Math. Struct. in Comp. Science (2004), *vol.* 14, *pp.* 685–714. © 2004 Cambridge University Press
DOI: 10.1017/S096012950400430X Printed in the United Kingdom

Minimisation and minimal realisation in Span(Graph)

ROBERT ROSEBRUGH[†], NICOLETTA SABADINI[‡] and
ROBERT F. C. WALTERS[‡]

[†]*Department of Mathematics and Computer Science, Mount Allison University, Sackville, N. B. E4L 1E6, Canada*

[‡]*Dipartimento di Scienze Chimiche, Fisiche e Matematica, Università dell'Insubria, via Valleggio, 11, Como, Italy*

Received 5 October 2002; revised 1 September 2003

The context of this article is a program studying the bicategory of spans of graphs as an algebra of processes, with applications to concurrency theory. The objective here is to study functorial aspects of reachability, minimisation and minimal realisation. The compositionality of minimisation has application to model-checking.

1. Introduction

In this article we extend the consideration of the bicategory of spans of graphs as an algebra of transition systems that is found in the work of Katis, Sabadini and Walters. They have shown that, with the operations also discussed in this article and certain other minimal constants, a discrete cartesian bicategory results. Moreover, this algebra is expressive of many examples in concurrency theory (Katis *et al.* 1997b; Katis *et al.* 1997c; Katis *et al.* 1998; Katis *et al.* 2000).

The main results here are a functorial minimisation and minimal realisation for labelled graphs. These are compositional and described by idempotent lax monads. We also find a functorial and compositional description of reachability by a colax comonad. The minimisation is obtained using the concept of bisimulation from concurrency theory. We consider a behaviour functor that takes its values in (equivalence classes of) labelled trees (and more generally in forests). Thus, our objective here is the study of functorial relations among minimisation, behaviour and minimal realisation in variants of **Span**(**Graph**).

As an application, we provide a simple model checking algorithm that in many interesting cases avoids the state explosion problem for verification of distributed systems with parallel components (Graf *et al.* 1996). We apply it here to the case of the Dining Philosopher Problem, and note that there is an infinite class of examples with similar properties (Katis *et al.* 2001).

By functorial minimisation and minimal realisation we mean the following general setup. For a pair of input and output objects A and B, which we may also view as interfaces, we require a category of machines from A to B. Classically, these are some transducer model. Here they are spans of graphs (with a restriction on morphisms). These should be viewed as non-deterministic transition systems with left and right labellings specified by the span, that is, the head graph defines states and transitions while the left and right legs of the span specify left and right labellings. Furthermore, the machines should have a composition providing a machine from A to C given one from A to B and one from B to C. Notice that this is distinct from the local composition in the category of machines from A to B. In fact, we expect these two compositions to interact according to the *interchange law* so that we obtain a bicategory of machines. For a classical version, see Rosebrugh *et al.* (1998), where the authors have also considered minimisation and minimal realisation in several bicategories of automata.

For minimisation, a local minimisation functor M is defined on each local machine category (from A to B say). It is an idempotent monad and the algebras (the minimised machines) determine a reflective subcategory. We expect this category to be essentially discrete, and it is so classically and below. Minimisation usually requires a reachability restriction. The minimisation process M will be an identity-on-objects lax functor (bicategory morphism) on the machine bicategory; that is, there will be a comparison from the composite of minimised machines to the minimisation of their composite. M is locally an idempotent monad with discrete algebras and we get a lax monad in the sense of Carboni and Rosebrugh (1991). Thus, the minimised machines determine a locally discrete sub-bicategory of the machines.

There should be a behaviour functor E defined on each local machine category and taking values in a discrete behaviour category. This is classically a category of functions; below we take observational equivalence classes of trees. By minimal realisation we understand the functorial construction of a universal machine realising any behaviour, that is, we expect the minimal realisation process to be locally right adjoint to behaviour (Goguen 1972; Rosebrugh *et al.* 1998). The minimal realisation is denoted N (for Nerode). As with minimisation, behaviour carries the structure of a lax functor, and minimised machines and behaviours are (bi-)equivalent. The situation we have been describing can be summed up in the following diagram:

$$\begin{tikzcd}
& \mathbf{Mach} \arrow[dl, shift right, "M"'] \arrow[dr, shift left, "E"] & \\
\mathbf{MinMach} \arrow[ur, shift right, "I"'] \arrow[rr, shift left, "E'"] & & \mathbf{Beh} \arrow[ul, shift left, "N"] \arrow[ll, shift left, "N'"]
\end{tikzcd}$$

where the bicategory **MinMach** is locally discrete, **Beh** is locally discrete, M and E are locally left adjoint, I is locally an inclusion and E' and N' are an equivalence.

We wish to re-emphasise that the minimisation process described here is locally functorial as well as compositional, and is expressible as a local adjunction. This gives a mathematically precise expression of the meaning of compositional minimal realisation which has been a goal of other workers in model checking (Attie and Emerson 1998; Clarke *et al.* 1989; Graf *et al.* 1996). As was the case in the earlier work (Bloom *et al.* 1996), the minimisation process is essentially one that 'kills the 2-cells', that is the morphisms between machines, so that the bicategory of minimised machines is locally discrete.

We begin in Section 2 with a study of a minimisation for labelled graphs by generalising the notion of bisimulation for transition systems (Arnold 1992). To make the minimisation functorial, we consider 'path lifting' morphisms of labelled graphs and then consider points and reachability. The resulting minimisation process is an idempotent monad. In Section 3 we make the minimisation compositional by defining a bicategory of spans of graphs on which our minimisation is an idempotent lax monad (Carboni and Rosebrugh 1991). In Section 4 we consider the possibility of null edges, labels and branching bisimulation (van Glabbeek and Weijland 1996) using spans of reflexive graphs, and again obtain compositional minimisation results.

In Section 5 we describe the setting for our minimal realisation theory in considerable generality. We include a generalised version of our behaviour functor and show its relations with some toposes. We also show how reachability for spans of (multi-)pointed graphs is described by an idempotent comonad on each hom category. This provides an identity-on-objects comorphism of bicategories that is a colax comonad. We obtain a bicategory of coalgebras that are the spans of reachable graphs. This is similar to results found in Rosebrugh *et al.* (1998).

In Section 6 we consider a definition of behaviour for pointed labelled graphs. The labelled trees that arise in defining the local behaviour functors are those of interest in concurrency theory. The actual values of the local behaviour are bisimilarity classes of trees. We construct a minimal reachable pointed graph realising such a class of trees. Behaviour followed by minimal realisation defines an idempotent monad on each of the hom categories and we again have a lax monad. Our construction can be modified by adding final states to recover the classical minimal realisation theory for finite automata.

In Section 7 we consider the tensor structure and feedback on spans of graphs and their relationship to our minimisation. We describe the resulting compositional model checking algorithm and apply it to the Dining Philosopher Problem. Finally, we extend the behaviour-minimal realisation theory from trees to forests.

This work has been supported by NSERC Canada, Italian MURST and the Australian Research Council. The authors wish to thank the referees for suggestions that have improved this article.

2. Bisimulation and minimisation

In this section we will use the concept of bisimulation, and, in particular, greatest self-bisimulations to describe a minimisation functor on a suitable category of labelled graphs viewed as a slight generalisation of labelled transition systems. We find that minimisation is adjoint to the inclusion of the minimised labelled graphs.

Transition systems are often used for modelling parallel processes.

Definition 2.1. (Arnold 1992) Let A be an alphabet, that is, a finite set. A *transition system labelled by* A is $\mathscr{A} = (S, T, \alpha, \beta, \lambda)$ where S and T are sets of *states* and *transitions*, $\alpha : T \longrightarrow S$ and $\beta : T \longrightarrow S$ define the *source* and *target* of transitions, and the *labelling* $\lambda : T \longrightarrow A$ makes the mapping $\langle \alpha, \lambda, \beta \rangle : T \longrightarrow S \times A \times S$ injective. A transition t with source s and target s' labelled by a is denoted $t : s \xrightarrow{a} s'$.

Arnold is ambiguous about whether the condition on λ is required. It means that there is only one transition with a given label between two states, and this is sometimes expressed by saying that T is a subset of $S \times A \times S$ (see, for example, Joyal *et al.* (1996)).

In the rest of this paper we will be mainly concerned with variants of the category of *(directed) graphs*, **Graph**. This category has as objects the parallel pairs of mappings

$$G : G_1 \overset{s_G}{\underset{t_G}{\rightrightarrows}} G_0$$

that specify the set of edges G_1 and the set of vertices (or nodes) G_0, and the source and target mappings, s_G and t_G. A morphism $f : G \longrightarrow G'$ of graphs is a pair of mappings $(f_1 : G_1 \longrightarrow G'_1, f_0 : G_0 \longrightarrow G'_0)$ that is compatible with source and target, that is, $f_0 s_G = s_{G'} f_1$ and $f_0 t_G = t_{G'} f_1$. To determine notation, we recall that for any object G of **Graph**, the *slice category* **Graph**$/G$ has as objects pairs (G', f) with G' a graph and $f : G' \longrightarrow G$ a morphism of graphs with codomain G. A morphism in **Graph**$/G$ from (G', f) to (G'', g) is a graph morphism $h : G' \longrightarrow G''$ satisfying $gh = f$.

An alphabet A may be viewed as a graph with a single node and an edge for each member of the alphabet. From this point of view, a *graph G labelled by A* is a pair (G, l) where G is a graph and l is a graph morphism from G to A (all nodes of G map to A's single node, edges of G map to members of the alphabet A). That is, we have an object of **Graph**$/A$. A label-preserving morphism of graphs is the same thing as a morphism of **Graph**$/A$. It is useful to bear in mind for the rest of the paper that *a transition system labelled by A is exactly the special case of an object of* **Graph**$/A$ *satisfying the condition on* λ *in Definition 2.1.*

A *homomorphism* of transition systems labelled by an alphabet A is a morphism of **Graph**$/A$ between labelled transition systems. Indeed, the full subcategory of transition systems is reflective in **Graph**$/A$.

More generally, if A is *any* graph, then an object of **Graph**$/A$ is a *graph labelled by A*. Here the label of an edge e of G must be an edge of A whose source and target are compatible under the labelling with the source and target of e. Thus, any object of **Graph**$/A$ may be viewed as a labelled transition system with a varying set of labels that does not necessarily satisfy the condition on λ, and a morphism of **Graph**$/A$ is a label-preserving homomorphism of labelled graphs.

A notion that has received considerable attention in the study of concurrency is (*strong*) *bisimulation.*

Definition 2.2. (Arnold 1992) Let $\mathscr{A}_1 = (S_1, T_1, \alpha_1, \beta_1, \lambda_1)$ and $\mathscr{A}_2 = (S_2, T_2, \alpha_2, \beta_2, \lambda_2)$ be two transition systems labelled by the same alphabet A. A *bisimulation* (Arnold 1992) between $\mathscr{A}_1$ and $\mathscr{A}_2$ is a relation R from S_1 to S_2 such that

(ia) The projection of R on S_1 is onto.
(ib) The projection of R on S_2 is onto.
(iia) For every transition $t_1 : s_1 \xrightarrow{a} s_2$ in $\mathscr{A}_1$ and state s'_1 such that $s_1 R s'_1$ there is a transition $t_2 : s'_1 \xrightarrow{a} s'_2$ with $s_2 R s'_2$ in $\mathscr{A}_2$.
(iib) Similarly for transitions in $\mathscr{A}_2$.

It should be noted that there are several related concepts in the literature and the one just defined is usually called ‘strong bisimulation’. The idea is that any letter that may act on a state of the first transition system can also act on *any* related state in the second and yield a state of the second system related to the result state when the letter acts on the first system, and *vice versa*.

Bisimulations have been studied extensively, but our interest is primarily in the *greatest bisimulation* between a transition system and itself. It is easy to see that the union of a family of bisimulations is also a bisimulation. Furthermore, for a labelled transition system $\mathscr{A}$, the identity relation on its states is clearly a bisimulation. Thus, the union of all bisimulations that contain the identity relation on S is the *greatest self-bisimulation* $\sim_{\mathscr{A}}$ on the transition system $\mathscr{A}$. Moreover, $\sim_{\mathscr{A}}$ is an equivalence relation on S. For details, see Arnold (1992).

The definition of bisimulation and the considerations of the preceding paragraph can immediately be extended to **Graph**$/A$ for any labelling graph A.

Recall that we denote objects of **Graph**$/A$ by $G = (G, l)$ where $G \xrightarrow{l} A$ is the labelling.

