Span(Graph): A Categorical Algebra of Transition Systems

Piergiulio Katis,
School of Mathematics and Statistics,
University of Sydney, N.S.W. 2006, Australia
e_mail: katis_p@maths.usyd.edu.au

N. Sabadini,
Dipartimento di Scienze dell'Informazione,
Università di Milano, Via Comelico 39/41, I-20135 Milano MI, Italy
FAX: +39-2-55006276; e_mail: sabadini@dsi.unimi.it

R.F.C. Walters.
School of Mathematics and Statistics,
University of Sydney, N.S.W. 2006, Australia
e_mail: walters_b@maths.usyd.edu.au

15th May 1997

1 Introduction.

Structured transition systems, or non-deterministic automata, have been widely used in the specification of computing systems, including concurrent systems [A94],[Z85]. We describe here an algebra of transition systems, an algebra in fact already known to category theorists but without any consciousness of its relation to concurrency. The algebra is closely related to, and may be regarded as an extension of, the algebra of Arnold and Nivat [A94] (which book contains comparisons with other models of concurrency – Petri nets, process algebras). What it has in addition to Arnold and Nivat's algebra is that there is a geometry associated with the new algebra along the lines of Penrose's algebra of tensors [P71] (a subject much developed of late in relation to the geometry of manifolds, and quantum field theory [JS91], [JS93], [S95], [T95], [M95]). In this paper we demonstrate how this geometry reflects the geometry of distributed systems.

For category theorists we can say in a line what the algebra is: it is the discrete cartesian bicategory structure [CW87] on **Span(Graph)** [B67]. (The category of relations also has this structure and, in the equivalent form of allegories, it has been used in [BJ94] to give a semantics of Ruby [JS90].) The second section of the paper is devoted to describing the operations (though not the equations) of this algebra, and their geometric interpretation. In the third section we show how traditional transition systems fit into the algebra, and we describe a specification of a distributed system using the algebra. In the fourth section we prove some language theoretic results about the effect of operations in

Span(Graph) on behaviours; we study special systems (linearizable spans) for which interleaved behaviour is reasonable, and we give a semantics to Hoare's parallel operation as a derived operation in our algebra. The final section of the paper contains a precise definition of discrete cartesian bicategory, and we derive relations between feedback, composition, tensor and parallel (in the sense of Hoare) in such a bicategory.

Further developments of this algebra – an analogue of bisimulation [P81], model-checking, a semantics of CCS, and specific applications to concurrent systems will appear elsewhere. This work arose out of earlier study of bicategories of processes [BSW96], [KSW97], [K96], [SWW93], [SWW94] which itself arose from the study of distributive categories and imperative programming [W89], [W92a], [W92b], [KW93]. A computer program [GKW96] has been written by Robbie Gates for computing in **Span(Graph)** and hence for checking properties of concurrent systems expressed in this way.

Much of the notation in this paper originates in category theory; for this the reader may consult a text such as [M70] or [W92a].

We acknowledge the support for this project by the Australian Research Council, Italian MURST 40% and the Italian CNR.

2 Span(Graph).

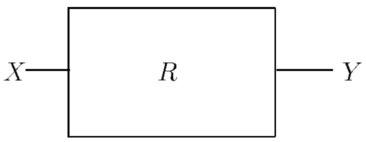
2.1 Objects and arrows of Span(Graph).

The objects of **Span(Graph)** are (finite) directed graphs. Given graphs X and Y an arrow from X to Y consists of a graph R and two graph morphisms $\partial_0 : R \to X, \partial_1 : R \to Y$. Such an arrow is called a span of graphs. It is often denoted as follows:

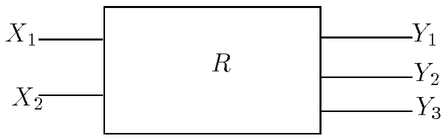
$$\begin{tikzcd} & R \arrow[dl, "\partial_0"'] \arrow[dr, "\partial_1"] & \\ X & & Y \end{tikzcd}$$

We call the graph R the head of the span and the morphisms ∂_0 and ∂_1 the two legs of the span. We will, by abuse of notation, denote the two legs of any span by the same symbols ∂_0 and ∂_1. We will often denote the span simply as $R : X \to Y$. We call X and Y the domain and codomain, respectively, of R; or the boundaries of R.

We represent a span $R : X \to Y$ by a picture of the following form:



If the objects are given as products of graphs, for example $X = X_1 \times X_2$, $Y = Y_1 \times Y_2 \times Y_3$ we picture the span in a more detailed way as:



In this case we may call each of X_1, X_2, Y_1, Y_2, Y_3, boundaries of R.

