ISSN 1361 - 6161

THE UNIVERSITY
of MANCHESTER

Computer Science
University of Manchester

An Opfibration Account of Typed DPO and DPB Graph Transformation: General Productions

R. Banach, A. Corradini

Department of Computer Science
University of Manchester
Technical Report Series
UMCS-96-11-2

An Opfibration Account of Typed DPO and DPB Graph Transformation: General Productions*

R. Banach, A. Corradini

Department of Computer Science
University of Manchester
Oxford Road, Manchester, U.K.
banach@cs.man.ac.uk

Dipartimento di Informatica
Università di Pisa
Corso Italia 40, Pisa, Italy
andrea@di.unipi.it

31 November 1996

Recent technical reports issued by the Department of Computer Science, Manchester University, are available by anonymous ftp from ftp.cs.man.ac.uk in the directory pub/TR. The files are stored as PostScript, in compressed form, with the report number as filename. They can also be obtained on WWW via URL http://www.cs.man.ac.uk/csonly/cstechrep/index.html. Alternatively, all reports are available by post from The Computer Library, Department of Computer Science, The University, Oxford Road, Manchester M13 9PL, UK.

* Refereed by: Reiko Heckel.

An Opfibration Account of Typed DPO and DPB Graph Transformation: General Productions

R. Banach
Computer Science Dept., Manchester University, Manchester, M13 9PL, U.K.
banach@cs.man.ac.uk

A. Corradini
Dipartimento di Informatica, Università di Pisa, Corso Italia 40, Pisa, Italy.
andrea@di.unipi.it

Abstract: The "in the large" properties of typed graph transformation systems in the double pushout framework and a double pullback variation of it for arbitrary (i.e. not necessarily injective) productions are reexamined. Preceding accounts utilising a fixed choice of pullbacks (whether adopted directly or via partial morphisms) are seen to be excessively sensitive to the precise graphs involved for comfort. The properties of a fixed choice of pullbacks are seen to be analogous to those of a splitting of an opfibration, and so a framework of graph transformations utilising unsplit opfibrations is constructed, which proves to be free of the excessive sensitivity noted. The extra generality means that different constructions on graph transformation steps, require the use of different classes of diagrams for their description, reflecting the different isomorphism properties for the graphs involved in the diagrams in each particular case. In the new framework, graph grammars and graph transition systems, derivation grammars and derivation systems, computation grammars computation systems and computation sequence systems, are all considered. In each case the adjunction appropriate to the relationship between grammar and system is constructed. These are variously left or right adjunctions, conventional or weak adjunctions, depending on the details of the particular case in question.

Key Words: Graph grammar, DPO and DPB rewriting, opfibrations.

1 Introduction

In Corradini et al. (1996b), a categorical account of typed double pushout (DPO) rewriting was given by constructing mappings from arrows a of the category of type graphs and concrete spans, to functors between categories of (in turn) grammars, transition systems, and derivation systems typed over the domain and codomain of a. Essentially the same idea will work for the single pushout approach (Löwe (1991), Löwe and Ehrig (1991)) by reinterpreting the span used in a double pushout rule (or direct derivation step) as a partial morphism (see Ribeiro (1996) who also uses partial morphism spans for retyping). In Ehrig et al. (1996), Heckel et al. (1997), a similar treatment was presented for a double pullback rewriting construction, essentially intended as a "looser" version of the double pushout construction rather than an independent construction (and in particular not to be confused with work of Bauderon on rewriting via pullbacks, see eg. Bauderon (1995)), whose close relationship to the traditional DPO construction relies crucially on the injectivity of the arrows in a DPO rule (or direct derivation step) in the two formulations being considered. In the former work the composition of concrete spans in the category of type graphs arises from a fixed choice of pullbacks; and the steps from categories of grammars to categories of transition systems and derivation systems are made via free constructions. In the latter the same effect for the category of type graphs is achieved by considering only partial morphisms of type graphs, a ruse which effectively forces a specific choice of pullbacks since one of the arrows of a span that is a concrete partial morphism must be an inclusion; and the steps from categories of grammars to categories of transition systems and derivation systems are made via a Kleisli construction and a co-free construction. In fact partial morphism retyping and the use of the Kleisli construction in this manner arose in Heckel et al. (1996).

One notices two related things in these approaches. The first is that because a fixed choice of pullbacks is needed to enable the composition of two concrete spans to be a third concrete span, the action of the functors constructed subsequently is extremely fussy about the concrete graphs involved. Eg. when one graph grammar is the functorial image of another, changing the start graph in the target to an isomorphic graph will not do; the start graph has to be precisely the one given by the functor, and the grammar with the other start graph is not related to the source. The second thing is that the axioms imposed on the choice of pullbacks in order that the various constructions work smoothly, are just like those required for the splitting of a split opfibration.

This suggests that the functors corresponding to the arrows a mentioned above, indeed glue together to form an indexed category corresponding to a split opfibration. And furthermore that there is an underlying opfibration behind the constructions which is not split, so that the fussiness regarding specific graphs may well be attributable to having forced a splitting where there was no naturally arising one. This turns out to be true, and there is a general lesson to be learned from this, which states the following. When the search for a functor which describes some situation categorically forces one to enhance the data being managed by the adoption of specific choices, it may well be better not to make the choices, to be satisfied with a pseudofunctor (or equivalent construction) rather than a functor, and to describe the converse construction using an (unsplit) opfibration. Similar remarks apply to contravariant functors and (unsplit) fibrations. All of this is in contrast to the use of split opfibrations as a methodology for describing graph rewriting in Banach (1993, 1994, 1995). The relationship between these techniques will be explored elsewhere.

In this paper we re-engineer and extend the central material of the key papers refered to in the first paragraph above, resisting the temptation to force a fixed choice of pullbacks (by whatever means). As predicted, it turns out that the flexibility in choice of pullbacks (up to isomorphism, as per standard theory) is neatly captured in an (unsplit) opfibration. This relationship extends to categories of transitions, derivations, and other constructions. Furthermore, we do not insist that productions are injective — the conventional supposition — which lends an additional degree of generality to our work. (Injectivity entails limited but significant changes to the material described here; the whole theory is redone *ab initio* taking into account the consequences of injectivity in the companion paper Banach and Corradini (1996).) A(nother) consequence of lack of injectivity is that the connection between the DPO and DPB constructions is considerably weakened, which makes the DPB theory of less intrinsic interest; however we include it nontheless as a specific illustration of the idea that "all" diagram-based graph rewriting techniqes might have corresponding typed reformulations roughly along the lines of the theory presented here. It is also interesting to note that despite the weakened connection, most of the technicalities in the present paper are no worse than when we have injectivity.

The rest of this paper is as follows. In section 2 we recall the basic concepts we need on graphs, double pushout and double pullback rewriting, and opfibrations. The opfibrations material, illustrating Grothendieck's correspondence between split opfibrations $\mathcal{P}: \mathcal{E} \to \mathcal{B}$ and functors $\mathcal{F}: \mathcal{B} \to \mathcal{C}at$, is a useful backdrop to the constructions in section 3, where what is in effect a nondeterministic analogue of the Grothendieck construction is made, creating opfibrations $\mathcal{P}: \mathcal{E} \to \mathcal{B}$ which are not split, in order to build the categories that prove useful later.

In section 4 we address the category of graph grammars, while in section 5 we deal with the category of transition systems. The relationship between a graph grammar and the transition system it freely generates is shown to be a weak left adjoint (see eg. Krishnan (1981) whose terminology we use), the weakness of the adjunction being attributable to the extra flexibility bought by the unsplitness of the relevant opfibration.

In section 6 we introduce the notion of derivation grammar, related to that of graph grammar, but differing in the isomorphism properties permitted for the middle graph of a production, because of the different kind of composition property desired. In section 7, we construct the corresponding systems, the derivation systems, and show that the relationship between a derivation grammar and the derivation system it freely generates is (this time an honest to goodness) left adjunction.

In section 8 we introduce computation grammars, where this time the elements of the grammar are complete derivation steps up to the appropriate isomorphisms, this being forced by the notion of composition to be used later. It is shown that computation grammars can be freely generated from derivation grammars, and the relationship between them is a weak right adjunction. In section 9, we construct the computation systems, exploiting the property of (an extension to non-injective productions of) R-composition for derivation steps introduced in Banach (1995). The relationship between a computation grammar and the computation system it freely generates turns out to be another honest to goodness left adjunction. Finally in section 10, we consider computation sequence systems, generated from computation grammars by another free construction, but where the additional data are this time rather conventional sequences of computation steps. The appropriate adjunction is this time an honest right adjunction. Section 11 contains further discussion and concludes.

2 Basic Concepts

In this section we review graphs, opfibrations and the classic double pushout approach to graph rewriting.

2.1 Concrete and Abstract Graphs and Spans

Definition 2.1.1 A concrete graph G is a tuple (E, V, s, t, l_E, l_V) where E and V are (finite) sets of edges and vertices respectively, $s, t : E \to V$ map each edge to its source and target respectively, and $l_E : E \to \Omega_E$, $l_V : V \to \Omega_V$ map edges and vertices to their labels drawn from the edge and vertex label alphabets Ω_E, Ω_V.

In fact the vertex and edge labels form a kind of typing system, classifying vertices and edges in a rather crude manner, and below we will be concerned with a more sophisticated kind of type system, where types are themselves graphs. All that we say subsequently will carry through unaltered irrespective of whether labels are present or not, and readers may prefer to forget about the labels altogether.

Definition 2.1.2 A concrete graph morphism $f : G \to G'$ is a pair of functions $f_E : E \to E'$, $f_V : V \to V'$ such that Fig. 1 below commutes in the obvious way

$$\begin{tikzcd}
& E \arrow[r, shift left, "s"] \arrow[r, shift right, "t"'] \arrow[dl, "l_E"'] \arrow[dd, "f_E"] & V \arrow[dr, "l_V"] \arrow[dd, "f_V"'] & \\
\Omega_E & & & \Omega_V \\
& E' \arrow[ul, "l_E'"] \arrow[r, shift left, "s'"] \arrow[r, shift right, "t'"'] & V' \arrow[ur, "l_V'"'] &
\end{tikzcd}$$

Fig. 1

Where necessary, we will systematically use primes, or subscripts identifying the graph (eg. $l_{V,G}$) on the various components, when several graphs are being discussed at once, to disambiguate as above. This gives us the category $\mathcal{Gr}$ of concrete graphs and morphisms with obvious identities and composition of morphisms.

One of the things we wish to do in this paper is to raise the level of abstraction from individual concrete graphs (which are in any case inappropriate for certain of the constructions), to abstract graphs, which are suitable isomorphism classes of concrete graphs. But we cannot just naively collect together all the isomorphic objects, since in general arrows do not behave well. To overcome this, we need the notion of standard isomorphism between isomorphic objects in $\mathcal{Gr}$, as explained in Corradini et al. (1994a,b). We can get these easily by picking a skeleton subcategory of $\mathcal{Gr}$.

Thus for each isomorphism class of graphs we pick one graph G_0, say, and for each graph G isomorphic to G_0, we choose arbitrarily one isomorphism g : $G_0 \to G$ (choosing of course the identity if $G = G_0$) to act as the standard one between the two. Then given any H also isomorphic to G_0, with standard isomorphism $h : G_0 \to H$, a unique standard isomorphism from G to H is given by $h \circ g^{-1}$.

Definition 2.1.3 The category $\langle \mathcal{Gr} \rangle$ has as objects graphs up to standard isomorphism written $\langle G \rangle$, and as arrows isomorphism classes of concrete arrows $\langle h : G \to H \rangle$ up to standard isomorphism. The latter $\langle h : G \to H \rangle$ say, are defined by picking a concrete arrow $h : G \to H$ say, and extending to all other members of $\langle G\rangle$ and $\langle H\rangle$ in the only possible way using the standard isomorphisms. Thus for any $\langle h : G \to H\rangle$ there is a bijection between concrete arrows $h : G \to H$ in $\langle h : G \to H\rangle$, and ordered pairs G, H taken from $\langle G\rangle$ and $\langle H\rangle$. Identities are the equivalence classes of concrete identities, and composition of arrows $\langle h : G \to H\rangle$ and $\langle k : H \to K\rangle$ is given by composing the concrete arrows in the two respective classes in the only possible way, which forms another equivalence class[1].

When we need to refer to the isomorphism class of a graph G but admitting arbitrary isomorphisms we write $[G]$. We cannot build a category out of these as we said above. Nevertheless there are situations in which such classes are of benefit, namely in the interiors of diagrams. If one has diagrams of a certain shape, one often wishes to compose them in some way to build some category. Thus certain of the objects in diagram A need to be identified with objects in diagram B in order to form the composition. These are the exterior objects. The remaining objects are the interior ones. If the diagrams are in $\mathcal{G}r$, then to make the construction up to isomorphism, one needs to take the exterior objects up to standard isomorphism, while the interior objects may be taken up to arbitrary isomorphism. This is because the consistency of arrow composition in the face of isomorphisms that must hold for the construction to be valid as a whole, must be supplied externally when one considers identifying the exterior objects of two unrelated diagrams (hence standard isomorphisms), while for the interior objects, all the arrows that are incident on them in the result are present already in their original diagrams, and the consistency of arrow composition comes from requiring the commutativity of the original diagram to be preserved under (now arbitrary) isomorphisms of the interior objects. Of course all of this depends on how one chooses to combine the diagrams. If one chooses to combine the same diagrams in a different manner, particular objects may well switch roles, changing from interior to exterior or vice versa. A further possibility that will arise, is the intermediate one in which the exterior objects are taken as concrete, while the interior ones are up to arbitrary isomorphism. We will see examples of all of these things shortly.

1. Note that foundationally speaking, the above definition is suspect. Since the collection of concrete graphs isomorphic to any given one forms a proper class, so does the collection of concrete arrows in any isomorphism class of arrows between any two abstract graphs. Now we are in trouble since the homset between any two objects in a category (eg. two abstract graphs in $\langle \mathcal{G}r\rangle$) must be a set, and a set cannot have proper classes as members. We will not worry unduly about this, since we never use *membership* of these large collections in any way that could cause us problems. Nevertheless we will state all results in both concrete and abstract form, each prone to its own doubts foundationally if understood naively. Perhaps the only formulations of what we do that are truly free from foundational defects, are (1) a formulation in terms of Grothendieck's Universes, where proper classes can in effect be chosen small enough, (2) a formulation in terms of the previously selected skeleton. One can regard the concrete versions of all of our results as a notation for the former, and the abstract versions as a notation for the latter. In the latter case $\langle A\rangle$ is the skeleton graph up to identity corresponding to the concrete graph A, and $[A]$ is the same thing up to arbitrary automorphisms.

The comma category $(\mathcal{G}r{\downarrow}A)$ has the abstract counterpart $(\langle \mathcal{G}r\rangle{\downarrow}\langle A\rangle)$ consisting of abstract arrows $\langle f : X \to Y\rangle : \langle x : X \to A\rangle \to \langle y : Y \to A\rangle$. This is perhaps the simplest example of diagram composition; all items must be up to standard isomorphism.

A concrete span is a coinitial pair of arrows, i.e a diagram of the form $A \leftarrow B \to C$. Concrete spans support a notion of span morphism which takes one concrete span $A \leftarrow B \to C$ to another $A' \leftarrow B' \to C'$ via a triple of concrete arrows ($a : A \to A'$, $b : B \to B'$, $c : C \to C'$) such that Fig. 2 commutes. Concrete spans with span morphisms as arrows and obvious identities thus form a category $\mathcal{S}p$. There is a subcategory of $\mathcal{S}p$ called $\mathcal{S}p$-DPO in which both commuting squares of Fig. 2, are pushouts in $\mathcal{G}r$. A further subcategory of $\mathcal{S}p$, called $\mathcal{S}p$-DPB features arrows which are span morphisms in which both squares are pullbacks.

$$\begin{tikzcd}
A \arrow[d] & B \arrow[l] \arrow[r] \arrow[d] & C \arrow[d] \\
A' & B' \arrow[l] \arrow[r] & C'
\end{tikzcd}$$

Fig. 2

A special abstract span is an abstract diagram of the form $\langle A\rangle \leftarrow \langle B\rangle \to \langle C\rangle$, i.e. one for which any two concrete representatives $A \leftarrow B \to C$, $A' \leftarrow B' \to C'$ must be related by Fig. 2 such that the three vertical arrows are standard isomorphisms. For special abstract spans there is a notion of span morphism which takes one special abstract span $\langle A\rangle \leftarrow \langle B\rangle \to \langle C\rangle$ to another $\langle A'\rangle \leftarrow \langle B'\rangle \to \langle C'\rangle$ via a triple of abstract arrows ($\langle a : A \to A'\rangle, \langle b : B \to B'\rangle, \langle c : C \to C'\rangle$) such that the abstract analogue of Fig. 2 commutes. Note that all the objects in such a diagram are exterior, so all must be defined only up to standard isomorphism. Special abstract spans with span morphisms as arrows and obvious identities thus form a category $\langle \mathcal{S}p\rangle$. There is a subcategory of $\langle \mathcal{S}p\rangle$ called $\langle \mathcal{S}p\text{-DPO}\rangle$ in which both commuting squares in any concrete representative of a span morphism (which looks like Fig. 2 once more), are pushouts. A further subcategory of $\langle \mathcal{S}p\rangle$, called $\langle \mathcal{S}p\text{-DPB}\rangle$ features arrows which are span morphisms in which both squares are pullbacks.

A semiconcrete span is a partially abstract diagram of the form $A \leftarrow [B] \to C$. This means that any two concrete representatives $A \leftarrow B \to C$, $A \leftarrow B' \to C$ must be related as in Fig. 3, in which the two outer graphs are the same, and the vertical arrow is an arbitrary isomorphism.

Because $\mathcal{G}r$ has all pullbacks (being a diagram category of $\mathcal{S}et$), semiconcrete spans support a notion of horizontal composition. Thus if $A \leftarrow [B] \to C$ and $C \leftarrow [D] \to E$ are two semiconcrete spans, we compose them to get $A \leftarrow [M] \to E$ where $B \leftarrow M \to D$ is a pullback of $B \to C \leftarrow D$ for some arbitrary concrete arrows $B \to C$ and $C \leftarrow D$,

$$\begin{tikzcd}
& B \arrow[dl] \arrow[dr] \arrow[dd] & \\
A & & C \\
& B' \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Fig. 3

elements of the isomorphism classes of arrow $[B] \rightarrow C$ and $C \leftarrow [D]$. This is well defined by pullback properties. Note how pullbacks being defined only up to arbitrary isomorphism gells neatly with our notion of semiconcrete span. Concrete graphs with semiconcrete spans as arrows and obvious identities $A \leftarrow [A] \rightarrow A : A \rightarrow A$ thus form a category $\mathcal{G}r\text{-}\mathcal{S}p$.

An abstract span is an abstract diagram of the form $\langle A \rangle \leftarrow [B] \rightarrow \langle C \rangle$, where the requirement is that for two concrete representatives $A \leftarrow B \rightarrow C$, $A' \leftarrow B' \rightarrow C'$, Fig. 2 commutes, such that the two outer vertical arrows are standard isomorphisms while the inner one is an arbitrary isomorphism.

Abstract spans support a notion of horizontal composition just like semiconcrete spans. Because the outer vertical arrows of Fig. 2 are unique for the given pairs of source and target objects, the reasoning that justifies the composition is essentially as above. Thus $\langle A \rangle \leftarrow [B] \rightarrow \langle C \rangle$ and $\langle C \rangle \leftarrow [D] \rightarrow \langle E \rangle$ compose to give $\langle A \rangle \leftarrow [M] \rightarrow \langle E \rangle$ where M is as previously, and the "up to isomorphism" nature of pullbacks again fits well. Graphs up to standard isomorphism with abstract spans $\langle A \rangle \leftarrow [B] \rightarrow \langle C \rangle : \langle A \rangle \rightarrow \langle C \rangle$ as arrows thus form a category $\langle \mathcal{G}r\text{-}\mathcal{S}p \rangle$.

Note how concrete spans when abstracted correspond to special abstract spans, and semiconcrete spans when abstracted correspond to abstract spans. To do more requires extra ad hoc machinery. For example to compose concrete or special abstract spans horizontally, we must put in place a fixed choice of pullbacks (see Corradini et al. (1996b), Ehrig et al. (1996), and Heckel et al. (1997)).

The general pattern is clear. Interior objects of a diagram should be up to arbitrary isomorphism, while the exterior ones can be taken as concrete, or up to standard isomorphism, and in fact there is a meta level result which states that any proposition that holds for the exterior-concrete formulation, holds also in a suitable form for the exterior--standard-isomorphism formulation. This works by noticing that when one replaces a commuting triangle (such as ABB' in Fig. 3) in the exterior-concrete version by a commuting square (such as $ABB'A'$ in Fig. 2) in the exterior-standard-isomorphism version, the isomorphism $A \rightarrow A'$ ensures that $ABB'A'$ factors into the original triangle ABB' together with the unique triangle $AA'B'$, and thus one adjoins a small amount of fairly standard reasoning about triangles such as $AA'B'$ to the original reasoning about the exterior-concrete case. Below we will state results in both formulations, but typically prove them only in the simpler exterior-concrete case.

Definition 2.1.4 A concrete production is a concrete span ($l : K \to L,\ r : K \to R$) in $\mathcal{G}r$, i.e. an object of $\mathcal{S}p$. (In particular a production is not required to be injective.) A special abstract production is thus merely an object of $\langle \mathcal{S}p \rangle$.

Definition 2.1.5 A direct DPO derivation of H from G using a given production, is a concrete DPO span morphism (g, d, h) i.e. an arrow of $\mathcal{S}p$-DPO (see Fig. 4). A direct DPB derivation of H from G using a given production, is a concrete DPB span morphism (g, d, h) i.e. an arrow of $\mathcal{S}p$-DPB (see Fig. 4 again).

Definition 2.1.6 A special abstract DPO direct derivation is an arrow of $\langle \mathcal{S}p$-DPO$\rangle$. A special abstract DPB direct derivation is an arrow of $\langle \mathcal{S}p$-DPB$\rangle$.

Note that such definitions remember in the derivation step not only the source and result graphs of the step, but the rule used, and all the other data too.

$$\begin{tikzcd}
L \arrow[d, "g"'] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "d"] & R \arrow[d, "h"] \\
G & D \arrow[l, "l^*"] \arrow[r, "r^*"'] & H
\end{tikzcd}$$

Fig. 4

Consider Fig. 4 again, but now taking K and D up to arbitrary isomorphism. When we need an explicit notation, we will write such a diagram as (($L \leftarrow [K] \to R$) -[g,d,h]-› ($G \leftarrow [D] \to H$)).

Definition 2.1.7 A semiconcrete production is a semiconcrete span, i.e. an arrow of $\mathcal{G}r$-$\mathcal{S}p$, ($L \leftarrow [K] \to R$). An abstract production is thus an arrow of $\langle \mathcal{G}r$-$\mathcal{S}p\rangle$.

