Graph Transformation via Abstract Diagrams

R. Banach[a], A. Corradini[b]
[a]Computer Science Dept., Manchester University, Manchester, M13 9PL, U.K.
[b]Dipartimento di Informatica, Università di Pisa, Corso Italia 40, Pisa, Italy.
banach@cs.man.ac.uk , andrea@di.unipi.it

1 Introduction

This paper gives a very terse (for lack of space) account of typed double pushout (DPO) graph transformation as discussed in [6], using abstract diagram techniques which lend a satisfying degree of abstraction to the theory (unlike concrete diagrams as used in traditional approaches). See eg. [7, 8, 9] for the untyped version of graph transformation. We show that graph grammars, graph transition systems, and graph derivation systems are opfibrations over abstract type change, and that transition and derivation systems are generated by left adjoints to the evident forgetful functors. For a less breakneck presentation see the full version [1]. Section 2 sketches abstract diagram theory, the foundation of the paper. Section 3 presents the main technical constructions for the sequel. Section 4 points out how the preceding applies to DPO graph transformation. Sections 5, 6 and 7 formalise respectively graph grammars, transition systems, and derivation systems. Section 8 concludes. To save space many standard concepts are used as needed without introduction; also there are no proofs.

2 Concrete and Abstract Diagrams

Let $\underline{\mu}$ be a directed graph, $\mathbf{C}$ be a category, and $\underline{\gamma} : \underline{\mu} \to U\mathbf{C}$ be a graph morphism from $\underline{\mu}$ to the underlying graph of $\mathbf{C}$. Then $\underline{\gamma}$ is a concrete diagram of shape $\underline{\mu}$ in $\mathbf{C}$. Let μ be the path category of $\underline{\mu}$. Then the standard free construction extends $\underline{\gamma} : \underline{\mu} \to U\mathbf{C}$ to a functor $\gamma : \mu \to \mathbf{C}$. If in addition, for all pairs of objects m_0, m_1 in μ, for all paths $(e_1, \ldots, e_k)$ from m_0 to m_1 in μ, if the internal composition $(\gamma(e_k) \circ \ldots \circ \gamma(e_1))$ in $\mathbf{C}$ always yields the same arrow $f : \gamma(m_0) \to \gamma(m_1)$, then the diagram is a commuting concrete diagram of shape μ. Henceforth we will only consider commuting diagrams. A morphism of concrete diagrams is just a natural transformation $n : \gamma \to \delta$.

An abstract diagram $\mathbf{D}$ is a subcategory of the functor category $[\mu,\mathbf{C}]$ such that for any two objects γ and δ in $\mathbf{D}$, there is at least one arrow $n : \gamma \to \delta$ in $\mathbf{D}$, and all the $\mathbf{C}$ arrows that make up such an n are isomorphisms. An abstract diagram $\mathbf{D}$ is maximal iff **(** γ is an object of $\mathbf{D}$ and $n : \gamma \to \delta$ is a morphism such that all the $\mathbf{C}$ arrows that make up n are isomorphisms **)** $\Rightarrow$ **(** n is a morphism in $\mathbf{D}$ **)**.

A morphism $c : \mathbf{D}_0 \to \mathbf{D}_1$ of abstract diagrams is a functor from $\mathbf{D}_0$ to $\mathbf{D}_1$. A morphism $c : \mathbf{D}_0 \to \mathbf{D}_1$ is mediated by a family Ξ of arrows of $\mathbf{C}$ iff there is a function $\chi : (\mathrm{Vert}(\mu) \times \mathrm{Obj}(\mathbf{D}_0)) \to \mathrm{Arr}(\mathbf{C})$, with range Ξ, mapping pairs (m_0, γ) to arrows of $\mathbf{C}$ such that: (1) for any fixed γ of $\mathbf{D}_0$, the $\chi(m_0, \gamma)$ form a concrete diagram morphism $\gamma \to c(\gamma)$; (2) for any fixed $n : \gamma \to \delta$ of $\mathbf{D}_0$, the collection of the $\chi(m_0, \gamma)$ and $\chi(m_0, \delta)$ forms a morphism from $n : \gamma \to \delta$ to $c(n : \gamma \to \delta) : c(\gamma) \to c(\delta)$, naturally.

