METRIC PROCESS MODELS

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
Roger Frederick Crew
November 1991

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.

Vaughan Pratt
(Principal Adviser)

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.

John Mitchell

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.

José Meseguer
(Center for Study of Language and Information)

Approved for the University Committee on Graduate Studies:

Abstract

Among the various formal models proposed to provide semantics for concurrency constructs in programming languages, partial orders have the advantages of conceptual simplicity, mathematical tractability, and economy of expression. We first observe that the theory of enriched categories supplies a natural abstraction of the notion of partial order, the $\mathcal{D}$-schedule. Varying the choice of temporal domain $\mathcal{D}$ allows for other forms of temporal constraint beyond that available from simple ordering. For example, having the constraints on inter-event delays be numeric bounds produces a generalized metric-space structure suitable for the discussion of real-time computation. We then construct an algebra of *processes* parametrized by notion of time. Here a process is a structure based on schedules that also incorporates nondeterminism. Since the model is category-based, we can define operations on $\mathcal{D}$-schedules and processes via universal constructions that depend little on the choice of $\mathcal{D}$. Also, given a suitable choice of process structure and process morphism, the constructions used for process operations and schedule operations are remarkably similar.

Acknowledgements

I would first of all like to thank my advisor, Vaughan Pratt, whose ideas, inspirations and criticisms have contributed substantially to this work. Ross Casley and José Meseguer, as collaborators, have also had a significant influence. Committee members John Mitchell and Robert Floyd provided useful comments. Conversations with various other people, including John Power, Mike Johnson, Phil Scott, Bard Bloom and Albert Meyer have also been quite helpful.

This research was supported in part by a National Science Foundation Graduate Fellowship.

And yes, I used Paul Taylor's commutative diagram macros. For that matter, I also used Don Knuth's TeX, Leslie Lamport's LaTeX, Supoj Sutanthavibul's `fig` package, Richard Stallman's GNU Emacs and numerous other pieces of software without which this work would have taken much longer to produce. Nor would it be right to ignore the excellent facilities and support provided by Stanford's Computer Science Department and CSD-CF.

Lastly, I am profoundly grateful to my parents (not to mention the rest of my family for that matter) for the encouragement and support they have provided over the years.

Contents

Outline

In Chapter 1, we present the concept of schedule, a collection of events with temporal constraints, which is rooted in the various extant partial order models of concurrency [8, 7, 15, 14, 18, 16], particularly that of the labeled partial order or *pomset* model of Grabowski [7] and Pratt [18]. Partial orders themselves are a sort of primaeval form of schedule having no further temporal information beyond some order constraints between events. We consider the similarities of the axioms of partial orders and metric spaces the goal being to define abstract schedule operations that are applicable to all. This is done by having an appropriate notion of schedule map or *morphism* and then using various universal constructions from category theory. Much of the material presented in this chapter is actually joint work with Casley, Meseguer and Pratt, and as such is mainly developed elsewhere (see [4, 3]). It does, however provide necessary groundwork for the process model presented in the following chapter.

Chapter 2 is the main focus of the thesis and is concerned with introducing choice or nondeterminism into the model. The initial vehicle for this is the disjunctive-normal-form (set of sets) representation of process that is inherent in the original partial order model and indeed in the trace theoretic model from which it derives. Once the appropriate additional structure is determined and once we have an appropriate definition for *process* morphism, many of the categorical definitions for operations go through as for schedules.

As an example, Chapter 3 considers the application of this model to dataflow nets, recasting a result of Gaifman and Pratt [6] from partial orders to the more general setting of physical $\mathcal{D}$-schedules.

Chapter 1

Temporal Structures

1.1 Events and Constraints

The most basic notion of *schedule* is that of set of events together with some further collection of constraints on these events that determine when they are allowed to occur. At first glance, it may seem sufficient to refer to one's intuitive notion of an event, remarking that we leave our events uninterpreted; i. e., the question of whether a particular event is a disk-read or a light bulb turning on is of no concern to us.

However, it should be clarified that in our scheme, events are atomic or *instantaneous* entities. This becomes important when we generalize the partial order notion of time, with its vague notions of 'before' and 'after', to a metric notion with actual numerical delays. For example, if we have events corresponding to a light bulb turning on and a door closing, what we actually need are particular moments, e. g., the instant in which the filament achieves two-thirds of its eventual brightness and the instant the door first contacts the doorjamb, in order to make meaningful a phrase like "the door closes 5 seconds after the light goes on."

It may seem unduly restrictive to insist that events be atomic when, in reality, almost none of the particular actions one wishes to model are actually atomic or instantaneous. However, in our usage, "event" means the referential moment rather than the full action. In our example, the closing of a door, which undoubtedly takes a few seconds to complete, is not an event *per se*. What usually happens, however is that one is not concerned with the duration of the door-closing action, and so one merely choses or takes as implicit an arbitrary moment during the swing of the door and identifies that event with the entire process of the door being closed.

This interpretation makes it possible to consider "refinement of events" in a realm where events are atomic, since now one is not refining an event so much as refining the particular tasks or actions of which that event is a part. The result of a refinement is *not* a subdivided event, but rather the inclusion of several additional referential moments of a given task or action into our schedule. The events themselves remain instantaneous, we just see more of them.

We next consider the question of the form the constraints should take. Perhaps the coarsest notion of constraint on a set of events is that of an ordering, namely that of a set of events, E, with a partial order relation $\leq$ constraining the temporal order in which these events may occur. In this case $\leq$ satisfies the usual axioms.

$$x \leq x \tag{R}$$

$$x \leq y \wedge y \leq z \Longrightarrow x \leq z \tag{T}$$

$$x \leq y \wedge y \leq x \Longrightarrow x = y \tag{A}$$

Recall the typical interpretation for $x \leq y$, namely that of being an abbreviation for $(x < y \text{ or } x = y)$, that is, "Either x occurs strictly before y or x and y are the *same event*." Curiously enough, this is not merely a statement about the relative timings of x and y, but the latter half makes an additional assertion about the identities of the events themselves. In particular, the antisymmetry axiom makes it impossible to constrain two distinct events to be simultaneous, since to do so in a partial order setting means imposing on a pair of events (x, y) both of the constraints $x \leq y$ and $y \leq x$, which leads inescapably to their not being distinct. Aside from this question of why one should not be allowed to constrain distinct events to be simultaneous, it will be seen that the omission of the antisymmetry axiom allows for a simpler framework. Thus, for the most basic notion of schedule, we prefer instead to use *preorders* (sometimes known as *quasi-orders*) which only satisfy (R) and (T).

As an alternative notion of temporal constraint, consider the standard axioms for a metric space $(E, d : E \times E \to \mathbf{R}^+)$.

$$d(x,x) = 0 \tag{R}$$

$$d(x,y) + d(y,z) \geq d(x,z) \tag{T}$$

$$d(x,y) = 0 \Longrightarrow x = y \tag{A}$$

$$d(x,y) = d(y,x) \tag{S}$$

Here, it suffices to note the similarity of form between these axioms and those for partial orders. The metric space is our paradigm for a more detailed notion of schedule, one specifying actual numerical delays between events rather than simply indicating the temporal precedences. As with partial orders, there are objections that may be raised to various axioms, e. g., the axiom (A) inspires the same objection as the correspondingly labeled axiom for partial orders, namely that it forbids distinct simultaneous events. As with partial orders, we shall see that discarding the offending axioms, in this case (A) and (S), leads to a useful structure in its own right. However, in this case what we are left with is actually quite different from the usual notion of metric space.

1.2 The Temporal Domain

We now consider the particular common features of preorders and metric spaces that should be kept when abstracting to a general notion of schedule. There is a set of events E, and a constraint map $\delta : E \times E \to \mathcal{D}$ giving the temporal constraint the schedule assigns to each pair of events. Here $\mathcal{D}$ is an abstract domain of temporal constraints, with the following attributes:

- some constraints imply others, so we at least have a partial ordering $\sqsubseteq$ on the constraints of $\mathcal{D}$
- a particular constraint (call it the *zero* or *instantaneous* delay) I that bounds the delay between an event and itself.

- a sequential composition of constraints $\otimes$, i. e., given constraints d, e on two consecutive delays, we should be able to infer a constraint $d \otimes e$ on the combined delay.

Naturally, $\otimes$ should be monotone, i. e., if either d or e are strengthened, $d \otimes e$ should be a correspondingly stronger constraint. Also, for any 3 consecutive constraints, it should not matter which two are composed first. Finally, if we compose a constraint with the instantaneous constraint I, we should not obtain any new information.

Definition 1 *An* ordered (commutative) monoid $(\mathcal{D}, \sqsubseteq, \otimes, I)$ *is a set* $\mathcal{D}$ *with a partial order* $\sqsubseteq$*, a binary operation* $\otimes : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$*, and a distinguished element* $I \in \mathcal{D}$ *such that* $(\mathcal{D}, \otimes, I)$ *is a (commutative) monoid, i. e., for all* d, e, f *in* $\mathcal{D}$

$$d \otimes (e \otimes f) \;=\; (d \otimes e) \otimes f \tag{1.1}$$

$$d \otimes I \;=\; d \;=\; I \otimes d \tag{1.2}$$

$$(d \otimes e \;=\; e \otimes d) \tag{1.3}$$

and such that $\otimes$ *is monotonic with respect to* $\sqsubseteq$*, i. e.,*

$$d \sqsubseteq e \wedge f \sqsubseteq g \implies (d \otimes f) \sqsubseteq (e \otimes g) \tag{1.4}$$

There are also logical operations.

- A given set of constraints $\{d_i\}_i$ applying to some inter-event delay can be equivalently stated as a join (conjunction) $\bigvee_i d_i$,

- There is a null (weakest) constraint, 0.

To formalize the above, we recall the following definition

Definition 2 *For any partial order* $(\mathcal{D}, \sqsubseteq)$ *and any subset* S *of* $\mathcal{D}$*, a* join *of* S *(denoted* $\bigvee S$ *or* $\bigvee_{s \in S} s$*) is any element* $\bar{s} \in \mathcal{D}$ *for which*

- $s \sqsubseteq \bar{s}$ *for all* $s \in S$*, and*

- *if* $t \in \mathcal{D}$ *is such that* $s \sqsubseteq t$ *for all* $s \in S$*, then* $\bar{s} \sqsubseteq t$*.*

Antisymmetry of $\sqsubseteq$ guarantees that joins are unique.

Definition 3 *A* (join) semilattice $(\mathcal{D}, \sqsubseteq, 0)$ *is a partial order* $(\mathcal{D}, \sqsubseteq)$ *together with a least element* 0 *having the property that every finite subset of* $\mathcal{D}$ *has a join,* 0 *being the join of the empty set. A semilattice is* complete *if* every *subset has a join.*

As it happens, a semilattice can just as easily have specified using only the join rather than the ordering $\sqsubseteq$:

Definition 3 [alternate] *A* complete semilattice $(\mathcal{D}, \bigvee, 0)$ *is a set* $\mathcal{D}$*, together with a distinguished element* 0 *and a function* $\bigvee : \mathcal{P}(\mathcal{D}) \to \mathcal{D}$ *(here* $\mathcal{P}(\mathcal{D})$ *is the powerset of* $\mathcal{D}$*) such that*

1. $\bigvee\{\} = 0$
2. $\bigvee\{d\} = d$
3. *For any collection* S *of subsets of* $\mathcal{D}$*,* $\bigvee \bigcup_{s \in S} s = \bigvee\{\bigvee s | s \in S\}$

These two definitions of complete semilattice are equivalent[9]; given a semilattice of the second form, the ordering $\sqsubseteq$ can be derived from the join operation

$$d \sqsubseteq e \iff \bigvee\{d, e\} = e$$

The corresponding alternate definition for ordinary semilattices uses $\mathcal{F}(\mathcal{D})$, the set of *finite* subsets of $\mathcal{D}$, and only requires (3) to hold for finite collections of finite subsets of $\mathcal{D}$.

In any case, we see that the desired structure for our temporal domain is something that is a combination of a complete join semilattice and an ordered monoid.

Definition 4 *A* (commutative) closed semiring *is any tuple* $(\mathcal{D}, \bigvee, \otimes, 0, I)$ *such that* $(\mathcal{D}, \bigvee, 0)$ *is a complete semilattice,* $(\mathcal{D}, \otimes, I)$ *is a (commutative) monoid and such that the following distributive properties hold*

$$\bigvee_i (d_i \otimes e) = (\bigvee_i d_i) \otimes e \tag{1.5}$$

$$\bigvee_i (d \otimes e_i) = d \otimes \bigvee_i e_i \tag{1.6}$$

Note that this distributivity implies the monotonicity (1.4) stipulation for ordered monoids.

We have now arrived at the desired notion of temporal domain:

Definition 5 *A* temporal domain $\mathcal{D}$ *is a commutative closed semiring* $(\mathcal{D}, \bigvee, 0, \otimes, I)$.

Though there is no reason to suppose *a priori* that the sequential composition $\otimes$ need be a commutative operation, restricting ourselves to this case simplifies the discussion.

1.2.1 Monoidal Categories

Closed semirings are a special case of monoidal categories that are closed and cocomplete. Recall that a (small) category $\mathcal{D}$ consists of

- a set of *objects*, $\mathrm{ob}(\mathcal{D})$
- for every pair of objects, $d, e \in \mathrm{ob}(\mathcal{D})$, a set (*homset*) of *morphisms* $\mathtt{Hom}(d, e)$ (a typical morphism therein being denoted $m : d \to e$),
- for every object $d \in \mathrm{ob}(\mathcal{D})$ a distinguished *identity* morphism $1_d : d \to d$, and
- for every triple of objects $c, d, e \in \mathrm{ob}(\mathcal{D})$, a *composition* operation $\circ : \mathtt{Hom}(d, e) \times \mathtt{Hom}(c, d) \to \mathtt{Hom}(c, e)$

with the usual associativity of $\circ$ and identity laws:

$$
\begin{aligned}
(l \circ m) \circ n &= l \circ (m \circ n) \\
m \circ 1_c &= 1_d \circ m = m
\end{aligned}
$$

for all morphisms $l : d \to e$, $m : c \to d$, and $n : b \to c$.

We then note that a preorder is equivalent to a small category in which we constrain all of the homsets to contain at most one morphism. Essentially, each instance of a $\sqsubseteq$ relationship between two constraints d, e becomes a morphism $d \to e$; if $d \sqsubseteq e$ does not hold, then we take $\mathtt{Hom}(d, e)$ to be empty. Each identity morphism is an instance of the reflexivity axiom, while the composition operation provides transitivity.

In the general categorical setting, we now have the possibility of having more than one morphism between constraints. The morphisms in some sense are *named implications*: whereas before we were satisfied with the knowledge $c \sqsubseteq d$, i. e., that d is a stronger constraint than c, we now have a morphism $m : c \to d$ designating *how* we know that c is a stronger constraint than d. The identity and associativity laws provide proof rules

$$\frac{}{1_d : d \to d} \qquad \frac{m : c \to d \qquad n : b \to c}{n \circ m : b \to d}$$

whereby we can derive additional constraint implications.

Taking the join of a set of constraints corresponds in the categorical setting to taking a *colimit* of a diagram of constraints; insisting that all joins be present is a matter of asserting *cocompleteness* for the corresponding category.

For a closed semiring, which has the additional operation $\otimes$, the analogous notion is that of a *monoidal biclosed* category or a *symmetric monoidal closed* category in the case of a commutative closed semiring. Here, we give only a brief, informal overview (see Mac Lane [13] for the full definition). A *monoidal* category $\mathcal{D}$ is an ordinary category together with

1. a functor $\otimes : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$
2. a distinguished object $I \in \mathrm{ob}(\mathcal{D})$
3. 3 natural isomorphisms

$$\begin{aligned} \alpha_{bcd} &: (b \otimes c) \otimes d \to b \otimes (c \otimes d) \\ \lambda_c &: I \otimes c \to c \\ \rho_c &: c \otimes I \to c \end{aligned}$$

such that for all $b, c, d, e \in \mathrm{ob}(\mathcal{D})$ the following diagrams commute:

$$\begin{tikzcd} ((b \otimes c) \otimes d) \otimes e \arrow[r, "\alpha"] \arrow[d, "\alpha \otimes 1"'] & (b \otimes c) \otimes (d \otimes e) \arrow[r, "\alpha"] & b \otimes (c \otimes (d \otimes e)) \\ (b \otimes (c \otimes d)) \otimes e \arrow[rr, "\alpha"] & & b \otimes ((c \otimes d) \otimes e) \arrow[u, "1 \otimes \alpha"'] \end{tikzcd}$$

$$\begin{tikzcd} (b \otimes I) \otimes c \arrow[rr, "\alpha"] \arrow[dr, "\rho \otimes 1"'] & & b \otimes (I \otimes c) \arrow[dl, "1 \otimes \lambda"] \\ & b \otimes c & \end{tikzcd}$$

These last conditions, known as *coherence* conditions, provide the actual "monoidal" structure. It would be far too restrictive to actually require $(b \otimes c) \otimes d = b \otimes (c \otimes d)$, even though this happens to hold in the preorder case; normally, the best we can do is stipulate that they be canonically isomorphic. The coherence conditions are sufficient to guarantee that, for every pair of functors $\mathcal{D}^n \to \mathcal{D}$ built up from $\otimes$ and I that are equivalent in a monoidal sense, i. e., one is obtainable from the other via applications of the associativity and identity rules for monoids, there is a unique natural isomorphism — that is, every means of arriving at such an isomorphism via combinations of α, λ, and ρ yields the same morphism. For a proof of this and a full discussion of coherence, see Chapter 7 of Mac Lane [13].

A *symmetric* monoidal category is one in which we have an additional natural isomorphism

$$\gamma : c \otimes d \to d \otimes c$$

and an additional set of coherence conditions. In the case of a preorder category, these are again vacuously satisfied and the whole structure reduces to our previous notion of commutative ordered monoid.

A symmetric monoidal category is *closed* if, for all $c \in \mathrm{ob}(\mathcal{D})$, the functor $c \otimes -$ (and hence, the functor $- \otimes c$ as well) has a right adjoint. If a functor possesses a right adjoint it preserves colimits. Furthermore in the case of preorder categories, the converse holds as well. Recall that in the preorder case, colimits are merely joins; thus, the statement that $c \otimes -$ preserves joins is merely one of the distributive laws we saw earlier (1.6). It is then straightforward to see that closed semirings are the preorder case of symmetric monoidal categories.

1.3 Schedules

Given a temporal domain $\mathcal{D}$, describing the corresponding notion of schedule is straightforward.

Definition 6 *A $\mathcal{D}$-schedule, $\mathtt{p} = (E, \delta)$ consists of a set E and a function $\delta : E \times E \to \mathcal{D}$ satisfying*

$$I \sqsubseteq \delta(x, x) \tag{R}$$

$$\delta(x, y) \otimes \delta(y, z) \sqsubseteq \delta(x, z) \tag{T}$$

for all $x, y \in E$.

A physical *$\mathcal{D}$-schedule, $\mathtt{p}$, is a $\mathcal{D}$-schedule $\mathtt{p} = (E, \delta)$ with the additional property*

$$(P) \qquad \delta(x, x) = I$$

for all $x \in E$.

The significance of physical $\mathcal{D}$-schedules is that while the more general domains of $\mathcal{D}$-schedules have a desirable completeness property, it allows for schedules that are overconstrained in the sense that no possible occurrences of actual events could satisfy it. The (P) axiom rules out these schedules, leaving only those which will be the actual objects of interest.

At this point we should consider some examples.

1.3.1 Preorders

The smallest nontrivial temporal domain **2** has just two constraints, a null constraint 0 and a tighter constraint 1 which also serves as the instantaneous constraint I. The join operation, $\vee$, is the the usual disjunction of truth values, while $\otimes$ is just $\wedge$. The map δ then defines a relation $\leq$ on E. The definition of **2**-schedule is equivalent to the axioms for $\leq$ being a preorder. Since 1 is the strongest constraint available, the additional condition (P) for physical **2**-schedules is vacuous.

1.3.2 Prosset orders

We next consider a slightly larger temporal domain $\mathbf{3} = \{0, 1, 2\}$, where $I = 1$, $0 \sqsubseteq I \sqsubseteq 2$ and $\otimes$ is given by

$$\begin{array}{c|ccc} \otimes & 0 & I & 2 \\ \hline 0 & 0 & 0 & 0 \\ I & 0 & I & 2 \\ 2 & 0 & 2 & 2 \end{array} \tag{1.7}$$

It so happens that given $I = 1$, this is the only possible choice for $\otimes$. Here, a map $\delta : E \times E \to \mathbf{3}$ effectively defines *two* relations, which we will denote $\leq$ and $\prec$ and define as follows:

$$x \leq y \Longleftrightarrow \delta(x, y) \sqsupseteq I \tag{$D\leq$}$$

$$x \prec y \Longleftrightarrow \delta(x, y) = 2 \tag{$D\prec$}$$

The rules (1.7) for $\otimes$ imply that $\leq$ is a preorder and that

$$x \prec y \implies x \leq y \tag{1.8}$$

$$x \prec y \leq z \implies x \prec z \tag{1.9}$$

$$x \leq y \prec z \implies x \prec z \tag{1.10}$$

It should also be clear that for any structure $(E, \leq, \prec)$ satisfying these properties, there is a unique map $\delta : E \times E \to \mathbf{3}$ such that $(D\leq)$ and $(D\prec)$ both hold.

A *physical* **3**-schedule is one in which $\prec$ is irreflexive. Essentially the axiom (P) rules out all $\prec$ cycles and when combined with (1.10) implies $x \leq y \Longrightarrow y \not\prec x$.

