

Vaughan Pratt@rAdvisor José Meseguer  
SRI International John C. Mitchell JanuaryFebruaryMarchAprilMayJune  
JulyAugustSeptemberOctoberNovemberDecember January 1991 1991

A DISSERTATION  
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE  
AND THE COMMITTEE ON GRADUATE STUDIES  
OF STANFORD UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

By

JanuaryFebruaryMarchAprilMayJune JulyAugustSeptemberOctoberNovemberDecember

©

Copyright by  
All Rights Reserved

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality,  
as a dissertation for the degree of Doctor of Philosophy.

to 4inwidth 3in height 0.4pt  
(Principal Adviser@r)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality,  
as a dissertation for the degree of Doctor of Philosophy.

to 4inwidth 3in height 0.4pt

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality,  
as a dissertation for the degree of Doctor of Philosophy.

to 4inwidth 3in height 0.4pt

Approved for the University Committee on Graduate Studies:

to 4inwidth 3in height 0.4pt Dean of Graduate Studies

Chapter

## Preface

In order to make the presentation more self-contained, this dissertation contains some material that is well-developed in the mathematical literature but unfamiliar to computer scientists. Sections ?, ? , ? and ? are such introductory material, as is the definition of  $F!$  in ?.

This dissertation also contains some results that have been previously published in jointly-authored work [CCMP89]. Except as otherwise noted in the text the results presented here are my own work, under the guidance of my advisor, Professor Vaughan Pratt.

## Chapter

## Acknowledgements

I would like to thank my advisor, Vaughan Pratt. His ideas, advice and critical reading have contributed a great deal to this work. Roger Crew and José Meseguer, as collaborators, have

influenced it in many ways. John Mitchell provided useful comments on rather short notice. Conversations with Bill Lawvere, John Power, Bob Walters and Mike Johnson have been very helpful.

I am very grateful to my wife, Mei Lin, for her support, encouragement and patience throughout the years I have been working at this dissertation.

This research was supported in part by the National Science Foundation under grant CCRm8814921.

## **Contents**

## **List of Tables**

## **List of Figures**

## Abstract

In models of concurrent processes constraints on the order of events are often represented by partial orders, and schedules of events are then defined using an algebra of standard operations such as sequential and parallel composition.

In this dissertation the notion of partial order is replaced by that of a set with a metric which takes values in a given ordered monoid. Partial orders are the simple case of a monoid whose two elements represent the presence or absence of a constraint.

An ordered monoid can be seen as a monoidal category, and schedules based on it are categories enriched in the monoid. Algebraic operations on schedules can then be defined as constructions in the category of schedules. These definitions rely on certain properties of a category of schedules, such as closure and completeness. To simplify proofs of these properties, two constructions are defined. The first creates a category of unlabeled schedules from a system of constraints. The second adds labels to unlabeled schedules. Many categories of interest can be constructed from simple categories using these two methods. The main results of the dissertation derive the required properties of categories so constructed from similar, more easily verified properties of the base categories.

Several notions of timing constraint can be viewed in a uniform way in this framework. An example is the Gaifman-Pratt system, essentially the partial order model with additional specification as to whether two events may occur simultaneously. It corresponds to a monoid whose three elements represent strict precedence, lax precedence (simultaneity is permitted), and absence of constraint. Real-valued timing constraints correspond to the additive monoid of the real numbers.

## Chapter 1

# Algebras of Labeled Measured Sets

In this chapter measured sets and operations for combining them are described informally, using the partial order model of concurrency as motivation.

## 1.1 Outline

In the pomset model of concurrent processes [Gra81, Gis84, Pra86], a process is specified by a set of *schedules* of events. Schedules are formally defined as *pomsets* (partially ordered multisets), consisting of a poset with labels on each element. Each label specifies a particular action. The partial order relation constrains the order in which events occur.

As an example, consider the pomset represented by the following Hasse diagram, in which arrows run from smaller to larger elements in the partial order and each element is represented by an occurrence of its label. It denotes a poset of five elements each of which is labeled with an element chosen from the set  $a, b, c, d$ . (Redundant arrows, such as an arrow from  $a$  to  $d$ , are usually omitted from such diagrams. In other words, the diagram is used as an abbreviation for its transitive closure.)

When this pomset is interpreted as a schedule, the four labels represent four *actions* and the five elements represent five *events*, each of which is a single occurrence of one of the actions. The schedule specifies that one occurrence of action  $a$  must happen before any other action. Then



actions  $b$  and  $c$  may happen. Action  $d$  may happen after  $c$  has happened, and a second occurrence of action  $a$  occurs after both  $b$  and  $c$ .

The interpretation as schedules suggests a number of ways to combine several pomsets to create another. Examples are concurrence (running two schedules in parallel) and concatenation (all events of one schedule are constrained to happen before all events of the other). Formal definition of these operations leads to an algebra of pomsets.

An important goal of this thesis is to extend pomsets and their algebra to more complex schedules, such as schedules in which one may specify allowable delays between events, rather than merely the order in which events occur. For example, delay specifications might be added to the previous diagram as follows:

One interpretation, though not the only possibility, is that this diagram specifies that  $b$  follows  $a$  after a delay of at least 1 second and  $c$  follows  $a$  after at least 4 seconds, and so forth.

It would be natural to use a structure similar to metric spaces for such specifications. However, the usual definition of metric space cannot be adopted wholesale. For example, the symmetry axiom  $d(a,b)=d(b,a)$  must clearly be abandoned because the delay from  $a$  to  $b$  will hardly ever be equal to the delay from  $b$  to  $a$ . Furthermore, it would be very convenient to allow distances to be other than real numbers because many different structures can then be treated within the same overall framework. For example, partially ordered sets can be seen as  $k$ metric spaces in which the distance function is two-valued:  $d(a,b)=1$  corresponds to  $a*b$  and  $d(a,b)=0$  to  $a$  and  $b$  unrelated.

Using this generalization other models of processes can also be treated as measured sets. For example, Gaifman and Pratt [GP87] allow for more complex relationships between events in that event  $a$  might precede event  $b$  either *weakly*, meaning that  $a$  and  $b$  may actually be simultaneous, or *strongly*, meaning that  $a$  must occur strictly before  $b$ . Note that if  $a$  weakly precedes  $b$ , which strongly precedes  $c$ , then  $a$  must strongly precede  $c$ . Thus a weak constraint composed with a strong constraint (in either order) yields a strong constraint. Gaifman later proposed another variant where the relation  $ka$  *causes*  $bl$  is distinct from  $ka$  happens to precede  $b$  but does not cause it [Gai88]. In this second model a strong constraint composed with a weak one yields a *weak* constraint. Both these models can be seen as measured set models in which the distance function is 3-valued. The difference between the two corresponds to different algebraic structures on the space of distances.

Moreover, the algebraic operations applicable to partially ordered sets of events have analogues in the more complex cases.

Such generalized metric spaces will be called *measured classes* (*measured sets* when the class is actually a set). A schedule is treated as a *labeled measured set*, which is simply a measured set, a set of labels called the *alphabet* (thought of as the set of possible actions) and an assignment of one label to each point of the measured set. A measured class whose distance function takes values in a domain  $D$  will be called a *class (or set) measured in  $D$* .

In addition I define an algebra of labeled or unlabeled measured sets. Its operations are defined uniformly over a wide class of possible measurements and model concurrent composition of schedules, concatenation etcetera.

Formal definitions will be given in chapter ?; in the current chapter this generalization of metric space will be presented informally.

## 1.2 Algebra of labeled measured sets

This section is an informal description of the operations which are useful in an algebra of labeled measured sets. They will be called the *normal* operations. Each operation is illustrated using certain labeled measured sets whose measurements are taken from the positive reals together with  $?*$  and  $*$ . Such structures can be considered as schedules by taking  $d(a,b)$  to be the minimum delay from  $a$  to  $b$ . If  $b$  need not follow  $a$  at all, define  $d(a,b)=?*$ . The subclass of such structures in which measurements are restricted to the two values  $*$  and  $?*$  can be identified with the class of pomsets by considering  $a*b$  to be the same as  $d(a,b)=*$ . Each of the examples given below can be converted to a pomset example by replacing each positive measurement by  $*$ .

