1. Motivation

Goal: Define “concurrent process”

Uses (or, why do semantics?)

- Denotational semantics for operational models and realizations of concurrency.
- Concurrent programming language design:
  - Refine existing approaches and operations,
  - Suggest new ones.
- Compiler verification.
- Reconcile documentation (manual) with implementation.
2. Requirements

- Expressive denotational semantics for Petri nets, UML, MSC, ...
- Cater for nondeterminism
- Definability of termination and runtime.
- Useful family of definable operations.
The organization of LPA parallels both linear algebra and point set topology in certain fundamental ways.

Its algebra parallels that of linear logic: the definable operations include all the additive, multiplicative, and exponential operations.

These give two reasons for the name "linear process algebra."
4. Process \((A, X)\)

A set \(A\) of **events** \(a, a', \ldots\) together with a set \(X\) of possible configurations or **states** \(x, x', \ldots\) of those events.

Each event can be in any of four event states or **scalars**:

- **ready** 0,
- **ongoing** \(\triangleright\),
- **done** 1,
- **cancelled** \(\times\)

Denote \(\{0, \triangleright, 1, \times\}\) by \(K\), structured as the following (upward) directed reflexive graph (\(3 + 4 = 7\) edges).

\[
K = \begin{align*}
0 & \rightarrow \triangleright \\
\triangleright & \rightarrow 1 \\
1 & \rightarrow \times
\end{align*}
\]
5. Analogies

<table>
<thead>
<tr>
<th>Process graph $K$</th>
<th>Language $\subseteq \Sigma^n$</th>
<th>Topological space</th>
<th>Vector space</th>
</tr>
</thead>
<tbody>
<tr>
<td>event state</td>
<td>alphabet $\Sigma$</td>
<td>Sierpinski space</td>
<td>field $\mathbb{R}$ etc.</td>
</tr>
<tr>
<td></td>
<td>position word</td>
<td>point</td>
<td>vector</td>
</tr>
<tr>
<td></td>
<td></td>
<td>open set</td>
<td>functional</td>
</tr>
</tbody>
</table>

Difference: Only linear algebra has scalar multiplication.
6. Process maps

Map \( f : (A, X) \rightarrow (B, Y) \):

- a function \( f : A \rightarrow B \)
- s.t. \( \forall y : B \rightarrow K \) in \( Y \) (all states of the target)
- the inverse image \( yf = A \xrightarrow{f} B \xrightarrow{y} K \) of \( y \) by \( f \) is in \( X \).

Analogies: Language homomorphisms, continuous functions, and linear transformations are all definable in this way.
A state can be viewed as an $A$-dimensional vector over $K$, having coordinates $x_a = x(a)$. Orient it to make it a column vector of height $A$.

All vectors in $X$ share the same index set. Hence all the states can be placed side by side to create an $A \times X$ matrix $P$ with entries $P_{ax} = x(a) = x_a$. 
8. Examples

Sequence $a.b$

Choice $a + b$
9. Concurrence vs mutex

Concurrence $a \parallel b$

mutex($a, b$)
10. Late vs early branching

Late branching $a(b + c)$

<table>
<thead>
<tr>
<th>a</th>
<th>011111</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0001x</td>
</tr>
<tr>
<td>c</td>
<td>000x1</td>
</tr>
</tbody>
</table>

Early branching $ab + ac$

<table>
<thead>
<tr>
<th>a</th>
<th>011111</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0001xxxx</td>
</tr>
<tr>
<td>c</td>
<td>0xxxx01</td>
</tr>
</tbody>
</table>

Diagram:

- Late branching:
  - $a$:
    - 011111
  - $b$:
    - 0001x
  - $c$:
    - 000x1

- Early branching:
  - $a$:
    - 011111
  - $b$:
    - 0001xxxx
  - $c$:
    - 0xxxx01
Motivation: mutex(a, b) should take longer than a||b.

Step: a pair (x, y) of states of X such that ∀a ∈ A, (xa, ya) is an edge of K.

