UNIVERSITY OF SUSSEX

COMPUTER SCIENCE

Two Papers Concerning Categories in Concurrency Theory

David Murphy

Report 5/94

April 1994

Computer Science
School of Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QH

ISSN 1350–3170

Introduction

This report contains two papers which use categorical methods to present accounts of concurrency–theoretic material. In both cases the aim has been to uncover essential underlying structure by giving a suitably universal characterisation of it.

The first paper, *A Functorial Semantic for Ob erved Concurrency*, is joint work with Axel Poigné (GMD St. Augustin), and was done during this author's Royal Society Fellowship at the GMD. It presents a unifying account of concurrency theories based on the idea of observing systems over time. This material first appeared in [1], and the paper appearing here is a slightly modified version of *op. cit.*

The second paper presented here, *Proce Synchroni ation a Glueing*, is joint work with Stefano Kasangian (Venice) and Anna Labella (Rome), gives a model of processes with multiway synchronisation *à la* CSP. This account is parametric, being based on glued forests of trees, and thus can be instantiated as a variety of interleaving or 'true concurrency' models depending on how the forest is read as a transition system (or, more generally, behavioural structure). A satisfactory account of process synchronisation is given as a glueing construction over forests, and other process combinators are universally characterised. The paper presented here concentrates on showing how the glueing construction works at a relatively concrete level: a more comprehensive discussion, treating additionally notions of behaviour and equivalence, and giving further details on the modelling of the process combinators, will appear in a later paper.

On a more personal note, to conclude, it is my hope that the use of categories will continue to grow in concurrency theory. One feels a certain hostility to their use in some quarters, akin one supposes to the hostility most new and challenging mathematical tools face: only time will tell if this is a luddite reaction as one often supposes or a reasoned criticism.

David Murphy,
Brighton,
Easter 1994.

[1] D. Murphy and A. Poigné, *A Functorial Semantic for Ob erved Concurrency*, in the Proceedings of Mathematical Foundations of Computer Science (MFCS) 1992, Springer-Verlag LNCS 629.

A Functorial Semantics for O served Concurrency

David Murphy and Axel Poigné

28th April, 1994

Abstract

This paper presents a meta-model of observation in concurrency theory; it allows us to unify notions of observation in many different behavioural settings. We treat traces, process trees and event structures, and show how observations of them fit into a common framework. Behaviour and observation will both be modeled as categories and linked using the notions of 'functor' and 'adjunction'.

Timing will be our chief example of observation; we present a timed traces model, and show how it generalises to timed process trees (branching time) and timed 'true concurrency.' Our general framework sees timing as a way of embedding observations into time. Stable categories of embeddings are then natural metamodels of timed observation.

I always console myself with the tho ght that whatever can be known of me is by definition not me, is hetronomo s to my a -thentic being, since the s bject cannot be capt red in an objective representation.

Terry Eagleton

§1. Introduction

Concurrency theory is now a large area; there are many concurrency theories, and many notions of 'concurrent system.' This paper is a contribution towards unifying this chaos; we provide a meta-model of concurrency. In particular, the notion of 'behaviour',—what a system does,—is separated from 'observation',—what can be seen of it. These two notions are formulated as categories and linked using adjunctions, giving a model that specialises to several well–known concurrency theories.

The rest of the paper is structured thus; in the rest of this section we provide some background. Then, as a motivating example, we present the essentials of a timed traces model. The elementary mechanics of the meta-model are then presented. We next show how to tweak the parameters of our meta-model to get timed process trees and timed true concurrency versions. Finally we refine the meta-model and show that the category of all models enjoys on. Worst case processes correspond to observing 'all the time' and hence define the structure of time appropriate to that kind of observation.

Our main result is to define various observers and to show that they are canonical with respect to certain types of behaviour; traces, branching time and true concurrency. This makes the relationship between behaviour and observation somewhat clearer. We then give a general framework of which all of our examples are instances and which provides pleasing general structure.

§2. A Timed Traces Model

A basic notion of behaviour given in Hoare's [12] is that of a trace: the behaviour of (an execution of) a concurrent system is represented by a sequence of actions:

Definition 1. Given a set of possible actions a process P might engage in, A, the set of *trace* of P, $\mathrm{tr}(P)$, is a subset of the set of all possible sequences of actions $\mathrm{tr}(P) \subseteq A^*$.

Each occurrence of an action $\mathtt{a}$ in a given trace s can be identified uniquely by its position in the trace, so we can assume as given a set of *unique* or *L–labeled* occurrences of actions, $E = L \times A$. This set of *event* , with typical members $\mathtt{e}, \mathtt{f}$, will be more convenient to work with than A.

§2.1. Timing

In this section we will extend the traces model to timed traces; our treatment is a little idiosyncratic, because we want to emphasise some points that will be important later. A more standard presentation is [9], where a good introduction can be found.

We will suppose as usual that points of time are reals and that things start at $t \geq 0$. Then, a timed trace of P is a trace $s \in \mathrm{tr}(P)$ together with a function $\tau : E \longrightarrow \mathbb{R}^+$ that assigns a nonegative real to an (assumed atomic) event.

Definition 2. A *trace timing* is a function τ which is

Consistent. Write $\leqslant_s$ for the sequence order of the trace s, [illegible] a function $\alpha : \mathbb{R}^+ \to E$ that tells us the last thing that happened at $r \in \mathbb{R}^+$ satisfying

$$\begin{aligned} \alpha(r) = \mathtt{e} \quad \Longrightarrow \quad & \tau(\mathtt{e}) \leq r \quad \wedge & (1) \\ & \tau(\mathtt{e}) \leq \tau(\mathtt{f}) \leq r \quad \Longrightarrow \quad \mathtt{e} = \mathtt{f} & (2) \end{aligned}$$

The connection between α and τ is thus

$$\alpha(\tau(\mathtt{e})) = \mathtt{e} \qquad (3)$$

$$\tau(\alpha(r)) \leq r \qquad (4)$$

Notice that we do not allow events to be simultaneous, since α is a function, not a relation. If we wanted to have events happening simultaneously, we could adopt the trick of dealing with sequences of sets of actions rather than sequences of actions.