2.2 Operations of Span(Graph).

Composition of spans.
The composite of spans $R : X \to Y$ and $S : Y \to Z$ is the span $R; S : X \to Z$ whose head is the graph with vertex set

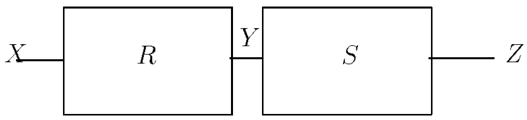
$$\{(r, s); r \text{ is a vertex of } R, s \text{ is a vertex of } S, \partial_1(r) = \partial_0(s)\}$$

and with edge set

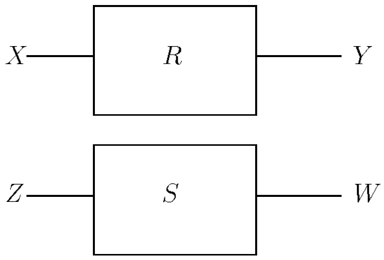
$$\{(\rho, \sigma); \rho \text{ is a edge of } R, \sigma \text{ is a edge of } S, \partial_1(\rho) = \partial_0(\sigma)\}.$$

Beginnings and ends of edges have the obvious definitions. (Technically, $R; S$ is the pullback $R \times_Y S$).

The pictorial representation of the composition of two spans R and S is as follows (with the obvious modification if the objects are products of graphs!):



Tensor of spans.
The tensor of two spans $R : X \to Y$ and $S : Z \to W$ is the span denoted $R \otimes S : X \times Z \to Y \times W$, and defined by: the head of $R \otimes S$ is $R \times S$; the legs of $R \otimes S$ are $\partial_0 \times \partial_0$ and $\partial_1 \times \partial_1$. The pictorial representation of the tensor of two spans is:



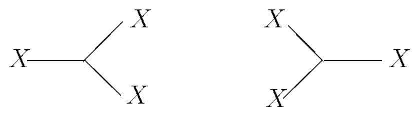
In addition to these operations there are the following constants of the algebra.

The identity of X.
The identity span $1_X : X \to X$ has head X and two legs $1_X, 1_X$. It is denoted by a plain wire.

The diagonal and the reverse diagonal of X.
The span with head X and legs $1_X : X \to X, \Delta_X : X \to X \times X$ is called the diagonal of X, and is denoted also $\Delta : X \to X \times X$.

The span with head X and legs $\Delta_X : X \to X \times X, 1_X : X \to X$ is called the reverse diagonal, and is denoted $\Delta^* : X \times X \to X$.

The two spans are pictured thus:



Diagonal   Reverse diagonal

Projections and reverse projections.

The span with head $X \times Y$ and legs $1_{X \times Y} : X \times Y \to X \times Y$, $p_X : X \times Y \to X$ is denoted p_X and is called a projection, and is pictured by the termination of the wire Y. There is also a similarly defined reverse projection denoted p_X^*.

The spans $\eta_X : I \to X \times X$ and $\varepsilon_X : X \times X \to I$.

The terminal graph, denoted I, has one vertex and one edge, by necessity a loop. The span with head X and legs $! : X \to I, \Delta : X \to X \times X$ is called η_X.

The span with head X and legs $\Delta : X \to X \times X, ! : X \to I$ is called ε_X.

The two spans are pictured thus:

$\eta_{\mathbf{X}}$ $\varepsilon_{\mathbf{X}}$

Technically, η and ε are the unit and counit of the self-dual compact-closed structure on **Span(Graph)**. It will become clear that their role in the context of this paper is to permit a feedback operation on distributed systems.

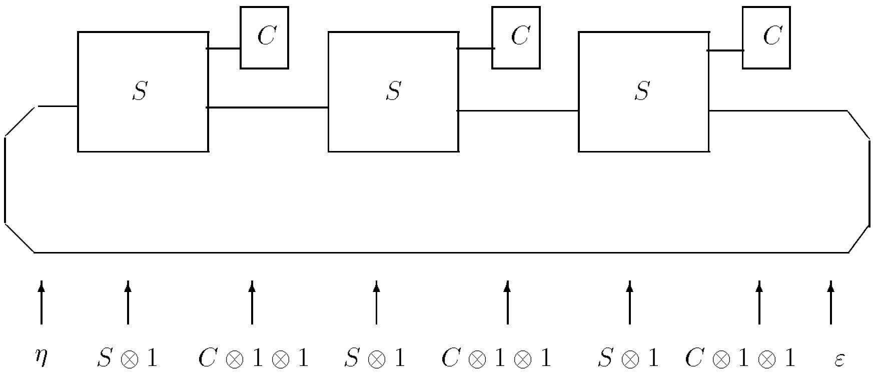
The correspondence between constants and operations, and the geometric representations given above, result in the fact that expressions in the algebra have corresponding circuit or system diagrams. We illustrate this by an example.

2.3 Example.

Given spans $S : X \to X \times X, C : X \to I$, the expression

$$\eta_X; (S \otimes 1_X); (C \otimes 1_X \otimes 1_X); (S \otimes 1_X); (C \otimes 1_X \otimes 1_X); (S \otimes 1_X); (C \otimes 1_X \otimes 1_X); \varepsilon_X$$

has system diagram:



We have indicated the correspondence between parts of the expression and of the diagram using arrows. This diagram might be (with specific interpretation of the components S (server) and C (client)) a specification of a simple token ring.

Remark.

The reader may have noticed that apart from the fact that 'wires' are distinguished as appearing on the left or right of components we have not indicated an orientation on the 'wires', by placing for example an arrowhead. The reason is that in this algebra no such orientation is possible, and this will be reflected later in discussing concurrent systems by the fact that at this level of abstraction wires represent input/output, and not either input or output channels.

