Investigation of concurrent processes by means of homotopy functors*

Stefan Sokołowski†

Manhattan, 30 August 1999

Contents

*This research has been made possible by the support of the Institute of Computer Science PAS, of the State Committee for Scientific Research in Poland (grant 8 T11C 037 16) and of the Dept. of Computing and Information Sciences KSU.

†PERMANENT ADDRESS: Polish Academy of Sciences, Institute of Computer Science, Gdańsk Division, ul. Abrahama 18, 81-825 Sopot, Poland, e-mail: `stefan@ipipan.gda.pl`. ADDRESS UNTIL MAY 2000: Kansas State University, Dept. of Computing and Information Sciences, 212 Nichols Hall, Manhattan, KS 66506, USA, e-mail: `stefan@cis.ksu.edu`.

0 Introduction

In theories of job scheduling and of distributed computing, there have been many attempts to introduce tools originating from algebraic and combinatorial topology. To mention just a few, Herlihy and Shavit [11] have reduced the task solvability problem to the existence of a certain mapping between simplicial complexes, which represent the task's input and the task's output. Herlihy and Rajsbaum [9] have used chain homotopies and acyclic carrier theorem to derive some lower bounds and impossibility results for distributed computing. Hoest and Shavit [12] have studied asynchronous complexity by the way of chromatic subdivisions of complexes. Tutorial papers in this area are [10] by Herlihy and Rajsbaum, and [8] by Herlihy.

In another line of research, Gunawardena [7] has given an intuitively appealing proof, based on investigations of homotopy properties of certain simple topological spaces, that "two phase locking is safe" — an important theorem on database consistency. Fajstrup and Raussen have proposed an efficient deadlock detecting algorithm based on geometrical considerations — see [4].

Probably the most deeply "algebraicized" of all is the contribution by Goubault and Jensen [6] where the geometric view of concurrency (cf. Pratt [14] and van Glabbeek [5]) is provided with a complete construction of homology groups in arbitrary dimensions.

0.1 How algebra helps topologists

Algebraic topology is about a translation of difficult topological problems to easier algebraic ones. One of them is the classification problem: given two topological spaces, are they topologically identical (homeomorphic)? To prove they are, produce a homeomorphism. To prove they are not, calculate their homotopy groups (or homology groups, or cohomology rings — there exist many usable tools) and prove they are not the same. The translation from topology to algebra abstracts from secondary details and the resulting algebraic problems stand a good chance to be easier solvable. The key point is that homeomorphic spaces are assigned identical algebraic structures.

The conceptually simplest algebraic tool for the investigation of topological spaces is the one-dimensional homotopy group, so called fundamental group. Its elements are circular paths (continuous images of circle with orientation) identified up to continuous deformations (homotopy). The group operation is derived from path composition: *first* follow one path, *then* the other. Higher dimensional homotopy groups are defined likewise, with higher dimensional spheres replaced for circle. More sophisticated tools, e.g. homology groups, carry less information about topological spaces but are easier to construct and to compare effectively.

Informally, the fundamental group gives an account of the nature of two-dimensional "holes" in a topological space. For instance, a solid square has no holes, thus its fundamental group is trivial (one-element). In contrast, a circular ring has one hole, thus its fundamental group is *Int* (integer numbers). These groups are not isomorphic, hence solid square and circular ring are not topologically equivalent. Of course, this is a very simple example, also solvable by more primitive "pre-algebraic" invariants, such as Euler numbers. The Euler number of a solid square is 1 and that of a circular ring is 0, thus the spaces are not topologically equivalent.

The application of algebraic methods to solving subtler topological problems is mostly based on the fact, that the translation from topology to algebra is functorial. This means, not only algebraic structures are assigned to topological spaces but also homomorphisms between these structures are assigned to respective continuous mappings between the spaces. A classical proof, that every continuous transformation of a solid disk into itself has a fixed point, goes by a translation into group theory. The problem is reduced to the question, whether one can go via a group homomorphism from *Int* to the one-element group and then back to *Int* so that the composition is the identity on *Int*. The negative answer is obvious.

0.2 How a similar approach might help computer scientists

In the realm of processes, "holes" in the spaces may correspond to forbidden configurations; e.g., where several processes are within the same critical region. This simple observation is reason enough for attempting to use similar techniques in concurrency.

However, many topological properties, technically necessary for the construction of the fundamental group, have no counterparts, or only artificial ones, in concurrent processes.

The paths corresponding to process executions are not cyclic, because time only flows forward, and they cannot be as naturally composed as the looping curves in topological spaces. The fundamental "group" lacks therefore its group operation. In [3], Fajstrup, Goubault and Raussen argue that directed homotopy, i.e., homotopy respecting the direction of time flow, is a more appropriate notion for concurrency investigations than the traditional one; but this does not lead to setting up any counterpart of the fundamental group.

The situation is better in homologies, since their group structure is derived from combinatorial properties of respective simplicial complexes rather than from process executions. Homology groups for task solvability problems are therefore defined (Herlihy and Shavit [11], Herlihy and Rajsbaum [10], Herlihy [8], Goubault and Jensen [6], etc.). But still, in most considerations important for computer science, either the conclusions are drawn from the combinatorial arguments that precede the construction of these groups, or the homologies are used only marginally, the only issue of importance being whether or not they are all trivial, i.e. whether or not the corresponding complex is acyclic. Goubault and Jensen [6] seems to be an exception but even they deal with drastically simplified "modulo 2" homologies in order to factor out the direction of time flow. To sum up this critical paragraph: my feeling is that the payout from homotopy and homology considerations for computer scientists has been so far more limited than the payout from such considerations for topologists.

0.3 The contribution of this paper

This paper deals with homotopy rather than homology. It puts forward a simple remedy for the shortcomings: if you cannot find a useful group operation on your homotopy classes, settle for a less requiring structure, e.g. *homotopy cpo*[1]. The ordering in this cpo reflects the direction of time flow and the homotopy considered is a discrete version of the dihomotopy from [3]. As will be shown, the transition from vectors of processes to their respective cpo-s is functorial, preserves information about the "holes" and abstracts from inessential details. This renders the approach a potentially useful tool for investigating admissible runs of concurrent processes.

The functoriality referred to above seems a novelty in computer science applications of algebraic topology. People have assigned algebraic structures to processes but, to the best of my knowledge, nobody has ever translated maps between processes to homomorphisms of respective structures. That non-functorial approach allows for a classification of processes the way Euler numbers classify manifolds, but it comes short of a subtler analysis of dependencies between processes. As will be shown, the functoriality of homotopy cpo-s may be applied to the problem of implementability of processes using other processes.

Acknowledgement.

This report is a thoroughly revised and updated [15].

[1]The notion of cpo, chain-wise complete partial order, is explained below in Sec. 1.2, page 5.

1 Preliminary notions

Auxiliary terminology and notations are introduced here.

1.1 Set theory

Given sets X and Y, a function $f : X \to Y$, a subset $A \subseteq X$ and a subset $B \subseteq Y$, denote the *image of* A *via* f by $f^{\to}A$ and the *coimage of* B *via* f by $f^{\leftarrow}B$:

$$f^{\to}A \stackrel{\text{def}}{=} \{fx \in Y \mid x \in A\} \quad \text{and} \quad f^{\leftarrow}B \stackrel{\text{def}}{=} \{x \in X \mid fx \in B\}$$

The *restriction* of function f to subset A will be denoted by $f|_A$:

$$\begin{cases} f|_A : A \to Y \\ f|_A\, x \stackrel{\text{def}}{=} f\, x \end{cases}$$

The *generalized Cartesian product* of a family $\{X_i\}_{i \in I}$ of sets is denoted by $\prod_{i\in I} X_i$. It consists of functions f over the index set I such that $f i \in X_i$ for all $i \in I$. In the special case, when $I = \{0, 1\}$, the product is denoted by $X_0 \times X_1$ and its elements by $\langle x_0, x_1\rangle$ with $x_0 \in X_0$ and $x_1 \in X_1$.

The set of all subsets of a certain X will be denoted by $\mathcal{P}(X)$.

1.2 Chain-complete partial orders

Cpo, i.e. *chain-complete partial order*, is any triple $\mathcal{X} = \langle X, 0, \sqsubseteq\rangle$ where

- $\langle X, \sqsubseteq\rangle$ is a partial order closed upon least upper bounds (*lub*s) of countable ascending *chains*:

$$x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \ldots \;\Rightarrow\; \text{there exists} \bigsqcup_{n\in Nat} x_n$$

- $0 \in X$ is the bottom element: $\forall_{x\in X}\; 0 \sqsubseteq x$.

Closure of a subset $Y \subseteq X$ in a certain cpo $\mathcal{X} = \langle X, 0, \sqsubseteq\rangle$ is the set of lubs of its chains:

$$\mathrm{cl}_{\mathcal{X}}\, Y \stackrel{\text{def}}{=} \{\textstyle\bigsqcup_{n\in Nat} y_n \mid (\forall_{n\in Nat}\; y_n \in Y)\ \&\ y_0 \sqsubseteq y_1 \sqsubseteq y_2 \sqsubseteq \ldots\}$$

Subset Y is *dense in* $\mathcal{X}$ whenever $\mathrm{cl}_{\mathcal{X}}\, Y = X$.

Downcone of a subset $Y \subseteq X$ in a certain cpo $\mathcal{X} = \langle X, 0, \sqsubseteq\rangle$ is the set of elements smaller than some elements of Y:

$$\downarrow_{\mathcal{X}} Y \stackrel{\text{def}}{=} \{x \in X \mid \exists_{y\in Y}\; x \sqsubseteq y\}$$

The $\mathcal{X}$ in $\downarrow_{\mathcal{X}} Y$ will be omitted whenever there arises no doubt as to the cpo.

Assume $\mathcal{X} = \langle X, 0, \sqsubseteq\rangle$ and $\mathcal{X}' = \langle X', 0', \sqsubseteq'\rangle$ are cpo-s. A function $f : X \to X'$ is *continuous* iff it is monotone and preserves the bottom and the limits:

- $x_1 \sqsubseteq x_2 \Rightarrow f x_1 \sqsubseteq' f x_2$,
- $f0 = 0'$,
- $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \ldots \;\Rightarrow\; f(\bigsqcup_{n \in Nat} x_n) = \bigsqcup'_{n \in Nat} f x_n$.

Proposition 1 (cpo-s form a category)
Chain complete partial orders and continuous mappings form a category.

The category referred to in Prop. 1 is denoted by $\underline{\underline{Cpo}}$.

Cpo $\mathcal{X}$ is called *straight* iff any two elements, which have a common upper bound, are comparable:

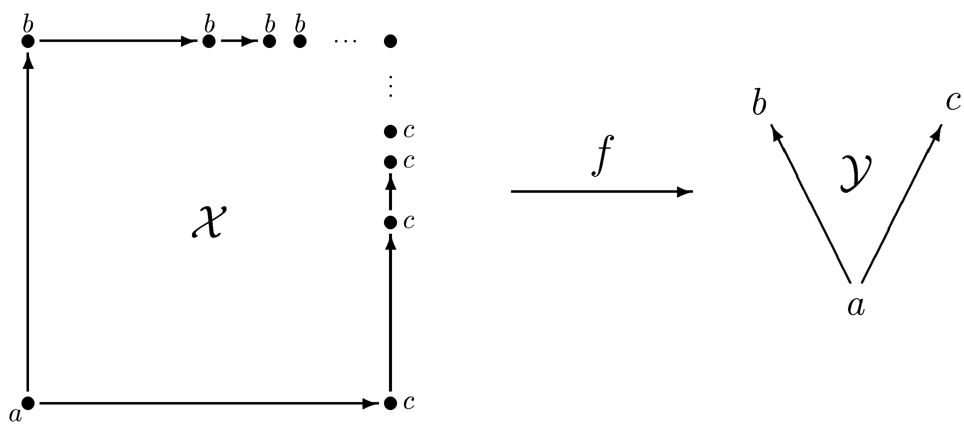
$$\forall_{x,y,z \in X}\ x \sqsubseteq z \ \&\ y \sqsubseteq z \Rightarrow x \sqsubseteq y \vee y \sqsubseteq x$$

Theorem 1 (on continuous extensions)
Assume $\mathcal{X} = \langle X, 0, \sqsubseteq \rangle$ and $\mathcal{Y} = \langle Y, 0, \sqsubseteq \rangle$ are cpo-s, $\mathcal{X}$ is straight and a subset X_0 is dense in X. Then every monotone function $f : X_0 \to Y$ has a unique continuous extension $\bar{f} : X \to Y$ (such that $\bar{f}|_{X_0} = f$).

Thm. 1 is not necessarily true if $\mathcal{X}$ fails to be straight. For a counterexample, take the set

$$X \stackrel{\text{def}}{=} \{\langle 0,0 \rangle\} \cup \left\{ \left\langle 1, \tfrac{n}{n+1} \right\rangle \,\middle|\, n \in Nat \right\} \cup \left\{ \left\langle \tfrac{n}{n+1}, 1 \right\rangle \,\middle|\, n \in Nat \right\} \cup \{\langle 1,1 \rangle\}$$

with the coordinate-wise ordering and with a dense subset $X_0 \stackrel{\text{def}}{=} X \setminus \{\langle 1,1 \rangle\}$ (i.e., X without the upper right point):



(small letters a, b and c in $\mathcal{X}$ refer to the points in $\mathcal{Y}$, to which function f maps respective points.