Figure 1.1: Two labeled measured sets

*Concurrence.* Denoted  $P|Q$ .  $P$  and  $Q$  are juxtaposed with no constraints (that is,  $?*$  constraints) between an element of either of the sets and an element of the other. This corresponds to running two schedules independently in parallel. The alphabet of  $P|Q$  is the disjoint union of the alphabet of  $P$  and the alphabet of  $Q$ , so that the two occurrences of action  $a$  are distinguished in the example of  $P|Q$  in the figure. For this reason concurrence is sometimes called *disjoint concurrence*. Taking the ordinary union instead of disjoint union gives rise to *union concurrence*.

*Concatenation.* Denoted  $P \cdot Q$ . The elements of  $P \cdot Q$  are also the disjoint union of the elements of  $P$  and  $Q$ , but to the constraints of  $P|Q$  are added a constraint of at least  $x$  from each event of  $P$  to each event of  $Q$ . Note that  $P \cdot Q$  is identical to concurrence, and that  $PQ$  is the usual concatenation of pomsets. Disjoint and union concatenation are defined similarly to disjoint and union concurrence. In the figure the distance from  $a$  to  $b$  is 12.

Figure 1.2: Concurrence and concatenation

*Orthocurrence.* Denoted  $P*Q$ . Each element is an ordered pair consisting of one element from  $P$  and one from  $Q$ . Measurements are defined by  $d(a,b \succ a',b' \succ) = d(a,a') + d(b,b')$ , with  $*+*$  defined to be  $?*$ . A typical use of this operation is when  $P$  specifies messages arriving at a channel and  $Q$  is the schedule of transmission to stations along the channel. If there is a 1-second delay between the arrival of message 1 at the input and the arrival of message 2, and the channel has at least .5 seconds transmission time, then there will be a minimum delay of 1.5 seconds between arrival of message 1 and departure of message 2.

*Pairing.* Denoted  $P \bowtie Q$ . Each element is an ordered pair, as in orthocurrence, but the measurement from  $a,b \succ$  to  $a',b' \succ$  is the *minimum* of the measurement from  $a$  to  $a'$  and the measurement from  $b$  to  $b'$ . Note that pairing is the same as orthocurrence when dealing with pomsets.

Figure 1.3: Orthocurrence and product

*Standard Constants.* The empty measured set, denoted  $\mathbf{0}$ , is the identity for concurrence and concatenation. Two different measured singletons are of interest. One is denoted  $\mathbf{I}$ , and the distance from the single element  $*$  to itself is 0. The other is denoted  $\mathbf{1}$  and defined by  $d(*,*) = *$ .  $\mathbf{I}$  is the identity for orthocurrence and  $\mathbf{1}$  is the identity for pairing. If the only distances are  $*$  and  $?*$  then  $*$  is an identity for addition. Thus, for pomsets,  $\mathbf{I} = \mathbf{1}$ .

*Exponentiation.* Denoted  $[Q,P]$ . The elements of this measured set are the distance-preserving functions from  $Q$  to  $P$ , where  $f$  preserves distances if  $d(fa,fb) \leq d(a,b)$  for all  $a$  and  $b$  in  $Q$ . (Observe that such functions between posets are precisely the monotone functions.) Define the distance from  $f$  to  $g$  to be the infimum of the distances from  $fa$  to  $ga$  taken over the elements of  $Q$ . When  $Q$  and  $P$  are schedules  $[Q,P]$  can be interpreted as being derived from  $P$  by redefining an event to be a *pattern* of events defined by  $Q$ . In the running example there is only one distance-preserving function from  $Q$  to  $P$ , the one taking the  $a$  event to  $c$  and the  $d$  event to  $a$ . Thus  $[Q,P] = \mathbf{I}$ . In the corresponding pomset example, however,  $[Q,P]$  has five elements. In the figure the label  $ab$  indicates the function  $d*a$  and  $a*b$ .

Figure 1.4: Exponentiation  $[Q, P]$ , pomset case only

The next set of operations rely on a concept of the *location* where an action occurs. This arises, for example, in a multiprocessor system, where each action of the system is actually performed by one of its components. Think of the components as being physically separated, so that each possible action of the whole system can be associated with the location where it could occur. A formal model of this situation is to specify a set of locations and a map to it from the alphabet of actions. Two actions that are assigned the same location are called *colocated*, and two events are said to be colocated if their actions are colocated. For the purpose of the examples, we will assume that only  $b$  and  $d$  are colocated (and, of course, that each action is colocated with itself).

*Local concatenation*, denoted  $\cdot$ , differs from concatenation in that additional constraints are introduced only between colocated elements rather than between all of them.

Figure 1.5: Local concatenation: " $\cdot$ "

*Local pairing*, denoted  $P$  is that subset of the pairing containing only pairs of colocated events. For the running example  $P$  has two elements labeled  $bd$  and  $aa$  and no constraints.

*Fusion* is a method of glueing measured sets together along a common subset. The data for fusion are two or more distance-preserving functions with the same domain, say  $f: P \rightarrow Q$  and  $g: P \rightarrow R$ . The result is a measured set with the elements of  $Q$  and  $R$  except that the element  $fa$  is identified with  $ga$  for each  $a$  in  $P$ . In the following example think of  $Q$  as the transmission of 3 messages ( $a$ ,  $b$  and a second  $a$ ) from location 1 to location 2.  $R$  is the transmission of 2  $a$  messages from 2 to 3.  $P$  is the forwarding of the  $a$  messages at location 2. The result is the processing of all messages through the system. Distances are not particularly important here and have been omitted from the figure.

Figure 1.6: Fusion of  $Q$  and  $R$  at  $a_2$  events

### 1.3 Other interpretations of measured sets

Many control structures and data types other than schedules can be viewed as measured classes or labeled measured classes. An automaton is one example in the states are the elements of the class and the transitions are the measurements between them. (Start and finish states are not considered here.)

In fact, even when attention is confined to the simple types of measurement considered in the previous section there are two possible views of a measured set. In that section I frequently took the *conjunctive* view, where a set is a schedule. Its elements are events (*all* of which must happen, hence *kconjunctive*), and the measurements require execution of the events to be followed by executions of related events. In the *disjunctive* view, the measured set is like an automaton: its elements are states, and the measurements allow passage from one state to others.

Furthermore, many other data types can be construed as classes of labeled measured sets. The class of sets is the class of measured sets when there is only one possible value for a measurement. A category is a measured class in which the measurements are sets, that is  $d(a, b)$  is the set of arrows from  $a$  to  $b$ . An order enriched category (see Wand [Wan79]) is a measured class where the measurements are partially ordered sets rather than unstructured sets. As noted above a metric space is a class where measurements are positive real numbers. Labeling adds some more possibilities. One example is multisets, which can be seen as labeled sets, although a single multiset corresponds to many isomorphic labeled sets.

Consider how some of the operations defined with concurrency in mind may be interpreted on data types.

*Concurrence* is disjoint union of data types. For automata concurrence corresponds to nondeterministic choice between two automata. (This view works best if automata have a *set* of start states and the start states of the result defined to be the disjoint union of the original sets of start states.)

On data types *orthocurrence* and *pairing* are two product operations. For automata both describe how two automata can be run in parallel, with each state of the combined automaton representing a state of each of the components. Either orthocurrence or pairing may be appropriate depending on the intended interpretation. If the measurement between state  $a$  and state  $b$  describes the resources required to move from  $a$  to  $b$ , then orthocurrence is the better description of their parallel composition. If, on the other hand, the measurement specifies a maximum permissible transition time between states then pairing would be a more useful operation.

*Concatenation* on data types is similar to disjoint union. for automata  $P;Q$  represents an automaton which may begin behaving like  $P$  but escape to behaving like  $Q$  at any time. Local concatenation is similar but restricts the states which can escape to  $Q$ .

*Exponentiation* is easily interpreted for data types: it creates function spaces. It does not appear to have a straightforward interpretation for automata.