Definition 2.1.8 A semiconcrete DPO derivation step is a diagram of the form (($L \leftarrow [K] \to R$) -[g,d,h]-› ($G \leftarrow [D] \to H$)) where both squares are PO's. Semiconcrete DPO derivation steps form a discrete category DPO$\mathcal{D}er\mathcal{S}t$. A semiconcrete DPB derivation step is a diagram of the form (($L \leftarrow [K] \to R$) -[g,d,h]-› ($G \leftarrow [D] \to H$)) where both squares are PB's. Semiconcrete DPB derivation steps form a discrete category DPB-$\mathcal{D}er\mathcal{S}t$.

Definition 2.1.9 An abstract DPO derivation step is a diagram of the form (($\langle L \rangle \leftarrow [K] \to \langle R \rangle$) -[$g$,$d$,$h$]-› ($\langle G \rangle \leftarrow [D] \to \langle H \rangle$)) where both squares are abstract PO's. Abstract DPO derivation steps form a discrete category $\langle$DPO$\mathcal{D}er\mathcal{S}t\rangle$. An abstract DPB derivation step is a diagram of the form (($\langle L \rangle \leftarrow [K] \to \langle R \rangle$) -[$g$,$d$,$h$]-› ($\langle G \rangle \leftarrow [D] \to \langle H \rangle$)) where both squares are abstract PB's. Abstract DPB derivation steps form a discrete category $\langle$DPB$\mathcal{D}er\mathcal{S}t\rangle$.

Note the contrast between the last three definitions and the preceding three. In definitions 2.1.4 – 2.1.6, a production is a kind of diagram, and a derivation step is a morphism of that kind of diagram. In definitions 2.1.7 – 2.1.9, a production is also a diagram, but a derivation step is not a morphism of them because some objects are not up to identity or standard isomorphism; instead a derivation step is a different kind of diagram. This has repercussions later in the paper as we consider different ways of composing derivation data.

2.2 Opfibrations

Let $\mathcal{P} : \mathcal{E} \to \mathcal{B}$ be a functor from the subject category $\mathcal{E}$ to the base $\mathcal{B}$. Suppose $\mathcal{P}(e_1 : E_0 \to E_1) = b_1 : B_0 \to B_1$. The arrow $e_1 : E_0 \to E_1$ is opcartesian for B_0 and b_1, iff for every arrow $e_{12} : E_0 \to E_2$ and any $b_2 : B_1 \to B_2$ such that $\mathcal{P}(e_{12} : E_0 \to E_2) = b_2 \circ b_1 : B_0 \to B_2$, we have a unique $\theta_2 : E_1 \to E_2$ such that $e_{12} = \theta_2 \circ e_1$ and $\mathcal{P}(\theta_2) = b_2$. See Fig. 5. An opfibration is a functor $\mathcal{P} : \mathcal{E} \to \mathcal{B}$, such that for every pair $(E, b : \mathcal{P}(E) \to B)$, there is an opcartesian arrow for E and b. A particular choice of opcartesian arrow $\kappa(E, b)$ for each pair (E, b) is called an opcleavage κ of the opfibration.

$$
\begin{tikzcd}
 & & E_2 \arrow[dd, "\mathcal{P}"] \\
E_0 \arrow[rru, "e_{12}"] \arrow[r, "e_1"'] \arrow[d, "\mathcal{P}"'] & E_1 \arrow[ru, "\theta_2"'] \arrow[d, "\mathcal{P}"] & \\
B_0 \arrow[r, "b_1"'] & B_1 \arrow[r, "b_2"'] & B_2
\end{tikzcd}
$$

Fig. 5

In general, writing $b = b : B_0 \to B_1$, any arbitrary choice of opcleavage induces a functor $F_{\kappa(-,\, b)} : \mathcal{P}^{-1}(B_0) \to \mathcal{P}^{-1}(B_1)$ where $\mathcal{P}^{-1}(B_i)$ $(i = 0, 1)$ is the subcategory of $\mathcal{E}$ over B_i. This works by $F_{\kappa(-,\, b)}(E_0) = \mathrm{cod}(\kappa(E, b))$ and $F_{\kappa(-,\, b)}(e : E_0 \to E_1) = \theta : \mathrm{cod}(\kappa(E_0, b)) \to \mathrm{cod}(\kappa(E_1, b) \circ e)$ where θ is the unique arrow promised by the universal property. In general, there are natural isomorphisms between $\mathrm{Id}(\mathcal{P}^{-1}(B))$ and $F_{\kappa(-,\, \mathrm{id} : B \to B)}$, and also between $F_{\kappa(-,\, b_2 \circ b_1)}$ and $F_{\kappa(-,\, b_2)} \circ F_{\kappa(-,\, b_1)}$. If $\kappa(E, \mathrm{id} : \mathcal{P}(E) \to \mathcal{P}(E)) = \mathrm{id}_E$ for all E, and $\kappa(E, b_2 \circ b_1) = \kappa(\mathrm{cod}(\kappa(E, b_1)), b_2) \circ \kappa(E, b_1)$ for all relevant E, b_1, b_2, we say that the opfibration is split and the opcleavage is a splitting.

For split opfibrations (with a specified splitting) we have the theorem of Grothendieck which states that they correspond exactly to functors $\mathcal{F} : \mathcal{B} \to \mathcal{C}at$. The functors $\mathcal{P} : \mathcal{E} \to \mathcal{B}$ and $\mathcal{F} : \mathcal{B} \to \mathcal{C}at$ determine each other. From $\mathcal{F}$ we reconstruct $\mathcal{E}$ up to isomorphism by the Grothendieck construction, which builds $\mathcal{G}(\mathcal{B}, \mathcal{F})$, the Grothendieck category of $\mathcal{B}$ and $\mathcal{F}$. The objects of $\mathcal{G}(\mathcal{B}, \mathcal{F})$ are (E_0, B_0) for E_0 in $\mathcal{F}(B_0)$ for B_0 an object of $\mathcal{B}$, and the arrows of $\mathcal{G}(\mathcal{B}, \mathcal{F})$ are $(e_1 : \mathcal{F}(b_1)(E_0) \to E_1, b_1) : (E_0, B_0) \to (E_1, B_1)$ for $b_1 : B_0 \to B_1$ an arrow of $\mathcal{B}$; with composition of $(e_1 : \mathcal{F}(b_1)(E_0) \to E_1, b_1) : (E_0, B_0) \to$

(E_1, B_1) and $(e_2 : \mathcal{F}(b_2)(E_1) \to E_2, b_2) : (E_1, B_1) \to (E_2, B_2)$ being given by $(e_2 \circ \mathcal{F}(b_2)(e_1) : \mathcal{F}(b_2 \circ b_1)(E_0) \to E_2, b_2 \circ b_1) : (E_0, B_0) \to (E_2, B_2)$; and with obvious identities.

Fig. 6 shows how the components of the Grothendieck category relate to one another, the picture on the right showing the constituent parts of an arrow of $\mathcal{G}(\mathcal{B}, \mathcal{F})$, with other related data shown dashed. Note in particular that such an arrow consists of two parts, a change of base arrow b_1 and an in-fibre morphism e_1. For an accessible introduction to opfibrations, and further key references, see Barr and Wells (1990).

$$\begin{tikzcd}
 & \mathcal{E} \arrow[d, "\mathcal{P}"'] \\
\mathcal{C}at & \mathcal{B} \arrow[l, "\mathcal{F}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
 & E_1 \\
E_0 \arrow[r, dashed] \arrow[d, dashed] & \mathcal{F}(b_1)(E_0) \arrow[u, "e_1"'] \arrow[d, dashed] \\
B_0 \arrow[r, "b_1"'] & B_1
\end{tikzcd}$$

Fig. 6

We have spelled these things out here in fair detail because in the rest of the paper we will be concerned exclusively with opfibrations $\mathcal{P}: \mathcal{E} \to \mathcal{B}$ which are not split, so we will not be able to find a handy functor $\mathcal{F}: \mathcal{B} \to \mathcal{C}at$ to conveniently visualise the inverse relationship. Nevertheless we will be dealing with subject categories $\mathcal{E}$ which arise most naturally by making what is in effect a nondeterministic brute force analogue of the Grothendieck construction. More specifically we identify a projection, call it $\mathcal{P}_0 : \mathcal{E}_0 \to \mathcal{B}$ say, and prove that all arrows in $\mathcal{E}_0$ are opcartesian. We then typically enrich $\mathcal{P}_0 : \mathcal{E}_0 \to \mathcal{B}$ to a projection $\mathcal{P}: \mathcal{E} \to \mathcal{B}$ by "adjoining in-fibre morphisms", showing that the properties of an opfibration continue to hold — the fibres are evidently $\mathcal{P}^{-1}(b)$ for b an object of $\mathcal{B}$. This amounts to showing that $\mathcal{P}: \mathcal{E} \to \mathcal{B}$ is an opfibration directly from the definition because it has "enough strong opcartesian morphisms" in the terminology of Gray (1966), Grothendieck (1961). This in turn is equivalent to constructing an opfibration via a pseudofunctor $\mathcal{F}_P : \mathcal{B} \to \mathcal{C}at$ which chooses a cleavage that is not necessarily a splitting, by making an arbitrary choice of opcartesian arrow for each (E, b) pair. Our more direct approach avoids the distraction of making such a choice, only for it to be disregarded later.

3 The Opfibration $\langle \mathcal{P} \rangle : \langle \mathcal{G}r^{\uparrow} | \mathcal{S}p\text{-}\mathcal{G}r \rangle \to \langle \mathcal{G}r\text{-}\mathcal{S}p \rangle$ etc.

A typed graph over a (type) graph TG is simply an object of the comma category $(\mathcal{G}r{\downarrow}TG)$, i.e., a graph morphism $G \to TG$. As explained in the introduction, various works (Corradini et al. (1996b), Ribeiro (1996), Heckel et. al (1997)) address, with various techniques, the issue of relating graphs typed over different graphs. By exploiting an opfibrational framework, we propose a solution that aims at the greatest generality. In this section we shall define an opfibration where for each type graph TG we have a fibre including all graphs typed over TG, and where in the base category, morphisms between type graphs TG_0 and TG_2 are semiconcrete spans. The "change of base" operation is performed by a pullback construction, and the opfibrational framework allows us to keep the natural non-determinism of this construction. A number of auxiliary opfibrations will be introduced as well. Since this section sets up technical results needed later, we will avoid refering to "typing" etc., prefering a more neutral terminology.

We start with some lemmas for concrete graphs. The first is partly by way of a warmup exercise, but illustrates vividly how the absence of a fixed choice of pullbacks might not be such a handicap.

Lemma 3.1 Let $A \leftarrow [B] \to C$ be a semiconcrete span. Let $X \to A$ be an arrow of $\mathcal{G}r$; let $Y \to B$, $Y \to X$ be a pullback of $X \to A$, $B \to A$ for some representative B of $[B]$; and let $Y \to C$ be given by $B \to C \circ Y \to B$. Let $X' \to A$ be another arrow of $\mathcal{G}r$, and similarly for $Y' \to B'$, $Y' \to X'$, the pullback of $X' \to A$, $B' \to A$ for some other representative B' of $[B]$ with iso $q : B \to B'$; and let $Y' \to C$ be $B' \to C \circ Y' \to B'$. Let $f : X \to X'$ be an arrow such that $XX'A$ commutes. Then:

(1) There is a unique arrow $g : Y \to Y'$ such that $Y'X'XY$ and $YY'B'B$ commute.

(2) $Y'X'XY$ is a pullback.

(3) If f is monic, then so is g.

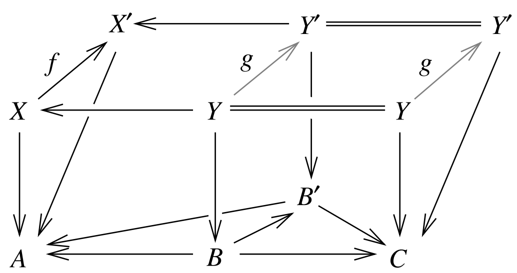

Fig. 7

Proof. See Fig. 7. Let $YXAB$ be a pullback and let $Y'X'AB'$ be the other pullback. Then $Y \to X \to X'$ and $Y \to B \to B'$ close $X' \to A$ and $B' \to A$. So by pullbacks, there is a unique g as required, giving (1). Since $B \to B'$ is an isomorphism, $YXAB'$ is a pullback; whence $Y'X'XY$ is a pullback by pullback properties. And (3) is obvious. ☺

We note that rather importantly, the facts pertaining to the upper layer, namely X, X', Y, Y', are independent of the specific representatives of the lower layer chosen, namely of B, B' in $A \leftarrow [B] \to C$. Moreover in the upper layer, the middle graphs of the concrete spans such as $X \leftarrow Y \to Y$ are fixed automatically once the right hand graph Y is chosen, by the requirement that the right hand arrow is an identity. (Of course the right hand graph Y must be chosen to be isomorphic to the requisite pullback.) The extra flexibility afforded by having distinct representatives B, B' will not be needed below.

Definition 3.2 Let $A \leftarrow [B] \to C$ be a semiconcrete span, and let $X \to A$ be an object of $(\mathcal{G}r{\downarrow}A)$. We write $X \text{ -}{\scriptstyle ABC}\text{-}\!\!\rightarrow Y$ whenever Y is a pullback of $X \to A$ and $B \to A$, i.e. whenever the front face of Fig. 7, relating $X \to A$ from $(\mathcal{G}r{\downarrow}A)$ and $Y \to C$ from $(\mathcal{G}r{\downarrow}C)$ via the span $A \leftarrow [B] \to C$ up to choice B of representative of $[B]$, prevails. We write $X \text{ -}{\scriptstyle ABC}\text{-}\!\!\rightarrow Y \to Y'$ whenever $X \text{ -}{\scriptstyle ABC}\text{-}\!\!\rightarrow Y$ and $Y \to Y'$ is an arrow of $(\mathcal{G}r{\downarrow}C)$, (i.e. there is an arrow $Y' \to C$); which relates $X \to A$ from $(\mathcal{G}r{\downarrow}A)$ and $Y' \to C$ from $(\mathcal{G}r{\downarrow}C)$ via the span $A \leftarrow [B] \to C$ (up to choice B) and arrow $Y \to Y'$. We write $X \text{ -}{\scriptstyle ABC}\text{-}\!\!\rightarrow [Y] \to Y'$, or more briefly $X \text{ -}{\scriptstyle ABC}\text{=}\!\!\Rightarrow Y'$ whenever $X \text{ -}{\scriptstyle ABC}\text{-}\!\!\rightarrow Y \to Y'$ is taken up to choice Y of representative of $[Y]$.

Notational Convention: Readers should note how the strictly speaking incomplete nomenclature used above works. When we write $X \to A$, we typically do not merely mean that there is some arrow from X to A, we intend that the notation $X \to A$ names such an arrow. Similarly $X \text{ -}{\scriptstyle ABC}\text{-}\!\!\rightarrow Y$ not only says that the construction described in definition 3.2 is possible, but implicitly names the arrows $X \to A$, $A \leftarrow B$, $B \to C$, $Y \to C$ on which it depends. We use such a terse convention for the sake of brevity, on the understanding that when a notation like $X \to A$ is used, it will never be necessary to simultaneously consider more than one arrow $X \to A$. Similar remarks apply below, where relatively succinct notations will be needed for constructions that are even more complicated. As a rule, we will mention all the data implicit in a notation the first time it is introduced, and rely on the reader's good sense to fill in the missing details in subsequent use. The next lemma provides a case in point.

Lemma 3.3

(1) Let $X \text{ -}{\scriptstyle ABC}\text{-}\!\!\rightarrow Y$ and $Y \text{ -}{\scriptstyle CDE}\text{-}\!\!\rightarrow Z$. Then $X \text{ -}{\scriptstyle AME}\text{-}\!\!\rightarrow Z$ where $M \to B$ and $M \to D$ is a pullback of $B \to C$ and $D \to C$. This operation is associative.

(2) Let $X \text{ -}{\scriptstyle ABC}\text{-}\!\!\rightarrow Y \to Y'$ and $Y' \text{ -}{\scriptstyle CDE}\text{-}\!\!\rightarrow Z' \to Z''$. Then $X \text{ -}{\scriptstyle AME}\text{-}\!\!\rightarrow Z \to Z''$ where $M \to B$ and $M \to D$ is a pullback of $B \to C$ and $D \to C$, and $Z \to Y$, $Z \to Z'$ is a pullback of $Y \to Y'$ and $Z' \to Y'$. This operation is associative up to choice of Z.

(3) Let $X \text{ -}{\scriptstyle ABC}\text{=}\!\!\Rightarrow Y'$ and $Y' \text{ -}{\scriptstyle CDE}\text{=}\!\!\Rightarrow Z''$ (or more explicitly $X \text{ -}{\scriptstyle ABC}\text{-}\!\!\rightarrow [Y] \to Y'$ and $Y' \text{ -}{\scriptstyle CDE}\text{-}\!\!\rightarrow [Z'] \to Z''$). Then $X \text{ -}{\scriptstyle AME}\text{=}\!\!\Rightarrow Z''$, where M and Z are as in clause (2). This operation is associative.

Proof. Consider the lower part of Fig. 8, i.e. disregard the primed objects Y', Z', Z'' and their incident arrows. Let $M \to B$ and $M \to D$ be a pullback of $B \to C$ and $D \to C$. Since $Z \to Y \to B$ and $Z \to D$ close $B \to C$ and $D \to C$, there is a unique arrow $Z \to M$ because $MBCD$ is a pullback. Now $Z \to M \to D$ factorises $Z \to D$ so $ZYBM$ is a pull-

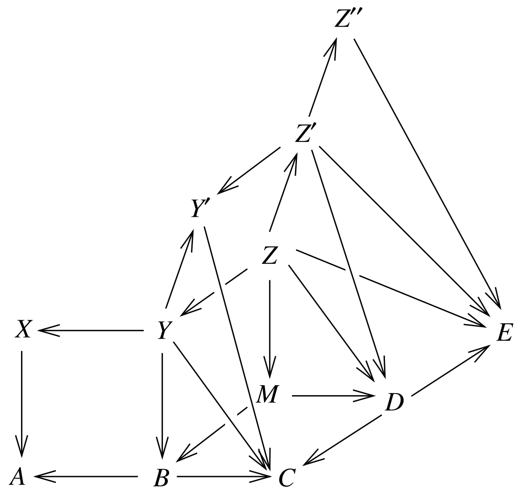

Fig. 8

back by pullback properties, because *ZYCD* and *MBCD* are both pullbacks. Hence *ZXAM*, which is the composition of *YXAB* and *ZYBM* is a pullback. We should check that the construction is independent of the particular representatives B, M, D, of $[B]$, $[M]$, $[D]$ that we choose to work with; but this is made clear by considering other representatives together with the relevant isomorphisms, somewhat as in lemma 3.1. Associativity is now straightforward and we leave it to the reader. This gives the first part.

For the second part consider the upper line of arrows $X \leftarrow Y \to Y' \leftarrow Z' \to Z''$. Let $M \to B$ and $M \to D$ be a pullback of $B \to C$ and $D \to C$ and $Z \to Y, Z \to Z'$ be a pullback of $Y \to Y'$ and $Z' \to Y'$. Composing $YY'Z'Z$ with $CY'Z'D$ shows that *ZYCD* is a pullback. As above there is a unique arrow $Z \to M$ because *MBCD* is a pullback, and again we deduce that *ZYBM* and *ZXAM* are pullbacks. So X -*AME*-› $Z \to Z''$ as claimed, and we can check independence of the specific choice of representatives of $[B]$, $[M]$, $[D]$ as before. This gives the second part aside from associativity. The third part follows once we notice that Z was an arbitrary pullback of $Y \to Y'$ and $Z' \to Y'$, and thus that we can use similar arguments as work for the bottom line, to show that the result is independent of the choice of representatives Y, Z, Z', of $[Y]$, $[Z]$, $[Z']$. Associativity is straightforward. We are done. ☺

Note how the second and third parts in effect exploited lemma 3.1.

Definition 3.4

(1) The category $\mathcal{G}r \,|\, \mathcal{G}r\text{-}\mathcal{S}p$ has as objects $(X \to A)$ (i.e. arrows in $\mathcal{G}r$), and as arrows confections of the form (X -*ABC*-› Y) : $(X \to A) \to (Y \to C)$. Identities are (X -*AAA*-› X) : $(X \to A) \to (X \to A)$, and composition is according to lemma 3.3.(1).

(2) The category $\mathcal{G}r^{\uparrow} \mid \mathcal{G}r\text{-}\mathcal{S}p$ has as objects $(X \rightarrow A)$ (i.e. arrows in $\mathcal{G}r$), and as arrows confections of the form $(X \text{ -}\mathit{\scriptstyle ABC}\text{=> } Y') : (X \rightarrow A) \rightarrow (Y' \rightarrow C)$. Identities are $(X \text{ -}\mathit{\scriptstyle AAA}\text{=> } X) : (X \rightarrow A) \rightarrow (X \rightarrow A)$, and composition is according to lemma 3.3.(3).

By the first and third clauses of lemma 3.3, it is clear that $\mathcal{G}r \mid \mathcal{G}r\text{-}\mathcal{S}p$ and $\mathcal{G}r^{\uparrow} \mid \mathcal{G}r\text{-}\mathcal{S}p$ are both categories. We *don't* have a category corresponding to the second clause, because there is no way of uniquely choosing the constructed object Z (without ad hoc means such as a fixed choice of pullbacks).

Definition 3.5

(1) The category $\langle \mathcal{G}r \mid \mathcal{G}r\text{-}\mathcal{S}p \rangle$ has as objects $\langle X \rightarrow A \rangle$ (i.e. arrows in $\langle \mathcal{G}r \rangle$), and as arrows abstract diagrams of the form of Fig. 9.(a), which we write as $\langle X \text{ -}\mathit{\scriptstyle ABC}\text{-> } Y \rangle : \langle X \rightarrow A \rangle \rightarrow \langle Y \rightarrow C \rangle$. Identities are $\langle X \text{ -}\mathit{\scriptstyle AAA}\text{-> } X \rangle : \langle X \rightarrow A \rangle \rightarrow \langle X \rightarrow A \rangle$, and composition is according to the abstract version of lemma 3.3.(1).

(2) The category $\langle \mathcal{G}r^{\uparrow} \mid \mathcal{G}r\text{-}\mathcal{S}p \rangle$ has as objects $\langle X \rightarrow A \rangle$ (i.e. arrows in $\langle \mathcal{G}r \rangle$), and as arrows abstract diagrams of the form of Fig. 9.(b), which we write as $\langle X \text{ -}\mathit{\scriptstyle ABC}\text{=> } Y' \rangle : \langle X \rightarrow A \rangle \rightarrow \langle Y' \rightarrow C \rangle$ Identities are $\langle X \text{ -}\mathit{\scriptstyle AAA}\text{=> } X \rangle : \langle X \rightarrow A \rangle \rightarrow \langle X \rightarrow A \rangle$, and composition is according to the abstract version of lemma 3.3.(3).

$$
\begin{tikzcd}
\langle X \rangle \arrow[d] & \langle Y \rangle \arrow[l] \arrow[d] \arrow[dr] & \\
\langle A \rangle \arrow[ur, phantom, "\text{(PB)}"] & {[B]} \arrow[l] \arrow[r] & \langle C \rangle
\end{tikzcd}
$$

Fig. 9.(a)

$$
\begin{tikzcd}
\langle X \rangle \arrow[d] & {[Y]} \arrow[l] \arrow[r] \arrow[d] \arrow[dr] & \langle Y' \rangle \arrow[d] \\
\langle A \rangle \arrow[ur, phantom, "\text{(PB)}"] & {[B]} \arrow[l] \arrow[r] & \langle C \rangle
\end{tikzcd}
$$

Fig. 9.(b)

It is not hard to see that $\langle \mathcal{G}r \mid \mathcal{G}r\text{-}\mathcal{S}p \rangle$ and $\langle \mathcal{G}r^{\uparrow} \mid \mathcal{G}r\text{-}\mathcal{S}p \rangle$ are both categories. We note that we have the required flexibility up to arbitrary isomorphisms for the interior objects already by lemma 3.3; the extra flexibility up to standard isomorphisms for the exterior objects can be easily retrofitted by considering other representatives $A^{\sim}$, $C^{\sim}$, $X^{\sim}$, $Y^{\sim}$, $Y'^{\sim}$ of $\langle A\rangle$, $\langle C\rangle$, $\langle X\rangle$, $\langle Y\rangle$, $\langle Y'\rangle$ and the unique standard isomorphisms $A \to A^{\sim}$ etc. We leave the somewhat tedious details to the interested reader.