2.4 Behaviours of a span.

Definition.

A behaviour π of a span $R : X \to Y$ is a finite path in the graph R, the head of the span.

Notice that applying the legs of the span to a behaviour π yields two paths, one $\partial_0(\pi)$ in X and the other $\partial_1(\pi)$ in Y. The graphs X and Y may be thought of as the (left and right) boundaries of the system R. Then $\partial_0(\pi)$, $\partial_0(\pi)$ may be thought of as the behaviour of the boundaries of the system corresponding to the behaviour π, or the behaviour of the system reflected on the boundaries.

The following result is straightforward.

Proposition.

(i) A behaviour of the composite $R;S$ of two spans is a pair of behaviours, one ρ of R, the other σ of S, such that $\partial_1(\rho) = \partial_0(\sigma)$. That is, a behaviour of $R;S$ consists of a behaviour of R and a behaviour of S which agree (synchronize) on the common boundary.

(ii) A behaviour of the tensor $R \otimes S$ of two spans is just a pair of behaviours, one ρ of R, the other σ of S.

(iii) A behaviour of $\eta_X : I \to X \times X$ is a path in X reflected (synchronously and equally) on the two boundaries. The behaviours of ε are similarly described.

(iv) A behaviour of $\Delta_X : X \to X \times X$ is a path in X reflected (synchronously and equally) on the three boundaries. The behaviours of Δ_X^* are similarly described.

3 Finite labelled transition systems.

The aim of this section is to relate traditional transition systems/automata [A94] to (arrows in) **Span(Graph)** .

3.1 Labelled transitions systems and Span(Graph).

Given an alphabet $\mathcal{A} = \{a_1, a_2, ...\}$ we may construct a graph, which we denote A, as follows: the graph A has one vertex, usually denoted '$*$', and a loop on this vertex for each element of the alphabet $\mathcal{A}$.

Proposition.

A span of graphs $R : A \to I$ is the same thing as a labelled transition system on the alphabet $\mathcal{A}$, in the sense of Arnold-Nivat.

proof.

A span consists of a graph R and two graph morphisms $\partial_0 : R \to A, \partial_1 : R \to I$. Since I is the terminal graph there is no information in ∂_1. The information in ∂_0 is precisely the assignment to each edge of R of a loop in A – since every vertex of R must be assigned to the single vertex $*$ of A. But this is exactly a labelling of the graph R by letters of the alphabet A.

Remark.

If $\mathcal{A}$ and $\mathcal{B}$ are alphabets and A and B are the corresponding one vertex graphs then a general span of graphs $R : A \to B$ is a transition system labelled by the product alphabet $\mathcal{A} \times \mathcal{B}$. More generally, if $\mathcal{A}_1$, $\mathcal{A}_2$, . . ., $\mathcal{A}_m$, $\mathcal{B}_1$, $\mathcal{B}_2$, . . ., $\mathcal{B}_n$, are alphabets then a span of graphs $R : A_1 \times A_2 \times ... \times A_m \to B_1 \times B_2 \times ... \times B_n$ is a transition system labelled by the product alphabet $\mathcal{A}_1 \times \mathcal{A}_2 \times ... \times \mathcal{A}_m \times \mathcal{B}_1 \times \mathcal{B}_2 \times ... \times \mathcal{B}_n$. The separation of the alphabet into a left- and right-hand part seems to be a very minor change from the formalism of Arnold and Nivat but it is exactly this separation which makes the algebra we describe in this paper possible.

Proposition.

Let $\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_m$ be alphabets and $R_1 : \mathcal{A}_1 \to I, R_2 : \mathcal{A}_2 \to I, ..., R_m : \mathcal{A}_m \to I$ be spans of graphs. Then $R_1 \otimes R_2 \otimes ... \otimes R_m$ is the free product in the sense of Nivat-Arnold of the corresponding labelled transition systems.

proof. Clear.

Remark.

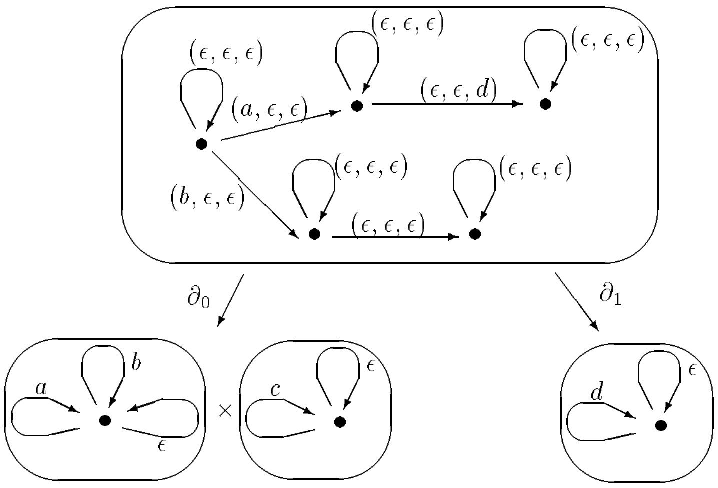
The key to the calculus of Arnold and Nivat are subsystems of the free product called the synchronous products in which one is allowed to restrict the transitions that occur by specifying that they must be labelled by certain given synchronization vectors in the product alphabet. We will show how in **Span(Graph)** this may be achieved by the composition of spans.