1.3 Words

For any set A (called *alphabet*), A^* denotes the set of finite sequences over A called finite A-*words*, A^ω denotes the set of countably infinite A-words and $A^s \stackrel{\text{def}}{=} A^* \cup A^\omega$ denotes the set of finite and infinite A-words. For any set A, $\mathcal{S}_A = \langle A^s, \varepsilon, \sqsubseteq\rangle$, where ε is the empty word and $\sqsubseteq$ is the prefix order, denotes the cpo of A-words.

The set $\mathit{Pref}\, w \stackrel{\text{def}}{=} \downarrow\{w\}$ (cf. the definition of downcone in Sec. 1.2) is the *set of prefixes* of word w.

Proposition 2
$\mathcal{S}_A$ is a straight cpo and A^* is dense in $\mathcal{S}_A$.

For any finite word $w = a_0 \ldots a_{n-1} \in A^*$, let $\sharp w \stackrel{\text{def}}{=} n$ denote the *length of* w. For any infinite word $w = a_0 a_1 \ldots \in A^\omega$, let $\sharp w \stackrel{\text{def}}{=} \omega$, a special element considered bigger than any natural number. For any finite word $w = a_0 \ldots a_{n-1} \in A^*$ and any natural $k < n$, let $w[k]$ denote a_k, the k-th letter in w, with $w[k]$ undefined for $k \geq n$. Similarly, for any infinite word $w = a_0 a_1 \ldots \in A^\omega$ and any natural k, let $w[k]$ denote a_k, the k-th letter in w. Define the operation $\frown$ of *word concatenation* by

$$(w\frown v)[k] \stackrel{\text{def}}{=} \begin{cases} w[k] & \text{if } k < \sharp w \\ v[k - \sharp w] & \text{if } k \geq \sharp w \end{cases}$$

Thus, $w\frown v = w$ whenever $\sharp w = \omega$. For any word w and any natural $k \in \mathit{Nat}$, w^k is the word w repeated k times: $\underbrace{w\frown \ldots \frown w}_{k}$, which can be formally defined as follows:

$$w^k[n] \stackrel{\text{def}}{=} \begin{cases} w[n \bmod \sharp w] & \text{if } 0 \leq n < k \cdot \sharp w \\ \text{undefined} & \text{otherwise} \end{cases}$$

where $n \bmod \omega \stackrel{\text{def}}{=} n$. Analogously, w^ω is the word w repeated infinitely many times: $\underbrace{w\frown w\frown \ldots}_{\omega}$.

$$w^\omega[n] \stackrel{\text{def}}{=} \begin{cases} w[n \bmod \sharp w] & \text{if } \sharp w > 0 \\ \text{undefined} & \text{if } \sharp w = 0 \end{cases}$$

Given alphabets A and B, any function $f : A^* \to B^*$ such that

- $f\varepsilon = \varepsilon$ and
- $f(w\frown v) = f(w)\frown f(v)$ for $w, v \in A^*$

is called a *word-homomorphism*. Word-homomorphisms are often given as extensions of single letter mappings; notably,

Proposition 3 (on extension of function defined on an alphabet)
An arbitrary function $f : A \to B^*$ assigning words to single letters has a unique extension to a word-homomorphism $f : A^* \to B^*$.

The word-homomorphism extending a certain $f : A \to B^*$ in compliance with Prop. 3 will also be denoted by f.

Proposition 4 (on continuous extension of word-homomorphism)
An arbitrary word-homomorphism $f : A^* \to B^*$ *has a unique continuous extension*

$$\sigma f : A^s \to B^s$$

(such that $(\sigma f)|_{A^*} = f$ *).*

Proof of Proposition 4:

By virtue of Prop. 2 and Thm. 1, it only has to be shown that f is monotone. But this follows directly from the definition of the prefix order in $\mathcal{S}_A$.
□

Proposition 5
For any word-homomorphisms $A^* \xrightarrow{f} B^* \xrightarrow{g} C^*$,

$$\sigma(g \circ f) = \sigma g \circ \sigma f$$

The extension $\sigma f : A^s \to B^s$ generated by a word-homomorphism discussed above may fail to satisfy the homomorphic law

$$\sigma f(w^\frown v) = \sigma f w^\frown \sigma f v$$

For a counterexample, consider the word-homomorphism $f : A^* \to A^*$ for $A \stackrel{\text{def}}{=} \{a, b\}$ which is the unique extension of:

$$\begin{cases} f\,a \stackrel{\text{def}}{=} a \\ f\,b \stackrel{\text{def}}{=} \varepsilon \end{cases}$$

(f erases all occurrences of letter b from its argument word) and let

$$w \stackrel{\text{def}}{=} bbb\ldots \quad \text{and} \quad v \stackrel{\text{def}}{=} a$$

Then $w^\frown v = w$ and $\sigma f(w^\frown v) = \sigma f w = \varepsilon$; on the other hand,

$$\sigma f w^\frown \sigma f v = \varepsilon^\frown a = a \neq \varepsilon$$

Proposition 6
For any word-homomorphism $f : A^* \to B^*$ *and any words* $w \in A^*$ *(finite!) and* $v \in A^s$,

$$\sigma f(w^\frown v) = \sigma f w^\frown \sigma f v$$

Assume $A_1 \subseteq A$ is a subalphabet; A_1-*projection* is the unique continuous extension $\sigma\pi_{A_1} : A^s \to A_1^s$ of the unique word homomorphism $\pi_{A_1} : A^* \to A_1^*$ given by

$$\pi_{A_1} a \stackrel{\text{def}}{=} \begin{cases} a & \text{if } a \in A_1 \\ \varepsilon & \text{if } a \notin A_1 \end{cases}$$

The σ in $\sigma\pi_{A_1}$ will be omitted.

Proposition 7
$\pi_{A_1} \circ \pi_{A_2} = \pi_{A_1 \cap A_2}$.

For any letter $a \in A$, denote the *number of occurrences of* a in a word

$$\sharp_a : A^s \to \mathit{Nat} \cup \{\omega\} \quad \text{by}$$
$$\sharp_a \stackrel{\text{def}}{=} \sharp \circ \pi_{\{a\}}$$

For any word $w \in A^s$, let

$$\mathit{Alph}\, w \stackrel{\text{def}}{=} \left\{ a \in A \,\middle|\, \sharp_a w > 0 \right\}$$

be called the *alphabet of word* w; likewise, for any set W of A-words let

$$\mathit{Alph}\, W \stackrel{\text{def}}{=} \bigcup_{w \in W} \mathit{Alph}\, w$$

be called the *alphabet of set* W.

Proposition 8
For any subalphabet $A_1 \subseteq A$ and any word $w \in A^s$, $\mathit{Alph}\,(\pi_{A_1} w) \subseteq \mathit{Alph}\, w$.

A set of words will be referred to as *disjoint* if the alphabets of its words are pairwise disjoint:

$$W \subseteq A^s \;\&\; (\forall_{w_1, w_2 \in W}\; \mathit{Alph}\, w_1 \cap \mathit{Alph}\, w_2 \neq \emptyset \Rightarrow w_1 = w_2)$$

A word-homomorphism $g : A^* \to B^*$ is *non-glueing for* a disjoint set W of A-words if

$$\forall_{w_1, w_2 \in W}\; \mathit{Alph}\,(g w_1) \cap \mathit{Alph}\,(g w_2) \neq \emptyset \Rightarrow w_1 = w_2$$

i.e. if it preserves the disjointness of the alphabets of the words in W.

Proposition 9
If a word-homomorphism $g : A^* \to B^*$ is non-glueing for a certain disjoint set W of A-words, then

$$(\pi_{\mathit{Alph}\,(\sigma g w)} \circ \sigma g)|_{(\mathit{Alph}\, W)^s} = (\sigma g \circ \pi_{\mathit{Alph}\, w})|_{(\mathit{Alph}\, W)^s}$$

for any $w \in W$; i.e. the following diagram commutes:

$$\begin{tikzcd}
(\mathit{Alph}\, W)^s \arrow[r, "\sigma g"] \arrow[d, "\pi_{\mathit{Alph}\, w}"'] & B^s \arrow[d, "\pi_{\mathit{Alph}\,(\sigma g w)}"] \\
(\mathit{Alph}\, w)^s \arrow[r, "\sigma g"'] & (\mathit{Alph}\,(\sigma g w))^s
\end{tikzcd}$$

Proof of Proposition 9:

First consider single-letter words $a \in Alph\, W$. If $a \in Alph\, w$ then

$$\sigma g a \in (Alph\,(\sigma g w))^*$$

and

$$\pi_{Alph\,(\sigma g w)}(\sigma g a) = \sigma g a = \sigma g(\pi_{Alph\, w}\, a)$$

If, in contrast, $a \in Alph\, W \setminus Alph\, w$ then $a \in Alph\, w'$ for a certain other $w' \in W$ and, as proved above,

$$\pi_{Alph\,(\sigma g w')}(\sigma g a) = \sigma g(\pi_{Alph\, w'}\, a) = \sigma g a$$

By the non-glueing property of g for W, no letters of $\sigma g a$ belong to $Alph\,(\sigma g w)$; hence

$$\pi_{Alph\,(\sigma g w)}(\sigma g a) = \varepsilon$$

and also

$$\sigma g(\pi_{Alph\, w}\, a) = \sigma g \varepsilon = \varepsilon$$

To complete the proof, extend it by continuity to multi-letter words from $(Alph\, W)^s$.
□

1.4 Factorization of partial orders and cpo-s

(The reader is referred to [16] for motivations, examples and proofs of facts discussed in this section.)

Assume X is a set with a partial order $\sqsubseteq\, \subseteq X \times X$. An equivalence $\sim\, \subseteq X \times X$ is said to be *consistent*[2] with $\sqsubseteq$ if it satisfies the following two requirements:

1. for any $x, y, z \in X$, if $x \sqsubseteq y \sqsubseteq z$ and $x \sim z$ then $x \sim y$; i.e.

   if $\begin{tikzcd} & y \arrow[dr] & \\ x \arrow[ur] \arrow[rr, dash] & & z \end{tikzcd}$ then $\begin{tikzcd} & y \arrow[dr, dash] & \\ x \arrow[ur, dash] \arrow[rr, dash] & & z \end{tikzcd}$

   (arrows denote the partial order, lines without arrowheads denote the equivalence).

2. for any $x, y, z \in X$ s.t. $x \sim y \sqsubseteq z$, there exists a $v \in X$ s.t. $x \sqsubseteq v \sim z$; i.e.

[2] Marek Bednarczyk calls such an equivalence "upper consistent" with the partial order and he considers a dual notion of lower consistency.

every diagram
$$\begin{tikzcd} x \arrow[d, dash] & \\ y \arrow[r] & z \end{tikzcd}$$
may be completed to
$$\begin{tikzcd} x \arrow[r] \arrow[d, dash] & v \arrow[d, dash] \\ y \arrow[r] & z \end{tikzcd}$$

Equivalence $\sim$ is *ω-consistent with* partial order $\sqsubseteq$ if it is consistent with $\sqsubseteq$ and satisfies the following additional condition called ω-rule:

3. for any ascending chain $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \ldots$ and any sequence of $\sim$-related upper bounds on x_i-s

$$y_0 \sim y_1 \sim y_2 \sim \ldots \quad \text{such that} \quad x_i \sqsubseteq y_i \quad \text{for } i \in \mathit{Nat}$$

   there exists an upper bound y on x_i-s $\sim$-related to y_i-s; i.e., every infinite diagram

$$\begin{tikzcd} x_0 \arrow[r] \arrow[d] & x_1 \arrow[r] \arrow[d] & x_2 \arrow[r] \arrow[d] & \cdots \\ y_0 \arrow[r, dash] & y_1 \arrow[r, dash] & y_2 \arrow[r, dash] & \cdots \end{tikzcd}$$

   may be completed to

$$\begin{tikzcd} x_0 \arrow[r] \arrow[dd] \arrow[drrrr] & x_1 \arrow[r] \arrow[dd] \arrow[drrr] & x_2 \arrow[r] \arrow[dd] \arrow[drr] & \cdots & & \\ & & & & y & \\ y_0 \arrow[r, dash] \arrow[urrrr, dash] & y_1 \arrow[r, dash] \arrow[urrr, dash] & y_2 \arrow[r, dash] \arrow[urr, dash] & \cdots & & \end{tikzcd}$$

Proposition 10
Equality is ω-consistent with any partial order.

Whenever an equivalence $\sim$ is consistent with a partial order $\sqsubseteq$, the quotient set of X by $\sim$ has a natural partial order inherited from $\langle X, \sqsubseteq \rangle$:

$$\begin{array}{l} \sqsubseteq \,\subseteq\, (X/_\sim) \times (X/_\sim) \\ [x]_\sim \sqsubseteq [y]_\sim \overset{\text{def}}{\Longleftrightarrow} \exists_z\ x \sqsubseteq z \sim y \end{array}$$

$$\begin{tikzcd} x \arrow[r] & z \arrow[d, dash] \\ & y \end{tikzcd}$$

Proposition 11
The above definition is correct, i.e. it does not depend on a selection of x and y from their equivalence classes.

Proposition 12
If $\sim$ is consistent with $\sqsubseteq$ then the above defined relation $\sqsubseteq \subseteq (X/_\sim) \times (X/_\sim)$ is a partial order. Moreover, the natural quotient mapping

$$\begin{cases} q : X \to X/_\sim \\ q\,x \overset{\text{def}}{=} [x]_\sim \end{cases}$$

is monotone.

