grammars [8] and later generalised to high level replacement structures (cf. §6) in [11] which satisfy additional axiom sets, there called HLR2 and HLR2*. Roughly, the concurrency theorem states that given two derivations in a sequence, together with information about how they are related, one may construct a single derivation which internalises the two original derivations and performs them "concurrently". Moreover, one may reverse this process and deconstruct a

concurrent derivation into two related sequential derivations. Here we state and prove the concurrency theorem for adhesive grammars without the need for extra axioms.

We shall first need to recall the notions of dependency relation, dependent derivation and concurrent production.

Definition 5.6 (Dependency Relation). Suppose that p_1 and p_2 are linear productions. A *dependency relation* for $\langle p_1, p_2 \rangle$ is an object X together with arrows $s : X \to R_1$ and $t : X \to L_2$ for which r_1, s, t, and l_2 can be incorporated into a diagram

$$\begin{array}{ccccc} & & X & & \\ & \swarrow^{s} & & \searrow^{t} & \\ K_1 & \xrightarrow{r_1} & R_1 & \xleftarrow{l_2} & K_2 \\ {\scriptstyle g_1'}\downarrow & \searrow^{h_1'} & & \swarrow_{f_2'} & \downarrow{\scriptstyle g_2'} \\ E_1' & \xrightarrow[w_1']{} & D' & \xleftarrow[v_2']{} & E_2' \end{array} \tag{4}$$

in which all three regions are pushouts.

Definition 5.7 (Dependent Derivation). Consider a derivation $C \Longrightarrow_{p_1,f_1} D_1 \Longrightarrow_{p_2,f_2} D$ as illustrated in *(i)* below

$$\begin{array}{ccccccccc} L_1 & \xleftarrow{l_1} & K_1 & \xrightarrow{r_1} & R_1 & & L_2 & \xleftarrow{l_2} & K_2 & \xrightarrow{r_2} & R_2 \\ {\scriptstyle f_1}\downarrow & & {\scriptstyle g_1}\downarrow & & \searrow^{h_1} & \swarrow_{f_2} & & & \downarrow{\scriptstyle g_2} & & \downarrow{\scriptstyle h_2} \\ C & \xleftarrow[v_1]{} & E_1 & \xrightarrow[w_1]{} & D_1 & & \xleftarrow[v_3]{} & E_3 & \xrightarrow[w_3]{} & D & \end{array}$$

(i)

$$\begin{array}{ccccccc} & & & & X & & \\ & & & \swarrow^{s} & & \searrow^{t} & \\ K_1 & \xrightarrow{r_1} & R_1 & & \swarrow_{h_1'} & \searrow_{f_2'} & L_2 & \xleftarrow{l_2} & K_2 \\ {\scriptstyle g_1'}\downarrow & & & & & & & & \downarrow{\scriptstyle g_2'} \\ E_1' & & \xrightarrow[w_1']{} & & D' & & \xleftarrow[v_2']{} & & E_2' \\ {\scriptstyle e_1}\vdots & & & & \vdots{\scriptstyle d} & & & & \vdots{\scriptstyle e_2} \\ E_1 & & \xrightarrow[w_1]{} & & D_1 & & \xleftarrow[v_2]{} & & E_2 \end{array}$$

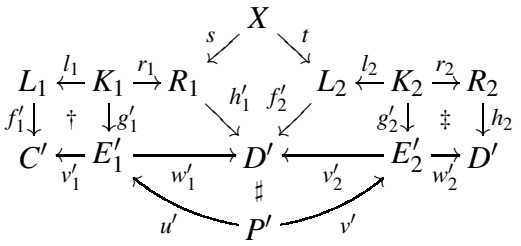
(ii)

and a dependency relation X for $\langle p_1, p_2 \rangle$. The derivation is said to be X-*dependent* if $h_1 s = f_2 t$ and there exist morphisms $e_1 : E_1' \to E_1$ and $e_2 : E_2' \to E_2$ satisfying $e_1 g_1' = g_1$ and $e_2 g_2' = g_2$, and if moreover the unique map $d : D' \to D_1$ satisfying $d h_1' = h_1$ and $d f_2' = f_2$ also satisfies $d w_1' = w_1 e_1$ and $d v_2' = v_2 e_2$ (see *(ii)*).