Run: a finite or infinite sequence x0, x1, . . . of states every consecutive pair of which is a step.

Time of a run: one less than the number of steps in it.

Examples. a||b takes time 1 while mutex(a, b) takes time 2.
12. Higher dimensional automata

Introduced by P in POPL’91.

A state can be understood as a face of an $A$-dimensional cube, whose dimension is the number of ongoing events in that state.

$\text{mutex}(a, b)$: a square with the interior missing: $3^2 - 1 = 8$ states.

$(3, 2)$ mutex: Three children each allowed to ride one of two ponies once around the track. Process is a 3D cube with $3^3 - 1 = 26$ states.
13. Distinguished states

**Initial state**: All events are ready (zero vector).

Write $x_0$ for the (necessarily unique) initial state when it exists.

**Disposed-of event**: either done or cancelled.

**Final state**: all events disposed of.

Write $X_F$ for the set of final states of $(A, X)$.

**Parfinal state**: a state $x$ for which there exists $x' \in X$ such that $(x, x')$ is a step. (Optional stopping point, corresponding to final states in automata theory.)
14. Types of processes

**Initialized**: Initial state $x_0$ exists.

**Connected**: $\forall x \in X$ there exists a finite run $x_0, x_1, \ldots, x$.

**Nonblocking**: Every ongoing event is permitted to complete. Formally, for every state $x \in X$ in which some event $a$ is ongoing ($x_a = \top$) there exists a state $y$ with $y_a = 1$ and having a run from $x$ to $y$.

**Prefix closed**: All states parfinal. (So no obligation need be fulfilled.)
15. Operations

Concurrence \( (A, X) \parallel (B, Y) = (A + B, X \times Y) \)

\((x, y) \in X \times Y\) maps \(a \in A + B\) to \(x(a)\) and \(b \in A + B\) to \(y(b)\).

Application: noninteracting concurrency (''parallel play'').

Sequence \((A, X). (B, Y) = (A + B, X \times \{y_0\} \cup X_F \times Y)\).

Meaning: As for concurrence but restricted to states in which either \((B, Y)\) has not yet started or \((A, X)\) has terminated. Note: no attempt to segue.

Choice \((A, X) + (B, Y) = (A + B, \{(x_0, y_0)\} \cup X' \times \{x\} \cup \{x\} \times Y')\)

where \(X'\) denotes \(X\) less \(x_0\) and likewise for \(Y'\).

Meaning: An initial state, together with states of \(X\) with \(B\) cancelled, and states of \(Y\) with \(A\) cancelled.
Orthocurrence 

$\text{Orthocurrence } (A, X) \otimes (B, Y) = (A \times B, \mathcal{F}) \text{ where } \mathcal{F} \text{ is the set of all states } x : A \times B \to K \text{ such that } x(-, b) : A \to K \text{ is in } X \text{ and } x(a, -) : B \to K \text{ is in } Y.$

That is, all $A \times B$ matrices whose rows are states of $B$ and whose columns are states of $A$.

Example:
Trains $T$ then $t$, stations $S$ then $s$.

$$
\begin{array}{c|cccccccc}
T & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
$$

Applications:
Flow of one process through another.
Communication via flow of data through a channel.
Conventional approach: An algebraic structure is a set, more generally a topological space, equipped with certain operations and satisfying certain equations, and transforming via continuous operation-preserving functions called (continuous) homomorphisms.

This approach: An algebraic structure with topology is an object of a dense extensional pointed category.

This is simpler than it sounds because functors and natural transformations are not mentioned, only categories themselves.
(… as distinct from category theory)

Approach: via the notion of free category $G^*$ on a graph $G$. 
19. Free categories

**Directed multigraph**: a graph $G = (V, E)$ permitting multiple edges between two vertices. Graph for short.

**Path** in $G$: a finite sequence $p$ of consecutive edges of $G$.
Write $E^*$ for the set of paths in $G$.
$E^*$ contains one empty path for each vertex.