Proposition 13
Assume that an equivalence $\sim$ on X is consistent with a partial order $\sqsubseteq \subseteq X \times X$ and that an equivalence $\sim'$ on X' is consistent with a partial order $\sqsubseteq' \subseteq X' \times X'$. Assume further that a certain monotone map $f : X \to X'$ preserves the equivalences:

$$\forall_{x,y}\ x \sim y \Rightarrow fx \sim' fy \tag{1}$$

Then the quotient map

$$\begin{cases} \bar{f} : X/_\sim \to X'/_{\sim'} \\ \bar{f}\,[x]_\sim \overset{\text{def}}{=} [fx]_{\sim'} \end{cases}$$

is well-defined and monotone wrt natural quotient partial orders.

If $\langle X, 0, \sqsubseteq\rangle$ is a cpo and an equivalence $\sim \subseteq X \times X$ is consistent with $\sqsubseteq$, then the quotient structure $\langle X/_\sim\,,\ [0]_\sim\,,\ \sqsubseteq\rangle$, while being a partial order by virtue of Prop. 12, may fail to be a cpo — for an example see [16]. This is why the stronger notion of ω-consistency has to be considered.

Proposition 14
If $\langle X, 0, \sqsubseteq\rangle$ is a cpo and an equivalence $\sim \subseteq X \times X$ is ω-consistent with $\sqsubseteq$, then $\langle X/_\sim\,,\ [0]_\sim\,,\ \sqsubseteq\rangle$ is a cpo too. Moreover, the natural quotient mapping

$$\begin{cases} q : X \to X/_\sim \\ q\,x \overset{\text{def}}{=} [x]_\sim \end{cases}$$

is continuous.

Proposition 15
Assume $\langle X, 0, \sqsubseteq\rangle$ *is a cpo and an equivalence* $\sim$ *on* X *is* ω*-consistent with* $\sqsubseteq$. *Assume further that*

$$\begin{array}{l} x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \ldots \quad \textit{and} \\ y_0 \sqsubseteq y_1 \sqsubseteq y_2 \sqsubseteq \ldots \end{array}$$

are ascending chains. If $x_n \sim y_n$ *for any* $n \in Nat$ *then*

$$\left(\bigsqcup_{n \in Nat} x_n\right) \sim \left(\bigsqcup_{n \in Nat} y_n\right)$$

Proposition 16
Let $\langle X, 0, \sqsubseteq\rangle$ *and* $\langle X', 0', \sqsubseteq'\rangle$ *be cpo-s. Assume that an equivalence* $\sim$ *on* X *is* ω*-consistent with the partial order* $\sqsubseteq \subseteq X \times X$ *and that an equivalence* $\sim'$ *on* X' *is* ω*-consistent with the partial order* $\sqsubseteq' \subseteq X' \times X'$. *Assume further that a certain continuous map* $f : X \to X'$ *preserves the equivalences:*

$$\forall_{x,y}\ x \sim y \Rightarrow fx \sim' fy$$

Then the quotient map

$$\begin{cases} \bar{f} : X/_{\sim} \to X'/_{\sim'} \\ \bar{f}\,[x]_{\sim} \stackrel{\text{def}}{=} [fx]_{\sim'} \end{cases}$$

is well-defined and continuous wrt natural quotient partial orders.

2 Concurrent processes

Vectors of processes, introduced here, are the main object of our investigations. Later in the paper, the tools for such investigations based on the notion of homotopy will be discussed.

2.1 Vectors of processes

Vector of processes[3] is any quadruple

$$\mathcal{V} = \langle P, A, L, I\rangle$$

where

- P is an arbitrary set of *processes*.

[3] Wiesław Pawłowski has proposed the term "multiprocess".

- $A = \{A_p\}_{p \in P}$ is a family of mutually disjoint sets of *actions*. Sometimes the letter A will also be used to denote the set of all actions, $A = \bigcup_{p \in P} A_p$.

- $L : \prod_{p \in P} A_p^s$ is a *software for* vector $\mathcal{V}$ [4]; it consists of *programs for* particular processes, $Lp \in A_p^s$ for $p \in P$. Programs for processes are assumed to involve all actions in their respective alphabets A_p:

$$Alph\,(Lp) = A_p \tag{2}$$

  for $p \in P$. The product $Conf\,\mathcal{V} \stackrel{\text{def}}{=} \prod_{p \in P} Pref\,(Lp) \subseteq \prod_{p \in P} A_p^s$ [5] will be called the set of *configurations* and the product $Conf_f\,\mathcal{V} \stackrel{\text{def}}{=} \prod_{p \in P}(A_p^* \cap Pref\,(Lp))$ will be called the set of *finitary configurations*. The *empty configuration* $\bar{\varepsilon}$ is defined by $\bar{\varepsilon}\,p \stackrel{\text{def}}{=} \varepsilon$ for $p \in P$.

- $I \subseteq Conf_f\,\mathcal{V}$ is a set of *admissible configurations*; we assume that $\bar{\varepsilon} \in I$. The finitary configurations which are not admissible, those in $O \stackrel{\text{def}}{=} Conf_f\,\mathcal{V} \setminus I$, are called *excluded configurations* or *obstacles*.

Informally, a program Lp is the sequence of actions that process p "wants" to engage in, but it may be delayed or prevented from doing so by an interference from other processes. A configuration is a collection of prefixes of programs and this collection characterizes the state of execution of particular processes. The set $Conf\,\mathcal{V}$ of configurations has a natural cpo structure given by the component-wise prefix ordering. Some finitary configurations, for instance the ones in which the requirement of an exclusive access to a scarce resource is violated, may be forbidden (excluded configurations).

A vector of processes $\mathcal{V} = \langle P, A, L, I \rangle$ as above generates sequences of actions which are consistent with the vector's software and admissible, in the following sense. Define the function *endpt* that yields the projections of a given finite A-word to A_p-s for $p \in P$:

$$\begin{cases} endpt\ : A^s \to \prod_{p \in P} A_p^s \\ endpt\,w\,p \stackrel{\text{def}}{=} \pi_{A_p} w \end{cases}$$

Function $endpt\,w$ is called the *endpoint* of word w. A word $w \in A^s$ is said to be *consistent with* vector $\mathcal{V}$ if the endpoints of its finite prefixes are configurations: $endpt\,v \in Conf\,\mathcal{V}$ for all $v \in A^*$, $v \sqsubseteq w$.

Proposition 17
A word $w \in A^s$ *is consistent with* $\mathcal{V}$ *if and only if*

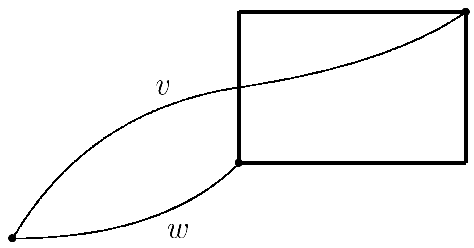
$$\forall_{p \in P}\ \pi_{A_p} w \sqsubseteq Lp$$

Informally, a word is consistent with $\mathcal{V}$ if it is a "zip-join" of prefixes of programs. The set of words which are consistent with vector $\mathcal{V}$ will be denoted by $Cons\,\mathcal{V}$.

[4] Recall form Sec. 1.1 on page 5 that $\prod$ denotes the generalized Cartesian product.
[5] Recall from Sec. 1.3 on page 7 that $Pref\,w$ is the set of prefixes of word w.

Any two consistent words define a *cube* describing the configurations contained between their endpoints:

$$\begin{cases} \Box : \mathit{Cons}\,\mathcal{V} \times \mathit{Cons}\,\mathcal{V} \to \mathcal{P}(\mathit{Conf}\,\mathcal{V}) \\ \Box\,\langle w, v\rangle \stackrel{\text{def}}{=} \left\{ c \,\middle|\, \mathit{endpt}\, w \sqsubseteq c \sqsubseteq \mathit{endpt}\, v \right\} \end{cases}$$



Proposition 18
For any finite word $w \in A^*$ and any action $a \in A$, such that $w\frown a \in \mathit{Cons}\,\mathcal{V}$, the cube between w and $w\frown a$ consists of exactly two configurations:

$$\Box\,\langle w\,,\, w\frown a\rangle = \{\mathit{endpt}\, w\,,\, \mathit{endpt}\,(w\frown a)\}$$

A word $w \in A^s$ is *admissible* (or *avoids obstacles*) in $\mathcal{V}$ if the endpoints of its all prefixes are in the set I of admissible configurations:

$$\forall_{v\in A^*}\; v \sqsubseteq w \Rightarrow \mathit{endpt}\, v \in I$$

Since $I \subseteq \mathit{Conf}\,\mathcal{V}$, the admissibility is a stronger property than the consistency. An admissible word will be called a *path generated* by vector $\mathcal{V}$:

$$\mathit{Pth}\,\mathcal{V} \stackrel{\text{def}}{=} \left\{ w \in A^s \,\middle|\, \forall_{v\in A^*}\; v \sqsubseteq w \Rightarrow \mathit{endpt}\, v \in I \right\}$$

— this denotes the set of paths generated by vector $\mathcal{V}$.

2.2 Example: reader and writer

Consider the vector

$$\mathcal{V}^{rw} = \langle P^{rw}, A^{rw}, L^{rw}, I^{rw}\rangle$$

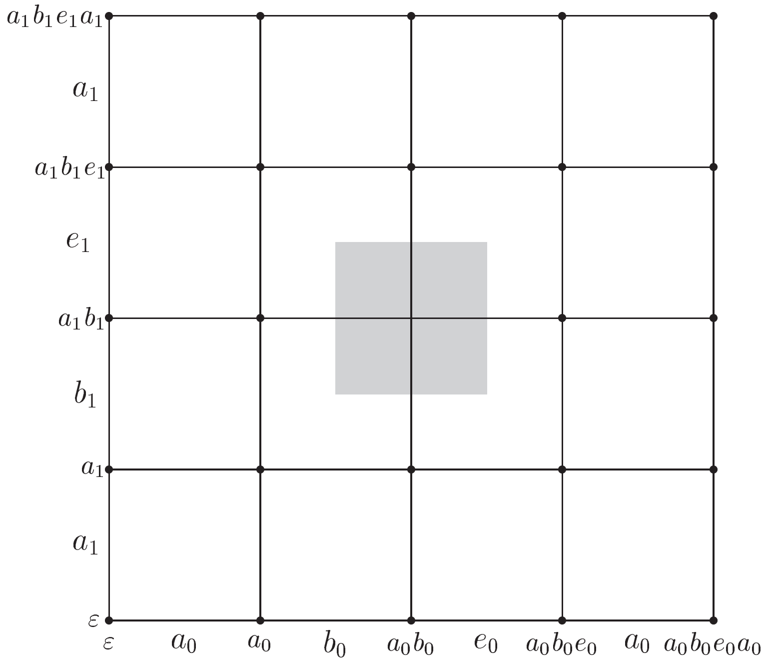
consisting of processes named 0 and 1 performing actions b_0 (beginning of reading), e_0 (end of reading), b_1 (beginning of writing), e_1 (end of writing) and a_0, a_1 (anything else):

$$P^{rw} \stackrel{\text{def}}{=} \{0, 1\}$$
$$L^{rw}\,0 \stackrel{\text{def}}{=} a_0 b_0 e_0 a_0 \quad L^{rw}\,1 \stackrel{\text{def}}{=} a_1 b_1 e_1 a_1$$

This means process 0 performs the neutral action a_0, then reads $b_0 e_0$, then runs a_0 again; and process 1 goes a_1, then writes $b_1 e_1$, then a_1 again; A^{rw} is thus implicitely defined as $\{a_0, b_0, e_0, a_1, b_1, e_1\}$. The set of obstacles consists of a single configuration $\langle a_0 b_0, a_1 b_1\rangle$ corresponding to the forbidden situation when the reader and the writer are simultaneously using the resource:

$$I^{rw} \stackrel{\text{def}}{=} \mathit{Conf}_f\,\mathcal{V}^{rw} \setminus \{\langle a_0 b_0, a_1 b_1\rangle\}$$

The paths generated by $\mathcal{V}^{rw}$ are the finite ones beginning in $\langle\varepsilon,\varepsilon\rangle$ and going from left to right and from bottom upwards in the configuration space $\mathit{Conf}\,\mathcal{V}^{rw}$ and not intersecting the shaded area, i.e. the obstacle, at the following picture:



2.3 Example: complete sequentialization

Consider the vector

$$\mathcal{V}^{cs} = \langle P^{cs}, A^{cs}, L^{cs}, I^{cs}\rangle$$

consisting of processes named 0 and 1 indefinitely performing actions a_0 and a_1, respectively:

