A Fibred Approach to Rewriting

—

How the Duality between Adding and Deleting
Cooperates with the Difference between Matching and Rewriting

WOLFRAM KAHL

Fakultät für Informatik
Universität der Bundeswehr München

Technical Report Nr. 9702

Abstract

We present a new approach to rewriting obtained by enhancing and unifying existing variants inside the algebraic (or better categorical) approach to (graph) rewriting. Our approach is motivated by second-order term graph rewriting and stresses on one hand the two-step nature of rule application consisting of deleting and adding items and on the other hand the heterogeneous nature of the rewriting setup where rule steps should be clearly distinguished from matching of rule sides into redexes.

Complementing the existing opfibration approach with a dual fibration step turns out to yield a natural and flexible approach with useful new applications. The resulting fibred approach takes advantage of the heterogeneous setting and appropriately reflects the duality between deleting and adding in the course of rewriting, in contrast with the double-pushout approach which simplifies this duality into a symmetry. An important contribution is the universal characterisation of the host object, which has to be found as a pushout-complement in the double pushout approach.

The fibred approach is presented in abstract and independent from any concrete application categories in the manner of High-Level Replacement Systems. Our original motivation for the development of the fibred approach comes from term graphs with bound variables where all other approaches failed; in this paper we present an unusual view on term rewriting as running example.

Contents

1. Background and Related Work

Abstract theoretical consideration of rewriting first came up in connection with Church's λ-calculus [Church, 1941] and Curry's Combinatory Logic [Curry, 1930]. There the object of interest is the *rewrite relation*, the properties of which may be studied independent of the underlying domain of rewritten objects, and theorems can be derived about properties of *derivations*, i.e., *sequences* of rewriting steps.

Abstract consideration of *single* rewriting steps, i.e., an abstract specification of how rewriting rules are to be applied, can be found most prominently in the **algebraic approach** to (graph) rewriting of [Ehrig *et al.*, 1973; Ehrig, 1978]. There, the starting point is to consider a category[1] with graphs as objects and appropriate graph homomorphisms as arrows, and then the category-theoretic pushout construction is employed for the information transfer.

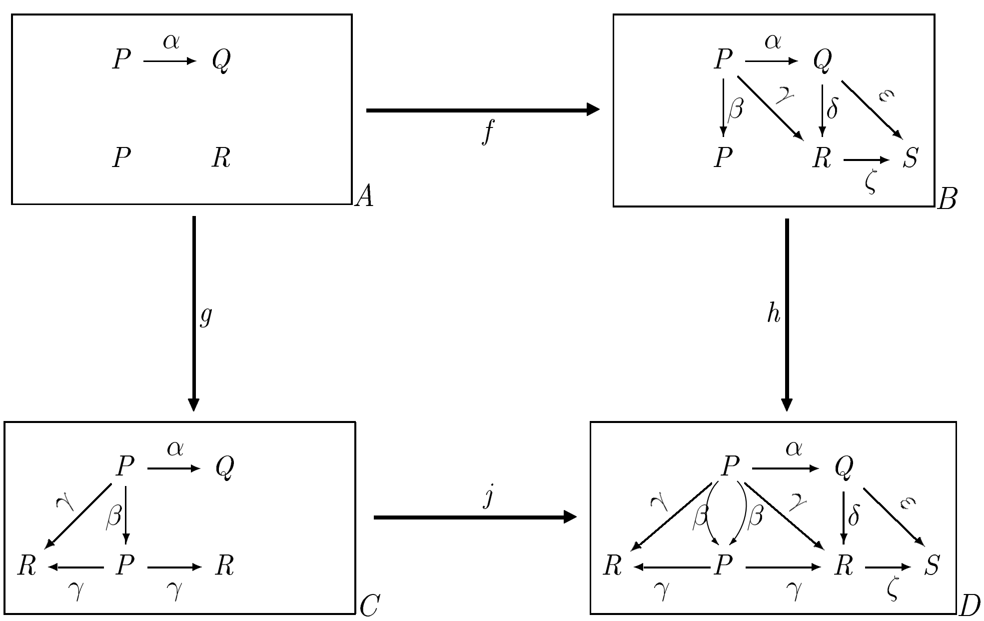
In the survey and tutorial paper [Ehrig *et al.*, 1990b], the double-pushout approach to (graph) rewriting is put forward as resulting naturally from the perceived *symmetry* of *adding* and *deleting* parts of the graph in the course of rule application.

But is there really a symmetry in general? As we see it, this is not the case. We argue that there is only a kind of duality between adding and deleting in rule applications. In this section, we shall provide the necessary background for subsequently substantiating this claim. Therefore we first review the foundations, development and variety of the algebraic approach to rewriting.

First let us try to provide some grip on the fundamental concept of pushouts. We provide the standard category theoretic definition and illustrate it with an example pushout on graphs:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "g"'] & B \arrow[d, "h"] \\ C \arrow[r, "j"] & D \end{tikzcd}$$

In a category, for three objects A, B, and C and two arrows $f : A \to B$ and $g : A \to C$, a **pushout** is an object D together with two arrows $h : B \to D$ and $j : C \to D$ with $f \mathbin{;} h = g \mathbin{;} j$, such that for every object D' together with two arrows $h' : B \to D'$ and $j' : C \to D'$ with $f \mathbin{;} h' = g \mathbin{;} j'$ there is a unique arrow $u : D \to D'$ such that $h \mathbin{;} u = h'$ and $j \mathbin{;} u = j'$. So a pushout can be seen as a *join* of the information in B and C along the interface A as embedded by f and g. An example for a pushout in the category of node- and edge-labelled graphs is the following:



[1] Category theory is a branch of abstract mathematics that offers a unified treatment of certain constellations that occur in many different mathematical subjects; a category consists of *objects* that can be understood as generalised sets and *arrows* or *morphisms* that can be understood as generalised functions between sets; every object has an identity arrow and the essential operation in a category is composition of arrows which has to be associative with identities as left and right units. See Section 3 for detailed definitions.

Here we may observe that there is only *one* image of the interface graph A — this is the result of the commutativity condition $f \fatsemi h = g \fatsemi j$. The parts of B and C *outside* the image of the interface graph A have *both* been copied *separately* into the pushout object and no other items have been introduced — this is the result of the universality condition that selects the so-to-speak minimal commutative square. So the graph A together with its embeddings into B and C serves as the *interface* for *gluing together* the two graphs B and C.

For a detailed introduction to the use of the pushout concept in graph rewriting see [Ehrig *et al.*, 1990b]. Here we only shortly recall the historical development.

The original rewriting pattern is the **double pushout** with L as *left-hand side* of the *pushout rule*, R as *right-hand side* and G as *gluing graph* of the pushout rule. Furthermore A is the *application graph*, H the *host graph* and B the *result graph*. We shall frequently subsume the gluing graph and the host graph under the name of **intermediate** graphs. A given pushout rule is applicable to a given application graph via a given matching $L \to A$ if a host graph H, i.e. a **pushout complement**, exists. For this the following two sets are considered:

$$\begin{tikzcd} L \arrow[d] & G \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\ A & H \arrow[l] \arrow[r] & B \end{tikzcd}$$

DANGLING: the set of all nodes in L that have images in A with predecessors outside the image of L.

IDENTIFICATION: the set of all nodes in L, that in A are identified with other nodes of L.

Then the so-called **gluing condition** can be formulated:

$$\mathit{DANGLING} \cup \mathit{IDENTIFICATION} \subseteq G_L \quad ,$$

i.e., nodes contained in one of these two sets have to be in the image G_L of G in L. At least the homomorphism from G to L is usually required to be a monomorphism (i.e., injective). An important result of the double-pushout approach is that existence of a pushout complement H is equivalent to the validity of the gluing condition.

Raoult[1984] and Kennaway [1987; 1990] and independently Löwe [1990] developed the approach of the **single pushout** by moving to the corresponding category of graphs with partial morphisms $\dashrightarrow$. The only application condition is that the embedding of the left-hand rule-side into the application graph is a total morphism; no gluing condition is required anymore (except when one desires the morphism from the right-hand side to the result graph to be total, too). The rewriting mechanism resulting from single pushouts is strictly more powerful than that using double pushouts.

$$\begin{tikzcd} L \arrow[r, dotted] \arrow[d] & R \arrow[d, dotted] \\ A \arrow[r, dotted] & B \end{tikzcd}$$

For moving away from graphs to wider application areas, the foundations of the algebraic approach up to double- and single-pushout derivations have been abstracted away from the category of graphs under the name of "High Level Replacement Systems", axiomatically describing requirements on the underlying categories for the different aspects of algebraic rewriting to work, see [Ehrig *et al.*, 1990a; Padberg, 1993] — our approach uses this independence from concrete underlying categories, too. Main motivation for this generalisation was the application of the algebraic rewriting approach to algebraic specifications, which also served as initial motivation for the next extension of the algebraic approach:

A double pushout with an additional morphism tacked onto the lower left corner was introduced by Ehrig and Parisi-Presicce [1991; 1993] under the name of "**restricting derivations**". Parisi-Presicce [1993] showed that this approach is strictly more powerful then the single-pushout approach.

$$\begin{tikzcd} & L \arrow[d] & G \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\ A & A' \arrow[l] & H \arrow[l] \arrow[r] & B \end{tikzcd}$$

One of the motivations [Kennaway, 1987] for developing the single pushout had already come from the context of algebraic modelling of the graph rewriting language DACTL [Glauert *et al.*, 1990].

$$\begin{tikzcd} L \arrow[r] \arrow[d] & R \arrow[d] \\ A \arrow[r] & B \end{tikzcd}$$

As a further development of this approach, Banach [1993a; 1994] undertook a modelling through **opfibrations** — the bold arrows are mappings from the rule sides to elements of their index categories. The advantage of this approach is the introduction of heterogeneity, since the rule morphisms need not come from the same category as the embeddings from rule sides into application and result graphs. We shall call a square like this one a **heterogeneous pushout** (in Def. 5.1) since it obeys essentially the same conditions as a pushout.

The **fibred approach** we are going to present in this paper is motivated from applications to term graphs with bound variables [Kahl, 1996] where all other variants of the algebraic approach failed, and the fibred approach enabled the first successful treatment of graph reduction of λ-graphs inside the algebraic approach to rewriting. Nevertheless we shall present the fibred approach in a way that is independent from the underlying categories, in the spirit of high-level replacement systems. That way it is easier to try to apply this approach to completely different application areas.

For reasons of complexity we do not motivate our development with the original term graph examples of [Kahl, 1996] in this paper, but resort to simple term rewriting examples. On one hand, this may seem to be a little contrived in a few places, but we still can reflect many effects from term graphs to terms by accordingly adapting the definitions, so terms can still appropriately illustrate the issues at hand, and terms are much simpler to deal with in full detail than term graphs — the definitions necessary for term graphs would obliterate the real subject of this paper. On the other hand, our approach also opens up another categorical treatment of term rewriting that, unlike 2-categories or sesqui-categories [Stell, 1994], cleanly extends to term graph rewriting, — the short excursion in [Gaducci, 1996] to term graphs clearly shows the limitations of that kind of approach.

Organisation of this Paper

In Section 2 we start with informally motivating our approach through exploring an innovative treatment of term rewriting in the algebraic approach starting from heterogeneous pushouts as a running example.

For the formal presentation of our approach, we then first fix our category theoretic notations and list the perhaps not universally familiar definitions of fibrations and opfibrations in Section 3.

In Section 4 we begin with laying down the framework of categories inside which we discuss rewriting, introducing a pattern category for the rules and categories of matchings for redex arrows.

Inside this setting we then give a formal definition of heterogeneous pushouts in Section 5 and establish simple connections between this concept and that of opcartesian arrows.

In Section 6 we first perform the Grothendiek construction along the lines of [Banach, 1994], but adapted to our more abstract setting, and then we use the Grothendiek category to extract a useful subcategory from our original matching category such that the *pattern functor* is a split opfibration for that subcategory.

Section 7 explicitly states the dual definition to the heterogeneous pushout, the definition of *hosts*. Because of the duality, the developments of sections 5 and 6 need not be repeated for arriving at the corresponding results.

We use these to put everything together into our *fibred approach to rewriting* in Section 8.

2. A Double-Heterogeneous-Pushout Approach?

Banach's argument for heterogeneity in his opfibration approach is that rule transitions are in their essence different from redexes, i.e. embeddings of rule sides into application graphs. We share this point of view, and we have arrived at the same conclusion in our work [Kahl, 1996] that opens the algebraic approach to rewriting of term graphs with bound variables.

Rewriting in [Banach, 1993a; Banach, 1994] is achieved by single-heterogeneous-pushout steps; "vertical" arrows are homomorphisms of his simple kind of term graphs and "horizontal" arrows are so-called "redirection pairs", i.e. pairs consisting of two functions from nodes of the left-hand side into the right-hand side: an embedding of the left-hand side into the right-hand side (Banach feels "garbage retention" to be essential for the feasibility of his approach) and a partial redirection function. These "redirection pairs" can be seen as derived from a special kind of partial arrow — remember that the same DACTL project was also one of the origins of the single-pushout approach.

The problem with partial arrows is that they hide a subobject the properties of which may not be well-understood. In fact, with indirectly or ad-hoc defined "partial arrows" such a subobject might only exist in an extension of the original category. Therefore a double-heterogeneous-pushout approach with explicit intermediate objects appears to be much more feasible for more complicated structures.

As an example consider a simple version of term rewriting — for reasons mainly of space we do not delve into the details of term trees or term graphs here.

Running Example: Term Rewriting

Even restricting ourselves only to first-order term rewriting, we shall need a certain kind of second-order substitution for the intermediate steps, and therefore we give rather detailed definitions for getting the example onto its way; we flag the definitions, lemmata and theorems used for this running example with "**TRS**" to keep them well distinguished from the presentation of the fibred approach itself, which is the main topic of this paper.

TRS Definition 2.1. A **term** is are either a **term variable** $x, y, z, \ldots$, or, for n terms $t_1, \ldots, t_n$ (with $n \geq 0$) and an n-ary function symbol f the construct $f(t_1, \ldots, t_n)$.

A **term substitution** is a partial function with finite domain from term variables to terms.

Application of a term substitution σ **to a term** t is defined as usual and written $\sigma\ t$.

A **context** is a term with *one* occurrence of a **hole** "[ ]" *instead of* a term variable, and *context substitution* "$C[t]$" substitutes the term t for the hole inside the context $C[\,]$.

The **application of a term substitution** σ **to a context** $C[\,]$, written $(\sigma\ C)[\,]$, is the term $\sigma\ (C[\,])$ considered again as a context.

Composition $\sigma \mathbin{⨾} \tau$ of two term substitutions σ and τ is defined by[2]

$$\sigma \mathbin{⨾} \tau := \{(x \mapsto t) : \sigma \bullet x \mapsto \tau\ t\} \cup \{(x \mapsto t) : \tau \mid x \notin \operatorname{dom} \sigma\} \ .$$

A **term homomorphism** from a term t_1 to another term t_2 is a triple $(t_1, C[\,], \sigma)$ consisting of its domain t_1, a context $C[\,]$ and a substitution σ such that $t_2 = C[\sigma\ t_1]$. □

It is not hard to see that this concept of term homomorphisms is the concept that is commonly used for finding redexes of term rewriting rules, and that this way we even obtain a category of terms with term homomorphisms as arrows. However, this category has only very few pushouts, since pushouts in terms amount to unification. There are also no terms that could be used for the intermediate rôles in the rewriting process, i.e., no gluing terms and no host terms. For this purpose we propose an extension of terms with placeholder variables — we choose the names "placeholder variables" etc. instead of names like "context variables" etc., since we already need contexts in another rôle in term homomorphisms. Parts of the following definition are very technical and may seem to be pretty unusual, but all will be motivated by their use, some only in later sections of this paper.