Definition 2.3. Let $G = (G, l)$ and $G' = (G', l')$ be objects of **Graph**$/A$. A *bisimulation* from G to G' is a relation R from G_0 to G'_0 that satisfies:

(ia) The projection of R on G_0 is onto.
(ib) The projection of R on G'_0 is onto.
(iia) For every edge $e : s_1 \xrightarrow{a} s_2$ in G and node s'_1 such that $s_1 R s'_1$ there is an edge $e' : s'_1 \xrightarrow{a} s'_2$ in G' with $s_2 R s'_2$.
(iib) Similarly for edges in G'.

This definition clearly generalises Definition 2.2, and in the rest of the paper when we write ‘bisimulation’ we mean it in the sense just given, and say that G and G' are *bisimilar*.

Lemma 2.1.

(i) The union of bisimulations is a bisimulation, as is the identity relation.
(ii) There is a largest self-bisimulation $\sim_G$ on any G in **Graph**$/A$. $\sim_G$ is an equivalence relation on G.

The proofs are easy and follow Arnold (1992).

The quotient set of the states S of a labelled transition system $\mathscr{A}$ by the largest self-bisimulation $\sim_{\mathscr{A}}$ is the state set for a quotient transition system with the same labels (Arnold 1992). Though objects of **Graph**$/A$ allow multiple edges with the same label between two states, it is still easy to define a suitable quotient. We first note that $\sim_G$ induces an equivalence relation $\sim^e_G$ on the edges G_1 of G defined by $e_1 \sim^e_G e_2$ iff e_1 and e_2 have the same label and their sources and targets are $\sim_G$ related (note also that this does not require that the sources are different).

Definition 2.4. Let $G = (G, l)$ be an object of **Graph**$/A$. The *minimisation* of G is the labelled graph $MG = (MG, l_M)$ in **Graph**$/A$ where MG is the graph whose:

— nodes MG_0 are the quotient set of the nodes G_0 of G by $\sim_G$;
— edges MG_1 are the quotient of the edges G_1 of G by $\sim^e_G$;

— $(l_M)_0([g]) = l_0(g)$ where $[g]$ is the $\sim_G$ class of the node g in G.
— $(l_M)_1([e]) = l_1(e)$ where $[e]$ is the $\sim^e_G$ class of the edge e in G_1.

The *projection morphism* $\pi_G : G \longrightarrow MG$ in **Graph**$/A$ is defined on nodes g and edges e of G by $(\pi_G)_0(g) = [g]$, $(\pi_G)_1(e) = [e]$.

Notice that the definition of $\sim^e_G$ implies that there is at most one edge labelled a from $[g]$ to $[g']$ in MG, so MG is actually a transition system.

Proposition 2.1. MG and the projection morphism $\pi_G : G \longrightarrow MG$ are well-defined.

Proof. We need to show that the labelling l_M is independent of representatives and that π_G preserves labelling. The latter is obvious when the former is proved, so suppose that $g \sim_G g'$. If there is an edge labelled a leaving g say, then there is also one leaving g'. Hence both g and g' are labelled by the same node of the labelling graph A, that is, $l_0(g) = l_0(g')$. If neither g nor g' has any edges leaving, they must be identified by $\sim_G$ since (iia) and (iib) are trivially satisfied. Thus l_M is well-defined on nodes. By the definition of $\sim^e_G$, it is clear that if $e_1 \sim^e_G e_2$, they have the same label, so l_M is well-defined on edges. □

Example 2.1. Consider the depth two binary tree with all edges labelled a (the left illustration below). Its minimisation is the depth two unary tree (on the right below):

$$\begin{tikzcd}
 & & & b \arrow[dll, "a"'] \arrow[drr, "a"] & & & \\
 & c \arrow[dl, "a"'] \arrow[dr, "a"] & & & & d \arrow[dl, "a"'] \arrow[dr, "a"] & \\
e & & f & & g & & h
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
{[b]} \arrow[d, "a"'] \\
{[c]} \arrow[d, "a"'] \\
{[e]}
\end{tikzcd}$$

Consider the infinite ‘comb’ at left below. Its minimisation is the labelled graph to the right.

$$\begin{tikzcd}
 & c_0 \arrow[dl, "a"'] \arrow[dr, "b"] & & \\
d_0 & & c_1 \arrow[dl, "a"'] \arrow[dr, "b"] & \\
 & d_1 & & \ddots
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & {[c_0]} \arrow[dl, "a"'] \arrow[loop right, "b"] \\
{[d_0]} &
\end{tikzcd}$$

We also note the following proposition.

Proposition 2.2. The identity relation Δ_{MG_0} on MG_0 is the largest self-bisimulation $\sim_{MG}$ on MG, and $\sim^e_{MG}$ is the identity relation on MG_1. Hence the projection $MG \xrightarrow{\pi_{MG}} MMG$ is an identity arrow, and minimisation is idempotent.

Proof. It is clear that the largest bisimulation on MG is the identity relation: this relation is a bisimulation and unequal nodes of MG cannot be related by $\sim_{MG}$. Indeed, if unequal nodes could be made bisimilar, $\sim_G$ would have already related them. It follows that if two edges in MG_1 are related by $\sim^e_{MG}$, they must have the same source, label and target (since $\sim_{MG}$ is the identity), and any two such edges were already related by $\sim^e_G$, so the two edges are identical. Thus the quotient arrow $\pi_{MG} : MG \longrightarrow MMG = MG/\sim_{M(G)} = MG$ is the identity on the nodes and edges of MG. □

Remark 2.1. In the case of a *finite* labelled graph there is an algorithm to compute the largest self-bisimulation on a labelled graph G. It begins with the total relation on the nodes of G and successively separates nodes that cannot be related by any bisimulation. The iterative step is as follows:

For any edge $g \xrightarrow{a} g'$ and node h, if there is no edge labelled a emanating from h, or if every edge labelled a emanating from h has target not equivalent to g', then g and h are not equivalent.

The algorithm terminates when the relation stabilises at the largest self-bisimulation. It relies on finiteness for termination, is detailed in Katis *et al.* (2000) and is similar to the algorithm for maximal self-bisimulation in Arnold (1992).

To extend minimisation to arrows we will need an important restriction.

Definition 2.5. Let G and G' be labelled graphs in **Graph**$/A$. A **Graph**$/A$ morphism $\varphi : G \longrightarrow G'$ is called *path lifting* if whenever $e' : \varphi(g) \xrightarrow{a} g'$ is an an a-labelled edge in G' with source $\varphi(g)$ and target g', there is an (a-labelled) edge $e : g \xrightarrow{a} g_1$ in G that satisfies $\varphi(e) = e'$ and hence $\varphi(g_1) = g'$.

Path lifting morphisms have been considered by several authors (see Joyal *et al.* (1996) or Bunge and Fiore (2000), and references there). Clearly, a graph isomorphism is path lifting, and path lifting morphisms compose, so there is a category of labelled graphs and path-lifting morphisms. Notice first that a path lifting morphism that is onto on nodes is also onto on edges. Indeed, any edge in the codomain then has a source that is the image of a domain node, so, by the definition, that edge is the image of a domain edge. Thus, an onto on nodes path lifting morphism is an epimorphism in **Graph**$/A$. Next, we consider the interplay of bisimulations and path lifting morphisms. Let G, G' be in **Graph**$/A$ and $r : R \hookrightarrow G_0 \times G'_0$ be a relation on their node sets. Define $\hat{R}$ to be the labelled graph with nodes R and an edge $(g_1, g'_1) \xrightarrow{a} (g_2, g'_2)$ exactly when there are edges $g_1 \xrightarrow{a} g_2$ in G and $g'_1 \xrightarrow{a} g'_2$ in G'. Thus an edge $(g_1, g'_1) \xrightarrow{a} (g_2, g'_2)$ of $\hat{R}$ is an edge of $G \times G'$, so we can extend r to a morphism $\hat{r} : \hat{R} \hookrightarrow G \times G'$ in **Graph**$/A$.

Lemma 2.2. A relation r on the node sets of G, G' in **Graph**$/A$ is a bisimulation if and only if

$$p_0\hat{r} : G \longleftarrow \hat{R} \longrightarrow G' : p_1\hat{r}$$

is a span of onto on nodes path lifting morphisms in **Graph**$/A$.

Proof. If r is a bisimulation, then, by (ia) and (ib), $p_0\hat{r}$ and $p_1\hat{r}$ are onto on nodes. To see that $p_0\hat{r}$ is path-lifting, suppose that $g_1 \xrightarrow{a} g_2$ has $g_1 = (p_0\hat{r})_0(g_1, g'_1)$. Since r is a bisimulation, (iia) delivers an edge $(g_1, g_1') \xrightarrow{a} (g_2, g_2')$ in $\hat{R}$ as required. Similarly, $p_1\hat{r}$ is path lifting.

Conversely, if $p_0\hat{r}$ is onto on nodes and path lifting and g_1 is a node of G, then there is a node (g_1, g_1') in $\hat{R}$, so the projection of R is onto G_0. If in addition $g_1 \xrightarrow{a} g_2$, then, since $p_0\hat{r}$ is path lifting, there is an edge $(g_1, g_1') \xrightarrow{a} (g_2, g_2')$ in $\hat{R}$, so there is an edge $g_1' \xrightarrow{a} g_2'$ in G' with $g_2 R g_2'$, so (iia) is satisfied. The proof for (iib) is similar. □

Furthermore, *any* span of path lifting morphisms that is onto on nodes determines a jointly monic path lifting span arising from a bisimulation.

Lemma 2.3. Let $G \xleftarrow{\alpha} H \xrightarrow{\beta} G'$ be a span of path lifting morphisms that are onto on nodes. Then $R = \{(\alpha_0 h, \beta_0 h) \mid h \in H_0\}$ is a bisimulation from G to G'.

Proof. First, R has onto projections since α, β are onto on nodes. If $\alpha_0 h \xrightarrow{a} g_1$, there is an edge $h \xrightarrow{a} h_1$ in H with $g_1 = \alpha_0 h$ so that $\beta_0 h \xrightarrow{a} \beta_0 h_1$ in G' and $\alpha_0 h_1 R \beta_0 h_1$, so (iia) is satisfied. The proof for (iib) is similar. □

An onto on nodes path lifting morphism $\varphi : G \longrightarrow G'$ provides the path lifting span $1_G : G \longleftarrow G \longrightarrow G' : \varphi$ of the sort just considered, and hence R of the lemma provides a *functional bisimulation* from G to G'.

As the next two lemmas show, path lifting morphisms behave well with respect to bisimulation.

Lemma 2.4. If $G \xrightarrow{\varphi} H$ is a path lifting morphism that is onto on nodes and R_G is a bisimulation on G, then the relation R_H on H defined by hR_Hh' iff $\exists g, g'$ such that gR_Gg' & $h = \varphi(g)$ & $h' = \varphi(g')$ is a bisimulation on H.

Proof. First, for any node h of H, there is a g such that $h = \varphi(g)$ and a g' such that gR_Gg'. Hence $hR_H\varphi(g')$, so (ia) is satisfied for R_H. The proof for (ib) is similar.

Now suppose that $e' : h \xrightarrow{a} h_1$ and hR_Hh'. Since φ is onto we have g with $\varphi(g) = h$, g_1 and $e : g \xrightarrow{a} g_1$ with $\varphi(e) = e'$ and g' with gR_Gg' and $\varphi(g') = h'$. By the path lifting property and (iia) for R_G, there is then an edge $e_1 : g' \xrightarrow{a} g_1'$ with $g_1R_Gg_1'$. Thus $\varphi(g_1')R_H\varphi(g_1)$ and $\varphi(e_1) : h' \xrightarrow{a} \varphi(g_1')$. So (iia) is satisfied for R_H. The proof for (iib) is similar. □

A more conceptual proof of the preceding lemma is provided by the observations that a self-bisimulation R on G determines a jointly monic span of path lifting morphisms $G \longleftarrow \hat{R} \longrightarrow G$ and that path lifting morphisms compose.

Lemma 2.5. If $G \xrightarrow{\varphi} H$ is a path lifting morphism that is onto on nodes and R_H is a bisimulation on H, then the relation R_G defined by gR_Gg' iff $\varphi(g)R_H\varphi(g')$ is a bisimulation on G.

Proof. First, for any node g of G, there is an h such that $\varphi(g)R_Hh$, but $h = \varphi(g')$ for some node g' of G, hence gR_Gg', so (ia) is satisfied for R_G. The proof for (ib) is similar.

