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 Semantics for Observed 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, *Process Synchronisation as 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 Semantics for Observed Concurrency*, in the Proceedings of Mathematical Foundations of Computer Science (MFCS) 1992, Springer-Verlag LNCS 629.

A Functorial Semantics for Observed 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 thought that whatever can be known of me is by definition not me, is hetronomous to my authentic being, since the subject cannot be captured 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

---

Authors' Postal Address: School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH (on leave from the School of Computer Science, University of Birmingham) and GMD I5, Postfach 1316, 5205 Sankt Augustin 1, Germany.
E–Mail Addresses: dvm@computer-science.birmingham.ac.uk, ap@gmdzi.gmd.de.

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 good logical properties.

§1.1. The Conceptual Framework

The notions of *observation* and *behaviour* have always been central to concurrency theory, and are inextricably intertwined. Various ideas of behaviour have been proposed: traces, which record information about the sequence of events a system might display to an observer; branching time models, which also capture information about the states which can display different behaviour on different executions; and so–called 'true concurrency' models which also record information about the distribution of states or happenings: these are all important examples. Surveys of various models are given by Pnueli [22], the first author [20] and, for the timed case, Jeffrey [13].

The aim of this paper is to make the connection between behaviour and observation more precise and more abstract. Central to the notion of observation is a concept of *time*: we need to know at least when one observation happens after another, so we shall concentrate on observers who note *when* as well as *what* happens. Our elementary observations, then, will be of the form 'an event e happens at time t', for a suitably abstract idea of time. We shall explain the way the nature of the behaviour observed gives structure to these elementary observations. This structuring often gives rise to a canonical observer; one with the least structure necessary to see anything that might happen. Various 'worst case' processes, for which all of this observer's power is necessary, are found to be already well–known for other reasons.

This work requires a rather broader understanding of the notion of 'time' than is conventional. Time gives structure to observations (a point first explicitly made by Russell, but which has its roots in the thinking of Leibnitz: see [1] and [26]), and so must be the same kind of object as an (extended) observation. Thus branching time observations give rise to branching time, 'true' concurrency to many 'strands' of time and so

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 *traces* 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 *events*, 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 $\tau : E \to \mathbb{R}^+$ 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, so that $\mathtt{e} \leqslant_s \mathtt{f}$ if $\mathtt{e}$ comes before $\mathtt{f}$ in s. Then, given $\mathtt{e}, \mathtt{f} \in E$, if $\mathtt{e} <_s \mathtt{f}$, $\mathtt{f}$ should happen after $\mathtt{e}$, so for consistency we should have $\tau(\mathtt{e}) < \tau(\mathtt{f})$: whenever something happens, time passes.

Complete. Everything should happen at some time, and if we look for all time then we should see everything. This means that there is

a function $\alpha : \mathbb{R}^+ \to E$ that tells us the last thing that happened at $r \in \mathbb{R}^+$ satisfying

$$\alpha(r) = \mathtt{e} \quad \Longrightarrow \quad \tau(\mathtt{e}) \leq r \quad \wedge \tag{1}$$

$$\tau(\mathtt{e}) \leq \tau(\mathtt{f}) \leq r \quad \Longrightarrow \quad \mathtt{e} = \mathtt{f} \tag{2}$$

The connection between α and τ is thus

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

$$\tau(\alpha(r)) \quad \leq \quad r \tag{4}$$

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 will refine the meta-model.

Definition 3. The *behaviour* of a concurrent system P will be modeled as a category $\mathfrak{P}$ (which structure we present in due course).

Notice first that our notion of behaviour is very general, as many models of behaviour can be described as categories:

Traces. For a trace $s = (E, \leqslant_s)$, we get a skeletal category $\mathfrak{S}$, the objects being occurrences of events, and with an arrow from $\mathtt{e}$ to $\mathtt{f}$ iff $\mathtt{e} \leqslant_s \mathtt{f}$.

Branching time models. For a poset, $M = (E, \leqslant_m)$ we get a skeletal category $\mathfrak{M}$, the objects again being occurrences of events and the arrows are similarly generated by $\leqslant_m$.

True concurrency models. Symmetric monoidal categories or variants thereof have a close connection with both Petri nets (elaborated by Meseguer and Montanari amongst others [19]) and linear logic (discussed by, for example, Engberg and Winskel [11]). These categories, however, concentrate on the *structure* of nets, rather than their behaviour. Hence we will have cause to introduce new categories for true concurrency models.

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 notion of 'before' we want to model with the arrows of the category. (In general we think of there being an arrow $t \to t'$ if t is before t'.) The connection between time and behaviour can then be captured by an adjointness:

Definition 4. A canonical $\mathfrak{T}$–timing of a behaviour $\mathfrak{B}$ is a functor $f^* : \mathfrak{T} \to \mathfrak{B}$ and a full and faithful functor $f_! : \mathfrak{B} \to \mathfrak{T}$ left–adjoint to f^*.

The requirement that $f_!$ be full and faithful arises because time can have no more structure than that given to it by observation.

§2.3. Timed traces in the meta-model

Our timed trace model is described in this way by taking $\mathfrak{B}$ as the category $\mathfrak{S}$ derived from a trace s, and by taking $\mathfrak{T}$ as the category whose objects are the reals and where there is one arrow from r to s iff $r \leq s$. This category will be written $\mathbb{R}^+$.

Proposition 5. Any timing $\tau : E \to \mathbb{R}^+$ of a trace $s = (E, \leqslant_s)$ gives rise to a functor $f_! : \mathfrak{S} \to \mathbb{R}^+$ defined by $f_!(\mathtt{e}) = \tau(\mathtt{e})$ on objects.

The associated last thing function α gives rise to a functor $f^* : \mathbb{R}^+ \to \mathfrak{S}$ defined by $f^*(r) = \alpha(r)$. This is a coreflection right–adjoint to $f_!$.

Proof. Functoriality follows from the definitions. Adjointness for presets reduces to

$$\begin{array}{cc} f^* = f^* f_! f^* & f_! = f_! f^* f_! \\ \mathrm{id} \leqslant_s f^* f_! & f_! f^* \leq \mathrm{id} \end{array}$$

which follow from (3,4). For f^* to be a coreflection, $f_!$ must be full and faithful; the former is obvious since everything is skeletal, and the latter reduces to $\mathtt{e} \neq \mathtt{f} \implies \tau(\mathtt{e}) \neq \tau(\mathtt{f})$ which again holds.* □

---

* Faithfulness is the condition that things which are trace–ordered can't be simultaneous. For the consistency condition, we wrote $\mathtt{e} <_s \mathtt{f} \implies \tau(\mathtt{e}) < \tau(\mathtt{f})$ rather than the more obvious definition $\mathtt{e} \leqslant_s \mathtt{f} \implies \tau(\mathtt{e}) \leq \tau(\mathtt{f})$ to ensure that this holds.
This property, of a partial order $\leqslant_s$ factoring into a strict partial order $<_s$ plus equality is quite common in concurrency theory. In future we will without comment write $<_?$ for the strict version of any given partial order $\leq_?$.

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).