There is a straightforward embedding $\tau : \langle \mathcal{G}r \,|\, \mathcal{G}r\text{-}\mathcal{S}p\rangle \to \langle \mathcal{G}r^{\uparrow} \,|\, \mathcal{G}r\text{-}\mathcal{S}p\rangle$ since every concrete diagram corresponding to Fig. 9.(a) is a valid concrete diagram corresponding to Fig. 9.(b) such that the arrow in $[Y] \to \langle Y'\rangle$ is an identity. If Y has nontrivial automorphisms, then $[Y] \to \langle Y\rangle$ will contain arrows other than identities-up-to-standard-isomorphism, so when we partition an abstract arrow such as $[Y] \to \langle Y\rangle$ into equivalence classes of arrows-up-to-standard-isomorphism, the embedding becomes faithful but not full. We now come to the main results of this section.

Theorem 3. 6

(1) The projection $\mathcal{P}_{\varnothing} : \mathcal{G}r \,|\, \mathcal{G}r\text{-}\mathcal{S}p \to \mathcal{G}r\text{-}\mathcal{S}p$ that takes (X -ABC-› Y) : $(X \to A) \to (Y \to C)$ to $(A \leftarrow [B] \to C) : A \to C$ is an opfibration, where all arrows of $\mathcal{G}r \,|\, \mathcal{G}r\text{-}\mathcal{S}p$ are opcartesian.

(2) The projection $\mathcal{P} : \mathcal{G}r^{\uparrow} \,|\, \mathcal{G}r\text{-}\mathcal{S}p \to \mathcal{G}r\text{-}\mathcal{S}p$ that takes (X -ABC=› Y') : $(X \to A) \to (Y' \to C)$ to $(A \leftarrow [B] \to C) : A \to C$ is an opfibration, for which arrows (X -ABC-› $[Y] \to Y'$) such that $[Y] \to Y'$ contains only isomorphisms (i.e. Y and Y' are isomorphic) are opcartesian.

Proof. We easily see that both $\mathcal{P}_{\varnothing}$ and $\mathcal{P}$ are functors, so we just need to check the conditions for opcartesian arrows. See Fig. 10.

For the first part, consider Fig. 10.(a). We are given (X -ABC-› Y) : $(X \to A) \to (Y \to C)$ and (X -AME-› Z) : $(X \to A) \to (Z \to E)$ such that M is a pullback of $B \to C$ and $D \to C$ (i.e. the objects and solid arrows), and we must construct a factorisation (X -AME-› Z) = (Y -CDE-› Z) $\circ$ (X -ABC-› Y) which is unique. Now $Z \to X$ and $Z \to M \to B$ close $X \to A$ and $B \to A$, so there is a unique arrow $Z \to Y$ by pullback properties. Now $ZXAM$ and $YXAB$ are pullbacks, so $ZYBM$ is a pullback too. We compose $ZYBM$ with $MBCD$, another pullback, to deduce that $ZYCD$ is a pullback. This gives a (Y -CDE-› Z) and we just have to show uniqueness. Firstly $Z \to Y$ is unique by construction; secondly if there were an alternative arrow $Z \to D$, then some item of Z could be sent to two different images in D, these would have different preimages under $M \to D$, and the triangle ZMD would fail to commute.

For the second part, consider Fig. 10.(b). We are given (X -ABC-› $[Y] \to Y'$) : $(X \to A) \to (Y' \to C)$ where $Y \to Y'$ is an iso, and (X -AME-› $[Z] \to Z''$) : $(X \to A) \to (Z'' \to E)$ such that M is a pullback of $B \to C$ and $D \to C$ (i.e. the objects and solid arrows), and we must construct a factorisation (X -AME-› $[Z] \to Z''$) = (Y' -CDE-› $[Z'] \to Z''$) $\circ$ (X -ABC-› $[Y] \to Y'$) which is unique. Just as in the first part, we can argue that there is a pullback $ZYCD$ via arrows $Z \to Y$ and $Z \to D$ (not shown in Fig. 10.(b)). Now we can form the pullback of $Y' \to C$, $D \to C$ getting some $Z' \to Y'$, $Z' \to D$, and since $Y \to Y'$ is an iso, there is a unique iso Z ▬ Z' that makes the cube $ZYY'Z'MBCD$ commute. Now Z' ▬$^{-1}$$Z \to Z''$ gives us the remaining arrow in (Y' -CDE-› $[Z'] \to Z''$). We must show uniqueness. The object Z' is only needed up to isomorphism, so we merely need to check the arrows. Now the arrows $Z' \to Y'$, $Z' \to D$ are fixed by the requirement that

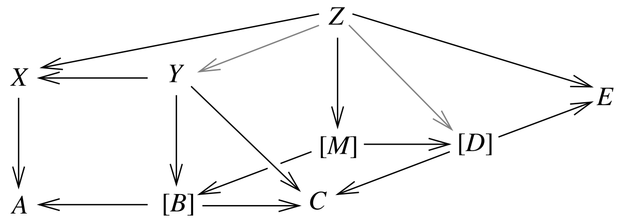

Fig. 10.(a)

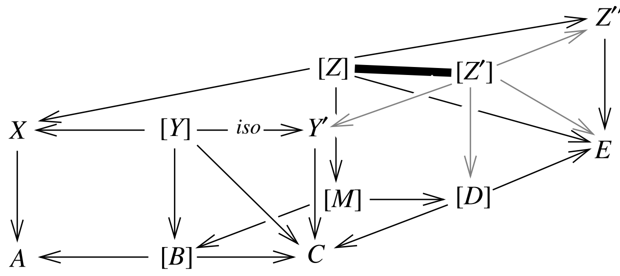

Fig. 10.(b)

the cube $ZYY'Z'MBCD$ commutes; while the arrows, $Z' \to Z''$, $Z' \to E$ are fixed by the requirement that the triangles $ZZ'Z''$, $ZZ'E$ commute (because ▬ is an iso). We are done. ☺

The above immediately generalises by the usual technique of making the exterior objects defined only up to standard isomorphism, so that:

Theorem 3. 7

(1) The projection $\langle \mathcal{P}_{\varnothing}\rangle : \langle \mathcal{G}r \,|\, \mathcal{G}r\text{-}\mathcal{S}p\rangle \to \langle \mathcal{G}r\text{-}\mathcal{S}p\rangle$ that takes $\langle X \text{ -}{\scriptstyle ABC}\text{-> } Y\rangle : \langle X \to A\rangle \to \langle Y \to C\rangle$ to $\langle A\rangle \leftarrow [B] \to \langle C\rangle : \langle A\rangle \to \langle C\rangle$ is an opfibration, where all arrows of $\langle \mathcal{G}r \,|\, \mathcal{G}r\text{-}\mathcal{S}p\rangle$ are opcartesian.

(2) The projection $\langle \mathcal{P}\rangle : \langle \mathcal{G}r^{\uparrow} \,|\, \mathcal{G}r\text{-}\mathcal{S}p\rangle \to \langle \mathcal{G}r\text{-}\mathcal{S}p\rangle$ that takes $\langle X \text{ -}{\scriptstyle ABC}\text{=> } Y'\rangle : \langle X \to A\rangle \to \langle Y' \to C\rangle$ to $\langle A\rangle \leftarrow [B] \to \langle C\rangle : \langle A\rangle \to \langle C\rangle$ is an opfibration, for which arrows $\langle X \text{ -}{\scriptstyle ABC}\text{-> } [Y] \to Y'\rangle$ such that $[Y] \to \langle Y'\rangle$ contains only isomorphisms (i.e. Y and Y' are isomorphic) are opcartesian.

We see that in both theorems 3.6 and 3.7 above the first category is effectively the opcartesian subcategory of the second. So we have created the second opfibration by identifying first the opcartesian arrows, and then enhancing this to include further in-fibre arrows, showing that this preserved the opcartesian properties. This is what we meant by our nondeterministic analogue of the Grothendieck construction. Note how we avoided making what for us would have been a spurious choice of cleavage in order to build a pseudofunctor $\mathcal{G}r\text{-}\mathcal{S}p \to \mathcal{C}at$ even though the end results would have been equivalent. Such a choice of cleavage would have amounted to some choice of pullbacks. Demanding that such a cleavage was a splitting, yielding a genuine functor $\mathcal{G}r\text{-}\mathcal{S}p \to \mathcal{C}at$, would have forced the choice of pullbacks to satisfy the conditions demanded in eg. Corradini et al. (1996b).[1]

This two stage approach to constructing an opfibration will also occur below. Where appropriate we highlight the two cases separately because in the main, results about rewriting that we are interested in will centre on the opcartesian subcategories while the larger categories will frequently provide vital technical results for reasoning about diagrams. Here is a specific example.

By the fact that $\mathcal{G}r^{\uparrow} \mid \mathcal{G}r\text{-}\mathcal{S}p$ includes both the arrows of the comma categories $(\mathcal{G}r{\downarrow}A)$, and arrows that pertain to the change of base $A \leftarrow [B] \to C$, we recover good behaviour for all diagrams *Diag* over a base graph A, i.e. diagrams into $(\mathcal{G}r{\downarrow}A)$ which we write as $(Diag{\downarrow}A)$, under change of base $A \leftarrow [B] \to C$. This works as follows.

Normally a diagram is defined as a graph morphism $\gamma : \mu \to U\mathcal{C}$ where μ is a graph, the shape graph, and $U\mathcal{C}$ is the underlying graph of a category $\mathcal{C}$. A diagram morphism is then a natural transformation $n : \mu \to \delta$ in $[\mu, U\mathcal{C}]$ where we have used functor category notation for the category of graph morphisms from μ to $U\mathcal{C}$. Thus diagram morphisms map the arrows of $\gamma(\mu)$ to arrows of $\delta(\mu)$. A special case of this arises when we have an opfibration $\mathcal{P} : \mathcal{E} \to \mathcal{B}$. Then, fixing an arrow $b : B_0 \to B_1$ in the base, a diagram $\gamma : \mu \to U(\mathcal{P}^{-1}(B_0))$ into the fibre over B_0, can map to a diagram $\delta : \mu \to U(\mathcal{P}^{-1}(B_1))$ into the fibre over B_1, by choosing an arrow $\gamma(m_0) \to E_1$ (opcartesian for $(\gamma(m_0), b)$) for each vertex m_0 in μ, and using the unique factorisation properties of opfibrations to fill in the required arrows $E_1 \to E'_1$ for all arrows in $\mathcal{P}^{-1}(B_0)$ which are γ images of edges $m_0 \to m'_0$ in μ. An extra special case arises when we choose the opcartesian arrows in bijection with the $\gamma(m_0)$ rather than in surjection with the m_0. Then we are in effect dealing with diagrams $\gamma(\mu)$ of shape $\gamma(\mu)$, and the codomain of such a diagram morphism will also be of shape $\gamma(\mu)$. (It may well be of smaller shape too.) In such a case we can essentially drop explicit mention of the shape in talking about diagram morphisms. It is this extra special case that we will need to apply in the rest of the paper where the intended shape will be evident from the context.

The preceding considerations motivate the following somewhat nonstandard definitions; readers should bear in mind that nothing whatsoever would block more general and more conventionally based definitions of $\mathcal{D}iag \mid \mathcal{G}r\text{-}\mathcal{S}p$ etc.

Definition 3.8 The category $\mathcal{D}iag \mid \mathcal{G}r\text{-}\mathcal{S}p$ has as objects concrete diagrams over a graph (i.e. objects of $(Diag{\downarrow}A)$, of shape *Diag*, for A in $\mathcal{G}r$), and as arrows semiconcrete

1. Note incidentally that any choice of pullbacks made for this purpose is independent of any choice of pullbacks made for the purpose of giving a deterministic horizontal composition for concrete spans, because one can independently wiggle the four graphs at the corners of a pullback square up to isomorphism without destroying the pullback properties.

change of base diagram morphisms $(Diag_0 \text{ -}ABC\text{-> } Diag_2) : (Diag_0 \to A) \to (Diag_2 \to C)$, which are collections of $\mathcal{Gr} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp}$ arrows in bijection with the objects of $(Diag_0 \to A)$ as follows. For each object $(X_0 \to A)$ in $(Diag_0 \to A)$, there is an object $(X_2 \to C)$ in $(Diag_2 \to C)$ and a $\mathcal{Gr} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp}$ arrow $(X_0 \text{ -}ABC\text{-> } X_2) : (X_0 \to A) \to (X_2 \to C)$ in the collection, such that for each arrow $(X_0 \to X'_0) : (X_0 \to A) \to (X'_0 \to A)$ in $(Diag_0 \to A)$, there is a unique arrow $(X_2 \to X'_2) : (X_2 \to C) \to (X'_2 \to C)$ in $(Diag_2 \to C)$, obtained by applying theorem 3.6.(2) to $(X_0 \text{ -}ABC\text{-> } X_2)$ and $(X'_0 \text{ -}ABC\text{-> } X'_2) \circ (X_0 \to X'_0)$. The action on objects and arrows of $(Diag_0 \to A)$ must be onto the objects and arrows of $(Diag_2 \to C)$, and $Diag_2$ is perforce of shape $Diag_0$ as well as of shape $Diag_2$.

Definition 3.9 The category $\mathcal{Diag}^{\uparrow} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp}$ has as objects concrete diagrams over a graph (i.e. objects of $(Diag{\downarrow}A)$, of shape $Diag$, for A in $\mathcal{Gr}$), and as arrows semiconcrete change of base plus in-fibre diagram morphisms $(Diag_0 \text{ -}ABC\text{-> } [Diag_1] \to Diag_2) : (Diag_0 \to A) \to (Diag_2 \to C)$, which are collections of $\mathcal{Gr} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp}$ arrows in bijection with the objects of $(Diag_0 \to A)$ as follows. For each object $(X_0 \to A)$ in $(Diag_0 \to A)$, there is an object $(X_2 \to C)$ in $(Diag_2 \to C)$ and a $\mathcal{Gr} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp}$ arrow $(X_0 \text{ -}ABC\text{-> } [X_1] \to X_2) : (X_0 \to A) \to (X_2 \to C)$ in the collection, such that for each arrow $(X_0 \to X'_0) : (X_0 \to A) \to (X'_0 \to A)$ in $(Diag_0 \to A)$, there is a unique arrow $(X_2 \to X'_2) : (X_2 \to C) \to (X'_2 \to C)$ in $(Diag_2 \to C)$ which makes all required compositions commute. The action on objects and arrows of $(Diag_0 \to A)$ must be onto the objects and arrows of $(Diag_2 \to C)$, and $\langle Diag_2 \rangle$ is perforce of shape $\langle Diag_0 \rangle$ as well as of shape $\langle Diag_2 \rangle$.

The abstract definitions are:

Definition 3.10 The category $\langle \mathcal{Diag} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ has as objects abstract diagrams over an abstract graph (i.e. objects of $(\langle Diag \rangle{\downarrow}\langle A \rangle)$, of shape $\langle Diag \rangle$, for $\langle A \rangle$ in $\langle \mathcal{Gr} \rangle$), and as arrows abstract change of base diagram morphisms $\langle Diag_0 \text{ -}ABC\text{-> } Diag_2 \rangle : \langle Diag_0 \to A \rangle \to \langle Diag_2 \to C \rangle$, which are collections of $\langle \mathcal{Gr} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ arrows in bijection with the objects of $\langle Diag_0 \to A \rangle$ as follows. For each object $\langle X_0 \to A \rangle$ in $\langle Diag_0 \to A \rangle$, there is an object $\langle X_2 \to C \rangle$ in $\langle Diag_2 \to C \rangle$ and a $\langle \mathcal{Gr} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ arrow $\langle X_0 \text{ -}ABC\text{-> } X_2 \rangle : \langle X_0 \to A \rangle \to \langle X_2 \to C \rangle$ in the collection, such that for each arrow $\langle X_0 \to X'_0 \rangle : \langle X_0 \to A \rangle \to \langle X'_0 \to A \rangle$ in $\langle Diag_0 \to A \rangle$, there is a unique arrow $\langle X_2 \to X'_2 \rangle : \langle X_2 \to C \rangle \to \langle X'_2 \to C \rangle$ in $\langle Diag_2 \to C \rangle$, obtained by applying theorem 3.7.(2) to $\langle X_0 \text{ -}ABC\text{-> } X_2 \rangle$ and $\langle X'_0 \text{ -}ABC\text{-> } X'_2 \rangle \circ \langle X_0 \to X'_0 \rangle$. The action on objects and arrows of $\langle Diag_0 \to A \rangle$ must be onto the objects and arrows of $\langle Diag_2 \to C \rangle$, and $Diag_2$ is perforce of shape $Diag_0$ as well as of shape $Diag_2$.

Definition 3.11 The category $\langle \mathcal{Diag}^{\uparrow} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ has as objects abstract diagrams over an abstract graph (i.e. objects of $(\langle Diag \rangle{\downarrow}\langle A \rangle)$, of shape $\langle Diag \rangle$, for $\langle A \rangle$ in $\langle \mathcal{Gr} \rangle$), and as arrows abstract change of base plus in-fibre abstract diagram morphisms $\langle Diag_0 \text{ -}ABC\text{-> } [Diag_1] \to Diag_2 \rangle : \langle Diag_0 \to A \rangle \to \langle Diag_2 \to C \rangle$, which are collections of $\langle \mathcal{Gr} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ arrows in bijection with the objects of $\langle Diag_0 \to A \rangle$ as follows. For each object $\langle X_0 \to A \rangle$ in $\langle Diag_0 \to A \rangle$, there is an object $\langle X_2 \to C \rangle$ in $\langle Diag_2 \to C \rangle$ and a $\langle \mathcal{Gr} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ arrow $\langle X_0 \text{ -}ABC\text{-> } [X_1] \to X_2 \rangle : \langle X_0 \to A \rangle \to \langle X_2 \to C \rangle$ in the collection, such that for each arrow $\langle X_0 \to X'_0 \rangle : \langle X_0 \to A \rangle \to \langle X'_0 \to A \rangle$ in $\langle Diag_0 \to A \rangle$, there is a unique arrow $\langle X_2 \to X'_2 \rangle : \langle X_2 \to C \rangle \to \langle X'_2 \to C \rangle$ in $\langle Diag_2 \to C \rangle$ which makes all required compositions commute. The action on objects and arrows of $\langle Diag_0 \to A \rangle$ must be onto the objects and arrows of $\langle Diag_2 \to C \rangle$, and $\langle Diag_2 \rangle$ is perforce of shape $\langle Diag_0 \rangle$ as well as of shape $\langle Diag_2 \rangle$.

Furthermore we have the following:

Theorem 3.12

(1) The projection $\mathcal{P}_{\mathcal{Diag}\varnothing} : \mathcal{Diag} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \to \mathcal{Gr}\text{-}\mathcal{Sp}$ that takes $(Diag_0 \text{ -}ABC\text{-}\rangle\ Diag_2) : (Diag_0 \to A) \to (Diag_2 \to C)$ to $(A \leftarrow [B] \to C) : A \to C$ is an opfibration, where all arrows of $\mathcal{Diag} \mid \mathcal{Gr}\text{-}\mathcal{Sp}$ are opcartesian.

(2) The projection $\mathcal{P}_{\mathcal{Diag}} : \mathcal{Diag}^{\uparrow} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \to \mathcal{Gr}\text{-}\mathcal{Sp}$ that takes $(Diag_0 \text{ -}ABC\text{-}\rangle\ [Diag_1] \to Diag_2) : (Diag_0 \to A) \to (Diag_2 \to C)$ to $(A \leftarrow [B] \to C) : A \to C$ is an opfibration, for which arrows $(Diag_0 \text{ -}ABC\text{-}\rangle\ [Diag_1] \to Diag_2)$ such that $[Diag_1] \to Diag_2$ contains only isomorphisms (i.e. $Diag_1$ and $Diag_2$ are isomorphic diagrams) are opcartesian.

Abstractly:

Theorem 3.13

(1) The projection $\langle \mathcal{P}_{\mathcal{Diag}\varnothing} \rangle : \langle \mathcal{Diag} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle \to \langle \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ that takes $\langle Diag_0 \text{ -}ABC\text{-}\rangle\ Diag_2 \rangle : \langle Diag_0 \to A \rangle \to \langle Diag_2 \to C \rangle$ to $\langle A \rangle \leftarrow [B] \to \langle C \rangle : \langle A \rangle \to \langle C \rangle$ is an opfibration, where all arrows of $\langle \mathcal{Diag} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ are opcartesian.

(2) The projection $\langle \mathcal{P}_{\mathcal{Diag}} \rangle : \langle \mathcal{Diag}^{\uparrow} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle \to \langle \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ that takes $\langle Diag_0 \text{ -}ABC\text{-}\rangle\ [Diag_1] \to Diag_2 \rangle : \langle Diag_0 \to A \rangle \to \langle Diag_2 \to C \rangle$ to $\langle A \rangle \leftarrow [B] \to \langle C \rangle : \langle A \rangle \to \langle C \rangle$ is an opfibration, for which arrows $\langle Diag_0 \text{ -}ABC\text{-}\rangle\ [Diag_1] \to Diag_2 \rangle$ such that $[Diag_1] \to \langle Diag_2 \rangle$ contains only isomorphisms (i.e. $\langle Diag_1 \rangle$ and $\langle Diag_2 \rangle$ are isomorphic abstract diagrams) are opcartesian.

There are also extensions of the above for diagrams with some parts concrete, other parts abstract (up to standard or arbitrary isomorphism, as needed), rather as in the detailed discussion of spans in section 2.1. Where required, we will refer to such generalisations using the generic notation *generalised-*$\mathcal{Diag}$ etc. In the sections below, we will call upon the above results in reasoning about diagrams in categories that represent graph rewriting phenomena.