3.2 A notation for spans of graphs.

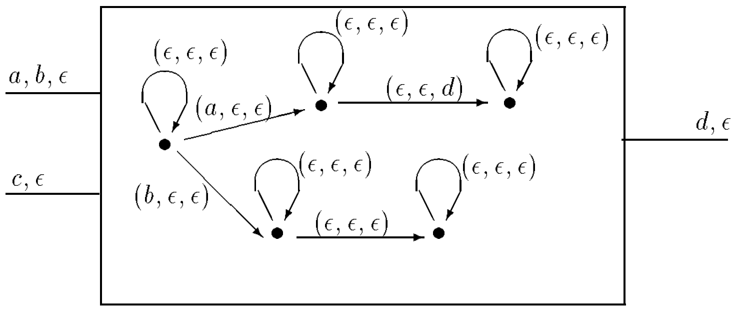
We introduce a simplified notation for spans of graphs which it is easiest to describe by giving an example.

Example.

Consider the span of graphs:

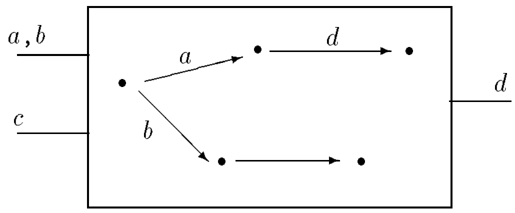


We will represent this graph by the following picture:



3.3 Systems with null actions.

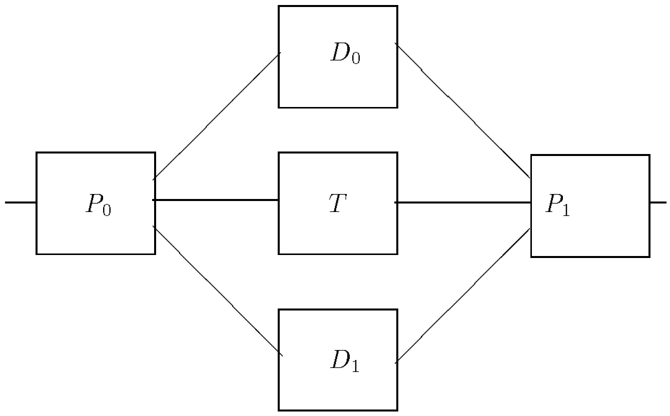
Many systems, of which this last is an example, have the special form that all the graphs involved have a designated loop at each vertex called the null loop, denoted always ϵ. Further the morphisms involved preserve the null loops. Such graphs are called reflexive graphs. Properly speaking when dealing with such systems we are working in another category **Span(ReflexiveGraphs)**. For such systems we often omit, when unambiguous, mention of the null action. For example, the system in 3.2 may be concisely denoted:



3.4 The Peterson mutual exclusion algorithm.

We give a description of Peterson's mutual exclusion algorithm for two processes. We choose to do this since it is the first example in [A94] and the reader should thus be in a position to compare our mode of description with that of Arnold-Nivat. All the spans below are reflexive.

Peterson's algorithm concerns two processes P_0, P_1 each of which has a critical section, and only one of the processes may be in this critical section at a time. In order to accomplish this the two processes communicate through three (parallel) shared binary variable, called D_0, T and D_1. The geometry of the system is therefore of the following form:

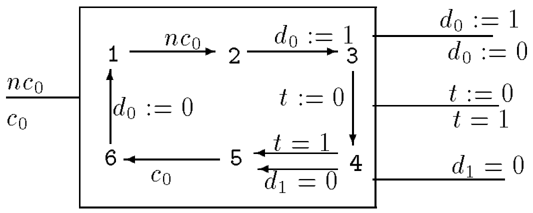


Hence we would expect to be able to represent the system by an expression in **Span(Graph)** of the form

$$P_0; (D_0 \otimes T \otimes D_1); P_1.$$

Each of the processes cycles goes through a cycle as follows: non-critical section; some communication actions with D_0, T and D_1; the critical section; further communication actions; and then the non-critical section again, and so on. The communication actions involve the processes writing or 'reading' the binary variables d_0, t, or d_1. We apostrophize the word 'reading' here because, although that is what the action is often called, we feel that it is a misnomer at this level of abstraction.

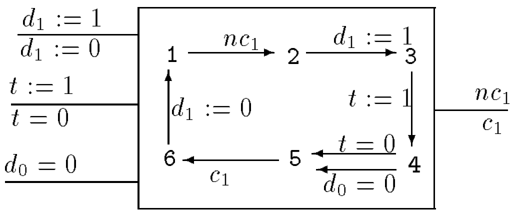
We now describe precisely the process P_0. It is the span of graphs pictured as follows.