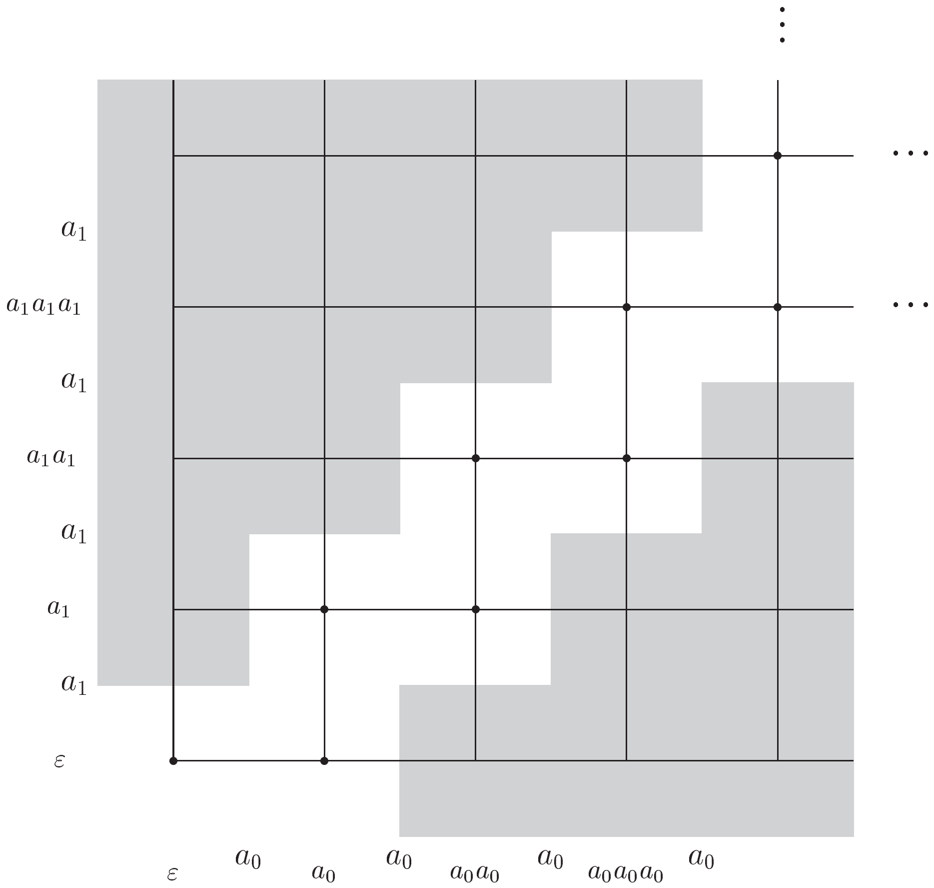
$$P^{cs} \stackrel{\text{def}}{=} \{0,1\}$$
$$A^{cs} \stackrel{\text{def}}{=} \{a_0, a_1\}$$
$$L^{cs}\,0 \stackrel{\text{def}}{=} (a_0)^\omega \quad L^{cs}\,1 \stackrel{\text{def}}{=} (a_1)^\omega$$

with admissible configurations:

$$I^{cs} \stackrel{\text{def}}{=} \big\{\big\langle a_0^n, a_1^k\big\rangle \,\big|\, k \le n \le k+1\big\}$$

This means that the admissible paths begin with a_0 and no two adjacent actions are equal. Thus, any admissible path is a prefix of $(a_0a_1)^\omega$.

The paths generated by $\mathcal{V}^{cs}$ are the finite and infinite ones beginning in $\langle\varepsilon,\varepsilon\rangle$ and going from left to right and from bottom upwards in the configuration space $\mathit{Conf}\,\mathcal{V}^{cs}$ depicted below and not intersecting the shaded area:



2.4 Example: producer and consumer with a 1-place buffer

Consider the vector

$$\mathcal{V}^{pc1} = \left\langle P^{pc1}, A^{pc1}, L^{pc1}, I^{pc1} \right\rangle$$

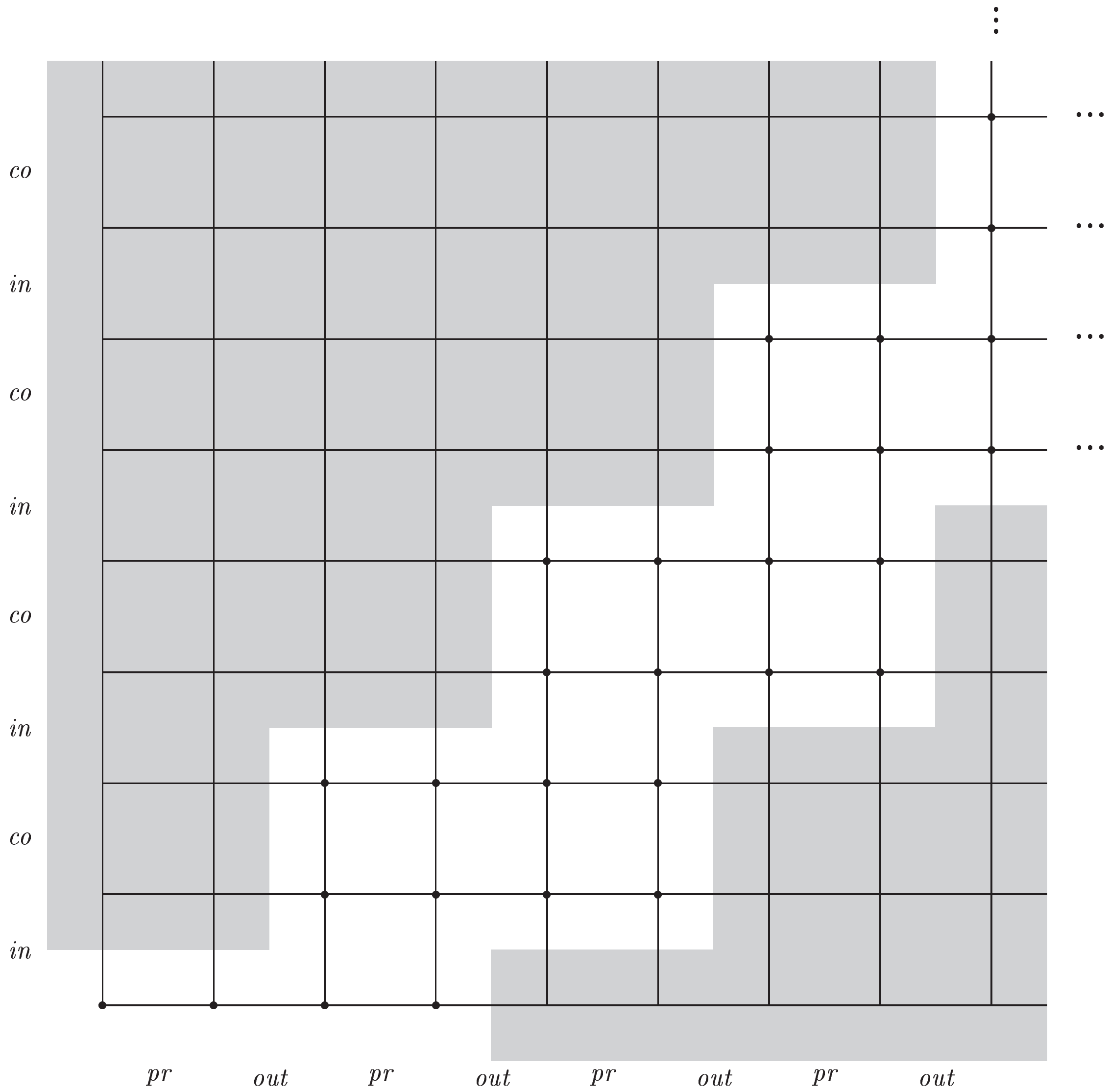
consisting of processes named 0 and 1 indefinitely performing actions *pr* (produce), *out* (output to buffer), *in* (input from buffer) and *co* (consume):

$$P^{pc1} \stackrel{\text{def}}{=} \{0, 1\}$$
$$L^{pc1}\, 0 \stackrel{\text{def}}{=} (pr\ out)^{\omega} \quad L^{pc1}\, 1 \stackrel{\text{def}}{=} (in\ co)^{\omega}$$

with the obvious admissible configurations:

$$I^{pc1} \stackrel{\text{def}}{=} \left\{ \langle w, v \rangle \sqsubseteq \langle (pr\ out)^{\omega}, (in\ co)^{\omega} \rangle \,\middle|\, \sharp_{in} w \leq \sharp_{out} w \leq \sharp_{in} w + 1 \right\}$$

The paths generated by $\mathcal{V}^{pc1}$ are the ones that go from left to right and from bottom upwards and do not intersect the shaded area at the following picture:



2.5 Example: producer and consumer with a 2-place buffer

A 2-place buffer is assisted by an additional process which shifts elements from the first to the second place in the buffer:

$$\mathcal{V}^{pc2} = \left\langle P^{pc2}, A^{pc2}, L^{pc2}, I^{pc2} \right\rangle$$

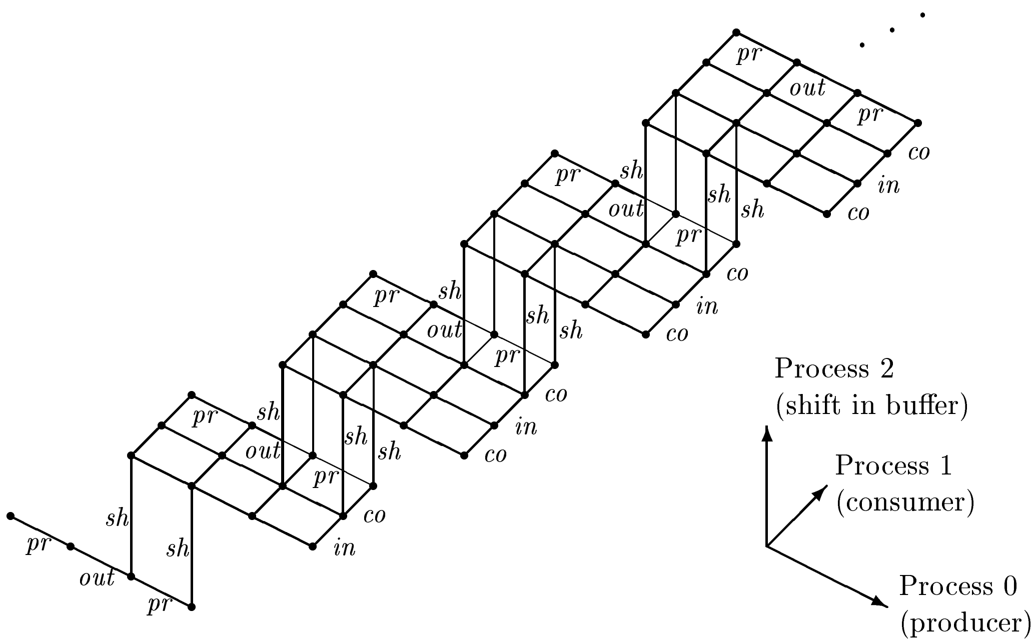
where

$$P^{pc2} \stackrel{\text{def}}{=} \{0, 1, 2\}$$
$$L^{pc2}0 \stackrel{\text{def}}{=} (pr\ out)^\omega \quad L^{pc2}1 \stackrel{\text{def}}{=} (in\ co)^\omega \quad L^{pc2}2 \stackrel{\text{def}}{=} sh^\omega$$

(sh — shifting) with the obvious admissible configurations:

$$I^{pc2} \stackrel{\text{def}}{=}$$
$$\left\{ \langle w, v, u \rangle \sqsubseteq \langle (pr\ out)^\omega, (in\ co)^\omega, sh^\omega \rangle \,\middle|\, \begin{array}{l} \sharp_{sh} u \le \sharp_{out} w \le \sharp_{sh} u + 1\ \& \\ \sharp_{in} v \le \sharp_{sh} u \le \sharp_{in} v + 1 \end{array} \right\}$$

In this case, the set of admissible configurations is 3-dimensional:



2.6 Example: two dining philosophers

This is a simplified version of Dijkstra's famous problem of 5 dining philosophers. In this version, there are only two of them facing each other; each other's left fork is his companion's right fork, and each other's right fork is his companion's left fork.

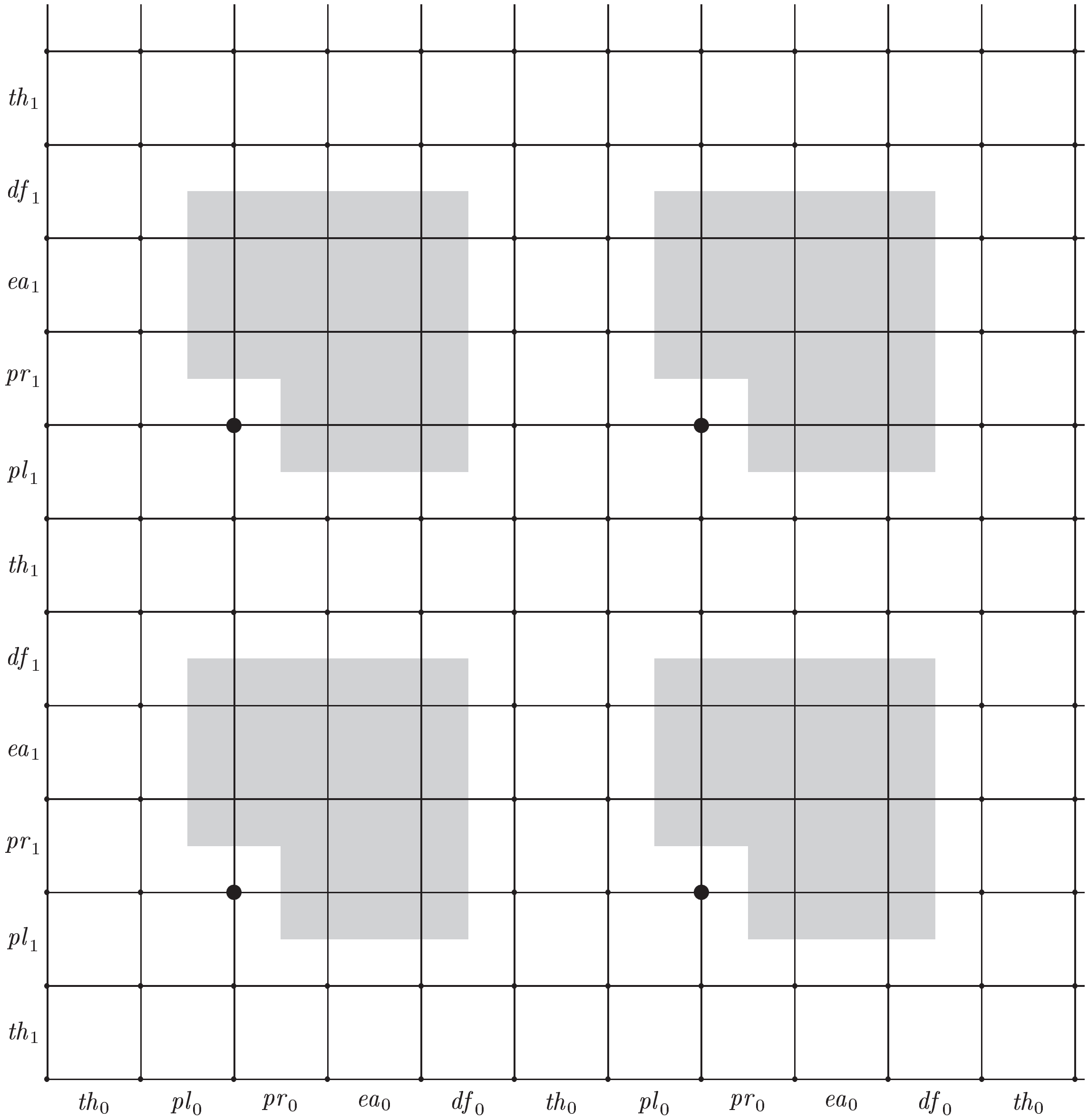
$$\mathcal{V}^{ph2} = \left\langle P^{ph2}, A^{ph2}, L^{ph2}, I^{ph2} \right\rangle$$