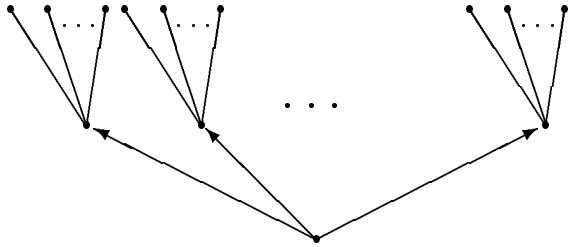
§3.1. Branching time

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 nondeterminism. 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 *process tree* $M = (E, \leqslant_m)$ gives rise to category $\mathfrak{M}$ as indicated in section 2.2. (The tree $\mathtt{e} \begin{array}{l} \nearrow \mathtt{f} \\ \searrow \mathtt{g} \end{array}$ 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 tree, $\mathcal{B} = (B, \leq_b)$, is constructed thus; take a single point and add a countable number of points, constructing a tree 1 branch high. Repeat the construction for each leaf, giving a tree 2 branches high, as below. (An arrow from p to q indicates $p \leq_b q$.)



To construct the whole tree, repeat the procedure a countably infinite number of times, leaving a structure where each branch starting from the root is countably infinitely long and each point displays countable nondeterminism.

Proposition 8. Any branching time behaviour M can be embedded in the Baire tree.

Our interest in the Baire tree is now clear; it is the one needed to time (certain) branching time processes. Without countable nondeterminism and finite causes, we might not be able to embed M into $\mathcal{B}$.

These properties (which constrain the allowed branching and 'length' of behaviours) are instance of a more general property of *realisability*:

Definition 9. A (real–timed) process is *realisable* iff it satisfies all of the following conditions.

(i) in any finite interval of time it can only display a finite number of events (i.e. no Zeno machines),

(ii) at any state only a countable number of states are possible next and

(iii) no event depends on more than a finite number of previous events (i.e. finite causes). In particular this means that no action happens more than a finite number of times, and the set E is thus countable.

A process which violates condition (i) is a *Zeno machine*; these are discussed by Joseph in [14]. Conditions (ii) and (iii) are known respectively as 'countable nondeterminism' and 'finite causes'. The difficulties associated with doing away with (ii) are discussed by Bergstra and Klop [2], (iii) being treated by Best and Fernandez [4]. For the remainder of the paper we will confine our attention to realisable processes.

Definition 10. A abstract timing of a branching time behaviour M is a function $\tau : E \to B$ satisfying

Consistency. $\mathtt{e} <_m \mathtt{f} \implies \tau(\mathtt{e}) <_b \tau(\mathtt{f})$.

Completeness. There is a 'last thing' function $\alpha : B \to E$ satisfying

$$\begin{aligned} \alpha(r) = \mathtt{e} \quad \implies \quad & \tau(\mathtt{e}) \leq_b r \quad \wedge \\ & \tau(\mathtt{e}) \leq_b \tau(\mathtt{f}) \leq_b r \quad \implies \quad \mathtt{e} = \mathtt{f} \end{aligned}$$

Proposition 11. The connection between τ and α is as before:

$$\alpha(\tau(\mathtt{e})) \;=\; \mathtt{e} \qquad \text{and} \qquad \tau(\alpha(p)) \;\leq_b\; p$$

Moreover, if $\mathtt{e}$ is not the last action of a branch, then we can 'extend' our knowledge at $\tau(\mathtt{e})$, seeing more sometime later

$$\mathtt{e} <_m \mathtt{f} \quad \implies \quad \exists p \in B \,.\, \tau(\mathtt{e}) <_b p \;\wedge\; \alpha(p) = \mathtt{f}$$

Thus earlier observations are subobjects of later ones.

Proposition 12. Any timing $\tau : E \to B$ of a branching time process M gives rise to a functor $f_! : \mathfrak{M} \to \mathcal{B}$ defined by

$$f_!(\mathtt{e}) = \tau(\mathtt{e})$$

The associated last thing function α gives rise to a functor $f^* : \mathcal{B} \to \mathfrak{M}$ defined by

$$f^*(p) = \alpha(p)$$

This functor is a coreflection right–adjoint to $f_!$.

The 'worst case' process, for which f^* as well as $f_!$ is full and faithful, is the `Chaos` of CSP,

$$\mu x \,.\, \sqcap_{\mathtt{a} \in A} \mathtt{a} \to x$$

which clearly gives a process tree isomorphic to $\mathcal{B}$. (Thus the CSP refinement ordering $\sqsubseteq$ given in [5] is derivable from the inclusion ordering on subtrees of $\mathcal{B}$.)

We may be interested in real–timing a branching time behaviour, i.e. in ignoring the natural structure of observations and just attaching a real to each node of $\mathfrak{M}$ in a way that respects $\leq_b$. To do this, we need merely supply a functor from $\mathfrak{M}$ to $\mathbb{R}^+$. This factors as a composite

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

$$\mathfrak{M} \underset{f^*}{\overset{f_!}{\rightleftarrows}} \mathcal{B} \xrightarrow{U} \mathbb{R}^+$$

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 process* 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 $\mathtt{e} \begin{smallmatrix} \nearrow \mathtt{f} \\ \searrow \mathtt{g} \end{smallmatrix}$ 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 Time. $r^{\mathtt{a}} \leq_a s^{\mathtt{b}} \iff r \leq s \wedge \mathtt{a} = \mathtt{b}$. That is, two times are only related if they are possible timing of occurrences of the same action.

Global Time. $r^{\mathtt{a}} \leq_a s^{\mathtt{b}} \iff r \leq s$. That is, times are related by the usual order of $<$ on $\mathbb{R}^+$; observers have synchronised clocks.

To see which order suffices, we need a notion of timing a purely concurrent process:

Definition 14. A *purely concurrent process timing* is a function $\tau : E \to (\mathbb{R}^+)^{|L|}$ satisfying

Consistent. This is causal ordering implies temporal ordering as usual

$$(l, \mathtt{a}) <_q (m, \mathtt{b}) \implies \tau(l, \mathtt{a}) <_a \tau(m, \mathtt{b}) \tag{5}$$

Complete. This is the existence of a last thing function $\alpha : (\mathbb{R}^+)^{|L|} \to E$ satisfying[†]

$$\alpha(r^{\mathtt{a}}) = (l, \mathtt{a}) \implies \tau(l, \mathtt{a}) \leq_a r^{\mathtt{a}} \quad \wedge \tau(l, \mathtt{a}) \leq_a \tau(m, \mathtt{b}) \leq_a s^{\mathtt{a}} \tag{6}$$

$$\implies l = m \;\&\; \mathtt{a} = \mathtt{b} \tag{7}$$

Unfortunately things break down here:

Proposition 15. The equations (5) and (6), which make τ and α respectively functorial cannot in general both be satisfied by either the local time version of $\leq_a$ or the global time version.[‡]

Proof. If we take local time, then given $(l, \mathtt{a}) \leqslant_q (m, \mathtt{b})$ with $\mathtt{a} \neq \mathtt{b}$ we can't have $\tau(l, \mathtt{a}) \leq_a \tau(m, \mathtt{b})$. Conversely, if we take global time, we can have $r^{\mathtt{a}}$ and $s^{\mathtt{b}}$ with $r < s$ hence $r^{\mathtt{a}} \leq_a s^{\mathtt{b}}$ but there need be no connection between $\alpha(r^{\mathtt{a}})$ and $\alpha(s^{\mathtt{b}})$. □

In other words, neither local nor global time leads to canonical observers for purely concurrent processes (and hence for any extension of them incorporating nondeterminism.) There are some special cases in which we can get the desired result;

No causality. Here there is no connection between different actions, and we can setup the expected adjunction. The 'worst case' process here is

$$\Big|\Big|_{\mathtt{a} \in A} (\mu x \,.\, \mathtt{a} \to x)$$

Linear time. If $\mathtt{e} \leqslant_q \mathtt{f}$ or $\mathtt{e} \leqslant_q \mathtt{f}$ for all $\mathtt{e}, \mathtt{f} \in E$ then local time is just global time and again the adjunction goes through. This is just the phenomena familiar from Lamport's [17] of global time forbidding (true) concurrency; there is no canonical way of linearising a poset that isn't already linear.