With this addition, the resulting set of axioms for $\leq$ and $\prec$ is exactly those for *prosset orders* as defined in [6]. Thus we obtain the simplest notion of temporal constraint (i. e., smallest temporal domain $\mathcal{D}$) in which we can represent the notion of strict precedence.

1.3.3 Other Finite Temporal Domains

As a particularly trivial case, one could also consider the temporal domain **1**, having a single (null) constraint. A **1**-schedule necessarily assigns the same constraint to all pairs of events. The resulting structures are indistinguishable from sets since the "constraints" provide no additional information.

In [4], a class of n-element, totally-ordered, idempotent ($a \otimes a = a$) temporal domains is described and it is noted that given n, there exist 2^{n-2} choices of $\otimes$ and I yielding an idempotent temporal domain. Both of the temporal domains **2** and **3** are instances of this class, as is another 3-element domain $\{0, 1, I\}$ having $\otimes$ given by

$\otimes$	0	1	I
0	0	0	0
1	0	1	1
I	0	1	I

Actually, we will have nothing further to say about the latter domain here, since we are more interested in schedules over temporal domains for which the physical axiom (P) is not vacuously true as it is in any case where I is the maximal constraint. For $n \leq 3$ this exhausts the possibilities.

1.3.4 Interval Spaces

We now return to the question of a domain of numeric delays. Take $\mathcal{D}$ to be $\bar{\mathbf{R}}^{op}$, the reals with $\pm\infty$ ordered in reverse; $+\infty$ is the null constraint, I=0, and we sequentially compose constraints via addition:

$\otimes$	∞	s	$-\infty$
∞	∞	∞	∞
r	∞	$r+s$	$-\infty$
$-\infty$	∞	$-\infty$	$-\infty$

The r's as constraints can be thought of as upper bounds on delays.

The $\bar{\mathbf{R}}^{op}$-schedule axioms are then

$$0 \geq \delta(x, x) \tag{R}$$

$$\delta(x, y) + \delta(y, z) \geq \delta(x, z) \tag{T}$$

Even if one stipulates the (P) axiom $\delta(x,x)=0$, this is still not a metric space. since we are missing both the symmetry axiom and the axiom that zero distance implies identity. The latter axiom is the one that in our context would forbid distinct simultaneous events. Lack of symmetry means that for every pair of distinct events, x, y, the metric δ actually specifies two bounds, an upper bound $\delta(x,y)$ and a *lower* bound $-\delta(y,x)$. A more intuitive name for $\bar{\mathbf{R}}^{op}$-schedule might be *interval space*.

In this case the significance of the (P) axiom is that, together with the triangle inequality (T), it serves to imply that all of the intervals are of nonnegative width, which is to say that the schedule is not overconstrained, but actually in some sense *physically realizable*, hence the motivation for the name *physical* $\mathcal{D}$-schedules.

1.3.5 $\mathcal{D}$-Categories

When we take the more general version in which $\mathcal{D}$ is a monoidal category the corresponding notion of schedule is that of a small category enriched over $\mathcal{D}$, otherwise known as a $\mathcal{D}$-*category* [1].

A $\mathcal{D}$-category, $\mathtt{p}$, consists of

- a set $V\mathtt{p}$ of "objects",
- a function $\delta : (V\mathtt{p})^2 \to \mathcal{D}$,
- a collection of morphisms in $\mathcal{D}$, $m_{xyz} : \delta(x,y) \otimes \delta(y,z) \to \delta(x,z)$ for each $x,y,z \in V$
- another collection of morphisms in $\mathcal{D}$ $j_x : I \to \delta(x,x)$ for each $x \in V$.

[1] It should be noted that in much of the literature on enriched categories, the monoidal category is held fixed and denoted as V, resulting in the name "V-categories." This conflicts with the use for V for vertex set which is nearly universal for graph theory.

such that the collections m and j satisfy the following commutativity conditions

$$\begin{tikzcd}
(\delta(u,v)\otimes\delta(v,w))\otimes\delta(w,x) \arrow[rr, "\alpha_{\delta(u,v)\delta(v,w)\delta(w,x)}"] \arrow[d, "m_{uvw}\otimes 1"'] & & \delta(u,v)\otimes(\delta(v,w)\otimes\delta(w,x)) \arrow[d, "1\otimes m_{vwx}"] \\
\delta(u,w)\otimes\delta(w,x) \arrow[r, "m_{uwx}"] & \delta(u,x) & \delta(u,v)\otimes\delta(v,x) \arrow[l, "m_{uvx}"']
\end{tikzcd}$$

$$\begin{tikzcd}
I\otimes\delta(u,v) \arrow[r, "\lambda_{\delta(u,v)}"] \arrow[d, "j_u\otimes 1"] & \delta(u,v) \arrow[d, equal] & \delta(u,v)\otimes I \arrow[l, "\rho_{\delta(u,v)}"'] \arrow[d, "1\otimes j_v"'] \\
\delta(u,u)\otimes\delta(u,v) \arrow[r, "m_{uuv}"] & \delta(u,v) & \delta(u,v)\otimes\delta(v,v) \arrow[l, "m_{uvv}"']
\end{tikzcd}$$

See Kelly [12] for the detailed definitions, though in the case where $\mathcal{D}$ is a preorder, this all reduces to the conditions given above; the further requirements are vacuous.

1.4 Schedule Maps

A model of concurrent computations should not only provide descriptions of individual schedules, but should also provide maps between schedules. Maps have a number of uses, of which we will mention two:

- A map relates a schedule to a particular observation of that schedule. For this to make sense, we need the notion of an observation as a maximally constrained schedule. In the case of a partially ordered set, the corresponding notion of a maximally constrained schedule is that of a total order, one in which every pair of distinct events is ordered. The natural notion of maps between partial orders is that of a monotone map, and we note that a particular observation satisfies the constraints of a particular schedule precisely when there exists a monotone map from the schedule to the observation.

- Given two schedules describing respectively e. g., the behavior of some component and the behavior of some constituent subcomponent, we may note that every event of the subcomponent schedule should appear somewhere in the schedule for the entire component. Furthermore any temporal constraints that exist between two such component events will apply to the delay between the corresponding events of the larger schedule for the component, which in general may be more constrained. In short, the map embeds the subcomponent schedule in the schedule for the larger component.

First we need a general definition of schedule map

Definition 7 *For general $\mathcal{D}$-schedules $\mathtt{p} = (E, \delta)$, $\mathtt{p}' = (E', \delta')$, a map $\mathtt{f} : \mathtt{p} \to \mathtt{p}'$ is any function $f : E \to E'$ for which $\delta(x, y) \sqsubseteq \delta'(f(x), f(y))$ for all $x, y \in E$*

For the specific examples, we have maps as follows.

- In the preorder case, this is just the notion of monotone map.
- For prosset orders, this is a function that is monotone w.r.t $\leq$, strictly monotone w.r.t. $\prec$.
- For interval spaces, this is a contraction function that takes intervals to possibly-smaller intervals, that is, given the interval $[-\delta(y, x), \delta(x, y)]$ bounding the delay between x and y, the above definition stipulates

$$\delta(x, y) \geq \delta(f(x), f(y))$$
$$-\delta(y, x) \leq -\delta(f(y), f(x))$$

  and hence $[-\delta(f(y), f(x)), \delta(f(x), f(y))] \subseteq [-\delta(y, x), \delta(x, y)]$.

- When $\mathcal{D}$ is a monoidal category, we have the corresponding notion of a $\mathcal{D}$-functor between $\mathcal{D}$-categories, $\mathtt{f} : \mathtt{p} \to \mathtt{q}$, which consists of
  - a set function ("object map") $V\mathtt{f} : V\mathtt{p} \to V\mathtt{q}$
  - a $\mathcal{D}$-morphism $\mathtt{f}_{xy} : \delta_{\mathtt{p}}(x, y) \to \delta_{\mathtt{q}}(V\mathtt{f}x, V\mathtt{f}y)$

  along with various diagrams ensuring the preservation of "compositions" and "identities" (which commute automatically when $\mathcal{D}$ is a preorder). This is more fully described by Kelly [12].

1.5 Schedule Operations

One of the motivating applications of this framework is that of providing an algebra of computations or processes while having the various operation definitions be relatively independent of our choice of temporal domain. Characterizing them as categorical operations is particularly useful towards this end. For schedules, this question has already been covered in [4], which describes a collection of schedule operations, some of which we review here.

1.5.1 Concurrence

Concurrence is the idea of combining schedules, or more generally processes or computations, in parallel. The key element is not that they are required to run at the same time as one might infer from "concurrent," but rather that there are no interactions between the processes, and that, therefore, they *could* be running at the same time with no particular impact on either.

Expressing this for partial order or preorder schedules is particularly straightforward: we take the disjoint union of the event sets keeping whatever orderings have already been given, leaving the remaining pairs of events unordered. This extends immediately to schedules over an arbitrary temporal domain, in which we take the union of the event sets, use the given constraints and place an initial or vacuous constraint on all of the delays between events from different schedules.

In this construction, when we take the concurrence $\mathtt{p} \parallel \mathtt{q}$ of two schedules $\mathtt{p}$ and $\mathtt{q}$, the canonical injections $i_1 : \mathtt{p} \to \mathtt{p} \parallel \mathtt{q}$ and $i_2 : \mathtt{q} \to \mathtt{p} \parallel \mathtt{q}$ are schedule maps in the above sense. The concurrence is, in fact, a *coproduct* in the category of $\mathcal{D}$-schedules. Since coproducts are always unique up to isomorphism, and since they exist in the category of $\mathcal{D}$-schedules as proved in [4], this can serve as the formal definition of our temporal-domain-independent notion of schedule concurrence.

We also note the existence of an identity for concurrence, namely the initial schedule $\mathtt{0}$, which has no events at all.

We eschew the more usual notation $\mathtt{p} + \mathtt{q}$ for coproduct, so as not to confuse concurrence with the disjunctive choice $(+)$ operation introduced in the next chapter.

1.5.2 Concatenation

Concatenation is the idea of sequentially composing computations. In the context of schedules, this means that all of the events of one schedule are constrained to occur before any of the events of the next schedule. This is similar to concurrence except that we have to introduce additional constraints to enforce this ordering between the schedules. Some temporal-domain independence is sacrificed here, since we have to choose which additional constraint to impose. Fixing such a constraint $d \in \mathcal{D}$, we can then go on to describe the concatenation of two schedules $\mathtt{p}$ and $\mathtt{q}$ as the schedule $\mathtt{p}\,;\mathtt{q}$ having the same events as $\mathtt{p} \parallel \mathtt{q}$, but whose constraints are the minimal set of constraints such that

$$\begin{aligned} \delta_{\mathtt{p}}(x,y) &\sqsubset \delta_{\mathtt{p};\mathtt{q}}(x,y) && \text{for all } x,y \in V(\mathtt{p}) \\ \delta_{\mathtt{q}}(x,y) &\sqsubset \delta_{\mathtt{p};\mathtt{q}}(x,y) && \text{for all } x,y \in V(\mathtt{q}) \\ d &\sqsubset \delta_{\mathtt{p};\mathtt{q}}(x,y) && \text{for all } x \in V(\mathtt{p}),\ y \in V(\mathtt{q}) \end{aligned}$$

This too can be characterized categorically. However, rather than refer specifically to a constraint $d \in \mathcal{D}$, we prefer to refer to a schedule $\mathsf{2} \in \mathcal{D}$-**Phys** having two events and a map $k : \mathtt{I} \parallel \mathtt{I} \to \mathsf{2}$. The schedule $\mathsf{2}$ may be thought of as a concatenation prototype with the afore-mentioned d appearing as the $\delta(0,1)$ constraint of $\mathsf{2}$.

Definition 8 *Given schedules* $\mathtt{p}$, $\mathtt{q}$, *and a map* $k : \mathtt{I} \parallel \mathtt{I} \to \mathsf{2}$, *their* concatenation *is defined to be an object* $\mathtt{p}\,;\mathtt{q}$, *together with a map* $c : \mathtt{p} \parallel \mathtt{q} \to \mathtt{p}\,;\mathtt{q}$, *such that*

1. *for any pair of maps* $(e : \mathtt{I} \to \mathtt{p}, e' : \mathtt{I} \to \mathtt{q})$, *there exists a unique map* $e;e' : \mathsf{2} \to \mathtt{p}\,;\mathtt{q}$ *making the following diagram commute:*

$$\begin{tikzcd} \mathtt{I} \parallel \mathtt{I} \arrow[r, "e \parallel e'"] \arrow[d, "k"'] & \mathtt{p} \parallel \mathtt{q} \arrow[d, "c"] \\ \mathsf{2} \arrow[r, "e;e'"] & \mathtt{p}\,;\mathtt{q} \end{tikzcd}$$

2. $c : \mathtt{p} \parallel \mathtt{q} \to \mathtt{p}\,;\mathtt{q}$ *factors uniquely through any other map* $\mathtt{p} \parallel \mathtt{q} \to \mathtt{r}$ *satisfying (1).*

Notice that we can express this as a single diagram as appears in [4]

$$\begin{tikzcd} \hom(\mathtt{I}, \mathtt{p} \times \mathtt{q}) \otimes (\mathtt{I} \parallel \mathtt{I}) \arrow[r] \arrow[d, "1 \otimes k"'] & \mathtt{p} \parallel \mathtt{q} \arrow[d, "c"] \\ \hom(\mathtt{I}, \mathtt{p} \times \mathtt{q}) \otimes 2 \arrow[r, "!"] & \mathtt{p} \,;\, \mathtt{q} \end{tikzcd}$$

so that concatenation is in fact a colimit.

1.6 Schedule Composition

Considering the second use of schedule map described above, one may then describe a notion of system composition as follows. We start with a collection of components each of which may be described by a schedule. These components may contain other (sub)components, so we also provide maps giving the subcomponent/component relationships.

Specifically, we have a directed *graph* $G = (V, W)$, each of whose vertices $v \in V$ is associated with some component and thence to the schedule $\mathtt{p}_v$ for that component; each edge $w \in W$ has a source $w_0 \in V$ and a target $w_1 \in V$ and a corresponding schedule map $f_w : \mathtt{p}_{w_0} \to \mathtt{p}_{w_1}$. Call this collection $(V, W, (\mathtt{p}_v)_{v\in V}, (f_w)_{w\in W})$ a *diagram* of $\mathcal{D}$-schedules.

The question then arises of what single schedule $\mathtt{p}$ best describes this entire system of components. The following two properties should hold of this system schedule.

1. All of the component schedules map into the system schedule $\mathtt{p}$ in a consistent way. Formally this means we have schedule maps $g_v : \mathtt{p}_v \to \mathtt{p}$ for all $v \in V$, and we require that for any edge $w \in W$ and any event $e \in E_{w_0}$ that $g_{w_1}(f_w(e)) = g_{w_0}(e)$ (i. e., no matter which component we view a particular event from, the g maps take it to the same event of $\mathtt{p}$).

Definition 9 *The collection $(\mathtt{p}, (g_v)_{v\in V})$ is said to be* consistent *with the given diagram if it satisfies this property ($g_{w_1} \circ f_w = g_{w_0}$ for all $w \in W$)*

2. The schedule $\mathtt{p}$ is the least constrained such schedule. That is, we now consider all schedules consistent with the given diagram; given any alternative consistent schedule $\mathtt{p}'$ with inclusions $(g'_v : \mathtt{p}_v \to \mathtt{p}')_{v \in V}$, the least constrained schedule $\mathtt{p}$ should have a map $h : \mathtt{p} \to \mathtt{p}'$ into it, and the inclusions should agree, i. e., $h(g_v(e)) = g'_v(e)$ for all components $v \in V$ and all events $e \in E_v$.

One further desired requirement is that for any given $\mathtt{p}'$ with inclusions $(g'_v : \mathtt{p}_v \to \mathtt{p}')_v$ h be unique. Note first of all, that once we include this in our list of desired properties for $\mathtt{p}$, we have immediately that $\mathtt{p}$ is unique up to isomorphism (since if $\mathtt{p}'$ also satisfies these properties, then we must have a unique map $h' : \mathtt{p}' \to \mathtt{p}$, and also unique maps $\mathtt{p} \to \mathtt{p}$ and $\mathtt{p}' \to \mathtt{p}'$ which by uniqueness must be identities, so hh' and $h'h$ are thus identities). Secondly, we could also view the $g' : \mathtt{p}_v \to \mathtt{p}'$ as representing a family of observations of all of the components $\mathtt{p}_v$, in which case this last uniqueness requirement gives us a bijective correspondence between such families of observations and single observations $h : \mathtt{p} \to \mathtt{p}'$ of $\mathtt{p}$.

Combining, we get

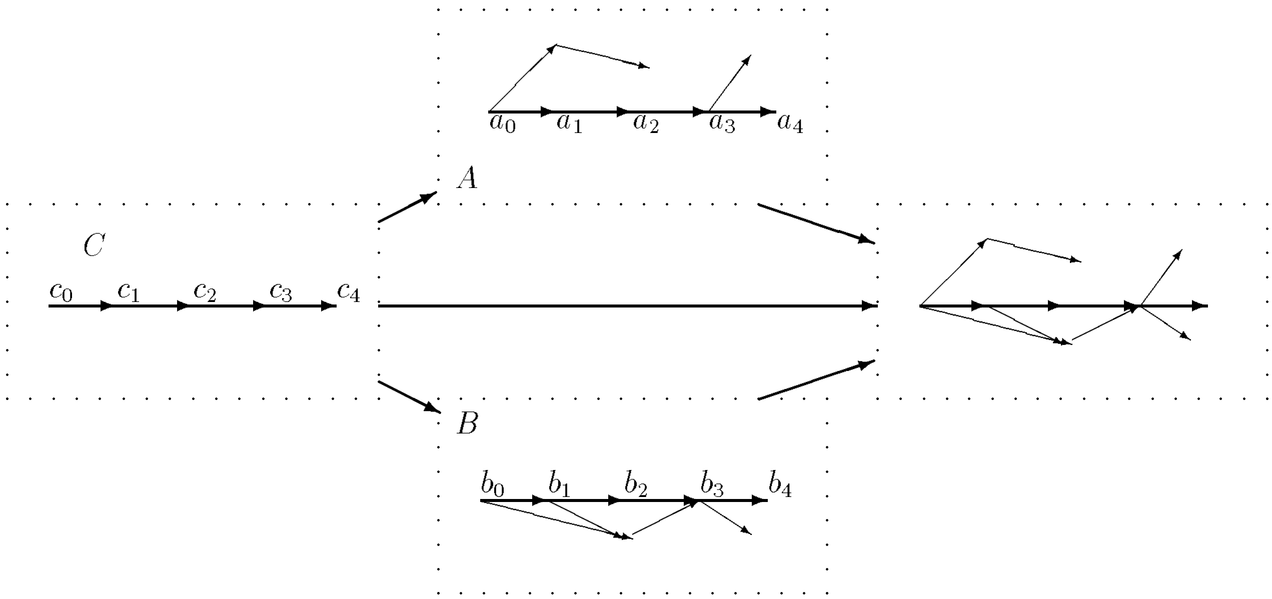
Definition 10 *Given a diagram of $\mathcal{D}$-schedules, $(V, W, (\mathtt{p}_v)_{v \in V}, (f_w)_{w \in W})$, its* composition, *is a consistent $\mathtt{p}, (g_v : \mathtt{p}_v \to \mathtt{p})_{v \in V}$ such that for all other consistent $\mathtt{p}', (g'_v : \mathtt{p}_v \to \mathtt{p}')_{v \in V}$ there is a unique $h : \mathtt{p} \to \mathtt{p}'$ such that $h \circ g_v = g'_v$ for all $v \in V$.*

In the terminology of category theory, a consistent $\mathtt{p}', (g'_v : \mathtt{p}_v \to \mathtt{p}')_{v \in V}$ is actually a *cocone* (dual to a *cone* in which the morphisms point *away* from the apex object $\mathtt{p}$), and the single $\mathtt{p}, (g_v : \mathtt{p}_v \to \mathtt{p})_{v \in V}$ through which all others map is nothing more or less than a *colimit* of the given diagram in the category of $\mathcal{D}$-schedules.

A particular example for the case of preorder schedules has (V, W) being the graph

$$\begin{tikzcd} & A \\ C \arrow[ur] \arrow[dr] & \\ & B \end{tikzcd}$$

i. e., 3 components and 2 component relationships, with the following corresponding schedules and schedule maps



In short, we have taken the processes A,B and synchronized the events a_i with b_i.

Given any directed graph (V, W) there is a corresponding category obtained by the standard construction of taking the objects to be the vertices and morphisms to be paths of edges. A diagram is then a functor from this category to the category of $\mathcal{D}$-schedules, a consistent schedule with its inclusions forms a *co-cone* over this diagram, and the notion of the composition of this diagram of schedules can be seen as the *co-limit* of this diagram.

Now, at this point, there is no guarantee that for any given diagram of schedules, the composition will actually exist given any such collection of component $\mathcal{D}$-schedules and maps. However, for the cases of interest to us we have

Theorem 11 *If $\mathcal{D}$ is a closed semiring, compositions as defined above always exist.*

One way of obtaining this is by applying the categorical sledgehammer and deriving it as a special case of the following result of Betti, Carboni, Street and Walters [1]:

Theorem 12 *If $\mathcal{D}$ is a closed monoidal category that is cocomplete, then $\mathcal{D}$-**Cat** is also cocomplete.*

But another way to understand this result in the closed semiring case is to consider the actual construction of the composition in the simple case where all of the event sets are finite.

First we consider the underlying event set of the composition.