The set *Kind* = {id, std, iso} will label shape vertices according to the kind of isomorphisms permitted above them between objects of an abstract diagram. Thus $\mathbf{D}$ of kinded shape μ conforms to its kind iff for each vertex m_0 in μ: (1) $kind(m_0) = \mathsf{id} \Leftrightarrow$ for each $n : \gamma \to \delta$ in $\mathbf{D}$, the component of n at m_0 is $\mathrm{id}_{\gamma(m_0)}$; (2) $kind(m_0) = \mathsf{std} \Leftrightarrow$ for

each $n : \gamma \to \delta$ in D, the component of n at m_0 is a standard isomorphism (see Corradini et al. (1994a,b)); (and $kind(m_0)$ = iso does not restrict the isomorphism at m_0). Kindedness interacts with maximality in the obvious way. Sensitivity to kinds enables a detailed correspondence between our theory and other approaches to be set up, but because of lack of space, these aspects will be neglected here, and all kinds will be iso in this paper.

A concrete interface-diagram category is a category whose objects and arrows are concrete diagrams of suitable shapes, such that the objects arise as (source and target) subdiagrams of the arrow diagrams, and all the expected laws of a category hold w.r.t. some specific notion of composition of arrows (which might in fact be given in terms of some quite complicated manipulations of the constituent arrow diagrams). The idea is a variant of internal category theory, with a pushout-like construction called pasting (in which two commuting diagrams are glued together along a well defined interface — provided the result is still a commuting diagram) playing the role of pullbacks in the latter. For lack of space we suppress the formal definition, outlining an example. Thus the shape $\bullet_0 \to \bullet_1$ is a possible shape for arrow diagrams, with $\bullet$ a suitable shape for object diagrams, the source and target injections of $\bullet$ into $\bullet_0 \to \bullet_1$ being obvious, these in turn inducing the requisite object subdiagrams of the arrow diagrams. Equally obvious is the composition manipulation for these arrow diagrams: it is just normal arrow composition. Note that our tiny example is canonical in the sense that "anything" that satisfies the laws of a category must be capable of being projected down to $\bullet$ and $\bullet_0 \to \bullet_1$.

The same idea carries over to abstract diagrams. An abstract interface-diagram category simply uses abstract diagrams instead of concrete ones, where the abstract notion of pasting becomes the collection of pastings of all possible concrete pastings of the constituent concrete diagrams. All the categories we will utilise below for formalising graph transformation phenomena are abstract interface-diagram categories.

One way of getting (say a concrete) interface-diagram category is to consider a diagram morphism $n : \gamma \to \delta$. If we consider the arrows of γ and δ and the arrows of n to be of equal status, we can regard $n : \gamma \to \delta$ as a bigger diagram whose shape is two copies of the shape of γ (or δ) linked by fresh edges, one fresh edge for each arrow of the natural transformation n. These arrow diagrams support object subdiagrams which are the original γ and δ; the law of composition is relatively clear. This construction gives the interface-diagram category generated from the shape of γ; our tiny example above was an instance of it, generated from the object shape $\bullet$. The same idea carries over to mediated morphisms of abstract diagrams, except that the arrow abstract diagrams generated, must be taken as the maximal abstract diagrams containing the mediating family of concrete arrows between the collections of concrete diagrams in question, a technically unproblematic closure operation. These techniques smoothly port constructions at the concrete diagram level to the abstract diagram level, a phenomenon we exploit fully in this work.

3 Spans and the Opfibration $[P] : [Gr \downarrow Gr\text{-}Sp] \to [Gr\text{-}Sp]$

We fix the shape digraph for spans to be $\bullet_1 \leftarrow \blacklozenge \to \bullet_2$ which we call $\underline{\eta}$. An abstract span is an abstract diagram of the form $[A] \leftarrow [B] \to [C]$, i.e. an abstract diagram of shape η in the category of graphs and graph morphisms Gr, with all kinds iso.

The category $[\mathsf{Sp}]$ is the interface-diagram category generated from η by the trick just mentioned, i.e. $[\mathsf{Sp}]$ is the category of abstract span morphisms. A morphism of $[\mathsf{Sp}]$

is $([A] \leftarrow [B] \rightarrow [C])$ -$[a,b,c]$-› $([A'] \leftarrow [B'] \rightarrow [C'])$ where -$[a,b,c]$-› is a notation for three concrete graph morphisms representing the abstract span morphism.