§2.2. A more abstract setting

Any model of observation must answer the question ‘What is the connection between the behaviour and the possible observations of a process ?’ To answer this we must say how to formalise ‘behaviour’ and ‘observation’. This formalism will be provisional; in section 4 we to-.

Our observations are just the times things happen, so we will be interested in the usual time domains $\mathbb{R}^+, \mathbb{Q}^+$ and $\mathbb{N}^+$. These can be made into categories in various ways, depending on the precise

Thus a real timing of a trace s is precisely a canonical $\mathbb{R}^+$ timing of the behaviour $\mathfrak{S}$, justifying the previous definition somewhat. It is clear that definitions of integer–timed and rational–timed trace models follow just by writing $\mathbb{N}$ or $\mathbb{Q}$ for $\mathbb{R}$.

§2.4. Embedding

Our requirement that $f_!$ is a full and faithful functor left–adjoint to f^* is chosen because we want to think of $f_!$ as *embedding behaviour into time*. This will be a common theme; a $\mathfrak{T}$–timing of $\mathfrak{B}$ will be a way of telling what happened when; an embedding of $\mathfrak{B}$ into $\mathfrak{T}$. Thus our real interest is the power observers must have, – what structure $\mathfrak{T}$ must have, – in order to be able to define this embedding. The last proposition merely amounts to saying that one observer with a clock is enough to 'see' an execution in the traces model.

§3. An abstract view of other theories

Many other models of concurrency fit into the setting outlined above. Here we consider branching time (represented by process trees); pure concurrency (essentially Shields' cubical automata); and true concurrency (Winskel's event structures).

§3.1. Branching time

An obvious elaboration of behaviour beyond a trace model is *branching time*. Here we represent a process by a tree, with the branches indicating possible executions, and the branching structure recording non-determinism. Observationally, this corresponds to an observer who, each time a choice is presented, makes separate copies of himself to explore each alternative. This naturally gives rise to a branching structure—hence 'branching time'.

Definition 6. A *branching time behaviour* is a tree $(E, \leqslant_m)$ with countable underlying set E. Each *proce tree* $M = (E, \leqslant_m)$ gives rise to category $\mathfrak{M}$ as indicated in section 2.2. (The tree $\mathtt{e}\begin{smallmatrix}\nearrow\ \mathtt{f}\\ \searrow\ \mathtt{g}\end{smallmatrix}$ indicates that the traces $\langle \mathtt{e}, \mathtt{f}\rangle$ and $\langle \mathtt{e}, \mathtt{g}\rangle$ are possible, but not $\langle \mathtt{e}, \mathtt{f}, \mathtt{g}\rangle$: the event $\mathtt{e}$ causes $\mathtt{f}$ or $\mathtt{g}$ but not both.)

In order to be able to time a branching time process, the observer must be able to branch. A suitable category to time branching time processes therefore is the Baire tree:

Definition 7. The Baire

Definition 10. A abstract timing of a branching time behaviour

$f_! : \mathfrak{M} \to \mathcal{B}$ (a canonical timing) followed by $U : \mathcal{B} \to \mathbb{R}^+$ (forgetting the branching structure):

$$\begin{tikzcd} \mathfrak{M} \arrow[r, shift left, "f_!"] & \mathcal{B} \arrow[l, shift left, "f^*"] \arrow[r, "U"] & \mathbb{R}^+ \end{tikzcd}$$

Clearly U cannot in general enjoy any universal properties, as there are no canonical linearisations of trees.

§3.2. Pure concurrency

The results of our model for purely concurrent processes (ones with no nondeterminism) are quite suprising. We begin with a notion of such processes closely related to the cubical automata of Shields [23].

Definition 13. A *purely concurrent proce* is a poset $Q = (E, \leqslant_q, 0)$ with a least element. Each such gives rise to category $\mathfrak{Q}$ in the usual way. (The poset $\begin{tikzcd}[row sep=tiny] & \mathtt{f} \\ \mathtt{e} \arrow[ur] \arrow[dr] & \\ & \mathtt{g} \end{tikzcd}$ indicates that the traces $\langle \mathtt{e}, \mathtt{f}, \mathtt{g} \rangle$ and $\langle \mathtt{e}, \mathtt{g}, \mathtt{f} \rangle$ are possible. The event $\mathtt{e}$ causes or must happen before $\mathtt{f}$ and $\mathtt{g}$; the events that enable a given one are just those in its down-closure.)

To time such a purely concurrent process we need to have an observer everywhere that a distributed transition might happen. We will take the Petri net view and assume that a single observer can see every occurrence of the same action since that action always happens at a given transition of the net, and the observer need merely wait there. Thus the timing space of interest is $(\mathbb{R}^+)^L$ for some set of locations L, and in the Petri net world we can identify L with the set of possible transitions A. It is reasonable to assume that L is countable, so we need only deal with countable powers of $\mathbb{R}^+$.