Now suppose that $g \xrightarrow{a} g_1$ and gR_Gg' in G. Thus $\varphi(g) \xrightarrow{a} \varphi(g_1)$ and $\varphi(g)R_H\varphi(g')$, so there is $e' : \varphi(g') \xrightarrow{a} h$ in H with $\varphi(g_1)R_Hh$. But $h = \varphi(g_1')$ for some g_1' in G, so $g_1R_Gg_1'$, and since φ is path lifting, there is an edge $e : g' \xrightarrow{a} g'_1$ in G with $\varphi(e) = e'$. Hence (iia) is satisfied. The proof for (iib) is similar. □

We also note the following result.

Lemma 2.6. If the cospan $G \xrightarrow{\alpha} H \xleftarrow{\beta} G'$ in **Graph**$/A$ has α path lifting, its pullback $G \xleftarrow{\beta'} P \xrightarrow{\alpha'} G'$ in **Graph**$/A$ has α' path lifting.

Proof. The proof is a straightforward exercise. □

The following result is important for us.

Proposition 2.3. The projection morphism $\pi_G : G \longrightarrow MG$ is a path lifting morphism that is onto on nodes and on edges.

Proof. The proof is immediate: any edge $[e]$ in MG is the image under π_G of an edge in G with the same label, so π_G is also onto on nodes. □

We can also extend M to suitable path-lifting arrows.

Definition 2.6. Let $G \xrightarrow{\varphi} H$ be a path lifting morphism that is onto on nodes. We define $MG \xrightarrow{M\varphi} MH$ on nodes of G by $(M\varphi)_0([g]) = [\varphi(g)]$ and on edges by $(M\varphi)_1([e]) = [\varphi(e)]$.

To see that $M\varphi$ is well-defined, suppose that $g \sim_G g'$. By Lemma 2.4, there is a bisimulation R_H on H with $\varphi(g) R_H \varphi(g')$, so it follows that $\varphi(g) \sim_H \varphi(g')$. Similarly, φ respects edges that are identified by $\sim^e_G$. Since φ is path lifting, so is $M\varphi$.

Moreover, we have the following proposition.

Proposition 2.4. If $G \xrightarrow{\varphi} H$ is a path lifting morphism that is onto on nodes and edges, then $MG \xrightarrow{\overset{M\varphi}{\cong}} MH$ in **Graph**$_A$.

Proof. By the definition of $M\varphi$, we have $M\varphi\pi_G = \pi_H\varphi$ and the latter is onto on nodes and edges, hence so is $M\varphi$. So to see that $M\varphi$ is an isomorphism, we need only observe that it is injective on nodes and edges. Suppose that $M\varphi([g]) = M\varphi([g'])$, that is, $[\varphi(g)] = [\varphi(g')]$ or $\varphi(g) \sim_H \varphi(g')$. By Lemma 2.5, there is a bisimulation R_G on G such that gR_Gg', hence $g \sim_G g'$. Thus $[g] = [g']$. By a similar argument, $M\varphi$ is injective on edges. □

This gives another characterisation of bisimilarity.

Corollary 2.1. G and G' in **Graph**$/A$ are bisimilar if and only if $MG \cong MG'$.

Proof. If $MG \xrightarrow{\overset{M\varphi}{\cong}} MG'$ say, then $\varphi\pi_G$ and $\pi_{G'}$ determine a cospan from G to G' of path lifting morphisms whose pullback determines a bisimulation from G to G' by Lemmas 2.6 and 2.3. By Lemma 2.2, a bisimulation from G to G' determines a path lifting span whose legs, being onto on nodes and edges, are inverted by M. □

For the study of minimisation and minimal realisation we are interested in *reachable pointed graphs*. Notice that the construction of MG above does not require either an (initial) point or reachability. The definition of transition system in Joyal *et al.* (1996), for example, includes an 'initial state' as part of the data, and morphisms preserve initial states. Indeed, pointed graphs and their morphisms are appropriate for our considerations. Moreover, as is usual in automata theory, we require reachability for our compositional minimisation. Note that this was not needed above, and there is a similar situation in Rosebrugh *et al.* (1998).

A *pointed graph* is a pair (G, g_0) where G is an object of **Graph** and g_0 is an element of G_0. A *morphism of pointed graphs* is a **Graph** morphism that preserves the specified element, and we call the resulting category Pt**Graph**. This category may also be described as $N/$**Graph** where N is the graph with one node and no edges (see Section 5). A pointed graph (G, g_0) is *reachable* if there is a path of edges in G_1 from g_0 to any node in G_0. We will have more to say about these concepts in Section 5. For now we denote the full subcategory of Pt**Graph** whose objects are reachable by RchPt**Graph**. For any pointed graph G its *reachable part* is the subgraph consisting of reachable nodes (and edges) that we denote by G_R. We write $\rho_G : G_R \longrightarrow G$ for the inclusion.

Notation 2.1. The (non-full) subcategory of RchPt**Graph** determined by the objects in RchPt**Graph** and morphisms that are path lifting graph morphisms in RchPt**Graph** is denoted by PL**Graph**.

Let $(A, *)$ be a pointed graph. We will write PL**Graph**$_A$ for the category in which an object $((G, g_0), l)$ is a pointed graph morphism $l : (G, g_0) \longrightarrow (A, *)$ from a reachable pointed graph (G, g_0) to $(A, *)$, and an arrow $\varphi : ((G, g_0), l) \longrightarrow ((G', g'_0), l')$ is a path lifting morphism $\varphi : (G, g_0) \longrightarrow (G', g'_0)$ of reachable pointed graphs such that $l'\varphi = l$. Thus, we have the following.

Notation 2.2. PL**Graph**$_A$ has reachable pointed graphs (G, l) labelled by $(A, *)$ as objects. Its morphisms are path lifting and label-preserving.

Since the domains of objects of PL**Graph**$_A$ are pointed, there is no harm in making the same assumption for our labelling graph A. The results of this section do not depend on this assumption, but our compositional minimisation theory will. Note that we do not require $(A, *)$ to be reachable nor l to be path lifting.

We note the following result.

Lemma 2.7. PL**Graph**$_A$ has pullbacks.

Proof. Suppose that $G \xrightarrow{\alpha} H \xleftarrow{\beta} G'$ is a cospan in PL**Graph**$_A$. Let P with projections $G \xleftarrow{\beta'} P \xrightarrow{\alpha'} G'$ be the pullback in **Graph**$/A$ of α along β and $P_R \xrightarrow{\rho_P} P$ be the inclusion of its reachable part. Now P_R is clearly pointed and we claim that P_R (with projections $\alpha'\rho_P$ and $\beta'\rho_P$) is the pullback in PL**Graph**$_A$. As noted above in Lemma 2.6, α' and β' are path lifting, and, of course, so is ρ_P. Thus, if $G \xleftarrow{\gamma} T \xrightarrow{\delta} G'$ in PL**Graph**$_A$ satisfies $\alpha\gamma = \beta\delta$, then the universal $T \longrightarrow P$ clearly factors through P_R since T is reachable. □

Lemma 2.8. The arrows of $\mathsf{PL}\mathbf{Graph}_A$ are onto on nodes and edges, and so all arrows of $\mathsf{PL}\mathbf{Graph}_A$ are epi.

Proof. Let $\varphi : (G, l) \longrightarrow (G', l')$ be a morphism of $\mathsf{PL}\mathbf{Graph}_A$ and g' be a node of G'. Since G' is reachable, there is a path of edges in G', $e'_1, \ldots, e'_n$ labelled $a_1, \ldots, a_n$, say, from the point of G' to g'. Since the point of G' is the image of the point of G, the path lifting property provides a path of edges $e_1, \ldots, e_n$ (and labelled $a_1, \ldots, a_n$) that φ maps to the given path. Thus g' is in the image of φ. Now if e' is any edge from g', the path lifting property shows that it is in the image of φ.

The epimorphism cancellation property for all arrows of $\mathsf{PL}\mathbf{Graph}_A$ follows immediately. $\square$

Indeed, as the proof shows, any path lifting morphism whose codomain is a reachable graph is onto on nodes and edges. Now we have functorial minimisation on $\mathsf{PL}\mathbf{Graph}_A$.

Proposition 2.5. $M : \mathsf{PL}\mathbf{Graph}_A \longrightarrow \mathsf{PL}\mathbf{Graph}_A$ is a functor. The arrows π_G are the components of a natural transformation $\pi : 1_{\mathsf{PL}\mathbf{Graph}_A} \longrightarrow M$.

Proof. M is defined on arrows by Definition 2.6 using Lemma 2.8. For $G \xrightarrow{\varphi} G'$ in $\mathsf{PL}\mathbf{Graph}_A$, the definition of $M\varphi$ clearly makes it point preserving. If, also, $G' \xrightarrow{\psi} G''$ in $\mathsf{PL}\mathbf{Graph}_A$, then

$$M\psi M\varphi([g]) = M\psi([\varphi(g)]) = [\psi\varphi(g)] = M(\psi\varphi)([g]).$$

Naturality of π is equally immediate, as noted in the proof of Proposition 2.4. $\square$

The following are crucial for the rest of the paper.

Lemma 2.9. If $G \xrightarrow{\alpha} G$ is a morphism in $\mathsf{PL}\mathbf{Graph}_A$, then the relation R_α defined on G_0 by $gR_\alpha g'$ iff $\alpha(g) = g'$ is a bisimulation on G.

Proof. Since α is onto on nodes, R_α is the functional bisimulation defined in Lemma 2.3 for the span $1_G : G \longleftarrow G \longrightarrow G : \alpha$. $\square$

Corollary 2.2. The only automorphism of MG in $\mathsf{PL}\mathbf{Graph}_A$ is the identity.

Proof. Suppose $MG \xrightarrow{\alpha} MG$ is an automorphism. By the previous lemma, its graph is a bisimulation R_α and is contained in the largest bisimulation on MG, namely the identity relation by Proposition 2.2. The only graph of a function contained in the identity relation is the identity relation. $\square$

Lemma 2.10. If $G \underset{\beta}{\overset{\alpha}{\rightrightarrows}} H$ are in $\mathsf{PL}\mathbf{Graph}_A$, then $M\alpha = M\beta$.

Proof. By Proposition 2.4, $M\alpha$ is invertible. Thus $(M\alpha)^{-1}M\beta$ is an automorphism of MG, and hence it is the identity by Corollary 2.2, and the result follows. $\square$

Lemma 2.11. If $H \underset{\beta}{\overset{\alpha}{\rightrightarrows}} MG$ are in $\mathsf{PL}\mathbf{Graph}_A$, then $\alpha = \beta$.

Proof. Consider

$$\begin{tikzcd} H \arrow[r, shift left, "\alpha"] \arrow[r, shift right, "\beta"'] \arrow[d, "\pi_H"'] & MG \arrow[d, "\pi_{MG}"] \\ MH \arrow[r, "M\alpha = M\beta"'] & MMG \end{tikzcd}$$

Since π_{MG} is the identity by Proposition 2.2, naturality of π and Lemma 2.10 give $\alpha = M\alpha\pi_H = M\beta\pi_H = \beta$. □

Corollary 2.3. The projection $G \xrightarrow{\pi_G} MG$ is the unique arrow from G to MG in $\mathsf{PL}\mathbf{Graph}_A$, and hence $\pi_{MG} = M\pi_G$.

Lemma 2.12. If $G \xrightarrow{\alpha} H$ is in $\mathsf{PL}\mathbf{Graph}_A$, there is a unique $H \xrightarrow{\beta} MG$ such that $\beta\alpha = \pi_G$.

$$\begin{tikzcd} G \arrow[rr, "\alpha"] \arrow[dr, "\pi_G"'] & & H \arrow[dl, "\beta"] \\ & MG & \end{tikzcd}$$

Proof. Define $\beta = (M\alpha)^{-1}\pi_H$, then $\beta\alpha = \pi_G$ by Lemma 2.11. By the same result, β is the unique arrow from H to MG. □

By Corollary 2.3, the pair (M, π) is the data for an idempotent monad

$$M : \mathsf{PL}\mathbf{Graph}_A \longrightarrow \mathsf{PL}\mathbf{Graph}_A.$$

Notation 2.3. The full subcategory of $\mathsf{PL}\mathbf{Graph}_A$ whose objects are the image of the idempotent monad M is denoted $\mathsf{MPL}\mathbf{Graph}_A$. We write M_A for the minimisation functor $\mathsf{M}_A : \mathsf{PL}\mathbf{Graph}_A \longrightarrow \mathsf{MPL}\mathbf{Graph}_A$ and $\mathsf{I}_A : \mathsf{MPL}\mathbf{Graph}_A \longrightarrow \mathsf{PL}\mathbf{Graph}_A$ for the inclusion, so $M = \mathsf{I}_A\mathsf{M}_A$.