## 1.4 History and related work

The use of partial orders as a model for concurrency arose in several different ways.

One thread originates with Kahn's model of communicating processes [Kah74], which used histories of each channel in a network to record its state. Attempts to extend this model to nondeterministic processes uncovered an anomaly [BA81] and Brock and Ackerman proposed adding precedence information between histories to resolve it. Pratt [Pra82, Pra85, Pra86] gradually abandoned the association of each event with a channel, leaving only a partial order of events. He also emphasized the algebra of partial orders, and this algebra was investigated by Gischer [Gis84], although similar notions were proposed by Birkhoff as early as 1937 [Bir37, Bir42]. The extensions by Gaifman and Pratt [GP87], and Gaifman [Gai88] are of particular interest since they arise very naturally as measured sets.

A second thread comes from the theory of Petri nets (see, for example, Reisigl's introduction to the subject [Rei85]) where partial orders have often been advocated for describing how one event enables another. Grabowski's use of labeled partial orders [Gra81] is part of this thread, as is Winskel's event structures model [Win80, NPW81].

Other models based on partial orders are Degano and Montanari's Concurrent Histories [DM85b, DM87] and actor systems [Agh86].

Of course, several of the best-known approaches to concurrency treat the concurrent execution of two actions as their sequential execution in an unspecified order. These models therefore deal with strings of events rather than partially ordered sets. Examples are Milner's CCS [Mil80, Mil85b] and CSP, due to Brooks, Hoare and Roscoe [Bro83, BHR, Hoa85], and the temporal logic system of Manna and Pnueli [MP83]. Two ways to adapt the models presented here to such total order models are possible. The simpler of the two would just restrict attention to the total orders in the categories of partially-ordered sets and its analogues. More interesting though, are sets measured in the real numbers (or the integers) with the trivial ordering (see section ? below). Such measured sets can be seen as linearly ordered, possibly with gaps in the order. However, this solves only part of the problem, for the system presented here does not provide a means to represent the internal decisions of a process by which it may refuse to perform an action. Perhaps one solution would be to introduce internal actions to represent decisions, a model similar to that discussed by Montanari and Simonelli [MS80].

The essential mathematics of this thesis is the theory of monoidal and enriched categories, used in a way suggested by Lawvere [Law73]. Recently several researchers have applied some of the same theory, but it must be stressed that they use it in a rather different way.

Main and Benson [Ben87a, Ben87b, Ben87c, MB84], for example, use free modules over

semirings to model the inputs and outputs of a nondeterministic computation. Tensor products (hence monoidal categories) arise naturally here when dealing with multiple inputs or outputs. Similarly, Degano, Meseguer and Montanari treat multisets as free modules over the semiring of the natural numbers to give an algebraic treatment of Petri nets [MM88b, DMM90]. In this case arrows of the category represent transitions, and tensor product corresponds to running a number of transitions concurrently. (In the system presented in this thesis the concurrence operation turns out to be categorical sum, not tensor product.) Meseguer and Martí-Oliet [MM89] show that Girard's linear logic [Gir87], Petri nets and monoidal categories are all related, but while this work certainly links concurrency and monoidal categories in a very interesting way, no deep connection between it and this thesis is known.

## 1.5 Summary of following chapters

In Chapter 2, I define the MCE-categories to be those that are symmetric monoidal, closed, complete and cocomplete. Such categories possess those properties needed to define all the normal operations. I define the normal operations in this setting and show that the category of pomsets is one such category with operations given by the general definition. A wider class of categories, called the MC-categories, lacking only the closure property, is also of interest.

But proving any given category to be MC or MCE may be somewhat tedious, and many of the categories of interest can be constructed from simpler categories by combinations of two constructions called *enrichment* and *labeling*. In chapters 3 and 4 a number of results are proved to show the existence of (co)limits in the constructed categories when given related properties of the simpler categories from which they were constructed. Similar results show closure of the constructed categories under appropriate conditions.

The result can be used as an algebra of schedules in which the notion of time has become a parameter rather than an unchangeable fundamental. One can choose whatever notion of time is appropriate to a particular application and, when expressed as a suitable monoidal category, this choice generates a corresponding category of schedules. Operations for combining schedules are defined uniformly and independently of the choice of time, and are automatically interpreted in a way suitable for that choice.

As regards prerequisites, I assume that the reader is familiar with the notions of functor, natural transformation, limit, colimit, adjunction and comma category at the level to be found in any introduction to category theory, such as Mac Lane's [Mac71]. The theory of 2-categories is also used at some points. Mac Lane treats these briefly, more information can be found in a paper by Kelly and Street [KS74].

As regards notation: function and functor application are usually denoted without parentheses (that is,  $FX$  rather than  $F(X)$ ). The composite arrow is denoted  $gf$  (not  $g*f$  or  $fg$ ). Subscripts denoting the components of natural transformations are omitted whenever they can be derived from the context. So if  $*:F \rightarrow G$  is a natural transformation, the component will often be written simply as  $*:FX \rightarrow GX$ . Identity arrows are denoted by  $1$ .

5

@page" @twocolumn"

## Chapter 2

### MC Categories

In this chapter I define the normal operations formally, using a framework provided by the theory of monoidal categories. I define the MC-categories, in which the normal operations other than exponentiation are well-defined, and the MCE-categories, in which all the normal operations are defined. The MC-categories include the categories of measured classes (along with other possibilities). I show that the category of pomsets is MC and that the abstract definitions yield the expected operations (sometimes with minor variants) in this case.

#### 2.1 Monoidal categories

Roughly speaking, a monoidal category is a category equipped with a multiplication (that is, a functor of two arguments) called *tensor product* and denoted by  $*$ , and an identity element denoted by  $I$ . Such categories play two roles in the theory described here. Firstly, they serve as the domains of distance functions of measured sets. In this case each object of the monoidal category represents a distance and tensor product represents addition of distances. The arrows of the category represent certain relations between distances that arise when distances are seen as constraints. For simple cases, in which the category is a partial order,  $A*B$  means that  $A$  is a weaker constraint than  $B$  in the sense that two events satisfying constraint  $B$  will also satisfy  $A$ . For example, if constraints are lower bounds on distances then two events separated by at least 5 units will certainly be separated by at least 3 units, so  $3*5$ .

The second role of monoidal categories arises because the class of sets measured in a given domain forms a monoidal category, in a manner to be described below. It is convenient to define the normal operations purely in terms of this monoidal category structure without specific reference to the fact that the objects are measured sets. In this case the tensor product is the orthocurrence operation described in section 2.

The two roles are related because given a space of measurements viewed as monoidal category, a standard construction yields its category of measured sets.

The following definition of monoidal category is standard (Mac Lane [Mac71] and Arbib and Manes [AM75b] treat them briefly; detailed treatments are by Eilenberg and Kelly [EK66a] and Kelly [Kel82]). The usual associativity and identity axioms of a monoid are weakened by requiring only that  $(A*B)*C$  be *isomorphic* to  $A*(B*C)$  rather than that the two expressions be equal. Similar isomorphisms replace the standard left and right identity laws. In order to lift results from the identity case to the isomorphism case, though, it is necessary that these isomorphism be natural, and that compositions of them satisfy certain conditions called *kcoherence*.<sup>1</sup> (Note that if  $*$ ,  $*$  and  $*$  are all identity maps then naturality and coherence hold trivially.) If we insisted on identities rather than isomorphisms many important examples would fail to satisfy the definition, including the categories of measured sets mentioned in the introduction to this chapter.

**Definition .**

A *monoidal category* consists of

1. A category ;
2. A functor (usually written in infix notation);
3. An object  $I$  of  $\mathbf{D}$ ;
4. An isomorphism  $*$ :  $(A*B)*C \rightarrow A*(B*C)$  natural in  $A$ ,  $B$  and  $C$ ;
5. An isomorphism  $*$ :  $I*A \rightarrow A$  natural in  $A$ ; and

6. An isomorphism  $\alpha: A \otimes I \otimes A \rightarrow A$  natural in  $A$ .

These data must satisfy certain *coherence conditions*, namely the commutativity of the following two diagrams.

height6pt width4pt depth0pt

The coherence laws imply that any two parallel arrows constructed solely by composition,  $\otimes$  and taking inverses from  $\otimes$ ,  $\otimes^{-1}$ ,  $\otimes$  and identity arrows must be equal [Mac71].