---

[2] We consider (partial) functions as sets of pairs and we write such a pair "$x \mapsto y$".

For set-comprehension we employ Z-notation [Spivey, 1989], which explicitely introduces the local variables and uses the pattern "{ *signature* | *predicate* • *term* }" instead of the otherwise frequently observed more implicit pattern "{ *term* | *predicate* }". So we have for example $\{n : \mathbb{N} \mid n < 4 \bullet n^2\} = \{0, 1, 4, 9\}$. If the *predicate* is constantly true, then we can also just write "{ *signature* • *term* }" in accordance to common Z usage, and similarly, if the *term* is just the only variable of the *signature*, we can omit the *term*, as in $\{n : \mathbb{N} \mid n < 4\} = \{0, 1, 2, 3\}$.

TRS Definition 2.2. An **extended term** is are either a **term variable** $x, y, z, \ldots$, or, for n extended terms $t_1, \ldots, t_n$ (with $n \geq 0$) and an n-ary function symbol f the construct $f(t_1, \ldots, t_n)$, or, for n extended terms $t_1, \ldots, t_n$ (with $n \geq 0$) and an n-ary **placeholder variable** U the construct $U[t_1, \ldots, t_n]$.

A **dummy term** is defined as an extended term, but in the place of term variables alternatively there can also be **dummy variables** $\delta_1, \delta_2, \ldots$

A **dummy pattern** is a dummy term $U[\delta_{i_1}, \ldots, \delta_{i_n}]$ consisting only of one n-ary placeholder variable U and n distinct dummy variables $\delta_{i_1}, \ldots, \delta_{i_n}$ such that $1 \leq j < k \leq n$ implies $i_j < i_k$.

An **extended dummy pattern** is a dummy pattern with terms (not extended terms) interspersed between the dummy variables, as in $U_5[\delta_1, f(a,b), x, \delta_2, y]$.

A **placeholder substitution** is a partial function with finite domain from dummy patterns into dummy terms; in the image of a dummy pattern only those dummy variables may occur that already occur in the pattern, and if the image contains placeholders, then it has to be an extended dummy pattern.

A placeholder substitution is called **full** iff for every pattern p in its domain the image associated with p contains *all* the dummy variables that occur in p.

Application "$\varphi\ t$" of a placeholder substitution φ to an extended term t is defined essentially as usual substitution; only for $t = U_n[t_1, \ldots, t_n]$ if there is a dummy pattern $U_n[\alpha, \ldots, \nu]$ in the domain of φ with image r, then it is defined via substitution for the dummy variables in the image: $\varphi\ t = \{\alpha \mapsto t_1, \ldots, \nu \mapsto t_n\}\ r$.

Application of term substitutions is naturally extended to extended terms.

Application "$\sigma\ \varphi$" of a term substitution σ to a placeholder substitution φ is defined as the placeholder substitution where σ has been applied to all images: $\sigma\ \{p_1 \mapsto e_1, \ldots, p_n \mapsto e_n\} := \{p_1 \mapsto \sigma\ e_1, \ldots, p_n \mapsto \sigma\ e_n\}$.

Composition $\varphi ⨾ \psi$ of two placeholder substitutions φ and ψ is defined by

$$\varphi ⨾ \psi := \{(x \mapsto t) : \varphi \bullet x \mapsto \psi\ t\} \cup \{(x \mapsto t) : \psi \mid x \notin \operatorname{dom} \varphi\} \ .$$

A **placeholder arrow** from the extended term t_1 to the extended term t_2 is a pair (t_1, φ) consisting of its domain and a *full* placeholder substitution φ such that $t_2 = \varphi\ t_1$. □

With term homomorphisms for the vertical arrows and placeholder arrows for the horizontal arrows we can throw together what already looks like a double-heterogeneous-pushout diagram. We show an example rewriting step involving the rule of commutativity of multiplication "$*$" — the context and substitution are the same for all three vertical arrows:

$$\begin{tikzcd}
*(x,y) \arrow[d, "{(f(1,[\ ])\,,\ \{x\mapsto 3, y\mapsto 4\})}"] & U_2[x,y] \arrow[l, "{\{U_2[\delta_1,\delta_2]\mapsto *(\delta_1,\delta_2)\}}"'] \arrow[r, "{\{U_2[\delta_1,\delta_2]\mapsto *(\delta_2,\delta_1)\}}"] \arrow[d] & *(y,x) \arrow[d] \\
f(1,*(3,4)) & f(1, U_2[3,4]) \arrow[l, "{\{U_2[\delta_1,\delta_2]\mapsto *(\delta_1,\delta_2)\}}"'] \arrow[r, "{\{U_2[\delta_1,\delta_2]\mapsto *(\delta_2,\delta_1)\}}"] & f(1,*(4,3))
\end{tikzcd}$$

Still missing is of course a composition of term homomorphisms and placeholder arrows to a kind of "mixed substitutions" for making the diagram commute. For this purpose we subsume both kinds of arrows into one generalised definition of morphisms between extended terms:

TRS Definition 2.3. An **extended term morphism** from an extended term t_1 to another extended term t_2 is a quadruple $(t_1, C[\ ], [\sigma]_{t_1}, [\varphi]_{(\sigma\ t_1)})$ consisting of its domain t_1, a context $C[\ ]$, an equivalence class of term substitutions $[\sigma]_{t_1}$ relative to the equivalence relation $\equiv_{t_1}$ which is defined by $\sigma \equiv_{t_1} \tau :\Leftrightarrow \sigma\ t_1 = \tau\ t_1$, and an equivalence class of *full* placeholder substitutions $[\varphi]_{(\sigma\ t_1)}$, such that $t_2 = C[\varphi'(\sigma'\ t_1)]$ for every $\sigma' \in [\sigma]_{t_1}$ and for every $\varphi' \in [\varphi]_{(\sigma\ t_1)}$.

Composition of two extended term morphisms is defined by

$$\begin{aligned}
&(t_1, C[\ ], [\sigma]_{t_1}, [\varphi]_{(\sigma\ t_1)}) ⨾ (t_2, D[\ ], [\tau]_{t_2}, [\psi]_{(\tau\ t_2)}) \\
&\qquad := (t_1, D[(\tau\ C)[\ ]], [\sigma ⨾ \tau]_{t_1}, [(\tau\ \varphi) ⨾ \psi]_{((\sigma⨾\tau)\ t_1)}) \ .
\end{aligned}$$
□

It is easy to see how term homomorphisms and placeholder arrows can be mapped to corresponding extended term morphisms — in a different approach, we could have used equivalence classes of substitutions already in term homomorphisms and placeholder arrows, turning this mapping into an injective embedding, but it would not have made any difference for the remaining development.

Before proving well-definedness of this composition we first collect a few technical properties. For term substitutions we shall need the following laws:

TRS Lemma 2.4. Let $C[\,]$ be a variable for contexts, t, t_1, t_2 for extended terms, σ, τ for term substitutions and φ, ψ for placeholder substitutions.

i) Composition of term substitutions is well-defined wrt. application to (extended) terms, contexts and placeholder substitutions, i.e., in any case we have $(\sigma \mathbin{;} \tau)\ x = \tau\ (\sigma\ x)$; therefore composition of term substitutions is also associative.

ii) Term substitutions distribute over contexts: $\sigma\ (C[t]) = (\sigma\ C)\ [\sigma\ t]$.

iii) If $(\sigma\ C)[t_1] = (\tau\ C)[t_2]$, then also $(\sigma\ C)[\,] = (\tau\ C)[\,]$ and $t_1 = t_2$.

iv) Term substitutions distribute over placeholder substitutions: $\sigma\ (\varphi\ t) = (\sigma\ \varphi)\ (\sigma\ t)$.

Proof:

i) follows from the definitions of composition and application of term substitution;

ii) is obvious from the definitions of application of term substitutions;

iii) holds since term substitutions cannot change positions of subterms;

iv) follows from the absence of placeholders in the image of term substitutions. □

For placeholder substitutions and for their interplay with term substitutions we shall need even more laws:

TRS Lemma 2.5. Let $C[\,]$ be a variable for contexts, t, t_1, t_2 for extended terms, σ, τ for term substitutions and φ, ψ for placeholder substitutions.

i) Composition of placeholder substitutions is well-defined wrt. application to extended terms, i.e., for every extended term t we have $(\varphi \mathbin{;} \psi)\ t = \psi\ (\varphi\ t)$; therefore composition of placeholder substitutions is also associative.

ii) Placeholder substitutions commute with contexts: $\varphi\ (C[t]) = C[\varphi\ t]$.

iii) If φ is a full placeholder substitution, then $\sigma\ \varphi$ is a full placeholder substitution, too.

iv) If φ is a full placeholder substitution, then $\sigma\ (\varphi\ t) = \tau\ (\varphi\ t)$ implies $\sigma\ t = \tau\ t$.

v) If φ is a full placeholder substitution, then $\varphi\ (\sigma\ t) = \tau\ t$ implies $\varphi\ (\sigma\ t) = \sigma\ t$.

Proof:

i) follows from the definitions of composition and application of placeholder substitution;

ii) follows from the absence of placeholders in contexts;

iii) holds since the application of σ cannot remove any dummy-variable occurrences from the image patterns;

iv) holds since application of a full placeholder substitution cannot delete any term variables from t.

v) Obviously the image under φ of every dummy pattern p for a placeholder variable U occurring in t has to be an extended dummy pattern containing the same placeholder variable U; since φ is full and because of the restrictive definition of dummy patterns the image then has to be the *same* dummy pattern p. But substitution elements of the shape $p \mapsto p$ have no effect, so φ has the same effect on $\sigma\ t$ as the empty placeholder substitution. □

It is, however, not in general the case that $\varphi\ (\sigma\ t) = \psi\ (\sigma\ t)$ implied $\varphi\ t = \psi\ t$, just consider the example of $t = U_1[x]$, $\sigma = \{x \mapsto a\}$, $\varphi = \{U_1[\delta_1] \mapsto p(\delta_1, a)\}$ and $\psi = \{U_1[\delta_1] \mapsto p(a, \delta_1)\}$, where the former holds, but the latter not. This is the reason why equivalence classes of placeholder substitutions in extended morphisms (Def. 2.3) are relative to $\sigma\ t_1$ instead of just to t_1.

Also, for a placeholder substitution φ which is not full, $\sigma\ (\varphi\ t) = \tau\ (\varphi\ t)$ does not necessarily imply $\sigma\ t = \tau\ t$, just consider $t = U_1[x]$, $\varphi = \{U_1[\delta_1] \mapsto q\}$, $\sigma = \{x \mapsto a\}$ and $\tau = \{x \mapsto b\}$, where $\sigma\ (\varphi\ t) = q = \tau\ (\varphi\ t)$, but $\sigma\ t = U_1[a] \neq U_1[b] = \tau\ t$.

We still have to show that composition is well-defined: Consider two extended term morphisms $(t_1, C[\ ], [\sigma]_{t_1}, [\varphi]_{(\sigma\ t_1)})$ from t_1 to t_2 and $(t_2, D[\ ], [\tau]_{t_2}, [\psi]_{(\tau\ t_2)})$ from t_2 to t_3. For arbitrary $\sigma' \in [\sigma]_{t_1}$, $\tau' \in [\tau]_{t_2}$, $\varphi' \in [\varphi]_{\sigma\ t_1}$ and $\psi' \in [\psi]_{\tau\ t_2}$ we than have $\sigma'\ t_1 = \sigma\ t_1$ and $\tau'\ t_2 = \tau\ t_2$, and $\varphi'(\sigma\ t_1) = \varphi(\sigma\ t_1)$ and $\psi'\ (\tau\ t_2) = \psi\ (\tau\ t_2)$.

The equation $\tau'\ t_2 = \tau\ t_2$ implies via

$$\begin{aligned}
(\tau\ C)[(\tau\varphi)\ (\tau(\sigma\ t_1))] &= \tau\ (C[\varphi\ (\sigma\ t_1)]) && \text{2.4.ii), 2.4.iv)}\\
&= \tau\ t_2 \ = \ \tau'\ t_2 && \text{2.3: } t_2 = C[\varphi\ (\sigma\ t_1)]\\
&= \tau'\ (C[\varphi\ (\sigma'\ t_1)]) && \text{2.3: } t_2 = C[\varphi\ (\sigma'\ t_1)]\\
&= (\tau'\ C)[(\tau'\varphi)\ (\tau'(\sigma'\ t_1))] && \text{2.4.ii), 2.4.iv)}
\end{aligned}$$

and 2.4.iii), iii), iv) among other facts that $\tau\ (\sigma\ t_1) = \tau'\ (\sigma'\ t_1)$ and therefore $(\sigma ⨾ \tau) \in [\sigma ⨾ \tau]_{t_1}$. In much the same way, we obtain:

$$\begin{aligned}
& (\tau'\ C)[((\tau'\varphi') ⨾ \psi')\ (\tau'(\sigma'\ t_1)))] \\
=\ & (\tau'\ C)[\psi'\ ((\tau'\varphi')\ (\tau'(\sigma'\ t_1)))] && \text{2.5.i)}\\
=\ & \psi'\ ((\tau'\ C)[(\tau'\varphi')\ (\tau'(\sigma'\ t_1))]) && \text{2.5.ii)}\\
=\ & \psi'\ (\tau'\ (C[\varphi'(\sigma'\ t_1)])) && \text{2.4.ii), 2.4.iv)}\\
=\ & \psi'\ (\tau'\ t_2) && \text{2.3: } t_2 = C[\varphi'\ (\sigma'\ t_1)]\\
=\ & \psi\ (\tau\ t_2) && \psi' \in [\psi]_{\tau\ t_2}, \tau' \in [\tau]_{t_2}\\
=\ & \psi\ (\tau\ (C[\varphi(\sigma\ t_1)])) && \text{2.3: } t_2 = C[\varphi\ (\sigma\ t_1)]\\
=\ & \psi\ ((\tau\ C)[(\tau\varphi)\ (\tau(\sigma\ t_1))]) && \text{2.4.ii), 2.4.iv)}\\
=\ & (\tau\ C)[\psi\ ((\tau\varphi)\ (\tau(\sigma\ t_1)))] && \text{2.5.ii)}\\
=\ & (\tau\ C)[((\tau\varphi) ⨾ \psi)\ (\tau(\sigma\ t_1)))] && \text{2.5.i)}
\end{aligned}$$

This implies with $\tau(\sigma\ t_1) = \tau'(\sigma'\ t_1)$ from above and with 2.4.iii) that that the placeholder substitutions, too, are in the same equivalence class: $(\tau'\varphi') ⨾ \psi' \in [(\tau\ \varphi) ⨾ \psi]_{(\sigma⨾\tau)\ t_1}$, thus justifying our use of the representatives for the proof of well-definedness of composition:

$$\begin{aligned}
t_3 &= D[\psi(\tau\ t_2)]\\
&= D[\psi(\tau(C[\varphi(\sigma\ t_1)]))] && \text{2.3: } t_2 = C[\varphi\ (\sigma\ t_1)]\\
&= D[\psi((\tau\ C)[\tau(\varphi(\sigma\ t_1))])] && \text{2.4.ii)}\\
&= D[(\tau\ C)[\psi((\tau\varphi)(\tau(\sigma\ t_1)))]] && \text{2.5.ii), 2.4.iv)}\\
&= D[(\tau\ C)[((\tau\varphi) ⨾ \psi)((\sigma ⨾ \tau)\ t_1)]]\ . && \text{2.4.i), 2.5.i)}
\end{aligned}$$