where

$$P^{ph2} \stackrel{\text{def}}{=} \{0, 1\}$$
$$L^{ph2} p \stackrel{\text{def}}{=} v_p^\omega \quad \text{with} \quad v_p \stackrel{\text{def}}{=} th_p\, pl_p\, pr_p\, ea_p\, df_p \quad \text{for } p \in \{0, 1\}$$

(th — thinking, pl — picking left fork, pr — picking right fork, ea — eating, df — dropping both forks) with the following admissible configurations:

$$I^{ph2} \stackrel{\text{def}}{=}$$
$$Conf_f\, \mathcal{V}^{ph2}$$
$$\setminus \bigcup_{n,k \in Nat} \Box \left\langle v_0^n\, v_1^k\, th_0\, pl_0\, th_1\, pl_1\,,\ v_0^n\, v_1^k\, th_0\, pl_0\, pr_0\, ea_0\, th_1\, pl_1\, pr_1\, ea_1 \right\rangle$$
$$\cup \left\{ \left\langle v_0^n\, th_0\, pl_0\,,\ v_1^k\, th_1\, pl_1 \right\rangle \,\middle|\, n, k \in Nat \right\}$$



Note the fat points at the picture which correspond to deadlocks — if both philosophers have picked their left forks, both will indefinitely wait for their right forks.

2.7 Higher dimensional process vectors

While elaborating the notion of vector of processes, I have been several times misled by simplified two-dimensional intuitions. It is, therefore, important also to have in mind higher dimensional pictures, in which the obstacles separate paths in a different way than they do in two dimensions.

Consider the vectors

$$\mathcal{V}^{\mathit{3d\text{-}up}} \stackrel{\text{def}}{=} \left\langle \{0,1,2\} \,,\, \bigcup_{i=0,1,2} \{b_i, e_i\} \,,\, L^{\mathit{3d}} \,,\, I^{\mathit{3d\text{-}up}} \right\rangle$$
$$\mathcal{V}^{\mathit{3d\text{-}dn}} \stackrel{\text{def}}{=} \left\langle \{0,1,2\} \,,\, \bigcup_{i=0,1,2} \{b_i, e_i\} \,,\, L^{\mathit{3d}} \,,\, I^{\mathit{3d\text{-}dn}} \right\rangle$$

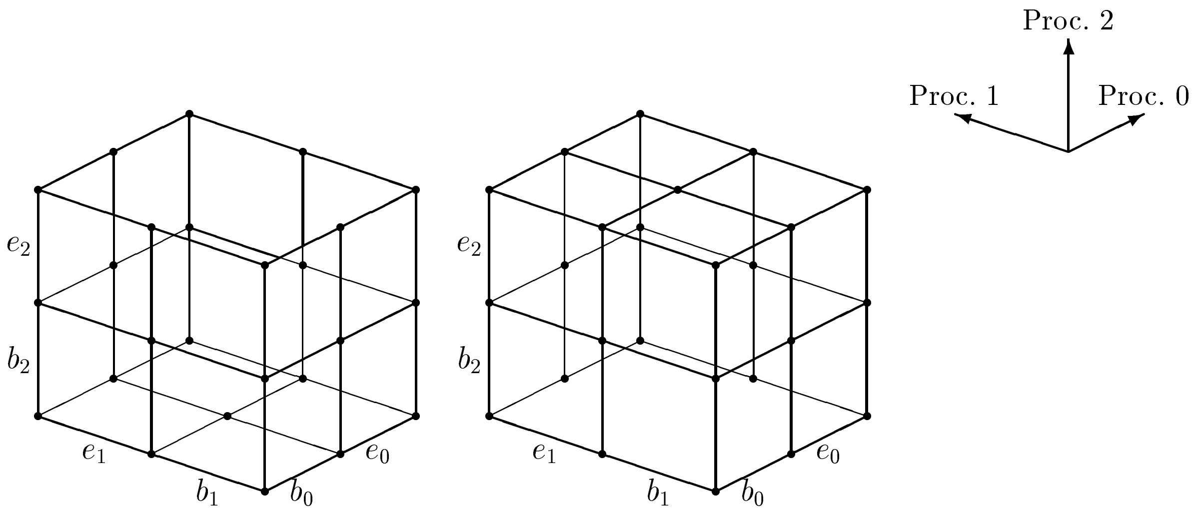
where

$$L^{\mathit{3d}}\, i \stackrel{\text{def}}{=} b_i e_i \quad \text{for } i = 0,1,2$$

is the software, the same for the both vectors; and

$$I^{\mathit{3d\text{-}up}} \stackrel{\text{def}}{=} \mathit{Conf}_f\, \mathcal{V}^{\mathit{3d\text{-}up}} \setminus \{\langle b_0, b_1, b_2\rangle \,, \langle b_0, b_1, b_2 e_2\rangle\}$$
$$I^{\mathit{3d\text{-}dn}} \stackrel{\text{def}}{=} \mathit{Conf}_f\, \mathcal{V}^{\mathit{3d\text{-}dn}} \setminus \{\langle b_0, b_1, \varepsilon\rangle \,, \langle b_0, b_1, b_2\rangle\}$$

The set of admissible configurations is in each case the three-dimensional cube with two points missing:



$\mathcal{V}^{\mathit{3d\text{-}up}}$ $\mathcal{V}^{\mathit{3d\text{-}dn}}$

There is nothing in the definition of the vector of processes that would prevent the set of processes from being infinite. Consider the vector

$$\mathcal{V}^{\omega} \stackrel{\text{def}}{=} \left\langle \mathit{Nat} \,,\, \left\{ a_n \,\middle|\, n \in \mathit{Nat} \right\} \,,\, L^{\omega} \,,\, I^{\omega} \right\rangle$$

with

$$L^{\omega}\, n \stackrel{\text{def}}{=} a_n \quad \text{for } n \in \mathit{Nat}$$

and $I^{\omega} \stackrel{\text{def}}{=} \mathit{Conf}_f\, \mathcal{V}^{\omega}$ (no obstacles).

The configuration space is in this case the ω-dimensional cube. Even though the program for any single process is finite (actually, one-letter) there exist infinite paths; e.g. $a_0 a_1 a_2 \ldots$.

3 Assigning cpo-s to vectors of processes

3.1 Vector of processes as path generator

Recall from Sec. 2.1 on page 15 the definition of paths generated by a vector of processes $\mathcal{V} = \langle P, A, L, I \rangle$:

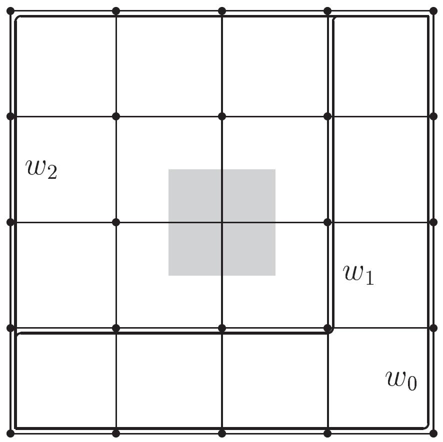
$$\mathit{Pth}\,\mathcal{V} \stackrel{\text{def}}{=} \left\{ w \in A^s \,\middle|\, \forall_{v \in A^*}\ v \sqsubseteq w \Rightarrow \mathit{endpt}\, v \in I \right\}$$

In Sections 2.2 through 2.7 (pages 15 through 21), path sets generated by example vectors were described.

Proposition 19
For any vector $\mathcal{V} = \langle P, A, L, I \rangle$ *of processes, the set* $\mathit{Pth}\,\mathcal{V}$ *with the prefix ordering is a cpo.*

3.2 Homotopy between paths

We want to distinguish between paths joining the same points of the configuration space but going at different sides around the shaded areas at the pictures from Sections 2.2 through 2.6; on the other hand, we want to identify the paths joining the same points and in the same way avoiding the shaded areas. Recall that the shaded areas are determined by the set of excluded configurations of the vector. To explain the intended identifications of paths, let us consider the reader-and-writer example from Sec. 2.2 (page 15):



There are no "holes" between w_0 and w_1, therefore these two paths will be identified. On the other hand, w_2 will stay separate since it avoids the shaded area in a different way.

More precisely, *path homotopy* in a vector $\mathcal{V} = \langle P, A, L, I \rangle$ of processes is the least equivalence relation $H \subseteq \mathit{Pth}\,\mathcal{V} \times \mathit{Pth}\,\mathcal{V}$ such that:

- if

$$\begin{aligned} w_1 &= w' {}^\frown a_1 a_2 {}^\frown w'' \in \mathit{Pth}\,\mathcal{V} \quad \text{and} \\ w_2 &= w' {}^\frown a_2 a_1 {}^\frown w'' \in \mathit{Pth}\,\mathcal{V} \end{aligned}$$

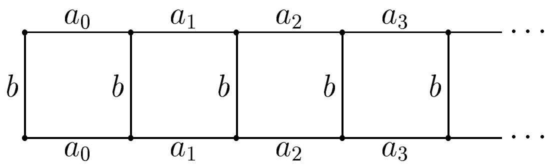
(i.e. w_1 and w_2 differ by a transposition of two adjacent actions) then $w_1 \, H \, w_2$;

- the ω-rule holds for H; i.e. (recall from Sec. 1.4 on p. 11) if $w_0 \sqsubseteq w_1 \sqsubseteq w_2 \sqsubseteq \ldots$ is an ascending chain of paths and $v_0 \, H \, v_1 \, H \, v_2 \, H \ldots$ is a sequence of H-related paths such that $w_i \sqsubseteq v_i$ for $i \in \mathit{Nat}$, then there exists a path u such that $w_i \sqsubseteq u$ and $v_i \, H \, u$ for any $i \in \mathit{Nat}$.

The first item in the above definition accounts for glueing together a path with any other path to which it may be brought by a finite number of transpositions of adjacent actions without ever leaving the admissible set I. For instance, it implies that the paths

$$\begin{array}{l} ba_0a_1a_2\ldots \\ a_0ba_1a_2\ldots \\ a_0a_1ba_2\ldots \\ a_0a_1a_2b\ldots \end{array}$$

in



are H-related to each other. The second item in the definition implies that the path

$$a_0a_1a_2\ldots$$

is also H-related to them. To see this, set w_i and v_i as follows:

$$\begin{aligned} w_i &\stackrel{\text{def}}{=} a_0a_1\ldots a_{i-1} \quad \text{and} \\ v_i &\stackrel{\text{def}}{=} a_0a_1\ldots a_{i-1}ba_{i+1}a_{i+2}\ldots \end{aligned}$$

and check that $w_i \sqsubseteq w_{i+1}$, $w_i \sqsubseteq v_i$ and $v_i \, H \, v_{i+1}$ for $i \in \mathit{Nat}$.

In algebraic topology, homotopy is a continuous deformation of mappings; in particular, two paths are homotopic if one can be continuously deformed into the other. This may be perceived as the limit of the above defined notion when the actions are replaced by greater and greater numbers of smaller and smaller actions.

Proposition 20
Equivalence H is ω-consistent with the prefix partial order $\sqsubseteq$ (cf. Sec. 1.4 on page 10).