The next definition is for a *symmetric* monoidal category, which would be a commutative monoid in the category of categories except that, once again, the monoid laws need hold only up to the specified natural isomorphisms. In the interpretation of a monoidal category as a space of measurements symmetry specifies that addition of distances must be commutative. Almost all the monoidal categories we encounter will be symmetric.

#### Definition .

A *symmetric monoidal category* consists of a monoidal category,  $(\mathcal{C}, \otimes, I)$ , together with an isomorphism  $\alpha: A \otimes B \rightarrow B \otimes A$  natural in both  $A$  and  $B$  and satisfying the following laws:

1.  $\alpha$  is an involution:
2.  $\alpha \circ \alpha = \text{id}$ :
3.  $\alpha$  and  $\otimes$  harmonize:

height6pt width4pt depth0pt

## 2.2 Enriched categories

Given a monoidal category  $\mathbf{D}$  we may define categories *enriched in  $\mathbf{D}$* , or simply  *$\mathbf{D}$ -categories*. They consist, in part, of a class of objects together with an assignment of an object of  $\mathbf{D}$  to each ordered pair of objects. When  $\mathbf{D}$  is interpreted as a space of measurements, a  $\mathbf{D}$ -category can be seen as a measured class: the objects of the  $\mathbf{D}$ -category are the elements of the measured class and the object of  $\mathbf{D}$  assigned to a pair is the distance between them. Further conditions of the definition correspond to a triangle inequality on measurements and a requirement that the distance from an element to itself is not arbitrary.

In addition to the preceding motivation, it should also be observed that **Set**, the category of sets, can be considered a monoidal category by defining  $\otimes$  to be cartesian product and  $I$  to be a singleton set. For the case  $\mathbf{D} = \mathbf{Set}$  the following definition is just an alternative definition of a category  $\mathbf{A}$  in which  $\mathbf{A}(A, B)$  is the set of arrows from  $A$  to  $B$ ,  $\alpha: \mathbf{A}(A, B) \otimes \mathbf{A}(B, C) \rightarrow \mathbf{A}(A, C)$  is the composition map  $\alpha(f, g) = gf$  and  $\alpha: 1 \otimes \mathbf{A}(A, A) \rightarrow \mathbf{A}(A, A)$  is the constant map to the identity arrow on  $A$ . The commutativity requirements are just the statements that composition is associative and composition with identity arrows is an identity.

#### Definition .

Let  $(\mathbf{D}, \otimes, I)$  be a monoidal category. A *category enriched in  $\mathbf{D}$*  is defined by:

1. A class  $|\mathbf{A}|$  called the *objects* of  $\mathbf{A}$ ; and
2. For each pair of objects  $A$  and  $B$  in  $|\mathbf{A}|$  an object  $\mathbf{A}(A, B)$ ; and



3. For each triple of objects  $A, B$  and  $C$  in  $|\mathbf{A}|$ , a morphism of  $\mathbf{D}$

4. For each object  $A$  of  $\mathbf{A}$  a morphism of  $\mathbf{D}$ .

These data must satisfy the following conditions:

1.  $*$  is associative:
2.  $*$  is a left identity
3.  $*$  is a right identity

If the class  $|\mathbf{A}|$  is a set then  $\mathbf{A}$  is called a *small*  $\mathbf{D}$ -category.

Finally, I define an important property that many of the examples of monoidal categories possess.

#### Definition .

A monoidal category  $\mathbf{D}$  is *closed* if for each object  $A$  of  $\mathbf{D}$  the functor

has a right adjoint.

Note that if  $\mathbf{D}$  is symmetric (as will usually be the case in our examples) then closure is equivalent to the existence of a right adjoint for either  $?*A$  or  $A*?$ .

In this case there exists a unique functor , denoted by

such that for each  $A$ ,  $[A, ?]$  is right adjoint to  $A*?$  [Mac71, page 100]. This functor is called the *internal-hom* of the closed monoidal category  $\mathbf{D}$ . (An alternative name for this functor is *exponentiation* and an alternative notation for  $[A, B]$  is  $B^A$ . I use the second notation only to refer to ordinary functor categories.)

A closed  $\mathbf{D}$  may be considered as a category enriched in itself, by defining  $\mathbf{D}(A, B)$  to be  $[A, B]$ .  $*$  and  $*$  can then be defined in terms of the adjunction between  $@$  and  $[?, ?]$ .

## 2.3 Examples

Here are some important examples of monoidal categories and the categories enriched in them.

1. Any category having finite products can be given a symmetric monoidal structure by defining  $*$  taken to be categorical product,  $I$  to be a terminal object, and defining  $*$ ,  $*$ ,  $*$  and  $*$  using the universal property of product. This structure is called a *cartesian monoidal category*, and if it is closed then it is *cartesian closed*.

The category **Set** has already been mentioned. Another important example is **Cat** the category of small categories. A **Cat**-category is also known as a 2-category. Mac Lane [Mac71] gives an alternative definition of 2-category in terms of objects, arrows and 2-cells. Kelly and Street [KS74] state that the two definitions are equivalent. Both **Set** and **Cat** are closed, and hence cartesian closed.

Note that many such examples would be excluded if  $*$ ,  $*$ ,  $*$  and  $*$  were required to be identities. Cartesian product of sets, for example, is *not* associative.  $(A*B)*C$  is merely isomorphic to  $A*(B*C)$  in an obvious way, and the necessary properties of this isomorphism are captured in the coherence axioms.

2. The category of modules over a commutative ring  $R$  is monoidal with  $*$  being the usual tensor product of such modules,  $I$  being  $R$  considered as an  $R$ -module. In particular the category of small abelian groups  $\mathbf{Ab}$  is such a category. Take  $R$  to be  $\mathbf{Z}$ .  $\mathbf{Ab}$ -categories are sometimes called preadditive categories.  $\mathbf{Ab}$  is closed, but not cartesian.

Examples like **Set**, **Cat** and **Ab** are historically important to the development of enriched category theory, but in applications to computer science other examples are of more interest.

1. The ordinal  $\mathbf{2}$  considered as a cartesian closed category. This can be seen as the category of boolean truth values, with product being conjunction.  $\mathbf{2}$ -categories are preorders: the composition law reduces to the statement of transitivity and the identity law to that of reflexivity. In order to avoid confusion with 2-categories I will call such categories **Bool**-categories.
2. The nonnegative real numbers with the usual ordering and tensor product defined to be ordinary addition. This monoidal category is not closed, for if tensor were a left adjoint it would preserve all colimits. In particular, it would preserve the initial object 0, but in fact  $x+0 \neq 0$  in general. However the category can be made closed by adjoining  $?\ast$  as a new object with the property  $x+(?\ast)=?\ast$  for all  $x$ . An  $\ast$  object is then required as the value of  $[?\ast, 0]$ . It has the property  $x+\ast=\ast$  for all  $x$  except  $x=\ast$ . This category (including both  $\ast$  and  $?\ast$ ) will be denoted by  $\mathbf{R}$ .

The internal-hom of  $\mathbf{R}$  is

Categories enriched over  $\mathbf{R}$  can be seen as schedules of events with minimum delays specified between each pair of events. The composition law is consistent with this interpretation; it states that the minimum delay from  $A$  to  $B$  must be at least as great as the minimum delay from  $a$  to  $c$  plus the minimum delay from  $c$  to  $b$ , for any event  $c$ . The identity law says that  $d(a,a)=0$  for all  $a$  and then the composition law implies that if  $d(a,a) > 0$  then  $d(a,a)=\ast$ . Further application of the composition law shows that for such an  $a$  all distances  $d(a,b)$  and  $d(b,a)$  are infinite.