Associativity is now easily shown:

$$\begin{aligned}
& \Big((t_1, C[\ ], [\sigma]_{t_1}, [\varphi]_{(\sigma\ t_1)}) ⨾ (t_2, D[\ ], [\tau]_{t_2}, [\psi]_{(\tau\ t_2)})\Big) ⨾ (t_3, E[\ ], [\upsilon]_{t_3}, [\chi]_{(\upsilon\ t_3)})\\
=\ & (t_1, D[(\tau\ C)[\ ]], [\sigma ⨾ \tau]_{t_1}, [(\tau\ \varphi) ⨾ \psi]_{((\sigma⨾\tau)\ t_1)}) ⨾ (t_3, E[\ ], \upsilon, \chi)\\
=\ & (t_1, E[(\upsilon\ D)[((\tau ⨾ \upsilon)\ C)[\ ]]], [(\sigma ⨾ \tau) ⨾ \upsilon]_{t_1}, [(\upsilon\ ((\tau\ \varphi) ⨾ \psi)) ⨾ \chi]_{(((\sigma⨾\tau)⨾\upsilon)\ t_1)})\\
=\ & (t_1, E[(\upsilon\ D)[((\tau ⨾ \upsilon)\ C)[\ ]]], [\sigma ⨾ (\tau ⨾ \upsilon)]_{t_1}, [((\tau ⨾ \upsilon)\ \varphi) ⨾ (\upsilon\ \psi) ⨾ \chi]_{(\sigma⨾(\tau⨾\upsilon)\ t_1)})\\
=\ & (t_1, C[\ ], [\sigma]_{t_1}, [\varphi]_{(\sigma\ t_1)}) ⨾ (t_2, E[(\upsilon\ D)[\ ]], [\tau ⨾ \upsilon]_{t_2}, [(\upsilon\ \psi) ⨾ \chi]_{(\tau\ t_2)})\\
=\ & (t_1, C[\ ], [\sigma]_{t_1}, [\varphi]_{(\sigma\ t_1)}) ⨾ \Big((t_2, D[\ ], [\tau]_{t_2}, [\psi]_{(\tau\ t_2)}) ⨾ (t_3, E[\ ], [\upsilon]_{t_3}, [\chi]_{(\upsilon\ t_3)})\Big)\ .
\end{aligned}$$

For an extended term t, the identity associated with it is $(t, [\ ], [\{\}]_t, [\{\}]_t)$, and the identity equations are easily verified. Thus we have proven:

TRS Theorem 2.6. Extended terms with extended term morphisms form a category. □

It is now easy to see informally that the term rewriting step given above actually is a double-heterogeneous-pushout rewriting step:

TRS Proposition 2.7. For the image $d := (P, [\ ], [\{\}]_P, [\varphi]_P)$ of a placeholder arrow (P, φ) from P to Q and the image $g := (P, C[\ ], [\sigma]_P, [\{\}]_{(\sigma\ P)})$ of a term homomorphism $(P, C[\ ], \sigma)$ from P to G (respectively their embeddings in extended term morphisms), a heterogeneous pushout is given by the image $e := (G, [\ ], [\{\}]_G, [\sigma\ \varphi]_G)$ of the placeholder arrow $(G, \sigma\ \varphi)$ from G to some object H and the image $h := (Q, C[\ ], [\sigma]_Q, [\{\}]_{(\sigma\ Q)})$ of the term homomorphism $(Q, C[\ ], \sigma)$ from Q to H.

$$\begin{tikzcd}
P \arrow[r, "{d=(P,[\ ],[\{\}]_P,[\varphi]_P)}"] \arrow[d, "{g=(P,C[\ ],[\sigma]_P,[\{\}]_{(\sigma\ P)})}"'] & Q \arrow[d, "{h=(Q,C[\ ],[\sigma]_Q,[\{\}]_{(\sigma\ Q)})}"] \\
G \arrow[r, "{e=(G,[\ ],[\{\}]_G,[\sigma\ \varphi]_G)}"] & H
\end{tikzcd}$$

Proof: Here we only show commutativity, for the full proof see Theorem 5.2. We have

$$d \fatsemi h = (P, C[\ ], [\sigma]_P, [\sigma\ \varphi]_{(\sigma\ P)}) = g \fatsemi e\ .$$ □

But beware! Not all term rewrite rules are modelled so easily. Pursuing this approach, we run into difficulties pretty fast; just consider the **projection rule** $\pi(p(x, y)) \leftarrow U_1[x] \rightarrow x$.[3] Here we cannot construct a double-heterogeneous-pushout rewriting step; we have to resort to the *restricting derivations* of [Ehrig and Parisi-Presicce, 1991]:

$$\begin{tikzcd}
 & \pi(p(x,y)) \arrow[dl] \arrow[d] & U_1[x] \arrow[l] \arrow[r] \arrow[d] & x \arrow[d] \\
\pi(p(t,s)) & \pi(p(t,y)) \arrow[l] & U_1[t] \arrow[l] \arrow[r] & t
\end{tikzcd}$$

The additional horizontal arrow may — in graphs — serve to get rid of *DANGLING* and *IDENTIFICATION* nodes before constructing a pseudo-pushout complement. In term graphs it allows "garbage avoidance" [Kahl, 1996] and garbage collection to be implemented directly in the rewriting step. The same effect could probably be achieved via a single-heterogeneous-pushout with partial morphisms, but at the expense of hiding the extension of the category to comprise extended terms with placeholder variables.

Universal Characterisation of the Host Object

However, we do not need the full liberty of restricting derivations where an arbitrary arrow can be tacked onto the left-hand-side square. The host object is always still derived in a systematic manner from the redex, and as main contribution of this paper we show how to give a universal characterisation of this systematic manner. It turns out to be the *dual* of the heterogeneous pushout, in that only the non-redex-arrows need to be turned around. And just as heterogeneous pushouts correspond to opfibrations, this **host construction** corresponds to *fibrations*.

For the right-hand arrow $G \longrightarrow R$ of a rewriting rule, our categorical rewriting model therefore uses opfibrations, but for the dual left-hand arrow $L \longleftarrow G$ we correspondingly use the dual *fibrations*. To our knowledge this has not been done before in the algebraic approach to rewriting. The advantages are that on one hand the host graph, which corresponds to the pushout complement, now is universally characterised, and that on the other hand some of

[3] Having $U_2[x, y]$ as the gluing term would make the right-hand-side placeholder substitutions not full; the corresponding choice in term graphs would have even more obstructing consequences.

the enhanced flexibility of restricting derivations can be carried over into an approach without any superfluous arbitrariness.

In our opinion this approach makes clear certain distinctions that have been blurred in other algebraic approaches to rewriting. As a result, by using our fibred approach the algebraic approach is now able to proceed into new territories that have been unattainable so far — we have already shown term rewriting, and the motivating application for the development of the fibred approach is rewriting of term graphs with bound variables [Kahl, 1996], comprising λ-graphs and for the first time giving an algebraic characterisation of graph reduction [Wadsworth, 1971].

3. Category Theoretic Foundations

Definition 3.1. A **category** $\mathcal{C}$ consists of objects introduced by "$A : \mathcal{C}$" and arrows (also called morphisms) introduced by "$f : A \xrightarrow{\mathcal{C}} B$", where A and B are objects of $\mathcal{C}$ called the source (or domain) respectively the target (or codomain) of f. The collection of all arrows between two objects A and B of $\mathcal{C}$ is called the **hom set** between A and B.

For every object $A : \mathcal{C}$ there is an identity arrow $\mathrm{id}_{A:\mathcal{C}} : A \xrightarrow{\mathcal{C}} A$, and for every two arrows $f : A \xrightarrow{\mathcal{C}} B$ and $g : B \xrightarrow{\mathcal{C}} C$ there is the composition "$f \mathbin{;} g$" which is an arrow of $\mathcal{C}$ from A to C— note the order of the two arrows! That way the notation for composition more closely resembles the appearance in the diagrams: $A \xrightarrow{f} B \xrightarrow{g} C$. The laws which are required to hold are:

i) $f \mathbin{;} \mathrm{id}_{B:\mathcal{C}} = f$ and $\mathrm{id}_{A:\mathcal{C}} \mathbin{;} f = f$;

ii) $f \mathbin{;} (g \mathbin{;} h) = (f \mathbin{;} g) \mathbin{;} h$.

A **subcategory** $\mathcal{D}$ of $\mathcal{C}$ has as its objects and arrows only objects resp.arrows of $\mathcal{C}$, and source, target and composition of arrows and identity arrows are the same as in $\mathcal{C}$.

A subcategory $\mathcal{D}$ of $\mathcal{C}$ is **wide** if it has the same objects as $\mathcal{C}$. □

Now we give a definition of the central notion of the original algebraic approach:

Definition 3.2. Let two arrows $d : L \xrightarrow{\mathcal{C}} R$ and $g : L \xrightarrow{\mathcal{C}} G$ between three objects of a category $\mathcal{C}$ be given. A *pushout* for d and g is an object H together with two arrows $e : G \xrightarrow{\mathcal{C}} H$ and $h : R \xrightarrow{\mathcal{C}} H$, such that $d \mathbin{;} h = g \mathbin{;} e$, and for every object H' with arrows $e' : G \xrightarrow{\mathcal{C}} H'$ and $h' : R \xrightarrow{\mathcal{C}} H'$ fulfilling $d \mathbin{;} h' = g \mathbin{;} e'$ there is a unique arrow $\varphi : H \xrightarrow{\mathcal{C}} H'$ with $h \mathbin{;} \varphi = h'$ and $e \mathbin{;} \varphi = e'$. □

$$\begin{tikzcd}
L \arrow[r, "d"] \arrow[d, "g"'] & R \arrow[d, "h"] \arrow[ddr, "h'"] & \\
G \arrow[r, "e"] \arrow[drr, "e'"'] & H \arrow[dr, dotted, "\varphi"] & \\
 & & H'
\end{tikzcd}$$

For moving into a heterogeneous settings we need functors, i.e., structure-preserving mappings between different categories:

Definition 3.3. A **functor** F from category $\mathcal{C}$ to category $\mathcal{D}$ consists of a mapping from objects of $\mathcal{C}$ to objects of $\mathcal{D}$ and a mapping from arrows of $\mathcal{C}$ to arrows of $\mathcal{D}$ such that

i) if $f : A \xrightarrow{\mathcal{C}} B$, then $F(f) : F(A) \xrightarrow{\mathcal{D}} F(B)$,

ii) for every object $A : \mathcal{C}$ we have $F(\mathrm{id}_{A:\mathcal{C}}) = \mathrm{id}_{F(A):\mathcal{D}}$,

iii) if $f : A \xrightarrow{\mathcal{C}} B$ and $g : B \xrightarrow{\mathcal{C}} C$, then $F(f \mathbin{;} g) = F(f) \mathbin{;} F(g)$.

A functor is **full** if the arrow mapping is surjective for every hom set.

A functor is **faithful** if the arrow mapping is injective for every hom set. □

The definitions for fibrations are essentially taken from [Barr and Wells, 1990].

Definition 3.4. Let $\mathcal{F} : \mathcal{C} \to \mathcal{B}$ be a functor between small categories, let $f : C \to D$ an arrow in $\mathcal{B}$, and let Y be an object of $\mathcal{C}$ with $\mathcal{F}(Y) = D$. An arrow $u : X \to Y$ in $\mathcal{C}$ is **cartesian** for f and Y, iff $\mathcal{F}(u) = f$ and for every arrow $v : Z \to Y$ in $\mathcal{C}$ and every arrow $h : \mathcal{F}(Z) \to C$ in $\mathcal{B}$ for which $h \mathbin{;} f = \mathcal{F}(v)$ there is a uniquely determined arrow $w : Z \to X$ in $\mathcal{C}$, such that $w \mathbin{;} u = v$ and $\mathcal{F}(w) = h$.

Dually for an arrow $f : C \to D$ in $\mathcal{B}$, and an object X of $\mathcal{C}$ with $\mathcal{F}(X) = C$ an arrow $u : X \to Y$ in $\mathcal{C}$ is **opcartesian** for f and X iff $\mathcal{F}(u) = f$ and for every arrow $v : X \to Z$ in $\mathcal{C}$ and every arrow $k : D \to \mathcal{F}(Z)$ in $\mathcal{B}$, for which $f \fatsemi k = \mathcal{F}(v)$, there is a uniquely determined arrow $w : Y \to Z$ in $\mathcal{C}$, such that $u \fatsemi w = v$ and $\mathcal{F}(w) = k$. □

Definition 3.5. A functor $\mathcal{F} : \mathcal{C} \to \mathcal{B}$ is a **fibration** iff for every arrow $f : C \to D$ in $\mathcal{B}$ and every object Y in $\mathcal{C}$ with $\mathcal{F}(Y) = D$ there is a cartesian arrow. $\mathcal{F} : \mathcal{C} \to \mathcal{B}$ is an **opfibration**, iff for every arrow $f : C \to D$ in $\mathcal{B}$ and every object X in $\mathcal{C}$ with $\mathcal{F}(X) = C$ there is an opcartesian arrow. □

Definition 3.6. A **cleavage** for a fibration $\mathcal{F} : \mathcal{C} \to \mathcal{B}$ is a function Σ that maps an arrow $f : C \xrightarrow{\mathcal{B}} D$ in $\mathcal{B}$ and an object Y in $\mathcal{C}$ with $\mathcal{F}(Y) = D$ to an arrow $\Sigma(f, Y)$ in $\mathcal{C}$ that is cartesian for f and Y.

Similarly an **opcleavage** Σ^{op} maps an arrow $f : C \xrightarrow{\mathcal{B}} D$ in $\mathcal{B}$ and an object X in $\mathcal{C}$ with $\mathcal{F}(X) = C$ to an arrow $\Sigma^{\mathrm{op}}(f, X)$ in $\mathcal{C}$ that is opcartesian for f and X. □

Definition 3.7. A cleavage Σ is a **splitting**, iff

i) for every two objects $D : \mathcal{B}$ and $Y : \mathcal{C}$ with $\mathcal{F}(Y) = D$ we have $\Sigma(\mathrm{id}_{D:\mathcal{B}}, Y) = \mathrm{id}_{Y:\mathcal{C}}$;

ii) for all arrows $f : C \xrightarrow{\mathcal{B}} D$ and $g : D \xrightarrow{\mathcal{B}} E$ and objects $Y : \mathcal{C}$ and $Z : \mathcal{C}$ with $\mathcal{F}(Z) = E$ and Y as domain of $\Sigma(g, Z)$ we have

$$\Sigma(f, Y) \fatsemi \Sigma(g, Z) = \Sigma(f \fatsemi g, Z)$$

A fibration is called **split**, iff it has a splitting.

In the same way an opcleavage is called splitting of an opfibration if the dual conditions hold; that opfibration is then also called split. □

4. Patterns and Matchings

We shall assume two categories $\mathcal{E}_{\mathrm{V}}$ and $\mathcal{E}_{\mathrm{H}}$ with the same objects and identities to be given. Their union is assumed to be extended to a category $\mathcal{E}$ such that $\mathcal{E}_{\mathrm{V}}$ and $\mathcal{E}_{\mathrm{H}}$ are wide subcategories of $\mathcal{E}$. The non-identical arrows in $\mathcal{E}_{\mathrm{H}}$ are called *horizontal*, those in $\mathcal{E}_{\mathrm{V}}$ *vertical*.

Let furthermore a subclass $\mathcal{H}$ of the arrows of $\mathcal{E}_{\mathrm{V}}$ be given, and let $\mathcal{T}$ be a subcategory of $\mathcal{E}_{\mathrm{H}}$. We call $\mathcal{T}$ the category of *patterns*. Without influencing the further development we could require that every arrow of $\mathcal{H}$ started from an object of $\mathcal{T}$.