Definition 5.8 (Concurrent Production). Given a dependency relation X for $\langle p_1, p_2 \rangle$, the X-*concurrent production* $p_1 ;_X p_2$ is the span

$$C' \xleftarrow{v_1' u'} P' \xrightarrow{w_2' v'} D'$$

obtained by taking the bottom row of the following extension of Diagram (4)



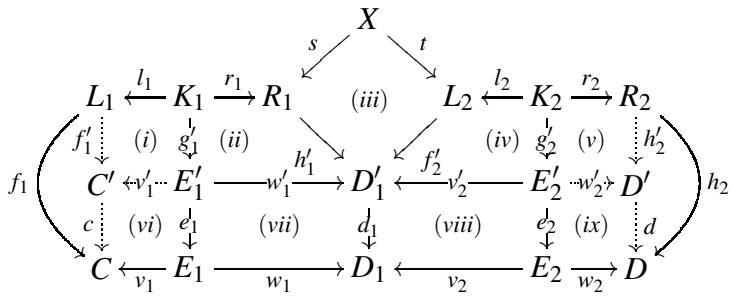
in which † and ‡ are pushouts and ♯ is a pullback.

Theorem 5.9 (Concurrency Theorem).

1. *Given an X-dependent derivation $C \Longrightarrow_{p_1,f_1} D_1 \Longrightarrow_{p_2,f_2} D$ there exists an X-concurrent derivation $C \Longrightarrow_{p_1;_X p_2} D$*

2. *Given an X-concurrent derivation $C \Longrightarrow_{p_1;_X p_2} D$, there exists an X-dependent derivation $C \Longrightarrow_{p_1,f_1} D_1 \Longrightarrow_{p_2,f_2} D$.*

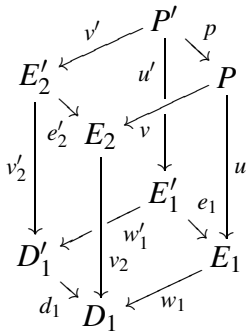
Proof. 1. Suppose that we have an X dependent-derivation, as in the solid part of the diagram



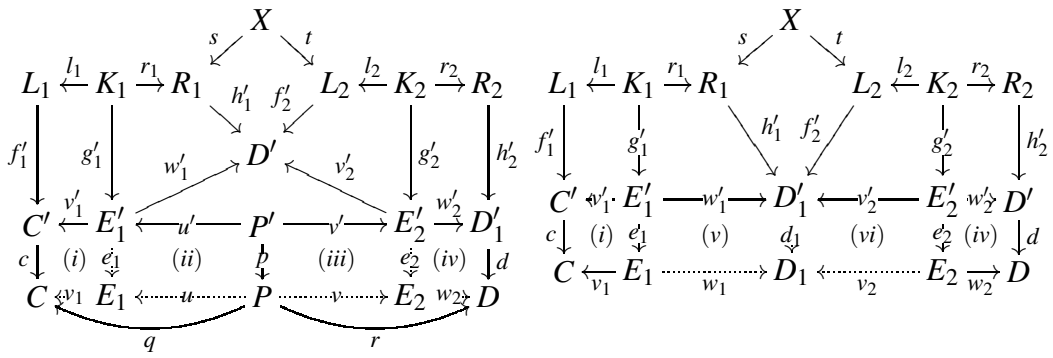
in which *(iii)*, *(i)*+*(vi)*, *(ii)*+*(vii)*, *(iv)*+*(viii)*, *(v)*+*(ix)*, *(ii)*, and *(iv)* are pushouts, so that also *(vii)* and *(viii)* are pushouts. Fill in the dotted parts of the diagram to obtain further pushouts *(i)*, *(v)*, *(vi)*, and *(ix)*.

By Lemma 4.2 both w'_1 and v'_2 are monomorphisms, and now by Lemma 4.3 both *(vii)* and *(viii)* are pullbacks. Consider the cube in which bottom and front left faces are the pullbacks *(vii)* and *(viii)*, and the remaining faces are constructed so as to be pullbacks. Since the bottom face is also a pushout, so is the top face. Similarly, since the front left face is a pushout, so is the back

right face.