3. (due to W. Lawvere [Law73]) The nonnegative real numbers with the opposite of the usual order, and tensor product being addition is closed with internal-hom defined by truncated subtraction.

However it is not cocomplete. To make it cocomplete (while preserving closedness), include  $\ast$  as an initial object. This category is denoted  $\mathbf{R}^+$ .

$\mathbf{R}$ -categories are similar to metric spaces, with  $d(a,b)$  being the distance from  $a$  to  $b$ . The identity law reduces to the statement that  $d(a,a)=0$  and the composition law to the triangle inequality.  $\mathbf{R}$ -categories need not satisfy the law  $d(a,b)=0$  implies  $a=b$  nor the symmetry law.  $\mathbf{R}$ -categories can be seen as schedules of events with maximum delays between each pair of events.

Pratt [Pra89] has observed that this category has appeared in computer science in the guise of a *semiring* of distances, in which semiring addition is **min** and semiring multiplication is addition of reals. In categorical terms, addition is coproduct, multiplication is tensor product, and distributivity is ensured by closure. The ordinal  $\mathbf{2}$  has also been treated the same way. Treating these sets as semirings allowed Robert and Ferland [RF68] to treat Roy [Roy59] and Warshall's [War62] algorithm for transitive closure of a relation, and Floyd's [Flo62] shortest-path algorithm, as a single algorithm on matrices over different semirings. Pratt also notes that Kleene's algorithm for closing an automaton is another example, in which the underlying category contains languages over a given alphabet, ordered by inclusion and with (non-symmetric) product being concatenation.

In the current setting a matrix over  $\mathbf{D}$  is a  $\mathbf{kD}$ -graph, and Robert and Ferland's algorithm is precisely the construction of a free  $\mathbf{D}$ -category on the  $\mathbf{D}$ -graph.

4. The category of sets of nonnegative real numbers ordered by inclusion is a monoidal closed category under the tensor product

$$X * Y =_{x+y|x * X \text{ and } y * Y}.$$

denoted by **SR**. The internal-hom is given by

$$[X, Y] = z | z + X * Y.$$

Both  $\mathbf{D}$  and  $\mathbf{SR}$  are sub-monoidal-categories of **SR**,  $\mathbf{D}$  consists of those intervals of the form  $[0, x]$  for  $0 \leq x \leq 1$ ,  $*$  corresponds to the set of all nonnegative reals and  $?$  to the empty set.  $\mathbf{SR}$  consists of the intervals  $[x, 1]$ . One can also define other interesting subcategories of **SR**. For example we could consider only the closed intervals rather than arbitrary subsets. **SR**-categories can be seen as schedules of events where  $d(a, b)$  is a set of possible delays from  $a$  to  $b$ .

5. Each example involving the real numbers has a corresponding example where the nonnegative reals are replaced by the natural numbers.
6. Any monoid can be made into a monoidal category by considering it as a discrete category. For such a category to be closed it must be a group. An example is the set of real numbers  $\mathbf{R}$  with the discrete ordering and tensor product being addition. A category enriched in  $\mathbf{R}$  is like a schedule with exact delays specified between each event; the time of execution of any single event determines the time of every other event in the schedule.
7. A simple monoidal category arises from the Gaifman-Pratt *proset* model of concurrency [GP87] in which the set of events is equipped with two relations: a preorder  $*$  and an irreflexive partial order  $<$  with the condition that  $a < b$  implies  $a * b$ . The intention is that  $a * b$  be interpreted as  $ka$  happens no later than  $bl$  and  $a < b$  as  $ka$  happens strictly earlier than  $bl$ . This model can be fitted into our framework by taking  $\mathbf{D}$  to be the ordinal number 3, with commutative tensor product defined by  $0 * x = 0$ ;  $1 * x = x$  and  $2 * x = 2$ . This category is denoted  $\mathbf{3}$ . Then we can identify a proset with a category enriched over  $\mathbf{3}$ , by taking  $d(x, y) = 2$  to mean  $x < y$  and  $d(x, y) = 1$  to mean  $(x * y) * (x * y)$ . Here the composition law becomes the statement that  $a < b < c$  implies  $a < c$  and  $a < b * c$  or  $a * b < c$  or  $a * b * c$  all imply  $a * c$ . Gaifman has also introduced another definition of processes in which there are two orderings: for causal relationships and for accidental temporal relationships. Here  $*$  implies but not the converse, and  $<$  together imply. Schedules with such relationships can be seen as  $\mathbf{3}$ -categories, where now the ordinal  $\mathbf{3}$  is considered as a cartesian monoidal category. For this tensor product  $0 @ x = 0$   $2 @ x = x$  and  $1 @ 1 = 1$ . We can identify  $\mathbf{D}$  with  $d(a, b) = 2$  and  $d(a, b) = 1$  with  $*$ . The composition law for  $\mathbf{3}$ -categories now becomes precisely the statement that successive temporal constraints imply a temporal constraint (since  $1 @ 1 = 1$ ) and that a temporal constraint composed with a causal constraint imply a temporal constraint (since  $1 @ 2 = 1$ ).

Both  $\mathbf{3}$  and  $\mathbf{3}'$  are special cases of a general way of representing multiple-partial-order models of concurrency. These models correspond to nonzero finite ordinals considered as categories and equipped with a symmetric idempotent tensor product. The ordering on elements of the ordinal correspond to implications between the partial order relations. The tensor product defines how compositions of different partial orders are to be interpreted; symmetry is the requirement that the order of composition is not relevant to the result and idempotence is required to show that each element does, in fact, correspond to a partial order.

Monoidal categories with all the properties described in the previous paragraph have a rather interesting structure. Firstly, note that one can define a new partial ordering  $*$  on the elements by  $a * b$  if and only if  $a * b = a$ . Symmetry ensures that  $*$  is reflexive and  $a * b * c$  iff  $a * b = a$  and  $b * c = b$ , whence  $a * c = (a * b) * c = a * (b * c) = a * b = a$ . So  $*$  is transitive. It is obvious that  $*$  is antisymmetric. Further properties of  $*$  are given by the following.

**Proposition 5** (due to V. Pratt.)

1.  $*$  is a total order;
2. For any  $a$  and  $b$   $a * b$  is the minimum of  $a$  and  $b$  under  $*$ ;
3. When elements are enumerated in  $*$ -order they first ascend in the  $*$ -order until  $I$  is reached, then descend in the  $*$ -order;
4. The ordinal is a closed monoidal category if and only if  $0$  is minimal under  $*$ .

*Proof:*

1. Without loss of generality assume that  $a * b$ . If  $I * a @ b$  then  $b = I @ b * a @ b @ b = a @ b * b @ b = b$ . Hence  $a @ b = b$ . If  $I * a @ b$  then  $a @ b * I$ , in which case  $a = a @ a @ a @ b = a @ a @ b * a @ I = a$ , so  $a @ b = a$ .
2.  $a @ a @ b = a @ b$  (by idempotence) so  $a @ b * a$  and similarly  $a @ b * b$ . But by part (i) either  $a @ b = a$  or  $a @ b = b$ , so  $a @ b$  must be the smaller of  $a$  and  $b$  under the  $*$  order.
3. By part (ii),  $I$  is maximal under  $*$ . If  $a * b * I$  then  $a = a @ a @ a @ b * a @ I = a$ , so  $a @ b = a$ , that is  $a * b$ . If  $I * a * b$  then  $b = I @ b * a @ b @ b = b$ , so  $b * a$ .
4. If the ordinal is closed then  $0 * a = 0$  because  $? * a$ , being a left adjoint, must preserve initiality. Conversely, if  $0 * a = 0$  then for every  $a$  and  $b$  the set  $x | a * x * b$  is nonempty, and by finiteness and the fact that  $*$  is a total order it contains an element maximal under  $*$ . Defining  $[a, b]$  to be this element constructs a right adjoint of  $*$ .

height6pt width4pt depth0pt

A consequence of this proposition is that Gaifman's two models of concurrency are the only possible two-partial-order models, as shown in the following.