Definition 4.1. A **matching** from a *pattern* object $L : \mathcal{T}$ to an (arbitrary) object $G : \mathcal{E}$ is an arrow $g : L \xrightarrow{\mathcal{H}} G$ (although $\mathcal{H}$ need not be a category, we shall employ this notation for arrows of $\mathcal{H}$ as well) in $\mathcal{H}$ from L to G.

We can easily define a **category $\mathcal{M}$ of all matchings** in which the objects are matchings, and for emphasising this object character of matchings we also use the notation $g = \underline{g : L \to G}$.

Arrows between two matchings $\underline{g : L \to G}$ and $\underline{h : R \to H}$ are defined to be pairs $\langle d, e\rangle$ of arrows of $\mathcal{E}$, where $d : L \xrightarrow{\mathcal{T}} R$, $e : G \xrightarrow{\mathcal{E}} H$ and $d \fatsemi h = g \fatsemi e$ in $\mathcal{E}$. For an object $\underline{g : L \to G} : \mathcal{M}$, its identity is the arrow $\langle \mathrm{id}_{L:\mathcal{T}}, \mathrm{id}_{G:\mathcal{E}}\rangle$, and composition is defined component-wise:

$$\langle d_1, e_1\rangle \fatsemi \langle d_2, e_2\rangle := \langle d_1 \fatsemi d_2, e_1 \fatsemi e_2\rangle \ . \qquad \square$$

It is easy to check that $\mathcal{M}$ is a well-defined category; it is a subcategory of the arrow category of $\mathcal{E}$.

If for a matching arrow $\langle d, e\rangle : g \xrightarrow{\mathcal{M}} h$ the component arrow e is in $\mathcal{E}_{\mathrm{H}}$, then we also write $\langle d, e\rangle : g \xrightarrow{\mathcal{M}_{\mathrm{H}}} h$; for e in $\mathcal{E}_{\mathrm{V}}$ we write $\langle d, e\rangle : h \xrightarrow{\mathcal{M}_{\mathrm{V}}} h$. Obviously we produce two subcategories $\mathcal{M}_{\mathrm{H}}$ and $\mathcal{M}_{\mathrm{V}}$ from $\mathcal{M}$ by correspondingly restricting the arrows.

As a restriction of the domain functor of the arrow category of $\mathcal{E}$ we obtain:

Definition 4.2. The **pattern functor** $\mathcal{P}$ from the matching category $\mathcal{M}$ to the pattern category $\mathcal{T}$ is defined by $\mathcal{P}(\underline{g : L \to G}) := L$ and $\mathcal{P}(\langle d, e\rangle) := d$. □

The check of the functorial properties of $\mathcal{P}$ is trivial.

In the following let $\mathcal{M}_>$ and $\mathcal{M}_<$, the "**outgoing**" and the "**incoming matching categories**", be sub-categories of $\mathcal{M}_\mathrm{H}$, and let $\mathcal{T}_>$ be the image of $\mathcal{M}_>$ under the pattern functor $\mathcal{P}$ and $\mathcal{T}_<$ the image of $\mathcal{M}_<$. Sometimes it can make sense to start with a sub-category $\mathcal{M}_\equiv$ of $\mathcal{M}_\mathrm{H}$ as the "*horizontal matching category*", and only then restrict further to sub-categories $\mathcal{M}_>$ and $\mathcal{M}_<$ of $\mathcal{M}_\equiv$.

These restrictions serve to satisfy special requirements of the underlying categories even where this is impossible just by restricting $\mathcal{E}_\mathrm{H}$. This successive restriction of $\mathcal{M}_\mathrm{H}$ to $\mathcal{M}_\equiv$ to $\mathcal{M}_>$ is not just performed for the sake of generality, but in the motivating application to term graphs with bound variables [Kahl, 1996], each of the restricting steps has its justification.

Gathering all the framework that we have established so far, but not listing all parts that are derivable from others like $\mathcal{M}$, we obtain a nine-tuple $(\mathcal{E}_\mathrm{V}, \mathcal{E}_\mathrm{H}, \mathcal{E}, \mathcal{H}, \mathcal{T}, \mathcal{T}_>, \mathcal{T}_<, \mathcal{M}_>, \mathcal{M}_<)$ which we shall call a **rewriting context** — the listing of $\mathcal{T}_>$ and $\mathcal{T}_<$ is of course redundant, but for their important rôle we also include these two. As an abstraction mirroring the desired heterogeneity in a stronger way we define:

Definition 4.3. A rewriting context $(\mathcal{E}_\mathrm{V}, \mathcal{E}_\mathrm{H}, \mathcal{E}, \mathcal{H}, \mathcal{T}, \mathcal{T}_>, \mathcal{T}_<, \mathcal{M}_>, \mathcal{M}_<)$ is called *consistent*, if in the image of $\mathcal{M}_\mathrm{V}$ under $\mathcal{P}$ there are only identities and if whenever for two arrows $h_1 : P \xrightarrow{\mathcal{H}} G_1$ and $h_2 : P \xrightarrow{\mathcal{H}} G_2$ starting from the same pattern object $P : \mathcal{T}$ there is an arrow $e : G_1 \xrightarrow{\mathcal{E}} G_2$ with $h_1 ; e = h_2$, then e is in $\mathcal{E}_\mathrm{V}$. □

Another formulation of this restriction is that $\mathcal{E}_\mathrm{V}$ has to contain exactly those arrows e for which $\langle \mathrm{id}_{L:\mathcal{T}}, e \rangle$ is a matching arrow for some object L.

Although this condition looks pretty strong, in fact it is not difficult to construct a consistent rewriting context. For our term rewriting example from the introduction let us select an appropriate class of objects that can serve as sources of pattern arrows:

TRS Definition 4.4. A **virtual term pattern** is an extended term consisting only of one n-ary placeholder variable and n distinct term variables. □

The pattern arrows themselves need a further restriction:

TRS Definition 4.5. A placeholder substitution is called **closed** iff it has no free term variables in its image.

For an extended term t, a placeholder substitution φ is **closed wrt.** t iff there is a closed placeholder substitution $\varphi_0 \in [\varphi]_t$. □

For closed placeholder substitution a few useful properties hold:

TRS Lemma 4.6. If the placeholder substitution φ is closed wrt. t, then

i) $[\varphi]_t = [\sigma\ \varphi]_t$ for all term substitutions σ,

ii) $\sigma\ \varphi$ is also closed wrt. $\sigma\ t$ for all term substitutions σ.

Proof:

i) $(\sigma\ \varphi)\ t = \varphi\ t$ since for a closed representant φ_0 of $[\varphi]_t$ we have $\sigma\ \varphi_0 = \varphi_0$.

ii) If φ_0 is a closed representant of $[\varphi]_t$, then

$$\begin{array}{rcll} \varphi_0\ (\sigma\ t) & = & (\sigma\ \varphi_0)\ (\sigma\ t) & \sigma\ \varphi_0 = \varphi_0 \\ & = & \sigma\ (\varphi_0\ t) & \text{2.4.iv)} \\ & = & \sigma\ (\varphi\ t) & \varphi_0 \in [\varphi]_t \\ & = & (\sigma\ \varphi)\ (\sigma\ t) & \text{2.4.iv)} \end{array}$$

and φ_0 is also a closed representant of $[\sigma\ \varphi]_{(\sigma\ t)}$ □.

Now consistency is easily achieved:

TRS Theorem 4.7. A **consistent rewriting context for term rewriting** is given by

– $\mathcal{E}$ being the category of extended terms with extended term morphisms,

– $\mathcal{E}_\mathrm{V}$ being the category of the images of term homomorphisms in $\mathcal{E}$, i.e., extended term morphisms of the shape $(t, C[\ ], [\sigma]_t, [\{\}]_{(\sigma\ t)})$,

- $\mathcal{E}_{\mathrm{H}}$ being the category of the images of placeholder arrows in $\mathcal{E}$, i.e., extended term morphisms of the shape $(t, [\,], [\{\}]_t, [\varphi]_t)$,
- $\mathcal{H} = \mathcal{E}_{\mathrm{V}}$;
- $\mathcal{T}$ has as non-identical arrows only images of placeholder arrows in $\mathcal{E}$ that start from virtual term patterns;
- $\mathcal{T}_>$ contains only those arrows $(t, [\,], [\{\}]_t, [\varphi]_t)$ of $\mathcal{T}$, for which φ is closed wrt. t, and $\mathcal{M}_>$ is the restriction of $\mathcal{M}_{\mathrm{H}}$ to matching arrows $\langle d, e\rangle$ where d is in $\mathcal{T}_>$, and
- $\mathcal{T}_< = \mathcal{T}$ and $\mathcal{M}_< = \mathcal{M}_{\mathrm{H}}$.

Proof: Since the dummy pattern definition precludes non-identity isomorphisms on virtual term patterns, there are — except identities — no placeholder arrows that are at the same time term homomorphisms, and the image of $\mathcal{M}_{\mathrm{V}}$ under $\mathcal{P}$ can therefore only contain identities.

For the other condition consider $h_1 = (t, C[\,], [\sigma]_{t_1}, [\{\}]_{(\sigma\ t)})$, $h_2 = (t, C'[\,], [\sigma']_{t_1}, [\{\}]_{(\sigma'\ t)})$ and $e = (t_1, D[\,], [\tau]_{t_1}, [\varphi]_{(\tau\ t_1)})$. Then

$$h_1 \mathbin{;} e = (t, D[(\tau\ C)[\,]], [\sigma \mathbin{;} \tau]_t, [\varphi]_{((\sigma ; \tau)\ t)})$$

and $h_1 \mathbin{;} e = h_2$ implies that $\varphi\ ((\sigma \mathbin{;} \tau)\ t) = \sigma'\ t$. With Lemma 2.5.v) we obtain $\varphi\ ((\sigma \mathbin{;} \tau)\ t) = (\sigma \mathbin{;} \tau)\ t$, and using this we can calculate:

$$\begin{array}{rcll}
\varphi\ (\tau\ t_1) & = & \varphi\ (\tau\ (C[\sigma\ t])) & \text{2.3: } t_1 = C[\sigma\ t] \\
& = & \varphi\ ((\tau\ C)[\tau\ (\sigma\ t)]) & \text{2.4.ii)} \\
& = & (\tau\ C)[\varphi\ (\tau\ (\sigma\ t))] & \text{2.5.ii)} \\
& = & (\tau\ C)[\tau\ (\sigma\ t)] & \varphi\ ((\sigma ; \tau)\ t) = (\sigma ; \tau)\ t \\
& = & \tau\ (C[\sigma\ t]) & \text{2.4.ii)} \\
& = & \tau\ t_1\ , & \text{2.3: } t_1 = C[\sigma\ t]
\end{array}$$

concluding $\varphi \in [\{\}]_{(\tau\ t_1)}$ and $e = (t_1, D[\,], [\tau]_{t_1}, [\{\}]_{(\tau\ t_1)})$, so e is in $\mathcal{E}_{\mathrm{V}}$. □

The restriction on $\mathcal{T}_>$ amounts to the usual restriction that right-hand sides of term rewriting rules must not introduce new variables.

5. Heterogeneous Pushouts

Before investigating further properties of the pattern functor, we introduce the notion of heterogeneous pushouts as a simple generalisation of pushouts. In this section the important subcategory of $\mathcal{M}_{\mathrm{H}}$ is $\mathcal{M}_>$; on first reading it is safe to assume $\mathcal{M}_> = \mathcal{M}_{\mathrm{H}}$ and $\mathcal{T}_> = \mathcal{T}$.

$$\begin{tikzcd}
L \arrow[r, "d"] \arrow[d, "g"'] & R \arrow[d, "h"] \arrow[ddr, "h'"] & \\
G \arrow[r, "e"] \arrow[drr, "e'"'] & H \arrow[dr, dotted, "\varphi"] & \\
& & H'
\end{tikzcd}$$

Definition 5.1. A **heterogeneous pushout** for a matching $\underline{g : L \to G}$ together with a pattern arrow $d : L \xrightarrow{\mathcal{T}_>} R$ is a matching $\underline{h : R \to H}$ together with an arrow $e : G \xrightarrow{\mathcal{E}_{\mathrm{H}}} H$, such that $\langle d, e\rangle : g \xrightarrow{\mathcal{M}_>} h$ is a matching arrow in $\mathcal{M}_>$ (this implies commutativity of the diagram, i.e., $d \mathbin{;} h = g \mathbin{;} e$), and for every matching arrow $\langle d, e'\rangle : g \xrightarrow{\mathcal{M}_>} \underline{h' : R \to H'}$ in $\mathcal{M}_>$ there is a unique arrow $\varphi : H \xrightarrow{\mathcal{E}_{\mathrm{V}}} H'$ with $h \mathbin{;} \varphi = h'$ and $e \mathbin{;} \varphi = e'$. □

(In the diagrams, we draw dotted arrows for $\mathcal{E}_{\mathrm{V}}$, but bold arrows for $\mathcal{H}$.) What Banach calls "local universality" in [1994, Theorem 6.6] is therefore an instance of heterogeneous pushouts. Note however, how in the above definition, matchings are not only used for ease of notation, but also bring with them a selection of possible heterogeneous pushouts according to the choice of the matching category $\mathcal{M}_>$.

TRS Theorem 5.2. The construction of Prop. 2.7 indeed yields heterogeneous pushouts in the term rewriting context of Theorem 4.7.

Proof: We have seen commutativity at Prop. 2.7. Let $d := (P, [\,], [\{\}]_P, [\varphi]_P)$ and $g := (P, C[\,], [\sigma]_P, [\{\}]_{(\sigma\ P)})$.

For any $e' := (G, [\,], [\{\}]_G, [\psi]_G)$ and $h' := (Q, D[\,], [\tau]_Q, [\{\}]_{(\tau\ Q)})$

$$\begin{tikzcd}
P \arrow[r, "{d=(P,[\,],[\{\}]_P,[\varphi]_P)}"] \arrow[d, "{g=(P,C[\,],[\sigma]_P,[\{\}]_{(\sigma\ P)})}"'] & Q \arrow[d, "{h'=(Q,D[\,],[\tau]_Q,[\{\}]_{(\tau\ Q)})}"] \\
G \arrow[r, "{e'=(G,[\,],[\{\}]_G,[\psi]_G)}"] & H
\end{tikzcd}$$

the equation $d \mathbin{⨾} h' = g \mathbin{⨾} e'$ implies

$$(P, D[\,], [\tau]_P, [\tau\ \varphi]_{(\tau\ P)}) = d \mathbin{⨾} h' = g \mathbin{⨾} e' = (P, C[\,], [\sigma]_P, [\psi]_{(\sigma\ P)}) \ ,$$

so we have $D[\,] = C[\,]$, $[\tau]_P = [\sigma]_P$ which implies $\tau\ P = \sigma\ P$, and, since d is in $\mathcal{T}_>$ so that Q has the same free variables as P, also $\tau\ Q = \sigma\ Q$. Therewith we obtain $h' = h$ and $\tau\ \varphi = \sigma\ \varphi$. The above equality furthermore implies

$$[\tau\ \varphi]_{(\tau\ P)} = [\sigma\ \varphi]_{(\sigma\ P)} = [\psi]_{(\sigma\ P)} \ ;$$

with Lemma 2.5.ii) this implies $[\psi]_G = [\tau\ \varphi]_G$ and $e' = e$. So, together with the restriction to $\mathcal{T}_>$, the commutativity condition alone suffices to uniquely determine a heterogeneous pushout for term rewriting. □

Without the restriction to $\mathcal{T}_>$ we could have obtained situations like the following:

$$\begin{tikzcd}
U_1[x] \arrow[r, "{\{U_1[\delta_1]\mapsto q(\delta_1,z)\}}"] \arrow[d, "{\{x\mapsto f(z)\}}"'] & q(x,z) \arrow[d, "{\{x\mapsto f(z)\}}"'] \arrow[dr, "{\{x\mapsto f(z),z\mapsto a\}}"] & \\
U_1[f(z)] \arrow[r, "{\{U_1[\delta_1]\mapsto q(\delta_1,z)\}}"] \arrow[drr, "{\{U_1[\delta_1]\mapsto q(\delta_1,a)\}}"'] & q(f(z),z) \arrow[dr, dotted, "?"] & \\
 & & q(f(z),a)
\end{tikzcd}$$

Here no substitution can turn $q(f(z), z)$ into $q(f(z), a)$.