[†] This definition hides the somewhat unjustified assumption that if $\tau(l, \mathtt{a}) = r^{\mathtt{a}}$ and $\tau(m, \mathtt{b}) = r^{\mathtt{b}}$ then $l = m$ and $\mathtt{a} = \mathtt{b}$, in other words, there are no simultaneous happenings.

[‡] Thomason, in [24], has noted a similar problem in assigning *durations* to events, so moving to nonatomic events will not help.

If the observers know the structure of causality before hand, of course, there is a canonical observer, but this can hardly be regarded as in the spirit of pure observation:

Proposition 16 (Observing Causality). For a purely concurrent process $Q = (E, \leqslant_q, 0)$, we can define timing and last thing functions that give rise to adjoint functors $f_! : \mathfrak{Q} \to (\mathbb{R}^+)^{|L|}$, $f^* : (\mathbb{R}^+)^{|L|} \to \mathfrak{Q}$ with $f_!$ faithful if $(\mathbb{R}^+)^{|L|}$ carries an ordering given by the causality of Q:

$$r^{\mathsf{a}} \leq_a s^{\mathsf{b}} \quad \Longleftrightarrow \quad \alpha(r^{\mathsf{a}}) \leqslant_q \alpha(s^{\mathsf{b}})$$

We *can* interpret this result observationally; consider a set of observers, one for each action, as in the Petri net example after definition 13. If these observers can see causality—a perfectly reasonable assumption, for instance, in an ethernet, if they come equipped with network analysers—then they are collectively canonical. It is reassuring to note that Mattern's algorithm [18] for assigning virtual time to a purely concurrent process ensures precisely that it carries the ordering $\leq_a$ defined above.

§3.3. True concurrency

Suppose we have an event structure [27] X and its associated domain of configurations $\mathcal{F}(X)$. The following is standard: see below for the appropriate order–theoretic terminology.

Proposition 17. If X is a prime event structure, then $(\mathcal{F}(X), \subseteq)$ is a dI-domain. If X is conflict–free, then $(\mathcal{F}(X), \subseteq)$ is also a frame.

This observation gives us key to one structure with which to time true concurrency. Consider a prime event structure X based on a set of events E. The poset $(2^{\mathrm{fin}(E)}, \subseteq)$ (of finite subsets of E ordered by inclusion) is a frame. Since we want to deal only with realisable structures, $\mathcal{F}_{\mathbb{N}} = (2^{\mathrm{fin}(\mathbb{N})}, \subseteq)$ seems a possible candidate for structuring observers of configurations. The next proposition shows that we could define timed configurations using this structure and they would fit into the pattern.

Proposition 18. Suppose X is a prime event structure. Then there is an adjunction $f^* \vdash f_!$ between the configurations $\mathcal{F}(X)$ and $\mathcal{F}_{\mathbb{N}}$, with $f_! : \mathcal{F}(X) \hookrightarrow \mathcal{F}_{\mathbb{N}}$.

The observational content of this proposition is that observers of configurations can safely structure themselves using just the inclusion order of finite sets of events; the worst an event structure can do is display all its events at once, giving in the configuration structure $(2^{\mathrm{fin}(E)}, \subseteq)$ which, for countably infinite E, is isomorphic to $\mathcal{F}_{\mathbb{N}}$.

§4. The Structure of All Observations

We now go on to discover structure beyond the adjunctions of the last sections. First, some standard definitions from order theory:

Definition 19 (Order–theoretic structures). A poset $(D, \sqsubseteq)$ with a least element $\bot$ is said to be *complete* if every directed subset has a least upper bound. A monotone function between posets is *continuous* if it preserves lubs of directed sets.

A point $x \in D$ is said to be *isolated* if, for every directed subset $M \subseteq D$ such that $x \sqsubseteq \bigsqcup M$, there is a $y \in M$ such that $x \sqsubseteq y$. The collection of isolated elements of a complete poset D is written I_D, and D is said to be *algebraic* if, for every $x \in M$, the set $M = \{x_0 \in I_D \mid x_0 \sqsubseteq x\}$ is directed and has lub x. Complete algebraic posets are often called *domains*.

If a subset $M \subseteq D$ is bounded, then we write $M\uparrow$. A domain is said to be *bounded complete* iff every directed subset has a lub. Notice that in a bounded complete domain, binary glbs (written $\sqcap$) exist.

A point $x \in D$ is said to be *very finite* if there are only finitely many points y such that $y \sqsubseteq x$.

A bounded complete domain D where every isolated point is very finite and where following the distribution law holds

$$\{y, z\}\uparrow \implies x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$$

is said to be a *dI-domain*. This definition is due to Berry, a good reference (which also touches on the connection with event structures) being Coquand et al. [8].

A *frame* is a poset with all lubs and binary glbs, satisfying the stronger distributive law

$$x \sqcap \bigsqcup M \;=\; \bigsqcup \{x \sqcap y \mid y \in M\}$$

Notice in particular that frames have a top element.

A function $f : D \to E$ is *stable* iff it is continuous and preserves glbs of bounded pairs.

§4.1. Moving on

These definitions can be used to give deeper insight into the meta–model of the last few sections. Recall definition 4;

> A canonical $\mathfrak{T}$–timing of a behaviour $\mathfrak{B}$ is a functor $f^* : \mathfrak{T} \to \mathfrak{B}$ and a full and faithful functor $f_! : \mathfrak{B} \to \mathfrak{T}$ left–adjoint to f^*.

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 $f : t \to t'$ 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 *observation category* iff

(i) it is essentially small. That is, its family of objects quotiented by isomorphism forms a set. It is very difficult to think of an observational setting where this is not true, so this is not a severe restriction.

(ii) all morphisms are monos. That is, a map from one observation to another is just an inclusion. This emphasises the fact that time has an arrow, and this is points in the directions of the arrows of the category: latter observations arise only as extensions of earlier ones.

(iii) it has all filtered colimits. That is, given any family of comparable observations, i.e. a connected diagram in $\mathfrak{C}$, there is a unique smallest observation including all those in the family i.e. a colimit for the diagram.

(iv) for each object, t, $\mathbf{sub}(t)$ forms a complete dI-domain. This is the most inscrutable condition; it requires that subobservations of a given observation have glbs, lubs of directed sets, and is, in a technical sense, finitary.

Loosely speaking, this means that given any set of observations, we can say the least subobservation they have in common, and, if they are all part of one observation, we can say what that observation is. Further details on the connection between (dI–)domains and observation can be found in Vickers' book [25].

(v) each arrow $f : t \to t'$ induces a rigid embedding/projection pair

$$\mathbf{sub}(t) \underset{f^*}{\overset{f_!}{\rightleftarrows}} \mathbf{sub}(t') \tag{8}$$

(so in particular $f^* f_! = \mathrm{id}$, as in the examples.)