The category [*Gr-Sp*] is the interface-diagram category whose arrows are abstract spans, i.e. an arrow is $([A] \leftarrow [B] \rightarrow [C]) : [A] \rightarrow [C]$. Composition is given by the relatively obvious pullback construction that takes $[A] \leftarrow [B] \rightarrow [C]$ and $[C] \leftarrow [D] \rightarrow [E]$ and forms the arrow $([A] \leftarrow [M] \rightarrow [E]) : [A] \rightarrow [E]$, where M is some concrete graph which is the pullback of $B \rightarrow C \leftarrow D$.

The two methods of composition involving spans can be brought together in one abstract double interface-diagram category [*D-Gr-Sp*]. Its double cells are effectively the morphisms of [*Sp*]. Vertical composition $*_v$ is the composition of [*Sp*]. Horizontal composition $*_h$ is essentially the composition of [*Gr-Sp*] (acting at the two levels of a [*Sp*] morphism).

The structure described is already powerful enough to formalise DPO rewriting when one specialises to [*Sp*] morphisms wherein the two squares are pushouts. However we go further to include types and type change. A typed graph over a (type) graph TG is simply an object of the comma category (*Gr*↓TG), i.e. $G \rightarrow TG$. Various works [6, 10, 11] address, with various techniques, the issue of relating graphs typed over different graphs. We exploit abstract diagrams and an opfibrational framework to gain the greatest generality.

The category [*Gr*↓*Gr-Sp*] is a horizontal subcategory of [*D-Gr-Sp*], such that two extra properties hold for every arrow $(([X] \leftarrow [Y] \rightarrow [Z])$ -$[a,b,c]$-› $([A] \leftarrow [B] \rightarrow [C])) : ([X] \rightarrow [A]) \rightarrow ([Z] \rightarrow [C])$ namely that: (1) the left square $XYBA$ of each concrete diagram in the arrow is a pullback; (2) the right arrow $Y \rightarrow Z$ of the source abstract span of each concrete diagram in the arrow is an isomorphism. We write $(([X] \leftarrow [Y] = [Z])$ -$[a,b,c]$-› $([A] \leftarrow [B] \rightarrow [C]))$ to signify that both properties hold.

Theorem 3.1 The projection $[P] : [Gr{\downarrow}Gr\text{-}Sp] \rightarrow [Gr\text{-}Sp]$ that takes

$$(([X] \leftarrow [Y] = [Z])\text{ -}[a,b,c]\text{-›}\ ([A] \leftarrow [B] \rightarrow [C])) : ([X] \rightarrow [A]) \rightarrow ([Z] \rightarrow [C])$$

to $([A] \leftarrow [B] \rightarrow [C]) : [A] \rightarrow [C]$ is a split opfibration, where all arrows of [*Gr*↓*Gr-Sp*] are opcartesian and belong to the splitting.

It turns out that when we have an opfibration, diagrams in the fibres of the subject category acquire morphisms mediated by families of opcartesian arrows, i.e. there is a covariant functor from the base to diagram morphisms. This allows us to introduce a triple category that will play a key role in the rest of the paper. The triple category adds a perpendicular dimension (i.e. the change of base via [*Gr-Sp*]) to the double category [*D-Gr-Sp*].

The triple category [*D-Gr-Sp*↓*Gr-Sp*] has as triple cells abstract diagrams of the shape in Fig. 1 (middle upper part suppressed). These may be combined using vertical, horizontal and perpendicular composition, $*_v$, $*_h$, $*_p$ respectively. Vertical composition is typed [*D-Gr-Sp*] vertical composition; horizontal composition is typed [*D-Gr-Sp*] horizontal composition; perpendicular composition is derived from [*Gr*↓*Gr-Sp*] horizontal composition by applying it to all the vertices of a [*D-Gr-Sp*] double cell. The key result is the following.