We may observe that heterogeneous pushouts are unique up to isomorphism the same way as pushouts: Let $(\underline{h_1 : R \to H_1}, e_1)$ and $(\underline{h_2 : R \to H_2}, e_2)$ be two heterogeneous pushouts for (g, d) and let φ_1 and φ_2 be the unique arrows from H_1 to H_2 and vice versa. Then we obtain $h_1 \mathbin{⨾} \varphi_1 \mathbin{⨾} \varphi_2 = h_2 \mathbin{⨾} \varphi_2 = h_1$ and $e_1 \mathbin{⨾} \varphi_1 \mathbin{⨾} \varphi_2 = e_2 \mathbin{⨾} \varphi_2 = e_1$. But the only uniquely determined arrow ι from H_1 to H_1 with $h_1 \mathbin{⨾} \iota = h_1$ and $e_1 \mathbin{⨾} \iota = e_1$ can be $\mathrm{id}_{H_1:\mathcal{E}_\mathrm{V}}$, so that $\varphi_1 \mathbin{⨾} \varphi_2 = \mathrm{id}_{H_1:\mathcal{E}_\mathrm{V}}$. Analogously we obtain $\varphi_2 \mathbin{⨾} \varphi_1 = \mathrm{id}_{H_2:\mathcal{E}_\mathrm{V}}$; therefore H_1 and H_2 are isomorphic.

Unlike ordinary pushouts, however, heterogeneous pushouts are not *per se* composable; this has to be checked carefully in the concrete categories underlying the chosen rewriting context, and we define:

Definition 5.3. A rewriting context **has composable heterogeneous pushouts** if it has all heterogeneous pushouts and for all matchings $\underline{g : L \to G}$ and for all pattern arrows $d_1 : L \xrightarrow{\mathcal{T}} M$ and $d_2 : M \xrightarrow{\mathcal{T}} N$, if (h, e_1) is a heterogeneous pushout for g and d_1 and (k, e_2) is a heterogeneous pushout for h and d_2, then $(k, e_1 \mathbin{⨾} e_2)$ is also a heterogeneous pushout for g and $d_1 \mathbin{⨾} d_2$.

$$\begin{tikzcd}
L \arrow[r, "d_1"] \arrow[d, "g"'] & M \arrow[r, "d_2"] \arrow[d, "h"'] & N \arrow[d, "k"'] \\
G \arrow[r, "e_1"] & H \arrow[r, "e_2"] & K
\end{tikzcd}$$

□

A claim that can be read in many category theoretic texts is that opfibrations are a generalisation of pushouts. This is mainly supported by the following

Folklore Theorem: A category B has pushouts if and only if the domain functor from the arrow category over B to B is an opfibration. □

(The dual version may be more familiar to most category theorists.) We employ our newly introduced heterogeneous pushouts for substantiating this claim in a very direct way and show how opfibrations can be considered as a generalisation of pushouts.

Lemma 5.4. In the context of the pattern functor $\mathcal{P}$ from $\mathcal{M}_>$ to $\mathcal{T}_>$ (we do not distinguish this restriction notationally) an opcartesian arrow $\langle d_1, e_1\rangle : \underline{g : L \to G} \xrightarrow{\mathcal{M}} \underline{h : M \to H}$ in $\mathcal{M}_>$ for the arrow $d_1 : L \xrightarrow{\mathcal{T}} M$ and the object $g : \mathcal{M}_>$ gives rise to a heterogeneous pushout (h, e_1) for (g, d_1), at least within a consistent context.

$$\begin{tikzcd}
L \arrow[r, "d_1"] \arrow[d, "g"] \arrow[rr, bend left, "d"] & M \arrow[r, "d_2"] \arrow[d, "h"] & N \arrow[dd, "k"] \\
G \arrow[r, "e_1"] \arrow[drr, "e"'] & H \arrow[dr, "e_2"] & \\
 & & K
\end{tikzcd}$$

Proof: The definition of opcartesian arrows demands, that for any arrow $\langle d, e\rangle : g \xrightarrow{\mathcal{M}} k$ and for any arrow $d_2 : M \xrightarrow{\mathcal{T}} N$, for which $d_1 \fatsemi d_2 = d$, there is a unique arrow $\langle d_2, e_2\rangle : h \xrightarrow{\mathcal{M}} k$ for which $\langle d_1, e_1\rangle \fatsemi \langle d_2, e_2\rangle = \langle d, e\rangle$. By letting $d_2 := \mathrm{id}_{M:\mathcal{T}}$, we obtain exactly the definition of a heterogeneous pushout for g and d_1 — since $M = N$, in a consistent context the arrow e_2 has to be in $\mathcal{E}_{\mathrm{V}}$. □

In [Kahl, 1996] the same result was arrived at for what is there called "separated contexts" which obey strictly stronger limitations than consistent contexts. In a separated context, composition of horizontal arrows is limited to cases where at least one factor is an identity, so a separated context trivially has composable heterogeneous pushouts as its main advantage.

The opposite direction to Lemma 5.4 is not so trivial, however, especially since it is not immediately clear how $\mathcal{M}_>$ should be extended to a sub-category of $\mathcal{M}$ that encompasses as many vertical arrows as possible without also being expanded to contain undesired non-vertical arrows that could be candidates for being opcartesian instead of those selected by the choice of $\mathcal{M}_>$. (It is of course not necessary to avoid the inclusion of non-vertical arrows that do not compete here.)

If there is a heterogeneous pushout for h and d_2, then the uniqueness of e_2, if existing, is guaranteed. But for the existence of an arrow $e_2 : H \xrightarrow{\mathcal{E}} K$ we need composability of heterogeneous pushouts in the actual rewriting context, which, when translated to a choice of heterogeneous pushouts, corresponds to the "global universality" of [Banach, 1994]:

Definition 5.5. A function HPO that takes a matching $\underline{g : L \to G}$ together with a pattern arrow $d : L \xrightarrow{\mathcal{T}} R$ to a heterogeneous pushout $(\rho, e) = HPO(g, d)$ for g and d is called a **choice of heterogeneous pushouts** if for all matchings $\underline{g : L \to G}$ and for all pattern arrows d_1, d_2 the following two requirements are satisfied:

$$\begin{tikzcd}
L \arrow[r, "d_1"] \arrow[d, "g"'] & M \arrow[r, "d_2"] \arrow[d, "h"'] & N \arrow[d, "k"'] \\
G \arrow[r, "e_1"] & H \arrow[r, "e_2"] & K
\end{tikzcd}$$

i) $HPO(g, \mathrm{id}_{L:\mathcal{T}}) = (g, \mathrm{id}_{G:\mathcal{E}_{\mathrm{H}}})$

ii) For $(h, e_1) := HPO(g, d_1)$ and $(k, e_2) := HPO(h, d_2)$ the equation $(k, e_1 \fatsemi e_2) = HPO(g, d_1 \fatsemi d_2)$ always holds. □

In case the first condition is not fulfilled for a given choice of heterogeneous pushouts, it can always be enforced by moving to the category of abstract graphs that is equivalent to $\mathcal{E}$ — see [Corradini *et al.*, 1993] for details. In a consistent context, condition i) always holds: Let $(\underline{h : L \to H}, e) := HPO(g, \mathrm{id}_{L:\mathcal{T}})$, then $e : G \to H$ according to the Def. 5.1 of heterogeneous pushouts is an arrow in $\mathcal{E}_{\mathrm{H}}$. But since g and h have the common domain L, according to the definition of a consistent context e must be in $\mathcal{E}_{\mathrm{V}}$, too, so e is the identity $\mathrm{id}_{G:\mathcal{E}_{\mathrm{H}}}$ and $G = H$.

The second condition is not so superficial: moving to abstract graphs is only sufficient to enforce it if the rewriting context has composable heterogeneous pushouts.

In our running example we have carefully excluded non-identical isomorphisms, and heterogeneous pushouts are indeed composable:

TRS Theorem 5.6. The choice of heterogeneous pushouts that has been effectively defined in Prop. 2.7 is well-defined.

Proof: First let the matching $\underline{g : L \to G}$ be given by $g = (L, C[\,], [\sigma]_L, [\{\}]_{(\sigma\ L)})$.

i) Keeping in mind that that for every extended term A the associated identity is given by $\mathrm{id}_{A:\mathcal{E}} = (A, [\,], [\{\}]_A, [\{\}]_A)$, we obtain

$$HPO(g, \mathrm{id}_{L:\mathcal{T}}) = \Big((L, C[\,], [\sigma]_L, [\{\}]_{(\sigma\ L)})\ ,\ (G, [\,], [\{\}]_G, [\{\}]_G)\Big) = (g, \mathrm{id}_{G:\mathcal{E}_\mathrm{H}})\ .$$

ii) Now let the pattern arrows $d_1 : L \xrightarrow{\mathcal{T}} M$ and $d_s : M \xrightarrow{\mathcal{T}} N$ be given by $d_1 = (L, [\{\}]_L, [\varphi]_L)$ and $d_2 = (M, [\,], [\{\}]_M, [\psi]_M)$.

Then let

$$(h, e_1) := HPO(g, d_1) = \Big((M, C[\,], [\sigma]_M, [\{\}]_{(\sigma\ M)})\ ,\ (G, [\,], [\{\}]_G, [\sigma\ \varphi]_G)\Big)\ ,$$

and let

$$(k, e_2) := HPO(h, d_2) = \Big((N, C[\,], [\sigma]_N, [\{\}]_{(\sigma\ N)}), (H, [\,], [\{\}]_H, [\sigma\ \psi]_H)\Big)\ .$$

With Lemma 2.4.iv) we obtain $\sigma\ \varphi \mathbin{;} \sigma\ \psi = \sigma\ (\varphi \mathbin{;} \psi)$. Then

$$\begin{aligned}(k, e_1 \mathbin{;} e_2) &= \Big((N, C[\,], [\sigma]_N, [\{\}]_{(\sigma\ N)})\ ,\ (G, [\,], [\{\}]_G, [(\sigma\ \varphi) \mathbin{;} (\sigma\ \psi)]_G)\Big)\\ &= \Big((N, C[\,], [\sigma]_N, [\{\}]_{(\sigma\ N)})\ ,\ (G, [\,], [\{\}]_G, [\sigma\ (\varphi \mathbin{;} \psi)]_G)\Big)\\ &= HPO(g, (L, [\,], [\{\}]_L, [\varphi \mathbin{;} \psi]_L))\\ &= HPO(g, d_1 \mathbin{;} d_2)\ .\end{aligned}$$

□

Banach [1993a; 1994] gives what we call a choice of heterogeneous pushouts with "redirection couples" for $\mathcal{E}_\mathrm{H}$ and "rigid" "homomorphisms" for $\mathcal{H}$, whereas $\mathcal{E}_\mathrm{V}$ are characterised as redirection couples in one place and as "rigid" homomorphisms in another place, where a second notion of rigidity is used. There is no abstract composition of arrows from $\mathcal{E}_\mathrm{H}$ with arrows from $\mathcal{H}$ or $\mathcal{E}_\mathrm{V}$, so Banach's formulation of his heterogeneous pushout conditions appears to be not abstract enough. There is also no restriction of redirection couples to a pure pattern category $\mathcal{T}$. Banach nevertheless provides a proof of composability [1994] and also makes use of abstract graphs for the same purpose as we indicated above (the "weak opfibrations" presented by the same author in another paper [Banach, 1993b], which might offer a solution in case of lacking composability, are therefore not needed). This proof of composability depends on "garbage retention" which also is a feature of DACTL [Glauert *et al.*, 1990], the motivating system for Banach's approach.

In the context of rewriting term graphs with bound variables [Kahl, 1996], garbage retention is not possible and composability of single heterogeneous pushouts therefore much harder to achieve.

6. Application of the Grothendiek Construction

The first part of this section corresponds to the exposition in [Banach, 1993a] resp. [Banach, 1994] that establishes rewriting functors and applies the Grothendiek construction, arriving at a Grothendiek category where heterogeneous pushouts can be seen as resulting from the splitting of an opfibration. That part has only been adapted to our more general approach that abstracts from the underlying categories; we are also more rigorous in the typing. After that we take advantage of our more abstract approach and establish a closer connection between the Grothendiek category and the original outgoing matching category $\mathcal{M}_>$.

Definition 6.1. The **index category** $\mathcal{G}^L$ for an object L of $\mathcal{T}$ is the subcategory of $\mathcal{M}_\mathrm{V}$ with objects g, for which $\mathcal{P}(g) = L$ (so we have $g = \underline{g : L \to G}$ for appropriate g and G) and arrows f, for which $\mathcal{P}(f) = \mathrm{id}_{L:\mathcal{T}}$; so all f must be of the shape $\langle \mathrm{id}_{L:\mathcal{T}}, \varphi \rangle$ for an appropriate arrow φ of $\mathcal{E}_\mathrm{V}$. □

Banach additionally demands *rigidity* of ϕ, a condition that seems to lack an immediate category theoretical counterpart, but amounts to a restriction on $\mathcal{M}_>$.

For an arrow $d : L \xrightarrow{\mathcal{T}} R$, we then obtain a functor Rew^d from the index category $\mathcal{G}^L$ of the domain of d to the index category $\mathcal{G}^R$ of its codomain:

Definition 6.2. Let $d : L \xrightarrow{\mathcal{T}} R$ be a pattern arrow. For two objects $\underline{g_1 : L \to G_1}$ and $\underline{g_2 : L \to G_2}$ of $\mathcal{G}^L$ let

$$(h_1, e_1) := \mathit{HPO}(g_1, d) \quad \text{and} \quad (h_2, e_2) := \mathit{HPO}(g_2, d) \ .$$

$$\begin{tikzcd}
& L \arrow[rr, "d"] \arrow[dl, "g_1"'] \arrow[dd, "g_2"] & & R \arrow[dl, "h_1"'] \arrow[dd, "h_2"] \\
G_1 \arrow[rr, "e_1"] \arrow[dr, dotted, "\varphi"'] & & H_1 \arrow[dr, dotted, "\psi"'] & \\
& G_2 \arrow[rr, "e_2"] & & H_2
\end{tikzcd}$$

Then we define the **individual rewriting functor** Rew^d from $\mathcal{G}^L$ to $\mathcal{G}^R$ on objects by $\mathit{Rew}^d(g_1) := h_1$, and for each arrow $\varphi_0 = \langle \mathrm{id}_{L:\mathcal{T}}, \varphi \rangle$ of $\mathcal{G}^L$ we determine the unique arrow $\psi_0 = \langle \mathrm{id}_{R:\mathcal{T}}, \psi \rangle$ for which $\langle d, e_1 \rangle \fatsemi \psi_0 = \varphi_0 \fatsemi \langle d, e_2 \rangle$. Such an arrow exists according to the definition of heterogeneous pushouts and we define $\mathit{Rew}^d(\varphi_0) := \psi_0$. □

It is easy to verify that Rew^d is indeed a functor. Using the composability of the choice of heterogeneous pushouts we can prove the composability of different such functors:

Lemma 6.3. If HPO is a choice of heterogeneous pushouts, then $\mathsf{Rew}(L) := \mathcal{G}^L$ and $\mathsf{Rew}(d) := \mathit{Rew}^d$ define a functor Rew from $\mathcal{T}_>$ to Cat.