Recall that we want to think of the subobjects of a given object as the subobservations possible of it. The condition given is then that if we have a time later than now, we can extend our observations to those possible in the future, and we can cut down future observa-

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 categories

The conditions (i–v) for a category $\mathfrak{C}$ to be an observation category suffice to prove several technical propositions that will be of use in determining the structure of all observations. In what follows assume $\mathfrak{C}$ is an observation category.

Proposition 21. $\mathfrak{C}$ has all pullbacks. This means that given an observation which extends two others, we can always find a greatest observation which can be extended into each of the two. It also means f^* can be defined by pulling back along f.

Proposition 22. For any object t of $\mathfrak{C}$, $\mathbf{sub}(t)$ has directed lubs.

Proposition 23. The functors f^* and $f_!$ preserve directed lubs.

The notion of observation category turns out to be already well-known under another name; our observation categories are exactly Coquand's stable categories of embeddings.

Proposition 24. An observation category is a stable category of embeddings. Moreover, all such categories arise this way.

This proposition means that we can lift results straight from [7] and use them in our setting. Furthermore, *op. cit.* gives other character-

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 *events* 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 *stable* if it preserves pullbacks, and *continuous* 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, $\mathrm{SC}[\mathfrak{C}, \mathfrak{D}]$, 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 boils down to showing that: –

1. **Tr** is essentially small. This follows since there is one isomorphism class for each $n \in \mathbb{N}$, all traces of the same length being isomorphic. No trace is longer than countable long, hence **Tr** is indeed essentially small.
2. All the maps in **Tr** are monos. This is obvious.
3. **Tr** has filtered colimits. This follows since $\Omega = (\mathbb{N}, \leq)$ is weakly terminal, and $\mathbf{sub}(\Omega)$ has directed sups. **Tr** thus inherits its filtered colimits from $\mathbf{sub}(\mathbb{N})$.
4. For any trace t, $\mathbf{sub}(t)$ is a complete dI-domain. The only case that requires thought is $\mathbf{sub}(t)$ for t countable, the lattice otherwise being finite and plainly a dI-domain. For the countable case, the isolated elements are the subtraces of finite length, with t itself being the only non-isolated element. Hence the lattice is a domain and all isolated points are very finite.

   All directed sets have least upper bounds, inherited from $\mathbb{N}$. Glbs are constructed as in $\mathfrak{Set}$, with the order inherited from the upper bound; the distributive law then holds.
5. Given a map $f : t \to t'$, the stable embedding/projection pair (8) exists. Construct f^* by pulling back along f, and $f_!$ by composition with f. Adjointness is automatic; we are left with showing that f^* is continuous. But this follows since an ω-chain in the image of f only arises as an ω-chain in the preimage.

Branching time. The proof of the case for traces was rather tedious; fortunately we can use proposition 26 to simplify the argument. Notice that the category of finite trees with embeddings as morphisms is a category of events. It is then routine that the category of countable trees is its ind-completion with the Baire tree as a weak terminator and hence is an observation category.

Event structures. Coquand et al. prove that the configuration structure of a prime event structure is a dI-domain, and that all dI-domains arise this way [8]. It is then known that the category of dI-domains and stable embedding/projection pairs is a category of embeddings [7].

□

§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 such functors, showing how they may be used to exploit some of the gross structure in categories of embeddings.

Our framework can be used to prove general theorems which specialise to individual concurrency theories. For instance, the following proposition concerning recursively defined objects is almost trivial for categories of embeddings, but rather tedious to prove for individually for traces, branching time, event structures, ...

Proposition 31. Suppose we have a functor $F \in \mathrm{SC}[\mathfrak{C}, \mathfrak{C}]$. Then the diagram

$$\bot \hookrightarrow F(\bot) \hookrightarrow F^2(\bot) \hookrightarrow \ldots$$

is filtered, and hence has a colimit in $\mathfrak{C}$ which the least fixed point of F.

We have seen that a weakly terminal object in an observation category plays the rôle of time, its structure being that of a canonical observer. It should, then, be possible to interpret one model in another by giving some $F \in \mathrm{SC}[\mathfrak{C}, \mathfrak{D}]$ which preserves the weakly-terminal object. It is also reasonable to require that F is *strict* (preserves the initial object) and faithful as an example demonstrates:

Example 32. Consider the categories $\mathbb{N}$ and $\mathcal{B}$ that are the categories of embeddings representing traces and branching time respectively. To give an interpretation of linear time in branching time, it suffices to give a functor $F \in \mathrm{SC}[\mathbb{N}, \mathcal{B}]$ which preserves the weakly terminal object. Our interpretation functors, then, assign finite trees to finite traces. Strictness is the property that empty observations map to empty observations, and faithfulness that increased information in one model gives increased information in the other.

The fact that $\mathrm{SC}[\mathfrak{C}, \mathfrak{D}]$ is itself an observation category gives structure to these interpretation functors: we can talk of the meet and directed join of two interpretations, for instance.

§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 $F : \mathfrak{B}^{\mathrm{op}} \to \mathfrak{Set}$, 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, *Process theory based on bisimulation semantics*, 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 abstract machine*, in Principles of Programming Languages, 1990, Proceedings of the 17th ACM Symposium.

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

[5] S. Brookes and A. Roscoe, *An improved failures model for communicating sequential processes*, 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 structures*, in Category Theory and Computer Science, Volume 389, Springer-Verlag LNCS, 1989.

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

[8] T. Coquand, C. Gunter, and G. Winskel, *dI-domains as a model of polymorphism*, 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 objects as observed processes*, in the Proceedings of REX/FOOL, 1991.

[11] U. Engberg and G. Winskel, *Petri nets as models of linear logic*, in the Proceedings of TAPSOFT (A. Arnold, Ed.), Volume 431, Springer-Verlag LNCS, 1990.

[12] C. Hoare, *Communicating sequential processes*, International series on computer science, Prentice-Hall, 1985.

[13] A. Jeffrey, *Timed process algebra* $\neq$ *time* $\times$ *process algebra*, Technical Report 79, Programming Methodology Group, Chalmers University, 1991.

[14] M. Joseph and A. Goswami, *Relating computation and time*, Technical Report RR 138, Department of Computer Science, University of Warwick, 1985.

[15] S. Kasangian and A. Labella, *On continuous real time agents*, in Mathematical Foundations of Programming Semantics (I. Guessarian, Ed.), Volume 469, Springer-Verlag LNCS, 1991.

[16] R. Koymans, *Specifying real–time properties with metric temporal logic*, Real–Time Systems, Volume 2 (1991), Pp. 255–299.

[17] L. Lamport, *On interprocess communication. part I: Basic formalism*, Distributed Computing, Volume 1 (1986), Pp. 77–85.

[18] F. Mattern, *Virtual time and global states of distributed systems*, in Parallel and Distributed Algorithms (M. Cosnard et al., Ed.), North-Holland, 1989.

[19] J. Meseguer and U. Montanari, *Petri nets are monoids: A new algebraic foundation for net theory*, in Logic in Computer Science, pp. 155–164, 1988, Proceedings of the 3rd Annual IEEE Symposium.

[20] D. Murphy, *Time, causality, and concurrency*, Ph.D. thesis, Department of Mathematics, University of Surrey, 1989, available as Technical Report CSC 90/R32, Department of Computing Science, University of Glasgow.