What order $\leq_a$ should we give to this space ? Two possibilities are obvious;

Local

[illegible]

Consistent. This is causal ordering implies temporal ordering as usual

If the observers know the structure of

§

There are many categories $\mathfrak{T}$ which it is difficult to interpret as a model of time, so we can profitably refine this definition. Behaviours are embedded into time, so we will look for a general category of observation in which observations are objects, an arrow $f : t \rightarrow t'$ indicates that the observation t can be extended to t'. Here we can identify a time with the worst possible observation we could have made up to it; the behaviours observable up to some time t form the subobjects $\mathbf{sub}(t)$ of that time.

Definition 20. A category is an *ob ervation category* iff

(is9)y

tions to those possible now. Moreover, these two functions form an embedding/projection pair.

The fact that $(f^*, f_!)$ is a *rigid* embedding/projection pair means that there are no gaps in the image of $f_!$; this is reasonable if translated into primitive terms: if $\mathsf{e} \leq \alpha(r)$ then there is a $r' \leq r$ such that $\mathsf{e} = \alpha(r')$.

§4.2. Objects of time

In most of the cases of timed observation we have seen, there is a worst possible behaviour Ω which is isomorphic to the structure of time. Any behaviour, then, is a subbehaviour of this one, and the timing and last thing functions τ and α give rise to the adjunction (8) above with $t' = \Omega$, giving the connection with our previous definition of canonical timing. In such cases, Ω is clearly a weakly terminal object. We do not require the existence of such an object in all observation categories, since there are some situations where there is no canonical observer and thus no worst case observation.

§4.3. Properties of observation

isations of stable categories of embeddings, which throw some light into the independence of our conditions.

The infinite objects in stable categories of embeddings arise only as filtered colimits of the finite ones. This means that we can see any observation category as a suitable completion of a category of finite observations:

Definition 25. A category of *event* is a small one where all arrows are monos and where each slice is a finite distributive lattice.

Proposition 26. Any observation category arises as the ind-completion of some category of events.

We have seen that pullbacks and filtered colimits are important properties of observation categories. This motivates

Definition 27. A functor is called *table* if it preserves pullbacks, and *continuou* if it preserves filtered colimits. (These being the obvious generalisations of the associated concepts for maps between ordered structures.)

Proposition 28. The category of stable and continuous functors between two observation categories $\mathfrak{C}$ and $\mathfrak{D}$ with cartesian natural transformations as morphisms, SC[$\mathfrak{C}, \mathfrak{D}$], is an observation category.

This last proposition gives us the key to the structure of the category of all observation categories, and hence to a possible meta-logic of observation;

Proposition 29. The category of observation categories with stable and continuous functors as morphisms is a cartesian-closed category.

§4.4. Instances of the situation

We are now in a position to show that our various models of timed observation give rise to observation categories.

Proposition 30. All our examples of timing behaviours: timed traces; timed process trees; and timed prime event structures: give rise to stable categories of embeddings.

Proof. [Sketch.] We will tackle each case separately:

Traces. We will sketch this in some detail, and rely on more general arguments for the remaining cases. We have to show that the category of traces **Tr** (that is the category whose objects are countable sets E endowed with a total order $\leqslant_s$ and whose morphisms are set maps which are also embeddings) is an observation category. This

§4.5. Gross structure

We have seen that the ‘right’ functors between categories of embeddings are stable and continuous ones. In this subsection we give two uses for

§5. Concluding Remarks

We have shown that various notions of behaviour give rise to canonical observers who can see the 'worst' behaviour in that class. Real timing information can then be recovered by forgetting the structure of such observers. The structure of all possible observations of a given type has also been discussed, and shown for the cases of interest to form a stable category of embeddings. These categories congregate in a cartesian closed category, giving some insight into the general logic of observation.

Further work

In this paper we have seen several examples and a general model. On one hand the examples fit into the model. On the other, the axioms of the model seem intuitively reasonable for observations. However, it is possible that we can further restrict the model without losing the ability to handle the examples. Further work includes:

- Throughout this paper we have concentrated on presheaves rather than sheaves. That is, we have not used the fact that different observations can be defined over different intervals of time and then glued together to form a larger observation. This is partly to keep the technical complexity of the paper manageable, and partly because sheaves sometimes occur automatically; all presheaves of traces, for instance, are sheaves. It would be interesting to consider the sheaf condition in general, though; this might give some insight into behaviours recursively defined over time. Topologically, sheafhood relates to issues of compactness which also deserve investigation.

- A standard way of recovering branching time information is to model a single observation as a domain, then combine information about different runs using powerdomains. This is really a monadic technique, so it would be profitable to investigate whether we can adapt it to our situation, rather than coding the branching structure of observers explicitly.

- Our approach is based on the notion of observing a set of events. For this reason, it is hard to see how an observation category that did not have a forgetful functor to $\mathfrak{Set}$ might arise; we should thus check that restricting ourselves to concrete observation categories does not significantly change our results.

- In this paper we have only dealt with posets rather than pomsets, and have generally ignored the issue of equivalences over behaviours: both of these are fairly serious omissions and should be rectified.

Related work

There are several other approaches to finding a meta-model of timed concurrency that are related to the one presented here:

- It is possible to make more of the arithmetic operations on $\mathbb{R}^+$ than we do. Koymans [16] uses this structure to define a temporal logic based on the reals.
- This approach can be taken further using the concept of enrichment; Pratt's group in [6] gives several categories based on the reals with a monoidal structure. These are then used to enrich behavioural categories. Such techniques are comprehensively displayed in Kasangian and Labella's [15].
- Another starting point is idea that we can associate a set of predicates with a state of a system—the things we know to be true of the behaviour by that point. This means that an observation is a functor $\ : \mathfrak{B}^{\mathrm{op}} \to \mathfrak{Set}$, and all possible observations live in the topos $[\mathfrak{B}^{\mathrm{op}}, \mathfrak{Set}]$. Such a sheaf–theoretic viewpoint is taken by Ehrich et al. in [10], (where the model of timed observation forms an observation category) and used by Phoa and the first author to discuss the logic of observation in these situations [21].
- It is wise to admit that there are concurrency-theoretic models that seem not to be related at all to the one presented here; the 'concurrency as chemistry' metaphor of the chemical abstract machine [3], for instance, is very far from our paradigm.