Proposition 21
Assume $\mathcal{V} = \langle P, A, L, I \rangle$ is a vector of processes and let $w, v \in A^*$ be finite words such that

- $w, w^\frown v \in \mathit{Pth}\,\mathcal{V}$, *and*
- $\Box \langle w, w^\frown v \rangle \subseteq I$, *i.e. the cube between w and v does not contain any obstacles.*

Let $u \in A^*$ be another finite word such that $\mathit{endpt}\, u = \mathit{endpt}\, v$. Then $w^\frown u \in \mathit{Pth}\,\mathcal{V}$ and

$$(w^\frown u) \, H \, (w^\frown v)$$

3.3 Homotopy cpo

Denote by $\mathit{Htp}\,\mathcal{V}$ the quotient set of paths generated by the relation H of path homotopy:

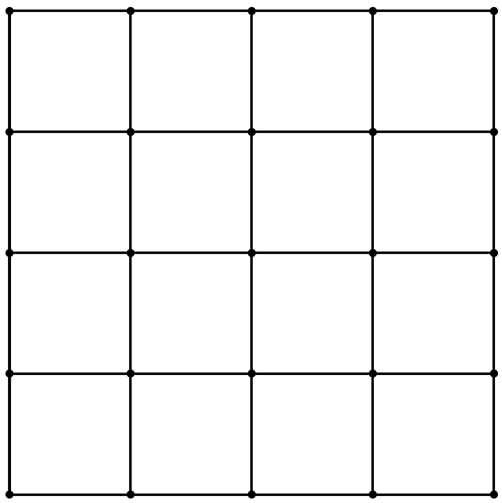
$$\mathit{Htp}\,\mathcal{V} \stackrel{\text{def}}{=} \mathit{Pth}\,\mathcal{V}/H$$

and let $\sqsubseteq$ denote the relation in $\mathit{Htp}\,\mathcal{V}$ defined by

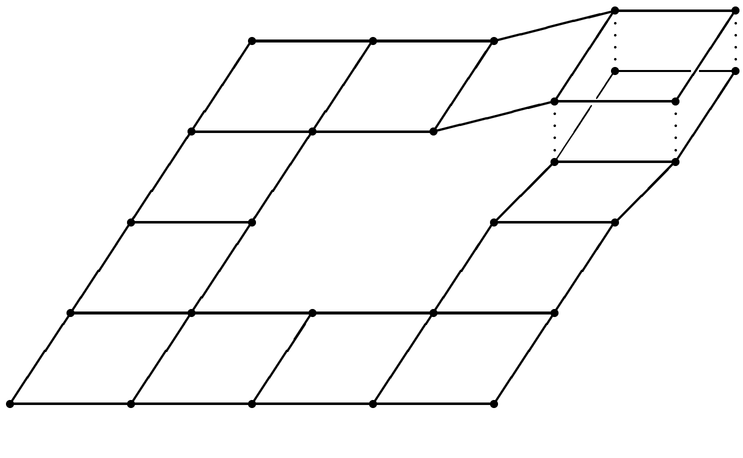
$$[w_1]_H \sqsubseteq [w_2]_H \stackrel{\text{def}}{\iff} \exists_v\ w_1 \sqsubseteq v\, H\, w_2$$

By Prop. 20 above and Prop. 14 (page 12), $\langle \mathit{Htp}\,\mathcal{V}, [\varepsilon]_H, \sqsubseteq\rangle$ is a cpo. It is called the *homotopy cpo* of vector of processes $\mathcal{V}$.

In the uninteresting case, when the vector of processes does not contain any "holes", i.e. when $I = \mathit{Conf}_f\,\mathcal{V}$, the paths that end in the same point are glued together. For instance, if $\mathcal{V}^{rw0}$ denotes the reader-and-writer example from Sec. 2.2 on page 15 but with $I^{rw0} \stackrel{\text{def}}{=} \mathit{Conf}_f\,\mathcal{V}^{rw0} = \mathit{Conf}\,\mathcal{V}^{rw0}$, then $\mathit{Htp}\,\mathcal{V}^{rw0}$ is isomorphic to the configuration space $\mathit{Conf}\,\mathcal{V}^{rw0}$:



with the ordering from left to right and from bottom upwards. A similar picture corresponds to $\mathit{Htp}\,\mathcal{V}$ in the case of non-trivial I, only it splits behind the holes. So $\mathit{Htp}\,\mathcal{V}^{rw}$ is:

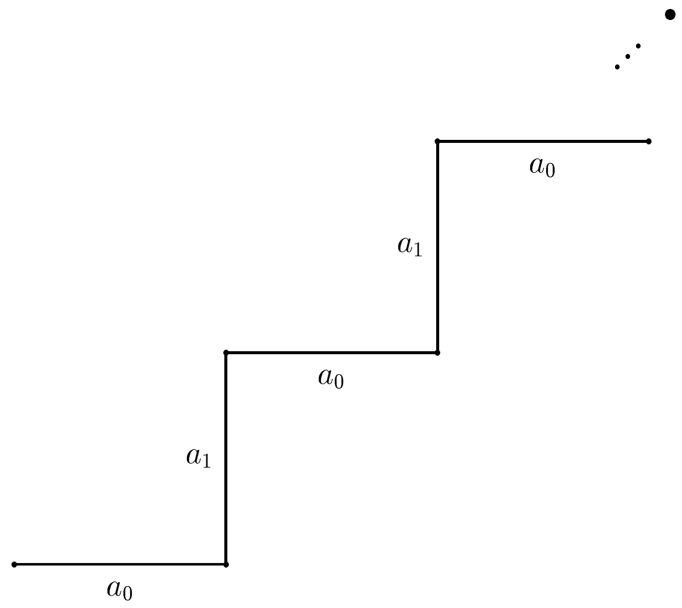


With more complex structure of excluded regions, the splitting becomes more complex too.

3.4 Homotopy cpo-s for example vectors of processes

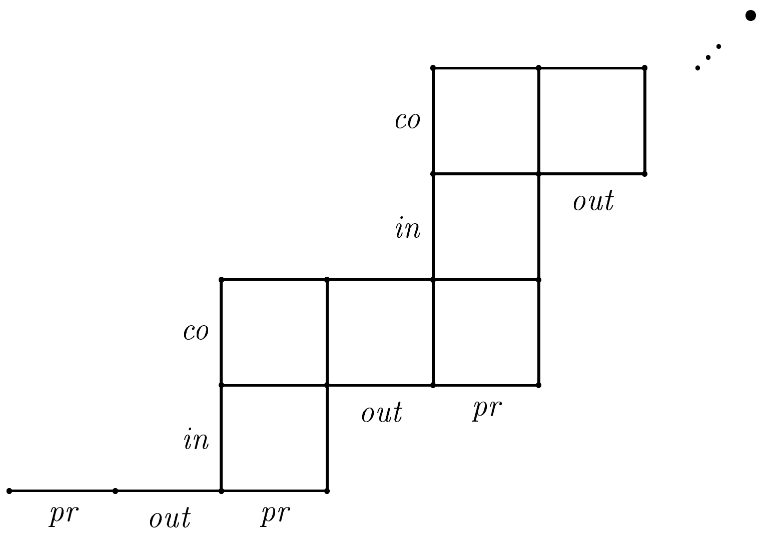
The homotopy cpo for the reader-and-writer example from Sec. 2.2 (page 15) has already been presented in Sec. 3.3.

For the complete-sequentialization example from Sec. 2.3 (page 16), $\mathit{Htp}\,\mathcal{V}^{cs}$ is linear with an extra point corresponding to the infinite path:

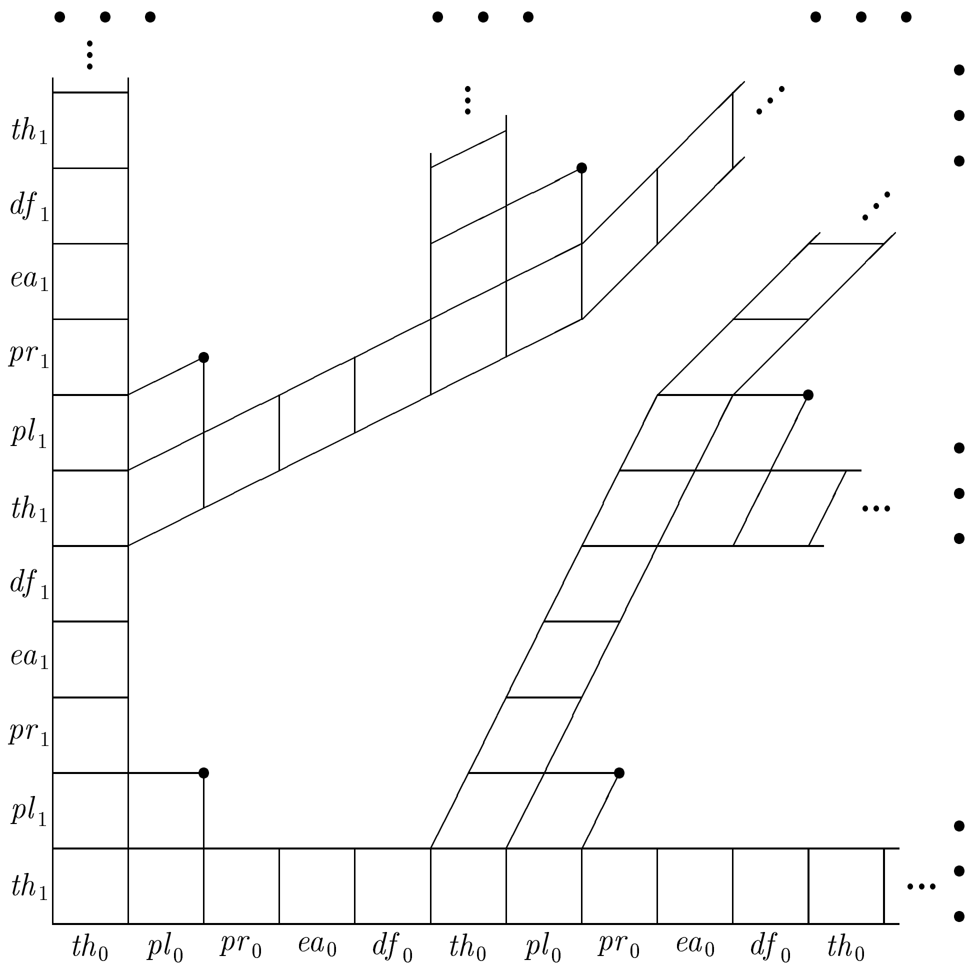


(ordered, as usual, from left to right and from bottom upwards).

The homotopy cpo $\mathit{Htp}\,\mathcal{V}^{pc1}$ for producer and consumer with a 1-place buffer from Sec. 2.4 (page 17) is very similar:

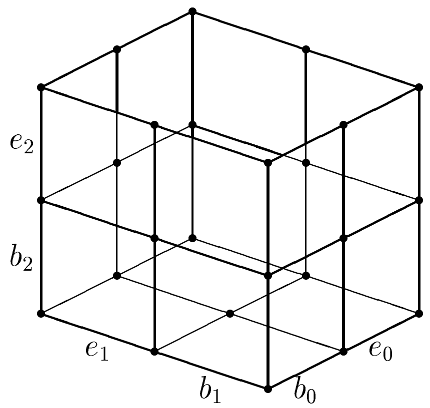


In the infinitary philosophers example from Sec. 2.6 (page 19), the picture will be split in infinitely many branches, behind every “hole”. Thus, $Htp\,\mathcal{V}^{ph2}$ is:

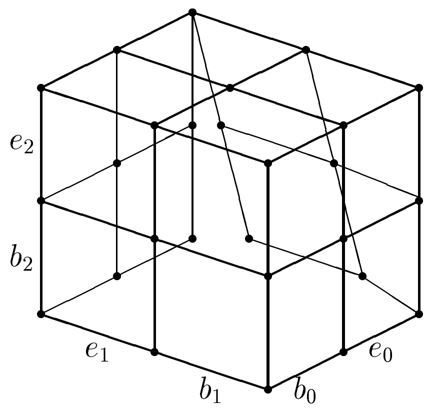


Note the maximal elements corresponding to finite paths — the deadlock points which have been preserved by *Htp*. Note also the boundary elements corresponding to infinite paths.

In the three-dimensional example $\mathcal{V}^{3d\text{-}up}$ from Sec. 2.7 (page 20), cpo $Htp\,\mathcal{V}^{3d\text{-}up}$ is again isomorphic to the configuration space:



This shows that in many dimensions, holes do not necessarily "split" the homotopy cpo. But sometimes they do, as witnessed by $\mathit{Htp}\,\mathcal{V}^{3d\text{-}dn}$:



4 Category of process vectors

To fully exploit the homotopic characterization of processes discussed in Chapter 3, we want not only to translate vectors of processes to cpo-s, but also morphisms between such vectors to continuous maps of cpo-s. And we want this translation to be *functorial*; i.e., to take the identity vector morphism to the identity map of cpo-s, and the composition of vector morphisms to the composition of the respective maps between cpo-s.

The main problem with this programme is that there is no commonly accepted notion of vector morphism. While the definition of vector of processes from Sec. 2 is close to everybody's intuitions, such vectors are usually investigated in isolation from other vectors.

The intuition behind the notion of vector morphism introduced in this chapter is that of the implementation of one system of processes using another system of processes. Any negative results stemming from the translation of vectors with such morphisms to cpo-s with continuous maps will, therefore, pertain to implementability of vectors using other vectors. This is explained in more detail below.

4.1 Consistency preserving

Assume $\mathcal{V} = \langle P, A, L, I\rangle$ and $\mathcal{V}' = \langle P', A', L', I'\rangle$ are vectors of processes. A word-homomorphism $g : A^* \to A'^*$ is *consistency preserving* if $(\sigma g)^{\to}(\mathit{Cons}\,\mathcal{V}) \subseteq \mathit{Cons}\,\mathcal{V}'$ [6].