2. Suppose that we have an X-concurrent derivation $C \Longrightarrow_{p_1;_X p_2,f} D$, as illustrated by the solid part of the diagram below left.



We construct pushouts *(ii)* and *(iii)* and obtain v_1 and w_2 using the universal properties. It now follows that *(i)* and *(iv)* are also pushouts. Now construct the pushout *(v)*; since *(ii)*+*(v)* and *(iii)* are pushouts, there is a unique map $v_2 : E_2 \to D_1$ so that *(vi)* is a pushout and *(iii)*+*(vi)* equals *(ii)*+*(v)*. The diagram on the right now provides the required X-dependent derivation $C \Longrightarrow_{p_1,cf'_1} D_1 \Longrightarrow_{p_2,d_1 f'_2} D$. □

6 Relation with High-Level Replacement Systems

High-level replacement systems [9, 10, 11] or HLRS encompass several attempts to isolate general categorical axioms which lead to categories in which one can define double-pushout graph rewriting and prove useful theorems such as the local Church-Rosser theorem and the concurrency theorem.

HLRS have axioms which are parametrised over an arbitrary class of morphisms $\mathcal{M}$. Here we give a simplified version of the definition which appears

in [9]. The simplification is that we take $\mathcal{M}$ to be the class of monomorphisms: we justify this by noting that this is the case in the majority of examples.

Definition 6.1 (High-Level Replacement System). A high-level replacement system is a category **S** satisfying the following axioms:

1. pairs $C \leftarrow A \rightarrow B$ with at least one of the arrows mono have a pushout;
2. pairs $B \rightarrow D \leftarrow C$ with both morphisms mono have pullbacks;
3. monos are preserved by pushout;
4. finite coproducts exist;
5. pushouts of monos are pullbacks;
6. pushout-pullback decomposition holds: that is, given a diagram

$$\begin{array}{ccccc} A & \xrightarrow{k} & B & \rightarrowtail^{r} & E \\ {\scriptstyle l}\downarrow & & \downarrow{\scriptstyle s} & & \downarrow{\scriptstyle v} \\ C & \xrightarrow[u]{} & D & \rightarrowtail_{w} & F \end{array}$$

   if the marked morphisms are mono, the whole rectangle is a pushout and the right square is a pullback, then the left square is a pushout.

Lemma 6.2. Any adhesive category with an initial object is an HLRS.

Proof. This follows immediately from Lemmas 4.2, 4.3, and 4.6. □

The axioms listed above are enough to prove the local Church-Rosser theorem (cf. Theorem 5.5), but *not* the concurrency theorem (cf. Theorem 5.9). To prove the latter, extra axioms had to be introduced in [11], such as the conclusion of the following lemma. Interestingly, it is almost the dual of the main axiom of adhesive categories.

Lemma 6.3 (Cube-pushout-pullback-lemma [11]). Given a cube in which all arrows in the top and bottom faces are mono, if the top face is a pullback and the front faces are pushouts, then the bottom face is a pullback if and only if the back faces are pushouts.

Proof. Since the front faces are pushouts along monomorphisms, they are also pullbacks.

If the bottom face is a pullback, then the back faces are pushouts by stability of the pushouts on the front faces. Suppose conversely that the back faces are pushouts; since they are pushouts along monomorphisms, they are also pullbacks. One now simply "rotates the cube": since the front right and back left faces are pushouts, and the top and back right faces are pullbacks, it follows by adhesiveness that the bottom square is a pullback. □

7 Conclusions and future work

We introduced the notions of van Kampen (VK) square and adhesive category. VK squares are "well-behaved pushouts", and a category is adhesive when pushouts along monos are VK. Adhesive categories are closely related to extensive categories.

Double-pushout (D-p) rewriting can be defined in an arbitrary adhesive category. We introduced adhesive grammars, which are adhesive categories with a set of linear productions. Adhesive grammars have sufficient structure for the development of a rich rewriting theory. In particular, we proved the local Church-Rosser and the so-called concurrency theorem within the setting of adhesive grammars. We have also shown that adhesive categories satisfy many of the axioms [9, 11] which were proposed in order to prove these theorems. Thus, we have arrived at a class of categories which supports such a theory of D-p rewriting, however, we believe that adhesive categories are mathematically elegant and less ad-hoc than previous proposals.