Lemma 13 *The underlying event set E of the $\mathcal{D}$-schedule composition of (E_j, δ_j), $(f_k : E_{k_1} \to E_{k_2})_{k \in K}$ is just the $\mathbf{1}$-schedule (set) composition of the sets $(E_j)_{j \in J}$ and functions $(f_k)_{k \in K}$.*

Proof: The forgetful functor $V : \mathcal{D}\text{-}\mathbf{Cat} \to \mathbf{Set}$, taking each $\mathcal{D}$-schedule $\mathtt{p}$ to its underlying event set $V\mathtt{p} = E$, has a right adjoint R taking a set E to a schedule $RE = (E, \delta_{RE})$ on that set, the schedule RE being one in which all constraints are maximal (i. e., $\delta_{RE}(e, e') = (\bigvee \mathcal{D})$ for all $e, e' \in E$). Any functor having a right adjoint necessarily preserves colimits; thus, V preserves schedule compositions. ∎

One may now take a $\mathcal{D}$-schedule diagram and determine the underlying events for the composed schedule. This is the standard construction for the colimit in set: take the disjoint union of all of the event sets E_v and then quotient out by the minimal equivalence relation identifying every pair of events joined by one of the f_w (i. e., every pair of events $e \in E_v$, $e' \in E_{v'}$ for which there exists $w \in W$ such that $w_0 = v$, $w_1 = v'$ and $f_w(e) = e'$).

This gives us the underlying event set E and the underlying functions $(g_v : E_v \to E)_v$. One then has to determine the matrix of constraints $\delta_{\mathtt{p}}$ for the composed schedule. Start by filling in all of the constraints that appear in any of the components. That is, we start with

$$\delta^{(0)}(e, e') = \bigvee_{(v, e_v, e'_v) | g_v(e_v) = e' \wedge g_v(e'_v) = e'} \delta_v(e_v, e'_v)$$

This produces a matrix that has all of the minimum constraints necessary, but may not satisfy the transitivity and reflexivity rules (T) and (R) from the definition of schedules. One may then run a transitive closure algorithm (one extended to general closed semirings, e. g., Floyd/Warshall[5, 24, 21]), one that at each step replaces a constraint $\delta^{(n)}(e,e')$ with a possibly stronger constraint $\delta^{(n+1)} = \delta^{(n)}(e,e') \vee \delta^{(n)}(e,e'') \otimes \delta^{(n)}(e'',e')$ (for some $e'' \in E$), to eventually obtain the desired matrix $\delta = \delta^{(\omega)}$ which does satisfy the axioms (R) and (T).

For physical $\mathcal{D}$-schedules, the compositions need not exist in general. However, we still have the following helpful result

Theorem 14 *If the composition of a given diagram of physical $\mathcal{D}$-schedules exists, then this composition is isomorphic to the composition of the same diagram as (general) $\mathcal{D}$-schedules.*

This follows from

Lemma 15 *If $h : \mathtt{p} \to \mathtt{p}'$ is a general $\mathcal{D}$-schedule map, and* $\mathtt{p}$*' is a physical $\mathcal{D}$-schedule, then so is* $\mathtt{p}$*.*

Proof: If E is the event set of $\mathtt{p}$, then for all $e \in E$, $I \sqsubseteq \delta_{\mathtt{p}}(e,e) \sqsubseteq \delta_{\mathtt{p}'}(h(e),h(e)) = I$ ∎

Proof of Theorem 14: Given a diagram, we know that the general $\mathcal{D}$-schedule composition $(\mathtt{p}, g_v : \mathtt{p}_v \to \mathtt{p})$ exists and that if the physical $\mathcal{D}$-schedule composition $(\mathtt{p}', g'_v : \mathtt{p}_v \to \mathtt{p}'$ exists, then we have a (general $\mathcal{D}$-schedule) map $h : \mathtt{p} \to \mathtt{p}'$ for which $h \circ g = g'$. Lemma 15 then implies that $\mathtt{p}$ must be a physical $\mathcal{D}$ schedule, and thence that the maps g_v and h are all physical $\mathcal{D}$-schedule maps, from which it is fairly easy to conclude as before that $\mathtt{p}$ and $\mathtt{p}'$ must be isomorphic. ∎

So, to find the composition of a diagram of *physical* $\mathcal{D}$-schedules, one can compute their composition as general $\mathcal{D}$-schedules, and, if that turns out to satisfy the $\delta(x,x) = I$ requirement, then this composition will be the physical $\mathcal{D}$-schedule composition as well.

An example of where this kind of theorem does not hold is the case of the composition of a diagram of partial order schedules with monotone maps, calculated in the realm of partial order schedules, i. e., the category **Pos** of partially ordered sets and monotone functions, which we take to be a subcategory of **2**-**Cat**, the realm of preorder schedules. Calculating the schedule composition directly in **2-Cat** can yield a schedule that is quite different from what we get by calculating it in **Pos**. Consider the case where we have a diagram of three schedules, $A \leftarrow C \rightarrow B$, each having the same underlying set of two events $\{e_1, e_2\}$ with the underlying set maps being the identity on this set, but with the events unordered in C, ordered $e_1 \le e_2$ in A and $e_2 \le e_1$ in B. Their composition as preorders yields a two-event schedule $\{e_1, e_2\}$ where both orderings hold, however this resulting schedule is *not* actually a partial order. If the analogue to Theorem 14 held, this would imply that the partial order composition does not exist, but actually, there is such a composition, namely a single event schedule.

Notice that if we instead consider partial orders not as the subclass of preorders consisting of those that happen to satisfy the antisymmetry axiom, but rather as the subclass of *prosset* orders (i. e., in **3-Phys**) consisting of those satisfying the axiom

$$\delta(x, y) = I \Longrightarrow x = y$$

or equivalently

$$x \le y \Longrightarrow x = y \vee x \prec y$$

then we *do* have a corresponding result, namely that *if* the composition of a diagram of partial-orders exists *then* it is isomorphic to their composition as general **3**-schedules. This is a matter of going through the explicit construction above and noting that if the composed schedule has $\delta(x, y) = I$ for some pair of distinct events (x, y), then one of the schedules comprising the composition must also have such a pair of events.

The reason this does not contradict the previous paragraph is that while we are considering the same class of schedules, the schedule *maps* we allow are only those which are *strictly* monotone ($x \prec y \Longrightarrow f(x) \prec f(y)$). That is, our schedule category is not **Pos** (partially ordered sets and monotone functions) but rather $\mathbf{Pos}_<$ (partially ordered sets and *strict* monotone functions). The eventual impact of the choice to use only strict monotone maps is that it prevents distinct ordered events from being synchronized, i. e., mapped to the same event in the codomain.

As noted above, the composition of physical $\mathcal{D}$-schedules does not always exist, that is to say, $\mathcal{D}$-**Phys** is not cocomplete. Nor, for that matter, is $\mathbf{Pos}_<$. Nevertheless, both $\mathcal{D}$-**Phys** and $\mathbf{Pos}_<$ satisfy a somewhat weaker condition, and this will turn out to be sufficient to our needs when considering the question of *process* composition in the next chapter.

Definition 16 *A category* $\mathcal{C}$ *is* almost cocomplete *if, every diagram either*

- *has a colimit, or*
- *has no co-cone at all, i. e., for no object* $\mathtt{q}$ *does there exist a collection of morphisms* $(g_v : \mathtt{p}_v \to q)_{v \in V}$ *from the objects of the diagram* $(\mathtt{p}_i)_{v \in V}$ *into* $\mathtt{q}$ *such that the required commutativity properties hold.*

Restating this for schedule categories, we see that a schedule category is almost cocomplete precisely when every diagram of schedules either has a composition or is such that no schedule is consistent with it.

Theorem 17 *If* $\mathcal{D}$ *is a closed monoidal category that is cocomplete, then* $\mathcal{D}$-**Phys** *is almost cocomplete.*

Proof: It suffices to show that if a co-cone $(\mathtt{p}' : g_v : \mathtt{p}_v \to \mathtt{p}')_{v \in V}$ over a given diagram exists, then a co-limit exists as well. We already know that if we consider the corresponding diagram in $\mathcal{D}$-**Cat**, there will be a colimit $(\mathtt{p} : g_v : \mathtt{p}_v \to \mathtt{p})_{v \in V}$ and thus a unique ($\mathcal{D}$-**Cat**) map $h : \mathtt{p} \to \mathtt{p}'$. By Lemma 15, $\mathtt{p}$ and hence this co-limit is present in $\mathcal{D}$-**Phys** as well. ∎

1.7 Labeled Schedules

Analogous to the notion of labeled partial order or *pomset* as described in [19], we have the notion of a *labeled schedule* $\langle \mathtt{p}, \mu, \Sigma \rangle$. Here $\mathtt{p} = (E, \delta)$ is a schedule in the previous sense, and $\mu : E \to \Sigma$ is a labeling map. $\Sigma \in$ **Set** is referred to as the *alphabet*. As with ordinary schedules, we can make a category out of these, here using the comma category construction [13].

Definition 18 *Given categories $\mathcal{A}, \mathcal{B}, \mathcal{C}$ and functors $V : \mathcal{A} \to \mathcal{C}, W : \mathcal{B} \to \mathcal{C}$ the* comma category $(V \downarrow W)$ *is the category for which*

- *objects are triples $\langle a, \gamma, b\rangle$, where $a \in \mathrm{ob}(\mathcal{A})$, $b \in \mathrm{ob}(\mathcal{B})$ and $\gamma : Va \to Wb$ in $\mathcal{C}$.*
- *morphisms are pairs $\langle \alpha : a \to a', \beta : b \to b'\rangle$ for which the following diagram commutes:*

$$\begin{tikzcd} Va \arrow[r, "V\alpha"] \arrow[d, "\gamma"'] & Va' \arrow[d, "\gamma'"'] \\ Wb \arrow[r, "W\beta"] & Wb' \end{tikzcd}$$

- $1_{\langle a,\gamma,b\rangle} = \langle 1_a, 1_b\rangle$, *while* $\langle \alpha' : a' \to a'', \beta' : b' \to b''\rangle \circ \langle \alpha : a \to a', \beta : b \to b'\rangle = \langle \alpha' \circ \alpha : a \to a'', \beta' \circ \beta : b \to b''\rangle$.

.

Starting with a given category of ordinary schedules $\mathcal{T}$, the corresponding category of labeled schedules is then $(V \downarrow 1_{\mathbf{Set}})$, where $V : \mathcal{T} \to \mathbf{Set}$ is the forgetful functor taking an ordinary schedule to its underlying event set and $1_{\mathbf{Set}} : \mathbf{Set} \to \mathbf{Set}$ is the identity functor. If there is no ambiguity about the functors involved we denote this $(\mathcal{T} \downarrow \mathbf{Set})$. A morphism of labeled schedules $\langle \mathtt{p}, \mu, \Sigma\rangle \to \langle \mathtt{q}, \nu, \Gamma\rangle$ is then a pair of maps $\langle f, \phi\rangle$ where $f : \mathtt{p} \to \mathtt{q}$ is an (unlabeled) schedule map and

$$\begin{tikzcd} V\mathtt{p} \arrow[r, "Vf"] \arrow[d, "\mu"'] & V\mathtt{q} \arrow[d, "\nu"'] \\ \Sigma \arrow[r, "\phi"] & \Gamma \end{tikzcd}$$

commutes. Operations that are defined via universal constructions (e. g., concurrence or concatenation) follow automatically from the definition of morphism. [4] discusses these at length and in any case they are not qualitatively different from the corresponding operations already described for unlabeled schedules.

Carrying out this construction using $\mathcal{T} = \mathbf{Pos}$ yields the category **Pom** of labeled partial orders (pomsets).

Of interest however are those operations arising from label manipulations. For the rest of this section, $\mathcal{B}$ refers to a category of labeled schedules arising from the comma construction on some category $\mathcal{T}$ of ordinary schedules, be it **Pos**, $\mathcal{D}$-**Cat** or $\mathcal{D}$-**Phys** for some temporal domain $\mathcal{D}$,

For any particular alphabet $\Sigma \in \mathbf{Set}$, we can find within $\mathcal{B}$ the category $\mathcal{B}_\Sigma$ of behaviors over that alphabet. $\mathcal{B}_\Sigma$ is actually $(V \downarrow \Sigma)$, where $\Sigma : \mathbf{1} \to \mathbf{Set}$ is the functor taking the single object of $\mathbf{1}$ to the set Σ.

Now, for any alphabet translation $t : \Gamma \to \Sigma$ there are two canonical functors we can define:

- The *renaming* $t_* : \mathcal{B}_\Gamma \to \mathcal{B}_\Sigma$, which takes a behavior $q = \langle \mathtt{q}, \nu, \Gamma \rangle$ and relabels all of its events to produce $\langle \mathtt{q}, t\nu, \Sigma \rangle$.

$$\begin{array}{ccc} V(\mathtt{q} = t_*\mathtt{q}) & & \\ \Big\downarrow{\scriptstyle \mu} & & \\ \Gamma & \xrightarrow{\ \ t\ \ } & \Sigma \end{array}$$

- The *restriction* $t^* : \mathcal{B}_\Sigma \to \mathcal{B}_\Gamma$, which, in the case of t being a monomorphism, takes $p = \langle \mathtt{p}, \mu, \Sigma \rangle$ to $\langle \mathtt{p} \cap \mu^{-1}t(\Gamma), t^{-1}\mu, \Gamma \rangle$, that is, we take t^*p to be that subbehavior of p whose events are those with labels in $t(\Gamma)$. If, in fact, the underlying temporal structure category $\mathcal{T}$ *is* $\mathcal{D}$-**Cat**, which is actually complete, we can instead obtain the unlabeled schedule $t^*\mathtt{p}$ by taking the pullback

$$\begin{array}{ccc} t^*\mathtt{p} & \longrightarrow & \mathtt{p} \\ \Big\downarrow & & \Big\downarrow{\scriptstyle \mu} \\ E\Gamma & \xrightarrow{\ Et\ } & E\Sigma \end{array}$$

Here, $E : \mathbf{Set} \to \mathcal{T}$ is the right adjoint to $\mathcal{T}$'s forgetful functor (e. g., the clique functor in the case of preorders). That this is an equivalent construction is proven in [4].

The projection $\sigma : \mathcal{B} \to \mathbf{Set}$ (taking any behavior $\langle \mathtt{p}, \mu, \Sigma \rangle$ to its alphabet Σ) is in fact a *fibration* [10]. For a more detailed discussion of this in the context of labeled Petri-nets see [26].

In the case of general $\mathcal{T}$ where t is not injective, it is not necessarily the case that t^* exists. For our purposes, it will suffice to note that for any t for which t_* and t^* both exist, we have that t^* is a right adjoint of t_*, which is actually a stricter condition.

Given our concern about colimits, we should note that if $\mathcal{T}$ is almost cocomplete and its forgetful functor preserves colimits, then $\mathcal{B}$ is also almost cocomplete[2]. Likewise if $\mathcal{T}$ is complete, then so is $\mathcal{B}$.

One can then go on to define some basic behaviors and behavior operations using colimits, limits, restrictions, translations. Another source of operations is the lifting of the symmetric monoidal structure $(\otimes, I)$ from $\mathcal{D}$ to $\mathcal{D}$-**Cat** and thence to $\mathcal{B}$, providing both of the latter categories with a $\otimes$ ($\otimes_{\mathcal{B}}$ is orthocurrence as in [20]) and an I. Details of the constructions are in [4].

---

[2]In the case of **Pom**, this result is not applicable, but we still have cocompleteness. However, colimits in **Pom** do not respect the alphabets, that is, $\sigma : \mathbf{Pom} \to \mathbf{Set}$ does not preserve colimits. For this reason we prefer, when considering partial orders to either

1. think of them as preorders, i. e., work in **2-Cat**, allowing for the possibility of distinct simultaneous events, or

2. interpret the order relations as being strict, i. e., work in $\mathbf{Pom}_< = (\mathbf{Pos}_< \downarrow \mathbf{Set})$,

and avoid using **Pom** altogether.

Chapter 2

Processes

2.1 Choice and Process Morphisms

The element missing from the the machinery presented thus far for schedules is that of nondeterminism or *choice*. In particular, as we have presented it, a schedule carries the implicit assumption that all of the constituent events will be enacted. If one wishes to indicate that an event need *not* occur in certain circumstances, let alone the possibility that two events may mutually exclude each other, additional structure is required. It is this augmented schedule-with-choices notion that we shall refer to as a *process*.

Winskel [15], in his event structure model, provides for nondeterminism by producing a single schedule (i. e., the partial order case thereof) and annotating it with a *conflict* relation. The interpretation placed on this is that only some conflict-free subset of the given events will occur. Thus, a process that enacts one of the events a or b but not both would be depicted as a schedule containing the events a and b, but also containing the pair (a, b) in its conflict relation. Note that in this case we still have to provide temporal information relating a and b even though conflict renders such information irrelevant. While the event structure model provides for a fairly simple structure, its conflation of causal and temporal order is troublesome when attempting to derive generalizations in which the temporal information is given by something other than a partial order. While we will not go so far as to claim that causality, unlike time, is fundamentally a binary relation, i. e., either one event causes another or it does not, we do however suggest that extensions to this depiction of causality will not necessarily take the same form as the corresponding extension to the notion of time as a partial order.

The approach we shall take is the one adopted by Pratt [20] and Grabowski [7] in the original pomset model, namely that which represents a choice by using distinct schedules. In this framework, a process is then a *set* of schedules. The interpretation here is that some schedule in the process' set will be followed, i. e., there exists a schedule in the process' set for which all events will occur in accordance with its temporal constraints. For example, the above process that can perform one of a or b is now a set containing two schedules, each having but a single event. Additional structure is clearly necessary, but for now we consider the implications of this set-of-sets notion of process by itself.

Henceforth, $\mathcal{B}$ shall refer to some category of schedules, labeled or unlabeled, as developed in the previous chapter (e. g., $\mathcal{D}$-**Phys** for some suitable temporal domain $\mathcal{D}$). $\mathcal{B}$ essentially serves as a parameter in the process construction.

2.1.1 Processes as DNF Propositions

So consider, for now, the individual schedules as bare sets, i. e., ignoring any temporal information that may be present. We then have an outer set, which we shall call the *alternative* or *index* set, i. e., the set of alternatives indexing the schedules from which the process has to choose. The first observation is that this form of process is essentially a disjunctive normal form (DNF) representation. That is, we have a set of disjuncts, namely the alternatives, each of which is itself a conjunction, i. e., a schedule of events which *all* occur. Taken as a whole, the process, stripped of its temporal information, is simply a DNF proposition concerning which events will actually occur.

Since a major goal of this work is to extend the notion of schedule composition to process composition, we should, in the course of establishing our notion of process, also establish a corresponding notion of morphism from one process to another, the intention being that these morphisms play exactly the same role for processes as the above-described schedule morphisms do for schedules, e. g., relating an observation to a process or embedding one process in another.

With our processes being DNF propositions, a natural choice of morphism to use to relate these propositions is that of *implication*. Consider the special case of conjunctions, where a suitable way to demonstrate that

$$\bigwedge_{i \in I} p_i \Longrightarrow \bigwedge_{j \in J} q_j$$

is to exhibit a function $f : J \to I$ such that for all $j \in J$ we have $p_{f(j)} \Longrightarrow q_j$. Such a function provides an elementary proof by demonstrating that every conjunct on the right is implied by something on the left. Moreover, if the implication holds, there will exist a proof of this form, though there is certainly no claim about it being unique. Dually, we may prove

$$\bigvee_{i \in I} p_i \Longrightarrow \bigvee_{j \in J} q_j$$

via a function $f : I \to J$ such that for all $i \in I$ we have $p_i \Longrightarrow q_{f(i)}$. Here, every disjunct on the left implies something on the right.

A category $\mathcal{C}$ determines a category of indexed families of objects of $\mathcal{C}$ via a standard construction.

Definition 19 *For any category $\mathcal{C}$, the category* $\mathbf{Fam}(\mathcal{C})$ *has objects of the form* $\mathbf{C} = (I, (C_i)_{i \in I})$ *for some index set* $I \in \mathbf{Set}$ *and some collection of objects* $C_i \in \mathrm{ob}(\mathcal{C})$*, and morphisms* $\mathbf{f} : \mathbf{C}^1 \to \mathbf{C}^2$ *of the form* $(f : I^1 \to I^2, (f_i : C_i^1 \to C_{f(i)}^2)_{i \in I^1})$*, where f is a function and the f_i are morphisms of $\mathcal{C}$. Composition and identity are given by* $\mathbf{gf} = (gf, (g_{f(i)} f_i)_{i \in I^1})$ *and* $1_{\mathbf{C}} = (1_I, (1_{C_i})_{i \in I})$ *respectively.*

That is, we provide a function between the index sets, and for every element of the domain's index set we provide a morphism relating the corresponding objects of $\mathcal{C}$. Thus, given a category $\mathcal{A}$ of atomic propositions and implications thereof, we can construct

- $\mathbf{Fam}(\mathcal{A})^{op}$, the category of conjunctive propositions on $\mathcal{A}$ (i. e., conjunctions of atomic propositions). Notice that in the case of conjunctions, the direction of the implication runs opposite to that of the functions involved, thus the need for the $-^{op}$.

- $\mathbf{Fam}(\mathbf{Fam}(\mathcal{A})^{op})$, the category of disjunctions of propositions of $\mathbf{Fam}(\mathcal{A})$, or, equivalently, the category of DNF propositions on $\mathcal{A}$.