Now, $\mathsf{MPL}\mathbf{Graph}_A$ is essentially discrete. Indeed, by Lemma 2.11, there is at most one arrow from any MG to any MH, so for any $MG \xrightarrow{\varphi} MH$ we have $\varphi = M\varphi$ and the latter is iso. We sum up with the following proposition.

Proposition 2.6. $\mathsf{MPL}\mathbf{Graph}_A$ is an essentially discrete full subcategory of $\mathsf{PL}\mathbf{Graph}_A$. There is an adjunction:

$$\mathsf{M}_A \dashv \mathsf{I}_A : \mathsf{MPL}\mathbf{Graph}_A \longrightarrow \mathsf{PL}\mathbf{Graph}_A$$

This is the local part of the left-hand adjunction in the diagram of the Introduction.

3. Compositional minimisation

In this section we consider the relationship of our minimisation theory to the composition of transition systems. The composition we use here is based on that for the bicategory of spans of graphs studied as an algebra of transition systems in Katis *et al.* (1997b). We find that there is a *comparison* from the composite of minimisations to the minimisation of a composite. Indeed, we find that minimisation is an idempotent *lax monad* (Carboni and Rosebrugh 1991) on our bicategory.

To begin, note that if A and B are pointed graphs (and usually we will not mention the point), then an object $(G, \langle l, r\rangle)$ of $\mathsf{PL}\mathbf{Graph}_{A\times B}$ has an underlying span of graphs:

$$A \xleftarrow{\;l\;} G \xrightarrow{\;r\;} B.$$

G is called the 'head' graph of the span, and l and r are its left and right legs.

Our first objective is to define a bicategory $\mathsf{SpPL}\mathbf{Graph}$ whose objects are those of $\mathsf{Pt}\mathbf{Graph}$, and whose hom category for objects A and B is $\mathsf{PL}\mathbf{Graph}_{A\times B}$. That is, the 1-cells of $\mathsf{SpPL}\mathbf{Graph}$ from A to B are to be objects of $\mathsf{PL}\mathbf{Graph}_{A\times B}$, which it is convenient to think of as spans, and the 2-cells are morphisms of $\mathsf{PL}\mathbf{Graph}_{A\times B}$.

The identity arrow from A to A in $\mathsf{SpPL}\mathbf{Graph}$ is $(A_R, \langle \rho_A, \rho_A\rangle)$ where A_R is the reachable nodes of A and ρ_A is the inclusion $A_R \longrightarrow A$. Notice that this is the reason for the requirement mentioned in the last section that a labelling graph (an object of $\mathsf{SpPL}\mathbf{Graph}$) be pointed, for without this requirement there would be no candidate for an identity arrow. Furthermore, for an object (G, l) of $\mathsf{PL}\mathbf{Graph}_A$ we did not require the labelling graph A to be reachable. The reason for this is that even if A and B are reachable, it is not usually the case that $A \times B$ is reachable, but we have defined our hom categories to be of the form $\mathsf{PL}\mathbf{Graph}_{A\times B}$ so that we can apply the minimisation theory of the previous section directly to them. It is true that we could have taken only reachable pointed graphs as the objects of our bicategory, and then defined the hom categories to be spans of these. However, doing so would require some reworking of the results above (which we consider to be of independent interest) before application of them in this section, and would not avoid the main considerations in the next paragraph. The interested reader can provide the details.

Next we consider the composition of 1-cells. Suppose that $(G, \langle l, r\rangle)$ and $(G', \langle l', r'\rangle)$ are 1-cells in $\mathsf{SpPL}\mathbf{Graph}(A, B)$ and $\mathsf{SpPL}\mathbf{Graph}(B, C)$, respectively. Thus we have underlying spans $G = A \xleftarrow{\;l\;} G \xrightarrow{\;r\;} B$ and $G_1 = B \xleftarrow{\;l'\;} G' \xrightarrow{\;r'\;} C$. Their composite $G'' = G'G$ must be in $\mathsf{SpPL}\mathbf{Graph}(A, C)$, so we need a reachable pointed graph G'' and a span $G'' = A \xleftarrow{\;l''\;} G'' \xrightarrow{\;r''\;} C$ whose legs are morphisms of pointed graphs. Now pullbacks exist in the category $\mathsf{Pt}\mathbf{Graph}$ of pointed graphs – just take the $\mathbf{Graph}$ pullback with the point given by the pair of original points. However, the $\mathsf{Pt}\mathbf{Graph}$ pullback of reachable pointed graphs need not be reachable. Thus, we define $G'' = (G \times_B G')_R$ to be the reachable part of the pointed graph pullback $G \times_B G'$ of r along l'. Denoting the inclusion $i : G'' \longrightarrow G \times_B G'$, we define $l'' = lpi$ and $r'' = r'p'i$ where p and p' are projections. This composition respects the identity 1-cells defined above. That requires the observation that

$$(A_R \times_A G)_R \cong (A \times_A G)_R \cong G_R = G$$

for reachable pointed graphs G – we leave the easy proof of this to the reader. It is also easy to see directly that this composition is associative up to isomorphism. These remarks also follow from the fact that spans of pointed graphs with reachable head are the coalgebra bicategory for the reachability colax comonad on spans of pointed graphs (see Section 5).

To complete the definition of SpPL**Graph**, we need to consider horizontal composition of 2-cells. Fortunately, the fact that they are path lifting means that this operation is essentially inherited from the one for spans of graphs.

Lemma 3.1. Suppose that $\psi : G_0 \longrightarrow G_1$ is an arrow of spans of graphs from $G_0 = A \xleftarrow{l_0} G_0 \xrightarrow{r_0} B$ to $G_1 = A \xleftarrow{l_1} G_1 \xrightarrow{r_1} B$, and $\phi : G_2 \longrightarrow G_3$ is an arrow of spans of graphs from $G_2 = B \xleftarrow{l_2} G_2 \xrightarrow{r_2} C$ to $G_3 = B \xleftarrow{l_3} G_3 \xrightarrow{r_3} C$. If ψ and ϕ are path lifting, then so is their **Span**(**Graph**) horizontal composite $\phi \circ \psi$.

Proof. For simplicity we assume that A and C have only a single node. Suppose that $(e_3, e_1) : \phi \circ \psi(g_2, g_0) \xrightarrow{a} (g_3, g_1)$ is an edge labelled a, that is, there are edges $e_3 : \phi g_2 \xrightarrow{a} g_3$ and $e_1 : \psi g_0 \xrightarrow{a} g_1$. But path lifting for ϕ and ψ then provides $e_2 : g_2 \xrightarrow{a} g_5$ and $e_1 : g_1 \xrightarrow{a} g_4$ in G' and G, respectively, that ϕ and ψ send to e_3 and e_1. Thus there is an edge $(e_2, e_0) : (g_2, g_0) \xrightarrow{a} (g_5, g_4)$ of $G'G$ that $\phi \circ \psi$ sends to the original (e_3, e_1). □

Now the composition of 1-cells in SpPL**Graph** differs from the composition of spans of pointed graphs by the restriction to the reachable part of the ordinary composite, but the important point is that the path lifting property is maintained when taking reachable parts, so we have the following proposition.

Proposition 3.1. For 2-cells φ in SpPL**Graph**(A, B) and ψ in SpPL**Graph**(B, C), their **Span**(**Graph**) horizontal composite $\varphi \circ \psi$ restricted to reachable parts is in fact in the subcategory SpPL**Graph**(A, C).

Proof. The proof follows from the preceding paragraph and Lemma 3.1. □

We note further that the interchange law inherited from **Span**(**Graph**) holds for our proposed vertical and horizontal composites, so this completes our definition.

Notation 3.1. The bicategory SpPL**Graph** has pointed graphs as objects, and for objects A and B the hom category is PL**Graph**$_{A\times B}$.

The compositional minimisation that we seek is defined locally by the minimisation functors in the previous section. While the composite of minimisations need not equal the minimisation of the composite, there is a comparison that makes the local minimisation lax functorial.

Proposition 3.2. The functors $M_{(A,B)}$: SpPL**Graph**$(A, B) \longrightarrow$ SpPL**Graph**(A, B) defined by $M_{(A,B)} = M$: PL**Graph**$_{A\times B} \longrightarrow$ PL**Graph**$_{A\times B}$ determine an idempotent lax monad on SpPL**Graph**.

Proof. A lax monad (Carboni and Rosebrugh 1991) is an identity on object morphisms of bicategories (lax functor) that is locally a monad and satisfies several equations. When the local monads are idempotent, Proposition 42 of Rosebrugh *et al.* (1998) simplifies the equations. In the case at hand, we need to define a lax functor

$$(M, \tau) : \text{SpPL}\mathbf{Graph} \longrightarrow \text{SpPL}\mathbf{Graph}$$

that is locally an idempotent monad, has identity components τ_A given by the local monad units and satisfies the equation at the end of this proposition making τ compatible with the monad units. By the results of the previous section, the minimisation functors $M_{(A,B)}$ already provide local idempotent monads.

We next consider the lax functor structure. Let

$$G = A \xleftarrow{l} G \xrightarrow{r} B \quad \text{and} \quad G' = B \xleftarrow{l'} G' \xrightarrow{r'} C$$

be spans defining arrows of SpPL**Graph**. We require a comparison arrow $\tau_{G'G}$ from $MG'MG$ to $MG'G$. We define this on a pair of representative nodes $[g'], [g]$ from MG and MG' that appear in the composite $MG'MG$ by noting that then (g', g) is in the composite $G'G$, so we let $\tau_{G'G}([g'], [g]) = [(g', g)]$. A similar formula defines $\tau_{G'G}$ on edges. Checking that this $\tau_{G'G}$ is well-defined and path lifting is done easily by recalling Propositions 2.1 and 2.3. We also need, for each object A of SpPL**Graph**, an identity component arrow $\tau_A : 1_A \longrightarrow M_{(A,A)}(1_A)$ (remembering that M is identity on objects so that $1_{MA} = 1_A$). For these τ_A we take the (path lifting) projections π_{A_R} from Section 2, the units for the local monads. With these definitions, it is easy to see that (M, τ) satisfies the requirements to be a lax functor.

It only remains to check compatibility of τ with the monad units, that is that the following commutes:

$$\begin{tikzcd} G'G \arrow[rr, "\pi_{G'G}"] \arrow[dr, "\pi_{G'} \circ \pi_G"'] & & M(G'G) \\ & MG'MG \arrow[ur, "\tau_{G'G}"'] & \end{tikzcd}$$

However, this is just another exercise in equating equivalence classes. □

Notation 3.2. We use SpMPL**Graph** to denote the algebras for the lax monad M on SpPL**Graph**.

The 1-cells in SpMPL**Graph**(A, B) are the full subcategory of SpPL**Graph**(A, B) consisting of the minimised graphs MPL**Graph**$_{A\times B}$. Note, however, that the composite of MG and MG' in SpMPL**Graph** is *not* their SpPL**Graph** composite, but rather its minimisation (see Carboni and Rosebrugh (1991) and Rosebrugh *et al.* (1998)).

The point of Proposition 3.2 is that minimisation is lax compositional. Thus, in order to compute the minimisation of a composite of spans it is enough to compute the minimisations of the factors and then minimise the composite of these, that is

$$M(G'G) = MM(G'G) \xleftarrow{\cong} M(MG'MG)$$

using idempotence, the comparison from above, and the fact that $M(\varphi)$ is always iso.

Although the next result could also be proved directly using the fact that a bisimulation determines a span of path lifting arrows, following a suggestion of a referee, the proof we will give is based on the remarks just made.

Corollary 3.1. Suppose that G and H in $\mathsf{SpPL}\mathbf{Graph}(A, B)$ are bisimilar, as are G' and H' in $\mathsf{SpPL}\mathbf{Graph}(B, C)$. Then $G'G$ is bisimilar to $H'H$.

Proof. Since G and H are bisimilar, $MG \cong MH$ and, similarly, $MG' \cong MH'$. Thus

$$M(G'G) \cong M(MG'MG) \cong M(MH'MH) \cong M(H'H).$$

By Corollary 2.1 we have $G'G$ bisimilar to $H'H$. □

Expressions in the algebra $\mathsf{SpPL}\mathbf{Graph}$ can be used to model synchronous distributed systems. Minimisation of the expression can be evaluated efficiently by minimising sub-expressions, which is also useful in detecting deadlock in the original system. In order to discuss asynchronous systems such as the Dining Philosopher Problem, we require an extension of the results above to the case of reflexive graphs, which we pursue in the next section.

4. Reflexive graphs and branching bisimulation

For this section we require a specified 'null' edge at each node of any graph, and require that morphisms, including labellings, preserve these. Thus, the label of the null edge is required to be the null label. The null label is generically denoted by '–' below. We will outline results for reflexive graphs that are analogous to those in Sections 2 and 3.