Theorem 3.2 The projection $[P_{D\text{-}Gr\text{-}Sp}] : [D\text{-}Gr\text{-}Sp{\downarrow}Gr\text{-}Sp] \rightarrow [Gr\text{-}Sp]$ that takes

$$(([([X_0] \leftarrow [Y_0] \rightarrow [Z_0])\text{ -}[x_0,y_0,z_0]\text{-›}\ ([X_0\tilde{}] \leftarrow [Y_0\tilde{}] \rightarrow [Z_0\tilde{}])) \rightarrow [A])\text{ -}[ABC]\text{=›}$$
$$(([([X_2] \leftarrow [Y_2] \rightarrow [Z_2])\text{ -}[x_2,y_2,z_2]\text{-›}\ ([X_2\tilde{}] \leftarrow [Y_2\tilde{}] \rightarrow [Z_2\tilde{}])) \rightarrow [C])$$

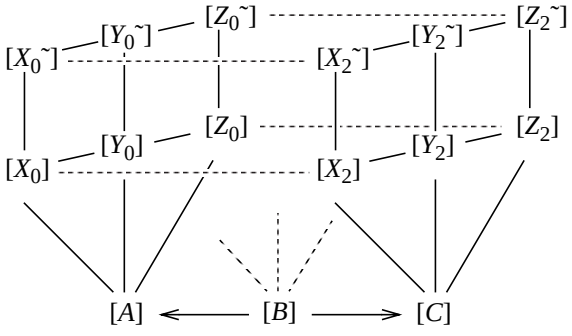

Fig. 1

to $([A] \leftarrow [B] \rightarrow [C]) : [A] \rightarrow [C]$ is a split opfibration, where all triple cells of [*D-Gr-Sp*↓*Gr-Sp*] are opcartesian and belong to the splitting.

4 Abstract Graph Rewriting

The preceding material has been very abstract, but provides the right framework for describing DPO graph transformation in an abstract manner. For this we need to recall that the normal presentation of the DPO approach requires that the span forming a production rule consists of monic arrows, and that the two squares forming the application of a rule to a redex are both pushouts. Luckily this combination of properties of a two-square diagram is preserved by the action of opcartesian arrows of [*Gr*↓*Gr-Sp*] by a relatively standard series of lemmas, which quickly leads to the identification of a triple subcategory of [*D-Gr-Sp*↓*Gr-Sp*] which we call [*D-Gr-MSp*↓*Gr-Sp*], in which the spans $[X_i^{(\sim)}] \leftarrow [Y_i^{(\sim)}] \rightarrow [Z_i^{(\sim)}]$ of Fig. 1 are monic, and a further triple subcategory of [*D-Gr-MSp*↓*Gr-Sp*] called [*D-Gr-MSp*-DPO↓*Gr-Sp*], such that the horizontal-vertical squares illustrated are pushouts. This latter forms the precise technical vehicle for the formalisations of the rest of this paper.

5 The Category of Typed Graph Grammars

We consider abstract graphs $[G]$ typed over an abstract type graph $[TG]$, or putting it another way, abstract graph morphisms $[G] \rightarrow [TG]$. Changing the type is done by means of an arbitrary abstract span eg. $([TG_0] \leftarrow [TG_1] \rightarrow [TG_2]) : [TG_0] \rightarrow [TG_2]$. The preceding sections show us how the various entities involved in graph transformation, transform under such a change of typing at the abstract level. The category of abstract typed graph grammars [*GraGra*] is as follows.

Objects: $([TG], [\overline{G}], P, \pi)$ where:
- $[TG]$ is an abstract type graph,
- $[\overline{G}]$ is an abstract start graph typed over $[TG]$,
  - i.e. an abstract graph morphism $[\overline{G}] \rightarrow [TG]$,
- P is a set of production names,
- $\pi : P \rightarrow HArr([D\text{-}Gr\text{-}MSp{\downarrow}Gr\text{-}Sp])$ is a map from P to
  - horizontal arrows of [*D-Gr-MSp*↓*Gr-Sp*],
  - i.e. abstract typed monic spans, typed over $[TG]$.