We now wish to generalize the two-step construction above by replacing $\mathbf{Fam}(\mathcal{A})$ by the schedule category $\mathcal{B}$. According to this generalization, a schedule can be interpreted as a generalized conjunction of event-occurrence propositions. Therefore, it is not actually necessary to produce a category $\mathcal{A}$ of atomic event-occurrence propositions, since in the case of schedules, this provides no new information about the identities of events beyond what the schedule morphism itself already provides. If a schedule morphism maps one event to another, those events are identified and thus the occurrence of one implies the occurrence of the other vacuously.

It is, however important to note that schedule morphisms run in a direction inverse to that of the implication between the corresponding propositions. That is, if we have a schedule morphism $R \to S$, any occurrence of events satisfying S necessarily satisfies R as well. Thus, as propositions, $S \implies R$. The actual category of schedules-as-propositions is then $\mathcal{B}^{op}$, which suggests that the corresponding category of processes, i. e., DNF propositions instead of pure conjunctive propositions, is going to be $\mathbf{Fam}(\mathcal{B}^{op})$.

2.1.2 Adding Structure to the Alternative Set

Of course, we should now consider what the actual structure on processes should be. While our main motive in using a DNF style of process is that of achieving a separation of causal and temporal information, the structureless set of schedules described thus far achieves, in some sense, much too complete a separation.

- Having an unstructured set of choices removes the possibility of describing *when* a particular choice takes place. Thus we have the generally recognized advantage of *branching-time* models.

- There is nothing to relate events on distinct schedules within a process.

To elaborate on the latter objection, if, e. g., one wishes to specify that, once events a and b occur, c *may* occur, one may consider a process of two schedules whose event sets are $\{a,b\}$ and $\{a,b,c\}$. However, unless we stipulate that all events used in schedules are drawn from some universal set of events, there is nothing to indicate that the two a's are supposed to refer to the same event ("same" in the sense that whether the event occurs has the same implications for observers of this process, no matter that it occurred in one schedule or the other). The situation becomes even more obscure should we be using labeled schedules and thus might want to specify something like "c *may* occur once a has occurred *twice*," though this is perhaps more properly a hazard of using event labels to specify events.

Thus the additional structure on processes should establish the required identities between events of the constituent schedules. We already have a mechanism to facilitate this, namely that of schedule morphisms. The alternative set will then be (at least) preordered, the idea being that if an event c may optionally occur but only given that some other collection of events has occurred, we essentially have two schedules (alternatives), one containing only the predecessor events, the other containing c as well; the latter alternative being later in the ordering on alternatives.

This mapping of alternative orderings to schedule morphisms suggests viewing the alternative set itself as a category, thus allowing for the concise definition of process as being a functor into the schedule category $\mathcal{B}$.

Definition 20 *Given a category (of schedules) $\mathcal{B}$, a* process *P over $\mathcal{B}$ is a category A_P together with a functor $P : A_P \to \mathcal{B}$.*

Having upgraded the alternative set from a mere set to a category, we should also present the corresponding upgrading of the $\mathbf{Fam}(\mathcal{B}^{op})$ notion of process morphism. As might be expected, however, we have both the order structure on the alternative set and the various schedules' temporal constraints to contend with. In this case, the function between the alternative sets becomes a functor and the family of schedule functions becomes a natural transformation.

Definition 21 *A* process morphism *$f : P \to Q$ is a functor $A_f : A_P \to A_Q$ together with a natural transformation $f : QA_f \stackrel{\cdot}{\to} P : A_P \to \mathcal{B}$. We denote the category of processes over $\mathcal{B}$ as $\mathcal{P}[\mathcal{B}]$.*

As a matter of convention, we take the direction of the process morphism to be the same as that of the implication of the corresponding propositions, which also happens to match the direction of the underlying set function on the alternative sets. It is only in the unfortunate situation of schedules and schedule morphisms that the directions of the set functions and the implications happen to be different. Given that we wish to adhere to previous conventions for schedule morphisms, the natural transformation must necessarily be in the opposite direction.

The import of the naturality itself

$$\begin{tikzcd} QA_f(p) \arrow[r, "f_p"] \arrow[d] & P(p) \arrow[d] \\ QA_f(p') \arrow[r, "f_{p'}"] & P(p') \end{tikzcd}$$

is simply that if satisfaction of P implies satisfaction of Q, P should "agree with" any identification of events made between schedules of Q. Otherwise an anomalous situation arises wherein for some pair of events of P, one has occurred, one has not, and Q has both of these events identified.

Despite the categorical machinery thus far invoked, we have not yet introduced any real structure since while this definition allows for the introduction of a nontrivial ordering on A_P and indicates how, in such a case, these orderings should be reflected in the constituent schedules, it does not require the ordering to be anything other than discrete. Further stipulations will be necessary to obtain a more recognizable notion of process. To that end, we now consider the various subcategories of $\mathcal{P}[\mathcal{B}]$ in which our true interest lies.

If we consider only processes P for which A_P is a discrete category, we are, as noted above, back to our previous definition which corresponds to the usual notion of process as a set of schedules with no further relationships between the various alternative schedules. We will refer to these as discrete processes and the corresponding (full) subcategory as $\mathcal{P}_d[\mathcal{B}]$.

While there is implicit in our choice of a DNF form for process a general desire to separate causal from temporal information, one can only take this so far. In particular, it is still generally accepted that cause precedes effect. In fact, it is only the converse implication of temporal order implying causation that we wish to suppress. Thus, we would like a stipulation that requires the schedule corresponding to a later alternative in the alternative set ordering to contain all of the events of any predecessor schedules and furthermore to not introduce any tighter temporal constraints between any pairs of such events.

For this purpose we develop the notion of prefix.

Prefixes

As noted earlier, if the structure of the process does not relate two of its alternative schedules in any way, then the choice between them is essentially made at the beginning of time. If we could identify events on separate schedules within a process, this would provide a way of representing a *deferred* choice in the sense that an occurrence of an event or collection of events common to both schedules does not necessarily commit the process to be following one or the other of the schedules in question. That is, the choice made between the schedules is not seen to take place until either

- we have the occurrence of some event that is present in only one of the schedules
- some event common to both schedules occurs, but the timing of the occurence violates the constraints given by one of the schedules.

However, given that the morphisms we use to identify events have an implicit directionality to them, we prefer to view the general situation of two schedules sharing some collection of events as rather that of two schedules sharing a common predecessor schedule. In other words, we have 3 schedules, one which the process follows "before" the choice is made and the other two which represent the possible outcomes of the choice. This view decomposes all choices into predecessor-successor pairs, i.e., all choices are now of the form, "Either we stay with this schedule or we advance to the next one." What remains is to clarify the relationship between a schedule $\mathtt{p}$ and any successor schedule $\mathtt{p}'$.

In the realm of partial orders, a prefix of an order $\mathtt{p}$ is a downward closed subset $\mathtt{p}_0$, i.e., such that for any elements $x, y \in \mathtt{p}$, if we have that $x \leq y$ and $y \in \mathtt{p}_0$ then x

must be in $\mathtt{p}_0$ as well. This can equivalently be represented as a pullback

$$\begin{tikzcd} \mathtt{p} \arrow[r, hook] \arrow[d, "!"'] & \mathtt{p}' \arrow[d, "\chi"] \\ \mathtt{1} \arrow[r, "0"] & \mathtt{2} \end{tikzcd}$$

where $\mathtt{2}$ is the usual two-element order $\{0, 1|0 \le 1\}$. The pullback essentially makes $\mathtt{p}$ be the inverse image under χ of 0 and it is the ordering on $\mathtt{2}$ that forces the prefix property.

One complication in translating this to the more general schedule frameork is that the terminal object $\mathtt{1}$ may not actually exist in $\mathcal{B}$. For example, this is usually the case if we take $\mathcal{B} = \mathcal{D}$-**Phys**. Likewise the self-constraints in $\mathtt{2}$ will need to be terminal in order for these pullbacks to exist in general. With $\mathcal{D}$-**Phys** we can, however, consider the corresponding diagram to be in $\mathcal{D}$-**Cat**. $\mathtt{1}$ and $\mathtt{2}$ thus need only be objects of $\mathcal{D}$-**Cat**. The general definition of prefix is then as follows:

Definition 22 *Given a category of schedules $\mathcal{B}$, a* prefix classifier *for $\mathcal{B}$ is a map $k : \mathtt{1} \to \mathtt{2}$ in a category $\overline{\mathcal{B}}$ of schedules, such that*

- *$\overline{\mathcal{B}}$ is complete and contains $\mathcal{B}$ as a full subcategory,*
- *the inclusion $i : \mathcal{B} \to \overline{\mathcal{B}}$ is a monoidal functor (i.e., preserves $\otimes$ and $\mathtt{I}$) and respects the forgetful functors into* **Set***, i. e., $V_{\overline{\mathcal{B}}} \circ i = V_{\mathcal{B}}$*
- *$\mathtt{1}$ is a terminal object in $\overline{\mathcal{B}}$.*
- *$\mathtt{2}$ is a schedule of $\overline{\mathcal{B}}$ having two events v and w, with $\delta_1(u, u) = \delta_2(v, v)$, $\delta_2(w, v)$ initial, $\delta_2(v, w)$ terminal.*

Definition 23 *Given a prefix classifier k for a category of schedules $\mathcal{B}$, a morphism $\mathtt{p} \to \mathtt{p}'$ is a* prefix morphism *(or* prefix map*) iff there exists a morphism $\chi : \mathtt{p}' \to \mathtt{2}$*

such that

$$\begin{tikzcd} \mathtt{p} \arrow[r, hook] \arrow[d, "!"'] & \mathtt{p}' \arrow[d, "\chi"] \\ \mathtt{1} \arrow[r, "k"] & \mathtt{2} \end{tikzcd}$$

is a pullback.

A proper *prefix is one for which the inclusion* $\mathtt{p} \to \mathtt{p}'$ *is not an isomorphism.*

For preorders **2-Cat**, the only choice for $\mathtt{2}$ is the usual ordinal, and $k : 1 \to 2$ is the inclusion of the lower element. This definition of prefix then yields the usual notion of prefix for a preorder (x in the prefix and $y \leq x$ imply y in the prefix).

Theorem 24 *Given a schedule category of the form* $\mathcal{B} = \mathcal{D}$-**Phys** *having a prefix classifier* $k : \mathtt{1} \to \mathtt{2}$*, and a* J*-diagram* $(\mathtt{f}_m : \mathtt{p}_i \to \mathtt{p}_j\ )_{(m:i\to j)\in J}$ *with colimit* $(\iota_i : \mathtt{p}_i \to \mathtt{p})_{i\in J}$*, and a morphism* $\chi : \mathtt{p} \to \mathtt{2}$ *or equivalently, a compatible collection of morphisms* $\chi_i : \mathtt{p}_i \to \mathtt{2}$ *(here,* $\chi_i = \chi\iota_i$*), we have that the diagram* $(\mathtt{g}_m : \mathtt{q}_i \to \mathtt{q}_j \mid m : i \to j \in J)$*, in which* $\mathtt{q}_i$ *is the* χ_i*-prefix of* $\mathtt{p}_i$ *and the* $\mathtt{g}_i$ *are the canonical maps, has the colimit* $(\mathtt{q}, \kappa_i : \mathtt{q}_i \to \mathtt{q})$*, where* $\mathtt{q}$ *is the* χ*-prefix of* $\mathtt{p}$*.*

In other words, the colimit of a diagram of prefixes is a prefix of the colimit.

Having fixed a particular prefix classifier in the schedule category $\mathcal{B}$ we can then consider the following class of processes:

Definition 25 *A process* P *is* prefix ordered *iff*

- A_P *is a preorder*
- *For all* $p, p' \in A_P$*:* $p \leq p'$ *implies* $P_{pp'} : P(p) \to P(p')$ *is a prefix morphism.*

A process morphism $f : P \to Q$ *is* prefix preserving *iff for each* $p \leq p'$ *in* A_P*, the naturality-of-*f_p *diagram*

$$\begin{tikzcd} QA_f(p) \arrow[r, "f_p"] \arrow[d, hook] & P(p) \arrow[d, hook] \\ QA_f(p') \arrow[r, "f_{p'}"] & P(p') \end{tikzcd}$$

is a pullback in $\mathcal{B}$.

The significance of the pullback requirement for prefix-preserving morphisms is that it eliminates the possibility of an event appearing in an "early" alternative of P without appearing in the corresponding "early" alternative of Q. That is, processes joined by a morphism should agree on the causal precedence of events that they have in common.

Prefix ordered processes and prefix preserving morphisms form a category which we will denote ($\mathcal{P}_k[\mathcal{B}]$).

Branching Time

The next observation is that prefix ordered processes allow for the presence of "useless" alternatives.

Definition 26 *Given a schedule category $\mathcal{B}$ having an initial object 0 and satisfying the condition*

$$s \in \mathrm{ob}(\mathcal{B}) \textit{there exists some morphism } s \to 0 \Longrightarrow s \textit{ is initial in } \mathcal{B}$$

a branching-time *process over $\mathcal{B}$ is a prefix ordered process P for which*

- *$P(p) = 0$ implies p initial in A_P*
- *$p \leq p' \in A_P$ implies that either $p = p'$ or the corresponding prefix map $P(p) \hookrightarrow P(p')$ is proper.*

$\mathcal{P}_{k\perp}[\mathcal{B}]$ denotes the full subcategory of $\mathcal{P}_k[\mathcal{B}]$ consisting of branching-time processes.

The intuitive content here is that in any ordered sequence of alternatives, each successive schedule should "make progress" in the sense of each one having additional events not present in its predecessor.

One might wish also to merge *every* pair of isomorphic schedules within a process, no matter how they are arrived at, thus requiring processes to be actual sets rather than the multisets thus far described. Unfortunately, for some choices of schedule category $\mathcal{B}$, it is possible for a pair of schedules to be isomorphic without being *canonically* isomorphic. Without further stipulations on the structure of a process, the only case where such a canonical isomorphism is available is that of the initial schedule, 0, so we can at least merge any initial alternatives that might exist (as we do above). The lack of canonical isomorphisms in general would defeat the construction of process limits as we shall discuss when we present that construction.

The condition imposed on $\mathcal{B}$ is true of all of the examples we have seen thus far ($\mathcal{D}$-**Cat**, $\mathcal{D}$-**Phys**, ($\mathcal{D}$-**Cat** $\downarrow$ **Set**), ($\mathcal{D}$-**Phys** $\downarrow$ **Set**), etc. . .). The conditions on a branching-time process P together imply that

- there is at most one object in A_P, which we will call $\perp$ if it exists, for which $P(\perp) = 0$.
- A_P is a partial order, for the second condition above enforces the antisymmetry axiom.

Finally, note that morphisms $f : P \to Q$ between such processes will automatically be strict, that is, $A_f(\perp_P) = \perp_Q$.

Closure

For some applications, it will be necessary to be able to infer a particular infinite schedule from the corresponding sequence of finite schedules that have it as a limit. The stipulations for the various types of process presented thus far do not require such infinite schedules to ever exist.

Definition 27 *Given a category $\mathcal{B}$ (of schedules) having all filtered colimits, a branching-time process P over $\mathcal{B}$ is* closed *iff for any directed subset S of A_P, there exists a least upper bound $\bigvee S$ in A_P satisfying $P(\bigvee S) = \mathrm{colim}_{s \in S} P(s)$. A morphism between closed processes $f : P \to Q$ is* continuous *iff A_f is continuous, i. e., preserves joins of directed subsets. We denote the category of closed branching-time processes and continuous morphisms as $\overline{\mathcal{P}_{k\perp}}[\mathcal{B}]$.*

Because of the naturality of $f : QA_f \dot{\to} P$, we have that for any directed set S in A_P, the map

$$f_{\bigvee S} : QA_f(\bigvee S) = \mathrm{colim}_{s \in S} QA_f(s) \to \mathrm{colim}_{s \in S} P(s)$$

is necessarily the usual canonical one.

Here, the word "closed" is being used in its topological sense, that of a closed set containing its limit points, rather than the categorical sense, that of possessing a right adjoint to the tensor product. The latter sense would be meaningless here, given that we do not even view a process P as a category, let alone have a definition of tensor product on P.

Labels

If we are considering processes over a category of labeled schedules (e. g., $\mathcal{B} = (\mathcal{D}\text{-}\mathbf{Cat} \downarrow \mathbf{Set})$), we have to contend with the orthogonal issue of the alphabet from which the process draws its labels. One generally expects that all of the constituent schedules of a given process will be over the same alphabet. While this can be dealt with simply by taking $\mathcal{B}$ to be $\mathcal{B}_\Sigma$, a category of labeled schedules over a single alphabet Σ, this does not allow for morphisms between processes over distinct alphabets as will often arise in the course of composing processes, i. e., one should not be compelled to specify a full system alphabet in advance.

Definition 28 *Given a category of labeled schedules $\mathcal{B}$, i. e., with a projection $\sigma : \mathcal{B} \to \mathbf{Set}$, a process over $\mathcal{B}$ is* alphabetically coherent *if it is also a process over $\mathcal{B}_\Sigma$ for some alphabet Σ. Given alphabetically coherent processes P over $\mathcal{B}_\Sigma$ and Q over $\mathcal{B}_\Gamma$, a process morphism (in $\mathcal{P}[B]$) $f : P \to Q$ is likewise* alphabetically coherent *if there exists a translation $t : \Gamma \to \Sigma$ such that, for all $p \in A_P$, $\sigma(f_p : QA_f(p) \to P(p)) = t$.*

$\mathcal{P}^\bullet[\mathcal{B}]$ denotes the subcategory of $\mathcal{P}[\mathcal{B}]$ consisting of alphabetically coherent processes and process morphisms.

As noted, this question of alphabetical coherence is orthogonal to the question of the particular alternative-set structure we wish to use. One may thus consider corresponding categories of alphabetically coherent processes, namely the categories of discrete, prefix ordered, branching-time or closed branching-time such processes, respectively $\mathcal{P}^{\bullet}_{d}[\mathcal{B}]$, $\mathcal{P}^{\bullet}_{k}[\mathcal{B}]$, $\mathcal{P}^{\bullet}_{k\perp}[\mathcal{B}]$, and $\overline{\mathcal{P}^{\bullet}_{k\perp}}[\mathcal{B}]$.

2.2 Properties

2.2.1 Pointwise Lifting of Schedule Functors

Pointwise lifting refers to the notion of taking a unary schedule operation and applying it uniformly to all of the constituent schedules of a given process. We need to establish what we mean by "uniformly" and present sufficient conditions on the schedule operation so that such a lifting will be possible. Essentially, if we are given a functor $F : \mathcal{B} \to \mathcal{B}'$ between two categories of schedules, we can apply this functor to all of the constituent schedules of a given process over $\mathcal{B}$ to yield a corresponding process over $\mathcal{B}'$.

Definition 29 *Given a functor $F : \mathcal{B} \to \mathcal{B}'$, the* lifted *functor $F_\# : \mathcal{P}[\mathcal{B}] \to \mathcal{P}[\mathcal{B}']$ is defined as follows: For any process $P \in \mathcal{P}[\mathcal{B}]$, $F_\# P$ is given by $A_{F_\# P} = A_P$ (i. e., same index set) and $F_\# P(: A_P \to \mathcal{B}') = FP$. For any morphism $f : P \to Q$, $F_\# f$ is given by $A_{F_\# f} = A_f$ (i. e., same index function) and $(F_\# f)_p(: FQA_f(p) \to FP(p)) = Ff_p$.*

Functoriality of $F_\#$ follows almost immediately from functoriality of F.

Similarly, natural transformations lift as well.

Definition 30 *Given a natural transformation $\eta : F \dot{\to} G : \mathcal{B} \to \mathcal{B}'$, the* lifted *natural transformation $\eta_\# : G_\# \dot{\to} F_\# : \mathcal{P}[\mathcal{B}] \to \mathcal{P}[\mathcal{B}']$ is defined by specifying the morphisms $\eta_{\# P} : G_\# P \to F_\# P$ as follows*

- $A_{\eta_{\# P}} = 1_{A_P}$, *recalling that $A_P = A_{G_\# P} = A_{F_\# P}$.*
- $(\eta_{\# P})_p = \eta_{P(p)}$

As one might expect, naturality of $\eta_\#$ follows almost immediately from naturality of η. Applying this lifting to the unit and counit transformations of an adjunction, we get

Proposition 31 *If $F : \mathcal{B} \to \mathcal{B}'$ is the left (right) adjoint of $G : \mathcal{B}' \to \mathcal{B}$, then $F_\# : \mathcal{P}[\mathcal{B}] \to \mathcal{P}[\mathcal{B}']$ is the right (left) adjoint of $G_\# : \mathcal{P}[\mathcal{B}'] \to \mathcal{P}[\mathcal{B}]$.*

Outline of Proof: If F is the left adjoint of G, we have unit $\eta : 1_\mathcal{B} \dot{\to} GF$ and counit $\epsilon : FG \dot{\to} 1_{\mathcal{B}'}$ natural transformations satisfying the usual triangular identities

$$\begin{aligned} \epsilon_{F\mathtt{X}} \circ F\eta_\mathtt{X} &= 1_{F\mathtt{X}} \\ G\epsilon_\mathtt{Y} \circ \nu_{G\mathtt{Y}} &= 1_{G\mathtt{Y}} \end{aligned}$$

These transformations lift to $\eta_\# : (GF)_\# \dot{\to} 1_{\mathcal{P}[\mathcal{B}]}$ and $\epsilon_\# : 1_{\mathcal{P}[\mathcal{B}']} \dot{\to} (FG)_\#$. Noting that $((GF)_\# = G_\# F_\#)$ and $((FG)_\# = F_\# G_\#)$, the triangular identities can likewise be shown to lift as well to

$$\begin{aligned} F_\#\eta_{\#X} \circ \epsilon_{\#F_\# X} &= 1_{F_\# X} \\ \nu_{\#G_\# Y} \circ G_\#\epsilon_{\#Y} &= 1_{G_\# Y} \end{aligned}$$

which suffices to demonstrate that $F_\#$ is a right adjoint of $G_\#$ with $\epsilon_\#$ and $\eta_\#$ being the unit and counit, respectively. ∎

In particular, for the case of $\mathcal{B}$ being a category of labeled schedules and for a given alphabet translation $t : \Gamma \to \Sigma$, we recall the corresponding translation and restriction schedule functors $t_* : \mathcal{B}_\Gamma \to \mathcal{B}_\Sigma$ and $t^* : \mathcal{B}_\Sigma \to \mathcal{B}_\Gamma$. We can then lift these to process functors $t_\# : \mathcal{P}[\mathcal{B}_\Gamma] \to \mathcal{P}[\mathcal{B}_\Sigma]$ and $t^\# : \mathcal{P}[\mathcal{B}_\Sigma] \to \mathcal{P}[\mathcal{B}_\Gamma]$ in the above manner. As a matter of simplifying the notation, it is understood that $t_\#$ and $t^\#$ are abbreviations for $t_{*\#}$ and $t^*_\#$ respectively. Given the adjunction $t_* \dashv t^*$, we can then infer a corresponding adjunction $t^\# \dashv t_\#$ from Proposition 31.