4 The Category of Typed Graph Grammars

For the purposes of typed graph rewriting theory, we want to consider graphs G as being typed over a type graph TG, so they will come as morphisms $G \to TG$, i.e. as elements of $(\mathcal{Gr}{\downarrow}TG)$, (or of $(\langle \mathcal{Gr} \rangle{\downarrow}\langle TG \rangle)$ taking a more abstract view). Changing the base type graph is done by means of an arbitrary semiconcrete or abstract span, eg. abstractly, $(\langle TG_0 \rangle \leftarrow [TG_1] \to \langle TG_2 \rangle) : \langle TG_0 \rangle \to \langle TG_2 \rangle$. Therefore the results of the previous section show how individual typed graphs or their abstract versions, transform under such a change of typing. Note the element of nondeterminism present in this process. For concrete graphs this is to be expected, as there is a proper class of distinct graphs of each particular isomorphism type. For abstract graphs the nondeterminism does not disappear completely, as nontrivial automorphisms of graphs still remain to potentially confound arrow composition, even when one wishes to abstract away isomorphisms. Thus our opfibrations are not split. Now we turn our attention to graph grammars.

Definition 4.1 The category of concrete typed graph grammars $\mathcal{GraGra}$ has the following constituents.

Objects: $(TG, \overline{G}, P, \pi)$ where:
TG is a concrete type graph, i.e. an object of $\mathcal{Gr}$,
$\overline{G}$ is a concrete start graph typed over TG,
i.e. an object of $(\mathcal{Gr}{\downarrow}TG)$,
P is a class[1] of production names,
$\pi : P \to |\ \mathcal{Sp}{\downarrow}TG\ |$ is a map from P to concrete typed spans,
i.e. objects of $(\mathcal{Sp}{\downarrow}TG)$.

Arrows: $(TG_0 \leftarrow [TG_1] \to TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0) \to (TG_2, \overline{G}_2, P_2, \pi_2)$ which is shorthand for a collection of arrows.
Firstly: an arrow of $\mathcal{Gr}\text{-}\mathcal{Sp}$, $(TG_0 \leftarrow [TG_1] \to TG_2) : TG_0 \to TG_2$,
Secondly: an arrow of $\mathcal{Gr} \mid \mathcal{Gr}\text{-}\mathcal{Sp}$,
$(\overline{G}_0 \text{ -}_{TG_0TG_1TG_2}\text{-}\!\!> \overline{G}_2) : (\overline{G}_0 \to TG_0) \to (\overline{G}_2 \to TG_2)$,
Thirdly: an arrow of $\mathcal{Class}$, $f : P_0 \to P_2$ i.e. a map,
Fourthly: for all $p \in P_0$ an arrow of $\mathcal{Diag} \mid \mathcal{Gr}\text{-}\mathcal{Sp}$,
$(\pi_0(p) \text{ -}_{TG_0TG_1TG_2}\text{-}\!\!> \pi_2(f(p)))) : (\pi_0(p) \to TG_0) \to (\pi_2(f(p)) \to TG_2)$
where the action of $\text{-}_{TG_0TG_1TG_2}\text{-}\!\!>$ has been specialised to spans.

Composition: $(TG_2 \leftarrow [TG_3] \to TG_4, g) \circ (TG_0 \leftarrow [TG_1] \to TG_2, f) =$
$(TG_0 \leftarrow [TG_{2'}] \to TG_4, g \circ f)$ where $TG_{2'}$ is a pullback of
$TG_1 \to TG_2$ and $TG_2 \leftarrow TG_3$.

Identities: $(TG \leftarrow [TG] \to TG, \mathrm{id}_P) : (TG, \overline{G}, P, \pi) \to (TG, \overline{G}, P, \pi)$.

The abstract version of this just makes the graph parts abstract, with obvious identities and composition.

Definition 4.2 The category of abstract typed graph grammars $\langle\mathcal{GraGra}\rangle$ has the following constituents.

Objects: $(\langle TG\rangle, \langle\overline{G}\rangle, P, \pi)$ where:
$\langle TG\rangle$ is an abstract type graph, i.e. an object of $\langle\mathcal{Gr}\rangle$,
$\langle\overline{G}\rangle$ is an abstract start graph typed over $\langle TG\rangle$,
i.e. an object of $(\langle\mathcal{Gr}\rangle{\downarrow}\langle TG\rangle)$,
P is a class of production names,
$\pi : P \to |\ \langle\mathcal{Sp}\rangle{\downarrow}\langle TG\rangle\ |$ is a map from P to special abstract typed spans, i.e. objects of $(\langle\mathcal{Sp}\rangle{\downarrow}\langle TG\rangle)$.

Arrows: $(\langle TG_0\rangle \leftarrow [TG_1] \to \langle TG_2\rangle, f) : (\langle TG_0\rangle, \langle\overline{G}_0\rangle, P_0, \pi_0) \to (\langle TG_2\rangle, \langle\overline{G}_2\rangle, P_2, \pi_2)$
which is shorthand for a collection of arrows.

---

1. We will see in the next section why we need a class rather than a set here. In principle this leads to more foundational problems.

Firstly: an arrow of $\langle \mathcal{Gr\text{-}Sp}\rangle$, $\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle : \langle TG_0\rangle \rightarrow \langle TG_2\rangle$,
Secondly: an arrow of $\langle \mathcal{Gr} \,|\, \mathcal{Gr\text{-}Sp}\rangle$,
$\langle \overline{G}_0 \text{ -}_{TG_0TG_1TG_2}\text{-›} \overline{G}_2\rangle : \langle \overline{G}_0 \rightarrow TG_0\rangle \rightarrow \langle \overline{G}_2 \rightarrow TG_2\rangle$,
Thirdly: an arrow of $\mathcal{Class}$, $f : P_0 \rightarrow P_2$ i.e. a map,
Fourthly: for all $p \in P_0$ an arrow of $\langle \mathcal{Diag} \,|\, \mathcal{Gr\text{-}Sp}\rangle$,
$(\pi_0(p) \text{ -}_{TG_0TG_1TG_2}\text{-› } \pi_2(f(p))) : \langle \pi_0(p) \rightarrow TG_0\rangle \rightarrow \langle \pi_2(f(p)) \rightarrow TG_2\rangle$
where the action of $\text{-}_{TG_0TG_1TG_2}\text{-›}$ has been specialised to special abstract spans.

Theorem 4.3 The projection $\mathcal{P}_{\mathcal{GraGra}} : \mathcal{GraGra} \rightarrow \mathcal{Gr\text{-}Sp}$ such that

$$\mathcal{P}_{\mathcal{GraGra}}((TG, \overline{G}, P, \pi)) = TG \quad \text{and}$$
$$\mathcal{P}_{\mathcal{GraGra}}((TG_0 \leftarrow [TG_1] \rightarrow TG_2, f)) = (TG_0 \leftarrow [TG_1] \rightarrow TG_2)$$

is an opfibration, in which the arrows $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2)$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

Proof. Let $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, iso) : GG_0 \rightarrow GG_2$ be an opcartesian arrow, and suppose we have an arrow $(TG_0 \leftarrow [TG_{2'}] \rightarrow TG_4, g) : GG_0 \rightarrow GG_4$ whose projection is an extension of $(TG_0 \leftarrow [TG_1] \rightarrow TG_2)$ by $(TG_2 \leftarrow [TG_3] \rightarrow TG_4)$. We have to find a unique $(TG_2 \leftarrow [TG_3] \rightarrow TG_4, g') : GG_2 \rightarrow GG_4$ such that $GG_0 \rightarrow GG_2 \rightarrow GG_4$ factors $GG_0 \rightarrow GG_4$.

Luckily all of the hard work has already been done in section 3. Firstly, the component $(TG_2 \leftarrow [TG_3] \rightarrow TG_4)$ is immediate. Secondly, the component $(TG_2 \leftarrow [TG_3] \rightarrow TG_4) : (\overline{G}_2 \rightarrow TG_2) \rightarrow (\overline{G}_4 \rightarrow TG_4)$ arises uniquely since $\mathcal{P}_{\varnothing} : \mathcal{Gr} \,|\, \mathcal{Gr\text{-}Sp} \rightarrow \mathcal{Gr\text{-}Sp}$ is an opfibration and its opcartesian arrows agree with the start graph component of opcartesian arrows of $\mathcal{GraGra}$. Thirdly, the fact that $iso : P_0 \rightarrow P_2$ is an isomorphism means that the requirement $g' \circ iso = g$ fixes g' uniquely. Fourthly, for each $p \in P_2$, we use theorem 3.12.(1) specialised to concrete spans to find a unique $(\pi_2(p) \text{ -}_{TG_2TG_3TG_4}\text{-› } \pi_4(g'(p))) : \langle \pi_2(p) \rightarrow TG_2\rangle \rightarrow \langle \pi_4(f(p)) \rightarrow TG_4\rangle$. ☺

We immediately deduce:

Theorem 4.4 The projection $\langle \mathcal{P}_{\mathcal{GraGra}}\rangle : \langle \mathcal{GraGra}\rangle \rightarrow \langle \mathcal{Gr\text{-}Sp}\rangle$ such that

$$\langle \mathcal{P}_{\mathcal{GraGra}}\rangle((\langle TG\rangle, \langle \overline{G}\rangle, P, \pi)) = \langle TG\rangle \quad \text{and}$$
$$\langle \mathcal{P}_{\mathcal{GraGra}}\rangle((\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f)) = \langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle$$

is an opfibration, in which the arrows $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f) : (\langle TG_0\rangle, \langle \overline{G}_0\rangle, P_0, \pi_0) \rightarrow (\langle TG_2\rangle, \langle \overline{G}_2\rangle, P_2, \pi_2)$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

Direct derivations behave well too.

Theorem 4.5 The category $\mathcal{Diag} \,|\, \mathcal{Gr\text{-}Sp}$ has the subcategory $\mathcal{Sp}$-DPB $|\, \mathcal{Gr\text{-}Sp}$ which describes how DPB direct derivation step diagrams are preserved under change of type. Also $\mathcal{Diag} \,|\, \mathcal{Gr\text{-}Sp}$ has the subcategory $\mathcal{Sp}$-DPO $|\, \mathcal{Gr\text{-}Sp}$ which describes how DPO direct derivation step diagrams are preserved under change of type.

Proof. We just have to check a couple of extra conditions. For DPB rewriting, consider a pullback diagram over TG_0, $WXYZ \rightarrow TG_0$ say, and its fate under a change of base arrow $TG_0 \leftarrow [TG_1] \rightarrow TG_2$. The crux of the construction is a pullback of the pullback square, i.e. a limit in $\mathcal{Gr}$ of three cofinal arrows. This gives a pullback cube. So the result of the change of base is a pullback diagram, and by gluing two such cubes together we change the base of a DPB rewrite. Since such changes compose well we obtain DPB-$\mathcal{Sp} \mid \mathcal{Sp}$.

For DPO rewriting, a similar thing is obtained by appeal to the cube lemma A2.(i) in the appendix of Corradini et al. (1996b). ☺

Evidently we have:

Theorem 4.6 The category $\langle \mathcal{Diag} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ has the subcategory $\langle \mathcal{Sp}\text{-DPB} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ which describes how DPB direct derivation step diagrams (with interface graphs in both productions and derivation steps defined up to standard isomorphism) are preserved under change of type. Also $\langle \mathcal{Diag} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ has the subcategory $\langle \mathcal{Sp}\text{-DPO} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ which describes how DPO direct derivation step diagrams (with interface graphs in both productions and derivation steps defined up to standard isomorphism) are preserved under change of type.

And of course:

Theorem 4.7 The obvious projection $\mathcal{P}_{\mathcal{Sp}\text{-DPB} \mid \mathcal{Gr}\text{-}\mathcal{Sp}\varnothing} : \mathcal{Sp}\text{-DPB} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \to \mathcal{Gr}\text{-}\mathcal{Sp}$ is an opfibration where all arrows of $\mathcal{Sp}\text{-DPB} \mid \mathcal{Gr}\text{-}\mathcal{Sp}$ are opcartesian.

Theorem 4.8 The obvious projection $\mathcal{P}_{\mathcal{Sp}\text{-DPO} \mid \mathcal{Gr}\text{-}\mathcal{Sp}\varnothing} : \mathcal{Sp}\text{-DPO} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \to \mathcal{Gr}\text{-}\mathcal{Sp}$ is an opfibration where all arrows of $\mathcal{Sp}\text{-DPO} \mid \mathcal{Gr}\text{-}\mathcal{Sp}$ are opcartesian.

Theorem 4.9 The obvious projection $\langle \mathcal{P}_{\mathcal{Sp}\text{-DPB} \mid \mathcal{Gr}\text{-}\mathcal{Sp}\varnothing} \rangle : \langle \mathcal{Sp}\text{-DPB} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle \to \langle \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ is an opfibration, where all arrows of $\langle \mathcal{Sp}\text{-DPB} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ are opcartesian.

Theorem 4.10 The obvious projection $\langle \mathcal{P}_{\mathcal{Sp}\text{-DPB} \mid \mathcal{Gr}\text{-}\mathcal{Sp}\varnothing} \rangle : \langle \mathcal{Sp}\text{-DPO} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle \to \langle \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ is an opfibration, where all arrows of $\langle \mathcal{Sp}\text{-DPO} \mid \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ are opcartesian.

5 The Category of Transition Systems

In this section and in much of what follows the start graph $\overline{G}$ of a graph grammar will just be a passenger; we quietly carry it around to save having to have a fresh bunch of definitions for everything. Also we rely purely on the fact that span morphisms compose. So the reasoning applies equally to DPB and DPO rewriting, and we will refer to both simultaneously by writing DP* (and as a matter of general policy in the sequel we will use * to refer to both DPB and DPO versions of anything, where there is a logical distinction to be made, but no technical necessity for two separate definitions).

Graph transition systems are enriched graph grammars which include all the result spans of direct derivation steps by their productions and such that the class of production names supports a partial action / by $Arr(\mathcal{Sp}\text{-DP}*)$ or its abstract counterpart, where $Arr(\mathcal{Z})$ is the class of arrows of $\mathcal{Z}$.

Notational Remark. From now on readers should assume that everything mentioned in this paper is well typed. Eg. the span morphisms that appear in the next definitions are to be understood as well typed. We suppress the extra information for brevity.

Definition 5.1 A concrete graph transition system is a quintuple $(TG, \overline{G}, P, \pi, /)$ where $(TG, \overline{G}, P, \pi)$ is a concrete graph grammar, and $/ : P \times Arr(\mathcal{Sp}\text{-DP}*) \nrightarrow P$ satisfies

(1) If $\mathrm{dom}((d_1, d_2, d_3)) = \pi(p)$ then $p/(d_1, d_2, d_3)$ is defined and $\pi(p/(d_1, d_2, d_3)) = \mathrm{cod}((d_1, d_2, d_3))$

(2) $p/\mathrm{id}_{\pi(p)} = p$

(3) $(p/(d_1, d_2, d_3))/(d'_1, d'_2, d'_3) = p/(d'_1 \circ d_1, d'_2 \circ d_2, d'_3 \circ d_3)$

The above definition makes plain why we need classes of production names rather than sets in the definition of a graph grammar. Since $Arr(\mathcal{Sp}\text{-DP*})$ is a proper class, P will also in general turn out to be a proper class. We do this to keep in line with previous work. However note that there is the additional possibility of allowing $p/(d_1, d_2, d_3)$ to be defined only for *some* pairs p and (d_1, d_2, d_3) permitted by definition 5.1.(1), and if we do this in such a manner that the LHS of 5.1.(3) is defined iff the RHS is defined, we get a perfectly reasonable definition. We can call such graph transition systems *partial*. Note that by a suitably stingy choice of defined $p/(d_1, d_2, d_3)$ in a partial graph transition system, we can prevent the cardinality explosion of a true graph transition system (though a partial graph transition system is by no means obliged to have a small class of production names).

Definition 5.2 An abstract graph transition system is a quintuple $(\langle TG\rangle, \langle\overline{G}\rangle, P, \pi, /)$ where $(\langle TG\rangle, \langle\overline{G}\rangle, P, \pi)$ is an abstract graph grammar, and $/ : P \times Arr(\langle\mathcal{Sp}\text{-DP*}\rangle) \nrightarrow P$ satisfies

(1) If $\mathrm{dom}((\langle d_1\rangle, \langle d_2\rangle, \langle d_3\rangle)) = \pi(p)$ then $p/(\langle d_1\rangle, \langle d_2\rangle, \langle d_3\rangle)$ is defined and $\pi(p/(\langle d_1\rangle, \langle d_2\rangle, \langle d_3\rangle)) = \mathrm{cod}((\langle d_1\rangle, \langle d_2\rangle, \langle d_3\rangle))$

(2) $p/\mathrm{id}_{\pi(p)} = p$

(3) $(p/(\langle d_1\rangle, \langle d_2\rangle, \langle d_3\rangle))/(\langle d'_1\rangle, \langle d'_2\rangle, \langle d'_3\rangle) = p/(\langle d'_1 \circ d_1\rangle, \langle d'_2 \circ d_2\rangle, \langle d'_3 \circ d_3\rangle)$

Definition 5.3 The category $\mathcal{GraTS}$* of concrete graph transition systems has as objects concrete graph transition systems, and as morphisms $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0, /_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2, /_2)$, where $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f)$ is a morphism of the underlying concrete graph grammar, such that for each p_0 in P_0 and (d_{01}, d_{02}, d_{03}) with $p_0/(d_{01}, d_{02}, d_{03})$ defined, we have a (d_{21}, d_{22}, d_{23}) with $f(p_0)/(d_{21}, d_{22}, d_{23})$ defined and $f(p_0/(d_{01}, d_{02}, d_{03})) = f(p_0)/(d_{21}, d_{22}, d_{23})$.

Note that given a concrete graph grammar morphism $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2)$, any (partial) transition system structure $(TG_0, \overline{G}_0, P_0, \pi_0, /_0)$ on $(TG_0, \overline{G}_0, P_0, \pi_0)$ induces a partial transition system structure $(TG_2, \overline{G}_2, P_2, \pi_2, /_2)$ on $(TG_2, \overline{G}_2, P_2, \pi_2)$ such that the morphism $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2)$ extends to a partial transition system morphism $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0, /_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2, /_2)$. This happens as follows. Let $\pi_0(p) = A_0 \leftarrow B_0 \rightarrow C_0$, and $\pi_0(p/_0(a_0,b_0,c_0)) = A_0' \leftarrow B_0' \rightarrow C_0'$. Let $((A_0 \leftarrow B_0 \rightarrow C_0) \xrightarrow{TG_0TG_1TG_2} (A_2 \leftarrow B_2 \rightarrow C_2))$ and $((A_0' \leftarrow B_0' \rightarrow C_0') \xrightarrow{TG_0TG_1TG_2} (A_2' \leftarrow B_2' \rightarrow C_2'))$ be part of the concrete graph grammar morphism, so that $\pi_2(f(p)) = A_2 \leftarrow B_2 \rightarrow C_2$. Then definition 3.8 supplies us with a unique span morphism $(a_2,b_2,c_2) : (A_2 \leftarrow B_2 \rightarrow C_2) \rightarrow (A_2' \leftarrow B_2' \rightarrow C_2')$. This allows us to define $/_2$ on the image of $(TG_0, \overline{G}_0, P_0, \pi_0)$ by $f(p)/_2(a_2,b_2,c_2) = f(p/_0(a_0,b_0,c_0))$ giving a partial transition system structure $(TG_2, \overline{G}_2, P_2, \pi_2, /_2)$. (It is clear that the correct compositionality properties now hold, regarding the whole of $\pi(P)$ for a concrete graph transition system as one huge diagram.)

Definition 5.4 The category $\langle \mathcal{GraTS}* \rangle$ of abstract graph transition systems has as objects abstract graph transition systems, and as morphisms $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f)$: $(\langle TG_0\rangle, \langle \overline{G}_0\rangle, P_0, \pi_0, /_0) \rightarrow (\langle TG_2\rangle, \langle \overline{G}_2\rangle, P_2, \pi_2, /_2)$, where $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle$, $f)$ is a morphism between the underlying abstract graph grammars, such that for each p_0 in P_0 and $(\langle d_{01}\rangle, \langle d_{02}\rangle, \langle d_{03}\rangle)$ with $p_0/(\langle d_{01}\rangle, \langle d_{02}\rangle, \langle d_{03}\rangle)$ defined, we have a $(\langle d_{21}\rangle, \langle d_{22}\rangle, \langle d_{23}\rangle)$ with $f(p_0)/(\langle d_{21}\rangle, \langle d_{22}\rangle, \langle d_{23}\rangle)$ defined and $f(p_0/(\langle d_{01}\rangle, \langle d_{02}\rangle, \langle d_{03}\rangle)) = f(p_0)/(\langle d_{21}\rangle, \langle d_{22}\rangle, \langle d_{23}\rangle)$.

Clearly each abstract graph grammar morphism extends to a partial abstract graph transition system morphism by $f(p/_0(\langle a_0\rangle,\langle b_0\rangle,\langle c_0\rangle)) = f(p)/_2(\mathrm{cod}(((\langle a_0\rangle,\langle b_0\rangle,\langle c_0\rangle)$ -$TG_0TG_1TG_2$-› $(\langle a_2\rangle,\langle b_2\rangle,\langle c_2\rangle))))$ as above.

Obviously there are forgetful functors $\mathcal{U}* : \mathcal{GraTS}* \rightarrow \mathcal{GraGra}$ and $\langle \mathcal{U}* \rangle : \langle \mathcal{GraTS}* \rangle \rightarrow \langle \mathcal{GraGra} \rangle$ which just ignore /. We now give the construction that will provide a weak left adjoint functor to $\mathcal{U}*$ (and one for $\langle \mathcal{U}* \rangle$).

Definition 5.5 Let $GG = (TG, \overline{G}, P, \pi)$ be a concrete graph grammar. Then the concrete graph transition system $GTS* = (TG, \overline{G}, PP, \pi\pi, /)$ is given by

(1) $PP = \{(p, (t_1, t_2, t_3)) \mid p \in P$, and (t_1, t_2, t_3) is an object of $(\pi(p)\downarrow \mathcal{Sp}\text{-DP}*)\}$

(2) $\pi\pi((p, (t_1, t_2, t_3))) = \mathrm{cod}((t_1, t_2, t_3))$

(3) If $\pi\pi((p, (t_1, t_2, t_3))) = \mathrm{dom}((d_1, d_2, d_3))$ then $(p, (t_1, t_2, t_3))/(d_1, d_2, d_3)$ is defined, equals $(p, (d_1 \circ t_1, d_2 \circ t_2, d_3 \circ t_3))$, and thus $\pi\pi((p, (t_1, t_2, t_3))/(d_1, d_2, d_3)) = \mathrm{cod}((d_1 \circ t_1, d_2 \circ t_2, d_3 \circ t_3))$

Definition 5.6 Let $\langle GG \rangle = (\langle TG\rangle, \langle \overline{G}\rangle, P, \pi)$ be an abstract graph grammar. Then the abstract graph transition system $\langle GTS* \rangle = (\langle TG\rangle, \langle \overline{G}\rangle, PP, \pi\pi, /)$ is given by

(1) $PP = \{(p, (\langle t_1\rangle, \langle t_2\rangle, \langle t_3\rangle)) \mid p \in P$, and $(\langle t_1\rangle, \langle t_2\rangle, \langle t_3\rangle)$ is an object of $(\langle \pi(p)\rangle\downarrow\langle \mathcal{Sp}\text{-DP}*\rangle)\}$

(2) $\pi\pi((p, (\langle t_1\rangle, \langle t_2\rangle, \langle t_3\rangle))) = \mathrm{cod}((\langle t_1\rangle, \langle t_2\rangle, \langle t_3\rangle))$

(3) If $\pi\pi((p, (\langle t_1\rangle, \langle t_2\rangle, \langle t_3\rangle))) = \mathrm{dom}((\langle d_1\rangle, \langle d_2\rangle, \langle d_3\rangle))$ then $(p, (\langle t_1\rangle, \langle t_2\rangle, \langle t_3\rangle))/(\langle d_1\rangle, \langle d_2\rangle, \langle d_3\rangle)$ is defined, equals $(p, (\langle d_1\rangle \circ \langle t_1\rangle, \langle d_2\rangle \circ \langle t_2\rangle, \langle d_3\rangle \circ \langle t_3\rangle))$, and thus $\pi\pi((p, (\langle t_1\rangle, \langle t_2\rangle, \langle t_3\rangle))/(\langle d_1\rangle, \langle d_2\rangle, \langle d_3\rangle)) = \mathrm{cod}((\langle d_1\rangle \circ \langle t_1\rangle, \langle d_2\rangle \circ \langle t_2\rangle, \langle d_3\rangle \circ \langle t_3\rangle))$

It is obvious that $GTS*$ and $\langle GTS* \rangle$ are concrete and abstract graph transition systems respectively.