$\mathbf{P_0}$

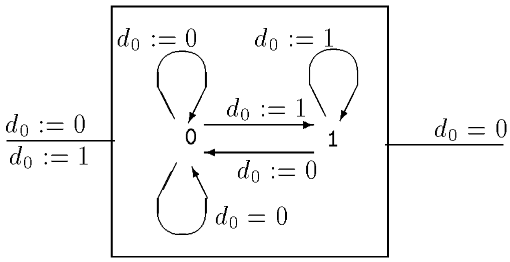
The actions nc_0 and c_0 are the non-critical and critical actions, respectively, of P_0. The actions $d_0 := 0$, $d_0 := 1$, $t := 0$ are write actions to the variables indicated. The actions $t = 1$, $d_1 = 0$ are actions enabled when the variables have the indicted values – these are the 'read' actions, but notice that there are no read actions $t = 0$ or $d_1 = 1$ in P_0. These actions (and similar ones for P_1) in synchronization vectors on page 30 of Arnold need to be omitted.

The process P_1 has a similar description.

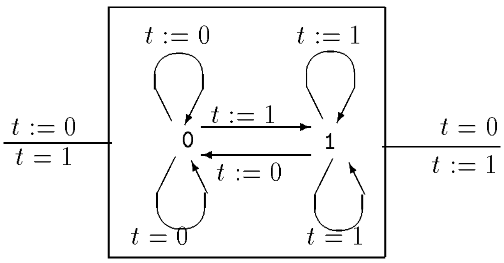


$\mathbf{P_1}$

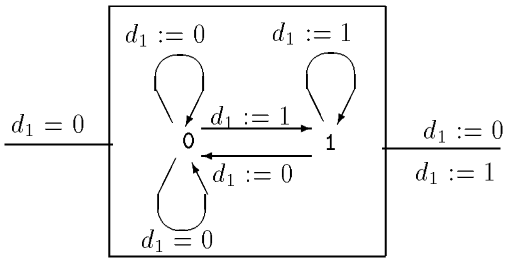
The three binary variables D_0, T and D_1 are defined as follows:



$\mathbf{D_0}$



$\mathbf{T}$



$\mathbf{D_1}$

An action of the composed algorithm $P_0; (D_0 \otimes T \otimes D_1); P_1$ is an action of each of the components P_0, D_0, T, D_1, P_1, that is a 5-tuple of edges, which agree on connecting wires. Notice that because of the special form of the span involved each action of a component is reflected on exactly one input/output wire.

Remark.

A remark on the comparison between the algebra **Span(Graph)** and the formalism of Arnold-Nivat. Our geometric picture of the system clarifies the algorithm. The components are described separately, and the operations allow a description of the system without needing to write down synchronization constraints for the system as a whole.

4 The language generated by a span.

One way of abstracting further from a span of graphs with initial vertex, is to forget the internal state and remember only the behaviours reflected on the boundaries. In this section we consider spans $R : X \to Y$ which have an assigned initial vertex $r_0 \in R$.

Definition.

The language generated by span $R : X \to Y$ with initial vertex r_0 is the set of pairs of paths of the same length (or paths of pairs), one in X and one in Y, of the form $(\partial_0(\pi), \partial_1(\pi))$ where π is a path in R beginning at the initial vertex. We denote the language of R by $Lang(R)$. We denote the set of all paths in $X \times Y$ commencing in the initial vertex as $(X \times Y)^*$. The language of a span from X to Y is thus a subset of $(X \times Y)^*$.

Example.

Consider the span of graphs in section 3.2 with the left-most vertex being taken as the initial vertex. The language generated is given by the prefix closure of the classical regular expression

$$(\epsilon,\epsilon,\epsilon)^*(a,\epsilon,\epsilon)(\epsilon,\epsilon,\epsilon)^*(\epsilon,\epsilon,d)(\epsilon,\epsilon,\epsilon)^* + (\epsilon,\epsilon,\epsilon)^*(b,\epsilon,\epsilon)(\epsilon,\epsilon,\epsilon)^*.$$

Remarks.

The reason the abstraction of a span to a language makes sense is that it is possible, as we show below, to deduce the laguage generated by an expression of spans knowing the languages of the components of the expression. It is clear that if the domain and codomain of R are products of one vertex graphs the languages that arise are prefix-closed regular languages in tuples of actions. What is interesting however is the operations arising from the algebra on these regular languages. These operations are closely related to but not the same as existing operations in the literature - we make a comparison below with operations on the traces of Hoare.

Theorem.

Consider spans $R : X \to Y, S : Y \to Z, T : Z \to W$.

(i) $Lang(R;S) = \{(x,y)$; there exists $l \in Lang(R), m \in Lang(S)$, so that $proj_1(l) = x, proj_2(l) = proj_1(m)$ and $proj_2(m) = y\}$;

(ii) $Lang(R \otimes T) = \{(l,m); length(l) = length(m)$ and $l \in Lang(R), m \in Lang(S)\}$.

proof.