Proposition 22
Let $\mathcal{V} = \langle P, A, L, I\rangle$ and $\mathcal{V}' = \langle P', A', L', I'\rangle$ be vectors of processes and $g : A^* \to A'^*$ be a consistency preserving word-homomorphism. If $v' \sqsubseteq \sigma g w$ for a certain $w \in \mathit{Cons}\,\mathcal{V}$

[6] Recall from Sec. 2 on p. 14 that $\mathit{Cons}\,\mathcal{V}$ denotes the set of words of actions which are consistent with the vector $\mathcal{V}$.

and $v' \neq \bar{\varepsilon}$ *then there exists a* $w_0 \in Cons\,\mathcal{V}$ *and a single letter* $a \in A$ *such that* $w_0 \frown a \sqsubseteq w$ *and* $\sigma g w_0 \sqsubsetneq v' \sqsubseteq \sigma g(w_0 \frown a)$.

($w \sqsubsetneq u$ means w is strictly less then u: $w \sqsubseteq u$ & $w \neq u$).

4.2 Vector morphisms

Vector morphisms, discussed in this section, may be informally thought of as translating higher-level processes to lower-level ones. A vector morphism consists of two components. The first one, f, contravariantly retrieves from any lower-level process p' the higher-level process fp' in whose implementation p' participates. For any higher-level process p, the coimage $f^{\leftarrow}\{p\} \stackrel{\text{def}}{=} \{p' \in P' \mid fp' = p\}$ may be thought of as consisting of processes which cooperate on a lower-level to achieve the goals originally achieved by p. The other component, g, translates sequences of high-level actions to sequences of low-level actions. The definition of vector morphism that follows contains a number of conditions accounting for consistency between f and g and between the both components and the structure of vectors of processes involved.

But more than this is needed for a pragmatically sound notion of implementation; the vector morphism defined and discussed below is just a “raw material”. Implementations are studied in Sec. 4.3 (page 32).

Given two vectors $\mathcal{V} = \langle P, A, L, I\rangle$ and $\mathcal{V}' = \langle P', A', L', I'\rangle$, a *vector morphism* from $\mathcal{V}$ to $\mathcal{V}'$ is any pair $\langle f, g\rangle : \mathcal{V} \to \mathcal{V}'$ with

- $f : P' \to P$ — (contravariant!) assigning processes to processes,
- $g : A^* \to A'^*$ — word-homomorphism on actions

satisfying the following conditions:

1. $\pi_{A'_{p'}}(\sigma g(Lp)) = \begin{cases} L'p' & \text{if } p = fp' \\ \varepsilon & \text{if } p \neq fp' \end{cases}$ for $p \in P$ and $p' \in P'$.
   Informally, the image via g of the program for a process $fp' \in P'$ is a “zip-join” of programs $L'p''$ for all processes $p'' \in P''$ that implement $f'p'$, i.e. satisfy $f'p'' = f'p'$.
2. $\Box\langle w, v\rangle \subseteq I$ implies $\Box\langle \sigma g w, \sigma g v\rangle \subseteq I'$ for $w, v \in Cons\,\mathcal{V}$.
   Informally, g avoids obstacles in the sense that no obstacles may appear between words $\sigma g w$ and $\sigma g v$ unless they have already been present between w and v.

Proposition 23
If $\langle f, g\rangle : \mathcal{V} \to \mathcal{V}'$ *is a vector morphism then*

$$Alph\,(g^{\rightarrow} A_p) = \bigcup \{A'_{p'} \mid fp' = p\}$$

for any process $p \in P$.

Proof of Proposition 23:

This follows from (2) on page 14 and from condition 1 in the definition of vector morphism.
□

Proposition 24
If $\langle f, g\rangle : \mathcal{V} \to \mathcal{V}'$ is a vector morphism then g is non-glueing for $L^{\to}P$ (cf. the definition of "non-glueing" on p. 9).

Proposition 25
If $\langle f, g\rangle : \mathcal{V} \to \mathcal{V}'$ is a vector morphism then g is consistency preserving.

Given two vector morphisms $\mathcal{V} \xrightarrow{\langle f', g'\rangle} \mathcal{V}' \xrightarrow{\langle f'', g''\rangle} \mathcal{V}''$, their *composition*

$$\langle f, g\rangle = \langle f'', g''\rangle \circ \langle f', g'\rangle : \mathcal{V} \to \mathcal{V}''$$

is defined in a natural way:

$$\begin{cases} f : P'' \to P \\ f \stackrel{\mathrm{def}}{=} f' \circ f'' \end{cases} \qquad \begin{cases} g : A \to A'' \\ g \stackrel{\mathrm{def}}{=} g'' \circ g' \end{cases}$$

Proposition 26
The composition $\langle f, g\rangle$ of morphisms defined as above is a morphism.

Proof of Proposition 26:

Take an arbitrary process $p' \in P'$ and denote the union $\bigcup \left\{ A''_{p''} \,\middle|\, f''p'' = p' \right\}$ by $Q_{p'}$. Since, by virtue of (2) on page 14, the set $\left\{ L'p' \,\middle|\, p' \in P' \right\}$ is disjoint and g'' is non-glueing (cf. Prop. 24), Prop. 9 on page 9 implies that

$$\begin{aligned} &\sigma g'' \circ \pi_{A'_{p'}} = \\ &= \pi_{Alph\,(\sigma g''(L'p'))} \circ \sigma g'' = \\ &= \pi_{Alph\,(g''^{\to}(Alph\,(L'p')))} \circ \sigma g'' = \\ &= \pi_{Alph\,(g''^{\to}A'_{p'})} \circ \sigma g'' = \qquad \text{(by Prop. 23)} \\ &= \pi_{Q_{p'}} \circ \sigma g'' \end{aligned}$$

Assume $p''_0 \in P''$ and apply the above equality to $f''p''_0$ for p':

$$\begin{aligned} &\pi_{A''_{p''_0}} \circ \sigma g'' \circ \pi_{A'_{f''p''_0}} = \\ &= \pi_{A''_{p''_0}} \circ \pi_{Q_{p'}} \circ \sigma g'' = \quad \text{(by Prop. 7, since } A''_{p''_0} \subseteq Q_{p'}\text{)} \\ &= \pi_{A''_{p''_0}} \circ \sigma g'' \end{aligned}$$

Therefore,

$$
\begin{aligned}
L''p_0'' &= && \left(\begin{array}{l}\text{since } \langle f'', g''\rangle \text{ satisfies Cond. 1 from}\\ \text{the definition of vector morphism}\end{array}\right)\\
&= \pi_{A''_{p_0''}}(\sigma g''(L'(f''p_0''))) = && \left(\begin{array}{l}\text{since } \langle f', g'\rangle \text{ satisfies Cond. 1 from}\\ \text{the definition of vector morphism}\end{array}\right)\\
&= \pi_{A''_{p_0''}}(\sigma g''(\pi_{A'_{f''p_0''}}(\sigma g'(L(f'(f''p_0'')))))) = &&\\
&= \pi_{A''_{p_0''}}(\sigma g''(\sigma g'(L(f'(f''p_0''))))) = && \text{(by Prop. 5 on page 8)}\\
&= \pi_{A''_{p_0''}}(\sigma g(L(f p_0''))) &&
\end{aligned}
$$

This proves that $\langle f, g\rangle$ satisfies Cond. 1.

The verification of Cond. 2 is straightforward.

□

For any vector $\mathcal{V}$ of processes define the *identity morphism* $I_{\mathcal{V}} = \langle f_0, g_0\rangle : \mathcal{V} \to \mathcal{V}$ by

$$
\begin{cases} f_0\, p \overset{\text{def}}{=} p \\ g_0\, a \overset{\text{def}}{=} a \end{cases}
$$

Proposition 27
The identity morphism defined above is a vector morphism.

Theorem 2
Vectors of processes with vector morphisms, composition and identity, as defined above, form a category.

Proof of Theorem 2:

By the direct verification of

- associativity of composition of vector morphisms:

$$(\langle f'', g''\rangle \circ \langle f', g'\rangle) \circ \langle f, g\rangle = \langle f'', g''\rangle \circ (\langle f', g'\rangle \circ \langle f, g\rangle)$$

- neutrality of identity wrt composition:

$$\langle f, g\rangle \circ I_{\mathcal{V}} = I_{\mathcal{V}'} \circ \langle f, g\rangle = \langle f, g\rangle$$

  for $\langle f, g\rangle : \mathcal{V} \to \mathcal{V}'$.

□

The category referred to in Thm. 2 is denoted by $\underline{\underline{\mathit{Vect}}}$.

4.3 Implementations

As has already been informally explained in Sec. 4.2, vector morphisms may serve to implementing vectors of processes in other vectors of processes. But for a pragmatically useful notion of implementation, one needs morphisms going both ways: the one from higher to lower level serving as the proper implementation, i.e. translating higher-level notions to lower-level ones; and the other from lower to higher level retrieving the high-level behaviour from its low-level elements.

Formally, a *implementation of* vector $\mathcal{V}$ in vector $\mathcal{V}'$ is any pair of morphisms

$$\begin{tikzcd} \mathcal{V} \arrow[d, bend right, "i"'] \\ \mathcal{V}' \arrow[u, bend right, "r"'] \end{tikzcd}$$

such that $r \circ i = I_{\mathcal{V}}$. Such a pair of morphisms is often called *retraction*. Morphism i serves to translate from high-level $\mathcal{V}$ to low-level $\mathcal{V}'$; morphism r reads the low-level $\mathcal{V}'$-behaviour back in high-level terms.

Given a “pipeline” (h then k) of morphisms and an implementation $\langle i, r\rangle$ of $\mathcal{V}$ in $\mathcal{V}'$:

$$\begin{tikzcd} \mathcal{V}_1 \arrow[r, "h"] & \mathcal{V} \arrow[r, "k"] \arrow[d, bend right, "i"'] & \mathcal{V}_2 \\ & \mathcal{V}' \arrow[u, bend right, "r"'] & \end{tikzcd}$$

the morphisms h and k may be refined to a lower-level pipe:

$$\begin{tikzcd} \mathcal{V}_1 \arrow[r] \arrow[dr, "i \circ h"'] & \mathcal{V} \arrow[r] \arrow[d, bend right] & \mathcal{V}_2 \\ & \mathcal{V}' \arrow[u, bend right] \arrow[ur, "k \circ r"'] & \end{tikzcd}$$

with no externally visible difference in behaviour, since

$$(k \circ r) \circ (i \circ h) = k \circ (r \circ i) \circ h = k \circ I_{\mathcal{V}} \circ h = k \circ h$$

4.4 Implementation of $\mathcal{V}^{cs}$ in $\mathcal{V}^{pc1}$

Consider the complete sequentialization example from Sec. 2.3 on page 16 and the producer and consumer example from Sec. 2.4 on page 17. Define mappings:

$$\begin{cases} f_i : P^{pc1} \to P^{cs} \\ f_i\, p \stackrel{\text{def}}{=} p \end{cases} \qquad \begin{cases} g_i : (A^{cs})^* \to (A^{pc1})^* \\ g_i\, a_0 \stackrel{\text{def}}{=} \mathit{pr\ out} \\ g_i\, a_1 \stackrel{\text{def}}{=} \mathit{in\ co} \\ g_i\, w \text{ — extended homomorphically} \end{cases}$$

and

$$\begin{cases} f_r : P^{cs} \to P^{pc1} \\ f_r\, p \stackrel{\text{def}}{=} \{p\} \end{cases} \qquad \begin{cases} g_r : (A^{pc1})^* \to (A^{cs})^* \\ g_r\, out \stackrel{\text{def}}{=} a_0 \\ g_r\, in \stackrel{\text{def}}{=} a_1 \\ g_r\, a \stackrel{\text{def}}{=} \varepsilon \quad \text{for } a \in \{pr, co\} \\ g_r\, w \text{ — extended homomorphically} \end{cases}$$

It is easy to check that

$$i = \langle f_i, g_i \rangle : \mathcal{V}^{cs} \to \mathcal{V}^{pc1} \quad \text{and} \quad r = \langle f_r, g_r \rangle : \mathcal{V}^{pc1} \to \mathcal{V}^{cs}$$

are vector morphisms. Furthermore,

$$f_i(f_r p) = p$$

and

$$g_r(g_i\, a_0) = g_r(pr\ out) = a_0$$
$$g_r(g_i\, a_1) = g_r(in\ co) = a_1$$

Therefore, $r \circ i = I_{\mathcal{V}^{cs}}$ and pair $\langle i, r \rangle$ is an implementation of vector $\mathcal{V}^{cs}$ in vector $\mathcal{V}^{pc1}$.

5 Functoriality of the cpo constructions

5.1 Functoriality of *Pth*

In this and the next sections, the mappings *Pth* and *Htp* translating process vectors to cpo-s will be extended to functors on $\underline{\underline{Vect}}$. This means, they will be additionally defined to map morphisms of vector processes to continuous functions on the respective cpo-s; and this will be done in a functorial way.

Recall from Sec. 2.1 on p. 15 (or Sec. 3.1 on p. 22) that

$$Pth\, \mathcal{V} \stackrel{\text{def}}{=} \left\{ w \in A^s \,\middle|\, \forall_{v \in A^*}\ v \sqsubseteq w \Rightarrow endpt\, v \in I \right\}$$

$Pth\, \mathcal{V}$ with the prefix ordering is a cpo.