Theorem 5.7 The forgetful functor $\mathcal{U}* : \mathcal{GraTS}* \rightarrow \mathcal{GraGra}$ has a weak left adjoint $\mathcal{TS}* : \mathcal{GraGra} \rightarrow \mathcal{GraTS}*$ where the feeble functor[1] $\mathcal{TS}*(GG) = GTS*$ is given by definition 5.5 for objects, and is given for arrows by

1. A feeble functor F maps objects of the domain to objects of the codomain, and arrows of the domain to sets (for us classes) of arrows of the codomain, such that in an obvious notation, $F(g) \circ F(f) \subseteq F(g \circ f)$; see Krishnan (1981), also for weak adjunctions.

$\mathcal{TS}^*((TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : GG_0 \rightarrow GG_2)$ is the class of morphisms $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, g_P) : GTS^*_0 \rightarrow GTS^*_2$ in $\mathcal{GraTS}^*$ such that for all p in P_0, $g_P((p, \mathrm{id}_{\pi(p)})) = (f(p), \mathrm{id}_{\pi(f(p))})$

Proof. Define the universal arrow $u : GG \rightarrow \mathcal{U}^*(\mathcal{TS}^*(GG) = GTS^*)$ for a concrete graph grammar $GG = (TG, \overline{G}, P, \pi)$ by

$$u : GG \rightarrow \mathcal{U}^*(GTS^*) = (TG \leftarrow [TG] \rightarrow TG, in)$$

where $(TG \leftarrow [TG] \rightarrow TG)$ is an identity in $\mathcal{Gr\text{-}Sp}$, and $in(p) = (p, \mathrm{id}_{\pi(p)})$. Given a morphism $f : GG \rightarrow \mathcal{U}^*(TT) = (TG_0 \leftarrow [TG_1] \rightarrow TG_2, f_P)$ into the forgetful image of some transition system TT, we must show that there is a morphism $g : \mathcal{TS}^*(GG) \rightarrow TT$ such that $f = \mathcal{U}^*(g) \circ u$. See Fig. 11.

$$\begin{tikzcd}
GG \arrow[r, "f"] \arrow[d, "u"'] & \mathcal{U}^*(TT) & TT \\
\mathcal{U}^*(GTS^*) \arrow[ur, "\mathcal{U}^*(g)"'] & \mathcal{TS}^*(GG) = GTS^* \arrow[ur, "g"'] &
\end{tikzcd}$$

Fig. 11

Let g be any morphism $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, g_P) : GTS^*_0 \rightarrow TT$ such that for all p in P, $g_P((p, \mathrm{id}_{\pi(p)})) = (f(p)/\mathrm{id}_{\pi(f(p))})$. That such g's exist can be seen as follows. Firstly, we can make the triangle in Fig. 11, restricting $\mathcal{U}^*(g)$ to the image $u(GG)$, commute. This we do just by setting $\mathcal{U}^*(g)((p, \mathrm{id}_{\pi(p)})) = f(p)$ for all p in P. Secondly, we extend this (though not uniquely) to all of $\mathcal{U}^*(GTS^*)$ as follows. For all spans $\pi_{GTS^*}((p, (d_1, d_2, d_3)))$, we choose a span $(X \leftarrow Y \rightarrow Z)$ such that $(\pi_{GTS^*}((p, (d_1, d_2, d_3)))$ -$TG_0TG_1TG_2$-> $(X \leftarrow Y \rightarrow Z))$, doing this consistently with the choice already made firstly. Now $(d_1, d_2, d_3) : \pi_{GTS^*}((p, \mathrm{id}_{\pi(p)})) \rightarrow \pi_{GTS^*}((p, (d_1, d_2, d_3)))$ as a span morphism, and definition 3.8 enables us to find a unique $(t_1, t_2, t_3) : \pi_{TT}(f(p)/\mathrm{id}_{\pi(f(p))}) \rightarrow (X \leftarrow Y \rightarrow Z)$, enabling us to simultaneously deduce that $(X \leftarrow Y \rightarrow Z)$ is a production of $\mathcal{U}^*(TT)$. This gives $\mathcal{U}^*(g) : \mathcal{U}^*(GTS^*) \rightarrow \mathcal{U}^*(TT)$. We now set $g_P((p, (d_1, d_2, d_3))) = (f(p)/\mathrm{id}_{\pi(f(p))})/(t_1, t_2, t_3) = g_P((p, \mathrm{id}_{\pi(p)}))/(t_1, t_2, t_3)$. It is clear that all the required equations hold so that we have $g : \mathcal{TS}^*(GG) \rightarrow TT$. (So in the end we accomplish the whole thing in the reverse order to what one might expect.) ☺

Note that the above required a choice, but a less potentially problematic one than a choice of pullbacks.

Theorem 5.8 The forgetful functor $\langle \mathcal{U}^* \rangle : \langle \mathcal{GraTS}^* \rangle \rightarrow \langle \mathcal{GraGra} \rangle$ has a weak left adjoint $\langle \mathcal{TS}^* \rangle : \langle \mathcal{GraGra} \rangle \rightarrow \langle \mathcal{GraTS}^* \rangle$ where $\langle \mathcal{TS}^* \rangle(\langle GG \rangle) = \langle GTS^* \rangle$ is given by definition 5.6 for objects, and is given for arrows by

$\langle\mathcal{TS}*\rangle((\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f) : \langle GG_0\rangle \rightarrow \langle GG_2\rangle)$ is the class of morphisms $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, g_P) : \langle GTS*_0\rangle \rightarrow \langle GTS*_2\rangle$ in $\langle \mathcal{GraTS}*\rangle$ such that for all p in P_0, $g_P((p, \mathrm{id}_{\pi(p)})) = (f(p), \mathrm{id}_{\pi(f(p))})$

Finally definitions 5.3 and 5.4, which describe how the extra structure in a graph transition system behaves under morphisms, make the following expected statements evident.

Theorem 5.9 The obvious projection $\mathcal{P}_{\mathcal{GraTS}*} : \mathcal{GraTS}* \rightarrow \mathcal{Gr\text{-}Sp}$ is an opfibration, where all arrows $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0, /_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2, /_2)$ of $\mathcal{GraTS}*$ such that f is an iso in $\mathcal{Class}$ are are opcartesian.

Theorem 5.10 The obvious projection $\langle\mathcal{P}_{\mathcal{GraTS}*}\rangle : \langle\mathcal{GraTS}*\rangle \rightarrow \langle\mathcal{Gr\text{-}Sp}\rangle$ is an opfibration, where all arrows $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f) : (\langle TG_0\rangle, \langle\overline{G}_0\rangle, P_0, \pi_0, /_0) \rightarrow (\langle TG_2\rangle, \langle\overline{G}_2\rangle, P_2, \pi_2, /_2)$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

6 The Category of Typed Derivation Grammars

Roughly speaking, the previous two sections dealt with diagrams like Fig. 2, considered as DP*-$\mathcal{Sp}$ morphisms typed over a suitable graph, and subsequently composed vertically. Now we want to do something similar, but with regard to horizontal composition. The material in section 2 indicates that without eg. a choice of pullbacks (or injectivity of the productions), the results in this and the next section will be disjoint from those of the previous two because horizontal and vertical composition of such diagrams require different isomorphism properties of the middle vertical arrow between concrete representatives of a span. This explains the form of what follows.

Definition 6.1 The category of concrete typed derivation grammars $\mathcal{DerGra}$ has the following constituents.

Objects: $(TG, \overline{G}, P, \pi)$ where:

TG is a concrete type graph, i.e. an object of $\mathcal{Gr}$,

$\overline{G}$ is a concrete start graph typed over TG, i.e. an object of $(\mathcal{Gr}{\downarrow}TG)$,

P is a class of production names,

$\pi : P \rightarrow Arr(\mathcal{Gr\text{-}Sp}{\downarrow}TG)$ is a map from P to semiconcrete typed spans, i.e. arrows of $(\mathcal{Gr\text{-}Sp}{\downarrow}TG)$.

Arrows: $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2)$ which is shorthand for a collection of arrows.

Firstly: an arrow of $\mathcal{Gr\text{-}Sp}$, $(TG_0 \leftarrow [TG_1] \rightarrow TG_2) : TG_0 \rightarrow TG_2$,

Secondly: an arrow of $\mathcal{Gr} \,|\, \mathcal{Gr\text{-}Sp}$, $(\overline{G}_0 \text{ -}_{TG_0TG_1TG_2}\text{-}\!\!> \overline{G}_2) : (\overline{G}_0 \rightarrow TG_0) \rightarrow (\overline{G}_2 \rightarrow TG_2)$,

Thirdly: an arrow of $\mathcal{Class}$, $f : P_0 \rightarrow P_2$ i.e. a map,

Fourthly: for all $p \in P_0$ an arrow of *generalised-*$\mathcal{Diag} \,|\, \mathcal{Gr\text{-}Sp}$, $(\pi_0(p) \text{ -}_{TG_0TG_1TG_2}\text{-}\!\!> \pi_2(f(p))) : (\pi_0(p) \rightarrow TG_0) \rightarrow (\pi_2(f(p)) \rightarrow TG_2)$ where the action of $\text{-}_{TG_0TG_1TG_2}\text{-}\!\!>$ has been extended to semiconcrete spans.

Composition: $(TG_2 \leftarrow [TG_3] \rightarrow TG_4, g) \circ (TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) =$
$(TG_0 \leftarrow [TG_{2'}] \rightarrow TG_4, g \circ f)$ where $TG_{2'}$ is a pullback of
$TG_1 \rightarrow TG_2$ and $TG_2 \leftarrow TG_3$.

Identities: $(TG \leftarrow [TG] \rightarrow TG, \mathrm{id}_P) : (TG, \overline{G}, P, \pi) \rightarrow (TG, \overline{G}, P, \pi)$.

Note that this differs from definition 4.1 only in the use of semiconcrete spans instead of concrete spans. A similar thing holds for the abstract version.

Definition 6.2 The category of abstract typed derivation grammars $\langle \mathcal{DerGra} \rangle$ is similar to definition 4.2 but using abstract typed spans (i.e. arrows of $(\langle \mathcal{Gr}\text{-}\mathcal{Sp} \rangle \downarrow \langle TG \rangle))$ instead of special abstract typed spans (i.e. objects of $(\langle \mathcal{Sp} \rangle \downarrow \langle TG \rangle))$.

We immediately deduce:

Theorem 6.3 The projection $\mathcal{P}_{\mathcal{DerGra}} : \mathcal{DerGra} \rightarrow \mathcal{Gr}\text{-}\mathcal{Sp}$ such that

$$\mathcal{P}_{\mathcal{DerGra}}((TG, \overline{G}, P, \pi)) = TG \text{ and}$$
$$\mathcal{P}_{\mathcal{DerGra}}((TG_0 \leftarrow [TG_1] \rightarrow TG_2, f)) = (TG_0 \leftarrow [TG_1] \rightarrow TG_2)$$

is an opfibration, in which the arrows $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2)$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

Theorem 6.4 The projection $\langle \mathcal{P}_{\mathcal{DerGra}} \rangle : \langle \mathcal{DerGra} \rangle \rightarrow \langle \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ such that

$$\langle \mathcal{P}_{\mathcal{DerGra}} \rangle((\langle TG \rangle, \langle \overline{G} \rangle, P, \pi)) = \langle TG \rangle \text{ and}$$
$$\langle \mathcal{P}_{\mathcal{DerGra}} \rangle((\langle TG_0 \rangle \leftarrow [TG_1] \rightarrow \langle TG_2 \rangle, f)) = (\langle TG_0 \rangle \leftarrow [TG_1] \rightarrow \langle TG_2 \rangle)$$

is an opfibration, in which the arrows $(\langle TG_0 \rangle \leftarrow [TG_1] \rightarrow \langle TG_2 \rangle, f) : (\langle TG_0 \rangle, \langle \overline{G}_0 \rangle, P_0, \pi_0) \rightarrow (\langle TG_2 \rangle, \langle \overline{G}_2 \rangle, P_2, \pi_2)$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

We get the expected results for direct derivations.

Theorem 6.5 The category $\mathcal{Diag} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp}$ has the subcategory $Arr(\mathcal{Gr}\text{-}\mathcal{Sp})$-DPB $|$-$\mathcal{Gr}\text{-}\mathcal{Sp}$ which describes how direct derivation steps under DPB rewriting (with interface graphs in both productions and derivation steps defined only up to arbitrary isomorphism) are preserved under change of type. Also $\mathcal{Diag} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp}$ has the subcategory $Arr(\mathcal{Gr}\text{-}\mathcal{Sp})$-DPO $|$ $\mathcal{Gr}\text{-}\mathcal{Sp}$ which describes how direct derivation steps under DPO rewriting (with interface graphs in both productions and derivation steps defined only up to arbitrary isomorphism) are preserved under change of type.

Theorem 6.6 The category $\langle \mathcal{Diag} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ has the subcategory $\langle Arr(\mathcal{Gr}\text{-}\mathcal{Sp})$-DPB $|$-$\mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ which describes how direct derivation steps under DPB rewriting (with interface graphs in both productions and derivation steps defined only up to arbitrary isomorphism) are preserved under change of type. Also $\langle \mathcal{Diag} \,|\, \mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ has the subcategory $\langle Arr(\mathcal{Gr}\text{-}\mathcal{Sp})$-DPO $|$ $\mathcal{Gr}\text{-}\mathcal{Sp} \rangle$ which describes how direct derivation steps under DPO rewriting (with interface graphs in both productions and derivation steps defined only up to arbitrary isomorphism) are preserved under change of type.

And of course:

Theorem 6.7 The obvious projection $\mathcal{P}_{Arr(\mathcal{Gr}\text{-}\mathcal{Sp})\text{-DP}*\varnothing} : Arr(\mathcal{Gr}\text{-}\mathcal{Sp})\text{-DP}* \,|\, \mathcal{Gr}\text{-}\mathcal{Sp} \rightarrow \mathcal{Gr}\text{-}\mathcal{Sp}$ is an opfibration where all arrows of $Arr(\mathcal{Gr}\text{-}\mathcal{Sp})$-DP* $|$ $\mathcal{Gr}\text{-}\mathcal{Sp}$ are opcartesian.

Theorem 6.8 The obvious projection $\langle \mathcal{P}_{Arr(\mathcal{Gr\text{-}Sp})\text{-DP}*\varnothing} \rangle : \langle Arr(\mathcal{Gr\text{-}Sp})\text{-DP}* \mid \mathcal{Gr\text{-}Sp} \rangle \to \langle \mathcal{Gr\text{-}Sp} \rangle$ is an opfibration, where all arrows of $\langle Arr(\mathcal{Gr\text{-}Sp})\text{-DP}* \mid \mathcal{Gr\text{-}Sp} \rangle$ are opcartesian.

There is a close relationship between graph grammars and derivation grammars. Given a graph grammar, if one simply allows the middle graphs of the spans that interpret production names to be taken up to arbitrary rather than standard isomorphism, one gets a derivation grammar. Here is the more formal construction.

Definition 6.9 Let $A \leftarrow B \to C$ be a concrete span, then $\mathcal{M}(A \leftarrow B \to C) = A \leftarrow [B] \to C$ which is the semiconcrete span such that $A \leftarrow B \to C$ is one of its representatives. Likewise let $\langle A \rangle \leftarrow \langle B \rangle \to \langle C \rangle$ be a special abstract span, then $\langle \mathcal{M} \rangle(\langle A \rangle \leftarrow \langle B \rangle \to \langle C \rangle) = \langle A \rangle \leftarrow [B] \to \langle C \rangle$ which is the abstract span such that each representative $A \leftarrow B \to C$ of $\langle A \rangle \leftarrow \langle B \rangle \to \langle C \rangle$ is also a representative of $\langle A \rangle \leftarrow [B] \to \langle C \rangle$.

Definition 6.10 Let $GG = (TG, \overline{G}, P, \pi_g)$ be a concrete graph grammar. Then the concrete derivation grammar $\mathcal{M}(GG) = DG = (TG, \overline{G}, P, \pi_d)$ is given by $\pi_d(p) = \mathcal{M}(\pi_g(p))$.

Definition 6.11 Let $\langle GG \rangle = (\langle TG \rangle, \langle \overline{G} \rangle, P, \pi_g)$ be an abstract graph grammar. Then the abstract derivation grammar $\langle \mathcal{M} \rangle(\langle GG \rangle) = \langle DG \rangle = (\langle TG \rangle, \langle \overline{G} \rangle, P, \pi_d)$ is given by $\pi_d(p) = \langle \mathcal{M} \rangle(\pi_g(p))$.

The next two results follow from our general consideration on diagrams, and because a grammar is essentially just a class of productions.

Proposition 6.12 The projection $\mathcal{M}_{\mathcal{GrDr}\varnothing} : \mathcal{GraGra} \to \mathcal{DerGra}$ given by definition 6.10 for objects with the obvious extension to arrows which makes it into a congruence w.r.t. arrow composition, has the property that all arrows $(TG_0 \leftarrow [TG_1] \to TG_2, f)$ of $\mathcal{GraGra}$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

Proposition 6.13 The projection $\langle \mathcal{M}_{\mathcal{GrDr}\varnothing} \rangle : \langle \mathcal{GraGra} \rangle \to \langle \mathcal{DerGra} \rangle$ given by definition 6.11 for objects with the obvious extension to arrows which makes it into a congruence w.r.t. arrow composition, has the property that all arrows $(\langle TG_0 \rangle \leftarrow [TG_1] \to \langle TG_2 \rangle, f)$ of $\langle \mathcal{GraGra} \rangle$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

Note that $\mathcal{M}_{\mathcal{GrDr}\varnothing}$ and $\langle \mathcal{M}_{\mathcal{GrDr}\varnothing} \rangle$ do not extend to opfibrations because the component f of a graph grammar projects identically (under $\mathcal{M}_{\mathcal{GrDr}\varnothing}$ or $\langle \mathcal{M}_{\mathcal{GrDr}\varnothing} \rangle$), also when f is not an iso.

7 The Category of Derivation Systems

Now that we have the preceding in place we can define derivation systems as enriched derivation grammars such that the class of production names supports a partial operation $;$ of horizontal composition.

Definition 7.1 A concrete graph derivation system is a quintuple $(TG, \overline{G}, P, \pi, ;)$ where $(TG, \overline{G}, P, \pi)$ is a concrete derivation grammar, and $; : P \times P \nrightarrow P$ satisfies

(1) If $\pi(p) = A \leftarrow [B] \to C$ and $\pi(q) = C \leftarrow [D] \to E$ then $p;q$ is defined and $\pi(p;q) = A \leftarrow [M] \to E$, where M is a pullback of $B \to C$ and $C \leftarrow D$

(2) If $\pi(p) = A \leftarrow [B] \rightarrow C$ then P contains a p_A with $\pi(p_A) = A \leftarrow [A] \rightarrow A$, an identity name such that $p_A ; p = p$, and also a similar identity name p_C with $\pi(p_C) = C \leftarrow [C] \rightarrow C$ and $p ; p_C = p$

(3) $;$ is associative

Definition 7.2 An abstract graph derivation system is a quintuple $(\langle TG\rangle, \langle\overline{G}\rangle, P, \pi, ;)$ where $(\langle TG\rangle, \langle\overline{G}\rangle, P, \pi)$ is an abstract derivation grammar, and $; : P \times P \leftrightarrow P$ satisfies

(1) If $\pi(p) = \langle A\rangle \leftarrow [B] \rightarrow \langle C\rangle$ and $\pi(q) = \langle C\rangle \leftarrow [D] \rightarrow \langle E\rangle$ then $p;q$ is defined and $\pi(p;q) = \langle A\rangle \leftarrow [M] \rightarrow \langle E\rangle$, where M is a pullback of $B \rightarrow C$ and $C \leftarrow D$

(2) If $\pi(p) = \langle A\rangle \leftarrow [B] \rightarrow \langle C\rangle$ then P contains a $p_{\langle A\rangle}$ with $\pi(p_{\langle A\rangle}) = \langle A\rangle \leftarrow [A] \rightarrow \langle A\rangle$, an identity name such that $p_{\langle A\rangle} ; p = p$, and also a similar identity name $p_{\langle C\rangle}$ with $\pi(p_{\langle C\rangle}) = \langle C\rangle \leftarrow [C] \rightarrow \langle C\rangle$ and $p ; p_{\langle C\rangle} = p$

(3) $;$ is associative

Definition 7.3 The category $\mathcal{GraDS}*$ of concrete graph derivation systems has as objects concrete graph derivation systems, and as morphisms $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0, ;_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2, ;_2)$, where $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f)$ is a morphism between the underlying concrete derivation grammars, such that for each identity name p_A in $P_0, f(p_A)$ is an identity name, and for each (p, q) pair defined for $;_0$, we have $f(p ;_0 q) = f(p) ;_2 f(q)$.

Definition 7.4 The category $\langle\mathcal{GraDS}*\rangle$ of abstract graph derivation systems has as objects abstract graph derivation systems, and as morphisms $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f) : (\langle TG_0\rangle, \langle\overline{G}_0\rangle, P_0, \pi_0, ;_0) \rightarrow (\langle TG_2\rangle, \langle\overline{G}_2\rangle, P_2, \pi_2, ;_2)$, where $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f)$ is a morphism between the underlying abstract derivation grammars, such that for each identity name $p_{\langle A\rangle}$ in $P_0, f(p_{\langle A\rangle})$ is an identity name, and for each (p, q) pair defined for $;_0$, we have $f(p ;_0 q) = f(p) ;_2 f(q)$.

As in section 5 there are forgetful functors $\mathcal{V}* : \mathcal{GraDS}* \rightarrow \mathcal{DerGra}$ and $\langle\mathcal{V}*\rangle : \langle\mathcal{GraDS}*\rangle \rightarrow \langle\mathcal{DerGra}\rangle$ which just ignore $;$. We now give the construction that will provide a left adjoint functor to $\mathcal{V}*$ (and one for $\langle\mathcal{V}*\rangle$).