The specification of the null edge at each node of a graph $G : G_1 \overset{s_G}{\underset{t_G}{\rightrightarrows}} G_0$ is a mapping $G_0 \xrightarrow{r_G} G_1$ such that $s_G r_G = 1_{G_0} = t_G r_G$; thus G is a *reflexive graph*. Morphisms of reflexive graphs are exactly graph morphisms that preserve the reflexivity r_G, and hence the null edges. We denote the category of reflexive graphs and their morphisms by $\mathbf{RfGraph}$. There is a forgetful functor $\mathbf{RfGraph} \longrightarrow \mathbf{Graph}$ that has a left adjoint (given by freely adjoining null edges). There is no difficulty in defining the concept of *pointed* reflexive graph and their morphisms, nor with *reachable* pointed reflexive graph. We denote the category of reachable pointed reflexive graphs and their morphisms by $\mathsf{RchPt}\mathbf{RfGraph}$.

When we consider a reflexive graph labelled by a reflexive graph A, that is, an object (G, l) of $\mathbf{RfGraph}/A$, it is possible that a non-null edge may be labelled by a null edge. The intuition here is that the machine may 'idle' by consuming no input or producing no output through state transitions labelled with the null edge, so we are considering asynchronous systems. We will have to take account of this possibility by modifying the definitions of bisimulation and path-lifting morphisms needed for minimisation. We do this by introducing a notion of *branching bisimulation* similar to that considered by van Glabbeek and Weijland (van Glabbeek and Weijland 1996), and reflexive path lifting morphisms. Branching bisimulation is stronger than the weak bisimulation notion often considered in concurrency theory, but we agree with van Glabbeek in arguing that it is more consistent with the requirements proposed by Hennessey and Milner (Hennessey and Milner 1980) for observational, or weak, equivalence.

We first consider our version of branching bisimulation.

Definition 4.1. Let A be a reflexive graph. Let $G = (G, l)$ and $H = (H, k)$ be objects of **RfGraph**$/A$. A *fully branching bisimulation* from G to H is a relation R from G_0 to H_0 that satisfies:

(ia) The projection of R on G_0 is onto

(ib) The projection of R on H_0 is onto

(iia) For every edge $e : g \xrightarrow{a} g'$ in G and node h such that gRh, there is a path in H:

$$h \xrightarrow{-} h_1 \xrightarrow{-} \dots h_k \xrightarrow{a} h' \quad k \geqslant 0$$

with $gRh_j, 1 \leqslant j \leqslant k$ and $g'Rh'$

(iib) Similarly for edges in H.

We note that van Glabbeek and Weijland's branching bisimulation requires gRh_j only for $j = k$, but they prove that a maximal branching bisimulation is fully branching in our sense. Moreover, there is no loss of generality in having a null-labelled path only *before* the edge labelled a in H in (iia) rather than, as is required for weak bisimulation, allowing also a path, say $l_1 \xrightarrow{-} l_2 \xrightarrow{-} \dots l_m \xrightarrow{a} h'$ with $g'Rl_j$ *after* $h_k \xrightarrow{a} l_1$. Indeed, one may clearly take l_1 for the h' whose existence is required.

Any reflexive graph has an underlying graph. It is clear that a bisimulation on the underlying graphs of labelled reflexive graphs G, H is a fully branching bisimulation, but a (fully) branching bisimulation need not be a bisimulation. It is easy to see that the identity relation, the composite of fully branching bisimulations, the union of fully branching bisimulations and the opposite of a fully branching bisimulation are all fully branching bisimulations. Thus we have the following analogue of Lemma 2.1.

Lemma 4.1.

(i) The union of fully branching bisimulations is a fully branching bisimulation, as is the identity relation.

(ii) There is a largest self fully branching bisimulation $\approx_G$ on any G in **RfGraph**$/A$, and $\approx_G$ is an equivalence relation on G.

The proofs proceed exactly as in the **Graph** case. There is again an equivalence relation $\approx_G^e$ on edges of G defined exactly as in Section 2. With that relation defined, minimisation is defined as follows.

Definition 4.2. Let $G = (G, l)$ be an object of **RfGraph**$/A$. The *minimisation* of G is the labelled graph $MG = (MG, l_M)$ in **RfGraph**$/A$ where MG is the graph whose:

— nodes MG_0 are the quotient set of the nodes G_0 of G by $\approx_G$;

— edges MG_1 are the quotient of the edges G_1 of G by $\approx_G^e$;

— the reflexivity r_{MG} sends a node $[g]$ in MG_0 to the $\approx_G^e$ class $[r_G(g)]$ of the null edge at g;

— the labelling is defined by $(l_M)_0([g]) = l_0(g)$ and $(l_M)_1([e]) = l_1(e)$ where $[g]$ is the $\approx_G$ class of the node g in G and $[e]$ is the $\approx_G^e$ class of an edge e.

The *projection morphism* $\pi_G : G \longrightarrow MG$ in **RfGraph**$/A$ is defined on nodes and edges of G by $(\pi_G)_0(g) = [g]$, $(\pi_G)_1([e]) = e$.

As in Section 2, the resulting MG is a transition system, indeed it is a reflexive transition system. The projection morphism is well-defined and the minimisation process is idempotent. Moreover, the projection morphism is an example of the following concept.

Definition 4.3. Let G and H be objects of $\mathbf{RfGraph}/A$. An $\mathbf{RfGraph}/A$ morphism $\varphi : G \longrightarrow H$ is *reflexive path lifting* if whenever $e' : \varphi(g) = h \xrightarrow{a} h'$ is an a-labelled edge in H with source $\varphi(g)$ and target h', there are nodes $g_1, g_2, \dots g_k, g', k \geqslant 0, g'$ in G such that $\varphi(g_1) = \varphi(g_2) = \dots \varphi(g_k) = h$, $\varphi(g') = h'$ and a path $g \xrightarrow{-} g_1 \xrightarrow{-} \dots g_k \xrightarrow{a} g'$ in G such that its null-labelled edges are mapped by φ to the *null* edge at $\varphi(g)$ and the image of the edge $g_k \xrightarrow{a} g'$ is e' (and hence $\varphi(g') = h'$).

By similar considerations to those in the remarks after Definition 2.5, there is a correspondence between fully branching bisimulations and spans of reflexive path lifting morphisms, and a reflexive path lifting morphism determines a functional fully branching bisimulation.

Let $(A, *)$ be a pointed reflexive graph. Again, we do not require $(A, *)$ to be reachable. As in Section 2, we will write $\mathsf{PL}\mathbf{RfGraph}_A$ for the category in which an object $((G, g_0), l)$ is a pointed reflexive graph morphism $l : (G, g_0) \longrightarrow (A, *)$ from a reachable pointed reflexive graph (G, g_0) to $(A, *)$, and an arrow $\varphi : ((G, g_0), l) \longrightarrow (G', g_0'), l')$ is a reflexive path lifting morphism $\varphi : (G, g_0) \longrightarrow (G', g_0')$ of reachable pointed reflexive graphs such that $l'\varphi = l$.

Notation 4.1. $\mathsf{PL}\mathbf{RfGraph}_A$ has reachable pointed reflexive graphs labelled by $(A, *)$ as objects. Its morphisms are reflexive path lifting and label preserving.

Now the results of Section 2 can be repeated for the reflexive case. We summarise with the main points in the following proposition.

Proposition 4.1.

(i) If $G \xrightarrow{\varphi} H$ is a morphism of $\mathsf{PL}\mathbf{RfGraph}_A$, then $MG \overset{M\varphi}{\xrightarrow{\cong}} MH$.

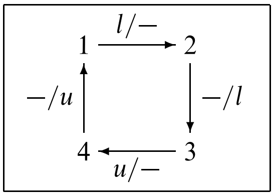
(ii) M is the functor part of an idempotent monad (M, π) on $\mathsf{PL}\mathbf{RfGraph}_A$, and so (M, π) has as Eilenberg–Moore algebras a full subcategory of $\mathsf{PL}\mathbf{RfGraph}_A$ denoted $\mathsf{MPL}\mathbf{RfGraph}_A$ (with objects of the form MG).

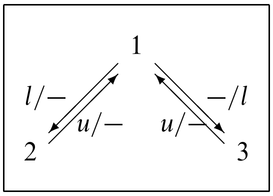
(iii) There is an adjunction $\mathsf{M}_A \dashv \mathsf{I}_A : \mathsf{MPL}\mathbf{RfGraph}_A \longrightarrow \mathsf{PL}\mathbf{RfGraph}_A$.

Furthermore, the results of Section 3 can also be extended to the reflexive case. The main point is the definition of a suitable bicategory of spans of reflexive graphs. In fact, all that is needed is to add reflexivity to the definitions of Section 3 and then note that Lemma 3.1 extends to the definition of reflexive path lifting arrow used in this section. With that we define the bicategory $\mathsf{SpPL}\mathbf{RfGraph}$ that has as objects pointed reflexive graphs and whose hom category $\mathsf{SpPL}\mathbf{RfGraph}(A, B)$ for pointed reflexive graphs A and B is $\mathsf{PL}\mathbf{RfGraph}_{A \times B}$. Then we have the following proposition.

Proposition 4.2. Defining $M_{(A,B)} : \mathsf{SpPL}\mathbf{RfGraph}(A, B) \longrightarrow \mathsf{SpPL}\mathbf{RfGraph}(A, B)$ by $M_{(A,B)} = M : \mathsf{PL}\mathbf{RfGraph}_{A \times B} \longrightarrow \mathsf{PL}\mathbf{RfGraph}_{A \times B}$, we obtain an idempotent lax monad on $\mathsf{SpPL}\mathbf{RfGraph}$.

Example 4.1. As examples of morphisms in $\mathsf{SpPL}\mathbf{RfGraph}$, consider the labelled reflexive graphs $\mathscr{P}$ and $\mathscr{F}$ below. In both cases, the label alphabets on both left and right are the null label denoted $-$ and l and u standing for 'lock' and 'unlock', respectively. In the diagrams null edges are omitted and the left and right labellings are separated by $/$.





$\mathscr{P}$ $\mathscr{F}$

The Dining Philosopher Problem arises from the idea that Philosophers $\mathscr{P}$ are seated around a table with Forks $\mathscr{F}$ between them. State 1 is the start state for both the Philosopher $\mathscr{P}$ and the Fork $\mathscr{F}$. According to the specification of the labelled graph $\mathscr{P}$ above, a Philosopher may pick up (or lock, l) the Fork to the left (the transition $l/-$) or right (transition $-/l$), and then put down (or unlock, u) the Fork to the left and then to the right. That is, the left fork must be picked up first (reaching state 2), then the right (so state 3 means left and right Forks have been picked up). Of course idling steps may always intervene. (At this point we imagine the Philosopher may eat, albeit with two Forks!) Then, the right fork must be put down first. A Fork is either free (unlocked) or in use (locked). State 1 of the Fork means it is free, 2 means it is locked (that is, has been picked up) by the Philosopher to its left, and 3 means the Fork is locked by the Philosopher to its right.

The Dining Philosopher Problem is discussed further in Example 7.1, but for now we note that the system consisting of n Dining Philosophers in a row, each with a Fork to the right is represented by the expression $(\mathscr{P}\mathscr{F})^n$. For example, the $\mathsf{SpPL}\mathbf{RfGraph}$ composite $\mathscr{P}\mathscr{F}$ expresses the possible actions of a Philosopher with a Fork to the right, and so on. Using the operation of *feedback* Fb (which we introduce in Section 7) we can represent the system of n Dining Philosophers 'in a circle' as $\mathsf{Fb}((\mathscr{P}\mathscr{F})^n)$.

5. The setting for minimal realisation

For consideration of minimal realisation in the rest of the paper we will work with various restrictions on the categories and functors that we describe in this section. Our purpose here is to underline the generality of our setting. Up to Proposition 5.3 this is well-known material, much of which can be found in Kasangian and Vigna (1997).

The base category for our discussion is the category of (directed) graphs **Graph** described at the beginning of Section 2.