**Corollary 6** There are precisely two idempotent closed symmetric monoidal structures on the ordinal  $\mathbf{3}$ , namely the monoidal categories  $\mathbf{3}$  and  $\mathbf{3}'$  defined above.

*Proof:* By part (iii) of the proposition such a category is completely specified by listing the three elements in the  $*$  order. If  $I = 0$  the only possible order is 210. If  $I = 2$ , the only possibility is 012. For  $I = 1$  both 021 and 201 are possible. By part (iv) of the proposition only two of these are closed, namely 012, which describes  $\mathbf{3}$ , and 021, which describes  $\mathbf{3}'$ . height6pt width4pt depth0pt

## 2.4 The category of D-categories

As noted at the beginning of section 2, the class of **D**-categories forms a category. I will now define this category.

The arrows of the category are analogous to functors. Indeed, in the case that **D** = **Set** (so that a **D**-category is an ordinary category) the following simply restates the usual definition of functor. However, it has the advantage that it also applies to other choices of **D**.

**Definition .**

Given a monoidal category **D** and **D**-categories **A** and **B**, a **D**-functor  $F: \mathbf{A} * \mathbf{B}$  consists of the following data:

1. A map (also called  $F$ ) of the objects of **A** to the objects of **B**;
2. For each pair  $A, B$  of objects of **A**, an arrow in **D**.

This data must make the following diagrams commute:

1.  $F$  preserves composition:
2.  $F$  preserves identity:

height6pt width4pt depth0pt

### 2.4.1 Examples of D-functors

1. **Set**-functors are ordinary functors between ordinary categories. For this case the first condition means that a functor preserves composition and the second means that a functor preserves identity arrows.
2. **Cat**-functors are precisely 2-functors.
3. **Bool**-functors are monotone maps between preordered classes.
4.  $\mathbf{S}$ -functors are *noncontracting* maps between schedules. That is,  $F: \mathbf{A} * \mathbf{B}$  must satisfy  $A(X, Y) * B(FX, FY)$  for all objects  $X$  and  $Y$  of **A**.
5.  $\mathbf{S}$ -functors are *nonexpanding* maps, which are the maps satisfying

$$A(X, Y) * B(FX, FY).$$

Define the identity **D**-functor on any **D**-category to be the identity on both objects and homobjects.

Given **D**-categories **A**, **B** and **C**, and **D**-functors  $F: \mathbf{A} * \mathbf{B}$  and  $G: \mathbf{B} * \mathbf{C}$ , define the composition  $GF: \mathbf{A} * \mathbf{C}$  by defining the underlying object map by

$$(GF)A = G(FA);$$

and then defining  $[GF]A = [G][FA]$  to be the composite

This composition is associative and the identity functors are in fact identities for the composition. Thus, we have the following proposition.

**Proposition 8** The class of small **D**-categories and **D**-functors is a category.

This category will be denoted by **D**!

## 2.5 Tensor product of D-categories

Not only do the **D**-categories form a category but the tensor product of **D** lifts to a tensor product of **D**-categories, provided that **D** is symmetric monoidal. This is important here because if **D** is a space of measurements then the lifted tensor product turns out to be exactly the orthocurrence of sets measured in **D**.

So let  $\mathbf{D}$  be symmetric monoidal. The tensor product of two  $\mathbf{D}$ -categories  $\mathbf{A}$  and  $\mathbf{B}$  is defined as follows (see Kelly [Kel82]). The objects of  $\mathbf{A} \otimes \mathbf{B}$  are of the form  $A, B$  where  $A$  is an object of  $\mathbf{A}$  and  $B$  an object of  $\mathbf{B}$ . Homobjects are defined by

To define composition in  $\mathbf{A} \otimes \mathbf{B}$  first define the interchange natural isomorphism in  $\mathbf{D}$ :

$$r: (W @ X) @ (Y @ Z) \rightarrow (W @ Y) @ (X @ Z).$$

There are several ways to construct such a map by combining  $*$ ,  $*$  and identity maps using tensor product, inversion and composition. The coherence conditions will ensure that any two constructions of this form denote the same map but, for definiteness, define  $r$  to be the composite:

, the composition map in  $\mathbf{A} \otimes \mathbf{B}$  is then defined to be the composite map:

, the identity map is defined to be the composite arrow

**Proposition 9**  $\mathbf{A} \otimes \mathbf{B}$ , with  $*$  and  $i$  defined above, is a  $\mathbf{D}$ -category.

*Proof:* I show only the associativity law for  $\mathbf{A} \otimes \mathbf{B}$ . The identity laws are proved by a similar argument.

Let  $a, b, c$  be any objects of  $\mathbf{A}$  and  $d, e, f, g, h$  be any objects of  $\mathbf{B}$ . In the diagram on the following page  $a @ b$  is abbreviated as  $a \cdot b$ ,  $b @ c$  is abbreviated as  $b \cdot c$  and so on. Similarly  $d @ e$  is abbreviated as  $d \cdot e$  and so on. I wish to show that the two outside edges of that diagram are equal.

But the pentagon in the bottom right corner commutes because it is a tensor product of the pentagons that express associativity in  $\mathbf{A}$  and  $\mathbf{B}$ . The squares in the top right and bottom left corners commute because  $r$  is natural, and the hexagon in the top left corner commutes by coherence of  $\mathbf{D}$ .

I now extend the definition of  $@$  to  $\mathbf{D}$ -functors. Suppose that  $F: \mathbf{A} \rightarrow \mathbf{A}'$  and  $G: \mathbf{B} \rightarrow \mathbf{B}'$  are  $\mathbf{D}$ -functors and that  $a, b$  are objects of  $\mathbf{A}$  and  $d, e$  are objects of  $\mathbf{B}$ . Define the object map of  $F @ G$  by

Now the arrow  $a @ b$  must be of the form

Using the definitions of tensor product on categories and  $F @ G$  on objects, we find that we require an arrow with domain  $a @ b$  and codomain  $a' @ b'$ . An obvious choice for such an arrow is  $a @ b$ , and this is the definition we take. To show that  $@$  is functorial is just a matter of expanding definitions.

Furthermore, natural isomorphisms for associativity, left and right identity and symmetry can be defined, thus making  $\mathbf{D}$  into a symmetric monoidal category. I give only the definitions of the morphisms here; the proofs that the components are well-defined  $\mathbf{D}$ -functors, that they are invertible, and that they have the required naturality and coherence properties are both tedious and unenlightening. The interested reader will surely be able to perform the necessary diagram manipulations without prompting.

*Associativity:* Let  $\mathbf{A}$ ,  $\mathbf{B}$  and  $\mathbf{C}$  be  $\mathbf{D}$ -categories. I must define a  $\mathbf{D}$ -functor

Define  $*$  to take an object  $*X, Y, Z$  to  $*X, Y, Z$ . So each homobject arrow of  $*$  must be an arrow (in  $\mathbf{D}$ ) from  $(\mathbf{A}(X, X') @ \mathbf{B}(Y, Y')) @ \mathbf{C}(Z, Z')$  to  $\mathbf{A}(X, X') @ (\mathbf{B}(Y, Y') @ \mathbf{C}(Z, Z'))$ . Define this arrow to be the evident component of the associativity transformation of  $\mathbf{D}$ .

*Tensor Identity:* Define the tensor identity  $\mathbf{I}$  to have a single object (denoted by  $*$ ) and further define  $\mathbf{I}(*, *) = I$ .  $*$ :  $\mathbf{I}(*, *) @ \mathbf{I}(*, *) \rightarrow \mathbf{I}(*, *)$  is  $*$ :  $I @ I$  and  $i: I @ \mathbf{I}(*, *)$  is the identity arrow at  $I$ .

*Identity Isomorphisms:* The object map of  $*$ :  $\mathbf{I} @ \mathbf{A} \rightarrow \mathbf{A}$  sends each object  $*X$  to  $X$ . Thus the homobject arrows must be of the form  $\mathbf{I}(*, *) @ \mathbf{A}(X, X') \rightarrow \mathbf{A}(X, X')$ , which is  $I @ \mathbf{A}(X, X') \rightarrow \mathbf{A}(X, X')$ . Take this arrow to be  $*$  as defined in  $\mathbf{D}$ . Similarly,  $*$ :  $\mathbf{A} @ \mathbf{I} \rightarrow \mathbf{A}$  is defined to send each object  $X, *$  to  $X$  and its homobject maps are components  $*$  as defined in  $\mathbf{D}$ .