Definition 7.5 Let $DG = (TG, \overline{G}, P, \pi)$ be a concrete derivation grammar. Then the concrete graph derivation system $GDS* = (TG, \overline{G}, PP, \pi\pi, ;_{PP})$ is given by constructing PPP, $\pi\pi\pi$ and $;_{PPP}$ as the smallest classes containing:

(1) (p) is in PPP for p in P, and $\pi\pi\pi((p)) = \pi(p)$

(2) (p_A) and (p_C) are in PPP for each p in P with $\pi(p) = A \leftarrow [B] \rightarrow C$, and $\pi\pi\pi((p_A)) = A \leftarrow [A] \rightarrow A$ and $\pi\pi\pi((p_C)) = C \leftarrow [C] \rightarrow C$, and $\pi\pi\pi((p_A) ;_{PPP} (p)) = \pi\pi\pi((p)) = \pi\pi\pi((p) ;_{PPP} (p_C))$

(3) (p, q) is in PPP for p, q in PPP such that $\pi\pi\pi(p) = A \leftarrow [B] \rightarrow C$ and $\pi\pi\pi(p) = C \leftarrow [D] \rightarrow E$; $(p, q) = p ;_{PPP} q$, and $\pi\pi\pi((p, q))$ is given via the pullback of $\pi\pi\pi(p)$ and $\pi\pi\pi(q)$ as per definition 7.1

And then letting PP, $\pi\pi$ and $;_{PP}$ be given by taking PPP, $\pi\pi\pi$ and $;_{PPP}$ modulo the associative law $((\mathrm{A} ;_{PPP} \mathrm{B}) ;_{PPP} \mathrm{C}) = (\mathrm{A} ;_{PPP} (\mathrm{B} ;_{PPP} \mathrm{C}))$ and identity laws $(p_A) ;_{PPP} (p) = (p) = (p) ;_{PPP} (p_C)$.

It is obvious that $\mathit{GDS*}$ is a concrete graph derivation system. Note that it has a pleasing smallness property. If we start out with a concrete derivation grammar having a set rather than a class of production names, then the $\mathit{GDS*}$ we construct also has a set rather than a class of production names, unlike the transition system construction of the previous section. Here is the abstract version.

Definition 7.6 Let $\langle DG \rangle = (\langle TG \rangle, \langle \overline{G} \rangle, P, \pi)$ be an abstract derivation grammar. Then the abstract graph derivation system $\langle GDS* \rangle = (\langle TG \rangle, \langle \overline{G} \rangle, PP, \pi\pi, ;_{PP})$ is given by constructing PPP, $\pi\pi\pi$ and $;_{PPP}$ as the smallest classes containing:

(1) (p) is in PPP for p in P, and $\pi\pi\pi((p)) = \pi(p)$

(2) $(p_{\langle A \rangle})$ and $(p_{\langle C \rangle})$ are in PPP for each p in P with $\pi(p) = \langle A \rangle \leftarrow [B] \rightarrow \langle C \rangle$, and $\pi\pi\pi((p_{\langle A \rangle})) = \langle A \rangle \leftarrow [A] \rightarrow \langle A \rangle$ and $\pi\pi\pi((p_{\langle C \rangle})) = \langle C \rangle \leftarrow [C] \rightarrow \langle C \rangle$, and $\pi\pi\pi((p_{\langle A \rangle});_{PPP}(p)) = \pi\pi\pi((p)) = \pi\pi\pi((p);_{PPP}(p_{\langle C \rangle}))$

(3) (p, q) is in PPP for p, q in PPP such that $\pi\pi\pi(p) = \langle A \rangle \leftarrow [B] \rightarrow \langle C \rangle$ and $\pi\pi\pi(p) = \langle C \rangle \leftarrow [D] \rightarrow \langle E \rangle$; $(p, q) = p;_{PPP}q$, and $\pi\pi\pi((p, q))$ is given via the pullback of $\pi\pi\pi(p)$ and $\pi\pi\pi(q)$ as per definition 7.2

And then letting PP, $\pi\pi$ and $;_{PP}$ be given by taking PPP, $\pi\pi\pi$ and $;_{PPP}$ modulo the associative law $((\mathrm{A};_{PPP}\mathrm{B});_{PPP}\mathrm{C}) = (\mathrm{A};_{PPP}(\mathrm{B};_{PPP}\mathrm{C}))$ and identity laws $(p_{\langle A \rangle});_{PPP}(p) = (p) = (p);_{PPP}(p_{\langle C \rangle})$.

Theorem 7.7 The forgetful functor $\mathcal{V}* : \mathcal{GraDS}* \rightarrow \mathcal{DerGra}$ has a left adjoint $\mathcal{DS}* : \mathcal{DerGra} \rightarrow \mathcal{GraDS}*$ where $\mathcal{DS}*(DG) = GDS*$ is given by definition 7.5 for objects, with the unique extension for arrows.

Proof. Define the universal arrow $v : DG \rightarrow \mathcal{V}*(\mathcal{DS}*(DG) = GDS*)$ for a concrete derivation grammar $DG = (TG, \overline{G}, P, \pi)$ by

$$v : DG \rightarrow \mathcal{V}*(GDS*) = (TG \leftarrow [TG] \rightarrow TG, in)$$

where $(TG \leftarrow [TG] \rightarrow TG)$ is an identity in $\mathcal{Gr\text{-}Sp}$, and $in(p) = (p)$. Given a morphism $f : DG \rightarrow \mathcal{V}*(TT) = (TG_0 \leftarrow [TG_1] \rightarrow TG_2, f_P)$ into the forgetful image of some derivation system TT, we must show there is a unique morphism $g : \mathcal{DS}*(DG) \rightarrow TT$ such that $f = \mathcal{V}*(g) \circ v$.

Let g be given by $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, g_P) : GDS* \rightarrow TT$ such that for all p in P, $g_P((p)) = f_P(p)$, extending by $;$ to the whole of $GDS*$. It is clear that the required properties hold. ☺

Theorem 7.8 The forgetful functor $\langle \mathcal{V}* \rangle : \langle \mathcal{GraDS}* \rangle \rightarrow \langle \mathcal{DerGra} \rangle$ has a left adjoint $\langle \mathcal{DS}* \rangle : \langle \mathcal{DerGra} \rangle \rightarrow \langle \mathcal{GraDS}* \rangle$ where $\langle \mathcal{DS}* \rangle(\langle DG \rangle) = \langle GDS* \rangle$ is given given by definition 7.6 for objects, with the unique extension for arrows.

Finally we have:

Theorem 7.9 The obvious projection $\mathcal{P}_{\mathcal{GraDS}*} : \mathcal{GraDS}* \rightarrow \mathcal{Gr\text{-}Sp}$ is an opfibration, where all arrows $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0, ;_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2, ;_2)$ of $\mathcal{GraDS}*$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

Theorem 7.10 The obvious projection $\langle \mathcal{P}_{\mathcal{G}ra\mathcal{DS}*} \rangle : \langle \mathcal{G}ra\mathcal{DS}* \rangle \to \langle \mathcal{G}r\text{-}\mathcal{S}p \rangle$ is an opfibration, where all arrows $(\langle TG_0 \rangle \leftarrow [TG_1] \to \langle TG_2 \rangle, f) : (\langle TG_0 \rangle, \langle \overline{G}_0 \rangle, P_0, \pi_0, ;_0) \to (\langle TG_2 \rangle, \langle \overline{G}_2 \rangle, P_2, \pi_2, ;_2)$ such that f is an iso in $\mathcal{C}lass$ are opcartesian.

The above results simply depend on the horizontal composability of semiconcrete or abstract spans. To yield something nontrivial, one must assume that the derivation grammar one starts with is sufficiently well endowed with composable productions that the $;$ operation on production names is nonempty. A convenient method for assuring this is to start with a graph grammar, to take all the spans produced via the transition system construction, to ignore the span morphism aspects, and to reinterpret the concrete or semiabstract spans as semiconcrete or special abstract spans via the construction at the end of section 6.

So let GG be a concrete graph grammar. The derivation grammar induced from GG via the construction just described is thus $\mathcal{M}_{\mathcal{G}r\mathcal{D}r\varnothing}(\mathcal{U}*(\mathcal{TS}*(GG)))$ and $\mathcal{DS}*(\mathcal{M}_{\mathcal{G}r\mathcal{D}r\varnothing}(\mathcal{U}*(\mathcal{TS}*(GG))))$ is the derivation system produced. Likewise at the abstract level, starting with an abstract graph grammar $\langle GG \rangle$ we get $\langle \mathcal{M}_{\mathcal{G}r\mathcal{D}r\varnothing} \rangle (\langle \mathcal{U}* \rangle (\langle \mathcal{TS}* \rangle (\langle GG \rangle)))$ and $\langle \mathcal{DS}* \rangle (\langle \mathcal{M}_{\mathcal{G}r\mathcal{D}r\varnothing} \rangle (\langle \mathcal{U}* \rangle (\langle \mathcal{TS}* \rangle (\langle GG \rangle))))$. Note that because $\mathcal{M}_{\mathcal{G}r\mathcal{D}r\varnothing}$ and $\langle \mathcal{M}_{\mathcal{G}r\mathcal{D}r\varnothing} \rangle$ cannot be extended to preserve the algebraic structure of a transition system, the remarks at the beginning of section 6 hold true. In particular the interchange law of Corradini et al. (1996b) that allows one to recover horizontal or vertical composition from a structure containing just the other is not valid. (In the injective production case, such a law arises as the standard interchange law for a double category which fails to exist in the general case. See Banach and Corradini (1996).)

8 The Category of Typed Computation Grammars

In the previous two sections, horizontal composability of productions had to be considered "on the nose", because of the different isomorphism properties required of the middle of a span for horizontal and vertical composition. But this is somewhat different from the usual way that rule based systems give their operational semantics. Normally, one seeks a redex for a rule in the current computational state, and then performs a derivation step resulting in a new state. The process then repeats with the new state. In this and the next section we give a construction that mimics this process more closely. To do so we have to consider the horizontal and vertical aspects together within the same diagram. The main idea is to consider "computation grammars", i.e., graph grammars where productions are replaced by a class of derivation step names, each having an associated DP* diagram suitably typed. Next "computation systems" will be defined as computation grammars where the class of derivation steps is closed under a partial composition operation that is applicable to two DP* diagrams sharing a common corner.

Definition 8.1 The category of concrete typed computation grammars $\mathcal{C}om\mathcal{G}ra*$ has the following constituents.

Objects: $(TG, \overline{G}, P, \pi)$ where:
TG is a concrete type graph, i.e. an object of $\mathcal{G}r$,
$\overline{G}$ is a concrete start graph typed over TG,

i.e. an object of ($\mathcal{Gr}{\downarrow}TG$),

P is a class of derivation step names,

$\pi : P \to |\ \mathrm{DP}{*}\mathcal{DerSt}{\downarrow}TG\ |$ is a map from P to semiconcrete typed derivation steps, i.e. objects of ($\mathrm{DP}{*}\mathcal{DerSt}{\downarrow}TG$).

Arrows: $(TG_0 \leftarrow [TG_1] \to TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0) \to (TG_2, \overline{G}_2, P_2, \pi_2)$ which is shorthand for a collection of arrows.

Firstly: an arrow of $\mathcal{Gr}\text{-}\mathcal{Sp}$, $(TG_0 \leftarrow [TG_1] \to TG_2) : TG_0 \to TG_2$,

Secondly: an arrow of $\mathcal{Gr}\,|\,\mathcal{Gr}\text{-}\mathcal{Sp}$,

$(\overline{G}_0 \text{-}_{TG_0TG_1TG_2}\text{->}\ \overline{G}_2) : (\overline{G}_0 \to TG_0) \to (\overline{G}_2 \to TG_2)$,

Thirdly: an arrow of $\mathcal{Class}$, $f : P_0 \to P_2$ i.e. a map,

Fourthly: for all $p \in P_0$ an arrow of *generalised-*$\mathcal{Diag}\,|\,\mathcal{Gr}\text{-}\mathcal{Sp}$,

$(\pi_0(p) \text{-}_{TG_0TG_1TG_2}\text{->}\ \pi_2(f(p))) : (\pi_0(p) \to TG_0) \to (\pi_2(f(p)) \to TG_2)$

where the action of $\text{-}_{TG_0TG_1TG_2}\text{->}$ has been extended to semiconcrete derivation steps.

Composition: $(TG_2 \leftarrow [TG_3] \to TG_4, g) \circ (TG_0 \leftarrow [TG_1] \to TG_2, f) =$
$(TG_0 \leftarrow [TG_{2'}] \to TG_4, g \circ f)$ where $TG_{2'}$ is a pullback of $TG_1 \to TG_2$ and $TG_2 \leftarrow TG_3$.

Identities: $(TG \leftarrow [TG] \to TG, \mathrm{id}_P) : (TG, \overline{G}, P, \pi) \to (TG, \overline{G}, P, \pi)$.

This differs from definition 6.1 in the use of semiconcrete derivation steps instead of semiconcrete spans. A similar thing holds for the abstract version.

Definition 8.2 The category of abstract typed computation grammars $\langle\mathcal{ComGra}{*}\rangle$ is similar to definition 4.2 but using abstract typed derivation steps (i.e. objects of ($\langle\mathrm{DP}{*}\text{-}\mathcal{DerSt}\rangle{\downarrow}\langle TG\rangle$)) instead of special abstract typed spans (i.e. objects of ($\langle\mathcal{Sp}\rangle{\downarrow}\langle TG\rangle$)).

From our general results on diagrams we immediately deduce:

Theorem 8.3 The projection $\mathcal{P}_{\mathcal{ComGra}*} : \mathcal{ComGra}{*} \to \mathcal{Gr}\text{-}\mathcal{Sp}$ such that

$$\mathcal{P}_{\mathcal{ComGra}*}((TG, \overline{G}, P, \pi)) = TG \text{ and}$$
$$\mathcal{P}_{\mathcal{ComGra}*}((TG_0 \leftarrow [TG_1] \to TG_2, f)) = (TG_0 \leftarrow [TG_1] \to TG_2)$$

is an opfibration, in which the arrows $(TG_0 \leftarrow [TG_1] \to TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0) \to (TG_2, \overline{G}_2, P_2, \pi_2)$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

Theorem 8.4 The projection $\langle\mathcal{P}_{\mathcal{ComGra}*}\rangle : \langle\mathcal{ComGra}{*}\rangle \to \langle\mathcal{Gr}\text{-}\mathcal{Sp}\rangle$ such that

$$\langle\mathcal{P}_{\mathcal{ComGra}*}\rangle((\langle TG\rangle, \langle\overline{G}\rangle, P, \pi)) = \langle TG\rangle \text{ and}$$
$$\langle\mathcal{P}_{\mathcal{ComGra}*}\rangle((\langle TG_0\rangle \leftarrow [TG_1] \to \langle TG_2\rangle, f)) = (\langle TG_0\rangle \leftarrow [TG_1] \to \langle TG_2\rangle)$$

is an opfibration, in which the arrows $(\langle TG_0\rangle \leftarrow [TG_1] \to \langle TG_2\rangle, f) : (\langle TG_0\rangle, \langle\overline{G}_0\rangle, P_0, \pi_0) \to (\langle TG_2\rangle, \langle\overline{G}_2\rangle, P_2, \pi_2)$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

Definition 8.5 The forgetful functor $\mathcal{Prod}{*} : \mathcal{ComGra}{*} \to \mathcal{DerGra}$ takes concrete computation grammars to concrete derivation grammars by $\mathcal{Prod}{*}((TG, \overline{G}, P, \pi_{ds})) = (TG, \overline{G}, P, \pi_d)$ where if $\pi_{ds}(p) = ((L \leftarrow [K] \to R)\ \text{-}[g,d,h]\text{->}\ (G \leftarrow [D] \to H))$ then $\pi_d(p) = (L \leftarrow [K] \to R)$.

Definition 8.6 The forgetful functor $\langle \mathcal{P}rod* \rangle : \langle \mathcal{C}om\mathcal{G}ra* \rangle \to \langle \mathcal{D}er\mathcal{G}ra \rangle$ takes abstract computation grammars to abstract derivation grammars by $\mathcal{P}rod*((\langle TG \rangle, \langle \overline{G} \rangle, P, \pi_{ds}))$ $= (\langle TG \rangle, \langle \overline{G} \rangle, P, \pi_d)$ where if $\pi_{ds}(p) = ((\langle L \rangle \leftarrow [K] \to \langle R \rangle)$ -$[g,d,h]$-› $(\langle G \rangle \leftarrow [D] \to \langle H \rangle))$ then $\pi_d(p) = (\langle L \rangle \leftarrow [K] \to \langle R \rangle)$.

So $\mathcal{P}_{\mathcal{C}om\mathcal{G}ra*} = \mathcal{P}_{\mathcal{D}er\mathcal{G}ra} \circ \mathcal{P}rod*$ and $\langle \mathcal{P}_{\mathcal{C}om\mathcal{G}ra*} \rangle = \langle \mathcal{P}_{\mathcal{D}er\mathcal{G}ra} \rangle \circ \langle \mathcal{P}rod* \rangle$.

We now give the construction that will provide (this time) a weak right adjoint functor to $\mathcal{P}rod*$ (and one for $\langle \mathcal{P}rod* \rangle$).

Definition 8.7 Let $DG = (TG, \overline{G}, P, \pi)$ be a concrete derivation grammar. Then the concrete computation grammar $CG* = (TG, \overline{G}, PP, \pi\pi)$ is given by constructing PP and $\pi\pi$ as the smallest classes such that:

> $(p, [g,d,h])$ is in PP for each p in P and $[g,d,h]$ such that $((L \leftarrow [K] \to R)$ -$[g,d,h]$-› $(G \leftarrow [D] \to H))$ is a semiconcrete derivation step, where $(L \leftarrow [K] \to R) = \pi(p)$ and $\pi\pi((p, [g,d,h])) = ((L \leftarrow [K] \to R)$ -$[g,d,h]$-› $(G \leftarrow [D] \to H))$

Definition 8.8 Let $\langle DG \rangle = (\langle TG \rangle, \langle \overline{G} \rangle, P, \pi)$ be an abstract derivation grammar. Then the abstract computation grammar $\langle CG* \rangle = (\langle TG \rangle, \langle \overline{G} \rangle, PP, \pi\pi)$ is given by constructing PP and $\pi\pi$ as the smallest classes such that:

> $(p, [g,d,h])$ is in PP for each p in P and $[g,d,h]$ such that $((\langle L \rangle \leftarrow [K] \to \langle R \rangle)$ -$[g,d,h]$-› $(\langle G \rangle \leftarrow [D] \to \langle H \rangle))$ is an abstract derivation step, where $(\langle L \rangle \leftarrow [K] \to \langle R \rangle) = \pi(p)$ and $\pi\pi((p, [g,d,h])) = ((\langle L \rangle \leftarrow [K] \to \langle R \rangle)$ -$[g,d,h]$-› $(\langle G \rangle \leftarrow [D] \to \langle H \rangle))$

Theorem 8.9 The forgetful functor $\mathcal{P}rod* : \mathcal{C}om\mathcal{G}ra* \to \mathcal{D}er\mathcal{G}ra$ has a weak right adjoint $\mathcal{CG}* : \mathcal{D}er\mathcal{G}ra \to \mathcal{C}om\mathcal{G}ra*$ where $\mathcal{CG}*(DG) = CG*$ is given by definition 8.7 for objects, and is given for arrows by

> $\mathcal{CG}*((TG_0 \leftarrow [TG_1] \to TG_2, f) : DG_0 \to DG_2)$ is the class of morphisms
> $(TG_0 \leftarrow [TG_1] \to TG_2, e_P) : CG*_0 \to CG*_2$ in $\mathcal{C}om\mathcal{G}ra*$ such that
> for all p in P_0, $e_P((p, [g_0,d_0,h_0])) = (f(p), \mathrm{cod}([g_0,d_0,h_0]$ -$TG_0TG_1TG_2$-› $[g_2,d_2,h_2]))$

Proof. Define the universal arrow $prod : \mathcal{P}rod*(\mathcal{CG}*(DG) = CG*) \to DG$ for a concrete computation grammar $DG = (TG, \overline{G}, P, \pi)$ by

> $prod : \mathcal{P}rod*(CG*) \to DG = (TG \leftarrow [TG] \to TG, out)$

where $(TG \leftarrow [TG] \to TG)$ is an identity in $\mathcal{G}r\text{-}\mathcal{S}p$, and $out((p, [g,d,h])) = p$. Given a morphism $f : \mathcal{P}rod*(TT) \to DG = (TG_0 \leftarrow [TG_1] \to TG_2, f_P)$ from the forgetful image of some computation grammar $TT = (TG_0, \overline{G}, T, \pi_T)$, we must show that there is a morphism $e : TT \to \mathcal{CG}*(DG)$ such that $f = prod \circ \mathcal{P}rod*(e)$. See Fig. 12.

We choose $e = (TG_0 \leftarrow [TG_1] \to TG_2, e_P)$ where, if for p in T $\pi_T(p) = ((L \leftarrow [K] \to R)$ -$[g,d,h]$-› $(G \leftarrow [D] \to H))$, then $e_P(p) = (f_P(p), \mathrm{cod}([g,d,h]$ -$TG_0TG_1TG_2$-› $[g',d',h']))$. In this, $[g,d,h]$ -$TG_0TG_1TG_2$-› $[g',d',h']$ is a suitable diagram morphism; these exist for any arrow $(TG_0 \leftarrow [TG_1] \to TG_2)$ by previous results. That e is a morphism is evident; likewise that $f = prod \circ \mathcal{P}rod*(e)$ for any such e. ☺

$$
\begin{tikzcd}
 & \mathcal{CG}*(DG) = CG* & & \mathcal{Prod}*(CG*) \arrow[d, "prod"] \\
TT \arrow[ur, "e"] & & \mathcal{Prod}*(TT) \arrow[ur, "\mathcal{Prod}*(e)"] \arrow[r, "f"'] & DG
\end{tikzcd}
$$

Fig. 12

Theorem 8.10 The forgetful functor $\langle \mathcal{Prod}* \rangle : \langle \mathcal{ComGra}* \rangle \to \langle \mathcal{DerGra} \rangle$ has a weak right adjoint $\langle \mathcal{CG}* \rangle : \langle \mathcal{DerGra} \rangle \to \langle \mathcal{ComGra}* \rangle$ where $\langle \mathcal{CG}* \rangle(\langle DG \rangle) = \langle CG* \rangle$ is given by definition 8.8 for objects, and is given for arrows by

> $\langle \mathcal{CG}* \rangle((\langle TG_0 \rangle \leftarrow [TG_1] \to \langle TG_2 \rangle, f) : \langle DG_0 \rangle \to \langle DG_2 \rangle)$ is the class of morphisms $(\langle TG_0 \rangle \leftarrow [TG_1] \to \langle TG_2 \rangle,\, e_P) : \langle CG*_0 \rangle \to \langle CG*_2 \rangle$ in $\langle \mathcal{ComGra}* \rangle$ such that for all p in P_0, $e_P((p, [g_0,d_0,h_0])) = (f(p), \mathrm{cod}([g_0,d_0,h_0] \text{ -}TG_0TG_1TG_2\text{->} [g_2,d_2,h_2]))$

Note the contrast between these results and theorems 5.7 and 5.8. Both pairs of results involve double square diagrams, but the former treat weak left adjoints while the latter treat weak right adjoints. The main reason for this is that an arbitrary graph transition system must include the results of all span morphisms applied to productions, hence arrows $g : \mathcal{TS}*(GG) \to TT$ will always exist; whereas an arbitrary graph computation grammar need not include all possible transitions so that one can at best hope for arrows $e : TT \to \mathcal{CG}*(DG)$. Allied to this is that in a graph transition system, the source span of a span morphism could be forgotten in mapping a production name to its interpretation (which is a span), blocking the construction of a functor $\mathcal{Prod}*$ that maps each transition back to the used production; whereas in a computation grammar, both source and target spans are always retained in the interpretation (which is a double square diagram), allowing the construction of $\mathcal{Prod}*$. On the other hand, graph transition systems feature span morphism composition unavailable in a computation grammar.