Bibliography

[1] J. van Benthem, *The logic of time*, D. Reidel, 1983.

[2] J. Bergstra and J-W. Klop, *Proce   theory ba ed on bi imulation e-mantic* , in Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency (J. de Bakker, W. de Roever, and G. Rozenberg, Eds.), Volume 354, Springer-Verlag LNCS, 1989, Proceedings of REX 1988.

[3] G. Berry and G. Boudol, *The chemical ab tract machine*, in Principles of Programming Languages, 1990, Proceedings of the 17th ACM Symposium.

[4] E. Best and C. Fernández, *Non equential proce e , a Petri net view*, EATCS Monographs on theoretical computer science, volume 13, Springer-Verlag, 1988.

[5] S. Brookes and A. Roscoe, *An improved failure model for communicating equential proce e* , in the Proceedings NSF-SERC Seminar on Concurrency, Volume 197, Springer-Verlag LNCS, 1985.

[6] R. Casley, R. Crew, J. Meseguer, and V. Pratt, *Temporal tructure* , in Category Theory and Computer Science, Volume 389, Springer-Verlag LNCS, 1989.

[7] T. Coquand, *Categorie of embedding* , Theoretical Computer Science, Volume 68 (1989), Pp. 221–237.

[8] T. Coquand, C. Gunter, and G. Winskel, *dI-domain a a model of polymorphi m*, in Mathematical Foundations of Programming Language Semantics, Volume 298, Springer-Verlag LNCS, 1988.

[9] J. Davies and S. Schneider, *An introduction to timed CSP*, Technical Report Number 75, Oxford University Computer Laboratory, 1989.

[10] H.-D. Ehrich, J. Goguen, and A. Sernadas, *A categorical theory of object a ob erved proce e* , in the Proceedings of REX/FOOL, 1991.

[11] U. Engberg and G. Winskel, *Petri net a model of linear logic*, [illegible]

[18] F. Mattern, *Virtual*

Process Synchronisation as Glueing

Stefano Kasangian   Anna Labella   David Murphy

28th April, 1994

Abstract

Process algebras based on the notion of concurrent processes cooperating on common actions are commonplace in the literature. Here we give a categorical model of such a notion of interprocess synchronisation,

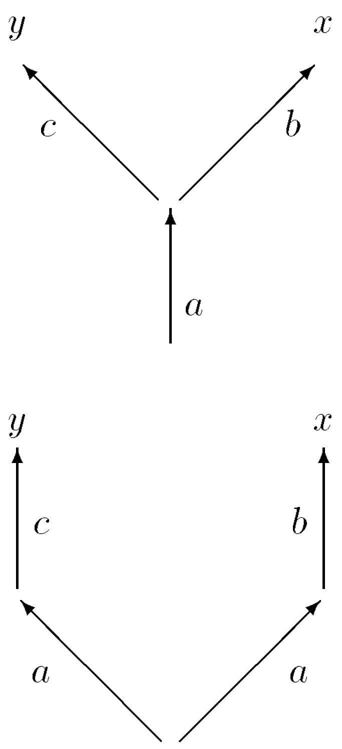


Figure 2: Two trees illustrating the rôle of agreement data.

We begin by giving a structure to capture the computations taking place along runs; a run will be labelled by what is, essentially, a word. Our definition, though, will make the later definition of agreement and glueing easier.

Definition 1. Consider as given an alphabet A of actions, and let A^* denote the set of words on A as usual, with ϵ being the empty word. Define $A_{\mathbb{N}}$ as the set $\mathbb{N} \times (A \cup \{\epsilon\})$, let a range over $A \cup \{\epsilon\}$ and say that a set $S \subseteq A_{\mathbb{N}}$ is

(i) *con i tent* if the following all hold:

  (a) for any $n \in \mathbb{N}$, there exists at most one $(n, a) \in S$;

  (b) if $(n, \epsilon) \in S$ then $n = 0$;

  (c) if $(0, a) \in S$ then $a = \epsilon$.

(ii) *prefix clo ed* if it is consistent and there is a (necessarily unique) $m \in \mathbb{N}$ such that

  (a) $n > m$ implies that $(n, a) \notin S$ for any a, and

  (b) $n \leq m$ implies that there exists an a such that $(n, a) \in S$.

  In this case we call m the length of S, written $|S| = m$.

Example 2. A word $s \in A^*$ gives rise to a prefix closed set $S \subseteq A_{\mathbb{N}}$ and vice versa, via the bijection $a_1 a_2 \dots a_m \leftrightarrow \{(0, \epsilon), (1, a_1), \dots, (m, a_m)\}$.

The sets $\{(1,a),(3,b)\}$ and $\{(1,a)\}$ are consistent but not

We will model trees as (a certain kind of) category enriched over $\mathcal{A}_{\mathbb{N}}$. Rather than give the general definition [12, 22] of a category enriched over a 2–category, we specialised at once to the case in hand.