In the subcategory of prefix ordered processes, we need additional conditions on the supplied functor $F : \mathcal{B} \to \mathcal{B}'$ in order that the resulting lifted functor $F_\# : \mathcal{P}_k[\mathcal{B}] \to \mathcal{P}_k[\mathcal{B}']$ indeed maps into $\mathcal{P}_k[\mathcal{B}']$, i. e., that the constructions for $F_\# P$ and $F_\# f$ indeed produce a prefix ordered process and a prefix preserving morphism when given prefix ordered P and prefix preserving f. The following set of conditions is sufficient to guarantee this:

- F preserves pullbacks.
- If $k : 1 \to 2$ is the prefix classifier for $\mathcal{B}$, then $Fk : F1 \to F2$ is a prefix in $\mathcal{B}'$.

Likewise, a natural transformation $\eta : F \dot{\to} G : \mathcal{B} \to \mathcal{B}'$ of schedule functors will only lift if its naturality diagram

$$\begin{tikzcd} Fs \arrow[r, "\eta_s"] \arrow[d, hook] & Gs \arrow[d, hook] \\ Fs' \arrow[r, "\eta_{s'}"] & Gs' \end{tikzcd}$$

is a pullback in the case where the $s \to s'$ morphism is a prefix. This is required in order for $\eta_{\#P}$ to actually be a prefix preserving morphism for any $P \in \mathcal{P}_k[\mathcal{B}]$. Sufficient conditions to guarantee this are the aforementioned conditions on F together with the condition that

$$\begin{tikzcd} F1 \arrow[r, "\eta_s"] \arrow[d, hook, "Fk"'] & G1 \arrow[d, hook, "Gk"] \\ F2 \arrow[r, "\eta_{s'}"] & G2 \end{tikzcd}$$

be a pullback.

In the case of branching-time processes, though we don't need any additional conditions on F beyond that for the prefix ordered case, there is more to do to lift a functor $F : \mathcal{B} \to \mathcal{B}'$ to a corresponding *induced* functor $F_{\#\perp} : \mathcal{P}_{k\perp}[\mathcal{B}] \to \mathcal{P}_{k\perp}[\mathcal{B}']$. For any process $P \in \mathcal{P}_{k\perp}[\mathcal{B}]$, we can construct an equivalence relation R on A_P generated by the set of all pairs (p, p') in A_P for which

- $p \leq p'$ and $FP(p \to p')$ is an isomorphism, or
- $FP(p)$ and $FP(p')$ are both initial.

Then the lifted *branching time* process, which we shall also denote $F_{\#}P$, is then a process whose alternatives are given by the quotient category A_P/R. We then take a representative p of each R-equivalence class $[p]$ and have $F_{\#}P : A_P/R \to \mathcal{B}$ be the functor taking $[p]$ to $FP(p)$. This construction is not canonical, since for any particular equivalence class of alternatives, there may be a variety of choices for p, though the functoriality of $P : A_P \to \mathcal{B}$ guarantees a canonical isomorphism between any pair of such choices. Thus, all of the possible $F_{\#}P$'s we can construct will themselves be canonically isomorphic.

For any morphism $f : P \to Q$, showing that there exists a well-defined morphism $F_{\#}f : F_{\#}P \to F_{\#}Q$ is a matter of showing that A_f maps R-equivalent alternatives of A_P to equivalent alternatives of A_Q. We have already noted that A_f is strict which handles the case of alternatives equivalent by virtue of being initial. The case of noninitial $p \leq p'$ and $FP(p \to p')$ being an isomorphism, is covered by F preserving prefixes and prefix pullbacks, i. e., the pullback in the naturality of f diagram is preserved, thus guaranteeing that $FQ(A_f p \to A_f p')$ is also an isomorphism.

Lifting a natural transformation $\eta : F \dot{\to} G$, is similarly complicated but likewise involves no additional stipulations. In the general prefix ordered case, we could rely on $A_P = A_{G_{\#}P} = A_{F_{\#}P}$, and thus have $A_{\eta_{\#}P}$ be the identity. In the branching time case, various alternatives of $A_{F_{\#}P}$ and $A_{G_{\#}P}$ will be identified in which case we need to ensure that we can still define the map $A_{\eta_{\#}P} : A_{G_{\#}P} \to A_{F_{\#}P}$. The essential observation is that if two alternatives p and p' of A_P are identified in $A_{G_{\#}P}$, this happens in one of the following two ways:

- GPp and GPp' are both initial, in which case FPp and FPp' have to be as well,
- $p \leq p$ and $GPp \hookrightarrow GPp'$ is an isomorphism, but then, since

$$\begin{tikzcd} FPp \arrow[r, "\eta_{Pp}"] \arrow[d, hook] & GPp \arrow[d, hook] \\ FPp' \arrow[r, "\eta_{Pp'}"] & GPp' \end{tikzcd}$$

  is a pullback, we have $FPp \hookrightarrow FPp'$ being an isomorphism as well.

In each case, it is seen that the offending alternatives must also be identified in $A_{F_\# P}$. Therefore the identity on A_P factors through the quotient to yield the desired alternative-set map $A_{\eta_\# P} : (A_P/R_G = A_{G_\# P}) \to (A_P/R_F = A_{F_\# P})$

In the case of closed branching-time processes F lifts as for branching-time processes. Hence, we need only that F preserves filtered colimits to ensure that the $F_\#(P)$ will be closed.

2.2.2 Process Limits and Coproducts

We now turn to the question of limits and colimits in the process category, starting with a result about the general process category $\mathcal{P}[\mathcal{B}]$.

As a preliminary observation, notice that there is a forgetful functor $A : \mathcal{P}[\mathcal{B}] \to$ **Cat** that discards the schedule information, i. e., takes a process P to its alternative set (category, actually) A_P and a morphism $f : P \to Q$ likewise to its alternative set map $A_f : A_P \to A_Q$.

Lemma 32 *If $\mathcal{B}$ has a terminal object, $A : \mathcal{P}[\mathcal{B}] \to$ **Cat** has a left adjoint.*

Proof: Define $L : \mathbf{Cat} \to \mathcal{P}[\mathcal{B}]$ as follows: For any category C, the process L_C is that whose alternative category A_{L_C} is C and whose schedules are all terminal, i. e., $L_C c = 1_{\mathcal{B}}$ for all $c \in \mathrm{ob}(C)$ with morphisms of C going to the canonical terminal schedule maps.

That this is a left adjoint is shown by exhibiting, for any $C \in \mathbf{Cat}$, a universal arrow $C \to A_{L_C}(= C)$ which we take to be the identity. For any process $P \in \mathcal{P}[\mathcal{B}]$ and any functor $f : C \to A_P$, we take $g : L_C \to P$ to be such that $A_g = f$ and such that the associated schedule maps $g_c : Pfc \to L_C c(= 1)$ be the terminal map. Clearly, g is the only morphism for which $A_g \circ 1_C = f$. ∎

From this, we learn that the A functor preserves limits, i. e., the alternative category of a limit of processes will necessarily be the limit of the corresponding alternative categories in **Cat**.

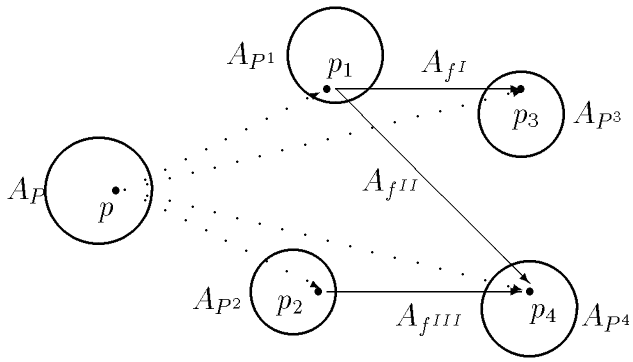
Theorem 33 *For any category $\mathcal{B}$ having all colimits of type J^{op}, $\mathcal{P}[\mathcal{B}]$ has all limits of type J.*

Proof: Consider a diagram J in $\mathcal{P}[\mathcal{B}]$ with processes $(P^i \mid i \in J)$ and morphisms $(f^m\colon P^i \to P^j \mid m\colon i \to j \in J)$. Just to provide names we will denote the limit as $(P\ ;\pi^i\colon P \to P^i \mid i \in J)$ (that is, P is the object and the π^i's are the projections).

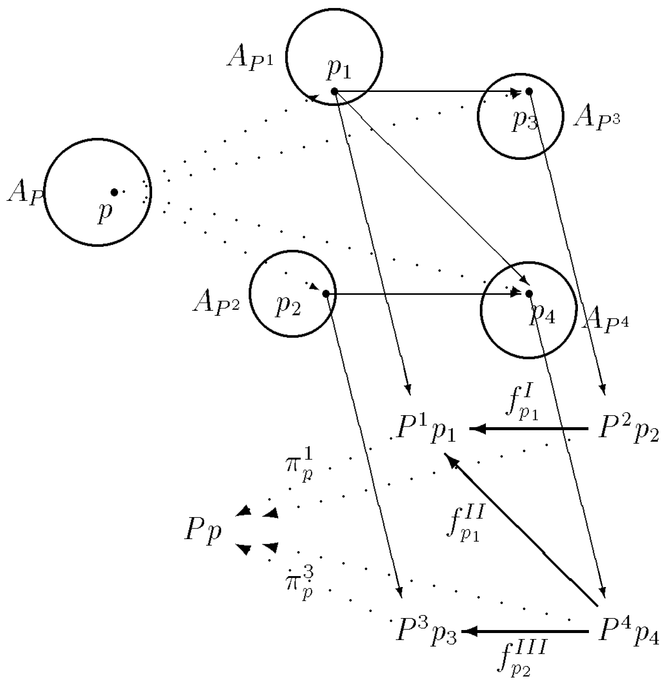
$$\begin{tikzcd}
 & P^1 \arrow[r, "f^I"] \arrow[ddr, "f^{II}"] & P^3 \\
P \arrow[ur, dotted, "\pi^1"] \arrow[urr, dotted, "\pi^3"] \arrow[dr, dotted, "\pi^2"'] \arrow[ddrr, phantom] \arrow[drr, dotted, "\pi^4"] & & \\
 & P^2 \arrow[r, "f^{III}"'] & P^4
\end{tikzcd}$$

From the lemma, we know that the alternative category A_P of the limit is obtained by taking the limit of the diagram $(A_{P^i}\ ; A_{f^m})$ in **Cat**. This limit in **Cat** also gives us a collection of projections which we can use for the alternative portion $(A_{\pi^i}\colon A_P \to A_{P^i})$ of the morphisms π^i.

As a consequence of the construction of limits in **Cat**, a typical object p of A_P will be a cone $(p_i\colon \{\bullet\} \to A_{P^i} \mid i \in J)$, or equivalently, a collection of objects $(p_i \in A_{P^i})_{i\in J}$ such that $p_j = A_{f^m} p_i$ for all $m\colon i \to j$ in J.



There will then be, for any such p, a corresponding collection of objects $P^i p_i$ in $\mathcal{B}$ and morphisms $f^m_{p_i}\colon P^j A_{f^m} p_i (= P^j p_j) \to P^i p_i$ for all $m\colon i \to j$ in J. This latter collection is, in fact, a diagram of type J^{op} in $\mathcal{B}$, which, by hypothesis, has a colimit $(Pp\ ; \pi^i_p\colon P^i p_i \to Pp)$. This provides the object map for the functor P as well as the definition for the natural transformation $\pi^i\colon P^i A_{\pi^i} \dot{\to} P$.



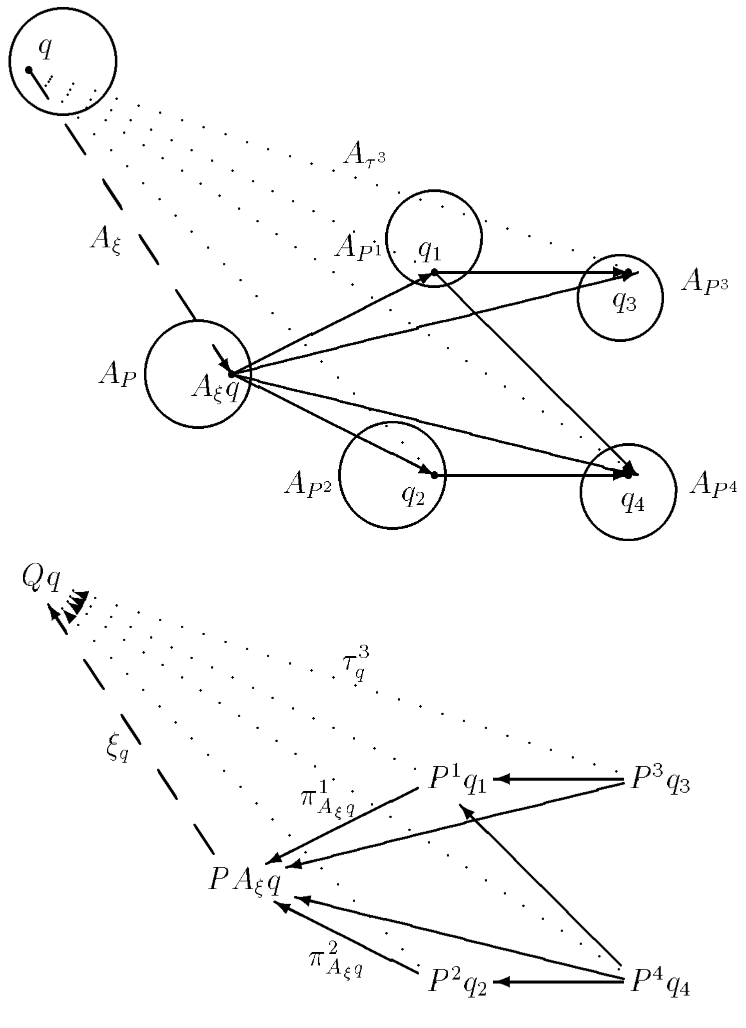
To get the action of P on morphisms, recall that a morphism $s\colon p \to p'$ in A_P will be a collection of maps $s_i\colon p_i \to p'_i$ in A_{P^i} such that $s_j = A_{f^m} s_i$ for all $m\colon i \to j$ in J. As with objects, we can consider the associated collection of morphisms $P^i s_i\colon P^i p_i \to P^i p'_i$ in $\mathcal{B}$. These constitute a natural transformation between the J^{op}-diagrams associated with p and p', since

$$\begin{array}{ccc}
P^j p_j = P^j A_{f^m} p_i & \xrightarrow{f^m_{p_i}} & P^i p_i \\
{\scriptstyle P^j s_j} \Big\downarrow {\scriptstyle = P^j A_{f^m} s_i} & & \Big\downarrow {\scriptstyle P^i s_i} \\
P^j p'_j = P^j A_{f^m} p'_i & \xrightarrow{f^m_{p'_i}} & P^i p'_i
\end{array}$$

commutes by naturality of f^m. Given any such transformation between the diagrams, there is a corresponding unique (canonical) morphism between the colimits $Ps\colon Pp \to Pp'$ making everything commute.

The reader may verify that $\pi^i\colon P^i A_{\pi^i} \dot{\to} P$ is a natural transformation.

Given some other cone $(Q\ ;\ \tau^i\colon Q \to P^i \mid i \in J)$, we now need to exhibit a unique morphism $\xi\colon Q \to P$ factoring this through the limit cone. We note that the limithood of $(A_P\ ;\ A_{\pi^i}\colon A_P \to A_{P^i} \mid i \in J)$ in **Cat** provides us with a unique map $A_\xi\colon A_Q \to A_P$ in **Cat**. Then, for every alternative $q \in A_Q$, there is a collection of alternatives $(q_i = A^i_\tau q \in P^i)_{i\in J}$ and then a corresponding schedule diagram with objects $(P^i q_i)_{i\in J}$ and morphisms $f^m_{q_i}\colon P^j q_j \to P^i q_i$ for all $m\colon i \to j$ in J.



This is a diagram we've already seen, namely the diagram under the single point cone $A_\xi q$ whose colimit is $(PA_\xi q\ ;\ \pi^i_{A_\xi q} P^i q_i \to PA_\xi q)$. Now Qq and the various τ^i_q provide us a co-cone over this diagram, so there is a unique map $\xi_q\colon PA_\xi q \to Qq$ factoring this co-cone through the colimit. The provision of these ξ_q for all alternatives

$q \in A_q$ completes the definition of $\xi: Q \to P$ and it should be clear that ξ is unique. ∎

Considering all possible diagrams J collectively, we obtain

Corollary 34 *$\mathcal{B}$ cocomplete implies $\mathcal{P}[\mathcal{B}]$ complete.*

Recall, however, that some of the schedule categories we use, $\mathcal{D}$-**Phys** in particular, are not necessarily cocomplete. Fortunately, we can obtain a stronger result applicable to schedule categories that satisfy a weaker condition than cocompleteness.

In the above proof, consider the case of a particular diagram of schedules for which there exists no co-cone at all. Suppose now that this very diagram arises in the course of a limit construction, say as the schedule diagram underlying some cone p of A_P. Then, for any process cone $(Q; \tau^i : Q \to P^i | i \in J)$, it will not be possible for any alternative q of A_Q to be mapped by the resulting A_ξ into the cone p, i. e., to have $A_{\tau^i} q = p_i$ for all $i \in J$. If it were possible, then there would be a corresponding co-cone down in the schedule category $\tau^i_q : P^i p^i \to Qq$ even though none exists for that particular diagram. Thus, if the troublesome alternative p were to be left out of A_P, it would still be the case that all cones Q over the process diagram would factor through P.

In the case of a category $\mathcal{B}$ which is *almost co-complete*, we need only revise our limit construction as follows. Construct the limit category A' over the diagram of $(A_{P^i}; A_{f^m})$ in **Cat**. Recall that the objects of A' are single-point cones over $(A_{P^i}; A_{f^m})$. Take A_P to be that subcategory of A' consisting only of those single-point cones $p = (p^i \in A_{P^i})_{i \in J}$ for which the $\mathcal{B}$-diagram $(P^i p^i; f^m_{p_i} : P^j p_j \to P^i p_i)$ has a co-cone and hence, by almost-cocompleteness, a colimit as well. Given this choice of A_P, for all $p \in A_P$ we take Pp to be the colimit schedule as before.

Note that in this case A_P is not necessarily the limit of the A_{P^i} in **Cat** but rather a subcategory thereof, contrary to what we inferred from Lemma 32. This is not a contradiction; Lemma 32 requires $\mathcal{B}$ to have a terminal object, which is not the case for, e. g., $\mathcal{B}$=$\mathcal{D}$-**Phys**. Indeed if $\mathcal{B}$ were to have a terminal object, then almost co-completeness would imply completeness since the terminal object provides *every* diagram with a cocone.

To summarize:

Theorem 35 *If $\mathcal{B}$ is almost J^{op}-cocomplete, $\mathcal{P}[\mathcal{B}]$ is J-complete.*

Since $\mathcal{B}$ invariably has an initial object, $\mathcal{P}[\mathcal{B}]$ likewise has a terminal object 1 having a single alternative $\perp$ mapping to the initial behavior. Essentially, this is the SKIP (or IDLE) process having a single behavior in which nothing happens.

In the situation where $\mathcal{B}$ a is category of labeled schedules, if we wish to construct a limit of alphabetically coherent processes in $\mathcal{P}^\bullet[\mathcal{B}]$, we can use the above construction provided the resulting limit process can be shown to be itself alphabetically coherent.

Corollary 36 *If $\mathcal{B}$ is almost J^{op}-cocomplete and $\sigma : \mathcal{B} \to$ **Set** preserves J-colimits, $\mathcal{P}^\bullet[\mathcal{B}]$ is J-complete.*

Outline of Proof: Given a diagram of type J in $\mathcal{P}[\mathcal{B}]$ all of whose objects and morphisms are coherent, the limit (together with all of the projections) will turn out to be coherent since every diagram of behaviors of which we have to take a colimit has the same underlying diagram of alphabets. The preservation of colimits by σ means that the alphabet of the colimit and the various injections into it are the same in all cases. ∎

It should be noted that, in the case of $\mathcal{B} = \mathbf{Pom}$, cod : $\mathbf{Pom} \to \mathbf{Set}$ does not preserve certain coequalizers [4] and thus fails the conditions of this theorem. One can indeed construct counterexamples in which a limit of pomset processes in $\mathcal{P}[\mathbf{Pom}]$ is not alphabetically coherent. These problems do not arise if we instead were to use $\mathbf{Pom}_<$, the category of pomsets with strict morphisms. Though $\mathbf{Pom}_<$ is not cocomplete like **Pom**, it is however *almost*-cocomplete. Furthermore, $\sigma : \mathbf{Pom}_< \to \mathbf{Set}$ preserves colimits in those cases where they actually exist, thus making $\mathcal{P}^\bullet[\mathbf{Pom}_<]$ complete.