9 The Category of Computation Systems

In this section we apply the concept of computation grammar to obtain a notion that captures the normal sequential application of productions to the current computational state. To do this we must compose derivation steps corner to corner. The basic techniques for this date back to the notion of R-related composition in Ehrig (1979).

Definition 9.1 Let $((L_1 \leftarrow [K_1] \to R_1) \text{ -}[g_1,d_1,j_1]\text{->} (G \leftarrow [D_1] \to J))$ and $((L_2 \leftarrow [K_2] \to R_2) \text{ -}[j_2,d_2,h_2]\text{->} (J \leftarrow [D_2] \to H))$ be two semiconcrete DP* derivation steps meeting on a corner (see Fig. 13).

$$\begin{tikzcd}
L_1 \arrow[dr, "g_1"'] & & {[K_1]} \arrow[ll, "l_1"'] \arrow[rr, "r_1"] \arrow[dr, "d_1"'] & & R_1 \arrow[dr, "j_1"'] & & L_2 \arrow[dl, "j_2"] & & {[K_2]} \arrow[ll, "l_2"'] \arrow[rr, "r_2"] \arrow[dl, "d_2"] & & R_2 \arrow[dl, "h_2"] \\
& G & & {[D_1]} \arrow[ll, "l_1*"] \arrow[rr, "r_1*"'] & & J & & {[D_2]} \arrow[ll, "l_2*"] \arrow[rr, "r_2*"'] & & H &
\end{tikzcd}$$

Fig. 13

We will construct a single semiconcrete production ($L \leftarrow [K] \rightarrow R$), such that there is a single semiconcrete DP* derivation step (($L \leftarrow [K] \rightarrow R$) -[$g$,$d$,$h$]-› ($G \leftarrow [D] \rightarrow H$)) deriving the graph H from G as in Fig. 14. The process is illustrated in the “St. Edward’s Crown” diagram of Fig. 15.

$$\begin{tikzcd}
L \arrow[d, "g"'] & {[K]} \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "d"] & R \arrow[d, "h"] \\
G & {[D]} \arrow[l, "l*"] \arrow[r, "r*"'] & H
\end{tikzcd}$$

Fig. 14

Pick some concrete representatives K_1, D_1, K_2, D_2 of $[K_1]$, $[D_1]$, $[K_2]$, $[D_2]$ to work with. Let $l^{**} : D \rightarrow D_1$, $r^{**} : D \rightarrow D_2$ be a pullback of $r_1{}^* : D_1 \rightarrow J \leftarrow D_2 : l_2{}^*$, and let $l^* = l_1{}^* \circ l^{**} : D \rightarrow G$ and $r^* = r_2{}^* \circ r^{**} : D \rightarrow H$.

Let L_0 be the “graph completion” of $G - l^*(D)$ i.e.

$$E_{L_0} = E_G - E_{l^*(D)}$$
$$V_{L_0} = (V_G - V_{l^*(D)}) \cup s(E_{L_0}) \cup t(E_{L_0})$$

and let $K_L \subseteq D$ be $l^{*-1}(L_0)$ which is just the node set $l^{*-1}(V_{L_0})$ as $E_{L_0} \cap E_{l^*(D)}$ is empty, with $k_{l^*} : K_L \rightarrow L_0$. Similarly we construct R_0, $D \supseteq K_R = r^{*-1}(R_0)$ and $k_{r^*} : K_R \rightarrow R_0$. If vertex and edge labels are in use, they are inherited in the obvious way. Let $K_{LR} = K_L \cap K_R$, with $K_{LR} \subseteq K_L$, $K_{LR} \subseteq K_R$ the inclusions. Let $K = K_L \cup K_R$, with inclusions k_L, k_R that make $K_{LR}K_RKK_L$ into a pushout. Form the pushout of $k_{l^*} : K_L \rightarrow L_0$ and $k_L : K_L \subseteq K$, i.e. $l : K \rightarrow L$ and $L_0 \subseteq L$, choosing as result the natural pushout object $L \subseteq G$

$$
\begin{tikzcd}
 & & & K_{LR} \arrow[dl] \arrow[dr] & & & \\
L_0 \arrow[dr, "\subseteq"'] \arrow[ddr, bend right, "\supseteq"'] & & K_L \arrow[ll, "k_{l*}"'] \arrow[dr, "k_L"] \arrow[ddl, bend right=80, "l^*|_{K_L}"'] & & K_R \arrow[dl, "k_R"'] \arrow[rr, "k_{r*}"] \arrow[ddr, bend left=80, "r^*|_{K_R}"] & & R_0 \arrow[dl, "\supseteq"] \arrow[ddl, bend left, "\subseteq"] \\
 & L \arrow[dd, "\supseteq"] & & K \arrow[ll, "l = l^*|_K"] \arrow[rr, "r = r^*|_K"'] \arrow[dd, "\subseteq"] & & R \arrow[dd, "\subseteq"] & \\
 & & & & & & \\
 & G & & D \arrow[ll, "l^*"'] \arrow[rr, "r^*"] & & H &
\end{tikzcd}
$$

Fig. 15

singled out by the unique factorisation properties of the pushout, such that the inclusions indicated on the left of Fig. 15 hold. Likewise form the pushout of $k_{r*} : K_R \to R_0$ and $k_R : K_R \subseteq K$, i.e. $r : K \to R$ and $R_0 \subseteq R$, using $R \subseteq H$.

Clearly ($L \leftarrow K : l$, $r : K \to R$) is a concrete general production, so taking K up to arbitrary isomorphism, ($L \leftarrow [K] \to R$) is a semiconcrete production, and taking also D up to arbitrary isomorphism, (($L \leftarrow [K] \to R$) -[$\supseteq$,$\subseteq$,$\subseteq$]-› ($G \leftarrow [D] \to H$)) is clearly a DP* derivation step, because all squares constructed (with the single possible exception of DD_1D_2J) have at least two parallel sides injective and have the cofinal arrow pairs jointly surjective, hence are both pushouts and pullbacks.

We use the notation (($L \leftarrow [K] \to R$) -g,d,h-› ($G \leftarrow [D] \to H$)) = (($L_1 \leftarrow [K_1] \to R_1$) -[$g_1$,$d_1$,$j_1$]-› ($G \leftarrow [D_1] \to J$)) ® (($L_2 \leftarrow [K_2] \to R_2$) -[$j_2$,$d_2$,$h_2$]-› ($J \leftarrow [D_2] \to H$)), or more briefly [$g$,$d$,$h$] = [$g_1$,$d_1$,$j_1$] ® [$j_2$,$d_2$,$h_2$] to express this construction which we call semiconcrete R-composition, and where we have reverted once more to the notation of Fig. 14. (End of definition 9.1.)

For what concerns the rest of this paper an alternative definition of semiconcrete R-composition which yielded (($G \leftarrow [D] \to H$) -[id,id,id]-› ($G \leftarrow [D] \to H$)) would actually serve quite well. The more complex construction we gave pays more attention to the way that different bits of ($G \leftarrow [D] \to H$) arise from the actions of the two original rules and is superior for this reason. Thus vertices and edges of L correspond to equivalence classes of vertices and edges of L_1 (under ${g_1}^{-1}$) and of vertices and edges of L_2 (under ${j_2}^{-1}$, where the equivalence class does not map under j_2 into a vertex or edge created (and not merely preserved) just beforehand by j_1); similarly vertices and edges of R correspond to equivalence classes of vertices and edges of R_2 (under ${h_2}^{-1}$) and of vertices and edges of R_1 (under ${j_1}^{-1}$, where the equivalence class does not map into a vertex or edge deleted (and not preserved) just afterwards by j_2). K is evidently the smallest naturally arising graph such that L and R interface together properly.

An important point to note, is that since everything that needs to be concrete (i.e. L and R and their inclusions into G and H) was constructed from specific subgraphs of G and H which are invariant when D is taken up to isomorphism, and that the horizontal composition of semiconcrete spans is associative, the whole construction composes associatively.

Definition 9.2 Abstract R-composition is defined in the obvious way and written $((\langle L\rangle \leftarrow [K] \rightarrow \langle R\rangle)$ -$[g,d,h]$-› $(\langle G\rangle \leftarrow [D] \rightarrow \langle H\rangle)) = ((\langle L_1\rangle \leftarrow [K_1] \rightarrow \langle R_1\rangle)$ -$[g_1,d_1,j_1]$-› $(\langle G\rangle \leftarrow [D_1] \rightarrow \langle J\rangle))$ ® $((\langle L_2\rangle \leftarrow [K_2] \rightarrow \langle R_2\rangle)$ -$[j_2,d_2,h_2]$-› $(\langle J\rangle \leftarrow [D_2] \rightarrow \langle H\rangle))$, or more briefly $[g,d,h] = [g_1,d_1,j_1]$ ® $[j_2,d_2,h_2]$ where no confusion will arise.

Now we can define computation systems as computation grammars enhanced with R-composition as an additional algebraic operation.

Definition 9.3 A concrete graph computation system is a quintuple $(TG, \overline{G}, P, \pi,$ ®$)$ where $(TG, \overline{G}, P, \pi)$ is a concrete computation grammar, and ® : $P \times P \nrightarrow P$ satisfies

(1) If $\pi(p) = ((L_1 \leftarrow [K_1] \rightarrow R_1)$ -$[g_1,d_1,j_1]$-› $(G \leftarrow [D_1] \rightarrow J))$ and $\pi(q) = ((L_2 \leftarrow [K_2] \rightarrow R_2)$ -$[j_2,d_2,h_2]$-› $(J \leftarrow [D_2] \rightarrow H))$ then p®q is defined and $\pi(p$®$q) = \pi(p)$®$\pi(q)$

(2) If $\pi(p) = ((L \leftarrow [K] \rightarrow R)$ -$[g,d,h]$-› $(G \leftarrow [D] \rightarrow H))$ then P contains a p_G with $\pi(p_G) = ((G \leftarrow [G] \rightarrow G)$ -[id,id,id]-› $(G \leftarrow [G] \rightarrow G))$, an identity for G with $\pi(p_G$®$p) = \pi(p)$, and a similar p_H for H with $\pi(p$®$p_H) = \pi(p)$

(3) ® is associative

Definition 9.4 An abstract graph computation system is a quintuple $(\langle TG\rangle, \langle\overline{G}\rangle, P, \pi,$ ®$)$ where $(\langle TG\rangle, \langle\overline{G}\rangle, P, \pi)$ is an abstract computation grammar, and ® : $P \times P \nrightarrow P$ satisfies

(1) If $\pi(p) = ((\langle L_1\rangle \leftarrow [K_1] \rightarrow \langle R_1\rangle)$ -$[g_1,d_1,j_1]$-› $(\langle G\rangle \leftarrow [D_1] \rightarrow \langle J\rangle))$ and $\pi(q) = ((\langle L_2\rangle \leftarrow [K_2] \rightarrow \langle R_2\rangle)$ -$[j_2,d_2,h_2]$-› $(\langle J\rangle \leftarrow [D_2] \rightarrow \langle H\rangle))$ then p®q is defined and $\pi(p$®$q) = \pi(p)$®$\pi(q)$

(2) If $\pi(p) = ((\langle L\rangle \leftarrow [K] \rightarrow \langle R\rangle)$ -$[g,d,h]$-› $(\langle G\rangle \leftarrow [D] \rightarrow \langle H\rangle))$ then P contains a $p_{\langle G\rangle}$ with $\pi(p_{\langle G\rangle}) = ((\langle G\rangle \leftarrow [G] \rightarrow \langle G\rangle)$ -[id,id,id]-› $(\langle G\rangle \leftarrow [G] \rightarrow \langle G\rangle))$, an identity for $\langle G\rangle$ with $\pi(p_{\langle G\rangle}$®$p) = \pi(p)$, and a similar $p_{\langle H\rangle}$ for $\langle H\rangle$ with $\pi(p$®$p_{\langle H\rangle}) = \pi(p)$

(3) ® is associative

Definition 9.5 The category $\mathcal{GraCS}$* of concrete graph computation systems has as objects concrete graph computation systems, and as morphisms $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0,$ ®$_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2,$ ®$_2)$, where $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f)$ is a morphism between the underlying concrete computation grammars, such that identities map to identities, and for each (p, q) pair defined for ®$_0$, we have $f(p$®$_0q) = f(p)$®$_2f(q)$.

Definition 9.6 The category $\langle \mathcal{GraCS*} \rangle$ of abstract graph computation systems has as objects abstract graph computation systems, and as morphisms $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f) : (\langle TG_0\rangle, \langle\overline{G_0}\rangle, P_0, \pi_0, ®_0) \rightarrow (\langle TG_2\rangle, \langle\overline{G_2}\rangle, P_2, \pi_2, ®_2)$, where $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f)$ is a morphism between the underlying abstract computation grammars, such that identities map to identities, and for each (p, q) pair defined for $®_0$, we have $f(p®_0q) = f(p)®_2f(q)$.

As in section 7 there are forgetful functors $\mathcal{W}* : \mathcal{GraCS}* \rightarrow \mathcal{ComGra}$ and $\langle \mathcal{W}* \rangle : \langle \mathcal{GraCS}* \rangle \rightarrow \langle \mathcal{ComGra} \rangle$ which just ignore ®. We now give the construction that will provide a left adjoint functor to $\mathcal{W}*$ (and one for $\langle \mathcal{W}* \rangle$).

Definition 9.7 Let $CG = (TG, \overline{G}, P, \pi)$ be a concrete computation grammar. Then the concrete graph computation system $GCS* = (TG, \overline{G}, PP, \pi\pi, ®_{PP})$ is given by constructing PPP, $\pi\pi\pi$ and $®_{PPP}$ as the smallest classes such that:

(1) $(p, [g,d,h])$ is in PPP for each p in P and $[g,d,h]$ such that $\pi(p) = ((L \leftarrow [K] \rightarrow R)$ -$[g,d,h]$-› $(G \leftarrow [D] \rightarrow H))$ and $\pi\pi\pi((p, [g,d,h])) = ((L \leftarrow [K] \rightarrow R)$ -$[g,d,h]$-› $(G \leftarrow [D] \rightarrow H))$

(2) If $\pi\pi\pi((p, [g,d,h])) = ((L \leftarrow [K] \rightarrow R)$ -$[g,d,h]$-› $(G \leftarrow [D] \rightarrow H))$ then PPP contains an identity for G $(p_G, [\mathrm{id,id,id}])$ such that $\pi\pi\pi((p_G, [\mathrm{id,id,id}])) = ((G \leftarrow [G] \rightarrow G)$ -[id,id,id]-› $(G \leftarrow [G] \rightarrow G))$ and $\pi\pi\pi((p_G, [\mathrm{id,id,id}])®_{PPP}(p, [g,d,h])) = \pi\pi\pi((p, [g,d,h]))$, and a similar $(p_H, [\mathrm{id,id,id}])$ identity for H

(3) If $(p, [g_1,d_1,j_1])$ is in PPP with $\pi\pi\pi((p, [g_1,d_1,j_1])) = ((L_1 \leftarrow [K_1] \rightarrow R_1)$ -$[g_1,d_1,j_1]$-› $(G \leftarrow [D_1] \rightarrow J))$ and $(q, [j_2,d_2,h_2])$ is in PPP with $\pi\pi\pi((q, [j_2,d_2,h_2])) = ((L_2 \leftarrow [K_2] \rightarrow R_2)$ -$[j_2,d_2,h_2]$-› $(J \leftarrow [D_2] \rightarrow H))$, then $((p, [g_1,d_1,j_1])®_{PPP}(q, [j_2,d_2,h_2]))$ is in PPP with $\pi\pi\pi(((p, [g_1,d_1,j_1])®_{PPP}(q, [j_2,d_2,h_2]))) = [g_1,d_1,j_1]®_{PPP}[j_2,d_2,h_2]$

And then letting PP, $\pi\pi$ and $®_{PP}$ be given by taking PPP, $\pi\pi\pi$ and $®_{PPP}$ modulo the associative law $((A®_{PPP}B)®_{PPP}C) = (A®_{PPP}(B®_{PPP}C))$ and identity laws $(p_G, [\mathrm{id,id,id}])®_{PPP}(p, [g,d,h]) = (p, [g,d,h]) = (p, [g,d,h])®_{PPP}(p_H, [\mathrm{id,id,id}])$.

Definition 9.8 Let $\langle CG\rangle = (\langle TG\rangle, \langle\overline{G}\rangle, P, \pi)$ be an abstract computation grammar. Then the abstract graph computation system $\langle GCS*\rangle = (\langle TG\rangle, \langle\overline{G}\rangle, PPP, \pi\pi\pi, ®_{PPP})$ is given by constructing PPP, $\pi\pi\pi$ and $®_{PPP}$ as the smallest classes such that:

(1) $(p, [g,d,h])$ is in PPP for each p in P and $[g,d,h]$ such that $\pi(p) = ((\langle L\rangle \leftarrow [K] \rightarrow \langle R\rangle)$ -$[g,d,h]$-› $(\langle G\rangle \leftarrow [D] \rightarrow \langle H\rangle))$ and $\pi\pi\pi((p, [g,d,h])) = ((\langle L\rangle \leftarrow [K] \rightarrow \langle R\rangle)$ -$[g,d,h]$-› $(\langle G\rangle \leftarrow [D] \rightarrow \langle H\rangle))$

(2) If $\pi\pi\pi((p, [g,d,h])) = ((\langle L\rangle \leftarrow [K] \rightarrow \langle R\rangle)$ -$[g,d,h]$-› $(\langle G\rangle \leftarrow [D] \rightarrow \langle H\rangle))$ then PPP contains an identity for G $(p_{\langle G\rangle}, [\mathrm{id,id,id}])$ such that $\pi\pi\pi((p_{\langle G\rangle}, [\mathrm{id,id,id}])) = ((\langle G\rangle \leftarrow [G] \rightarrow \langle G\rangle)$ -[id,id,id]-› $(\langle G\rangle \leftarrow [G] \rightarrow \langle G\rangle))$ and $\pi\pi\pi((p_{\langle G\rangle}, [\mathrm{id,id,id}])®_{PPP}(p, [g,d,h])) = \pi\pi\pi((p, [g,d,h]))$, and a similar $(p_{\langle H\rangle}, [\mathrm{id,id,id}])$ identity for H

(3) If $(p, [g_1,d_1,j_1])$ is in PPP with $\pi\pi\pi((p, [g_1,d_1,j_1])) = ((\langle L_1\rangle \leftarrow [K_1] \rightarrow \langle R_1\rangle)$ -$[g_1,d_1,j_1]$-› $(\langle G\rangle \leftarrow [D_1] \rightarrow \langle J\rangle))$ and $(q, [j_2,d_2,h_2])$ is in PPP with $\pi\pi\pi((q, [j_2,d_2,h_2])) = ((\langle L_2\rangle \leftarrow [K_2] \rightarrow \langle R_2\rangle)$ -$[j_2,d_2,h_2]$-› $(\langle J\rangle \leftarrow [D_2] \rightarrow \langle H\rangle))$, then $((p,$

$[g_1,d_1,j_1])®_{PPP}(q,\ [j_2,d_2,h_2]))$ is in PPP with $\pi\pi\pi(((p,\ [g_1,d_1,j_1])®_{PPP}(q,\ [j_2,d_2,h_2]))) = [g_1,d_1,j_1]®_{PPP}[j_2,d_2,h_2]$

And then letting PP, $\pi\pi$ and $®_{PP}$ be given by taking PPP, $\pi\pi\pi$ and $®_{PPP}$ modulo the associative law $((A®_{PPP}B)®_{PPP}C) = (A®_{PPP}(B®_{PPP}C))$ and identity laws $(p_{\langle G\rangle}, [id,id,id])®_{PPP}(p, [g,d,h]) = (p, [g,d,h]) = (p, [g,d,h])®_{PPP}(p_{\langle H\rangle}, [id,id,id])$.

Theorem 9.9 The forgetful functor $\mathcal{W}^* : \mathcal{GraCS}^* \rightarrow \mathcal{ComGra}$ has a left adjoint $\mathcal{CS}^* : \mathcal{ComGra} \rightarrow \mathcal{GraCS}^*$ where $\mathcal{CS}^*(\mathsf{CG}) = \mathsf{GCS}^*$ is given by definition 9.7 for objects, with the unique extension for arrows.

Proof. Define the universal arrow $w : \mathsf{CG} \rightarrow \mathcal{W}^*(\mathcal{CS}^*(\mathsf{CG}) = \mathsf{GCS}^*)$ for a concrete computation grammar $\mathsf{CG} = (TG, \overline{G}, P, \pi)$ by

$$w : \mathsf{CG} \rightarrow \mathcal{W}^*(\mathsf{GCS}^*) = (TG \leftarrow [TG] \rightarrow TG, in)$$

where $(TG \leftarrow [TG] \rightarrow TG)$ is an identity in $\mathcal{Gr}\langle\mathcal{Sp}\rangle$, and $in(p) = (p, \pi(p))$. Given a morphism $f : \mathsf{CG} \rightarrow \mathcal{W}^*(\mathsf{TT}) = (TG_0 \leftarrow [TG_1] \rightarrow TG_2, f_P)$ into the forgetful image of some computation system TT, we must show there is a unique morphism $e : \mathcal{CS}^*(\mathsf{CG}) \rightarrow \mathsf{TT}$ such that $f = \mathcal{W}^*(e) \circ w$.

We define $e = (TG_0 \leftarrow [TG_1] \rightarrow TG_2, e_P) : \mathsf{GCS}^* \rightarrow \mathsf{TT}$ such that for all p in P $e_P((p, [g_0,d_0,h_0])) = (f_P(p), \mathrm{cod}([g_0,d_0,h_0] \text{-}{\scriptstyle TG_0TG_1TG_2}\text{->}\ [g_2,d_2,h_2])$ extending by ® to the whole of GCS^*. It is clear that the required properties hold. ☺

Theorem 9.10 The forgetful functor $\langle\mathcal{W}^*\rangle : \langle\mathcal{GraCS}^*\rangle \rightarrow \langle\mathcal{ComGra}\rangle$ has a left adjoint $\langle\mathcal{CS}^*\rangle : \langle\mathcal{ComGra}\rangle \rightarrow \langle\mathcal{GraCS}^*\rangle$ where $\langle\mathcal{CS}^*\rangle(\langle\mathsf{CG}\rangle) = \langle\mathsf{GCS}^*\rangle$ is given given by definition 9.8 for objects, with the unique extension for arrows.