Definition 7. A category $\mathcal{X}$ enriched over $\mathcal{A}_{\mathbb{N}}$ (or an $\mathcal{A}_{\mathbb{N}}$-*category*) consists of a triple (X, ε, α) where

(i) X is a finite set (of *run*);

(ii) $\varepsilon : X$

Example 9. Consider the $\mathcal{A}\mathbb{N}$–category $(\{x, x'\}, \varepsilon, \alpha)$ where

$$\varepsilon(x) = \varepsilon(x') = \{(0, \epsilon), (1, a), (2, b)\}$$

and $\alpha(x, x') = \{(2, 2, b)\}$. This is clearly not a tree, as the two 'branches' start separately and then join. (An even more pathological example is obtained by setting $\alpha(x, x') = \{(0, 0, \epsilon), (2, 2, b)\}$.)

Example 10. An $\mathcal{A}\mathbb{N}$–category which consists of a forest of two trees rather than a single tree is given by the following data: $(\{x, x'\}, \varepsilon, \alpha)$ where $\varepsilon(x) = \varepsilon(x') = \{(0, \epsilon), (1, a)\}$, and $\alpha(x, x') = \emptyset$.

3. Glueing

In this section we discuss how to glue branches of trees together. Essentially to glue two trees, (X, ε, α) and (Y, ζ, β), we give a function $\gamma : X \times Y \to \mathrm{arr}(\mathcal{A}\mathbb{N})$, such that $\gamma(x, y)$ is a consistent (but not necessarily prefix–closed) set assigning a glueing to x and y. Thus a glued forest will consist of a collection of trees together with some glueing data connecting branches in different trees. We discuss some examples before giving the general construction.

Notation 11. We will use $\mathcal{X}$, $\mathcal{Y}$ etc. for $\mathcal{A}\mathbb{N}$–categories in general and trees in particular; the sense will be clear from context. Typically $\mathcal{X}$ will have components (X, ε, α): $\mathcal{Y}$, (Y, ζ, β), etc. Glueings will be represented by γ.

Example 12. Consider the process $a\,.\,(b\,.\,0{+}c\,.\,0) \mid\mid b\,.\,d\,.\,0$. It is natural to assign the forest in figure 3 to this process, that is the trees $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$ where

$$\begin{aligned} \varepsilon(x) &= \{(0, \epsilon), (1, a), (2, c), \} \\ \varepsilon(x') &= \{(0, \epsilon), (1, a), (2, b)\} \\ \alpha(x, x') &= \{(0, 0, \epsilon), (1, 1, a)\} \\ \zeta(y) &= \{(0, \epsilon), (1, b), (2, d)\} \end{aligned}$$

together with the glueing $\gamma(x', y) = \{(2, 1, b)\}$, $\gamma(x, y) = \emptyset$. We see that this glueing between different trees is a consistent but not prefix closed set.

Example 13. This example demonstrates multiple synchronisation. Consider

$$(a\,.\,b\,.\,0) \;\mid\mid\; (a\,.\,(b\,.\,0 + c\,.\,0)) \;\mid\mid\; (a\,.\,c\,.\,0)$$

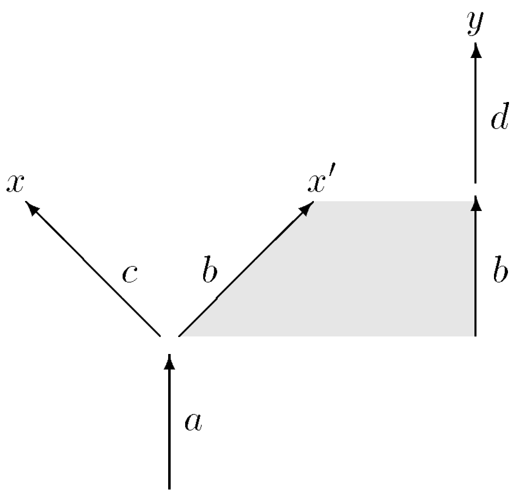
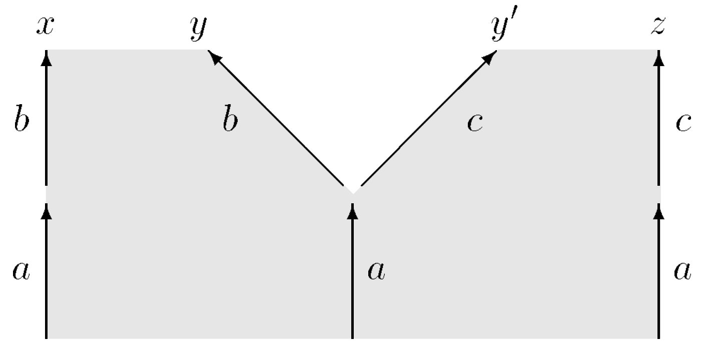


Figure 3: An example glueing.

A reasonable forest to associate with this process is shown below: here we have runs x, y, y', z (in three different trees) with

$$\begin{aligned} \varepsilon(x) = \zeta(y) &= \{(0,\epsilon),(1,a),(2,b)\} \\ \zeta(y') = \eta(z) &= \{(0,\epsilon),(1,a),(2,c)\} \\ \gamma(x,y) &= \{(1,1,a),(2,2,b)\} \\ \gamma(x,y') = \gamma(x,z) &= \{(1,1,a)\} \\ \beta(y,y') &= \{(0,0,\epsilon),(1,1,a)\} \\ \gamma(y,z) &= \{(1,1,a)\} \\ \gamma(y',z) &= \{(1,1,a),(2,2,c)\} \end{aligned}$$



Discussion 14. This example demonstrates several important points:

(i) We indicate that y and y' are runs *on the ame tree*, i.e. are not distributed, by $\beta(y,y') \ni (0,0,\epsilon)$. The need to be able to do this explains why we keep $(0,\epsilon)$ in every extent, and impose conditions (d) and (e) of definition 3.