Proof: The only non-trivial part is the composition of arrows. Here we have to show that $\mathsf{Rew}(d_1 \fatsemi d_2) = (\mathsf{Rew}(d_1)) \fatsemi (\mathsf{Rew}(d_2))$, which immediately reduces to $\mathit{Rew}^{d_1 \fatsemi d_2} = \mathit{Rew}^{d_1} \fatsemi \mathit{Rew}^{d_2}$.

Let $\underline{g_1 : L \to G_1}$ and $\underline{g_2 : L \to G_2}$ be two objects of $\mathsf{Rew}(L) = \mathcal{G}^L$ and let $d_1 : L \xrightarrow{\mathcal{T}} M$ and $d_2 : M \xrightarrow{\mathcal{T}} N$ be two composable arrows of $\mathcal{T}_>$. Then we define $(h_i, e_i) := \mathit{HPO}(g_i, d_1)$ and $(k_i, f_i) := \mathit{HPO}(h_i, d_2)$ for $i \in \{1, 2\}$. With the composability of HPO we obtain $(k_i, e_i \fatsemi f_i) = \mathit{HPO}(g_i, d_1 \fatsemi d_2)$.

$$\begin{tikzcd}
& L \arrow[rr, "d_1"] \arrow[dl, "g_1"'] \arrow[dd, "g_2"] & & M \arrow[rr, "d_2"] \arrow[dl, "h_1"'] \arrow[dd, "h_2"] & & N \arrow[dl, "k_1"'] \arrow[dd, "k_2"] \\
G_1 \arrow[rr, "e_1"] \arrow[dr, dotted, "\varphi"'] & & H_1 \arrow[rr, "f_1"] \arrow[dr, dotted, "\chi"'] & & K_1 \arrow[dr, dotted, "\psi"'] & \\
& G_2 \arrow[rr, "e_2"] & & H_2 \arrow[rr, "f_2"] & & K_2
\end{tikzcd}$$

For the application of $\mathit{Rew}^{d_1 \fatsemi d_2}$ to objects, we have

$$\mathit{Rew}^{d_1 \fatsemi d_2}(g_i) \ = \ k_i = \mathit{Rew}^{d_2}(h_i) = \mathit{Rew}^{d_2}(\mathit{Rew}^{d_1}(g_i)) \ = \ (\mathit{Rew}^{d_1} \fatsemi \mathit{Rew}^{d_2})(g_i) \ .$$

For an arrow $\varphi_0 = \langle \mathrm{id}_{L:\mathcal{T}}, \varphi \rangle : g_1 \xrightarrow{\mathcal{G}^L} g_2$ of $\mathsf{Rew}(L)$ we obtain

$$\begin{aligned}
\mathit{Rew}^{d_1 \fatsemi d_2}(\varphi_0) \ &= \ \langle \mathrm{id}_{N:\mathcal{T}}, \psi \rangle \ = \ \mathit{Rew}^{d_2}(\langle \mathrm{id}_{M:\mathcal{T}}, \chi \rangle) \\
&= \ \mathit{Rew}^{d_2}(\mathit{Rew}^{d_1}(\varphi_0)) \ = \ (\mathit{Rew}^{d_1} \fatsemi \mathit{Rew}^{d_2})(\varphi_0) \ .
\end{aligned}$$

Therefore, Rew is well defined as a functor from $\mathcal{T}_>$ in Cat. □

The Grothendiek construction, applied to the functor Rew from $\mathcal{T}_>$ to Cat, yields a category $\mathcal{G} := \mathbf{G}(\mathcal{T}, \mathsf{Rew})$ and a projection functor $\mathcal{F}$ from $\mathcal{G}$ to $\mathcal{T}_>$. An object of $\mathcal{G}$ is a pair (g, L), where L is an object of $\mathcal{T}$ and g an object of $\mathsf{Rew}(L) = \mathcal{G}^L$; so such an object turns out to be a redundant pair $(\underline{g : L \to G}, L)$.

An arrow $\langle\varphi_0, d\rangle : (g, L) \xrightarrow{\mathcal{G}} (h, R)$ consists of an arrow $d : L \xrightarrow{\mathcal{T}_>} R$ of $\mathcal{T}_>$ and an arrow $\varphi_0 : Rew^d(g) \xrightarrow{\mathcal{G}^R} h$ of $\mathsf{Rew}(R)$ (note that by definition $Rew^d(g)$ is an object of $\mathcal{G}^R = \mathsf{Rew}(R)$), so that $\varphi_0 = \langle \mathrm{id}_{R:\mathcal{T}}, \varphi\rangle$ for some arrow φ of $\mathcal{E}_{\mathrm{V}}$. For two arrows $\langle\varphi_1, d_1\rangle : (g, L) \xrightarrow{\mathcal{G}} (h, M)$ and $\langle\varphi_2, d_2\rangle : (h, M) \xrightarrow{\mathcal{G}} (k, N)$ of $\mathcal{G}$, their composition is defined by

$$
\begin{tikzcd}
L \arrow[r, "d"] \arrow[d, "g"'] & R \arrow[d] \arrow[dr, "h"] \arrow[dl, phantom, "Rew^d(g)"] & \\
G \arrow[r, "e"] & H_0 \arrow[r, dotted, "\varphi"] & H
\end{tikzcd}
$$

$$\langle\varphi_1, d_1\rangle \mathbin{;} \langle\varphi_2, d_2\rangle := \langle Rew^{d_2}(\varphi_1) \mathbin{;} \varphi_2, d_1 \mathbin{;} d_2\rangle \ .$$

$$
\begin{tikzcd}
L \arrow[r, "d_1"] \arrow[d, "g"] & M \arrow[r, "d_2"] \arrow[d, "h_0"'] \arrow[ddr, "h"] & N \arrow[d, "k_0"'] \arrow[ddr, "k_1"] \arrow[dddrr, "k"] & & \\
G \arrow[r, "e_1"] & H_0 \arrow[r, "e_2"] \arrow[dr, dotted, "\varphi_1"'] & K_0 \arrow[dr, dotted, "\psi"] & & \\
 & & H \arrow[r, "f_2"] & K_1 \arrow[dr, dotted, "\varphi_2"'] & \\
 & & & & K
\end{tikzcd}
$$

The projection functor $\mathcal{F}$ from the Grothendiek category $\mathcal{G}$ to the pattern category $\mathcal{T}_>$ is defined by $\mathcal{F}((g, L)) := L$ and $\mathcal{F}(\langle\varphi, d\rangle) := d$. According to the properties of the Grothendiek construction this functor $\mathcal{F}$ is a split opfibration, and for every arrow $d : L \xrightarrow{\mathcal{T}_>} R$ and every object (g, L) of $\mathcal{G}$ the splitting Σ^{op} is defined by

$$\Sigma^{\mathrm{op}}(d, (g, L)) := (\mathrm{id}_{Rew^d(g):\mathcal{G}^R}, d) \ .$$

It is now interesting to establish a relation between the Grothendiek category and the matching category, where the latter is not made explicit by Banach; by defining a functor $\mathcal{Z}$ from $\mathcal{G}$ to $\mathcal{M}$ in a natural way we quit the emulation of Banach's development and pursue new paths of our own:

Definition 6.4. Let (g, L) be an object of the Grothendiek category $\mathcal{G}$ and let $\langle\varphi_0, d\rangle : (g, L) \xrightarrow{\mathcal{G}} (h, R)$ be an arrow with $\varphi_0 = \langle \mathrm{id}_{R:\mathcal{T}}, \varphi\rangle : Rew^d(g) \xrightarrow{\mathcal{G}^R} h$.

We calculate $(h', e) := HPO(g, d)$. By $\mathcal{Z}((g, L)) := g$ and $\mathcal{Z}(\langle\varphi_0, d\rangle) = \langle d, e \mathbin{;} \varphi\rangle$ we define the functor $\mathcal{Z}$ from $\mathcal{G}$ to $\mathcal{M}$. □

Because of $\mathcal{Z}(\Sigma^{\mathrm{op}}(d, (g, L))) = \langle d, e\rangle = HPO(g, d)$ this gives us a direct correspondence to the choice of heterogeneous pushouts HPO in the shape of the splitting Σ^{op} for the projection functor $\mathcal{F}$. Furthermore we can factorise the projection functor over the pattern functor using $\mathcal{Z}$, i.e., $\mathcal{F} = \mathcal{Z} \mathbin{;} \mathcal{P}$.

The functor $\mathcal{Z}$ in general is not *full* (horizontal matching arrows outside $\mathcal{M}_>$ need not be covered), but we have:

Lemma 6.5. The functor $\mathcal{Z}$ is *faithful.*

Proof: Let two arrows $\langle\varphi_0, d_1\rangle, \langle\psi_0, d_2\rangle : (g, L) \xrightarrow{\mathcal{G}} (h, R)$ with $\varphi_0 = \langle \mathrm{id}_{R:\mathcal{T}}, \varphi\rangle : Rew^{d_1}(g) \xrightarrow{\mathcal{G}^R} h$ and $\psi_0 = \langle \mathrm{id}_{R:\mathcal{T}}, \psi\rangle : Rew^{d_2}(g) \xrightarrow{\mathcal{G}^R} h$ be given such that they have equal images under $\mathcal{Z}$, i.e., $\mathcal{Z}(\langle\varphi_0, d_1\rangle) = \mathcal{Z}(\langle\psi_0, d_2\rangle)$. Let $(\underline{h_1 : R \to H_1}, e_1) := HPO(g, d_1)$ and $(\underline{h_2 : R \to H_2}, e_2) := HPO(g, d_2)$; then

$$\langle d_1, e_1 \mathbin{;} \varphi\rangle = \mathcal{Z}(\langle\varphi_0, d_1\rangle) = \mathcal{Z}(\langle\psi_0, d_2\rangle) = \langle d_2, e_2 \mathbin{;} \psi\rangle$$

implies $d_1 = d_2$, and therefore $e_1 = e_2$ and $e_1 \mathbin{;} \varphi = e_1 \mathbin{;} \psi$. But with the equality of the heterogeneous pushouts we anyway have $h_1 = h_2$ and therefore $h_1 \mathbin{;} \varphi = h = h_1 \mathbin{;} \psi$, the definition of heterogeneous pushouts gives us $\varphi = \psi$ as uniquely determined arrow from H_1 to H and therefore $\langle \varphi_0, d_1 \rangle = \langle \psi_0, d_2 \rangle$. □

A slight adaptation of the proof shows that $\mathcal{Z}$ is even injective on arrows; injectivity on objects is obvious.

This provides us with the restriction of $\mathcal{M}$ that we could not produce in the last section; together with Lemma 5.4 we obtain the important result:

Theorem 6.6 (Pattern Functor as Opfibration).
Let $\overline{\mathcal{M}_>}$ be the sub-category of $\mathcal{M}$ that is the image of $\mathcal{G}$ under $\mathcal{Z}$. In a consistent rewriting context with composable heterogeneous pushouts, any choice of heterogeneous pushouts is then equivalent to a splitting for the pattern functor $\mathcal{P}$ from $\overline{\mathcal{M}_>}$ to $\mathcal{T}_>$, turning $\mathcal{P}$ into a split opfibration. □

$\overline{\mathcal{M}_>}$ essentially prohibits unwanted non-vertical arrows from $\mathcal{M}$ by only including those vertical arrows that have a heterogeneous pushout object as their domain.

If, however, $\mathcal{M}_> = \mathcal{M}_{\mathrm{H}}$, and if every matching arrow is decomposable into a horizontal and a vertical matching arrow, then $\mathcal{G}$ and $\mathcal{M}$ are in fact isomorphic:

Theorem 6.7. In a consistent rewriting context, if $\mathcal{M}_> = \mathcal{M}_{\mathrm{H}}$ and if for every arrow $f : g \xrightarrow{\mathcal{M}} h$ there are a matching $k : \mathcal{M}$ and two arrows $e : g \xrightarrow{\mathcal{M}_{\mathrm{H}}} k$ and $\varphi : k \xrightarrow{\mathcal{M}_{\mathrm{V}}} h$ such that $f = e \mathbin{;} \varphi$, then $\mathcal{Z}$ is a category isomorphism and $\overline{\mathcal{M}_>} = \mathcal{M}$.

Proof: We define a functor $\mathcal{I}$ from $\mathcal{M}$ into $\mathbf{G}$(,i)n the following way: For objects, let $\mathcal{I}(\underline{g : L \to G}) = (\underline{g : L \to G}, L)$, and for objects we obtain $\mathcal{I}(\mathcal{Z}((\underline{g : L \to G}, L))) = \mathcal{I}(\underline{g : L \to G}) = (\underline{g : L \to G}, L)$ and the other isomorphism equation $\mathcal{Z}(\mathcal{I}(\underline{g : L \to G})) = \mathcal{Z}((\underline{g : L \to G}, L)) = \underline{g : L \to G}$.

For an arrow $f : g \xrightarrow{\mathcal{M}} \underline{h : M \to H}$ with two arrows $\langle d, e \rangle : g \xrightarrow{\mathcal{M}_{\mathrm{H}}} k$ and $\langle \mathrm{id}_{M:\mathcal{T}}, \varphi \rangle : k \xrightarrow{\mathcal{M}_{\mathrm{V}}} h$ such that $f = \langle d, e \mathbin{;} \varphi \rangle$, let $(k', e') := \mathit{HPO}(g, d)$ and let $\psi : K \xrightarrow{\mathcal{E}_{\mathrm{V}}} K'$ be the unique arrow from the definition of the heterogeneous pushout. Then $\mathcal{I}(f) := \langle \langle \mathrm{id}_{M:\mathcal{T}}, \psi \mathbin{;} \varphi \rangle, d \rangle$ and

$$\mathcal{Z}(\mathcal{I}(f)) = \mathcal{Z}(\langle \langle \mathrm{id}_{M:\mathcal{T}}, \psi \mathbin{;} \varphi \rangle, d \rangle) = \langle d, e' \mathbin{;} \psi \mathbin{;} \varphi \rangle = f \ .$$

Vice versa, for an arrow $\langle \varphi_0, d \rangle : (g, L) \xrightarrow{\mathcal{G}} (h, M)$ with $\varphi_0 = \langle \mathrm{id}_{M:\mathcal{T}}, \varphi \rangle : \mathit{Rew}^d(g) \xrightarrow{\mathcal{G}^M} h$ and $(h', e) := \mathit{HPO}(g, d)$ we obtain

$$\mathcal{I}(\mathcal{Z}(\langle \varphi_0, d \rangle)) = \mathcal{I}(\langle d, e \mathbin{;} \varphi \rangle) = \langle \varphi_0, d \rangle$$ □

In our running example, $\mathcal{M}_>$ is restricted from $\mathcal{M}_{\mathrm{H}}$ only via a restriction of $\mathcal{T}$, and therefore will still get a reasonable closure:

TRS Theorem 6.8. In the context of term rewriting, $\overline{\mathcal{M}_>} = \mathcal{M}_>$ is the reverse image of $\mathcal{T}_>$ under the pattern functor $\mathcal{P}$ considered as a functor from the full matching category $\mathcal{M}$ to $\mathcal{T}$, that is, $\overline{\mathcal{M}_>}$ contains exactly those matching arrows $\langle d, e \rangle$ where d is an arrow of $\mathcal{T}_>$.