In order to back this claim and to further develop the theory of adhesive categories, we have demonstrated a number of useful properties. For instance, subobject union is formed as a pushout over the intersection, and subobject intersection distributes over subobject union. We have provided some closure properties which allow the construction of new adhesive categories from old. Any elementary topos is adhesive, but there are examples of adhesive categories which are not toposes. Adhesive categories include many well-known notions of graph structures used in computer science.

We believe that adhesive categories will be useful in the development of specific graphical models of computation and the development of semantic techniques for reasoning about such models. The rewriting theory needs to be developed further, with, for example, the construction of canonical dependency relations from derivations [11]. A possible direction for future work is to examine whether adhesive categories have enough structure so

that groupoidal relative pushouts [26] can be constructed in cospan bicategories over adhesive categories. Such cospan bicategories provide a way of understanding graphs in a modular fashion and will provide a general class of models which should include bigraphs [21] as examples. Another question to be resolved is whether demanding the good behaviour of pushouts only along some class of monomorphisms will result in further interesting categories.

Acknowledgement. The second author would like to thank Vladimiro Sassone for many discussions in the early stages of this project. Thanks also go to Marco Carbone, Paulo Oliva, Mogens Nielsen and especially Mikkel Nygaard for reading early drafts and providing many valuable comments and suggestions.

References

[1] P. Baldan, A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. Concurrent semantics of algebraic graph transformations. In H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 3, chapter 3, pages 107–187. World Scientific, 1999.

[2] R. Brown and G. Janelidze. Van Kampen theorems for categories of covering morphisms in lextensive categories. *J. Pure Appl. Algebra*, 119:255–263, 1997.

[3] A. Carboni, S. Lack, and R. F. C. Walters. Introduction to extensive and distributive categories. *Journal of Pure and Applied Algebra*, 84(2):145–158, February 1993.

[4] L. Cardelli. Bitonal membrane systems. Draft, 2003.

[5] L. Cardelli and A. D. Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structures, FOSSACS '98*. Springer Verlag, 1998.

[6] A. Corradini, H. Ehrig, R. Heckel, M. Lowe, U. Montanari, and F. Rossi. Algebraic approaches to graph transformation part i: Basic concepts and double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, pages 162–245. World Scientific, 1997.

[7] V. Danos and C. Laneve. Graphs for core molecular biology. In *International Workshop on Computational Methods in Systems Biology, CMSB '03*, 2003.

[8] H. Ehrig. Introduction to the algebraic theory of graph grammars. In *1st Int. Workshop on Graph Grammars*, Lecture Notes in Computer Science LNCS 73, pages 1–69. Springer Verlag, 1979.

[9] H. Ehrig, M. Gajewsky, and F. Parisi-Presicce. High-level replacement systems with applications to algebraic specificaitons and petri nets. In H. Ehrig, H.-J. Kreowsky, U. Montanari, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 3, chapter 6, pages 341–400. World Scientific, 1999.

[10] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. From graph grammars to high level replacement systems. In *1st Graph Grammar Workshop*, volume 73 of *Lecture Notes in Computer Science*, pages 1–69. Springer Verlag, 1991.

[11] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and concurrency in high-level replacement systems. *Math. Struct. in Comp. Science*, 1, 1991.

[12] H. Ehrig, M. Pfender, and H.J. Schneider. Graph-grammars: an algebraic approach. In *IEEE Conf. on Automata and Switching Theory*, pages 167–180, 1973.

[13] U. Montanari F. Gadducci. A concurrent graph semantics for mobile ambients. In *Mathematical Foundations of Programming Semantics MFPS '01*, volume 45 of *ENTCS*. Elsevier, 2001.

[14] F. Gadducci and R. Heckel. An inductive view of graph transformation. In *Workshop on Algebraic Development Techniques*, pages 223–237, 1997.