(ii) The conditions (a–c) of definition 3, in contrast, serve to ban impossible synchronisations: (a) and (b) forbid an action from synchronising with two different ones in the same run, i.e. they forbid

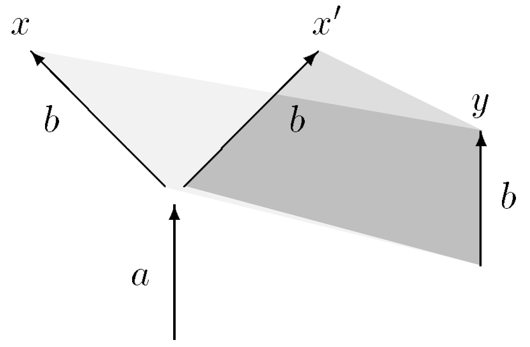


Figure 5: A multiple glueing.

Thus we cannot hope for glued forests to be ss $\mathcal{A}_{\mathbb{N}}$–cats in general, as condition (c) of definition 3 fails. We must instead look at the matter differently; first let us formally define valid glueing data between two trees:

Definition 16. A *glueing* γ between two ss $\mathcal{A}_{\mathbb{N}}$–cats $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$, written $\gamma : \mathcal{X} \rightarrow\!\!\!\!\!+ \mathcal{Y}$, is a function $\gamma : X \times Y \to \operatorname{arr}(\mathcal{A}_{\mathbb{N}})$ satisfying

(i) It glues runs together: $\gamma(x, y) \in \operatorname{Hom}[\varepsilon(x), \zeta(y)]$.

(ii) It does not glue roots together: $(0, 0, \epsilon) \notin \gamma(x, y)$ for any $x \in X$, $y \in Y$.

(iii) It respects agreement of runs on $\mathcal{X}$: $\gamma(x, y) \cdot \alpha(x', x) \subseteq \gamma(x', y)$.

(iv) It respects agreement of runs on $\mathcal{Y}$: $\beta(y, y') \cdot \gamma(x, y) \subseteq \gamma(x, y')$.

(v) It is symmetric: $\gamma(x, y) \cong \gamma(y, x)$ via the flip condition.

Conditions (iii) and (iv) ensure that if x and x' agree along a in $\mathcal{X}$ and we glue y along that a to x, then we have to glue it to x' along that a too; cf. point (iii) of discussion 14 above.

Fortunately, this notion of glueing is already well–known in the literature.

Definition 17. For $\mathcal{A}_{\mathbb{N}}$–categories $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$, a *bimodule* γ from $\mathcal{X}$ to $\mathcal{Y}$ is a function $\gamma : X \times Y \to \operatorname{arr}(\mathcal{A}_{\mathbb{N}})$ satisfying

(i) $\gamma(x, y) \in \operatorname{Hom}[\varepsilon(x), \zeta(y)]$.

(ii) It respects agreement of runs on $\mathcal{X}$: $\gamma(x, y) \cdot \alpha(x', x) \subseteq \gamma(x', y)$.

(iii) It respects agreement of runs on $\mathcal{Y}$: $\beta(y, y') \cdot \gamma(x, y) \subseteq \gamma(x, y')$.

We have again specialised our definition immediately to the case in hand; the general definition of bimodule over an enriched category is given in [2, 21].

The following is now immediate, and justifies the notation.

Theorem 18. A glueing γ between two trees $\mathcal{X}$ and $\mathcal{Y}$ is a bimodule $\gamma : \mathcal{X} \nrightarrow \mathcal{Y}$.

Example 15 shows that a pair of trees glued by a bimodule do not necessarily give rise to an $\mathcal{A}_{\mathbb{N}}$–category. We do, however, have

Proposition 19. If $\gamma : \mathcal{X} \nrightarrow \mathcal{Y}$ is a glueing between trees $\mathcal{X}$

(i) $Z = X \quad Y$ (modulo skeletality*)

(ii) $\eta(0,x) = \varepsilon(x)$, $\eta(1,y) = \zeta(y$

The inclusion functor of trees into forests (and hence that including trees in $\mathcal{A}\mathbb{N}$–**Cat**) clearly does not preserve coproducts, and so cannot have a right adjoint. There is a left inverse† which takes $\bigotimes_i \mathcal{X}_i$ and joins all the $\mathcal{X}_i$ together at the root, but this is of little concurrency–theoretic interest. Perhaps of more interest is the following characterisation of **Forest**:

Definition 26. A category $\mathcal{D}$ is the *free finite coproduct completion* [8] of a category $\mathcal{C}$ if the following two conditions are satisfied

(i) $\mathcal{D}$ has all finite coproducts and

(ii) For any category $\mathcal{C}'$ with all finite coproducts and any functor $\quad :$ $\mathcal{C} \to \mathcal{C}'$ there exists a finite coproduct preserving functor $\quad ' : \mathcal{D} \to \mathcal{C}'$ extending F which is unique up to a natural isomorphism.

Lemma 27. The following characterisation of free finite coproduct completion is equivalent [4] to the definition given above, and rather simpler to work with.

The free finite coproduct completion of a category $\mathcal{C}$, written $am(\mathcal{C})$, has as objects finite families $\{a_i\}_{i \in I}$ of objects of $\mathcal{C}$, and as arrows $\{a_i\}_{i \in I} \to \{b_j\}_{j \in J}$ pairs $(\phi, \{f_i\})$ where $\phi : I \to J$ is a **Set**–arrow, and $f_i : a_i \to b_{\phi(j)}$ is a $\mathcal{C}$–arrow.

Proposition 28. **Forest** is the free finite coproduct completion of **Tree**.