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 $[\mathit{Gr}\text{-}\mathit{Sp}]$,
$([TG_0] \leftarrow [TG_1] \rightarrow [TG_2]) : [TG_0] \rightarrow [TG_2]$,
Secondly: an arrow of $[\mathit{Gr}{\downarrow}\mathit{Gr}\text{-}\mathit{Sp}]$,
$(([\overline{G}_0] \leftarrow [\overline{G}_2] = [\overline{G}_2])$ -$[g_0,g_1,g_2]$-› $([TG_0] \leftarrow [TG_1] \rightarrow [TG_2])) :$
$([\overline{G}_0] \rightarrow [TG_0]) \rightarrow ([\overline{G}_2] \rightarrow [TG_2])$,
which projects under $[P]$ to the first arrow,
Thirdly: an arrow of Set, $f : P_0 \rightarrow P_2$ i.e. a map,
Fourthly: for all $p \in P_0$ a horizontal-perpendicular double cell of
$[\mathit{D}\text{-}\mathit{Gr}\text{-}\mathit{MSp}{\downarrow}\mathit{Gr}\text{-}\mathit{Sp}]$,
$([\pi_0(p)] \rightarrow [TG_0])$ -$[TG_0 TG_1 TG_2]$=› $([\pi_2(f(p))] \rightarrow [TG_2])$
which projects under $[P_{\mathit{D}\text{-}\mathit{Gr}\text{-}\mathit{Sp}}]$ to the first arrow.

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 $[TG_{2'}]$ arises from
the composition of $[\mathit{Gr}\text{-}\mathit{Sp}]$.

Identities: $([TG] \leftarrow [TG] \rightarrow [TG], \mathrm{id}_P) : ([TG], [\overline{G}], P, \pi) \rightarrow ([TG], [\overline{G}], P, \pi)$,
where the arrows in $[TG] \leftarrow [TG] \rightarrow [TG]$ are all isomorphisms.

Theorem 5.1 The projection $[P_{\mathit{GraGra}}] : [\mathit{GraGra}] \rightarrow [\mathit{Gr}\text{-}\mathit{Sp}]$ such that

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

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 Set are opcartesian.

Note that this opfibration is not split due to the absence of any canonical isomorphisms in Set. A choice of standard isomorphisms for Set would yield a splitting.

6 The Category of Transition Systems

Graph transition systems are enriched graph grammars which include all the result spans of direct derivation steps by their productions and such that the set of production names supports a partial action / by $\mathit{HVDCell}([\mathit{D}\text{-}\mathit{Gr}\text{-}\mathit{MSp}\text{-DPO}{\downarrow}\mathit{Gr}\text{-}\mathit{Sp}])$, the (horizontal-vertical) double cells of $[\mathit{D}\text{-}\mathit{Gr}\text{-}\mathit{MSp}\text{-DPO}{\downarrow}\mathit{Gr}\text{-}\mathit{Sp}]$. For notational compactness, we will write these double cells in future using a notation like $[d_1,d_2,d_3]$, refering to their alternative interpretation as abstract span morphisms, this in turn legitimising the use of dom and cod in the next definition.

An abstract typed graph transition system is a quintuple $([TG], [\overline{G}], P, \pi, /)$ where $([TG], [\overline{G}], P, \pi)$ is an abstract typed graph grammar, and $/ : P \times \mathit{HVDCell}([\mathit{D}\text{-}\mathit{Gr}\text{-}\mathit{MSp}\text{-DPO}{\downarrow}\mathit{Gr}\text{-}\mathit{Sp}]) \nrightarrow P$ 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) $(p/[d_1,d_2,d_3])/[d'_1,d'_2,d'_3] = p/[d'_1.d_1,d'_2.d_2,d'_3.d_3]$.

The category $[\mathit{GraTS}]$ of abstract graph transition systems has as objects abstract 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 abstract graph grammar, and such that for each p_0 in P_0 and each $[d_{01},d_{02},d_{03}]$ 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 $[f(p_0/[d_{01},d_{02},d_{03}])] = [f(p_0)/[d_{21},d_{22},d_{23}]]$.

Obviously there is a forgetful functor $[U] : [GraTS] \to [GraGra]$ which just ignores $/$. We now give the construction that will provide a left adjoint functor to $[U]$.

Let $GG = ([TG], [\overline{G}], P, \pi)$ be an abstract graph grammar. Then the abstract 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 a horizontal-vertical double cell of $[D\text{-}Gr\text{-}MSp\text{-}\mathrm{DPO}{\downarrow}Gr\text{-}Sp]$ with $\mathrm{dom}([t_1,t_2,t_3]) = [\pi(p)]\}$; (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 $(p, [d_1.t_1,d_2.t_2,d_3.t_3])$, and thus $[\pi\pi((p, [t_1,t_2,t_3])/[d_1,d_2,d_3])] = \mathrm{cod}([d_1.t_1, d_2.t_2,d_3.t_3])$.