(i) The path (x, y) lies in $Lang(R; S)$ iff there is a path t in $R; S$ such that $(\partial_0(t), \partial_1(t)) = (x, y)$. That is, by the definition of $R; S$, iff there are paths r in R, s in S, such that $\partial_0(r) = x, \partial_1(r) = \partial_0(s)$, and $\partial_1(s) = y$. That is, if and only if there is a path $n (= \partial_1(r) = \partial_0(s))$ in Y such that (x, n) is in $Lang(R)$, and (n, y) is in $Lang(S)$.

(ii) Straightforward.

4.1 Linearizable systems.

A large part of concurrency is based on the presumption that it is reasonable to consider *interleaved* actions in analysing systems. For a large subclass of reflexive spans, what we call *linearizable* spans, this is indeed the case, at least for checking certain properties like deadlock. For reasons of space the results in this section are stated without proof – the proofs are rather straightforward and will be given in an expanded version of the paper.

Definition.

Consider a reflexive span $R : X = X_1 \times X_2 \times ..X_m \to X_{m+1} \times ...X_{m+n} = Y$. Denote a label of the form $(\epsilon, \epsilon, \ldots, \epsilon)$ (only ϵ's) as $\tilde{\epsilon}$, and a label of the form $(\epsilon, \epsilon, \ldots, \epsilon, x_i, \epsilon, \ldots, \epsilon)$ (exactly one non-ϵ label) as $\tilde{x}_i$. A path in R is said to be linear if each edge is labelled either $\tilde{\epsilon}$ or $\tilde{x}_i$ for some i, $x_i \in X_i$.

The span R is said to be linearizable if it satisfies the following property: given any edge $\rho : r \to s$ in R labelled $(x_1, x_2, \ldots, x_{m+1}, \ldots, x_{m+n})$, and any order on the numbers $1, 2, 3, \ldots, m+n$, there is a linear path also from r to s in R whose edges, apart from null-labelled ones, are precisely $\tilde{x}_1$, $\tilde{x}_2$, $\ldots$, $\tilde{x}_{m+n}$ in the given order.

Theorem.

If R is linearizable then so are $R; S$ and $R \otimes S$.

Note.

A reflexive span $R : X_1 \times \ldots \times X_m \to Y_1 \times \ldots \times Y_n$ with the property that each action is labelled non-null on exactly one wire is linearizable. Hence the Peterson algorithm is linearizable since its components are, and it is formed from them using composition and tensor.

Note.

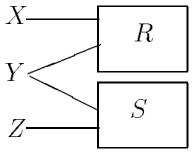
The identity, diagonal, η and ε are not linearizable, and hence linearizable systems do not form a subalgebra of **Span(Graph)**. Nevertheless various derived operations, for example the parallel operation described below, do yield linearizable spans when applied to linearizable spans. It is the fact that these derived operations, but not the constants Δ, η, ε, are linearizable which leads to the operations in such process algebras as CSP of Hoare.

Definition.

If $R : X \times Y \to I$, $S : Y \times Z \to I$ are spans of graphs then $R||_Y S : X \times Z \to I$ is the span defined by

$$R||_Y S = (1_X \otimes \Delta_Y \otimes 1_Z); (R \otimes S).$$

The geometry of this operation is:



$\mathbf{R}||_{\mathbf{Y}}\mathbf{S}$

Theorem.
If $R : X \times Y \to I$ and $S : Y \times Z \to I$ are linearizable then so is $R||_Y S$.

Definition.
Consider a linearizable span $R : X = X_1 \times X_2 \times \ldots \times X_m \to X_{m+1} \times \ldots \times X_{m+n} = Y$ with an intial vertex r_0. Let $N(X_i)$ denote the non-null edges in X_i. For a linear behaviour of R we may consider that the labelling lies in

$$[N(X_1) + N(X_2) + \ldots + N(X_{(m+n)})]^*$$

(where + denotes disjoint union) rather than $(X_1 \times \ldots \times X_{(m+n)})^*$, since we may look just at the non-null actions which occur in a behaviour. We define $Trace(R)$ to be the subset of $[N(X_1) + N(X_2) + \ldots + N(X_{(m+n)})]^*$ arising as labellings of linear behaviours of R commencing at r_0. We may also restrict a trace t to a summand U of $[N(X_1) + N(X_2) + \ldots + N(X_{(m+n)})]$ by omitting letters which are not in the summand; we denote the result by $t|_U$.

The connection with Hoare's parallel operation is evident from the following theorem.

Theorem.
Consider linearizable spans $R : X_1 \times \ldots X_m \times Y_1 \times \ldots Y_n \to I$ and $S : Y_1 \times \ldots Y_n \times Z_1 \times \ldots \times Z_p \to I$. Let

$$U = N(X_1) + N(X_2) + \ldots + N(X_m) + N(Y_1) + N(Y_2) + \ldots + N(Y_n),$$

$$V = N(Y_1) + N(Y_2) + \ldots + N(Y_n) + N(Z_1) + N(Z_2) + \ldots + N(Z_p),$$

$$W = N(X_1) + \ldots + N(X_m) + N(Y_1) + \ldots + N(Y_n) + N(Z_1) + \ldots + N(Z_p).$$

Then

$$Trace(R||_{Y_1 \times \ldots \times Y_n} S) = \{t \in W^*; t|_U \in Trace(R) \text{ and } t|_V \in Trace(S)\}.$$

Remark. Ebergen [E87] and others have used parts of CSP to specify asynchronous circuits. The geometry we have given for expressions in our algebra makes precise the relation between process algebras and circuit diagrams as used in Ebergen. There are advantages to using our algebra in the specification of asynchronous circuits - a single wire, the diagonal, η and ε are vary natural primitives in building circuits; using only linearizable components prevents the use of these primitives. In [E87] there is no such *component* as an instantaneous wire, or instantaneous fan-out.