Definition 29. The *inter ection* of two $\mathcal{A}\mathbb{N}$–categories, $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$, written

We have seen that bimodules characterise glued trees. Thus it is natural to organise these into a category.

Definition 32. The category whose objects are bimodules $\mathcal{X} \dashrightarrow \mathcal{Y}$, for $\mathcal{X}$ and $\mathcal{Y}$ trees, and whose arrows $(\mathcal{X} \dashrightarrow \mathcal{Y}) \longrightarrow (\mathcal{X}' \dashrightarrow \mathcal{Y}')$ are pairs $(\ , G)$ where $\ : \mathcal{X} \to \mathcal{X}'$, $G : \mathcal{Y} \to \mathcal{Y}'$ are $\mathcal{A}\mathbb{N}$–functors satisfying

$$\gamma(x, y) \ \subseteq \ \gamma'(\ (x), G(y)) \qquad (*)$$

will be written **Bim**(**Tree**).[‡]

This definition makes sense when it is realised that since $\mathcal{X} \dashrightarrow \mathcal{Y}$ is almost an $\mathcal{A}\mathbb{N}$–category (cf. proposition 19), the right notion of arrow between such objects should be 'almost' $\mathcal{A}\mathbb{N}$–functors:

Lemma 33. If $(X \quad Y, \varepsilon \quad \zeta, \alpha \quad \beta \quad \gamma)$ is an $\mathcal{A}\mathbb{N}$–category, and similarly for γ', then $\qquad G$ is an $\mathcal{A}\mathbb{N}$–functor, so our requirement $(*)$ that the 'square'

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "|" marking] \arrow[d] & \mathcal{Y} \arrow[d, "G"] \\
\mathcal{X}' \arrow[r, "|" marking] & \mathcal{Y}'
\end{tikzcd}$$

should 'commute' then reduces to requirement (ii) of definition 20 i.e. that arrows between $\mathcal{A}\mathbb{N}$–categories should increase glueing.

It is clear that the category $\mathbf{Tree}^2$ (of pairs of trees and pairs of $\mathcal{A}\mathbb{N}$–functors) can be obtained from **Bim**(**Tree**) by forgetting glueing. In fact, this extends to an adjointness:

Proposition 34. The forgetful functor $U : \mathbf{Bim}(\mathbf{Tree}) \to \mathbf{Tree}^2$ defined by

$$U : (\mathcal{X}, \mathcal{Y}, \gamma) \longmapsto (\mathcal{X}, \mathcal{Y})$$

has a left adjoint, $\emptyset : \mathbf{Tree}^2 \to \mathbf{Bim}(\mathbf{Tree})$ which assigns the empty glueing to two trees:

$$\emptyset : (\mathcal{X}, \mathcal{Y}) \longmapsto (\mathcal{X}, \mathcal{Y}, \emptyset)$$

[‡] Note that this is *not* the usual notion of category of bimodules over a category. Indeed, as $\mathcal{A}\mathbb{N}$ is not locally cocomplete, we cannot even define the usual notion, as the colimit used to define the composition of $\gamma : \mathcal{X} \dashrightarrow \mathcal{Y}$, and $\delta : \mathcal{Y} \dashrightarrow \mathcal{Z}$, namely $\delta\gamma(x, z) = \mathrm{colim}_{y,y'} \delta(y', z) \cdot \beta(y, y') \cdot \gamma(x, y)$ need not exist.

5. Maximal Glueing

We have shown that glueing branches of trees together can be modelled as a bimodule. However, this is not quite enough to characterise CSP's parallel composition; for there is also an element of compulsion: if a synchronisation *can* happen, it *will*. In this section, we introduce a special class of glueings which allow us to include this element of compulsion.

Definition 35. Given two trees, $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$, and a pair of runs $x \in X$, $y \in Y$, a *CSP agreement* between x and y is an element of $\mathrm{Hom}[\varepsilon(x), \zeta(y)]$ satisfying

(i) $(0, 0, \epsilon) \notin f$ (it does not glue roots together)

(ii) If $(n, m, a) \in f$ then, for all $n' < n$, $(n', b) \in \varepsilon(x)$ implies either

  (a) $\exists m' < m$ such that $(n', m', b) \in f$, or

  (b) $\neg\exists m'$ such that $(m', b) \in \zeta(y)$.

  So that for all elements before a glued one, either they too are glued, or there is nothing they could possibly be glued to.

(iii) Vice versa for m.

Lemma 36. Conditions (i) and (ii) of definition 35 do indeed define a family of consistent subsets of $A_{\mathbb{N}\mathbb{N}}$.

Lemma 37. The CSP agreements between two runs are linearly ordered by inclusion.

Notation 38. We will write $\lambda_{x,y}$ for an arbitrary CSP–agreement between two runs, and $\kappa_{x,y}$ for the largest such, the existence of which is guaranteed by the previous proposition. Collections of such will be written $\{\lambda_{x,y}\}$ and $\{\kappa_{x,y}\}$.

We will use the maximal CSP–agreement $\kappa_{x,y}$ to define a CSP glueing between two tmtesde85yv9u(u7pliTd6)gTj[0.rG8.y)-15000.3(the)-16000ucion between the $\mathcal{A}_{\mathbb{N}}$–categories $|Y|\mathcal{X}$ and $|X|\mathcal{Y}$ as follows. (We write x_y for the run x in the y–indexed copy of $\mathcal{X}$.)