Theorem 6.1 The forgetful functor $[U] : [GraTS] \to [GraGra]$ has a left adjoint $[TS] : [GraGra] \to [GraTS]$ where the functor $[TS](GG) = GTS$ is given above for objects, and is given for arrows by:

> $[TS](([TG_0] \leftarrow [TG_1] \to [TG_2], f) : GG_0 \to GG_2)$ is the unique morphism $([TG_0] \leftarrow [TG_1] \to [TG_2], g_P) : GTS_0 \to GTS_2$ in $[GraTS]$ such that for all p in P_0, $g_P((p, [\mathrm{id}_{[\pi(p)]}])) = (f(p), [\mathrm{id}_{[\pi(f(p))]}])$

Theorem 6.2 The obvious projection $[P_{GraTS}] : [GraTS] \to [Gr\text{-}Sp]$ is an opfibration, where all arrows $([TG_0] \leftarrow [TG_1] \to [TG_2], f) : ([TG_0], [\overline{G}_0], P_0, \pi_0, /_0) \to ([TG_2], [\overline{G}_2], P_2, \pi_2, /_2)$ such that f is an iso in Set are opcartesian.

7 The Category of Derivation Systems

Derivation systems are transition systems enriched with an operation $;$ of horizontal composition on production names, inherited from the corresponding property of $[D\text{-}Gr\text{-}MSp{\downarrow}Gr\text{-}Sp]$.

An abstract graph derivation system is a sextuple $([TG], [\overline{G}], P, \pi, /, ;)$ where $([TG], [\overline{G}], P, \pi, /)$ is an abstract graph transition system, 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 $B \to C \leftarrow D$; (2) if $\pi(p) = [A] \leftarrow [B] \to [C]$ then P contains a $p_{[A]}$ with $\pi(p_{[A]}) = [A] \leftarrow [A] \to [A]$, an identity name such that $p_{[A]};p = p$, and also a similar identity name $p_{[C]}$ with $\pi(p_{[C]}) = [C] \leftarrow [C] \to [C]$ and $p;p_{[C]} = p$; (3) $;$ is associative; (4) if both $(p;q)/([s_1,s_2,s_3]*_h[t_1,t_2,t_3])$ and $(p/[s_1,s_2,s_3]);(q/[t_1,t_2,t_3])$ are defined then $(p;q)/([s_1,s_2,s_3]*_h[t_1,t_2,t_3]) = (p/[s_1,s_2,s_3]);(q/[t_1,t_2,t_3])$.

The category $[GraDS]$ of abstract graph derivation systems has as objects abstract graph derivation systems, and as morphisms $([TG_0] \leftarrow [TG_1] \to [TG_2], f) : ([TG_0], [\overline{G}_0], P_0, \pi_0, /_0, ;_0) \to ([TG_2], [\overline{G}_2], P_2, \pi_2, /_2, ;_2)$, where $([TG_0] \leftarrow [TG_1] \to [TG_2], f)$ is a morphism between the underlying abstract transition systems, 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 $f(p;_0q) = f(p);_2f(q)$.

There is a forgetful functor $[V] : [GraDS] \to [GraTS]$ which just ignores $;$. We now give the construction that will provide a left adjoint functor to $[V]$.

Let $[GTS] = ([TG], [\overline{G}], P, \pi, /)$ be an abstract graph transition system. Then the abstract graph transition system $[GDS] = ([TG], [\overline{G}], PP, \pi\pi, /_{PP}, ;_{PP})$ is given by firstly, constructing PPP, $\pi\pi\pi$, $/_{PPP}$ and $;_{PPP}$ as the smallest sets satisfying the following properties: (1) (p) is in PPP for p in P, and $\pi\pi\pi((p)) = \pi(p)$; and whenever $p/[d_1,d_2,d_3]$ is defined, $(p)/_{PPP}[d_1,d_2,d_3] = (p/[d_1,d_2,d_3])$, (and $\pi\pi\pi((p)/_{PPP}[d_1,d_2,d_3]) = \mathrm{cod}([d_1,d_2,d_3])$); (2) $(p_{[A]})$ and $(p_{[C]})$ are in PPP for each p in P with $\pi(p) = [A] \leftarrow [B] \to [C]$, and $\pi\pi\pi((p_{[A]})) = [A] \leftarrow [A] \to [A]$ and $\pi\pi\pi((p_{[C]})) = [C] \leftarrow [C] \to [C]$ both