The functor $V : \mathbf{Graph} \longrightarrow \mathbf{Set}$ that remembers only the nodes has both left and right adjoints denoted $D \dashv V \dashv H : \mathbf{Set} \longrightarrow \mathbf{Graph}$. For a set X, DX is the *discrete graph* on X with nodes X and no edges. HX is the *chaotic graph* on X with nodes X and exactly one edge in each direction between each pair of nodes. In this section we will be interested in the comma category $D/\mathbf{Graph}$. Thus, the objects of $D/\mathbf{Graph}$ are triples (X, f, G) with X

a set, G a graph, and f a **Graph** arrow $f : DX \longrightarrow G$. We call them *multi-pointed graphs*. An object of $D/$**Graph** should be thought of as a graph G with a set X of entry points specified by f (not necessarily all distinct). An arrow $\varphi : (X, f, G) \longrightarrow (X', f', G')$ is a pair $\varphi = (\varphi^0, \varphi^1)$ where $\varphi^0 : X \longrightarrow X'$ is a mapping and $\varphi^1 : G \longrightarrow G'$ is a **Graph** arrow such that $\varphi^1 f = f' D\varphi^0$ in **Graph**.

To describe behaviours, we will require the subcategory of $D/$**Graph** consisting of *trees*. The *category of trees* **Tree** is defined to be the full subcategory of $D/$**Graph** whose objects $(\{r\}, j, T)$ have a specified *root* node r in T_0, where j is the name of r, and there is a unique path in the graph T from r to each other node of T. Thus a morphism of trees takes root nodes to root nodes and the unique path from the root to any node of the domain graph maps to the unique path to its image node.

We digress to consider some facts about *forests*. Intuitively, a forest is just a set of trees. We define **Forest** to be the full subcategory of $D/$**Graph** whose objects are (X, f, F) where F is a graph with the property that for each node v of F there is a unique element r_v of X and unique path in F from $f(r_v)$ to v. As a consequence of the definition, f is one–one. We denote the inclusion functor J : **Forest** $\longrightarrow D/$**Graph**.

An equivalent way to describe forests is by the functor category **Forest**$'$ = **Set**$^{0\leftarrow 1\leftarrow\cdots}$. An object F' of **Forest**$'$ defines a forest F as follows. $F'(0)$ specifies the root nodes of the trees. The nodes F_0 of F are the disjoint union of the $F'(i), i \geqslant 0$. The children of the root nodes are the set $F'(1)$ with parentage given by the transition mapping $F'(1) \longrightarrow F'(0)$, and so on. So there is an edge from v in $F'(n)$ to v' in $F'(n+1)$ (note direction!) exactly when v is the image of v' under the transition mapping $F'(n+1) \longrightarrow F'(n)$. Each node v in F_0 has a unique path to it from a unique node in $F'(0)$ specified by the iterated transition function acting on v. We use $j : F'(0) \longrightarrow F_0$ to denote the inclusion function. Then $(F'(0), j, F)$ is an object of $D/$**Graph** that is a forest as defined in the previous paragraph. Any arrow $\varphi : F_0 \longrightarrow F_1$ of **Forest**$'$ evidently defines a unique morphism of the corresponding multi-pointed graphs. Conversely, suppose ψ : $(F_0(0), j_0, F) \longrightarrow (F_1(0), j_1, F')$ is a morphism between multi-pointed graphs that arise from objects F_0 and F_1 of **Forest**$'$. Since $\psi_0|_{F_0(0)} : F_0(0) \longrightarrow F_1(0)$, the definition of edges in the corresponding graph means we have $\psi_0|_{F_0(1)} : F_0(1) \longrightarrow F_1(1)$, and so on. The requirements for a $D/$**Graph** arrow mean that ψ determines an arrow from F to F' in the functor category **Forest**$'$. Since these correspondences are mutually inverse up to isomorphism, **Forest**$'$ is equivalent to **Forest**. Thus we have the following well-known proposition.

Proposition 5.1. The category of forests **Forest** is a topos.

To a multi-pointed graph (X, f, G) (which we will often abuse notation by writing just f) there is associated a forest Uf, the *unfolding of* f. This is the forest (X, f, Uf_G) whose graph Uf_G consists of the trees built from paths in G originating from the root nodes specified by f. More precisely, and to establish notation, we will denote the nodes of the graph of Uf_G by $\langle xe_1e_2...e_n\rangle$, where $n \geqslant 0$, $v = f(x)$ is an entry node and $e_1, e_2, \ldots, e_n$ are the edges in a path in G with source v. The edges of Uf_G are denoted $\langle p\rangle e$ where $\langle p\rangle$ is a node of Uf_G and e is an edge whose source is the last node on the path p. If

$$\varphi = (\varphi^0, \varphi^1) : (X, f, G) \longrightarrow (X', f', G')$$

is an arrow of $D/\mathbf{Graph}$, we define

$$U\varphi : (X, f, Uf_G) \longrightarrow (X', f', Uf'_{G'})$$

by $(U\varphi)^0 = \varphi^0$, on nodes of Uf_G by

$$(U\varphi)^1_0(\langle xe_1 \ldots e_n\rangle) = \langle \varphi^0(x)\varphi^1_1(e_1)\ldots\varphi^1_1(e_n)\rangle$$

and on edges by

$$(U\varphi)^1_1(\langle p\rangle e) = (U\varphi)^1_0(\langle p\rangle)\varphi^1_1(e).$$

These define a functor $U : D/\mathbf{Graph} \longrightarrow \mathbf{Forest}$. Now we have the following proposition.

Proposition 5.2.

$$J \dashv U : D/\mathbf{Graph} \longrightarrow \mathbf{Forest}$$

Proof. Suppose f is a multi-pointed graph and F is a forest. To any arrow φ (in **Forest**) from F to Uf we can define a unique arrow from JF to f by simply sending the edges of (the graph part of) JF to the edges determined by φ. Conversely, remembering that entry points must be preserved, it is easy to see that an arrow ψ from JF to f determines a unique arrow from the forest F to the unfolding of f. $\square$

We should point out that for a multi-pointed graph (*even if finite*) that has a cycle reachable from a base point, the unfolding is a forest of countably infinite depth. As an example, consider the second minimised graph in Example 2.1.

The adjunction $D \dashv V$ means that there is an isomorphism of categories between $D/\mathbf{Graph}$ and $\mathbf{Set}/V$. Since V has a right adjoint, it is a left exact functor and so $\mathbf{Set}/V$ is a topos by the glueing construction (Wraith 1974). Thus $D/\mathbf{Graph}$ is also a topos. The functor J above is not left exact since the terminal object in **Forest** ($\sim$ **Forest**$'$!) is the tree with exactly one node at each level, while the terminal object of $D/\mathbf{Graph}$ has a graph with exactly one node and one edge. However, J does preserve pullbacks, so U is a *partial geometric morphism* as considered in Rosebrugh and Wood (1991).

Recall the category of *pointed graphs* **PtGraph**. It is the full subcategory of $D/\mathbf{Graph}$ with objects $(1, p, G)$ where $p : D1 \longrightarrow G$ is the 'point'. The category of trees **Tree** defined above is a full subcategory of **PtGraph**. Moreover, we have the following proposition.

Proposition 5.3. The functors J and U restrict to $J^{pt} : \mathbf{Tree} \longrightarrow \mathbf{PtGraph}$ and $U^{pt} : \mathbf{PtGraph} \longrightarrow \mathbf{Tree}$, and $J^{pt} \dashv U^{pt}$.

Given an object $f = (X, f, G)$ of $D/\mathbf{Graph}$, we construct the *reachable part of* f, $f_R = (X, f_R, G_R)$ as follows. G_R is the full subgraph of G whose nodes are those v such that there is a path to v from a node $v_0 = f(x)$ in the image of f. Such nodes are called *reachable nodes* of G. The function f_R is just the factorisation of f through the inclusion of G_R in G, and this clearly exists since any node in the image of f is trivially reachable.

Proposition 5.4. The correspondence $f \mapsto f_R$ is the object part of an idempotent functor R on $D/\mathbf{Graph}$. Define $\rho_f : f_R \longrightarrow f$ by the inclusion of G_R in G. Then ρ is a natural transformation from R to the identity functor on $D/\mathbf{Graph}$ and (R, ρ) is an idempotent comonad.

Proof. The idempotence of R as a functor is obvious. Using the fact that the putative counit for the monad is defined by the inclusion of G_R in G makes verification of the comonad equations trivial. $\square$

The coalgebras for (R, ρ), denoted $D/\mathbf{Graph}_R$, are graphs all of whose nodes are reachable by a path from an entry point. Note that $J : \mathbf{Forest} \longrightarrow D/\mathbf{Graph}$ factors through $D/\mathbf{Graph}_R$ and that, for any f in $D/\mathbf{Graph}$, Uf is reachable. It is evident that the comonad (R, ρ) restricts to the subcategory of pointed graphs $\mathsf{Pt}\mathbf{Graph}$. The coalgebras $\mathsf{RchPt}\mathbf{Graph}$ for the restricted comonad is the subcategory of *reachable pointed graphs* also considered above in Section 2.

The category of multi-pointed graphs has pullbacks, so there is a bicategory of spans, which we denote $\mathbf{Span}(D/\mathbf{Graph})$. We note that (R, ρ) extends to an identity-on-objects colax functor (= bicategory comorphism) on the bicategory $\mathbf{Span}(D/\mathbf{Graph})$. It is defined locally on a span $(X, f, G) \longleftarrow (Y, h, H) \longrightarrow (X', f', G')$ of multi-pointed graphs by taking the reachable part of the head graph (Y, h, H) and composing its inclusion $R(Y, h, H) \longrightarrow (Y, h, H)$ with the legs of the original span. The colax structure at the identity span is given by the counit of (R, ρ). To define the colaxity on a composite requires only the observation that the reachable part of a composite in $\mathbf{Span}(D/\mathbf{Graph})$ is contained in the composite of the reachable parts of the factors.

Furthermore, (R, ρ) is an idempotent colax comonad (in the sense of Carboni and Rosebrugh (1991)) on the bicategory $\mathbf{Span}(D/\mathbf{Graph})$. As in Proposition 3.2 above, since the local comonads are idempotent, we use Proposition 42 in Rosebrugh *et al.* (1998) to simplify the equations required for the colax comonad structure. In fact, all we need to show is that the counit is compatible with the colaxity for composition, but both are defined by taking the reachable part.

Thus, there is a bicategory of coalgebras $\mathbf{Span}(D/\mathbf{Graph})_R$ with the same objects as $\mathbf{Span}(D/\mathbf{Graph})$ and local coalgebras given by the reachability comonad. Notice that composition in the coalgebras $\mathbf{Span}(D/\mathbf{Graph})_R$ is obtained by taking the *reachable part* of the usual composite in $\mathbf{Span}(D/\mathbf{Graph})$. It is worth pointing out again that the identity arrow on an object $G = (X, f, G)$ is actually the span whose legs are the inclusions of the reachable part of G. We use $\mathsf{SpRchPt}\mathbf{Graph}$ to denote the locally full sub-bicategory of the coalgebras $\mathbf{Span}(D/\mathbf{Graph})_R$ determined by objects in $\mathsf{Pt}\mathbf{Graph}$ and 1-cells (spans) whose heads are reachable pointed graphs. Of course, $\mathsf{SpRchPt}\mathbf{Graph}$ is also describable as the bicategory of coalgebras for the reachability colax comonad restricted to spans of pointed graphs. The (not locally full) sub-bicategory of $\mathsf{SpRchPt}\mathbf{Graph}$ whose 2-cells are path lifting was denoted $\mathsf{SpPL}\mathbf{Graph}$ above.

6. Behaviour and minimal realisation

In order to discuss minimal realisation locally, we need a category of machines, a category of behaviours and a behaviour functor. In Section 2 we introduced the machine category we wish to consider, $\mathsf{PL}\mathbf{Graph}_A$. We need to define what we mean by behaviour of an object in $\mathsf{PL}\mathbf{Graph}_A$. As we will detail shortly, we consider that a certain equivalence class of synchronisation trees is the appropriate behaviour for a reachable pointed graph labelled by A. Our objective in this section is to define a behaviour functor on $\mathsf{PL}\mathbf{Graph}_A$ and a right adjoint minimal realisation functor to $\mathsf{PL}\mathbf{Graph}_A$ from the behaviours.

The theory of concurrency considers special labelled trees known as synchronisation trees.

Definition 6.1. (Joyal *et al.* 1996) Let A be an alphabet. A *synchronisation tree* labelled by A with root r is a labelled transition system with initial state r whose underlying graph is a tree.

Thus the underlying graph of the tree is reachable and acyclic, and each non-initial state is the target of a unique edge. As also noted in Joyal *et al.* (1996), a labelled transition system with an initial state unfolds to a synchronisation tree, and the inclusion-unfolding adjunction of Section 5 extends to an adjunction between the category of transition systems labelled by A and the synchronisation trees labelled by A. (Note that in Joyal *et al.* (1996), the label set may vary along a transition system homomorphism and the image of a transition may 'idle' in the codomain.)