Corollary 37 *If $\mathcal{B}$ is almost J^{op}-cocomplete,*

1. *$\mathcal{P}_d[\mathcal{B}]$ is J-complete.*

2. *$\mathcal{P}_k[\mathcal{B}]$ is J-complete.*

3. *$\mathcal{P}_{k\perp}[\mathcal{B}]$ is J-complete.*

4. *$\overline{\mathcal{P}_{k\perp}}[\mathcal{B}]$ is J-complete.*

Proof: (1) follows from the fact that a limit over a diagram of discrete categories is necessarily discrete, since the existence of a nontrivial morphism between single-point cones implies a nontrivial morphism somewhere in the diagram.

For (2) this follows immediately from prefixes preserving colimits (Theorem 24).

For (3), it suffices to demonstrate that the two conditions on branching-time processes hold for the limit process, provided they hold for all of the processes in the diagram. For this we use the notation from the proof of Theorem 33.

Since $s \to 0$ in $\mathcal{B}$ implies s initial, a noninitial object Pp^i in a colimit diagram guarantees the colimit schedule Pp to be noninitial via the injection $P^i\pi^i$. Thus there can be at most one initial alternative in the limit process P by virtue of there being at most one initial alternative in each of the processes P^i.

To establish the condition that P takes nontrivial alternative orderings to proper prefixes, we note that having the injection $\pi^i : P \to P^i$ be a prefix preserving morphism means that

$$\begin{tikzcd} P^ip^i \arrow[r, "\pi^i_p"] \arrow[d, hook] & Pp \arrow[d, hook] \\ P^i{p'}^i \arrow[r, "\pi^i_{p'}"] & Pp' \end{tikzcd}$$

is a pullback and thus if $P^ip^i \hookrightarrow P^i{p'}^i$ is a proper prefix for some $i \in J$, then so must $Pp \hookrightarrow Pp'$.

For (4), it suffices to recall that colimits commute with each other (see Mac Lane [13], ch. 9). ∎

Similar results for the coherent process categories follow in much the same way.

It will also be of interest to know when lifted schedule functors preserve process limits so we should also mention

Corollary 38 *Given any functor $F : \mathcal{B} \to \mathcal{B}'$ preserving colimits of type J^{op}, $F_\#$: $\mathcal{P}[\mathcal{B}] \to \mathcal{P}[\mathcal{B}']$ preserves limits of type J.*

This follows immediately from the construction of process limits. The corresponding results for the special cases also hold.

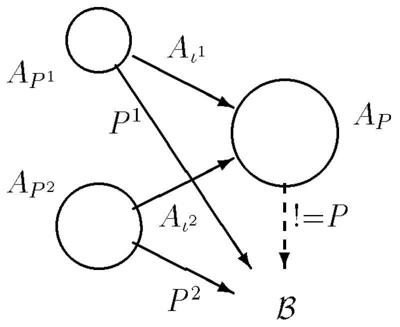
Corollary 39 *Given any functor $F : \mathcal{B} \to \mathcal{B}'$ preserving prefixes and colimits of type J^{op},*

- $F_{\#} : \mathcal{P}_k[\mathcal{B}] \to \mathcal{P}_k[\mathcal{B}']$ *preserves limits of type J.*
- $F_{\#} : \mathcal{P}_{k\perp}[\mathcal{B}] \to \mathcal{P}_{k\perp}[\mathcal{B}']$ *preserves limits of type J.*
- *if F preserves all filtered colimits, $F_{\#} : \overline{\mathcal{P}_{k\perp}}[\mathcal{B}] \to \overline{\mathcal{P}_{k\perp}}[\mathcal{B}']$ preserves limits of type J.*

We now consider the question of colimits in the process categories. In the case of coproducts, at least, the results are even simpler than for limits.

Theorem 40 *For* any *category $\mathcal{B}$, $\mathcal{P}[\mathcal{B}]$ has an initial object and all (small) coproducts.*

Proof: Given a J-indexed collection of processes $\{P^i \in \mathcal{P}[\mathcal{B}] : i \in J\}$, by the cocompleteness of **Cat**, we can construct $A_P = \coprod_{i \in J} A_{P^i}$ with injections $A_{\iota^i} : A_{P^i} \to A_P$. Since this is a coproduct in **Cat**, there exists a unique $P : A_P \to \mathcal{B}$ for which $PA_{\iota^i} = P^i$ which we take to be the coproduct process.



We then construct the process injections $\iota^i : P^i \to P$ using the alternative maps A_{ι^i} together with the identity natural transformation $PA_{\iota^i} = P^i$. That $(P, \iota^i)_{i \in J}$ has the required universal property is straightforward. ∎

Since A_P introduces no morphisms between alternatives in A_{P^i} and A_{P^j} for $i \neq j$, prefix closure of each of the P^i implies prefix closure of P. The morphisms ι^i are all prefix preserving in a trivial way.

In the case of branching time processes, we have to modify the construction in the case where more than one A_{P^i} contains an initial element $\perp$; these become identified in A_P. Since there is already a stipulation about the schedules that may be associated with $\perp$ by P^i if $P^i \in \mathcal{P}_{k\perp}[\mathcal{B}]$, i. e., only the initial schedule, the functor P will continue to be well-defined and serve as a colimit in $\mathcal{P}_{k\perp}[\mathcal{B}]$.

For $\mathcal{P}_{k\perp}[\mathcal{B}]$ we need to observe that a filtered diagram of objects in A_P must lie entirely within the image of one of the A_{ι^i}. Given this, the closure of P follows from the closure of the P^i.

These observations suffice to show

Corollary 41 *$\mathcal{P}_k[\mathcal{B}]$, $\mathcal{P}_{k\perp}[\mathcal{B}]$ and $\overline{\mathcal{P}_{k\perp}}[\mathcal{B}]$ have an initial object and all small coproducts.*

Notice that in order for coproducts to exist, there are no requirements on $\mathcal{B}$ aside from those we have already made for the sake of being able to construct $\mathcal{P}_k[\mathcal{B}]$, $\mathcal{P}_{k\perp}[\mathcal{B}]$ and $\overline{\mathcal{P}_{k\perp}}[\mathcal{B}]$ at all. The existence of coproducts relies, in some sense, strictly on the properties of **Cat**.

The initial process 0 is that for which the alternative set is empty. Essentially, this is the deadlocked STOP (or HALT) process that cannot do anything at all and thus is quite distinct from the SKIP process which is at least *able to do nothing*, i. e., in the sense of being able to exhibit an idle behavior.

Coequalizers in the process categories, where they exist at all do not, as far as we know, have any particularly pleasant characterization. It is fortunate that we have no particular use for them.

2.3 Process Operations

There is now sufficient machinery to describe some basic operations on processes analogous to those given for schedules. We have essentially three tools for defining operations on processes.

- Categorical limits and colimits, which we shall use to provide definitions for the concurrence and choice operations on processes.
- Pointwise lifting of schedule functors, which we have already seen for describing the lifting of alphabet renaming and restriction functors in the case of processes over labeled schedules. Pointwise lifting will also be used to describe process orthocurrence.
- Iteration, i. e., fixpoint constructions.

2.3.1 Choice

The union or *choice* operation is the one operation that is entirely new with the introduction of processes. Intuitively, the process $P + Q$ is that which may either behave like P or like Q. We see almost immediately that the coproduct in the process category essentially fills this role by taking the union of all of the alternative sets of the processes concerned without introducing any additional structure within the component processes, except in the case of branching-time processes ($\mathcal{P}_{k\perp}[\mathcal{B}]$ or $\overline{\mathcal{P}_{k\perp}}[\mathcal{B}]$), where we may need to merge initial alternatives.

Clearly the STOP or 0 process is an identity for the choice operation.

2.3.2 Concurrence

We recall for a moment our interpretation in which a process is a specification of event occurrences that may be satisfied by a given observation and a process morphism is a proof of implication between such specifications. Then, given a diagram of processes, an observation — which itself can be viewed as a particularly tight specification — that satisfies all of the specifications of the diagram in a consistent manner will essentially form a cone over that diagram. Having a single specification or process to represent the entire diagram is just the notion of having a limit over that diagram. Alternatively, the limit could be viewed as a most general observation that captures everything that is known in the diagram.

We saw, in the context of schedules that we are actually taking colimits, since the schedule morphisms run opposite to the direction of the implications they represent. *Process* morphisms, however, run parallel to the corresponding implications, so the corresponding notion of system composition is given by a *limit* rather than by a colimit.

Concurrence is the special case of system composition in which we compose a number of entirely independent processes, i. e., the diagram of processes being composed is discrete. The concurrence of processes P^1,P^2,...will then be their product $P^1 \times P^2 \times \ldots$, this being analogous to the situation of schedule concurrence as a coproduct.

Applying the limit construction to a discrete diagram of two processes P^1 and P^2, we first see that the alternative set $A_{P^1 \times P^2}$ is $A_{P^1} \times A_{P^2}$. This is intuitively sensible in that choosing an alternative of the combined process is merely a matter of independently choosing an alternative from each of the component processes. Given such a choice $p_1 \in A_{P^1}$, $p_2 \in A_{P^2}$, the corresponding schedule will be the schedule coproduct $P^1p_1 \| P^2p_2$; i. e., once we have picked a pair of alternatives, we form the concurrence of the schedules in the previous manner, so that every event of each schedule appears and no temporal contraints are introduced between events on distinct schedules.

As noted before we will, as a matter of clarity, denote the schedule concurrence using the previous notation $\|$ rather than $+$ for coproduct, in order that it not be confused with the process choice $(+)$ operation.

2.3.3 Orthocurrence

The process orthocurrence $P \otimes Q$ can be obtained via a pointwise lifting of schedule orthocurrences. That is, we take each possible choice of an alternative of P and an alternative of Q and construct the corresponding schedule orthocurrence.

However, this involves lifting a *binary* schedule operation and we have only thus far discussed how one lifts schedule operations defined as *unary* functors. If we can find some way to view schedule pairs as being schedules in their own right, then the desired construction follows automatically.

First of all, given a pair of schedule categories $\mathcal{B}$ and $\mathcal{B}'$, we have a canonical injection functor $\langle -, - \rangle : \mathcal{P}[\mathcal{B}] \times \mathcal{P}[\mathcal{B}'] \to \mathcal{P}[\mathcal{B} \times \mathcal{B}']$ Given a pair of processes $P \in \mathcal{P}[\mathcal{B}]$, $Q \in \mathcal{P}[\mathcal{B}']$, each of which we recall is actually a functor, we can pair these to obtain a functor, i. e., a process $\langle P, Q \rangle : A_P \times A_Q \to \mathcal{B} \times \mathcal{B}'$ in $\mathcal{P}[\mathcal{B} \times \mathcal{B}']$. Process morphisms pair in a similar manner.

For pairings of prefix-closed processes we need to consider the prefix classifiers k and k' for $\mathcal{B}$ and $\mathcal{B}$', respectively. In this case, we can use $\langle k, k' \rangle$ as a prefix classifier for $\mathcal{B} \times \mathcal{B}'$, i. e., a pair of maps defined to be a prefix map in $\mathcal{B} \times \mathcal{B}'$ iff both components are prefixes. Then the injection constructed above will take pairs of prefix-closed processes to prefix-closed processes and similarly for morphisms. Thus we also have a canonical injection $\langle -, - \rangle : \mathcal{P}_k[\mathcal{B}] \times \mathcal{P}_{k'}[\mathcal{B}'] \to \mathcal{P}_{\langle k, k' \rangle}[\mathcal{B} \times \mathcal{B}']$.

If P and Q are branching-time processes, then $\langle P, Q \rangle$ necessarily has only one initial alternative, seeing as the initial schedule of $\mathcal{B} \times \mathcal{B}'$ is $\langle 0_{\mathcal{B}}, 0_{\mathcal{B}'} \rangle$. Likewise a paired prefix will be proper iff at least one of its components is proper. These observations allow us to conclude that $\langle P, Q \rangle$ will be a branching-time process as well.

If P and Q are both closed branching-time processes, and $\mathcal{B}$, $\mathcal{B}'$ have all filtered colimits, then filtered colimits in $\mathcal{B} \times \mathcal{B}'$ will simply be pairs of colimits, $\langle P, Q \rangle$ is likewise a closed process, and again, a similar conclusion can be drawn for process morphisms.

Now we can recall the schedule orthocurrence functor $\otimes : \mathcal{B} \times \mathcal{B} \to \mathcal{B}$, and note that it satisfies the necessary properties for it to lift to a process functor. $\otimes_{\#} : \mathcal{P}[\mathcal{B} \times \mathcal{B}] \to \mathcal{P}[\mathcal{B}]$

Definition 42 *The* process orthocurrence *bifunctor* $\otimes$ *is the composition:*

$$\mathcal{P}[\mathcal{B}] \times \mathcal{P}[\mathcal{B}] \xrightarrow{\langle -, - \rangle} \mathcal{P}[\mathcal{B} \times \mathcal{B}] \xrightarrow{\otimes_{\#}} \mathcal{P}[\mathcal{B}]$$

.

It should be noted that the process orthocurrence bifunctor itself provides $\mathcal{P}[\mathcal{B}]$ with a monoidal structure, one for which the corresponding identity is the process I consisting of a single alternative whose schedule is $\mathtt{I}$, the orthocurrence identity in $\mathcal{B}$. A more descriptive name for I might be BANG or TICK since the single underlying schedule is effectively one which has a single indeterminate event.

2.3.4 Concatenation

For concatenation, recall the universal construction we used for schedules. The basic definition for concatenation on processes is essentially the same, albeit with reversed arrows and some careful consideration as to what the concatenation specifier should be.

In the following, $I_\perp$ is a two-alternative process in which the alternatives are ordered. The earlier one, $\perp$, has an initial schedule ($I_\perp(\perp) = \mathtt{0}_{\mathcal{B}}$) while the other alternative $\top$ points to the schedule $\mathtt{I}_{\mathcal{B}}$. In $\mathcal{P}_{k\perp}[\mathcal{B}]$ and $\overline{\mathcal{P}_{k\perp}}[\mathcal{B}]$, we have $I_\perp = I + 0$, but in the more general process categories, $I_\perp$ needs to be constructed explicitly to insert the $\perp < \top$ ordering in $A_{I_\perp}$.

A morphism $e : P \to I_\perp$ can be seen as picking out an "event" of P. In slightly more detail, each alternative p of A_P is taken by A_e either to $\perp$ or to $\top$. In the latter case we then have to choose a schedule map $I\top(= \mathtt{I}) \to Pp$, i. e., an actual event of Pp. Moreover these choices must be such that if we choose an event for a given alternative, we must choose the corresponding event from each later alternative. In the case of prefix-preserving maps, we must also choose the corresponding event from each earlier alternative as well. In short, $e : P \to I_\perp$ picks out an event as it threads its way through the various alternatives of the process.

Definition 43 *Given processes P, Q, and a map $d : 2 \to I_\perp \times I_\perp$ (a* concatenation specification*), a* pre-concatenation *$R = \langle R, c_R, \rho_R \rangle$ is*

- *a process R, together with*
- *a process map $c_R : R \to P \times Q$, and,*
- *for every pair of maps $(e : P \to I_\perp, e' : Q \to I_\perp)$, a particular schedule map $\rho_R(e, e') : R \to 2$ such that the following diagram commutes:*

$$\begin{tikzcd} R \arrow[r, "\rho_R(e,e')"] \arrow[d, "c_R"'] & 2 \arrow[d, "d"] \\ P \times Q \arrow[r, "e \times e'"] & I_\perp \times I_\perp \end{tikzcd}$$

The concatenation $P\,;Q$, *is a preconcatenation* $\langle P\,;Q, c, -\,;-\rangle$ *such that any preconcatenation* R *factors through it, i. e., there is a unique map* $R \to P\,;Q$ *making*

$$\begin{tikzcd} R \arrow[dr, "!"] \arrow[drr, "\rho_R(e,e')"] \arrow[ddr, "c_R"'] & & \\ & P\,;Q \arrow[r, "e\,;e'"'] \arrow[d, "c"] & 2 \arrow[d, "d"] \\ & P \times Q \arrow[r, "e \times e'"] & I_\perp \times I_\perp \end{tikzcd} \tag{2.1}$$

commute.

This definition has essentially the same content as in the schedule case: for each possible choice of an event from P and an event from Q we introduce an additional constraint as specified by d. It is only that the notion of picking an event from a process is complicated by there possibly being more than one schedule containing the event in question. As with schedule concatenation, having the concatenation factor through all of the preconcatenations ensures that the resulting structure is minimal in the sense of not introducing any unnecessary constraints or superfluous events. Again, as with schedule concatenation, we could express process concatenation in an equivalent manner as a single pullback

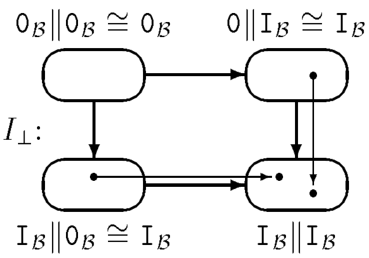
$$\begin{tikzcd} P\,;Q \arrow[r] \arrow[d, "c"'] & 2^{Hom(P+Q,I_\perp)} \arrow[d, "d^{Hom(P+Q,I_\perp)}"] \\ P \times Q \arrow[r] & (I_\perp \times I_\perp)^{Hom(P+Q,I_\perp)} \end{tikzcd} \tag{2.2}$$

in which the bottom arrow is the product over all possible choices of pairs of maps $(e : P \to I_\perp, e' : Q \to I_\perp)$. Though it is not clear this would be any more illuminating than the original definition, it does provide, via Theorem 33 and its corollaries, an assurance that the concatenation as defined always exists assuming $\mathcal{B}$ to be almost cocomplete.

This construction however will only yield a recognizable notion of concatenation if we make a sensible choice for $d : 2 \to I_\perp \times I_\perp$, much as the definition for schedule concatenation rests on the choice of the schedule concatenation specifier $\mathtt{d} : \mathtt{I}\|\mathtt{I} \to 2$. Thus we should devote some attention to the question of which map d we should actually use.

$I_\perp \times I_\perp$ is a process with four alternatives, which we may denote, for lack of better names, $\mathtt{OO}, \mathtt{OI}, \mathtt{IO}, \mathtt{II}$. Carrying through the product construction, the ordering on $A_{I_\perp \times I_\perp}$ is the usual product ordering and the corresponding schedule diagram is

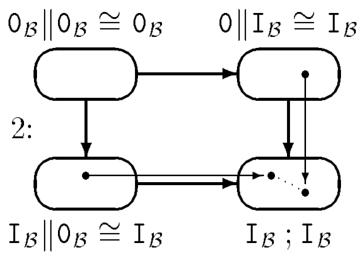
$I_\perp \times I_\perp$:



in which all of the maps are the canonical ones. If $\mathcal{B} = \mathcal{D}$-**Phys**, then the $\mathtt{I}$ schedules really are singleton event schedules and the underlying event maps will be as shown.

At first we shall take 2 to be just like $I_\perp \times I_\perp$ except that in the alternative $\mathtt{II}$, in which "both" events are present we introduce an extra temporal constraint between them. Otherwise the two processes are identical; in particular, we take A_d to be the identity map. As for "introducing the extra temporal constraint", this is done by taking the schedule $2(\mathtt{II})$ to be the schedule concatenation $\mathtt{I}_\mathcal{B} \,;\, \mathtt{I}_\mathcal{B}$ rather than the concurrence as above. The underlying schedules and schedule morphisms of 2 are thus

2:



(2.3)

In this diagram, the maps on the bottom and the right are those of the previous diagram composed with the canonical map $\mathtt{I}_{\mathcal{B}} \| \mathtt{I}_{\mathcal{B}} \to \mathtt{I}_{\mathcal{B}} ; \mathtt{I}_{\mathcal{B}}$ for schedule concatenation. We also use the latter for $d_{\mathtt{I}\mathtt{I}}$, the one nontrivial portion of the natural transformation underlying the *process* morphism $d : 2 \to I_\perp \times I_\perp$.

Proposition 44 *$A_c : A_{P\times Q} \to A_{P;Q}$ is an isomorphism. For any pair of alternatives $\langle p, q\rangle \in A_{P;Q}$, the corresponding schedule will be $Pp\,;Qq$. Likewise the schedule portions of the maps $c : P\,;Q \to P \times Q$ and $e\,;e' : P\,;Q \to 2$ are obtained from the canonical diagram for schedule concatenation*

$$\begin{array}{ccc} \mathtt{I}\|\mathtt{I} & \xrightarrow{e_p \| e'_q} & Pp\|Qq \\ {\scriptstyle \mathtt{d}}\big\downarrow & & \big\downarrow{\scriptstyle c_{\langle p,q\rangle}} \\ 2 & \xrightarrow{e\,;e'_{\langle p,q\rangle}} & Pp\,;Qq \end{array} \tag{2.4}$$

Outline of Proof: That A_c is an isomorphism follows immediately from consideration of the pullback construction of concatenation (2.2). There, we notice that the two alternative sets on the right are isomorphic. Since the alternative sets themselves form a corresponding pullback by the construction of Theorem 33, the two alternative sets on the left must be isomorphic as well.