identity abstract spans, and $\pi\pi\pi((p_{[A]})\mathbin{;}_{PPP}(p)) = \pi\pi\pi((p)) = \pi\pi\pi((p)\mathbin{;}_{PPP}(p_{[C]}))$; and whenever $[A] = \mathrm{dom}([d])$, $(p_{[A]})/_{PPP}[d,d,d] = (p_{\mathrm{cod}(d)})$, (and $\pi\pi\pi((p_{\mathrm{cod}(d)})) = \mathrm{cod}([d,d,d])$), and similarly for $(p_{[C]})$; (3) (p, q) is in PPP for p, q in PPP such that $\pi\pi\pi(p) = [A] \leftarrow [B] \rightarrow [C]$ and $\pi\pi\pi(p) = [C] \leftarrow [D] \rightarrow [E]$; $(p, q) = p\mathbin{;}_{PPP}q$, and $\pi\pi\pi((p, q))$ is given via the local pullback of $\pi\pi\pi(p)$ and $\pi\pi\pi(q)$; and whenever $p/_{PPP}[s_1,s_2,s_3]$, $q/_{PPP}[t_1,t_2,t_3]$ and $[s_1,s_2,s_3]*_h[t_1,t_2,t_3]$ are defined, $(p, q)/_{PPP}[s_1,s_2,s_3]*_h[t_1,t_2,t_3]$ is defined and $\pi\pi\pi((p, q)/_{PPP}[s_1,s_2,s_3]*_h[t_1,t_2,t_3]) = \mathrm{cod}([s_1,s_2,s_3]*_h[t_1,t_2,t_3])$. And then secondly, letting PP, $\pi\pi$, $/_{PP}$ and $\mathbin{;}_{PP}$ be given by taking PPP, $\pi\pi\pi$, $/_{PPP}$ and $\mathbin{;}_{PPP}$ modulo the composition law $(p/[d_1,d_2,d_3])/_{PPP}[d'_1,d'_2,d'_3] = p/_{PPP}[d'_1.d_1,d'_2.d_2, d'_3.d_3]$ and identity law $p/_{PPP}\mathrm{id}_{\pi(p)} = p$, and the associative law $((\mathrm{A}\mathbin{;}_{PPP}\mathrm{B})\mathbin{;}_{PPP}\mathrm{C}) = (\mathrm{A}\mathbin{;}_{PPP}(\mathrm{B}\mathbin{;}_{PPP}\mathrm{C}))$ and identity laws $(p_{[A]})\mathbin{;}_{PPP}(p) = (p) = (p)\mathbin{;}_{PPP}(p_{[C]})$.

Note that our constructions are based on properties of $[\mathcal{D}\text{-}\mathcal{G}\mathit{r}\text{-}\mathcal{M}\mathit{Sp}{\downarrow}\mathcal{G}\mathit{r}\text{-}\mathcal{S}\mathit{p}]$, so the interchange laws of $[\mathcal{G}\mathit{raDS}]$ in the span-transition lemma of [6], derive directly from those of the subcategory $[\mathcal{D}\text{-}\mathcal{G}\mathit{r}\text{-}\mathcal{M}\mathit{Sp}\text{-}\mathrm{DPO}{\downarrow}\mathcal{G}\mathit{r}\text{-}\mathcal{S}\mathit{p}]$.

Theorem 7.1 The forgetful functor $[\mathcal{V}] : [\mathcal{G}\mathit{raDS}] \rightarrow [\mathcal{G}\mathit{raTS}]$ has a left adjoint $[\mathcal{DS}] : [\mathcal{G}\mathit{raTS}] \rightarrow [\mathcal{G}\mathit{raDS}]$ where $[\mathcal{DS}](\mathcal{GTS}) = \mathcal{GDS}$ is given above for objects, with the unique extension for arrows.