Let $(A, *)$ be a pointed graph. We use $\mathsf{PL}\mathbf{Tree}_A$ to denote the full subcategory of $\mathsf{PL}\mathbf{Graph}_A$ with objects whose domains are trees. Thus the objects of $\mathsf{PL}\mathbf{Tree}_A$ are trees labelled by A with path lifting morphisms, which are then necessarily onto on nodes and edges.

Proposition 6.1. The functors J^{pt} and U^{pt} of Proposition 5.3 extend to

$$J^{pt} \dashv U^{pt} : \mathsf{PL}\mathbf{Graph}_A \longrightarrow \mathsf{PL}\mathbf{Tree}_A.$$

Proof. This only requires the observation that the labelled unfolding of a path lifting morphism of pointed graphs is a path lifting morphism of trees. □

Example 6.1. The unfolding of any object of $\mathsf{PL}\mathbf{Graph}_A$ is a synchronisation tree labelled by A. Indeed, the definition of unfolding guarantees that there are never two edges with the same label between nodes of the unfolding – after all, a tree has at most one edge between two nodes! Of course, every synchronisation tree arises, up to isomorphism, as its own unfolding.

The equivalence relation on objects of $\mathsf{PL}\mathbf{Tree}_A$ that we have been alluding to is simply bisimilarity. As shown in Joyal *et al.* (1996, Theorem 2), trees labelled by A are bisimilar if and only if they are linked by a span of path lifting arrows. Since $\mathsf{PL}\mathbf{Tree}_A$ has pullbacks, trees are linked by a span of path lifting arrows if and only if they are linked by a zig-zag of path lifting arrows if and only if they are in the same *connected component* of $\mathsf{PL}\mathbf{Tree}_A$. Now the behaviour we will define for an object of $\mathsf{PL}\mathbf{Graph}_A$ is an observational, that is bisimilarity, equivalence class of trees labelled by A, and the class associated to an object (G, l) is precisely that of its unfolding synchronisation tree. By the remarks at the beginning of this paragraph, this is the connected component of the unfolding of (G, l).

Recall the connected components functor defined on the category of categories: $\Pi_0 : \mathbf{Cat} \longrightarrow \mathbf{Set}$. It has a right adjoint D assigning the discrete category to a set, so for any category $\mathbf{C}$ there is a universal functor $P_{\mathbf{C}} : \mathbf{C} \longrightarrow D\Pi_0(\mathbf{C})$.

Notation 6.1. We use $\mathbf{Beh}_A$ to denote the (discrete) category $D\Pi_0(\mathsf{PL}\mathbf{Tree}_A)$ of behaviours.

We can now define the behaviour functor.

Definition 6.2. The *behaviour functor* E on $\mathsf{PL}\mathbf{Graph}_A$ is defined by

$$E = P_{\mathsf{PL}\mathbf{Tree}_A} U^{pt} : \mathsf{PL}\mathbf{Graph}_A \longrightarrow \mathbf{Beh}_A.$$

Our next objective is to define a minimal realisation functor N that is right adjoint to E. We will also find below that $\mathbf{Beh}_A$ is equivalent to $\mathsf{MPL}\mathbf{Graph}_A$, essentially *via* the behaviour and minimisation functors. We note immediately the functor $E' = E\mathsf{I}_A$: $\mathsf{MPL}\mathbf{Graph}_A \longrightarrow \mathbf{Beh}_A$. Moreover, the projection arrow $\pi_G : G \longrightarrow MG$ shows that G and MG are in the same component of $\mathsf{PL}\mathbf{Graph}_A$, whence $EG = EMG = E\mathsf{I}_A\mathsf{M}_A G = E'\mathsf{M}_A G$, and so $E = E'\mathsf{M}_A$.

Let $P(T) = P_{\mathsf{PL}\mathbf{Tree}_A}(T)$ be the component of an A labelled tree T in $\mathsf{PL}\mathbf{Tree}_A$. For each component c, choose a tree T_c in $P(T)$.

Definition 6.3. The *minimal realisation functor* N : $\mathbf{Beh}_A \longrightarrow \mathsf{PL}\mathbf{Graph}_A$ is defined by $N(P(T)) = MJ^{pt}(T_c)$.

This definition is independent of the choice of T_c since minimisation M is (essentially) constant on components of $\mathsf{PL}\mathbf{Graph}_A$.

For the rest of this section we omit the superscripts on U and J. The functorial property we expect of minimal realisation is the following proposition.

Proposition 6.2. There is an adjunction $E \dashv N : \mathbf{Beh}_A \longrightarrow \mathsf{PL}\mathbf{Graph}_A$.

Proof. First, suppose that we have a $\mathsf{PL}\mathbf{Graph}_A$ arrow $G \longrightarrow N(P(T)) = MJ(T_c)$. We want to show that there is an arrow $E(G) \longrightarrow P(T)$. Indeed, we have $E(G) = P(UG) = P(T)$ since we have arrows

$$UG \longrightarrow U\mathsf{I}_A\mathsf{M}_A J(T_c) \longleftarrow T_c$$

where the second arrow is the unit for $\mathsf{M}_A J \dashv U\mathsf{I}_A$. Hence $E(G) = P(UG) = P(T_c) = P(T)$. In the other direction, suppose that $E(G) = P(T)$ (since $\mathbf{Beh}_A$ is discrete). We need to find an arrow $G \longrightarrow N(P(T))$. By definition of E, we have $P(UG) = E(G) = P(T) = P(T_c)$, so in $\mathsf{PL}\mathbf{Tree}_A$ we have a span $UG \longleftarrow T' \longrightarrow T_c$. Using the counit for $J \dashv U$ to provide an arrow $JUG \longrightarrow G$ and applying MJ to the span just mentioned, in $\mathsf{PL}\mathbf{Graph}_A$ we have

$$G \xrightarrow{\pi_G} M(G) \xleftarrow{\cong} MJU(G) \xleftarrow{\cong} MJ(T') \xrightarrow{\cong} MJ(T_c) = N(P(T))$$

providing $G \longrightarrow N(P(T))$, as required. It remains to show that these passages are mutually inverse. Beginning from $E(G) \xrightarrow{=} P(T)$, we defined an arrow $G \longrightarrow N(P(T))$, and thence an arrow $E(G) \longrightarrow P(T)$ that must be the original equality arrow by discreteness. For the other required equation, starting from $G \xrightarrow{\varphi} N(P(T)) = MJ(T_c)$, the first process produces the cospan $UG \xrightarrow{U\varphi} UMJ(T_c) \longleftarrow T_c$. By pulling back, we get a span $UG \longleftarrow T' \longrightarrow T_c$ to which the second process applies yielding an arrow $G \longrightarrow MJ(T_c) = N(P(T))$. This latter arrow has a minimised object as codomain, and so by Lemma 2.10 it must equal φ. □

Note that $N(P(T)) = MJ(T_c)$ is minimised, so we can factor N through $\mathsf{MPL}\mathbf{Graph}_A$ as $N = \mathsf{I}_A N'$ where $N'(P(T)) = \mathsf{M}_A J(T_c)$.

Remark 6.1. The situation is summed up in the following diagram of categories and functors in which we have M_A and E are left adjoints, $\mathsf{I}_A N' = N$, $E = E'\mathsf{M}_A$.

$$\begin{tikzcd}
& \mathsf{PL}\mathbf{Graph}_A \arrow[dl, shift right, "\mathsf{M}_A"'] \arrow[dr, shift right, "E"'] & \\
\mathsf{MPL}\mathbf{Graph}_A \arrow[ur, shift right, "\mathsf{I}_A"'] \arrow[rr, shift left, "E'"] & & \mathbf{Beh}_A \arrow[ul, shift right, "N"'] \arrow[ll, shift left, "N'"]
\end{tikzcd}$$

It is also important to note the following proposition.

Proposition 6.3. $\mathbf{Beh}_A$ and $\mathsf{MPL}\mathbf{Graph}_A$ are equivalent.

Proof. For T in $\mathsf{PL}\mathbf{Tree}_A$ we have

$$E'N'(P(T)) = E'(\mathsf{M}_A J(T_c)) = E\mathsf{I}_A\mathsf{M}_A J(T_c) = P(U\mathsf{I}_A\mathsf{M}_A J(T_c)),$$

but since we have the unit $T_c \longrightarrow U\mathsf{I}_A\mathsf{M}_A J(T_c)$, the component of $U\mathsf{I}_A\mathsf{M}_A J(T_c)$ is $P(T_c) = P(T)$ and $E'N'$ is the identity.

In the other direction, for G in $\mathsf{MPL}\mathbf{Graph}_A$ we have

$$N'E'(G) = N'(P(U\mathsf{I}_A G)) = \mathsf{M}_A J((U\mathsf{I}_A G)_c).$$

Now, since $U\mathsf{I}_A G$ and $(U\mathsf{I}_A G)_c$ are linked by a span in $\mathsf{PL}\mathbf{Tree}_A$, we have

$$\mathsf{M}_A J((U\mathsf{I}_A G)_c) \cong \mathsf{M}_A J U\mathsf{I}_A(G)$$

and the counit for $\mathsf{M}_A J \dashv U\mathsf{I}_A$ provides $\mathsf{M}_A J U\mathsf{I}_A(G) \cong G$, so $N'E'(G) \cong G$. $\square$

With the results of Section 3, this proposition shows that our minimal realisation is also compositional. We have a diagram of bicategories, local adjunctions defined by the (equivalent) lax monads M and NE, and the (bi)equivalence in the bottom row of

$$\begin{tikzcd}
& \mathsf{SpPL}\mathbf{Graph} \arrow[dl, shift right, "\mathsf{M}"'] \arrow[dr, shift right, "E"'] & \\
\mathsf{SpMPL}\mathbf{Graph} \arrow[ur, shift right, "\mathsf{I}"'] \arrow[rr, shift left, "E'"] & & \mathsf{Sp}\mathbf{Beh} \arrow[ul, shift right, "N"'] \arrow[ll, shift left, "N'"]
\end{tikzcd}$$

where $\mathsf{Sp}\mathbf{Beh}$ is the locally discrete bicategory with the same objects as $\mathsf{SpPL}\mathbf{Graph}$ and $\mathsf{Sp}\mathbf{Beh}(A, B) = \mathbf{Beh}_{A\times B}$. Thus we have reached the goal announced in the Introduction.

Remark 6.2.

(i) There is no obstruction to extending the results of this section to reflexive graphs, reflexive path lifting morphisms and fully branching bisimulation. This would require modification of the codomain of the unfolding functor to be a category of reflexive graphs that are trees, apart from the fact that they have a reflexive node. Behaviours would then be reflexive path lifting components (fully branching bisimulation equivalence classes) of these.

(ii) We could extend our model to include final states and recover language recognisers. In that case bisimulations should respect final states in the sense that a final state may only be bisimilar to a final state and a non-final state to a non-final one. Morphisms must then also preserve and reflect the final states. Once again we can define a minimisation using the quotient by a maximal self-bisimulation of the extended type. With this setup we can recover the classical minimal realisation of Nerode as follows. For an alphabet A, a language is a subset of the words over A. It can be recognised by the tree that has an edge labelled by each letter of A at each node and final states given by selecting the nodes that correspond to paths labelled by words in the language. Of course, this tree is the unfolding of *any* deterministic labelled graph (automaton) that recognises the language. Now the minimisation described above, when applied to this tree, gives the classical minimal realisation of Nerode.

7. Other structures

In this section we consider the tensor structure on spans of graphs and its relationship to our minimal realisation functors, and apply this to the concept of feedback and the Dining Philosopher Problem. We also consider an extension of the results of the previous section to multi-pointed graphs and forests.

Definition 7.1. The *tensor* of 1-cells $G : A \xleftarrow{l} G \xrightarrow{r} B$ and $G' : C \xleftarrow{l'} G' \xrightarrow{r'} D$ in $\mathsf{SpPL}\mathbf{Graph}$ is the span

$$G \otimes G' : A \times C \xleftarrow{l''} (G \times G')_R \xrightarrow{r''} B \times D$$

where l'' is the inclusion of $(G \times G')_R$ in $G \times G'$ followed by $l \times l'$, and similarly for r''.

Definition 7.2. For any pointed graph A, we use η_A to denote the span

$$\eta_A : 1 \longleftarrow A_R \xrightarrow{\overline{\Delta}_R} A \times A$$

where $\overline{\Delta}_R$ is the inclusion of the reachable part followed by the diagonal. Similarly, we use ϵ_A to denote the span

$$\epsilon_A : A \times A \xleftarrow{\overline{\Delta}_R} A_R \longrightarrow 1.$$

Before considering our application we note the following proposition.