*Symmetry Isomorphism:* The object map of  $*$ :  $\mathbf{A} @ \mathbf{B} \rightarrow \mathbf{B} @ \mathbf{A}$  takes  $X, Y$  to  $Y, X$ , and the homobject arrows are  $*$ :  $\mathbf{A}(X, X') @ \mathbf{B}(Y, Y') \rightarrow \mathbf{B}(Y, Y') @ \mathbf{A}(X, X')$ .

This completes the definition of  $\mathbf{A} @ \mathbf{B}$ . The proof that these data define a  $\mathbf{D}$ -category is omitted.

## 2.5.1 Examples of tensor product

In the case that  $\mathbf{D}$  is cartesian monoidal the tensor product defined above is again categorical product. (This follows from the construction of limits in the proof of theorem ? below.) Thus for sets measured in  $\mathbf{2}$ , tensor product is the same as product. More interesting is the tensor product for sets measured in a non-cartesian space, such as  $\mathbf{or}$  or  $\mathbf{f}$ . For both  $\mathbf{or}$  and  $\mathbf{f}$  this tensor product is precisely the orthocurrence operation defined in section ?

## 2.6 MC-categories

In the preceding section it was observed that tensor product of  $\mathbf{D}!$  is orthocurrence of sets measured in  $\mathbf{D}$ . This is now taken further, by defining the other normal operations via purely category-theoretic constructs within  $\mathbf{D}!$ . Thus, the normal operations can formally be defined on *any* monoidal category possessing certain properties, whether that category arises as  $\mathbf{D}!$  for some  $\mathbf{D}$  or not. This is very useful because categories of measured sets need not be  $\mathbf{D}!$  for any  $\mathbf{D}$ . In this section I define the normal operations formally, along with a class of categories (the MC categories) to in which those operations are well-defined. In the next section I show that the operations defined here do match their previous definitions (with some minor changes). In outline: concurrence is coproduct, pairing is product, and exponentiation is internal-hom. Defining concatenation in categorical terms is more difficult, and requires reference to the property of *distributivity*, which will now be defined.

Let  $i$  and  $j$  denote the left and right injections into a coproduct. In any monoidal category with binary coproducts there exists a unique morphism  $*(A@C)+(B@C)*(A+B)@C$  making the following diagram commute.

### Definition .

A monoidal category with binary coproducts is *distributive* if each such  $*$  is an isomorphism.

Two facts about distributive categories are important. They are summarized in the following two propositions.

**Proposition 11**  $*$  is natural in  $A$ ,  $B$  and  $C$ .

*Proof* For arrows  $f:A^*A'$ ,  $g:B^*B'$  and  $h:C^*C'$  I must show commutativity of this diagram:

By the universal property of coproduct it is sufficient to show that the two routes around the square are equal when composed with the two injections into  $(A@C)+(B@C)$ . But

and

A similar argument applies for the composition with  $j$ .

**Proposition 12** A closed monoidal category with binary coproducts is distributive.

I first introduce notation that will shorten this proof and also be helpful in the following chapter when dealing with adjunctions.

Suppose that there is an adjunction  $F^*R$ . For any arrow  $g:FA^*B$  denote the adjoint transpose by  $\bar{g}$ . Dually, denote the adjoint transpose of  $f:A^*RB$  by  $\bar{f}$ . Observe that for appropriate arrows  $f, g$  and  $g'$  the following equations hold:

The dual results,  $\bar{f}g$  and  $\bar{g}f$  will also be useful.

Also, given arrows  $f:A^*C$  and  $g:B^*C$ , denote by  $(f, g):A+B^*C$  the unique arrow factoring  $f$  and  $g$  through  $A+B$ . Thus  $\bar{f}$  and  $\bar{g}$  and furthermore any arrow  $h:A+B^*C$  can be expressed as  $\bar{f}g$ .

*Proof* Let  $i$  and  $j$  be the injections, so that in the notation just introduced,  $i$  I wish to define  $\bar{f}g$ . This is equivalent, using the adjunction between tensor product and exponentiation, to defining an arrow  $A+B^*[C, ((A@C)+(B@C))]$ . This is in turn equivalent to defining two arrows: one  $A^*[C, ((A@C)+(B@C))]$ , the other  $B^*[C, ((A@C)+(B@C))]$ . Using the adjunction in the opposite direction, this is equivalent to defining an arrow  $A@C^*(A@C)+(B@C)$  and a corresponding arrow for  $B$ . An obvious choice for the last pair of arrows is the injections into the coproduct, which I will denote  $i$  and  $j$ , so that  $\bar{f}g$  is defined to be

I must show that this arrow is in fact  $\bar{f}g$ .

First observe that

Consider the arrow  $\bar{f}g$  occurring in the above expression. Its domain is  $A+B$  and composing with  $\bar{f}g$  yields

A similar argument shows that

hence that

But then

as required.

To show that composition in the other order also yields the identity it is sufficient to show that

and

But

A similar argument for composition with  $\bar{g}f$  completes the proof.

I am now able to give two basic definitions.

### Definition .

An *MCE category* is a monoidal category which is complete, monoidal, and distributive. An *MCE category* (measured class with exponentiation) is monoidal closed in addition to being MC.

It is important to note that, while categories whose objects are measured classes are important examples of MC categories, they are not the only ones. The structure of objects is important only insofar as it defines a tensor product; the normal operations will be defined below in terms of the tensor product without reference to any object level structure. For example, the cartesian closed ordinals, such as  $\mathbf{2}$ , discussed in the previous section, are MCE by definition though not categories of measured classes. Define the normal operations on objects of any MC-category as follows.

### Definition .

1. *Concurrence* is coproduct;
2. *Orthocurrence* is tensor product;
3. *Pairing* is categorical product;
4. *Constants*:  $\mathbf{0}$  is the initial object,  $\mathbf{1}$  the terminal object and  $I$  the tensor identity.
5. *Concatenation*. Concatenation is defined relative to a given object  $D$  of the MC category and arrow  $d:(I-I)^*D$ . The concatenation is defined by this pushout diagram:

where the map  $e$  is the composite

6. *Local Pairing*. The local operations are defined relative to an *object of locations*, denoted by  $L$  and thought of as being the set of locations. A *located object* of an MC category is an object  $A$  and an arrow  $f:A^*L$ . Think of this as an ordinary schedule  $(A)$  with an assignment of each event to one of the locations defined by  $L$ . I will frequently treat the location map  $f$  as understood and refer only to the located object  $A$ .

The local pairing of located objects  $f:A^*L$  and  $g:B^*L$  is their pullback:

Note that  $\cdot$  is located in  $L$ . Also the concurrence of two located objects can naturally be considered as an object located in  $L$  by taking the location map to be the following composite in which  $*$  denotes the co-diagonal map:

7. *Local Concatenation* is similar to ordinary concatenation but the pairing  $A*B$  appearing in the defining pushout is replaced by a local pairing  $\cdot$ . While this suffices to define the *events* of  $\cdot$  it does not provide a location map for it. We need an additional map  $l:D@L^*L$ , for then we can define the location map as the composition of  $l$  with the map defined in the following diagram.

To show that the diagram defines a map  $l$  must show that the outer edge commutes. The isomorphism  $L+L^*(l+\cdot)@L$  is the composite

Recall that

Hence the composite map around the top and right side of the diagram is

Naturality of  $*$  means that this is equal to

But by definition of local concatenation, so the naturality of  $*$  means this is equal to

as required.

In most applications of this concept there is no reason for  $L$  to enforce constraints on the events of different locations. For example, in the case of  $\cdot$ -categories  $L$  would usually be a set in which all distances were  $*$ . In such cases  $l:D@L^*L$  can simply be defined by  $l(d,a)=a$ .