$P\,;Q$ is indeed a preconcatenation, since assembling all of the schedule maps $c_{\langle p,q\rangle}$ and $e\,;e'_{\langle p,q\rangle}$ extracted from the diagrams 2.4 does produce actual process morphisms c and $e\,;e'$.

It then remains to be shown that any other preconcatenation R factors through it (as in 2.1) Having $A_c : A_{P;Q} \to A_{P\times Q}$ be an isomorphism makes determining the map $A_R \to A_{P;Q}$ a trivial matter. We are left with the task of producing the underlying schedule maps $Pp\,;Qq \to Rr$ for each alternative r of R (where, in each case, $\langle p, q\rangle = A_{c_R} r$). This is accomplished by showing that Rr is a *schedule* preconcatenation and hence the desired schedule maps $Pp\,;Qq \to Rr$ exist and are unique by virtue of $Pp\,;Qq$ being a schedule concatenation.

Showing that Rr is a *schedule* preconcatenation entails showing that for any pair of schedule maps ($\mathtt{e} : \mathtt{I} \to Pp, \mathtt{e}' : \mathtt{I} \to Qq$) we can find a map $\mathtt{e} \,;\mathtt{e}' : \mathtt{2} \to Rr$ making

$$\begin{tikzcd}
\mathtt{I}\|\mathtt{I} \arrow[r, "\mathtt{e}\|\mathtt{e}'"] \arrow[d, "\mathtt{d}"'] & Pp\|Qq \arrow[d, "c_{\langle p,q\rangle}"] \\
2 \arrow[r, "\mathtt{e}\,;\mathtt{e}'"] & Pp\,;Qq
\end{tikzcd}$$

commute.

Lemma 45 *Given a process P, some alternative p thereof, and any schedule map $\mathtt{e} : \mathtt{I} \to Pp$, there exists a process map $e : P \to I_\perp$ such that $e_p = \mathtt{e}$.*

Outline of Proof: Take $A_e(p') = \top$ if $p \le p'$, $\perp$ otherwise. Likewise, take $e_{p'} = P(p \le p') \circ \mathtt{e}$ if $p \le p'$ and the canonical initial map otherwise. ∎

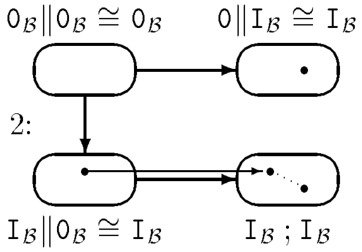
Outline of Proof of Proposition 44 (conclusion): Given schedule maps $\mathtt{e}$ and $\mathtt{e}'$, we have corresponding process maps e and e' from which we can take advantage of R being a process preconcatenation. This produces a process map $\rho(e, e') : R \to 2$ and then one has to check that the schedule map $\rho(e, e')_r : \mathtt{2} \to Rr$ is exactly the map we want for $\mathtt{e}\,;\mathtt{e}'$. ∎

This is the same notion of concatenation that appeared in the original pomset model, which in turn was inspired by that for trace theory. That is, $P\,;Q$ is the set of all possibilities involving a schedule of P concatenated with a schedule of Q.

One may now notice a problem in applying this to the case of prefix ordered processes: the map on the right side of diagram 2.3, i. e., the one mapping $\mathtt{0} \parallel \mathtt{I}_{\mathcal{B}} \to \mathtt{I}_{\mathcal{B}}\,;\mathtt{I}_{\mathcal{B}}$, is *not* a prefix map. That there is a problem should not actually be surprising since it is with the prefix ordering structure that we make any kind of connection between the alternative ordering and the temporal constraints. Consider that when we advance from an alternative p to a later alternative p', the new events of Pp' will necessarily be after the ones that have already appeared in Pp. But when forming the process concatenation $P\,;Q$, one would hope to capture the sense in which nothing happens in process Q until after P is "finished."

In our context, given alternatives $p < p'$ in A_P and $q < q'$ in A_Q, if we have an ordering $\langle p, q\rangle \le \langle p, q'\rangle$, then $\langle p, q'\rangle$ is a later stage of $A_{P;Q}$ at which we can arrive by "doing something" on Q. The fact that the Q portion of the process $P\,;Q$ is now active suggests that we should not then be able to have $\langle p, q'\rangle < \langle p', q'\rangle$, i. e., to be able to do something on P. In general, having $A_{P;Q}$ be isomorphic to $A_{P\times Q}$ turns out to be not so appropriate.

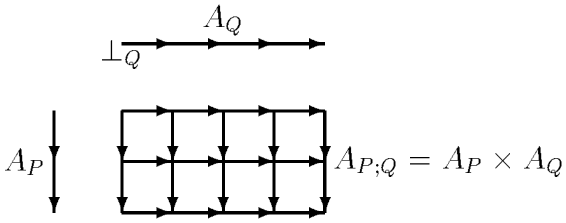
The immediate problem of $\mathtt{0} \parallel \mathtt{I}_{\mathcal{B}} \to \mathtt{I}_{\mathcal{B}}\,;\mathtt{I}_{\mathcal{B}}$ not being a prefix map is solved by using a different 2, one in which the troublesome ordering $\mathtt{0I} \le \mathtt{II}$ is simply not present in A_2. This can easily be done since we only need to have A_2 mapping into $A_{I_\perp \times I_\perp}$; they do not need to be isomorphic. Then we can take the underlying schedule maps to be



(2.5)

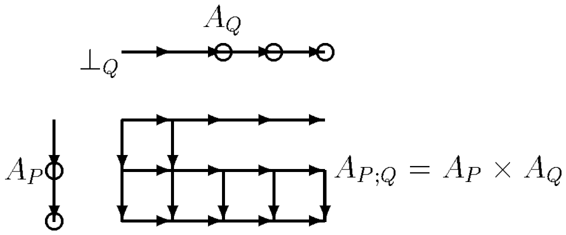
and carry through the previous concatenation construction as before.

Consider for example the concatenation of two processes P and Q each with a linear set of alternatives, the first alternative of each being initial (the empty schedule). Using the original version of 2 and ignoring the prefix ordering requirement, i. e., working in $\mathcal{P}[\mathcal{B}]$, the alternative set $A_{P;Q}$ of the concatenation is exactly like that of the concurrence $A_{P\times Q}$.



(2.6)

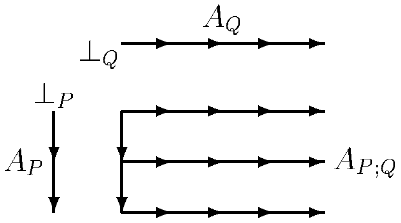
Using the revised 2 and working in the prefix ordered category $\mathcal{P}_k[\mathcal{B}]$, an argument similar to that of Proposition 44 shows that underlying sets of $A_{P;Q}$ and $A_{P\times Q}$ are the same, though we can no longer expect the ordering to be the same as well. The underlying object map of $A_{P;Q}$ and $A_{P\times Q}$ is thus fixed. That $P\,;Q$ is at least a preconcatenation requires that for every pair of maps $(e : P \to I_\perp, e' : Q \to I_\perp)$, we have a map $P\,;Q \to 2$, the monotonicity of which dictates that certain orderings not be present in $A_{P;Q}$. So for any particular (e, e') pair we can deduce the maximal order that $A_{P;Q}$ can have to allow for a map into A_2



(2.7)

in which we placed circles on those alternatives of A_P and A_Q that are sent by A_e (resp. $A_{e'}$) to the noninitial alternative $\mathtt{I}$ of $I_\perp$. Note that if $A_e p = \mathtt{I}$, then $A_e p' = \mathtt{I}$ for all $p' \geq p$ by functoriality.

Combining all of the constraints on the $A_{P;Q}$ ordering imposed by all of the possible choices for e and e', we obtain the actual ordering for $A_{P;Q}$



(2.8)

though we omit the proof that this alternative order together with the usual schedule concatenations constitutes the actual process $P\,;Q$ in $\mathcal{P}_k[\mathcal{B}]$. Notice that this ordering of $A_{P;Q}$ manages to omit exactly those pairs of alternatives for which the corresponding schedule map is *not* a prefix.

In the branching-time process category $\mathcal{P}_{k\perp}[\mathcal{B}]$, since we know that there exists a construction of concatenation as a limit, this limit will in fact be a branching-time process if its components are.

For the concatenation construction to be usable in the category of closed processes $\overline{\mathcal{P}_{k\perp}}[\mathcal{B}]$ it suffices that schedule concatenation, itself a colimit, commute with the taking of filtered colimits, but this follows from colimits commuting with colimits in general.

2.4 Iterated operations

Finally, we have to contend with the class of iterated operations. One such is the Kleene star or iterated concatenation, P^*, in which the alternatives are sequences of alternatives from P while the corresponding schedules are the corresponding schedule concatenations. Analogously, we also have the iterated concurrence $P^\dagger$ as introduced in [19]. These, among others, can be constructed by taking the fixpoint of a suitable process functor. In general, it will be useful to have criteria by which one may know that these fixpoints exist.

We start by stating a slight generalization of a result of Plotkin and Smythe [17]

Theorem 46 *Given two categories $\mathcal{C}$, $\mathcal{D}$, the latter having ω-colimits and an initial object 0, along with a functor $F : \mathcal{C} \times \mathcal{D} \to \mathcal{D}$ preserving ω-colimits, there exists an initial fixpoint $F^\omega : \mathcal{C} \to \mathcal{D}$. That is, we have*

- *a functor $F^\omega : \mathcal{C} \to \mathcal{D}$ and*
- *a canonical natural isomorphism $\tau : F\langle \mathrm{id}_{\mathcal{C}}, F^\omega\rangle \dot{\to} F^\omega$ that is initial in the category of pre-fixpoints of F, a pre-fixpoint being a pair (G, γ) in which $G : \mathcal{C} \to \mathcal{D}$ is a functor and $\gamma : F\langle \mathrm{id}\mathcal{C}, G\rangle \dot{\to} G$).*

Furthermore, if $\mathcal{C}$ has ω-colimits, then F^ω preserves them

Outline of Proof: We can fix an object A of $\mathcal{C}$, in which case the proof follows the same line as in [17], that is, we take the colimit of

$$0 \to F(A, 0) \to F(A, F(A, 0)) \to F(A, F(A, F(A, 0))) \to \cdots$$

to obtain $F^\omega A$ and a canonical isomorphism $\tau : FF^\omega A \to F^\omega A$. For any morphism, $f : A \to B$, we have a natural transformation between the respective diagrams on A and B which produces a morphism $F^\omega f : F^\omega A \to F^\omega B$.

The final statement is a consequence of the uniqueness of colimits in $\mathcal{D}$. ∎

Actually, we will be using the dual result:

Theorem 46 (dual) *Given two categories $\mathcal{C}$, $\mathcal{D}$, the latter having ω-limits and a terminal object 1, along with a functor $F : \mathcal{C} \times \mathcal{D} \to \mathcal{D}$ preserving ω-limits, there exists a fixpoint functor $F^\omega : \mathcal{C} \to \mathcal{D}$ with a canonical natural isomorphism $\tau : F^\omega \stackrel{\cdot}{\to} F\langle \mathrm{id}_\mathcal{C}, F^\omega\rangle$.*

Furthermore, if $\mathcal{C}$ has ω-limits, then F^ω preserves them

We use this to define the iterated concurrence operation $^\dagger : \mathcal{P}[\mathcal{B}] \to \mathcal{P}[\mathcal{B}]$ (see [20]) by taking the fixpoint of

$$\begin{array}{rcl} 1 + (\pi_1 \| \pi_2) : \mathcal{P}[\mathcal{B}] \times \mathcal{P}[\mathcal{B}] & \to & \mathcal{P}[\mathcal{B}] \\ (P, Q) & \mapsto & 1 + (P \| Q) \end{array}$$

That this functor preserves ω-limits is a matter of the functors $1 + -$ and $-\|-$ preserving ω-limits. By Theorem 46, we then have a functor

$$\begin{array}{rcl} ^\dagger : \mathcal{P}[\mathcal{B}] & \to & \mathcal{P}[\mathcal{B}] \\ P & \mapsto & P^\dagger \end{array}$$

along with a canonical morphism $P \to 1 + P \| P$ and the fact that † also preserves ω-limits.

In a similar manner, for a particular process P we can define the iterated concatenation P^* as the fixpoint of

$$1 + (P \,;\, \pi_2) : \mathcal{P}[\mathcal{B}] \to \mathcal{P}[\mathcal{B}]$$

which also provides a canonical morphism $P^* \to 1 + P \,;\, P^*$. Since the ; functor only preserves filtered colimits in its second argument, this is the best we can do. This construction is applicable in $\mathcal{P}_k[\mathcal{B}]$, $\mathcal{P}_{k\perp}[\mathcal{B}]$, and $\overline{\mathcal{P}_{k\perp}}[\mathcal{B}]$ as well.

However, in $\overline{\mathcal{P}_{k\perp}}[\mathcal{B}]$, unlike in the other cases we have the closure property. A closed process containing a particular ordered (a subcase of filtered) sequence of schedules must also contain the colimit schedule as an alternative. We have all finite concatenations of schedules from P as before, but now we also have all infinite concatenations of schedules from P as well.

Chapter 3

Example: Dataflow Networks

We now present an example representing a fairly well-known model, the dataflow model of Kahn [11]. The particular result that we derive has been obtained before, both by Rabinovich [22] for pomset semantics and by Gaifman and Pratt [6] for the dual-order prosset model. The difference here is that we use the limit semantics for system composition introduced above, and the temporal model is sufficiently abstract to allow for the use of arbitrary $\mathcal{D}$-**Phys** temporal structures (e.g., intervals).

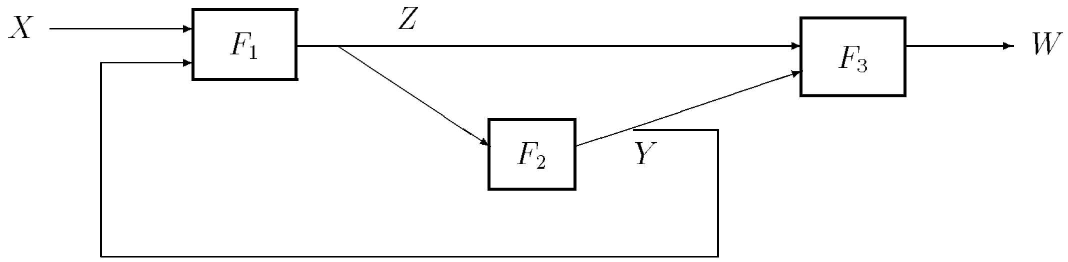
Informally, we consider a collection of *processors*, each of which has as its sole means of communication with the outside world a set of *ports*. Essentially the processor reacts to data items appearing on any of its input ports, possibly transmitting data items on one or more of its output ports.

A dataflow network consists of a set of processors together with a set of transmission *channels* linking them. One may distinguish between channels that are purely *input* channels, i.e., that connect only input ports and those channels which are driven by an output port on some processor. Data items that are broadcast on a given channel are then seen on the input ports to which that channel connects. Channels are sequential in the sense that only one data item may be broadcast at a time.

Taken collectively the network itself may be viewed as a processor in the previous sense. The set of input ports of the processor is the set of input channels of the network while the output ports are some designated set of channels driven by particular processors, the remaining channels becoming *internal* channels which are not seen by the outside world.

3.1 Kahn's Semantics

Kahn's result concerns a network of processors that "compute functions." This is formalized as follows. The channels carry data items drawn from some alphabet Σ, which may be different for different channels. We describe what happens on a particular channel during some computation via its history, which will be a (possibly infinite) string on the alphabet Σ, i. e., some element of $\Sigma^{\leq\omega}$, a cpo which we take to be ordered by prefix. The behavior of processor i is described by a monotone continuous (i. e., directed-join preserving) *function* relating the output string to the given input strings. For example, we could have a network of 3 processors



(3.1)

whose behaviors are described by the functions

$$\begin{aligned} F_1 : \Sigma_X^{\leq\omega} \times \Sigma_Y^{\leq\omega} &\to \Sigma_Z^{\leq\omega} \\ F_2 : \Sigma_Z^{\leq\omega} &\to \Sigma_Y^{\leq\omega} \\ F_3 : \Sigma_Y^{\leq\omega} \times \Sigma_Z^{\leq\omega} &\to \Sigma_W^{\leq\omega} \end{aligned}$$

Kahn's main result is that, by taking the least fixpoint of the system

$$\begin{aligned} Z &= F_1(X, Y) \\ Y &= F_2(Z) \\ W &= F_3(Y, Z) \end{aligned}$$

we get a function $\overline{F} : \Sigma_X^{\leq\omega} \to \Sigma_W^{\leq\omega}$ which describes the behavior of the network as a whole with respect to the input X and the output W.

3.2 Extending the Kahn Model

That the relation between the input histories and the output histories be a functional one for each of the processors is critical to the applicability of Kahn's theorem. Brock and Ackerman [2] discuss anomalies that occur when one considers networks with one or more processors having internal nondeterminism in their behavior.

One of the motivations of the original pomset model was to provide an extension of Kahn's model to treat situations with nondeterminism. One of the usual questions with such extensions is whether they are *conservative* in the sense of obtaining matching results in those situations in which the Kahn model is applicable. Rabinovich [22] and Gaifman and Pratt [6] answered this in the affirmative, given a particular representation of dataflow processors in terms of pomset processes and a particular notion of composition of processes.

Our goal here is to emulate the Gaifman-Pratt result, which in our terms, applies to processes constructed over the schedule category **3**-**Phys**, except that we will use our notion of composition of process specifications via categorical limit, rather than the *fusion* described in [6], and the result will apply to **D**-**Phys** for arbitrary choice of temporal domain **D**.

In our framework, we view each processor as a component and, moreover, we view each channel as a component as well. Thus, the above network becomes a diagram

$$\begin{tikzcd}
P_1 \arrow[d] \arrow[dr] \arrow[drrr] & & P_2 \arrow[dl] \arrow[dr] & & P_3 \arrow[dlll] \arrow[dl] \arrow[d] \\
\Sigma_{\overline{X}}^{\leq\omega} & \Sigma_{\overline{Y}}^{\leq\omega} & & \Sigma_{\overline{Z}}^{\leq\omega} & \Sigma_{\overline{W}}^{\leq\omega}
\end{tikzcd} \tag{3.2}$$

where each of the P_i's is a process that "computes the function F_i" in the sense to be described below. Our contention is that our notion of system composition via taking the limit of this diagram matches the least fixpoint semantics of Kahn, that is, taking the limit of this diagram yields a process P with morphisms $P \to \Sigma_{\overline{X}}^{\leq\omega}$ and $P \to \Sigma_{\overline{W}}^{\leq\omega}$ such that P computes the function $\overline{F}$.

3.2.1 Input Locations

For the purposes of this chapter we will be working within the category of closed branching-time coherent processes ($\overline{\mathcal{P}^{\bullet}_{k\perp}}[\mathcal{B}]$). In addition we need to be able to have temporal constraints forcing events to strictly precede other events, and to prohibit morphisms from identifying such events that are in strict precedence. Thus we need to have our underlying temporal domain be some $\mathcal{D}$-**Phys**.

Recall that we say an event x *strictly precedes* another event y (write $x < y$) if there exists no morphism $\delta(x, y) \to I$.

We first characterize the [6] notion of *input location* in our framework. Intuitively, a given component Q of P, i.e., where we have a morphism $f : P \to Q$, is an *input* if every alternative of Q has some alternative of P making use of it.

Definition 47 *A coherent process morphism $f : P \to Q$ is an* input *if*

- *A_Q has an initial element $\perp_Q$, $A_f^{-1}(\perp_Q)$ is nonempty (i.e., there exist alternatives of A_P mapped by A_f to $\perp_Q$).*

- *Given $p \in A_P$, let $q = A_f(p)$. If $q' \in A_Q$ is such that $q \leq q'$, there exists an extension $p' \in A_P$ (i.e., $p \leq p'$) such that $A_f(p') = q'$. Any such extension satisfies the following:*

  1. *every event in $P(p')$ not already in $P(p)$ or $f_{p'}Q(q')$ is strictly preceded by some event in $f_{p'}Q(q')$ not already in $f_pQ(q)$ (i.e., every new event of p' is strictly preceded by one of the new events of q'),*

  2. *if $Q(q)$ is empty (initial), $\delta(x, f_{p'}y) = 0$ for all $x \in P(p)$, $y \in Q(q')$ (i.e., the first input events can appear arbitrarily early with respect to the events of p).*

- *f is a restriction, i.e., $f = t^{\#}$ for some injective $t : \sigma Q \to \sigma P$.*

The essential idea is that the process P has a behavior to follow even if no input is received and any behavior involving a certain amount of input can be extended if we receive more input. The last clause ensures that P does not introduce stronger temporal constraints on the input events, other than those given by Q.

3.2.2 Computing a function

If we are to discuss how processes can compute functions, we first need to say how we are representing the values of these functions.

Definition 48 *A process Q in $\overline{\mathcal{P}^{\bullet}_{k\perp}}[\mathcal{B}]$* represents *a cpo $\mathbf{D}$ if A_Q is isomorphic to $\mathbf{D}$.*

Essentially, Q is a process describing one or more channels. Elements of $\mathbf{D}$ are then histories of these channels. The conditions on branching-time processes guarantee that when we advance from one alternative of $\mathbf{D}$ to the next, new events will appear.