The component $\gamma_{y,x}(x'_y, y'_x)$: it will be given by first passing to x by $\alpha(x', x)$, thence to y via the CSP agreement $\kappa_{x,y}$, thence to y' via $\beta(y, y')$:

$$\gamma_{y,x}(x'_y, y'_x) \;=\; \beta(y, y') \cdot \kappa_{x,y} \cdot \alpha(x', x)$$

Proposition 40. For any two trees $\mathcal{X}$ and $\mathcal{Y}$, $\gamma_{y,x} : y\mathcal{X} \longrightarrow\!\!\!\!\!\!\mapsto x\mathcal{Y}$ is a glueing.

We have shown that the individual components of κ are glueings. It is also true that κ as a whole is:

Proposition 41. For any two trees, $\mathcal{X}$ and $\mathcal{Y}$, $\kappa : |Y|\mathcal{X} \longrightarrow\!\!\!\!\!\!\mapsto |X|\mathcal{Y}$ is a glueing.

The effect of the duplication can be seen by reconsidering a previous example:

Example 42. For our previous problematic example, 15, we get the glued forest shown in figure 6.

Notice that

Definition 43. The category whose objects are triples $(\mathcal{X}, \mathcal{Y}, \{\lambda_{x,y}\})$ with $\mathcal{X}$ and $\mathcal{Y}$ are trees and $\{\lambda_{x,y}\}$ is a collection of CSP glueings between runs $x \in X$ and $y \in Y$, and whose arrows $(\mathcal{X}, \mathcal{Y}, \kappa) \to (\mathcal{X}', \mathcal{Y}', \kappa')$ are pairs of $\mathcal{A}_{\mathbb{N}}$–functors $\; : \mathcal{X} \to \mathcal{X}'$, $G : \mathcal{Y} \to \mathcal{Y}'$ not decreasing glueing, will be written $\mathbf{CSP}(\mathbf{Tree}^2)$.

Again, we clearly have a forgetful functor $U : \mathbf{CSP}(\mathbf{Tree}^2) \to \mathbf{Tree}^2$, and identical reasoning to that used in proposition 34 shows that $\emptyset : \mathbf{Tree}^2 \to \mathbf{CSP}(\mathbf{Tree}^2)$ (which assigns the empty CSP–agreement to all runs) is its left adjoint. There is also a right adjoint:

Theorem 44. The functor $K : \mathbf{Tree}^2 \to \mathbf{CSP}(\mathbf{Tree}^2)$ defined by

$$K : (\mathcal{X}, \mathcal{Y}) \longmapsto (\mathcal{X}, \mathcal{Y}, \{\kappa_{x,y}\})$$

is right adjoint to U.

In summary, then, we have the following adjointness

$$\begin{tikzcd} \mathbf{CSP}(\mathbf{Tree}^2) \arrow[r, shift left=2] & \mathbf{Tree}^2 \arrow[l, shift right=4, "K"'] \arrow[l, shift left=2, "\emptyset"] \end{tikzcd}$$

where K is right adjoint to U is right adjoint to $\emptyset$. This gives a satisfactory account of two–way process synchronisation: the obvious generalisation to $\mathbf{CSP}(\mathbf{Tree}^n)$ then smoothly treats multiway synchronisations.

6. Concluding Remarks

The programme of enriched category theory, at least for Lawvere [14], is to seek out base categories enrichment over which will describe a variety of mathematical structures. The basic machinery of enrichment, in a particular case, is seen giving a generalised logic of the situation. Here we have presented an instance of this scheme, defining a category $\mathcal{A}_{\mathbb{N}}$ such that $\mathcal{A}_{\mathbb{N}}$–categories are descriptions of the behaviour of processes. Process synchronisation has been characterised as a bimodule in $\mathcal{A}_{\mathbb{N}}$–$\mathbf{Cat}$, and maximal glueing, corresponding to maximal synchronisation, has been shown to enjoy a simple universal characterisation.

There have been several other categorical accounts of process algebra, the work of Bednarczyk [1] and Winskel et al. [17, 23] being particularly notable. These workers, though, give an account relative to an arbitrary synchronisation algebra rather than geared specifically to synchronisation on common actions. This, of necessity, makes their account less specific and less tightly characterised than ours. It is clear that there are many categories of interest to the concurrency theorist: we merely hope to have added to the list.

Acknowledgments

All three author's research was partially supported by the Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the CNR, Contract Number 91.00894.69. The third author's research was supported by an SERC postdoctoral fellowship at the University of Sussex.

Bibliography

[1] M. Bednarczyk. *Categorie of A ynchronou Sy tem* . PhD thesis, Department of Computer Science, University of Sussex, 1987. Available as Technical Report Number 3/87.

[2] R. Betti, A. Carboni, R. Street, and R. Walters. Variation through enrichment. *Journal of Pure and Applied Algebra*, 29:109–127, 1983.

[3] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Network and ISDN Sy tem* , 14:25–59, 1987.

[4] A. Carboni and P. Johnstone. Connected limits, familial representability and artin glueing. In *Proceeding of Category Theory and Computer Science*, 1993.

[5] I. Castellani and G. Zhang. Parallel product of event structures. Rapports de Recherche 1078, INRIA, 1989.

[6] J.-Y. Girard. Linear logic and parallelism. In M. Venturini-Zilli, editor, *Mathematical Model for the Semantic of Paralleli m*, volume 280. Springer-Verlag LNCS, 1987.

[7] C. Hoare. *Communicating Sequential Proce e* . International series on computer science, Prentice-Hall, 1985.

[8] M. Johnson. *Pa ting Diagram in $n-$categorie with application to coherence theorem and categorie of*

[23] G. Winskel. Synchronization trees. *Theoretical Computer Science*, 34:34–84, 1985.

[24] G. Winskel. Categories of Models of Concurrency. Chapter in the *Handbook of Logic in Computer Science.* (S. Abramsky et al., Eds.), OUP, 1992.