8. *Exponentiation* is defined only if the category in question is MCE, in which case it is defined by the monoidal exponentiation.
9. The *fusion* defined by maps  $f:P^*Q$  and  $g:P^*R$  is the pushout of  $f$  and  $g$ .

height6pt width4pt depth0pt

## 2.7 Pomsets as an MC-category

How do the definitions of the preceding section correspond with the definitions of the pomset model, as discussed briefly in section 2 and in more detail by Pratt [Pra86]?

The first step in reconciling the set-theoretic notion of pomset with the category-theoretic definitions of the operations must be to define a suitable category of pomsets. Recall that, formally, a pomset is a triple  $P,f,>$ , where  $P$  is a partially ordered set and  $f:UP^{**}$  is a function from the underlying set of  $P$  to  $*$ .

**Definition .**

Let **Set** be the category of sets, and **Pos** be the category of posets and monotone maps. Let  $U:\mathbf{Pos}^*\mathbf{Set}$  be the forgetful functor. The comma category  $U^*\mathbf{Set}$  is the category of pomsets, denoted **Pom**.

To justify this definition, note that the objects of  $U^*\mathbf{Set}$  are triples  $P,f,S,>$ , where  $P$  is a poset,  $S$  is a set and  $f:UP^{**}S$  is a function. This is the same as the usual definition of pomset as given two paragraphs above. An arrow of this category  $P,f,S,> \rightarrow P',f',S',>$  consists of a monotone map  $p:P \rightarrow P'$  and a function  $s:S \rightarrow S'$  such that  $sf=f'(Up)$ . Considering pomsets as schedules, such a pair maps each event of its domain into an event of the codomain, and each action of the domain to an action of the codomain. The commutativity condition ensures that if event  $x$  is an occurrence of action  $a$  then the image of  $x$  is an occurrence of the image of  $a$ .

The following proposition can be proven from first principles. However I defer proof until section 2 where the proof is simplified by use of a general theorem.

**Proposition 16** *Pom is bicomplete. Hence it is an MC category via its cartesian monoidal structure.*

Observe that product and coproduct in **Pom** can be calculated separately in the partial order and the alphabet, so that the coproduct of  $*:UA^{**}$  and  $*':UA'^{**}$  is  $*+*':UA+UA'^{**+*}$ .

Firstly, it is clear that orthocurrence and pairing, defined category-theoretically, correspond exactly to Pratt's original definition of orthocurrence. However concurrence, as defined for a general MC category, is not the same as the original pomset definition. In the original definition the alphabet of the result is the *union* of the alphabets of the operands. Under the new definition it is the *disjoint union*. However this new operation, which I will call *disjoint concurrence*, has a reasonable interpretation when measured classes are seen as schedules, since it represents two processes running in parallel without communicating but where each action is now labeled with the identity of the process to which it belongs.

In concatenation the distinction between *disjoint concatenation* and ordinary or union concatenation must also be observed.

First consider the concatenation of posets and later extend to concatenation of pomsets.

**Proposition 17** *Let  $A$  and  $B$  be posets and let  $d:1+1 \rightarrow 2$  be a monic from the discrete poset with two elements to the ordinal 2. Then  $\cdot$  is the usual concatenation of  $A$  and  $B$ .*

*Proof:* The map  $e:(1+1)^*A^*B^*A+B$  in the definition of concatenation has the property that

$$\begin{aligned} e(0,a,b) &= a \\ e(1,a,b) &= b \end{aligned}$$

$d^*1:(1+1)^*A^*B^*2^*A^*B$  is easily seen to be the identity map on points.

I must show that the following diagram is a pushout, where the map  $e'$  is defined by the same equations as  $e$ , and  $i$  is the standard inclusion of  $A+B$  into  $A^*B$ . Both maps are easily seen to be monotone.

Commutativity is proven by a diagram chase. Next, suppose that there exists a poset  $X$  and monotone maps  $f$  and  $g$  satisfying the appropriate commutativity conditions, and construct a new map  $j$ :

Since  $i$  is both 1-1 and onto there exists a unique function  $j$  such that  $ji=g$ . This shows that  $j$  is unique. To complete the proof I show that  $j$  is monotone and  $je'=f$ . The fact that  $je'=f$  follows immediately the fact that  $f(1^*d)=ge$ .

To show monotonicity suppose  $x,y \in A+B$  and that  $x < y$ . If both  $x$  and  $y$  are in  $A$  then  $x < y$  is in  $A+B$ , so  $j(x)=g(x) < g(y)=j(y)$ . Similar remarks hold if  $x$  and  $y$  are both in  $B$ . The remaining case is  $x \in A$  and  $y \in B$ . But consider the triples  $*0^*x,y>$  and  $*1^*x,y>$ , both of which are elements of  $(1+1)^*A^*B$ . By definition of  $e$  and  $j$  it follows that  $j(x)=g(x)=ge(*0^*x,y>)$  and  $j(y)=g(y)=ge(*1^*x,y>)$ . By the assumption that the outer square of the diagram commutes, we know that  $ge=f(d^*1)$ . But  $d^*1$  is the identity function on the underlying sets, hence  $j(x)=f(*0^*x,y>)$  and  $j(y)=f(*1^*x,y>)$ , where the triples are now considered elements of  $2^*A^*B$ . But  $f$  is monotone and  $*0^*x,y> < *1^*x,y>$ , hence  $j(x) < j(y)$ , as required. This completes the proof. height6pt width4pt depth0pt

To extend this result to pomsets first replace the posets  $2$  and  $1+1$  by corresponding pomsets, which will be called **2** and **|2|**. Define **|2|** to have vertex set **2**, alphabet **|2|** and the identity as labelling function. Define **2** to have vertex set **2** and alphabet **|2|**, with labelling function defined by  $0^*0$  and  $1^*1$ .

It is well known that **Pos** is cartesian closed and bicomplete (so it is an MCE category). However this is not true for **Pom**, as will now be shown.

**Proposition 18** *Product in Pom does not preserve coequalizers. Thus Pom is not cartesian closed.*

*Proof:* Let  $E$  be the empty pomset with singleton alphabet. For any pomset  $P$ ,  $P^*E$  is an empty pomset with an alphabet isomorphic to  $P$ . Now consider the linear pomset  $a;b:a$  with alphabet  $a,b$ , and the single-element pomset  $a$  with alphabet  $a$ . There are two maps from  $a$  to  $a;b:a$  that have the inclusion  $a^*a,b$  as the alphabet map. The coequalizer of these two maps is the singleton pomset with singleton alphabet. (The two occurrences of  $a$  are identified and this forces the identification of  $a$  with  $b$  in the alphabet.) However taking the product of the coequalizer diagram with  $E$  yields a diagram of empty pomsets, and the coequalizer of this diagram is the empty pomset with a 2-element alphabet. Thus product with  $E$  does not preserve coequalizers. height6pt width4pt depth0pt

It will be shown below that replacing **Pos** in the definition of **Pom** by **Pre**, the category of preordered sets, does yield an MCE category.

As noted above the definition of concurrence as coproduct yields disjoint concurrence in **Pom** rather than union concurrence. However, union concurrence can be recovered by considering certain full subcategories of **Pom**.

**Definition .**

1. (the category of pomsets with alphabet  $*$ ) is the comma category  $V^{**}$ . (In this context  $*$  is the constant functor from the unit category **1** to **Set** whose unique value is  $*$ .)
2. Let **Seti** be the category of sets and inclusion maps, and denote by  $W$  the forgetful functor from **Seti** to **Set**. Then the comma category  $V$  is denoted **Pomi**.

height6pt width4pt depth0pt

The proof of the following is once again straightforward, though tedious, from first principles but will follow from results in a later chapter.

**Proposition 20** *Union concurrence is coproduct in Pomi. Union concurrence of two pomsets with common alphabet  $*$  is coproduct in  $*$ .*

Note that, although it would certainly be possible to define a concatenation for **Pomi** and  $\cdot$ , neither category is monoidal so the general definition cannot be used.

.

@page" @twocolumn"





## Construction of MC Categories n Labels

7