[21] D. Murphy and W. Phoa, *A categorical temporal logic*, presented at the Durham Applications of Categories in Computer Science Meeting, 1991.

[22] A. Pnueli, *Linear and branching structures in the semantics and logics of reactive systems*, in Automata, Languages and Programming (W. Brauer, Ed.), Volume 194, Springer-Verlag LNCS, 1986, (12th Coll.).

[23] M. Shields, *Cube complexes as automata*, Technical Report CS–91–04, Department of Computing Sciences, University of Surrey, 1991.

[24] S. Thomason, *Free construction of time from events*, Journal of Philosophical Logic, Volume 18 (1989), Pp. 43–67.

[25] S. Vickers, *Topology via logic*, Tracts in Theoretical Computer Science, Volume 5, Cambridge University Press, 1989.

[26] G. Whitrow, *The natural philosophy of time*, Oxford University Press, 1980.

[27] G. Winskel, *Event structures*, in Petri Nets: Central Models and Their Properties (W. Brauer, W. Reisig, and G. Rozenberg, Eds.), Volume 254, Springer-Verlag LNCS, 1986, Proceedings of Advances in Petri Nets. Also available as a Cambridge University Computer Laboratory Technical Report.

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, and indicate how it can be extended to a model of full process algebra. Our main tool is the notion of bimodule over an enriched category: this turns out to be precisely the machinery needed to glue the behaviours of processes together and thus describe synchronisation. Maximal (CSP–style) synchronisation can then be given a universal characterisation.

1. Introduction

This paper presents a new model of process synchronisation. Intuitively, the behaviour of a system of processes which cooperate by synchronising on common actions can be understood as the behaviours of the sequential subprocesses glued together on synchronisations. This intuition is made precise, allowing us to give a semantics to the parallel composition of process algebras like CSP with this form of multiway cooperative synchronisation discipline [3, 5, 7, 19, 20] and more generally to provide a new categorical semantics for synchronising processes.

Our methods are categorical: an enriched category, whose objects include labelled trees, is introduced and shown to have enough structure to model many process algebraic combinators. This category is new, and of independent interest. A bimodule construction is then introduced which allows us to glue trees together to form the behaviour of a concurrent, cooperating system. A characteristic feature of CSP is compulsory synchronisation: parallel processes are forced to cooperate. This notion of maximal glueing is characterised as a right adjoint.

Space constraints force us to assume a basic familiarity with process algebra and labelled tree semantics. All the enriched category theory we use is defined: further insight, and more general results, can be found in the standard references [2, 12, 21]. Space also forces us to focus on process synchronisation here: a full account of process algebra semantics in our enriched setting, including the change-of-base machinery necessary to deal with hiding, all proofs, and much more detail on the process–theoretic implications of our account, can be found in [10]: cf. [24].

We conclude these introductory remarks by giving an example of the objects under consideration. Consider the process $a\,.\,(b\,.\,0{+}c\,.\,0) \mid\mid b\,.\,0$.

This will be modelled by the forest shown in figure 1, where the shaded area represents a glueing between the two synchronising bs. The reader may like to think of plastic sheeting (representing the communication) stretching between the glued branches.

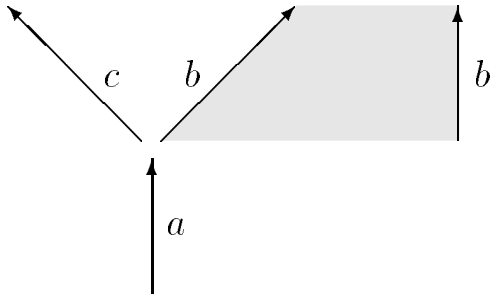


Figure 1: A b–synchronisation represented as a glueing.

2. Trees as Enriched Categories

We will model a labelled tree as a set of runs together with information as to where runs agree with each other. Thus the upper tree in figure 2, modelling the process

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

will be described by giving two runs, x and y say, representing the complete computations ab and ac, together with the information that these runs agree on the initial a.

The agreement data, then, distinguishes the upper tree from the lower since in the latter, the agreement between x and y is empty. This second tree, of course, represents the process $Q = (a\,.\,b\,.\,0) + (a\,.\,c\,.\,0)$, which is often distinguished from P [16].

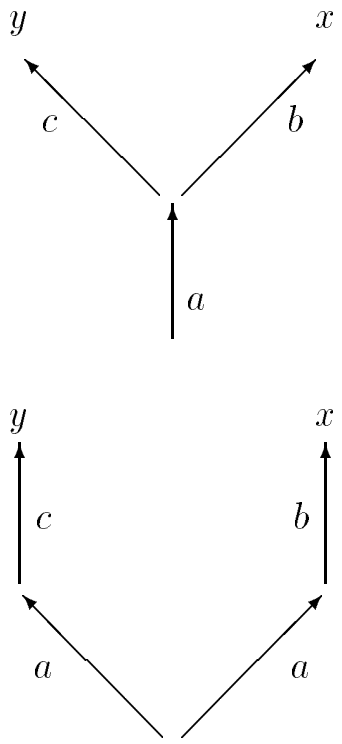


Figure 2: Two trees illustrating the rôle of agreement data.

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) *consistent* 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 closed* 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 $a_1 a_2 \ldots a_m \leftrightarrow \{(0, \epsilon), (1, a_1), \ldots, (m, a_m)\}$.

The sets $\{(1,a),(3,b)\}$ and $\{(1,a)\}$ are consistent but not prefix closed, while $\{(2,a),(2,b)\}$ and $\{(1,a),(3,\epsilon)\}$ are not even consistent.

We will use prefix closed subsets of $A_{\mathbb{N}}$ to label the computations along branches of a tree: agreements require a different structure:

Definition 3. Let $A_{\mathbb{NN}} = \mathbb{N} \times \mathbb{N} \times (A \cup \{\epsilon\})$. A subset $f \subseteq A_{\mathbb{NN}}$ is called

(i) *consistent* if

(a) $(n,m,a) \in f$ and $(n,m',a') \in f$ implies $m = m'$ and $a = a'$.

(b) $(n,m,a) \in f$ and $(n',m,a') \in f$ implies $n = n'$ and $a = a'$.

(c) $(n,m,a) \in f$ and $(n',m',a') \in f$ implies $n < n'$ iff $m < m'$.

(d) $(n,m,\epsilon) \in f$ implies $n = m = 0$.

(e) $(0,m,a) \in f$ or $(n,0,a) \in f$ implies $n = m = 0$ and $a = \epsilon$.

(ii) *prefix closed* if it is consistent and there is an $m \in \mathbb{N}$ such that $f = \{(n,n,a_n) \mid n \leq m\}$.

Example 4. The tree representing $a.(b+c)$ can be specified by giving a set $\{x,y\}$ of runs, a function ε assigning the following computations (= prefix closed subset of $A_{\mathbb{N}}$) to runs $\varepsilon(x) = \{(0,\epsilon),(1,a),(2,b)\}$, $\varepsilon(y) = \{(0,\epsilon),(1,a),(2,c)\}$, and a function α assigning the agreement (= consistent subset of $A_{\mathbb{NN}}$) $\{(0,0,\epsilon),(1,1,a)\}$ to x and y.