Finally definitions 9.5 and 9.6, which describe how the extra structure in a graph computation system behaves under morphisms, make the following expected statements evident.

Theorem 9.11 The obvious projection $\mathcal{P}_{\mathcal{GraCS}^*} : \mathcal{GraCS}^* \rightarrow \mathcal{Gr}\text{-}\mathcal{Sp}$ is an opfibration, where all arrows $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0, ®_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2, ®_2)$ of $\mathcal{GraCS}^*$ such that f is an iso in $\mathcal{Class}$ are are opcartesian.

Theorem 9.12 The obvious projection $\langle\mathcal{P}_{\mathcal{GraCS}^*}\rangle : \langle\mathcal{GraCS}^*\rangle \rightarrow \langle\mathcal{Gr}\text{-}\mathcal{Sp}\rangle$ is an opfibration, where all arrows $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f) : (\langle TG_0\rangle, \langle\overline{G}_0\rangle, P_0, \pi_0, ®_0) \rightarrow (\langle TG_2\rangle, \langle\overline{G}_2\rangle, P_2, \pi_2, ®_2)$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

The astute reader will by now have a question burning in his mind. If derivation steps are combined corner to corner, why is the production used not taken to be completely abstract viz. $[L] \leftarrow [K] \rightarrow [R]$, since the graphs L and R are not used in the composition? One answer is that we surely have enough different notions of span to contend with by now without introducing yet another one — which is certainly true. A better answer lies among matters not discussed in this paper, namely the construction of event structures for the derivations generated by graph grammars. In this, it is important to keep track of which node of a production rule graph gives rise to which node in the execution graph. Without taking L and R up to identity or standard isomorphism, one loses the ability to do this consistently. See Corradini et al. (1996a,b).

10 The Category of Computation Sequence Systems

In this section we take a different view of the activity of combining derivation steps corner to corner, namely we simply consider the sequences of transitions generated in their own right. This allows us to easily take note of infinite derivation sequences too, which cannot be included in the computation system framework without also complying with some notion of convergence, a burden we wish to avoid in this paper there being moreover quite a variety of such notions that one could choose to consider. Of course sequences support a natural associative notion of concatenation of pairs (when the first of the pair is a finite sequence) so we can define closed computation sequence systems as the ones already containing all concatenations. The computation sequence system framework also makes it natural to take into account the start graph of a grammar which is finally given a role to play via the subcollection of sequences that have it as starting point. This gives us the pointed computation sequence systems. (A similar thing could be done fairly naturally also in the preceding section.)

Definition 10.1 A concrete computation sequence is a finite or infinite sequence of semiconcrete typed derivation steps $[s_0, s_1, s_2, \ldots]$ where each s_i is of the form

$$((L_i \leftarrow [K_i] \rightarrow R_i) \text{ -}[g_i,d_i,h_i]\text{-›} (G_i \leftarrow [D_i] \rightarrow G_{i+1}))$$

(and the notation makes clear that the ending graph of one step is the starting graph of the next step).

Definition 10.2 An abstract computation sequence is a finite or infinite sequence of abstract typed derivation steps $[s_0, s_1, s_2, \ldots]$ where each s_i is of the form

$$((\langle L_i\rangle \leftarrow [K_i] \rightarrow \langle R_i\rangle) \text{ -}[g_i,d_i,h_i]\text{-›} (\langle G_i\rangle \leftarrow [D_i] \rightarrow \langle G_{i+1}\rangle))$$

Definition 10.3 A concrete computation sequence system is a quintuple $(TG, \overline{G}, P, \pi, S)$ where $(TG, \overline{G}, P, \pi)$ is a concrete computation grammar, and S is a class of sequences of production names eg. $s = [p_0, p_1, p_2, \ldots]$ such that the associated sequence $\pi(s) = [\pi(p_0), \pi(p_1), \pi(p_2), \ldots]$ is a concrete computation sequence built out of the steps of $\pi(P)$.

If $s_1 = [p_0, p_1, p_2, \ldots, p_n]$ is finite, and $s_2 = [q_0, q_1, q_2, \ldots]$, and they are such that whenever both are non-empty if

$$\pi(p_n) = ((L_n \leftarrow [K_n] \rightarrow R_n) \text{ -}[g_n,d_n,h_n]\text{-›} (G_n \leftarrow [D_n] \rightarrow G_{n+1}))$$

and

$$\pi(q_0) = ((L_0 \leftarrow [K_0] \rightarrow R_0) \text{ -}[g_0,d_0,h_0]\text{-›} (H_0 \leftarrow [U_0] \rightarrow H_1))$$

then $G_{n+1} = H_0$, then the concatenation of s_1 and s_2, $s_1 \mathbin{;} s_2$, is well defined and

$$s_1 \mathbin{;} s_2 = \begin{cases} [q_0, q_1, q_2, \ldots] & \text{if } s_1 = [], \\ [p_0, p_1, p_2, \ldots, p_n] & \text{if } s_2 = [], \\ [p_0, p_1, p_2, \ldots, p_n, q_0, q_1, q_2, \ldots] & \text{otherwise} \end{cases}$$

If S is closed under concatenation of all pairs of sequences where $;$ is defined, then we say that the concrete computation sequence system is closed. If the starting graph of the first step (if any) of all sequences in S is $\overline{G}$ then we say that the concrete computation sequence system is pointed.

Note that there is an alternative formulation of closed and pointed concrete computation sequence systems in which the class S is itself a category. The objects of the category are concrete graphs, and arrows are the sequences of production names as above. Arrow composition is the concatenation operation $\mathbin{;}$ above, and we now need a separate formal identity for each graph instead of the single empty sequence composing with all other sequences. To make the picture complete, we need a codomain graph for infinite sequences. This is the formal object $\top$, the overspecified graph, which is the domain for only its identity arrow. We leave as an easy exercise for the reader the easy modifications of the results below that reflect this change of perspective.

Definition 10.4 An abstract computation sequence system is defined as in definition 10.3 but using an abstract computation grammar and abstract computation sequences.

Definition 10.5 The category $\mathcal{GraCSS}*$ of concrete graph computation sequence systems has as objects concrete graph computation sequence systems, and as morphisms $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0, S_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2, S_2)$, where $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f)$ is a morphism between the underlying concrete computation grammars, such that for each sequence $[(p_0)_0, (p_1)_0, (p_2)_0, \ldots]$ in S_0, the induced sequence $[(p_0)_2, (p_1)_2, (p_2)_2, \ldots]$, in which each element $(p_i)_0$ is mapped according to the concrete computation grammar morphism, $(p_i)_2 = f((p_i)_0)$, is an element of S_2 (i.e. satisfies the ending/starting compatibility condition). The category $\mathcal{GraCSS}*$ contains the subcategories $\mathcal{GraCCSS}*$ of closed concrete graph computation sequence systems and $\mathcal{GraPCSS}*$ of pointed concrete graph computation sequence systems.

Definition 10.6 The category $\langle\mathcal{GraCSS}*\rangle$ of abstract graph computation sequence systems has as objects abstract graph computation sequence systems, and as morphisms $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f) : (\langle TG_0\rangle, \langle\overline{G}_0\rangle, P_0, \pi_0, S_0) \rightarrow (\langle TG_2\rangle, \langle\overline{G}_2\rangle, P_2, \pi_2, S_2)$, where $(\langle TG_0\rangle \leftarrow [TG_1] \rightarrow \langle TG_2\rangle, f)$ is a morphism between the underlying abstract computation grammars, such that for each sequence $[(p_0)_0, (p_1)_0, (p_2)_0, \ldots]$ in S_0, the induced sequence $[(p_0)_2, (p_1)_2, (p_2)_2, \ldots]$, is an element of S_2. The category $\langle\mathcal{GraCSS}*\rangle$ contains the subcategories $\langle\mathcal{GraCCSS}*\rangle$ of closed abstract graph computation sequence systems and $\langle\mathcal{GraPCSS}*\rangle$ of pointed abstract graph computation sequence systems.

As usual there are forgetful functors $\mathcal{X}* : \mathcal{GraCSS}* \rightarrow \mathcal{ComGra}$ and $\langle\mathcal{X}*\rangle : \langle\mathcal{GraCSS}*\rangle \rightarrow \langle\mathcal{ComGra}\rangle$ which just ignore the S component. We now give the construction that will provide a right adjoint functor to $\mathcal{X}*$ (and one for $\langle\mathcal{X}*\rangle$). Obviously there will be analogues for $\mathcal{GraCCSS}*$, $\mathcal{GraPCSS}*$, $\langle\mathcal{GraCCSS}*\rangle$, $\langle\mathcal{GraPCSS}*\rangle$.

Definition 10.7 Let $\mathsf{CG} = (TG, \overline{G}, P, \pi)$ be a concrete computation grammar. Then the concrete graph computation sequence system $\mathsf{GCCSS}* = (TG, \overline{G}, P, \pi, S)$ is given by constructing S as the class of all sequences of production names of P which map under π to concrete computation sequences. It is obvious that $\mathsf{GCCSS}*$ is closed. The pointed concrete graph computation sequence system $\mathsf{GPCSS}* = (TG, \overline{G}, P, \pi, \overline{S})$ is given by restricting S to $\overline{S}$, the subclass of sequences which start with the concrete start graph $\overline{G}$.

Definition 10.8 Let $\langle\mathsf{CG}\rangle = (\langle TG\rangle, \langle\overline{G}\rangle, P, \pi)$ be an abstract computation grammar. Then the abstract graph computation sequence system $\langle\mathsf{GCCSS}*\rangle = (\langle TG\rangle, \langle\overline{G}\rangle, P, \pi, S)$ is given by constructing S as the class of all sequences of production names of P

which map under π to abstract computation sequences. Obviously $\langle \mathsf{GCCSS*} \rangle$ is closed. The pointed abstract graph computation sequence system $\langle \mathsf{GPCSS*} \rangle = (\langle TG \rangle, \langle \overline{G} \rangle, P, \pi, \overline{S})$ is given by restricting S to $\overline{S}$, the subclass of sequences which start with the abstract start graph $\langle \overline{G} \rangle$.

Theorem 10.9 The forgetful functor $\mathcal{X}* : \mathcal{GraCSS}* \rightarrow \mathcal{ComGra}$ has a right adjoint $\mathcal{CSS}* : \mathcal{ComGra} \rightarrow \mathcal{GraCSS}*$ where $\mathcal{CSS}*(\mathsf{CG}) = \mathsf{GCCSS*}$ is given by definition 10.7 for objects, with the unique extension for arrows.

Proof. Define the universal arrow $x : \mathcal{X}*(\mathcal{CSS}*(\mathsf{CG}) = \mathsf{GCCSS*}) \rightarrow \mathsf{CG}$ for a concrete computation grammar $\mathsf{CG} = (TG, \overline{G}, P, \pi)$ by

$$x : \mathcal{X}*(\mathsf{GCCSS*}) \ \rightarrow \mathsf{CG} = (TG \leftarrow [TG] \rightarrow TG, \mathrm{id})$$

where $(TG \leftarrow [TG] \rightarrow TG)$ is an identity in $\mathcal{Gr}\langle\mathcal{Sp}\rangle$, and id is the identity on P. Given a morphism $f : \mathcal{X}*(\mathsf{TT}) \rightarrow \mathsf{CG} = (TG_0 \leftarrow [TG_1] \rightarrow TG_2, f_P)$ from the forgetful image of some concrete computation sequence system TT, we must show there is a unique morphism $e : \mathsf{TT} \rightarrow \mathcal{CSS}*(\mathsf{CG})$ such that $f = x \circ \mathcal{X}*(e)$.

We define $e = (TG_0 \leftarrow [TG_1] \rightarrow TG_2, e_P) : \mathsf{TT} \rightarrow \mathsf{GCCSS*}$ as just the extension of f to the sequences in the last component of TT in the obvious manner. This generates an inclusion into the last component of $\mathsf{GCCSS*}$. It is clear that the required properties hold. ☺

Theorem 10.10 The forgetful functor $\langle \mathcal{X}* \rangle : \langle \mathcal{GraCSS}* \rangle \rightarrow \langle \mathcal{ComGra} \rangle$ has a right adjoint $\langle \mathcal{CSS}* \rangle : \langle \mathcal{ComGra} \rangle \rightarrow \langle \mathcal{GraCSS}* \rangle$ where $\langle \mathcal{CSS}* \rangle(\langle \mathsf{CG} \rangle) = \langle \mathsf{GCCSS*} \rangle$ is given by definition 10.8 for objects, with the unique extension for arrows.

Finally the opfibration properties.

Theorem 10.11 The obvious projection $\mathcal{P}_{\mathcal{GraCSS}*} : \mathcal{GraCSS}* \rightarrow \mathcal{Gr\text{-}Sp}$ is an opfibration, where all arrows $(TG_0 \leftarrow [TG_1] \rightarrow TG_2, f) : (TG_0, \overline{G}_0, P_0, \pi_0, S_0) \rightarrow (TG_2, \overline{G}_2, P_2, \pi_2, S_2)$ of $\mathcal{GraCSS}*$ such that f is an iso in $\mathcal{Class}$ are are opcartesian.

Theorem 10.12 The obvious projection $\langle \mathcal{P}_{\mathcal{GraCSS}*} \rangle : \langle \mathcal{GraCSS}* \rangle \rightarrow \langle \mathcal{Gr\text{-}Sp} \rangle$ is an opfibration, where all arrows $(\langle TG_0 \rangle \leftarrow [TG_1] \rightarrow \langle TG_2 \rangle, f) : (\langle TG_0 \rangle, \langle \overline{G}_0 \rangle, P_0, \pi_0, S_0) \rightarrow (\langle TG_2 \rangle, \langle \overline{G}_2 \rangle, P_2, \pi_2, S_2)$ such that f is an iso in $\mathcal{Class}$ are opcartesian.

11 Conclusions

In the preceding sections we set up a general framework for reasoning about how typed diagrams of different kinds behave under change of typing, where the change of typing is controlled by a semiconcrete or abstract span, and the crucial observation turned out to be that the behaviour of diagrams is exactly captured by an opfibration over the type change category. Different graph rewriting phenomena were then reduced almost entirely to choosing the right kind of diagram to work with to describe the situation in question. In each case we were able to define a notion of grammar, and then via an appropriate free construction, to get the notion of system that captured some particular way of manipulating the grammar data algebraically. In many cases one operation proved to be incompatible with another due to the differing latitude in permitted isomorphisms for graphs of the diagrams involved, afforded by the differing manipulations being contemplated and the complete generality of the productions we considered. This was in turn necessitated by the necessity of dealing with systems "in the large", where knowledge of how one entity transforms into another forced careful consideration of the isomorphisms. Perhaps the most vivid example of this was the contrast between graph grammars and transition systems, and derivation grammars and derivation systems. The former allowed vertical composition but precluded horizontal composition because the result of a pullback was not specific enough, while the latter allowed horizontal composition but precluded vertical composition because the generality of the isomorphisms in the middle blocked a consistent law of vertical composition. We regard this clarification as a significant contribution of this paper, as well as the more general observation that "in the large", the flexibilty of taking one or other object in a diagram up to this or that kind of isomorphism, is irrevocably tied to the kind of manipulations that one intends to perform on it.

In fact in our opinion the graph grammar/transition system construction is the most problematic of all the constructions introduced in this paper as it inherently forces cardinality problems regarding the size of the collection of production names which are not forced in other constructions. Thus a transition system *must* have a proper class of production names, as it does not seem easy to make a natural but small choice of span morphisms under which the productions have to be closed (unless one simply hypothesises such a choice as we did for partial graph transition systems). One might make a similar criticism of the free construction that builds a computation grammar from a derivation grammar, but there it is just the free construction that causes this, not the very definition of computation grammar (which can easily have a small set of derivation step names), unlike the transition system definition. In any case, one can regard the derivation grammar/computation grammar construction as in a sense less important to the main concerns of the paper than the graph grammar/transition system construction.

We note that the situations where the adjunctions we discovered are of the weak kind[1], correspond precisely to those in which the constructions involved provoke the cardinality considerations we just discussed, i.e. the constructions do not preserve smallness. This is no surprise. These are precisely the constructions in which "additional data" is being imported into the grammar via the algebraic operation concerned. In both cases we mentioned the additional data is essentially the data of a span morphism (in the transition system case it is a genuine span morphism while in the computation grammar case it is a collection of arrows that turns a semiconcrete span into a double square), and since our change of type opfibrations are general enough to admit arrows to isomorphic objects, we immediately have to deal with proper classes. The other constructions merely manipulated the data already in a span or other object, and so did not suffer any explosion.

As we mentioned before at one point, for the grammar constructions, we concentrated on morphisms which were opcartesian (or nearly so for the grammars) as far as change of type was concerned. This was mainly to aid comparison with earlier work. There is

1. In fact we produced instances of every possible combination of weak/strong, left/right adjoints.

no reason at all why we could not have considered more extensively the more general case of change of base in which the change of base was followed by an in-fibre morpism; Fig. 9.(b) vs. Fig. 9.(a) so to speak. Little of the above would have changed and we leave the more general case as an exercise for the reader. However we point out one additional aspect of this observation. Since the constructions of Corradini et al. (1996b), Heckel et al. (1997), Ehrig et al. (1996), were split opfibrations in all but name, allowing the extra in-fibre morphisms there (specifically in-fibre isomorphisms only) could have recovered most of what we obtained in our more general framework here[1]. The point being that in such a situation, the precise graph in the middle of a change of base span is of little importance, and whether or not a specific choice of pullback for the change of base construction is made gets hidden by the final in-fibre arrow. Of course the price of this is the nondeterminism we had to contend with here.

12 Acknowledgements

Most of the work reported here was done while the first author was on leave at the Computer Science Department of the University of Pisa. The hospitality of that department, and the financial support of the British Council, the Consiglio Nazionale delle Ricerche, and the Staff Exchange Scheme of the European Commission's ERASMUS programme are gratefully acknowledged. The authors would also like to thank the referee Reiko Heckel, for many useful comments on earlier versions of this paper.

References

Banach R. (1993); A Fibration Semantics for Extended Term Graph Rewriting. *in*: Term Graph Rewriting: Theory and Practice. Sleep, Plasmeijer, van Eekelen (eds.), 91-100, John Wiley.

Banach R. (1994); Term Graph Rewriting and Garbage Collection Using Opfibrations. Theor. Comp. Sci. **131** 29-94.

Banach R. (1995); DPO Rewriting and Abstract Semantics via Opfibrations. *in*: Proc. SEGRAGRA-95, Corradini, Montanari (eds.), ENTCS **2**, Elsevier, http://www.elsevier.nl/locate/entcs/volume2.html. Extended version: University of Manchester Dept. of Computer Science Technical Report UMCS 96-8-1, http://www.cs.man.ac.uk/csonly/cstechrep/Abstracts/UMCS-96-8-1.html

Banach R., Corradini A. (1996); An Opfibration Account of Typed DPO and DPB Graph Transformation: Injective Productions. *Submitted to* Theor. Comp. Sci.

Barr M., Wells C. (1990); Category Theory for Computing Science. (1 ed.), Prentice-Hall.

Bauderon M. (1995); Parallel Rewriting of Graphs Through the Pullback Approach. *in*: Proc. SEGRAGRA-95, Corradini, Montanari (eds.), ENTCS **2**, Elsevier, http://www.elsevier.nl/locate/entcs/volume2.html.

Corradini A., Ehrig H., Löwe M., Montanari U., Rossi F. (1994a); Note on Standard Representations of Graphs and Graph Derivations. *in*: Graph Transformations in Computer Science. Schneider, Ehrig (eds.), L.N.C.S. **776** 104-118.

1. A similar effect is achieved via the Kleisli construction in Heckel et al. (1996).

Corradini A., Ehrig H., Löwe M., Montanari U., Rossi F. (1994b); Abstract Graph Derivations in the Double pushout Approach. *in*: Graph Transformations in Computer Science. Schneider, Ehrig (eds.), L.N.C.S. **776** 86-103.

Corradini A., Ehrig H., Löwe M., Montanari U., Rossi F. (1994c); An Event Structure Semantics for Safe Graph Grammars. *in*: Proc. I.F.I.P. Working Conference PROCOMET-94, Olderog (ed.), 423-446.

Corradini A., Ehrig H., Löwe M., Montanari U., Rossi F. (1996a); An Event Structure Semantics for Graph Grammars with Parallel Productions. *in*: Proc. Fifth International Workshop on Graph Grammars and their Application to Computer Science 1994, Williamsburg U.S.A., Cuny, Ehrig, Engels, Rozenberg (eds.), L.N.C.S. **1073** 240-256.

Corradini A., Ehrig H., Löwe M., Montanari U., Padberg J. (1996b); Typed Graph Grammars and their Adjunction with Categories of Derivations. *in*: Proc. Fifth International Workshop on Graph Grammars and their Application to Computer Science 1994, Williamsburg U.S.A., Cuny, Ehrig, Engels, Rozenberg (eds.), L.N.C.S. **1073** 56-74.

Ehrig H. (1979); Introduction to the Algebraic Theory of Graph Grammars (A survey). *in*: L.N.C.S. **73**, 1-69, Springer, Berlin.

Ehrig H. (1987); A Tutorial Introduction to the Algebraic Approach of Graph Grammars. *in*: Third International Workshop on Graph Grammars, L.N.C.S. **291** 3-14, Springer, Berlin.

Ehrig H., Heckel R., Padberg J., Taentzer G., Wolter U., Corradini A., Engels G. (1996); Synchronization of Views and Loose Semantics of Typed Graph Productions. Lecture and abstract, Dagstuhl Seminar 9637 on Graph Transformations in Computer Science.

Gray J.W. (1966); Fibered and Cofibered Categories. *in*: Proc. La Jolla Conference on Categorical Algebra, Eilenberg et al. (eds.), 21-83, Springer, Berlin.

Grothendieck A. (1961); Catégories Fibrés et Descente, Seminaire de Géométrie Algébrique de l'Institut des Hautes Études Scientifiques Paris, *also in*: L.N.M. **224**, 145-194, Springer, Berlin.

Heckel R., Corradini A., Ehrig H., Löwe M. (1996); Horizontal and Vertical Structuring of Typed Graph Transformation Systems. Math. Struct. Comp. Sci. *to appear.*

Heckel R., Corradini A., Ehrig H., Padberg J., Wolter U. (1997); Coalgebraic Loose Semantics as Specification Frame for Graph Transformation Systems. Draft. Applied Categorical Structures special issue on the Dagstuhl Seminar 9637 on Graph Transformations in Computer Science, *to be submitted.*

Krishnan V. (1981); An Introduction to Category Theory. North-Holland.

Löwe M. (1991); Extended Algebraic Graph Transformation. Doctoral thesis, Fachbereich Informatik, Technical University Berlin.

Löwe M., Ehrig H. (1991); Algebraic Approach to Graph Transformation Based on Single Pushout Derivations. *in*: Proc. Graph-Theoretic Concepts in Computer Science, Mohring ed., L.N.C.S. **484** 338-353.

Ribeiro L. (1996); Parallel Composition and Unfolding Semantics of Graph Grammars. Thesis Dr.-Ing. Technical University of Berlin.