Proof: That every arrow can be appropriately factored follows from the fact that for arbitrary extended term morphisms $(t_1, C[\ ], [\sigma]_{t_1}, [\varphi]_{(\sigma\ t_1)})$ we have

$$\begin{array}{ll} & (t_1, [\ ], [\{\}]_{t_1}, [\varphi]_{t_1}) \mathbin{;} (\varphi\ t_1, C[\ ], [\sigma]_{(\varphi\ t_1)}, [\{\}]_{(\sigma\ (\varphi\ t_1))}) \\ = & (t_1, C[\ ], [\sigma]_{t_1}, [\sigma\ \varphi]_{(\sigma\ t_1)}) \\ = & (t_1, C[\ ], [\sigma]_{t_1}, [\varphi]_{(\sigma\ t_1)}) \ . \qquad \text{4.6.i) with 4.7, Def. } \mathcal{T}_> \end{array}$$

The remainder of the proof is analogous to the proof of Theorem 6.7, we only have to restrict the context from the start so that $\mathcal{T}_> = \mathcal{T}$. □

As already mentioned, however, on term graphs with bound variables $\mathcal{M}_>$ has to be restricted with great care so as not to yield unexpected results, see [Kahl, 1996, Sect. 7.1].

7. Host Construction

The decisive step in the construction of a double-pushout derivation step is the construction of the pushout complement, or host graph when talking about graph grammars — see Sect. 1. As far as we know, there has been no direct universal characterisation of the pushout complement so far. In our generalised, heterogeneous approach, where we are prepared to tolerate at least certain restricting derivations, we are now able to supply such a characterisation, which even turns out to be dual to the heterogeneous pushout.

The "incoming matching category" $\mathcal{M}_<$ is going to be for hosts what the outgoing matching category $\mathcal{M}_>$ is for heterogeneous pushouts; on first reading it is again safe to assume $\mathcal{M}_< = \mathcal{M}_\equiv$.

Definition 7.1. A **host** for a matching $\underline{h : R \to H}$ and an arrow $d : L \xrightarrow{\mathcal{T}_<} R$ of $\mathcal{T}_<$, is a matching $\underline{g : L \to G}$ together with an arrow $e : G \xrightarrow{\mathcal{E}_\mathrm{H}} H$ of $\mathcal{E}_\mathrm{H}$, such that $\langle d, e\rangle : g \xrightarrow{\mathcal{M}_<} h$ is a matching arrow in $\mathcal{M}_<$ and for every matching arrow $\langle d, e'\rangle : \underline{g' : L \to G'} \xrightarrow{\mathcal{M}_<} h$ from $\mathcal{M}_<$ there exists a unique arrow $\varphi : G' \xrightarrow{\mathcal{E}_\mathrm{V}} G$ with $\varphi \mathbin{⨾} e = e'$ and $g' \mathbin{⨾} \varphi = g$. The codomain G of the matching g is called **host object**. □

$$\begin{tikzcd}
R \arrow[d, "h"'] & L \arrow[l, "d"'] \arrow[d, "g"] \arrow[ddr, "h'"] & \\
H & G \arrow[l, "e"'] & \\
& & H' \arrow[ull, "e'"] \arrow[ul, dotted, "\varphi"']
\end{tikzcd}$$

What is now the point of having this definition? In the conventional algebraic approach to rewriting, the pushout complement is not dual to the pushout. In our heterogeneous context, the host *is* indeed dual to the heterogeneous pushout.

In our running example we have:

TRS Theorem 7.2. For the image $d := (L, [\ ], [\{\}]_L, [\varphi]_L)$ of a placeholder arrow (L, φ) from L to R and the image $h := (R, C[\ ], [\sigma]_R, [\{\}]_{(\sigma\ R)})$ of a term homomorphism $(R, C[\ ], \sigma)$ from R to H, a host is given by the image $g := (L, C[\ ], [\sigma]_L, [\{\}]_{(\sigma\ L)})$ of the term homomorphism $(L, C[\ ], \sigma)$ from L to some object G and the image $e := (G, [\ ], [\{\}]_G, [\sigma\ \varphi]_G)$ of the placeholder arrow $(G, \sigma\ \varphi)$ from G to H.

$$\begin{tikzcd}
P \arrow[d, "{h=(R,C[\ ],[\sigma]_R,[\{\}]_{(\sigma\ R)})}"'] & Q \arrow[l, "{d=(L,[\ ],[\{\}]_L,[\varphi]_L)}"'] \arrow[d, "{g=(L,C[\ ],[\sigma]_Q,[\{\}]_{(\sigma\ L)})}"] \\
G & H \arrow[l, "{e=(G,[\ ],[\{\}]_G,[\sigma\ \varphi]_G)}"']
\end{tikzcd}$$

Proof: For any vertical arrow $g' := (L, C'[\ ], [\sigma']_L, [\{\}]_{(\sigma'\ L)})$ from L to some object G' and for any horizontal arrow $e := (G, [\ ], [\{\}]_G, [\varphi']_G)$ from G' to H the equation $d \mathbin{⨾} h = g' \mathbin{⨾} e'$ implies via

$$\begin{aligned}
(L, C[\ ], [\sigma]_L, [\sigma\ \varphi]_{(\sigma\ L)}) \;&=\; d \mathbin{⨾} h \;=\; g' \mathbin{⨾} e' \\
&=\; (L, C'[\ ], [\sigma']_L, [\varphi']_{(\sigma'\ L)})
\end{aligned}$$

that $C'[\ ] = C[\ ]$, $[\sigma']_L = [\sigma]_L$ and $[\varphi']_{(\sigma'\ L)} = [\varphi']_{(\sigma\ L)} = [\sigma\ \varphi]_{(\sigma\ L)}$, from where we see that $g' = g$ and $e' = e$. So just the condition of commutativity is enough to uniquely determine the host object, too. □

In particular, the host object in the example on page 9 can now be found directly via this host construction — note that $[\{x \mapsto t\}]_{U_1[x]} = [\{x \mapsto t, y \mapsto s\}]_{U_1[x]}$:

$$\begin{tikzcd}
\pi(p(x,y)) \arrow[d, "{\{x \mapsto t, y \mapsto s\}}"'] & U_1[x] \arrow[l, "{\{U_1[\delta_1] \mapsto \pi(p(\delta_1, y))\}}"'] \arrow[d, "{\{x \mapsto t\}}"] \\
\pi(p(t,s)) & U_1[t] \arrow[l, "{\{U_1[\delta_1] \mapsto \pi(p(\delta_1, s))\}}"']
\end{tikzcd}$$

With the term graphs of [Kahl, 1996] both the host and the heterogeneous pushout require quite sophisticated restrictions.

Being dual to heterogeneous pushouts, hosts give rise to cartesian arrows and a split fibration. The whole development is of course dual to that of section 5, so we do not need to repeat all of it.

$$\begin{tikzcd} N \arrow[d, "k"] & M \arrow[l, "d_2"'] \arrow[d, "h"] & L \arrow[l, "d_1"'] \arrow[ll, bend right, "d"'] \arrow[dd, "g"] \\ K & H \arrow[l, "e_2"'] & \\ & & G \arrow[ul, "e_3"'] \arrow[ull, "e"] \end{tikzcd}$$

We only demonstrate that in the context of the pattern functor $\mathcal{P}$ from $\mathcal{M}$ to $\mathcal{T}_<$ a cartesian arrow $\langle d_2, e_2 \rangle : h \xrightarrow{\mathcal{M}_<} k$ in $\mathcal{M}_<$ for the arrow $d_2 : M \xrightarrow{\mathcal{T}_<} N$ and the object $\underline{k : N \to K} : \mathcal{M}_<$ in a consistent context gives rise to a host (h, e_2) for (k, d_2).

The definition of cartesian arrows demands, that for any arrow $\langle d, e \rangle : g \xrightarrow{\mathcal{M}_<} k$ and for any arrow $d_1 : L \xrightarrow{\mathcal{T}_<} M$, for which $d_1 \fatsemi d_2 = d$, there is a unique arrow $\langle d_1, e_3 \rangle : g \xrightarrow{\mathcal{M}_<} h$, for which $\langle d_1, e_3 \rangle \fatsemi \langle d_2, e_2 \rangle = \langle d, e \rangle$. By letting $d_1 := \mathrm{id}_{M:\mathcal{T}}$, we obtain exactly the definition of a host for k and d_2 (again, e_3 is in $\mathcal{E}_{\mathrm{V}}$ since $M = L$ with the definition of a consistent context).

We can now go on to define composability of hosts and choices of hosts and show that, via the Grothendiek construction for a functor from $\mathcal{T}_<^{\mathrm{op}}$ to Cat, these give rise to a split fibration, turning the pattern functor on a corresponding restriction $\overline{\mathcal{M}_<}$ of $\mathcal{M}$ induced by $\mathcal{M}_<$ into a split fibration. Therefore host constructions essentially correspond to the splittings for split fibrations. All this is dual to the corresponding parts of 5 and 6, so we need not delve into the details anymore.

In our running example we have:

TRS Theorem 7.3. In the context of term rewriting we have $\mathcal{M}_< = \mathcal{M}_{\mathrm{H}}$, and therefore the pattern functor is a split fibration from $\mathcal{M}$ to $\mathcal{T}$. □

Proof: The only thing that remains to be shown is that the choice of hosts effectively defined in Theorem 7.2 is well-defined. Because of the similarity of that definition with that of Prop. 2.7 for heterogeneous pushouts, the proof is essentially the same as that of Theorem 5.6. □

8. The Fibred Approach to Rewriting Systems

We are now ready to join the different results into a new approach to rewriting: The *fibred approach to rewriting* requires a category of matchings $\mathcal{M}$ together with a pattern functor $\mathcal{P}$ into an underlying category of patterns $\mathcal{T}$ and two subcategories $\overline{\mathcal{M}_<}$ and $\overline{\mathcal{M}_>}$ of $\mathcal{M}$ (that may coincide with $\mathcal{M}$), where $\mathcal{P}$ is a split fibration from $\overline{\mathcal{M}_<}$ to $\mathcal{T}_<$ with splitting Σ and at the same time a split opfibration from $\overline{\mathcal{M}_>}$ to $\mathcal{T}_>$ with splitting Σ^{op}.

Alternatively, it is sufficient (in a consistent rewriting context) to demand a choice of hosts $HOST$ and a choice of heterogeneous pushouts HPO. Σ and Σ^{op} may then easily be reconstructed from $HOST$ and HPO.

Definition 8.1. A **fibre rule** consists of three objects and an arrow $l : G \xrightarrow{\mathcal{T}_<} L$ and an arrow $r : G \xrightarrow{\mathcal{T}_>} R$ in the shape of a span: $L \xleftarrow{l} K \xrightarrow{r} R$. This is essentially the same as in the conventional algebraic (double-pushout) approach.

A **rewrite step** is induced by a matching $\underline{g : L \to A}$ of the left-hand side L of the rule into an application object A. Then we let $\lambda_0 := \Sigma(l, g)$ be the cartesian arrow for l and g, so that λ_0 is of the shape $\langle l, \lambda \rangle$ for an appropriate λ, and we let $\underline{h : K \to H}$ be the domain of this arrow. H is now the host object, and we can proceed to construct the result: let $\langle r, \rho \rangle := \Sigma^{\mathrm{op}}(r, h)$ be the opcartesian arrow for r and the host matching h, and let $\underline{k : R \to B}$

be the codomain of that arrow, then B is the result object of the fibred rewriting step.

$$\begin{tikzcd}
& L \arrow[dl, "g"'] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "h"'] & R \arrow[d, "k"'] \\
A & & H \arrow[ll, "\lambda"'] \arrow[r, "\rho"] & B
\end{tikzcd}$$

Alternatively, we would have calculated $(h, \lambda) := HOST(g, l)$ and $(k, \rho) := HPO(h, r)$. □

Therefore a rule, consisting of arrows in dual directions, is applied by employing dual constructions corresponding to deleting and adding items. And as deleting and adding are not the same, but only dual in a certain sense, part of the additional freedom present in restricting derivations carries over to the categorical construction: The cartesian arrows are a kind of projection, deleting information we do not have to specify in the rule, while the opcartesian arrows correspond to raising a perpendicular, the ending point of which is completely specified by the rule and the host.

Looking at the diagram drawn above in a naïve way, we can arrive at an informal explanation that might shed some light on the ideas behind the fibred rewriting step: Imagine a vertical line erected on the rule through the gluing graph, then the host graph is the result of *projecting* the application graph A onto that line producing the cartesian arrow, whose length is determined by the relative position of A. Continuing, the result graph is arrived at by *erecting* a vertical on that line with its length determined by r — we hardly have another choice.

An important advantage of this approach is that the host object is uniquely determined and characterised in a universal and constructive manner.

In our running example, fibre rules can be constructed for all term rewriting rules and fibred rewriting steps the implement conventional term rewriting steps:

TRS Theorem 8.2. Term rewriting is modelled correctly by the fibred approach in the context given by Theorem 4.7.

Proof: A term rewriting rule is a pair (L, R) of terms, where the free variables of R are contained in those of L.

Application of a term rewriting rule (L, R) to a term A is defined if there are a context $C[\ ]$ and a substitution σ such that $A = C[\sigma\ L]$, and the result of the rewriting step is $B = C[\sigma\ R]$.

We now have to show that the fibred approach in the rewriting context given by Theorem 4.7 exactly emulates this kind of rewriting.

The rule (L, R) is translated into a rule $L \xleftarrow{l} K \xrightarrow{r} R$, where (let n be the number of distinct variables in R) the gluing term K consists of a n-ary placeholder variable and the variables of R; the placeholder arrow equivalence classes for $l = (K, [\ ], [\{\}]_K, [\varphi]_K)$ and $r = (K, [\ ], [\{\}]_K, [\psi]_K)$ are then uniquely determined.

This fibre rule is applicable exactly if there is a morphism of $\mathcal{H}$ from L to A, such a morphism is of the shape $g = (L, C[\ ], [\sigma]_L, [\{\}]_{(\sigma\ L)})$ and exists exactly when $A = C[\sigma\ L]$, so the applicability is the same as by the conventional definition.

The fibred rewriting step is then constructed as sketched in the following diagram:

$$\begin{tikzcd}
& L \arrow[dl, "{(C[\ ],\sigma)}"'] & K \arrow[l, "\varphi"'] \arrow[r, "\psi"] \arrow[d, "{(C[\ ],\sigma)}"'] & R \arrow[d, "{(C[\ ],\sigma)}"'] \\
A & & H \arrow[ll, "\sigma\ \varphi"] \arrow[r, "\sigma\ \psi"'] & B
\end{tikzcd}$$

So as result of the rewriting step we obtain $B = C[\sigma\ R]$, which is again compatible with the conventional definition. □

9. Outlook

In this paper, we have generalised the approach of [Kahl, 1996], which used separated contexts (see page 15), to using consistent contexts instead (see Def. 4.3). The main disadvantage of separated contexts is that composition of rewriting steps is hard to come to grips with. In a separated context, every non-identity pattern arrow is a monomorphism and an epimorphism, but of course not an isomorphism, and no nontrivial pullbacks exist.