An alternative characterisation of consistent subsets of $A_{\mathbb{NN}}$ will be useful: it is straightforward to prove

Lemma 5. A consistent set $f = \{(n_1,m_1,a_1),\ldots,(n_i,m_i,a_i)\}$ is equivalent to a strict monotone partial bijection $\mathbb{N} \to \mathbb{N}$, given by $\{(n_1,m_1),\ldots,(n_i,m_i)\}$, together with a partial labeling $\mathbb{N} \to (A \cup \{\epsilon\})$ defined wherever the bijection is and labeling only 0 with ϵ.

Proposition 6. The structure $\mathcal{A}_{\mathbb{N}}$ with

(i) *objects* prefix closed subsets of $A_{\mathbb{N}}$,

(ii) *arrows* $f : S \to T$ consistent subsets of $A_{\mathbb{NN}}$ such that $\pi(f) \subseteq S$, $\pi'(f) \subseteq T$ where
$\pi\{(n_1,m_1,a_1),\ldots,(n_l,m_l,a_l)\} = \{(n_1,a_1),\ldots,(n_l,a_l)\}$ and
$\pi'\{(n_1,m_1,a_1),\ldots,(n_l,m_l,a_l)\} = \{(m_1,a_1),\ldots,(m_l,a_l)\}$,

(iii) *2-cells* inclusions

is a semilattice–enriched category (i.e. each Hom is a semilattice) where composition is given by intersection.

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 *runs*);

(ii) $\varepsilon : X \to \mathrm{ob}(\mathcal{A}_{\mathbb{N}})$ is a function (the *extent* function, assigning a computation to each run);

(iii) $\alpha : X \times X \to \mathrm{arr}(\mathcal{A}_{\mathbb{N}})$ is a function (the *agreement* function, which says to what extent two runs agree).

We require that

(a) runs agree along some subset of their extents:

$$\alpha(x, y) \in \mathrm{Hom}[\varepsilon(x), \varepsilon(y)]$$

(b) a run agrees with itself along all its length:

$$\alpha(x, x) = \mathrm{id}_{\varepsilon(x)}$$

(c) the agreement between x and z is no less than the composition of that between x and y and that between y and z (for any y):

$$\alpha(y, z) \cdot \alpha(x, y) \subseteq \alpha(x, z)$$

An $\mathcal{A}_{\mathbb{N}}$–category $\mathcal{X} = (X, \varepsilon, \alpha)$ is called

(iv) *symmetric* iff $\alpha(x, y)$ is canonically isomorphic with $\alpha(y, x)$) (via the *flip condition*: $(n, m, a) \in \mathrm{Hom}[S, T]$ iff $(m, n, a) \in \mathrm{Hom}[T, S]$).

(v) *skeletal* iff $\mathrm{id}_{\varepsilon(x)} = \mathrm{id}_{\varepsilon(y)} = \alpha(x, y)$ implies $x = y$.

If $\mathcal{X}$ is a skeletal symmetric $\mathcal{A}_{\mathbb{N}}$–category, we often just say that it is an *ss* $\mathcal{A}_{\mathbb{N}}$*–cat*.

Proposition 8. Finite A–labelled trees correspond to ss $\mathcal{A}_{\mathbb{N}}$–categories where the agreement $\alpha(x, y)$ is a prefix closed set for any x and y and vice versa.

This proposition justifies us using the name *trees* for ss $\mathcal{A}_{\mathbb{N}}$–categories with $\alpha(x, y)$ always prefix closed. Further details of the use of these categories are given in references [9, 11, 18]. In the next section, we present a construction that allows us to glue trees together: we conclude here by giving examples of $\mathcal{A}_{\mathbb{N}}$–categories that are not trees.

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

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

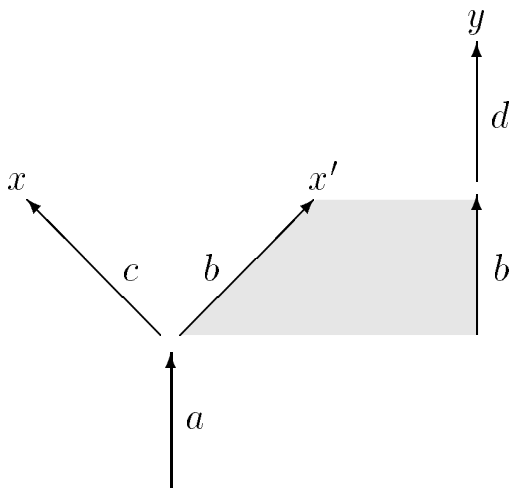
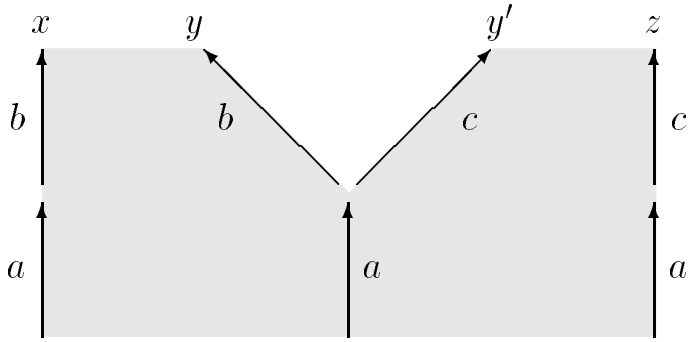


Figure 3: An example glueing.

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

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



Discussion 14. This example demonstrates several important points:

(i) We indicate that y and y' are runs *on the same 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

the causally impossible glueing $\gamma(x,y) = \{(1,1,a),(2,1,a)\}$ in the righthand forest of figure 4.

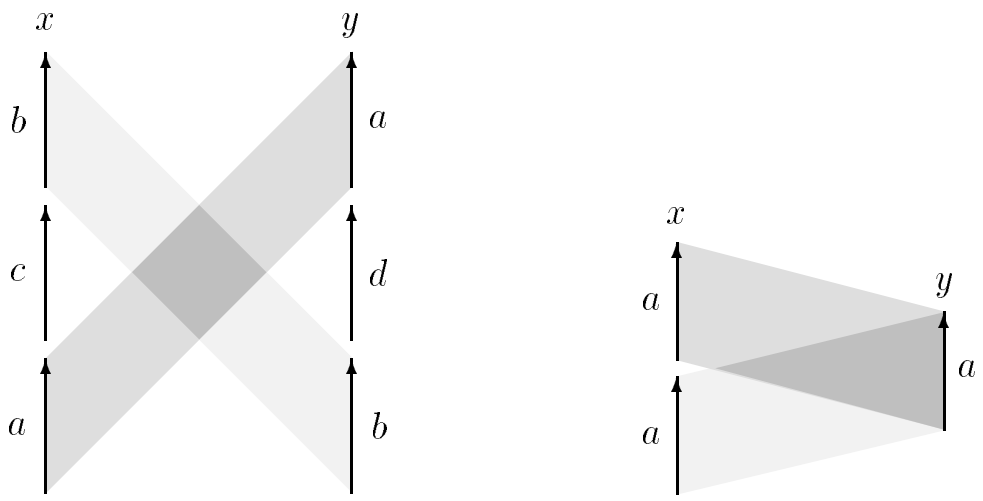


Figure 4: Two undesirable glueings.

Similarly, condition (c) forbids another pathological situation; $\gamma(x,y) = \{(1,3,a),(3,1,b)\}$ in the lefthand glued forest of figure 4.