**Identity** $i : V \rightarrow E^*$: $i(v)$ is the empty path at $V$.
i(v) is the identity for concatenation where defined.

**Concatenation**: a partial binary operation defined on two paths iff they are consecutive.
Concatenation is associative: $p(qr) = (pq)r$.

**Free category** $G^*$ on a graph $G$: The algebra of paths in $G$ under the operation of *converse* of concatenation.

Terminology: **object** = vertex, **morphism** = path.
Parallel: running between the same pair of vertices. (Applicable to both edges and paths.)

(Directed graph: a graph whose parallel edges are equal.)

Path congruence: an equivalence relation on parallel paths compatible with concatenation.

Category: the quotient of a free graph by a congruence.
A process is **rigid** when it has only one self-map (namely the identity).

A1. There exist two rigid processes $1$ and $K$ with four scalar maps $0, \top, 1, \times$ from $1$ to $K$.

An **event** of $P$ is a map from $1$ to $P$.

A **state** of $P$ is a map from $P$ to $K$.

Scalars are therefore simultaneously states (of $1$) and events (of $K$).
It is convenient (but not necessary) to use the language of set theory. Write $A_P$ and $X_P$ for the sets of respectively events and states of $P$.

Let $h : P \to Q$ be a map.

Define the **left action** of $h$ to be the function $\hat{h} : A_P \to A_Q$ defined by $\hat{h}(a) = ha$.

Dually the **right action** of $h$ is the function $\check{h} : X_Q \to X_P$ defined by $\check{h}(y) = yh$.

Call two parallel maps equivalent when they have the same left and right actions.

A2. (Extensionality) Equivalent maps are equal.

(A2 can be phrased without reference to sets thus. If for all events $a$ of $P$ and states $y$ of $Q$, $fa = ga$ and $yf = yg$, then $f = g$.)
Two functions $f : A_P \rightarrow A_Q$, $g : X_Q \rightarrow X_P$ are said to form an **adjoint pair** from $P$ to $Q$ when for all events $a$ of $P$ and states $y$ of $Q$, $yf(a) = g(y)a$.

**Proposition.** The left and right actions of a process map form an adjoint pair.

**Proof.**

\[
y\hat{h}(a) = y(ha) \quad \text{(definition)}
\]
\[
= (yh)a \quad \text{(associativity)}
\]
\[
= \hat{h}(y)a \quad \text{(definition)}
\]
24. Density

Concept of new entity: one that is adjoined to the ambient category.

An ordinary map is one that is neither an event nor a state.

A3. (No new maps) Any new ordinary map is equivalent to an old one.

Proposition (density). For any two processes $P, Q$ for which $P$ is not 1 and $Q$ is not $K$, every adjoint pair of functions from $P$ to $Q$ is the pair of actions of some map $g : P \rightarrow Q$.

A1-A3 define what it means to be a category of processes and their maps.
A1-A3 did not require that every process exist, in fact they are satisfied when 1 and $K$ are the only processes and the scalars are the only maps.

A weak requirement would be that whatever operations have been defined are total. A4 is a stronger requirement.

A4. (Completeness) Every new object is isomorphic to an old one.
26. Linear algebra

Defined in the same way as LPA, except that A1 is modified as follows.

A1’. There exists an object $K$, with a set $k$ of scalar maps from $K$ to itself forming a field.

Unlike the scalar maps of LPA (and of topology), those of linear algebra are composable. Composition denotes multiplication in the field (more generally any ring).

Addition is defined on parallel vectors, and is abelian. This is neatly achieved by assuming that homsets are abelian groups instead of sets.

A2-A4 work the same way. This gives one justification for referring to this process algebra as linear. The other is that the operations of concurrence $P \parallel Q$ and orthocurrence $P \otimes Q$ correspond to linear logic’s operations of sum $P \oplus Q$ and tensor $P \otimes Q$.

Orthocurrence as a process operation was introduced by P in 1983.