5 Discrete cartesian bicategories

We give a concise definition of discrete cartesian bicategory (relying for basic categorical concepts on [M70], [B67]). We describe without proof some consequences of the definition and sketch the geometric meaning of these consequences. The proofs are relatively straightforward and will be given in an expanded version of the paper.

Let **B** be a bicategory with objects denoted $X, Y, \ldots$, arrows $R, S, \ldots$, and 2-cells $\alpha, \beta, \ldots$, composition of arrows $R; S$. Let $Map(\mathbf{B})$ be the subbicategory of arrows with right-adjoints; we denote arrows of $Map(\mathbf{B})$ by $f, g, \ldots$, and their right adjoints by $f^*, g^*, \ldots$. For simplicity we will assume that $Map(\mathbf{B})$ is a category.

Definition

B is a discrete cartesian bicategory if

(i) $Map(\mathbf{B})$ has finite products, the operation of product being denoted $\times$, the terminal object I, the diagonals denoted Δ, and the projections p;

(ii) For each pair of objects X, Y the category $\mathbf{B}(X, Y)$ has finite products, the operation of product being denoted $\wedge$.

(iii) If $R : X \to Y, S : Z \to W$, the formula

$$R \otimes S = (p_X; R; p_Y^*) \wedge (p_Z; S; p_W^*) : X \times Z \to Y \times W$$

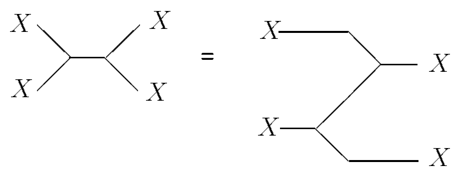
defines a tensor product on **B**;

(iv) (discreteness axiom)

$$\Delta^*; \Delta = (1_X \otimes \Delta); (\Delta^* \otimes 1_X) : X \times X \to X \times X.$$

It is straightforward to show that **Span(Graph)** is a discrete cartesisan bicategory. The 2-cells, which we have not mentioned to this point, are defined as follows: a 2-cell α from span $R : X \to Y$ to span $S : X \to Y$ is a graph morphism $\alpha : R \to S$ such that $\partial_0 \alpha = \partial_0$ and $\partial_1 \alpha = \partial_1$. Further an arrow of $Map(\mathbf{Span(Graph)})$ is a span $R : X \to Y$ in which $R = X$ and $\partial_0 = 1_X$ and hence $Map(\mathbf{Span(Graph)})$ is essentially **Graph**.

Notice the geometry of the discreteness axiom; using the rules above for drawing pictures of expressions we see that the axiom is pictured as:



Definition of η and ε.

Though we gave η and ε independently in section 2 (for their use in describing feedback) they are in fact derived operations of the algebra, as the pictures suggest, namely

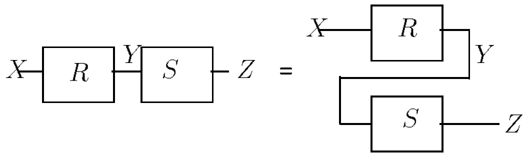
$$\eta_X = p_I^*; \Delta_X \text{ and } \varepsilon_X = \Delta_X^*; p_I.$$

Equation relating $\otimes$ and composition and feedback.

The following important equation holds:

$$R; S = (1 \otimes \eta); (R \otimes 1 \otimes S); (\varepsilon \otimes 1),$$

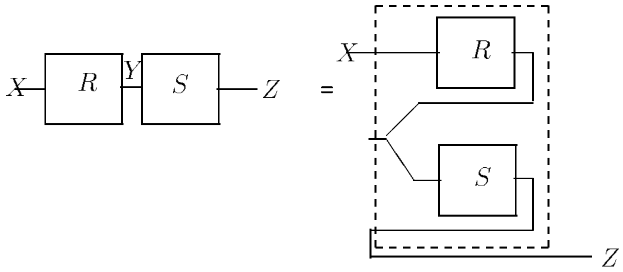
pictured as



Equation relating $\otimes$ and $||$ and composition.

We have described $||$ in terms of $\otimes$, Δ, and composition, but in fact general composition can be constructed using $||$ as follows:

$$R;S = (1 \otimes p_I^* \otimes \eta);[((R \otimes 1);\varepsilon)||((S \otimes 1);\varepsilon)] \otimes 1).)$$



(The dashed rectangle is $((R \otimes 1);\varepsilon)||((S \otimes 1);\varepsilon)$.)