For any vector morphism $\langle f, g \rangle : \mathcal{V} \to \mathcal{V}'$, define the derived mapping on respective path sets:

$$Pth\, \langle f, g \rangle : Pth\, \mathcal{V} \to Pth\, \mathcal{V}'$$
$$Pth\, \langle f, g \rangle \stackrel{\text{def}}{=} \sigma g|_{Pth\, \mathcal{V}}$$

Proposition 28
The definition of $Pth\, \langle f, g \rangle$ *is correct, i.e.* $\sigma g w \in Pth\, \mathcal{V}'$ *for any* $w \in Pth\, \mathcal{V}$.

Proof of Proposition 28:

Assume $w \in Pth\,\mathcal{V}$; we have to check that $\sigma g w \in Pth\,\mathcal{V}'$. Let $v' \in A'^*$ be a prefix of $\sigma g w$, i.e. $v' \sqsubseteq \sigma g w$. By virtue of Prop. 22 on p. 28,

$$\sigma g w_0 \ \sqsubsetneq v' \sqsubseteq \sigma g(w_0 \frown a) \tag{3}$$

for certain $w_0 \in A^*$ and $a \in A$ such that $w_0 \frown a \sqsubseteq w$. Since w is a $\mathcal{V}$-path, the endpoints of its prefixes are in I:

$$I \supseteq \{endpt\, w_0\,,\ endpt\,(w_0 \frown a)\} = \Box\,\langle w_0\,,\ w_0 \frown a\rangle$$

(the last equality comes from Prop. 18 on p. 15). Inequality (3) above and Cond. 2 in the definition of vector morphism (page 29) imply the following:

$$endpt\, v' \in \Box\,\langle \sigma g w_0\,,\ \sigma g(w_0 \frown a)\rangle \subseteq I'$$

This completes the proof that $\sigma g w$ is a $\mathcal{V}'$-path.
□

Proposition 29
$Pth\,\langle f, g\rangle$ is a continuous mapping of cpo-s $Pth\,\mathcal{V}$ and $Pth\,\mathcal{V}'$.

Theorem 3
Pth is a functor from category $\underline{\underline{Vect}}$ to category $\underline{\underline{Cpo}}$.

Pth is the first of the promised functors.

5.2 Functoriality of *Htp*

Recall from Sec. 3.3 on p. 24 that

$$Htp\,\mathcal{V} \ \stackrel{\text{def}}{=}\ Pth\,\mathcal{V}/H$$

where H is the path homotopy relation defined in Sec. 3.2 on pages 22–23; and $Htp\,\mathcal{V}$ is a cpo with the order given by

$$[w_1]_H \sqsubseteq [w_2]_H \ \stackrel{\text{def}}{\Longleftrightarrow}\ \exists_v\ w_1 \sqsubseteq v\, H\, w_2$$

Mapping $Pth\,\langle f, g\rangle : Pth\,\mathcal{V} \to Pth\,\mathcal{V}'$ induces a corresponding mapping on the respective homotopy cpo-s:

$$Htp\,\langle f, g\rangle : Htp\,\mathcal{V} \to Htp\,\mathcal{V}'$$
$$Htp\,\langle f, g\rangle [w]_H \ \stackrel{\text{def}}{=}\ [Pth\,\langle f, g\rangle w]_H$$

Proposition 30
Mapping $Htp\,\langle f, g\rangle$ is well-defined, i.e.

$$w\, H\, v \Rightarrow (Pth\,\langle f, g\rangle w)\, H\, (Pth\,\langle f, g\rangle v)$$

Proof of Proposition 30:

This follows from Prop. 21 (page 23) and from Cond. 2 in the definition of vector morphism (page 29).

□

Proposition 31
Htp $\langle f, g\rangle : Htp\,\mathcal{V} \to Htp\,\mathcal{V}'$ *is a continuous mapping of cpo-s.*

Proof of Proposition 31:

From Prop. 20 on page 23, Prop. 30 above, and Prop. 16 on page 13.

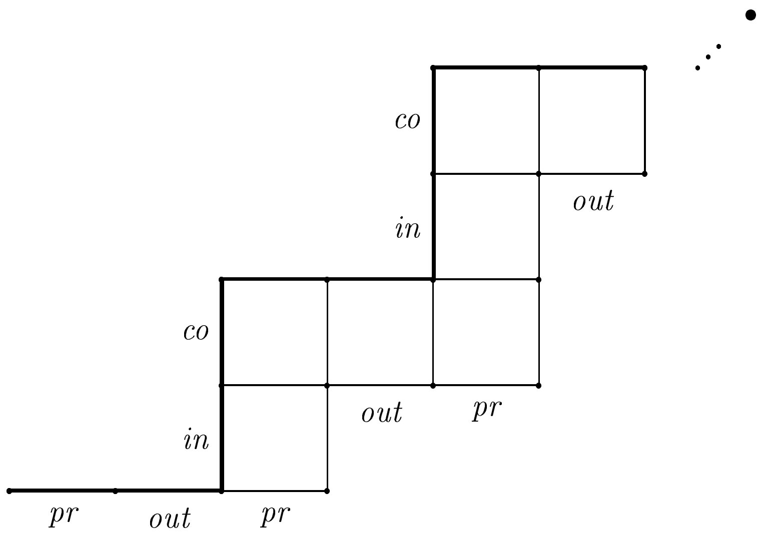
□

Theorem 4
Htp is a functor from category $\underline{\underline{Vect}}$ *to category* $\underline{\underline{Cpo}}$.

Htp is the second functor promised.

5.3 Example

Let us come back to the implementation of $\mathcal{V}^{cs}$ in $\mathcal{V}^{pc1}$ from Sec. 4.4 on p. 32.



The thick line on the picture above corresponds to the image of $Htp\,\mathcal{V}^{cs}$ in $Htp\,\mathcal{V}^{pc1}$ via the derived mapping $Htp\,\langle f_i, g_i\rangle$.

5.4 Application: non-existence of an implementation

The above considerations allow us to give the negative answer to the question:

> Does there exist an implementation of $\mathcal{V}^{ph2}$ in $\mathcal{V}^{pc1}$?

Indeed, assume that

$$\begin{tikzcd} & \mathcal{V}^{pc1} \arrow[dr, "r"] & \\ \mathcal{V}^{ph2} \arrow[ur, "i"] \arrow[rr, "I_{\mathcal{V}^{ph2}}"'] & & \mathcal{V}^{ph2} \end{tikzcd}$$

is an implementation (cf. Sec. 4.3, page 32). By the functoriality of *Htp* (Thm. 4), this commutative diagram of vectors of processes gives rise to the following commutative diagram of cpo-s:

$$\begin{tikzcd} & \mathit{Htp}\,\mathcal{V}^{pc1} \arrow[dr, "\mathit{Htp}\,r"] & \\ \mathit{Htp}\,\mathcal{V}^{ph2} \arrow[ur, "\mathit{Htp}\,i"] \arrow[rr, "I_{\mathit{Htp}\,\mathcal{V}^{ph2}}"'] & & \mathit{Htp}\,\mathcal{V}^{ph2} \end{tikzcd}$$

Recall from Sec. 3.4 (pp. 25–28) that $\mathit{Htp}\,\mathcal{V}^{ph2}$ is a cpo with many distinct maximal points while in $\mathit{Htp}\,\mathcal{V}^{pc1}$, any two points have a common upper bound. Let x_1 and x_2 be distinct maximal points in $\mathit{Htp}\,\mathcal{V}^{ph2}$ and let $y \in \mathit{Htp}\,\mathcal{V}^{pc1}$ be such that

$$(\mathit{Htp}\,i)\,x_1 \sqsubseteq y \qquad \text{and} \qquad (\mathit{Htp}\,i)\,x_2 \sqsubseteq y$$

Then

$$x_1 = (\mathit{Htp}\,r)\,((\mathit{Htp}\,i)\,x_1) \sqsubseteq (\mathit{Htp}\,r)\,y \qquad \text{and analogously}$$
$$x_2 \sqsubseteq (\mathit{Htp}\,r)\,y$$

Since x_1 and x_2 are maximal,

$$x_1 = (\mathit{Htp}\,r)\,y \qquad \text{and} \qquad x_2 = (\mathit{Htp}\,r)\,y$$

This contradicts the assumption that x_1 and x_2 are distinct.

The above proof is analogous to the classical algebraic topological proof that there is no retraction of a solid circle onto its circumference. Indeed, such a retraction would translate to the commutative diagram

$$
\begin{tikzcd}
 & \{0\} \arrow[dr] & \\
Int \arrow[ur] \arrow[rr, "I"'] & & Int
\end{tikzcd}
$$

of groups and group homomorphisms, which is impossible. The nonexistence of such a retraction implies the fixed point property of the solid circle (every continuous function from circle to itself has a fixed point).

6 Final remarks and prospects

This report has introduced functor *Htp* from category $\underline{\underline{\mathit{Vect}}}$ of vectors of processes to category $\underline{\underline{\mathit{Cpo}}}$ of chain-complete partial orders. As an example application, functor *Htp* has then been used to classify some vectors of processes and also to prove that a certain vector is not implementable in another vector. This contribution should be considered another small step toward an incorporation of methods originating from algebraic topology into a theoretical computer scientist's tool kit. Some shortcomings of the approach will, hopefully, lead to improvements.

One of such shortcomings is a rather restrictive concept of the vector of processes. The definition in Sec. 2.1 (p. 13) allows processes to be "straight-line programs" only, without as much as conditional branching, let alone loops or nondeterminism. In a different framework, this problem is successfully addressed in [2] by Fajstrup.

Another shortcoming is an unwieldy shape of the resulting cpo. The reader used to topological homotopy structures will certainly want to identify, say, $\mathit{Htp}\,\mathcal{V}^{cs}$ with $\mathit{Htp}\,\mathcal{V}^{pc1}$ (see Sec 3.4, page 25). Yet the definitions given do not allow to perform such an identification. To address this problem, I have attempted to factor the homotopy cpo by the so called maximal collapse equivalence (cf. [16]). However, I do not know how to do this without compromising the functoriality of the construction, short of imposing extremely severe and unnatural restrictions on vector morphisms.

The homotopy functor seems unsatisfactory for investigating vectors of more than two processes. This should come as no surprise: homotopy cpo-s, as discussed above, investigate just two-dimensional "holes" and this may be too weak when three or more processes are in play. As a motivating example, a server with two printers attached may satisfy two printing requests in the same time but refuse a third request. In this situation, three client processes may be in conflict while no two client processes ever are. More generally, one might want to study a generalization of critical regions, accessible to at most k processes at once. The "holes" in the configuration space are then higher, $(k+1)$-dimensional and require k-dimensional homotopy structures (groups? cpo-s? something else?) for investigation. As of now, I have no results on this.

References

[1] M.A. Bednarczyk. *Categories of Asynchronous Systems*. Ph.D. thesis, Sussex University, CST 1/88, 1988.

[2] L. Fajstrup. Loops, ditopology and deadlocks. Submitted to *Mathematical Structures in Computer Science*, July 1999.

[3] L. Fajstrup, É. Goubault, M. Raußen. Algebraic topology and concurrency. Preprint R-99-2008, Dept. of Mathematical Sciences, Aalborg University, Aalborg, Denmark, June 1999.

[4] L. Fajstrup, M. Raußen. Detecting deadlocks in concurrent systems. BRICS report RS-96-16, Dept. of Computer Science, Aarhus, Denmark, May 1996.

[5] R. van Glabbeek. Bisimulations for higher dimensional automata. E-mail communication available from `http://theory.stanford.edu/~rvg/hda`, Stanford, 1991.

[6] É. Goubault, T. Jensen. Homology of higher-dimensional automata. *CONCUR 92 — Proc. of the 3rd International Conference on Concurrency Theory, Stonybrook, New York*, 1992.

[7] J. Gunawardena. Homotopy and concurrency. *Bulletin of the EATCS*, Vol. 54, pp. 184–193, October 1994.

[8] M. Herlihy. *A Tutorial on Algebraic Topology and Distributed Computation*, Brown University, Providence, USA, Version of October 1997.

[9] M. Herlihy, S. Rajsbaum. Algebraic spans. *Proc. of the 14th Annual ACM Symp. on Principles of Distributed Systems*, Ottawa, pp. 90–99, August 1995.

[10] M. Herlihy, S. Rajsbaum. Algebraic topology and distributed computing. A primer. *Computer Science Today: Recent Trends and Developments*, LNCS 1000, pp. 203–217, 1995.

[11] M. Herlihy, N. Shavit. *The Topological Structure of Asynchronous Computability*. Brown University, Providence, USA, January 1996.

[12] G. Hoest, N. Shavit. Towards a topological characterization of asynchronous complexity. *Proc. of the 16th ACM Symp. on Principles of Distributed Computing*, Santa Barbara, California, USA, August 1997.

[13] A. Mazurkiewicz. Concurrent program schemes and their interpretations. Report DAIMI PB-78, Dept. of Computer Science, University of Århus, 1977.

[14] V. Pratt. Modeling concurrency with geometry. *Proc. of the 18th Annual ACM Symp. on Principles of Programming Languages*, pp. 311–322, January 1991.

[15] S. Sokołowski. Homotopy in concurrent processes. ICS PAS Report 854, 49 pages, March 1998. Electronically available at `ftp://ftp.ipipan.gda.pl/stefan/reports/58-homotopy.ps.gz`.

[16] S. Sokołowski. Point glueing in cpo-s. ICS PAS Report 863, 21 pages, June 1998. Electronically available at `ftp://ftp.ipipan.gda.pl/stefan/reports/60-partord-glue.ps.gz`.