In a consistent context, pullbacks can exist — and in our term rewriting example all pullbacks do indeed exist; they are just a form of disunification. An example pullback is the following:

$$\begin{tikzcd}
U_1[x] \arrow[r, "\{U_1[\delta_1] \mapsto U_2[\delta_1, z]\}"] \arrow[d, "\{U_1[\delta_1] \mapsto U_2[\delta_1, y]\}"'] & U_2[x, z] \arrow[d, "\{U_2[\delta_1, \delta_2] \mapsto q(\delta_1, y, \delta_2)\}"] \\
U_2[x, y] \arrow[r, "\{U_2[\delta_1, \delta_2] \mapsto q(\delta_1, \delta_2, z)\}"] & q(x, y, z)
\end{tikzcd}$$

This will, however, never occur in the composition of two rules since neither of the arrows ending in the lower right corner is in $\mathcal{T}_>$. The restriction to $\mathcal{T}_>$ for the right-hand arrow lets all pullbacks that occur in composition of rules contain an identity, so for example the two rules

$$p(x, y) \xleftarrow{\{U_2[\delta_1, \delta_2] \mapsto p(\delta_1, \delta_2)\}} U_2[x, y] \xrightarrow{\{U_2[\delta_1, \delta_2] \mapsto q(\delta_1, \delta_2, \delta_1)\}} q(x, y, x)$$

and

$$q(x, y, x) \xleftarrow{\{U_1[\delta_1 \mapsto q(\delta_1, y, \delta_1)\}} U_1[x] \xrightarrow{\{U_1[\delta_1 \mapsto r(\delta_1\}} r(x)$$

can be composed to one rule

$$p(x, y) \xleftarrow{\{U_1[\delta_1] \mapsto p(\delta_1, y)\}} U_1[x] \xrightarrow{\{U_1[\delta_1] \mapsto r(\delta_1)\}} r(x)$$

via a pullback of pattern arrows:

$$\begin{tikzcd}
 & & U_1[x] \arrow[dl, "\{U_1[\delta_1] \mapsto U_2[\delta_1, y]\}"'] \arrow[dr, "\{\}"] & & \\
 & U_2[x, y] \arrow[dl, "\{U_2[\delta_1, \delta_2] \mapsto p(\delta_1, \delta_2)\}"'] \arrow[dr, "\{U_2[\delta_1, \delta_2] \mapsto q(\delta_1, \delta_2, \delta_1)\}"'] & & U_1[x] \arrow[dl, "\{U_1[\delta_1] \mapsto q(\delta_1, y, \delta_1)\}"'] \arrow[dr, "\{U_1[\delta_1] \mapsto r(\delta_1)\}"'] & \\
p(x, y) & & q(x, y, x) & & r(x)
\end{tikzcd}$$

Note that it is the use of extended dummy patterns instead of dummy patterns in the definition of placeholder substitutions with images containing placeholders that makes nontrivial composition and therewith pullbacks possible. Extended dummy patterns had to permit terms instead of just term variables in order to preserve placeholder substitutions under applications of term substitutions.

With the existence of pullbacks — which may be required to all be of a certain shape according to the choices of $\mathcal{M}_>$ and $\mathcal{M}_<$ — much of the machinery of [Banach and Corradini, 1996] should be usable on fibred rewriting steps just as on the homogeneous double-pushout or double-pullback steps considered there.

Also other *in-the-large* properties of the fibred approach to rewriting, such as parallel moves or canonical derivations, deserve to be investigated after having settled the details of the single rewriting step in this paper.

In a completely different vein, comparison with the *pullback approach* of Bauderon [1995] might be interesting. A point to note in this context is that our host construction is also able to duplicate information just as the pullback construction, although with our fibred approach the general system of information transfer in rewriting is still organised as in the double-pushout approach. (Bauderon's pullback-rewriting should not be mixed up with the "double-pullback graph transition systems" also discussed in [Banach and Corradini, 1996] which are a variant of the double-pushout approach.)

10. Conclusion

We have elaborated the relation between the algebraic approach to rewriting using pushouts and the approach using opfibrations proposed by Banach. In the result we arrived at a generalised form of presentation for the opfibration approach, which in addition is also easier to grasp from the point of view of the pushout approach.

The heterogeneous setting we developed helped us to get a formal grasp on the construction of the host object; in the double-pushout approach it had to be found as a pushout complement, and now it can be arrived at via a host construction which is dual to the construction of the result object — this was impossible in the homogeneous pushout approach.

At the same time, the construction with fibrations comprises important sub-cases of another extension to the classical algebraic approach, namely restricting derivations, however without incurring their somewhat arbitrary over-generality.

Thus we have generalised and unified two variants of the algebraic approach to rewriting into a quite satisfying theoretical setting. We believe that the induced heterogeneous view corresponds better to the nature of rewriting than the (some cases artificial) homogeneity of the classical double-pushout approach.

The single-pushout approach was indeed a first step into this direction, and Banach's idea to use opfibrations was a further step, only that it is burdened by the heritage of the single pushout which should be taken apart for this purpose.

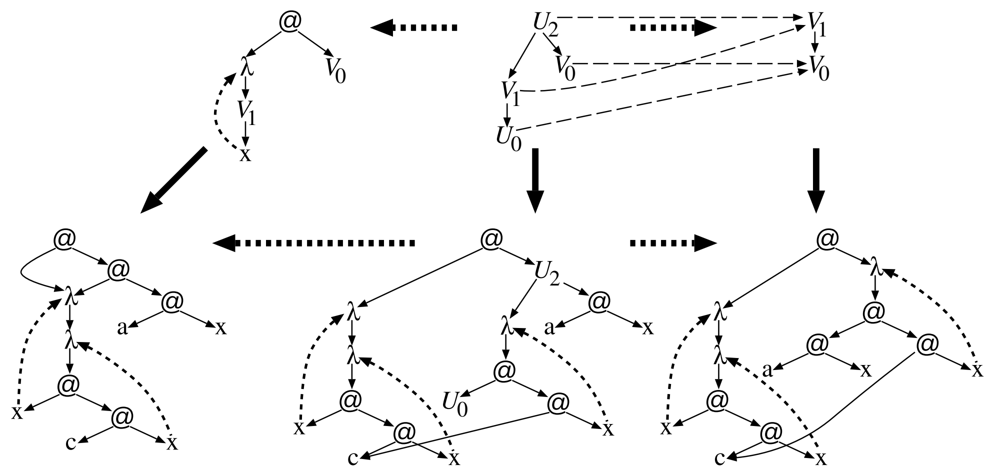
We have shown that in the setting of fibred categories this "taking apart" includes restricting derivations in a natural way. Furthermore, we have been able to obtain an opfibration directly from inside the categories involved in the rewriting step, taking the detour over the Grothendiek category solely to close the selected horizontal matching subcategories naturally when necessary, thus improving on Banach's formalisation.

We also believe that the fibred approach — and that of restricting derivations for that matter — is superior to the single-pushout approach, even to a single-heterogeneous-pushout approach, because the explicit mentioning of the interface and host objects avoids another treacherous homogeneity: Even the simple term rewriting example used throughout this paper required an extension of the considered class of objects through the introduction of placeholder variables for the interface and host objects. The intuition of such extensions of the underlying categories would be lost if one only considered partial morphisms without having a clear understanding of the underlying subobjects.

On the way from the double pushout approach to the fibred approach we may have lost certain graph grammars, for which an appropriate class $\mathcal{H}$ of matching arrows cannot be singled out, but we have, as shown in this paper, won term rewriting systems, and also Klop's [1980] Combinatory Reduction Systems (CRSs).

Furthermore, as already mentioned, we needed these constructions to adapt the algebraic approach to term graphs with bound variables in [Kahl, 1996], corresponding to a term graph implementation of CRSs, and we have thus obtained the first algebraic characterisation of graph reduction, an important foundation of the implementation of functional programming languages.

Here is what a typical fibred rewriting step for the rule of β-reduction looks like; the host construction takes care of the appropriate duplication of the "non-abstractable parts" [Wadsworth, 1971] of the body of the λ-abstraction.



The reduced redex is the second from top; in traditional notation this corresponds to the β-reduction

$$(\lambda x \,.\, \lambda y \,.\, x\ (c\ y))\ ((\lambda x \,.\, \lambda y \,.\, x\ (c\ y))(a\ x)) \quad \longrightarrow_\beta \quad (\lambda x \,.\, \lambda y \,.\, x\ (c\ y))\ (\lambda y \,.\, a\ x\ (c\ y)) \ .$$

We are currently trying to extend the scope of the fibred approach to cover other, more diverse, useful application areas, for example to an extension of the relational diagrams of [Kahl, 1997a].

On the whole, this fibred approach to rewriting extends the scope of the algebraic approach to rewriting to interesting and useful new applications, and also offers new insights into the mechanisms of the algebraic approach and of rewriting in general.

Acknowledgements

Some of the diagrams have been typeset using Paul Taylor's TEX macros for commutative diagrams, others with Francis Borceaux's package. I am grateful to Claudia Hattensperger and Boyd Buchin for comments on draft versions of this paper.

References

[Banach and Corradini, 1996] Richard Banach and Andrea Corradini. An opfibration account of typed DPO and DPB graph transformation: General productions. Technical Report UMCS-96-11-2, Department of Computer Science, University of Manchester, 1996.

[Banach, 1993a] Richard Banach. A fibration semantics for extended term graph rewriting. In M.R. Sleep, M.J. Plasmeijer, and M.C.J.D. van Eekelen, editors, *Term Graph Rewriting: Theory and Practice*, chapter 7, pages 91–100. John Wiley, 1993.

[Banach, 1993b] Richard Banach. Weak fibrations. *Journal of pure and applied algebra*, 86(1):7–22, April 1993.

[Banach, 1994] Richard Banach. Term graph rewriting and garbage collection using opfibrations. *Theoretical Computer Science*, 131(1):29–94, August 1994.

[Barr and Wells, 1990] Michael Barr and Charles Wells. *Category Theory for Computing Science*. Prentice Hall International Series in Computer Science. Prentice Hall, 1990.

[Bauderon, 1995] Michel Bauderon. A uniform approach to graph rewriting: The pullback approach. In Manfred Nagl, editor, *Graph-Theoretic Concepts in Computer Science*, volume 1017 of *Lecture Notes in Computer Science*, pages 101–115, 1995.

[Church, 1941] Alonzo Church. *The Calculi of Lambda Conversion*. Princeton University Press, Princeton, 1941.

[Corradini *et al.*, 1993] Andrea Corradini, Hartmut Ehrig, Martin Löwe, Ugo Montanari, and Francesca Rossi. Note on standard representations of graphs and graph derivations. In Hans Jürgen Schneider and Hartmut Ehrig, editors, *Graph Transformations in Computer Science, Proc. International Workshop Dagstuhl Castle, Germany, January 1993*, volume 776 of *LNCS*, pages 104–118. Springer-Verlag, 1993.

[Curry, 1930] H.B. Curry. Grundlagen der Kombinatorischen Logik. *American Journal of Mathematics*, 52:509–536, 789–834, 1930.

[Ehrig and Parisi-Presicce, 1991] Hartmut Ehrig and Francesco Parisi-Presicce. Interactions between graph grammars and modular system design. (draft), 1991.

[Ehrig and Parisi-Presicce, 1993] Hartmut Ehrig and Francesco Parisi-Presicce. Interaction between algebraic specification grammars and modular system design. In *AMAST '93*, 1993.

[Ehrig *et al.*, 1973] Hartmut Ehrig, M. Pfender, and H. J. Schneider. Graph grammars: An algebraic approach. In *Proc. IEEE Conf. SWAT'73*, pages 167–180, Iowa City, 1973.

[Ehrig *et al.*, 1990a] Hartmut Ehrig, Annegret Habel, Hans-Jörg Kreowski, and Francesco Parisi-Presicce. From graph grammars to high level replacement systems. In Ehrig et al. [1990c], pages 269–287.

[Ehrig *et al.*, 1990b] Hartmut Ehrig, Martin Korff, and Michael Löwe. Tutorial introduction to the algebraic approach of graph grammars based on double and single pushouts. In Ehrig et al. [1990c], pages 24–37.

[Ehrig *et al.*, 1990c] Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors. *Graph-Grammars and Their Application to Computer Science, 4th International Workshop*, volume 532 of *Lecture Notes in Computer Science*, Bremen, Germany, March 1990. Springer-Verlag.

[Ehrig, 1978] Hartmut Ehrig. Introduction to the algebraic theory of graph grammars. In Volker Claus, Hartmut Ehrig, and Grzegorz Rozenberg, editors, *Graph-Grammars and Their Application to Computer Science and Biology, International Workshop*, volume 73 of *Lecture Notes in Computer Science*, pages 1–69, Bad Honnef, November 1978. Springer-Verlag.

[Gaducci, 1996] Fabio Gaducci. *On the Algebraic Approach to Concurrent Term Rewriting*. PhD thesis, Università degli studi di Pisa, Dipartamento di Informatica, March 1996.

[Glauert *et al.*, 1990] J. R. W. Glauert, J. R. Kennaway, and M. R. Sleep. Dactl: An experimental graph rewriting language. In Ehrig et al. [1990c], pages 378–395.

[Kahl, 1996] Wolfram Kahl. *Algebraische Termgraphersetzung mit gebundenen Variablen*. Reihe Informatik. Herbert Utz Verlag Wissenschaft, München, 1996. ISBN 3-931327-60-4; also doctoral dissertation at Fakultät für Informatik, Universität der Bundeswehr München.

[Kahl, 1997a] Wolfram Kahl. Algebraic graph derivations for graphical calculi. In Fabrizio d'Amore, Paola G. Franciosa, and Alberto Marchetti-Spaccamela, editors, *Graph Theoretic Concepts in Computer Science, 22nd International Workshop, WG '96, Caddenabbia, Italy, June 1996, Proceedings*, volume 1197 of *LNCS*, pages 224–238. Springer-Verlag, 1997.

[Kahl, 1997b] Wolfram Kahl. Relational treatment of term graphs with bound variables. *Journal of the IGPL*, 1997. to appear.

[Kennaway, 1987] Richard Kennaway. On "On Graph Rewritings". *Theoretical Computer Science*, 52:37–58, 1987.

[Kennaway, 1990] Richard Kennaway. Graph rewriting in some categories of partial morphisms. In Ehrig et al. [1990c], pages 490–504.

[Klop, 1980] Jan Willem Klop. Combinatory reduction systems. Mathematical Centre Tracts 127, Centre for Mathematics and Computer Science, Amsterdam, 1980. PhDThesis.

[Löwe, 1990] Michael Löwe. Algebraic approach to graph transformation based on single pushout derivations. Technical Report 90/05, TU Berlin, 1990.

[Padberg, 1993] Julia Padberg. Survey of high-level replacement systems. Forschungsbericht des Fachbereichs Informatik 93-8, Technische Universität Berlin, March 1993.

[Parisi-Presicce, 1993] Francesco Parisi-Presicce. Single vs. double pushout derivation of graphs. In Ernst Mayr, editor, *Proc. 18th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 657 of *LNCS*, pages 248–262, Wiesbaden, June 1993. Springer-Verlag.

[Raoult, 1984] J. C. Raoult. On graph rewritings. *Theoretical Computer Science*, 32:1–24, 1984.

[Spivey, 1989] J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International Series in Computer Science. Prentice Hall, 1989.

[Stell, 1994] John G. Stell. Modelling term-rewriting systems by sesqui-categories. Technical Report TR94-02, Department of Computer Science, Keele University, Keele, Staffordshire, UK, January 1994.

[Wadsworth, 1971] Christopher Peter Wadsworth. *Semantics and Pragmatics of the Lambda Calculus*. D.Phil. thesis, Oxford University, September 1971.

Address for correspondence: Wolfram Kahl
UniBwM, INF2
D-85577 Neubiberg, Germany

Phone: +49-89-6004-3198
Fax: +49-89-6004-3560
e-mail: `kahl@informatik.unibw-muenchen.de`
URL: `http://inf2-www.informatik.unibw-muenchen.de/kahl/`