6 Conclusions.

We have shown that a natural algebraic structure on **Span(Graph)** allows the compositional specification of concurrent systems. Hoare's parallel operation appears as a derived operation in this algebra. The simpler basic operations of our algebra are possible because we do not insist on interleaving semantics: interleaving prevents consideration of the identity span, as well as other natural constants such as the diagonal. We have given some examples of transforming systems using the equations of the algebra. Associated to the algebra there is a geometry which expresses the distributed nature of a concurrent system. This relation between algebra and geometry makes precise the relation between process algebras and circuit diagrams as used, for example, in Ebergen [E87].

7 Bibliography.

[A94] A. Arnold, Finite transition systems, Prentice Hall, 1994.
[B67] J. Bènabou, Introduction to bicategories, Reports of the Midwest Category Seminar, Lecture Notes in Mathematics 47, pages 1–77, Springer-Verlag, 1967.
[BSW96] Bloom S., Sabadini N, Walters RFC, Matrices, machines and behaviors, Applied Categorical Structures, 4: 343-360, 1996.
[BJ94] C. Brown C. and A. Jeffrey, Allegories of circuits, in: Proc. Logical Foundations of Computer Science, St Petersburg, 1994.
[CW87] A. Carboni and R.F.C. Walters, Cartesian Bicategories I, Journal of Pure and Applied Algebra, 49, pages 11-32, 1987.

[E87] J.C. Ebergen, Translating Programs into Delay-insensitive Circuits, PhD thesis, Eindhoven University of Technology, 1987.
[GKW96] Robbie Gates, P. Katis and R.F.C. Walters, A program for computing with the cartesian bicategory **Span(Graph)**, School of Mathematics and Statistics, University of Sydney, 1996.
[H85] C.A.R. Hoare, Communicating sequential processes, Prentice Hall, Englewood Cliffs, NJ,1985.
[JS90] G. Jones and M. Sheeran, Circuit design in Ruby, in: Formal methods for VLSI design, North-Holland, 1990
[JS91] A. Joyal and R. Street, An introduction to Tanaka duality and quantum groups, Category Theory 1990, Como, Lecture Notes in Mathematics 1488, Springer Verlag, 1991.
[JS93] A. Joyal and R. Street, Braided tensor categories, Advances in Mathematics, 102: 20-78, 1993.
[K95] C. Kassel, Quantum Groups, Graduate Texts in Mathematics, Springer-Verlag, New York, 1995.
[KSW94] Katis P, Sabadini N, Walters RFC, The bicategory of circuits, Computing: Australian Theory Seminar, UTS, Sydney, 1994.
[KSW97] Katis P, Sabadini N, Walters RFC, Bicategories of processes, Journal of Pure and Applied Algebra, 115, no.2, pp 141 - 178, 1997
[K96] Katis P., Categories and bicategories of processes, PhD Thesis, University of Sydney, 1996.
[KW93] W. Khalil and R.F.C. Walters, An imperative language based on distributive categories II, Informatique Thèorique et Applications, 27, 503-522, 1993.
[M70] S. Mac Lane, Categories for the working mathematician, Springer Verlag, 1970.
[M95] Majid, Foundations of quantum field theory, Cambridge 1995.
[P81] D. Park, Concurrency and automata on infinite sequences, in 5th GI Conference on theoretical computer science, 167-183, LNCS 104, Springer, 1981.
[P71] R. Penrose, Applications of negative dimensional torsors, in Combinatorial Mathematics and its applications, (D. J. A. Welsh, Ed.) pp. 221-244, Academic Press, New York, 1971.
[RSW] Rosebrugh R., Sabadini N, Walters RFC, Minimal realization in bicategories of automata, to appear, Journal of Pure and Applied Algebra.
[SVW96] Sabadini N, Vigna S, Walters RFC, A note on recursive functions, Mathematical Structures in Computer Science, 6, 127-139, 1996.
[SW93] N. Sabadini and R.F.C. Walters, On functions and processors: an automata-theoretic approach to concurrency through distributive categories, School of Mathematics and Statistics Research Reports, University of Sydney, (93-7), 1993.
[SWW93] Sabadini N., Walters R.F.C., Weld Henry, Distributive automata and asynchronous circuits, Category Theory and Computer Science 5, Amsterdam, 28-32, 1993.
[SWW94] Sabadini N., Walters R.F.C., Weld Henry, Categories of asynchronous circuits, Computing: Australian Theory Seminar, UTS, Sydney, 1994.
[S95] R.H. Street, Higher categories, strings, cubes and simplex equations, Applied Categorical Structures, 3, 29- 77, 1995.
[T95] Turaev, Quantum invariants of knots and 3-manifolds, Featured review in Mathematical Reviews MR:95k 57014.
[W89] R.F.C. Walters, Data types in a distributive category, Bull. Austr. Math. Soc., 40:79–82, 1989.
[W92a] R.F.C. Walters, Categories and Computer Science, Carslaw Publications 1991, Cambridge University Press 1992.
[W92b] R.F.C. Walters, An imperative language based on distributive categories, Mathematical Structures in Computer Science, 2:249–256, 1992.
[Z85] W. Zielonka, Notes on Finite Asynchronous Automata, RAIRO, 27, 99-135, 1985.