Since we are dealing with labeled schedules, I has a terminal alphabet $\{\bullet\}$. For any alphabet Γ and any action $a \in \Gamma$, the translation $a : \{\bullet\} \to \Gamma$ lifts to a process translation which can then be applied to I to get the single-event process $a^{\#}I \in \mathcal{P}[\mathcal{B}_\Gamma]$. In fact the alphabet Γ itself can be constructed as a process

$$\coprod_{a\in\Gamma} a^{\#}I$$

obtained by taking the coproduct in $\mathcal{P}[\mathcal{B}_\Gamma]$. It should then be clear that

Proposition 49 *The process $\Gamma^{\leq\omega}$ represents the cpo $\Gamma^{\leq\omega}$.*

Proposition 50 *If $P, Q \in \overline{\mathcal{P}^{\bullet}_{k\perp}}[\mathcal{B}]$ represent $\mathbf{D}_P$ and $\mathbf{D}_Q$ respectively, $P \times Q$ represents $\mathbf{D}_P \times \mathbf{D}_Q$.*

For the purposes of constructing diagrams representing Kahn networks, we require that a given cpo $\mathbf{D}$ always be represented in the same way wherever it occurs, that is, all of the processes representing $\mathbf{D}$ in a particular diagram are canonically isomorphic. For this it suffices to take $\mathbf{D}$ to always be a product of $\Sigma^{\leq\omega}$'s.

Definition 51 *A process $P \in \overline{\mathcal{P}^{\bullet}_{k\perp}}[\mathcal{B}_\Sigma]$* computes a function *$F : \mathbf{D}_1 \times \ldots \times \mathbf{D}_n \to \mathbf{D}_0$ if there exist coherent morphisms $f_i : P \to Q_i$ $(i = 0 \ldots n)$, where Q_i represents $\mathbf{D}_i$, $A_{f_0\times\ldots\times f_n}(A_P) = \{(a_i)_i \in \mathbf{D}_0 \times \ldots \times \mathbf{D}_n | a_0 = F(a_1, \ldots, a_n)\}$, and $f_1 \times \ldots \times f_n$ is an input.*

In other words, a process computes a function if we can identify components of this process corresponding to the "input and output channels," such that

- every alternative behavior of the process corresponds to some allowed combination of input/output values, and
- all such allowed combinations have at least one behavior associated with them.

It should be noted that, while A_f is required to be strict for all process morphisms f in $\overline{\mathcal{P}^{\bullet}_{k\perp}}[,]$ this says nothing about the strictness of the function F computed by a given process P as above. [1]

3.2.3 Validity

Given the above definitions, we now discover that we have a valid extension of Kahn's semantics for data-flow networks, i. e., given a network of processors that compute functions, any corresponding diagram of processes (in the sense of the previous chapters) that compute the same functions will have a limit which will be a process that computes the least fixpoint.

One may construct any desired network through repeated operations of (1) juxtaposing processors and (2) connecting an output on a processor to one of its inputs. We treat the latter operation, demonstrating that for the case of a single process where we connect a single output channel to one of its inputs, the possible behaviors we obtain are just those obtained in Kahn's model.

Theorem 52 *If the process $P \in \overline{\mathcal{P}^{\bullet}_{k\perp}}[\mathcal{B}_\Sigma]$ computes a monotone, continuous function $F : \Gamma^{\leq\omega} \times \mathbf{D}' \to \Gamma^{\leq\omega}$, for which we have a process X representing $\mathbf{D}'$, and morphisms $f : P \to \Gamma^{\leq\omega}$, $g : P \to X$, $h : P \to \Gamma^{\leq\omega}$, where $f \times g$ is an input, then the equalizer of f and h, $\tau : R \to P$, is such that R computes $\overline{F} : \mathbf{D}' \to \Gamma^{\leq\omega}$, the least fixpoint of F.*

$$\begin{tikzcd}
 & & X \\
R \arrow[r, "\tau"] & P \arrow[ur, "g"] \arrow[r, shift left, "f"] \arrow[r, shift right, "h"'] & \Gamma^{\leq\omega}
\end{tikzcd}$$

[1]Nor does this even require that F be a function, but achieving that level of generality is well beyond the scope of this work.

Note that the equalizer τ is necessarily also a limit over the entire diagram of P,X, and $\Gamma^{\leq\omega}$.

Outline of Proof: We need to show that $A_{(h\times g)\tau}(A_R)$ is the graph of $\overline{F}$, and that $g\tau$ is an input.

By the construction of process limits in Theorems 33 and 35, A_R is a subset of the equalizer of A_f and A_h in **Set**, giving

$$A_{(h\times f\times g)\tau}(A_R) \subseteq \{(a,a,b) \in (\Gamma^{\leq\omega})^2 \times \mathbf{D}' | a = F(a,b)\}$$

The set inclusion can be proper since not all of the underlying behavior coequalizers need exist.

Now suppose we have $b \in \mathbf{D}'$ and $a < a' \in \Gamma^{\leq\omega}$ such that $a = F(a,b)$ and $a' = F(a',b)$. We can consider any alternative p of P such that $A_{f\times g}(p) = (a,b)$ and any extension p' for which $A_{f\times g}(p) = (a',b)$. Since F is functional we must have $A_h(p) = a$ and $A_h(p') = a'$. If h_p and f_p have a coequalizer then $h_{p'}$ and $f_{p'}$ cannot, for we can consider the earliest event x in $\Gamma^{\leq\omega}(a')$ not in $\Gamma^{\leq\omega}(a)$, guaranteed to exist by the branching-time conditions ($\Gamma^{\leq\omega}(a)$ is a proper prefix). Then neither of $h_{p'}x$ and $f_{p'}x$ can be in $P(p)$; they both lie in the extension. They must be distinct because their labelings are distinct. Condition 1 of f being an input implies that $f_{p'}x$ strictly precedes $h_{p'}x$, thus they cannot be identified as would be required in a coequalizer. In short, if an alternative of R corresponding to (a,b) exists, we cannot have an alternative corresponding to (a',b).

We are then reduced to showing that $A_{(h\times g)\tau}(A_R)$ contains $(\overline{F}(b),b)$ for all $b \in \mathbf{D}'$ and this is a matter of showing that given any b and $a = \overline{F}(b)$, there exists some $p \in A_P$ with $A_g p = b$, $A_f p = A_h p = a$, such that the co-equalizer of $f_p, h_p : \Gamma^{\leq\omega}(a) \to P(p)$ exists in ($\mathcal{D}$-**Phys**$\downarrow$**Set**)

Fix $b \in \mathbf{D}'$. We will inductively construct a sequence of alternatives of P whose sup is the desired p. Let $a_0 = \perp \in \Gamma^{\leq\omega}$ and let $p_0 \in A_P$ be such that $A_f \times A_g = (\perp, b)$. Given p_n such that $(A_f \times A_g)p_n = (a_n, b)$ and a_n has length n, if $F(a_n,b) = a_n$, we just let $p_m = p_n$ for all $m > n$. Otherwise, a_n is a prefix of $F(a_n,b)$ and we append the $(n+1)^{\text{th}}$ symbol of $F(a_n,b)$ to a_n to obtain a_{n+1}. p_{n+1} is then the extension of p_n for which $(A_f \times A_g)p_{n+1} = (a_{n+1},b)$ obtained via $f \times g$ being an input morphism. Note that the usual sequence one uses to construct $\overline{F}$ via induction (i.e., $F(\perp, b)$, $F(F(\perp, b), b)$, $F(F(F(\perp, b)))$, etc.) is a subsequence of the a_n's. Thus, $\bigvee a_n = \overline{F}(b) = a$. We then let $p = \bigvee p_n$ in A_P and by the continuity of f and h, $A_f p = A_h p = a$.

It now remains to show that the desired coequalizer exists. Let $r \to p$ be the coequalizer in $(\mathcal{D}\text{-}\mathbf{Cat}\downarrow\mathbf{Set})$; recall that this latter category is cocomplete. If r is actually in the subcategory $(\mathcal{D}\text{-}\mathbf{Phys}\downarrow\mathbf{Set})$, i.e., if its events satisfy the criterion $\delta(x, x) = I$, then we are done. Since the prefix of a coequalizer is itself a coequalizer, it suffices to inductively verify this criterion for all of the prefixes r_n of r, where $r_n \to p_n$ is the coequalizer of $f_{p_n}, h_{p_m} i_{nm} : \Gamma^{\leq\omega}(a_n) \to P(p_n)$ (where m is such that $F(a_n, b) = a_m$; i_{nm} is the prefix injection). That is, one verifies for all events x of r_n not already in r_{n-1}, that $\delta_r(x, x) = I$. Due to the way one constructs colimits in $\mathcal{D}$-**Cat** for $\mathcal{D}$ a preorder, this is not the case only if there is closed path in p_n of the form $y = y_0, y_1, \ldots, y_{2k+1} = y$ where $y_{2j-1} = \mathtt{f}_{p_n} z_j$ and $\mathtt{h}_{p_m} z_j = y_{2j}$ (z_j all in $\Gamma^{\leq\omega}(a_n)$) such that

$$\delta(y_0, y_1) \otimes \delta(y_2, y_3) \otimes \ldots \otimes \delta(y_{2k}, y_{2k+1}) > I$$

One then shows, for any event y in p_n not already in p_{n-1}, that such a loop does not exist, on the assumption that there are no such loops in p_{n-1}. The possibilities for loops in p_n are as follows. The case $n = 0$ is trivial since a_0 has no events. For $n = 1$, use condition 2 in the definition of input morphism. For $n > 1$, use condition 1, that successive events z, z' of $\Gamma^{\leq\omega}(a_n)$ have $\delta(z, z') = I$ and that f is a restriction (so that $\delta(\mathtt{f}_{p_n} z, \mathtt{f}_{p_n} z') = I$ as well).

Having shown that R has alternatives for each of the elements of the graph of $\overline{F}$, it only remains to show that $g\tau : R \to X$ is an input morphism. This can be left as an exercise as the arguments involved are not too dissimilar from what has gone before. ∎

To go from this to a general result is a matter of considering that for any given system of functional equations, the least fixpoint we obtain is the same, whether we compute it in stages progressively identifying one pair of variables at a time or identifying them all at once. Likewise, in any diagram of processes like (3.2), we obtain the same result whether we compute the limit all at once or substitute equalizers piecemeal.

Chapter 4

Directions for Future Work

Conjunctive Normal Form and Event Structures

It has already been noted that our process specifications are essentially presented in "disjunctive normal form". Can one devise any corresponding "conjunctive normal form"? This would be a model where the temporal relationships (rather than the choices) now form the outer structure, the disjunctive aspects being handled by some substructures where the events used to be. We consider a somewhat looser sense of CNF, taking a particular transformation one may perform on any prefix-ordered process P to obtain a form of annotated schedule.

The first observation is that the constituent behaviors, together with the morphisms relating them, form a diagram of type A_P. In the case of a prefix ordered process, prefix ordering guarantees that this diagram has a colimit.

Now as a matter of terminology, for each event e of the colimit and each of the alternatives a in A_P, we say that e is *present* in a iff the image of the schedule $P(a)$ under the injection into the colimit schedule E_P contains e. The subset of all alternatives of A_P in which e is present we denote $\phi_P(e)$. Since the process is prefix ordered, an event present in a will also be present in all "later" alteratives a', i. e., all alternatives a' for which $a \leq a'$; thus $\phi_P(e)$ is necessarily a *filter* (upward-closed subset) of A_P.

Essentially, we obtain a schedule E_P together with an annotation $\phi_P : E_P \to \mathcal{F}(E)$ of the schedule that indicates which subsets of events may collectively occur in any given scenario. In the case of a partial order temporal structure, this looks very much like one of Winskel's *prime event structures* [25].

Definition 53 *A prime event structure is a partially ordered set* $(E, \leq)$ *together with a* consistency *predicate* ζ *on finite subsets of* E *satisfying*

$$\begin{aligned} & \{e'|e' \leq e\} \text{ is finite} && (4.1)\\ & \{e\} \in \zeta && (4.2)\\ Y \subseteq X \wedge X \in \zeta \implies & Y \in \zeta && (4.3)\\ X \in \zeta \wedge \exists e' \in X : e \leq e' \implies & X \cup \{e\} \in \zeta && (4.4) \end{aligned}$$

for all $e, e' \in E$ *and all finite subsets* X, Y *of* E.

If one is willing to ignore the property (4.1), one may crudely derive an "event structure" from a prefix-closed process P in $\mathcal{P}_k[\mathbf{Pos}_<]$ by constructing the event set E_P via a colimit as described above and then defining ζ_P via

$$X \in \zeta_P \iff \bigcap_{e \in X} \phi(e) \neq \emptyset \tag{4.5}$$

that is, a finite subset $X \subseteq E$ of events is *consistent* iff there exists some alternative a in which they are all present. This reduction, however, loses information since, without any further conditions on the structure of A_P, there is, in general, no way to rederive the individual schedules $P(a)$ comprising the colimit schedule E_P. Also from Winskel:

Definition 54 *Given a prime event structure* $(E, \leq, \zeta)$*, a subset of* E *is* consistent *iff all of its finite subsets are in* ζ*. A subset* D *is* left-closed *iff* $e' \leq e \wedge e \in D \implies e' \in D$ *for all* e, e' *in* E.

The set of all left-closed, consistent subsets of E we denote $\mathbf{L}(E)$, which can in turn be ordered by inclusion. Clearly, $\mathbf{L}(E)$ will include all of the original schedules[1] comprising P, but it will also include all *prefixes* of those schedules as well. There may also be some orderings in $\mathbf{L}(E)$ that were not present in the original A_P.

[1]Actually, their images under the colimit injection into E

One way out is to consider more restrictions on the structure of the process P, i. e., to somehow stipulate that for every alternative a of A_P, and every possible distinct prefix $\mathtt{p}'$ of the schedule $P(a)$, there exists a unique alternative $a' \leq a$ such that $P(a') = \mathtt{p}'$. This is effectively what Rabinovich and Trakhtenbrot do in their work on *configuration structures* [23].

To formalize this in our framework, we first take note of a correspondence Winskel derives between prime event structures and *finitary prime algebraic domains*. First we need a rapid review of some domain theory terminology:

Definition 55 *Given a partial order* $(D, \sqsubseteq)$,

- *A subset* $D' \subseteq D$ *is* finitely compatible *iff each of its finite subsets* X *has a least upper bound* $\bigsqcup X$ *in* D.
- D *is* consistently complete *iff all of its finitely compatible subsets have least upper bounds.*
- *An element* $f \in D$ *is* finite *iff for all directed subsets* S *of* D, $f \sqsubseteq \bigsqcup S \Longrightarrow f \sqsubseteq s$ *for some* $s \in S$.
- D *is* algebraic *iff for every element* $d \in D$, *we have* $d = \bigsqcup\{f \sqsubseteq d | f \text{ is finite}\}$.
- *For any element* $d \in D$, *the* left closure, $\lfloor d \rfloor$, *is the set* $\{e | e \sqsubseteq d\}$.
- D *is* finitary *iff* $\lfloor f \rfloor$ *is finite for all finite* f
- *An element* p *is a* complete prime *of* D *iff for any subset* D' *with a least upper bound,* $p \sqsubseteq \bigsqcup D' \Longrightarrow \exists d \in D' : p \sqsubseteq d$.
- D *is* prime algebraic *iff for every element* $d \in D$, *we have* $d = \bigsqcup\{p \sqsubseteq d | p \text{ is a complete prime}\}$.

Now, for any finitary prime algebraic domain $(D, \sqsubseteq)$, we can take the set of its primes $\mathbf{Pr}(D)$ ordered by $\sqsubseteq$, and define a consistency predicate $\zeta_{\mathbf{Pr}(D)}$ that contains all sets of primes having a least upper bound, at which point Winskel then proves (see [25])

Theorem 56

- *For any finitary prime algebraic domain $(D, \sqsubseteq)$, we have that $(\mathbf{Pr}(D), \sqsubseteq, \zeta_{\mathbf{Pr}(D)})$ is a prime event structure, and $(\mathbf{LPr}(D), \subseteq)$ and $(D, \sqsubseteq)$ are isomorphic as partial orders.*
- *For any prime event structure $(E, \leq, \zeta)$, $(\mathbf{L}(E), \subseteq)$ is a finitary prime algebraic domain whose complete primes are the subsets $\lfloor e \rfloor$ for all $e \in E$ (and so $\mathbf{PrL}(E)$ and E are likewise isomorphic).*

The essential point is that we can stipulate a class of processes $\mathcal{P}_{pc}[\mathbf{2}\text{-}\mathbf{Cat}]$ which we call *prefix-closed* in which we require the alternative set A_P to be a finitary prime algebraic domain, and for all alternatives $a \in A_P$ either

- a is a complete prime and $P(a)$ contains exactly one event e that is not in the image of any other $P(a')$ and is maximal in $P(a)$.
- a is not a complete prime and all events in $P(a)$ are present in previous alternatives.

This class of processes, a subclass of branching-time processes is so severely restricted that building the colimit of the schedules, tossing away the alternative set and remembering only the consistency predicate loses no information; one can reconstruct the original process. Unfortunately, the restrictions are so severe that we cannot take advantage of any flexibility in choice of temporal domain $\mathcal{D}$; all of the schedules perforce look like partial order schedules.

Clearly, the more annotation we keep on our events when converting a process to an annotated schedule, the fewer requirements that need to be placed on the structure of the process in order that this conversion be bijective. The question of what annotation suffices, e.g., for branching-time or closed branching-time processes is an open one. Presumably this would yield some form of "live" event structure that would be able to express the notion of more than one event being forced to occur at a given stage, thus facilitating the discussion of liveness properties.

One area for possible improvement includes finding less restrictive notions of prefix. The current prefix notion for $\mathbf{R}$-$\mathbf{Phys}$ stipulates that if there is any upper bound at all from event x to event y and x is part of a prefix, so is y. This actually makes sense from the point of view that if alternatives are merely ordered, one can say nothing about the timing of decisions. If y is to be not in the prefix but part of the next stage, then y must be allowed to occur an arbitrarily long time after x since the decision to advance to the next stage may be indefinitely postponed. This suggests instead exploring some form of further enrichment for the alternative set, one which would not only provide orderings between alternatives, but also give time limits on the decisions between them.

Bibliography

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

[2] J.D. Brock and W.B. Ackerman. An anomaly in the specifications of nondeterministic packet systems. Technical Report Computation Structures Group Note CSG-33, MIT Lab. for Computer Science, November 1977.

[3] Ross Casley. *On the Specification of Concurrent Systems*. PhD thesis, Stanford University, January 1991.

[4] R.T Casley, R.F. Crew, J. Meseguer, and V.R. Pratt. Temporal structures. In *Proc. Conf. on Category Theory and Computer Science, LNCS 389*, Manchester, September 1989. Springer-Verlag. Revised version to appear in Math. Structures in Comp. Sci., **1**:2, 1991.

[5] R.W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[6] H. Gaifman and V.R. Pratt. Partial order models of concurrency and the computation of functions. In *Proc. 2nd Annual IEEE Symp. on Logic in Computer Science*, pages 72–85, Ithaca, NY, June 1987.

[7] J. Grabowski. On partial languages. *Fundamenta Informaticae*, IV.2:427–498, 1981.

[8] I. Greif. *Semantics of Communicating Parallel Processes*. PhD thesis, Project MAC report TR-154, MIT, 1975.

[9] P.T. Johnstone. *Stone Spaces.* Cambridge University Press, 1982.

[10] P.T. Johnstone. Fibred categories. Lecture notes, University of Cambridge, 1983.

[11] G. Kahn. The semantics of a simple language for parallel programming. In *Proc. IFIP Congress 74*. North-Holland, Amsterdam, 1974.

[12] G.M. Kelly. *Basic Concepts of Enriched Category Theory: London Math. Soc. Lecture Notes.* 64. Cambridge University Press, 1982.

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

[14] U. Montanari and C. Simonelli. On distinguishing between concurrency and nondeterminism. In *Proc. Ecole de Printemps on Concurrency and Petri Nets*, Colleville, 1980.

[15] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures, and domains, part I. *Theoretical Computer Science*, 13, 1981.

[16] S.S. Pinter and P. Wolper. A temporal logic to reason about partially ordered computations. In *Proc. 3rd ACM Symp. on Principles of Distributed Computing*, pages 28–37, Vancouver, August 1984.

[17] G.D. Plotkin and M.B. Smyth. The category-theoretic solution of recursive domain equations. Technical Report Research Report No. 60, D.A.I., 1978.

[18] V.R. Pratt. On the composition of processes. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Programming Languages*, January 1982.

[19] V.R. Pratt. The pomset model of parallel processes: Unifying the temporal and the spatial. In *Proc. CMU/SERC Workshop on Analysis of Concurrency, LNCS 197*, pages 180–196, Pittsburgh, 1984. Springer-Verlag.

[20] V.R. Pratt. Modeling concurrency with partial orders. *Int. J. of Parallel Programming*, 15(1):33–71, February 1986.

[21] V.R. Pratt. Enriched categories and the Floyd-Warshall connection. In *Proc. First International Conference on Algebraic Methodology and Software Technology*, pages 177–180, Iowa City, May 1989.

[22] A. Rabinovich. Pomset semantics is consistent with dataflow semantics. Technical report, Tel-Aviv University, 1986.

[23] A. Rabinovich and B.A. Trakhtenbrot. Behavior structures and nets. *Fundamenta Informatica*, 11(4):357–404, 1988.

[24] S. Warshall. A theorem on Boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962.

[25] G. Winskel. Event structures: Lecture notes for the advanced course on Petri nets. Technical report, University of Cambridge, July 1986.

[26] G. Winskel. A category of labelled Petri nets and compositional proof system. In *Proc. 3rd Annual Symposium on Logic in Computer Science*, Edinburgh, 1988. Computer Society Press.