[15] J. C. Godskesen, T. Hildebrandt, and V. Sassone. A calculus of mobile resources. In *Int. Conf. on Concurrency Theory, CONCUR '02*, volume 2421 of *Lecture Notes in Computer Science*, pages 272–287. Springer, 2002.

[16] O. H. Jensen and R. Milner. Bigraphs and transitions. In *Principles of Programming Languages, POPL '03*. ACM Press, 2003.

[17] H.-J. Kreowski. Transformations of derivation sequences in graph grammars. In *Lecture Notes in Computer Science*, volume 56, pages 275–286, 1977.

[18] S. Lack and P. Sobociński. Toposes are adhesive. Manuscript, 2003.

[19] J. Lambek and P. J. Scott. *Introduction to higher order categorical logic*, volume 7 of *Cambridge studies in advanced mathematics*. Cambridge University Press, 1986.

[20] J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In *Int. Conf. on Concurrency Theory, CONCUR '00*, Lecture Notes in Computer Science, pages 243–258. Springer, 2000.

[21] R. Milner. Bigraphical reactive systems: Basic theory. Technical Report 523, Computer Laboratory, University of Cambridge, 2001.

[22] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, (Parts I and II). *Information and Computation*, 100:1–77, 1992.

[23] U. Montanari, M. Pistore, and F. Rossi. Modelling concurrent, mobile and coordinated systems via graph transformations. In H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 3, chapter 4, pages 189–268. World Scientific, 1999.

[24] J. Power and K. Tourlas. An algebraic foundation for graph-based diagrams in computing. In *Mathematical Foundations of Programming Semantics MFPS '01*, volume 45 of *Electronic Notes in Computer Science*. Elsevier, 2001.

[25] V. Sassone and P. Sobociński. Deriving bisimulation congruences: 2-categories vs. precategories. In *Foundations of Software Science and Computation Structures FOSSACS '03*, volume 2620 of *Lecture Notes in Computer Science*. Springer Verlag, 2003.

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

Recent BRICS Report Series Publications

RS-03-31 Stephen Lack and Paweł Sobociński. *Adhesive Categories*. October 2003. 25 pp.

RS-03-30 Jesper Makholm Byskov, Bolette Ammitzbøll Madsen, and Bjarke Skjernaa. *New Algorithms for Exact Satisfiability*. October 2003. 31 pp.

RS-03-29 Aske Simon Christensen, Christian Kirkegaard, and Anders Møller. *A Runtime System for XML Transformations in Java*. October 2003. 15 pp.

RS-03-28 Zoltán Ésik and Kim G. Larsen. *Regular Languages Definable by Lindström Quantifiers*. August 2003. 82 pp. This report supersedes the earlier BRICS report RS-02-20.

RS-03-27 Luca Aceto, Willem Jan Fokkink, Rob J. van Glabbeek, and Anna Ingólfsdóttir. *Nested Semantics over Finite Trees are Equationally Hard*. August 2003. 31 pp.

RS-03-26 Olivier Danvy and Ulrik P. Schultz. *Lambda-Lifting in Quadratic Time*. August 2003. 23 pp. Extended version of a paper appearing in Hu and Rodríguez-Artalejo, editors, *Sixth International Symposium on Functional and Logic Programming*, FLOPS '02 Proceedings, LNCS 2441, 2002, pages 134–151. This report supersedes the earlier BRICS report RS-02-30.

RS-03-25 Biernacki Dariusz and Danvy Olivier. *From Interpreter to Logic Engine: A Functional Derivation*. June 2003.

RS-03-24 Mads Sig Ager, Olivier Danvy, and Jan Midtgaard. *A Functional Correspondence between Call-by-Need Evaluators and Lazy Abstract Machines*. June 2003. 13 pp.

RS-03-23 Korovin Margarita. *Recent Advances in Σ-Definability over Continuous Data Types*. June 2003. 26 pp.

RS-03-22 Ivan B. Damgård and Mads J. Jurik. *Scalable Key-Escrow*. May 2003. 15 pp.

RS-03-21 Ulrich Kohlenbach. *Some Logical Metatheorems with Applications in Functional Analysis*. May 2003. 55 pp. Slighly revised and extended version to appear in *Transactions of the American Mathematical Society*.