Theorem 7.2 The obvious projection $[\mathcal{P}_{\mathcal{G}\mathit{raDS}}] : [\mathcal{G}\mathit{raDS}] \rightarrow [\mathcal{G}\mathit{r}\text{-}\mathcal{S}\mathit{p}]$ is an opfibration, where all arrows $([TG_0] \leftarrow [TG_1] \rightarrow [TG_2], f) : ([TG_0], [\overline{G}_0], P_0, \pi_0, /_0, \mathbin{;}_0) \rightarrow ([TG_2], [\overline{G}_2], P_2, \pi_2, /_2, \mathbin{;}_2)$ such that f is an iso in $\mathcal{S}\mathit{et}$ are opcartesian.

8 Conclusions

The preceding sections presented the "in the large" version of the theory of graph grammars and the associated phenomena of transition systems and derivation systems. The vehicle for this was the abstract diagram in its most abstract incarnation, in which all isomorphisms of graphs and diagrams were permitted. However, there are parts of graph transformation theory which use a finer notion of equivalence than this, in particular [4, 6] which deal with event structure semantics, and where equivalence up to only standard isomorphisms plays a key role. Here we wish to point out that this kind of theory is perfectly accessible using our techniques.

In Section 2 we indicated that the kinds of vertices of shape graphs could be assigned arbitrarily, before we restricted attention exclusively to the kind iso theory. To describe the alternative variants needed, it is essentially enough to restrict judiciously chosen vertices to kind std. This enables the identification of corresponding concrete graphs at those vertices in the manner required. Specifically, the vertices of the graphs X_0, $X_0\tilde{}$, Z_0, $Z_0\tilde{}$, X_2, $X_2\tilde{}$, Z_2, $Z_2\tilde{}$, A, C of Fig. 1 must be of kind std. The requisite theory can be built up by straightforward analogues of Sections 3-7. However the lack of full generality of isomorphisms has three specific consequences compared with the theory set out above.

Firstly, various operations in the theory become nondetereministic, due to the loss of ability to relate concrete diagrams which differ only by nontrivial automorphims and which would otherwise be in the same abstract diagram. Secondly, those opfibrations which we presented as split above, lose the split property. Essentially this is a manifestation of the same phenomenon, the lack of a canonical choice of opcartesian arrow at a critical point of the theory. Thirdly, certain left adjoints in the theory become weak left adjoints, again as a result of an absence of canonical choice, this time of

universal arrow. Provided one is prepared to accept these relatively harmless modifications, the remainder of the theory goes through uneventfully.

Proceeding further, one can forget the internal structure of abstract diagrams, i.e. the morphisms between the concrete diagrams that make up an abstract diagram, to get a version of the theory in terms of equivalence classes of concrete diagrams. From there it is a short step to refashion the results in terms of concrete diagrams in the category of abstract graphs and abstract morphisms, bringing the theory into line with preceding accounts.

The one message that emerges clearly from this work is that in examining questions of abstractness where the subject matter is categorical, functor categories provide the most convincing approach, and treatments involving equivalence classes can be smoothly recovered from them post hoc.

References

1. Banach R., Corradini A. (1999); Abstract Diagrams and an Opfibration Account of Typed Graph Transformation. *Submitted to* T.C.S. *See* `http://www.cs.man.ac.uk/~banach/some.pubs/Abs.Diag.Op.Ty.Gr.Tr.ps.gz`
2. 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.
3. 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.
4. 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.
5. 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.
6. Corradini A., Ehrig H., Löwe M., Montanari U., Padberg J. (1996b); The Category of Typed Graph Grammars and its Adjunctions 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.
7. Corradini A., Montanari U., Rossi F., Ehrig H., Heckel R., Löwe M., (1997); Algebraic Approaches to Graph Transformation Part I: Basic Concepts and Double Pushout Approach. *in*: Handbook of Graph Grammars and Computing by Graph Transformation. Rozenberg (ed.), 163-245.
8. Ehrig H. (1979); Introduction to the Algebraic Theory of Graph Grammars (A survey). *in*: L.N.C.S. **73** 1-69, Springer, Berlin.
9. 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.
10. Heckel R., Corradini A., Ehrig H., Löwe M. (1996); Horizontal and Vertical Structuring of Typed Graph Transformation Systems. Math. Struct. Comp. Sci. **6**, 613-648.
11. Ribeiro L. (1996); Parallel Composition and Unfolding Semantics of Graph Grammars. Thesis Dr.-Ing. Technical University of Berlin.