(iii) Example 13 might lead us to suppose that glueing is always transitive, for certainly in this case we have

(a) $\gamma(y',z) \cdot \beta(y,y') \subseteq \gamma(y,z)$ and

(b) $\beta(y',y) \cdot \gamma(z,y') \subseteq \gamma(z,y)$.

Indeed, in general, for x, y, z all on different trees we expect that if x is glued to y along a consistent f and y to z along g, then x is glued to z at least along $g \cdot f$. The following example, however, shows that glueing is not completely transitive.

Example 15. Consider the process $a\,.\,(b\,.\,0 + b\,.\,0) \;||\; b\,.\,0$. We have the glued forest shown in figure 5 with runs x, x' and y where

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

and clearly $\gamma(y,x') \cdot \gamma(x,y) = \{(2,2,b)\}$ which is not contained in $\alpha(x,x')$.

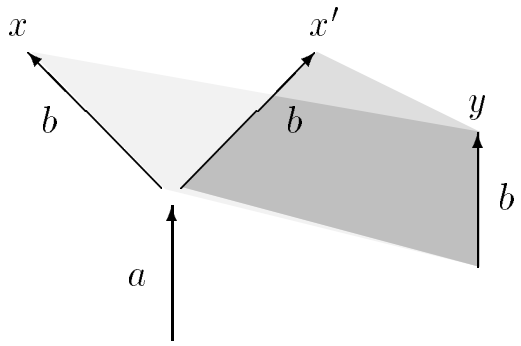


Figure 5: A multiple glueing.

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 $A_{\mathbb{N}}$–cats $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$, written $\gamma : \mathcal{X} \nrightarrow \mathcal{Y}$, is a function $\gamma : X \times Y \to \mathrm{arr}(\mathcal{A}_{\mathbb{N}})$ satisfying

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

(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 $\gamma : X \times Y \to \mathrm{arr}(\mathcal{A}_{\mathbb{N}})$ satisfying

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

(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 $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$ satisfying

(i) $\gamma(y, x') \cdot \gamma(x, y) \subseteq \alpha(x, x')$ (left transitivity)

(ii) $\gamma(x, y') \cdot \gamma(y, x) \subseteq \beta(y, y')$ (right transitivity)

then $(X \uplus Y, \varepsilon \uplus \zeta, \alpha \uplus \beta \uplus \gamma)$ is a skeletal symmetric $\mathcal{A}_{\mathbb{N}}$–category.

We now go on to examine the appropriate ways of comparing trees and glued trees.

4. Categories of Structures

In order to model all of the combinators of a typical process algebra, it is important to have sufficient universal structures to model various constructions on trees. Here we organise our forests into categories, and show that enough general structure does indeed exist to model many constructions of interest. For a more comprehensive discussion, see [10, 23].

The obvious notion of arrow between enriched categories is that of enriched functor [12].

Definition 20. An $\mathcal{A}_{\mathbb{N}}$–functor $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ for $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$ $\mathcal{A}_{\mathbb{N}}$–categories, is a function $F : X \to Y$ satisfying

(i) The extent of an object does not change under F: $\zeta(F(x)) = \varepsilon(x)$

(ii) F increases agreement $\alpha(x, y) \subseteq \beta(F(x), F(y))$

This immediately gives us the following categories:

Tree, with objects skeletal symmetric $\mathcal{A}_{\mathbb{N}}$–categories with prefix–closed agreement and arrows $\mathcal{A}_{\mathbb{N}}$–functors;

$\mathcal{A}_{\mathbb{N}}$–**Cat**, with objects $\mathcal{A}_{\mathbb{N}}$–categories and arrows $\mathcal{A}_{\mathbb{N}}$–functors;

These categories enjoy the following constructions:

Definition 21. The *sum* of two trees, $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$, written $\mathcal{X} + \mathcal{Y}$, is defined as the tree (Z, η, θ) where

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

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

(iii) $\theta((0,x),(0,x')) = \alpha(x,x')$
$\theta((0,x),(1,y)) = \theta((1,y),(0,x)) = \{0,\epsilon\}$
$\theta((1,y),(1,y')) = \beta(x,x')$.

Thus $\mathcal{X} + \mathcal{Y}$ joins $\mathcal{X}$ and $\mathcal{Y}$ together at the root.

Proposition 22. **Tree** has finite coproducts given by $+$.

Definition 23. Given two $\mathcal{A}_{\mathbb{N}}$–categories, $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$, $\mathcal{X}$ *times* $\mathcal{Y}$, written $\mathcal{X} \otimes \mathcal{Y}$, is defined as the tree (Z, η, θ) where

(i) $Z = X \uplus Y$

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

(iii) $\theta((0,x),(0,x')) = \alpha(x,x')$
$\theta((0,x),(1,y)) = \theta((1,y),(0,x)) = \emptyset$
$\theta((1,y),(1,y')) = \beta(x,x')$

Thus $\mathcal{X} \otimes \mathcal{Y}$ places $\mathcal{X}$ and $\mathcal{Y}$ side-by-side, giving no glueing between them. (The notation is chosen because of the similarity of this construction with the $\otimes$ of linear logic [6, 15].)

If n is a natural number, we write $n\mathcal{X}$ for the n–fold coproduct $\mathcal{X} \otimes \mathcal{X} \otimes \ldots \otimes \mathcal{X}$. The following is then routine.

Proposition 24. $\mathcal{A}_{\mathbb{N}}$–**Cat** has finite coproducts given by $\otimes$. Therefore, $n\mathcal{X}$ is given by the composition of the n-fold diagonal functor, [13] Δ_n with its left adjoint, $\otimes^n$.

The nature of coproducts in $\mathcal{A}_{\mathbb{N}}$–**Cat** gives us another subcategory, **Forest**, whose objects are finite coproducts $\bigotimes_i \mathcal{X}_i$ of trees (i.e. collections of trees with no glueing between them), and whose arrows are $\mathcal{A}_{\mathbb{N}}$–functors. We then have immediately

Proposition 25. The following diagram, where $\subset$ denotes a full subcategory, is correct:

$$\textbf{Tree} \quad \subset \quad \textbf{Forest} \quad \subset \quad \mathcal{A}_{\mathbb{N}}\text{–}\textbf{Cat}$$

---

* The only case where skeletality is important is if $\mathcal{X} = \mathcal{Y} = \mathbf{Nil}$, where **Nil** is the tree $(\{\bullet\}, \varepsilon(\bullet) = \{(0,\epsilon)\}, \alpha(\bullet,\bullet) = \{(0,0,\epsilon)\}$. Here we must set $\mathbf{Nil} + \mathbf{Nil} = \mathbf{Nil}$.

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 $F : \mathcal{C} \to \mathcal{C}'$ there exists a finite coproduct preserving functor $F' : \mathcal{D} \to \mathcal{C}'$ 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 $Fam(\mathcal{C})$, 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 *intersection* of two $\mathcal{A}_{\mathbb{N}}$–categories, $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$, written $\mathcal{X} \times \mathcal{Y}$, is defined as the $\mathcal{A}_{\mathbb{N}}$–category (Z, η, θ) where

(i) $Z = \{(x, y) \mid \varepsilon(x) = \eta(y)\}$

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

(iii) $\theta((x, y), (x', y')) = \min(\alpha(x, x'), \beta(y, y'))$.

Thus $\mathcal{X} \times \mathcal{Y}$ the largest subtree common to $\mathcal{X}$ and $\mathcal{Y}$.