Proposition 7.1. For any pointed graph A, in $\mathsf{SpPL}\mathbf{Graph}$ we have

$$M_{(A,A)} 1_A = 1_A \qquad M_{(1,A\times A)} \eta_A = \eta_A \qquad M_{(A\times A,1)} \epsilon_A = \epsilon_A.$$

Proof. These are all similar. The uniqueness of labels (on both sides for 1_A, on the right for η_A, on the left for ϵ_A) means that the largest self-bisimulation in each case is the identity relation, so minimisation is trivial. □

Proposition 7.2. For morphisms $G : A \xleftarrow{l} G \xrightarrow{r} B$ and $G' : C \xleftarrow{l'} G' \xrightarrow{r'} D$ in $\mathsf{SpPL}\mathbf{Graph}$ there is a comparison arrow

$$\mu_{G.G'} : MG \otimes MG' \longrightarrow M(G \otimes G').$$

Proof. This merely requires the observation that a node on the left is a pair of equivalence classes in (the reachable part of) the product of the quotient of G and G' by their largest self-bisimilarities that may be sent to the equivalence class of the same pair in $M(G \otimes G')$. □

Definition 7.3. The *feedback* of a 1-cell $G : A \times U \xleftarrow{l} G \xrightarrow{r} B \times U$ in $\mathsf{SpPL}\mathbf{Graph}(A \times U, B \times U)$ is $\mathsf{Fb}_\mathsf{R}(G)$ in $\mathsf{SpPL}\mathbf{Graph}(A, B)$ defined as the composite (in $\mathsf{SpPL}\mathbf{Graph}$):

$$\mathsf{Fb}_\mathsf{R}(G) = (1_B \otimes \epsilon_U)(G \otimes 1_U)(1_A \otimes \eta_U).$$

We have used the subscript R in the definition to emphasise the fact that the operations involved in the expression for feedback involve taking reachable parts. The definition of feedback is the same for **Span**(**Graph**), here modified to take account of path lifting morphisms. See Katis *et al.* (1997b) for examples and more details. For an account of the universal properties of feedback, in particular, the relationship with traced monoidal categories, see Katis *et al.* (2002).

Proposition 7.3. For G as in the definition, there is a comparison arrow

$$\mathsf{Fb}_\mathsf{R} M(G) \longrightarrow M(\mathsf{Fb}_\mathsf{R}(G)).$$

Proof. Combine the definition of feedback with Propositions 7.1 and 7.2 and the fact that M is lax to obtain:

$$\begin{aligned}
\mathsf{Fb}_\mathsf{R} M(G) &= (1_B \otimes \epsilon_U)(MG \otimes 1_U)(1_A \otimes \eta_U) \\
&\cong (M1_B \otimes M\epsilon_U)(MG \otimes M1_U)(M1_A \otimes M\eta_U) \\
&\longrightarrow M(1_B \otimes \epsilon_U)M(G \otimes 1_U)M(1_A \otimes \eta_U)) \\
&\longrightarrow M((1_B \otimes \epsilon_U)(G \otimes 1_U)(1_A \otimes \eta_U)) \\
&= M(\mathsf{Fb}_\mathsf{R}(G))
\end{aligned}$$ □

Corollary 7.1.

$$M\mathsf{Fb}_\mathsf{R} M(G) \cong M\mathsf{Fb}_\mathsf{R}(G).$$

All of these results extend to the reflexive case, so we can consider the following example.

Example 7.1. (**The Dining Philosophers**) A *deadlock state* of an asynchronous system is one for which the only available transitions are null. Deadlock states are preserved by our reflexive path lifting morphisms in the sense of Section 4, hence in searching for deadlocks it is often useful to calculate minimisation.

The following is a compositional model checking algorithm based on this idea:
Given a system presented as an expression in the operations of composition, tensor and feedback, calculate M of the system by successively evaluating an operation and calculating M of the result. Record the quotient morphisms π that arise in the calculation. When M of the system is obtained, find all deadlock states, and then trace inverse images that are deadlocks along the calculated π's.

When we apply the algorithm to the Dining Philosopher Problem, namely to detect deadlock in a circle of Philosophers and Forks as introduced above (see Example 4.1), we obtain the following results:

— $M(\mathscr{P} \cdot \mathscr{F})$ is a five node graph with 15 edges,
— $M(\mathscr{P} \cdot \mathscr{F} \cdot \mathscr{P} \cdot \mathscr{F})$ may then be evaluated as $M(M(\mathscr{P} \cdot \mathscr{F}) \cdot M(\mathscr{P} \cdot \mathscr{F}))$ with 8 nodes and 36 edges,
— when $M(\mathscr{P} \cdot \mathscr{F} \cdot \mathscr{P} \cdot \mathscr{F} \cdot \mathscr{P} \cdot \mathscr{F})$ is similarly evaluated we find that

$$M(\mathscr{P} \cdot \mathscr{F} \cdot \mathscr{P} \cdot \mathscr{F} \cdot \mathscr{P} \cdot \mathscr{F}) \cong M(\mathscr{P} \cdot \mathscr{F} \cdot \mathscr{P} \cdot \mathscr{F}),$$

— hence $M((\mathscr{P} \cdot \mathscr{F})^n) \cong M((\mathscr{P} \cdot \mathscr{F})^2)$ for all $n \geqslant 2$.

Thus the time complexity to calculate $M((\mathscr{P} \cdot \mathscr{F})^n)$ is linear. Note that similar results hold for $M(\mathscr{F} \cdot \mathscr{P})$, and so on.

The last stage of computing the minimisation of the Dining Philosophers is to calculate $\mathsf{Fb}(M((\mathscr{P} \cdot \mathscr{F})^n))$ and minimise the result yielding the terminal graph as expected.

Retrieving the deadlocks requires tracing back through π's considering only deadlock states along the way. Note that the single state of the final minimisation is a deadlock, but only one of its inverse images in $\mathsf{Fb}M((\mathscr{P} \cdot \mathscr{F})^n)$ is. In tracing back along the π's at each stage there is only one deadlock.

Finally, we also wish to extend the minimisation, behaviour, minimal realisation theory above from pointed to multi-pointed graphs and (for behaviour) from trees to forests. The material in Section 5 allows us to do this immediately, and we simply outline the results.

Let A be a multi-pointed graph. The category of multi-pointed graphs labelled by A is $(D/\mathbf{Graph})/A$. For a pair of objects of $(D/\mathbf{Graph})/A$ we continue to define bisimulation as in Section 2. Bisimulations again enjoy properties that provide a maximal self-bisimulation. Recall the discussion of reachability for multi-pointed graphs in Section 5. Reachability means existence of a path from at least one entry point, and we defined the coalgebra category $D/\mathbf{Graph}_R$. Since the notion of path lifting morphism also makes sense for multi-pointed graphs labelled by A, we have a category whose objects are reachable multi-pointed graphs labelled by A and whose arrows are path lifting that we will denote $\mathsf{PLD}\mathbf{Graph}_A$. (As always, A need not be reachable.) We obtain a minimisation functor on $\mathsf{PLD}\mathbf{Graph}_A$ just as in Section 2. Once again, the minimisation is an idempotent monad on $\mathsf{PLD}\mathbf{Graph}_A$ with algebras denoted $\mathsf{MPLD}\mathbf{Graph}_A$.

To make the minimisation just described compositional, we need a composition of spans from $\mathsf{PLD}\mathbf{Graph}_A$. As in Section 3, this composition may be derived from composition in $\mathbf{Span}((D/\mathbf{Graph})/A)$, the latter being a bicategory since $D/\mathbf{Graph}$, as mentioned above, is a topos so the slice category $(D/\mathbf{Graph})/A$ certainly has pullbacks. Restricting to appropriate head graphs and path-lifting two-cells, we obtain the bicategory $\mathsf{SpPLD}\mathbf{Graph}$ with objects multi-pointed graphs and hom categories defined by $\mathsf{SpPLD}\mathbf{Graph}(A, B) = \mathsf{PLD}\mathbf{Graph}_{A\times B}$. Minimisation is again a lax monad on $\mathsf{SpPLD}\mathbf{Graph}$ and provides a compositional minimisation for reachable multi-pointed labelled graphs.

We can also immediately extend the minimal realisation constructions of Section 6 but now with bisimilarity classes of labelled forests for behaviour categories as summed up in the following diagram

$$\begin{tikzcd}
 & \mathsf{PLD}\mathbf{Graph}_A \arrow[dl, shift right, "\mathsf{M}_A"'] \arrow[dr, shift left, "E"'] & \\
\mathsf{MPLD}\mathbf{Graph}_A \arrow[ur, shift right, "\mathsf{I}_A"'] \arrow[rr, shift left, "E'"] & & \mathsf{D}\mathbf{Beh}_A \arrow[ul, shift left, "N"'] \arrow[ll, shift left, "N'"]
\end{tikzcd}$$

We leave the reader to formulate the analogue of the last diagram in Section 6.

References

Arnold, A. (1992) *Finite Transition Systems*, Prentice-Hall.

Attie, P. C. and Emerson, E. (1998) Synthesis of concurrent systems with many similar processes. *ACM Trans. on Prog. Lang. and Systems (TOPLAS)* **20** 51–115.

Bloom, S., Sabadini, N. and Walters, R. F. C. (1996) Matrices, machines and behaviors. *Applied Categorical Structures* **4** 343–360.

Bunge, M. and Fiore, M. (2000) Unique factorisation lifting functors and categories of linearly-controlled processes. *Mathematical Structures in Computer Science* **10** 137–163.

Carboni, A. and Rosebrugh, R. (1991) Lax monads. *Journal of Pure and Applied Algebra* **76** 13–32.

Clarke, E., Long, D. and McMillan, K. (1989) Compositional model checking. In: *Proceedings of the Fourth Annual IEEE Symposium on Logic in Computer Science* 353–362.

van Glabbeek, R. and Weijland, P. (1996) Branching time and abstraction in bisimulation semantics. *Journal of the ACM* **43** 555–600.

Goguen, J. A. (1972) Minimal realization of machines in closed categories. *Bulletin of the AMS* **78** 777–783.

Graf, S., Steffen, B. and Lüttgen, G. (1996) Compositional minimization of finite state systems using interface specifications. *International Journal of Formal Aspects of Computing* **8** 607–616.

Hennessey, M. and Milner, R. (1980) On observing nondeterminism and concurrency. In: Automata, languages and programming (Proc. Seventh Internat. Colloq., Noordwijkerhout). *Springer-Verlag Lecture Notes in Computer Science* **85** 299–309.

Joyal, A., Nielson, M. and Winskel, G. (1996) Bisimulation from open maps. *Information and Computation* **127** 164–185.

Kasangian, S. and Vigna, S. (1997) The topos of labelled trees: A categorical semantics for SCCS. *Fundamenta Informaticae* **32** 27–45.

Katis, P., Rosebrugh, R., Sabadini, N. and Walters, R. F. C. (2000) An automata model of distributed systems. Proceedings of the Workshop on Trace Theory and Code Parallelization, Universita degli Studi di Milano 125–144.

Katis, P., Sabadini, N. and Walters, R. F. C. (1997a) Bicategories of processes. *Journal of Pure and Applied Algebra* **115** 141–178.

Katis, P., Sabadini, N. and Walters, R. F. C. (1997b) Span(Graph): a categorical algebra of transition systems. In: Proceedings AMAST '97. *Springer-Verlag Lecture Notes in Computer Science* **1349** 322–336.

Katis, P., Sabadini, N. and Walters, R. F. C. (1997c) Representing P/T nets in Span(Graph). In: Proceedings AMAST '97. *Springer-Verlag Lecture Notes in Computer Science* **1349** 307–321.

Katis, P., Sabadini, N. and Walters, R. F. C. (1998) On partita doppia. *Theory and Applications of Categories* (to appear).

Katis, P., Sabadini, N. and Walters, R. F. C. (2000) On the algebra of systems with feedback and boundary. *Rendiconti del Circolo Matematico di Palermo* Serie II, Suppl. **63** 123–156.

Katis, P., Sabadini, N. and Walters, R. F. C. (2001) Classes of finite state automata for which compositional minimization is linear time (preprint).

Katis, P., Sabadini, N. and Walters, R. F. C. (2002) Feedback, trace and fixed-point semantics. *Theor. Informatics Appl.* **36** 181–194.

Rosebrugh, R., Sabadini, N. and Walters, R. F. C. (1998) Minimal realization in bicategories of automata. *Mathematical Structures in Computer Science* **8** 93–116.

Rosebrugh, R. and Wood, R. J. (1991) Pullback preserving functors. *Journal of Pure and Applied Algebra* **73** 73–90.

Wraith, G. (1974) Artin glueing. *Journal of Pure and Applied Algebra* **4** 345–348.