Proposition 30. $\mathcal{A}_{\mathbb{N}}$–**Cat** has finite products given by $\times$. The empty $\mathcal{A}_{\mathbb{N}}$–cat, $0 = (\emptyset, \emptyset, \emptyset)$, is an initial object (but not a tree).

Proposition 31. $\mathcal{A}_{\mathbb{N}}$–**Cat** is finitely complete.

[†] I.e. a functor F : **Forest** $\to$ **Tree** such that $F(\mathcal{X}) \cong \mathcal{X}$ for a tree $\mathcal{X}$.

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 $\mathcal{X} \nrightarrow \mathcal{Y}$, for $\mathcal{X}$ and $\mathcal{Y}$ trees, and whose arrows $(\mathcal{X} \nrightarrow \mathcal{Y}) \longrightarrow (\mathcal{X}' \nrightarrow \mathcal{Y}')$ are pairs (F, G) where $F : \mathcal{X} \to \mathcal{X}'$, $G : \mathcal{Y} \to \mathcal{Y}'$ are $\mathcal{A}\mathbb{N}$–functors satisfying

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

will be written $\mathbf{Bim}(\mathbf{Tree})$.[‡]

This definition makes sense when it is realised that since $\mathcal{X} \nrightarrow \mathcal{Y}$ 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 $(X \uplus Y, \varepsilon \uplus \zeta, \alpha \uplus \beta \uplus \gamma)$ is an $\mathcal{A}\mathbb{N}$–category, and similarly for γ', then $F \uplus G$ is an $\mathcal{A}\mathbb{N}$–functor, so our requirement $(*)$ that the 'square'

$$\begin{array}{ccc} \mathcal{X} & \nrightarrow & \mathcal{Y} \\ F \downarrow & & \downarrow G \\ \mathcal{X}' & \nrightarrow & \mathcal{Y}' \end{array}$$

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 $\mathbf{Bim}(\mathbf{Tree})$ 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 $\gamma : \mathcal{X} \nrightarrow \mathcal{Y}$, and $\delta : \mathcal{Y} \nrightarrow \mathcal{Z}$, 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 $A^{\mathbb{NN}}$.

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 trees. This construction involves duplicating trees, partly to make the definition of the semantics of parallel composition smoother, and partly to avoid damaging nontransitivity of agreement, such as that demonstrated in example 15; we take a copy of $\mathcal{X}$ for each run in $\mathcal{Y}$ and vice versa, and then define a glueing between these forests.

Definition 39. The CSP glueing $\kappa : |Y|\mathcal{X} \nrightarrow |X|\mathcal{Y}$ between two trees $\mathcal{X} = (X, \varepsilon, \alpha)$ and $\mathcal{Y} = (Y, \zeta, \beta)$ is defined as the collection of maps

$$\{\gamma_{y,x}\} : y\mathcal{X} \nrightarrow x\mathcal{Y}$$

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 $\mathcal{Y}$, $\gamma_{y,x} : y\mathcal{X} \nrightarrow x\mathcal{Y}$ 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 $\mathcal{Y}$, $\kappa : |Y|\mathcal{X} \nrightarrow |X|\mathcal{Y}$ 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 the b on y_z is now no longer glued to things it will synchronise with *on different runs*. The duplication we have done allows us to present all possible process synchronisations in a consistent way.

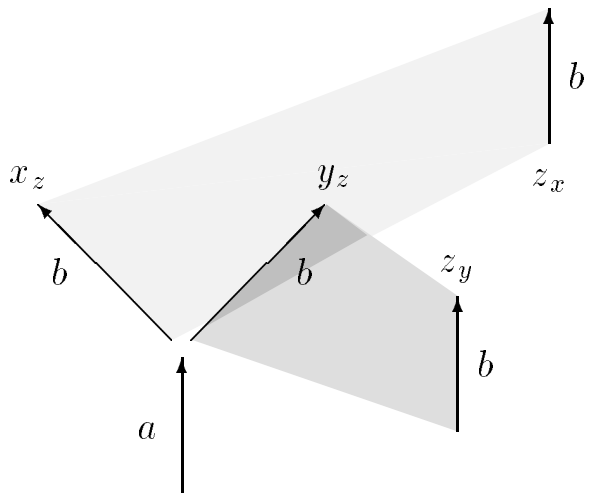


Figure 6: A problematic glueing revisited.

It is natural to organise the collection of CSP–glued trees into a category. We proceed just as for **Bim**(**Tree**):

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 $F : \mathcal{X} \to \mathcal{X}'$, $G : \mathcal{Y} \to \mathcal{Y}'$ 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

$$\mathbf{CSP}(\mathbf{Tree}^2) \underset{\emptyset}{\overset{K}{\underset{\longrightarrow}{\longleftarrow}}} \mathbf{Tree}^2$$

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. *Categories of Asynchronous Systems.* 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 Networks and ISDN Systems*, 14:25–59, 1987.

[4] A. Carboni and P. Johnstone. Connected limits, familial representability and artin glueing. In *Proceedings 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 Models for the Semantics of Parallelism*, volume 280. Springer-Verlag LNCS, 1987.

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

[8] M. Johnson. *Pasting Diagrams in n–categories with applications to coherence theorems and categories of paths.* PhD thesis, University of Sydney, 1988.

[9] S. Kasangian and A. Labella. Enriched categorical semantics for distributed calculi. *Journal of Pure and Applied Algebra*, 83:295–321, 1992.

[10] S. Kasangian, A. Labella, and D. Murphy. Process synchronisation as glueing. Technical Report to appear, Dipartimento dell'Scienze Informazione, Università di Roma, 1993. To be submitted for publication.

[11] S. Kasangian, A. Labella, and A. Pettorossi. Enriched categories for local and interaction calculi. In *Category theory and Computer Science*, volume 283. Springer-Verlag LNCS, 1987.

[12] G. Kelly. *Basic Concepts of Enriched Category Theory.* Cambridge University Press, 1982.

[13] S. Mac Lane. *Categories for the Working Mathematician.* Springer-Verlag Graduate Texts in Mathematics, 1971.

[14] F. Lawvere. Metric spaces, generalised logic and closed categories. *Rendiconti Seminario Matematico e Fisico Milano*, pages 135–156, 1974.

[15] J. Meseguer and U. Montanari. Petri nets are monoids: A new algebraic foundation for net theory. *Information and Computation*, 88(2):105–155, 1990.

[16] R. Milner. *Communication and concurrency.* International series on computer science, Prentice Hall International, 1989.

[17] M. Neilsen, V. Sassone, and G. Winskel. A classification of models for concurrency. In *the Proceedings of Concur*, volume 715. Springer-Verlag LNCS, 1993.

[18] R. De Nicola and A. Labella. A functorial assessment of bisimulations. Technical Report SI–92/06, Dipartimento di Scienze dell'Informazione, Università Roma La Sapienza, 1993.

[19] E-R. Olderog. *Nets, terms and formulas.* Cambridge Tracts in Theoretical Computer Science, Volume 23, 1991.

[20] A. Rensink. *Models and Methods for Action Refinement.* PhD thesis, Faculteit der Informatica, Universiteit Twente, 1993.

[21] R. Street. Enriched categories and cohomology. *Quaestiones Math.*, 6:265–283, 1993.

[22] R. Walters. Sheaves and cauchy-complete categories. *Cahiers de topologie et geometrie diff.*, 22:283–286, 1981.

[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.