Fault-tolerant Protocols and Directed Algebraic Topology

Eric Goubault
CEA LIST, Ecole Polytechnique

ATMCS, Edinburgh, 4th of July 2012

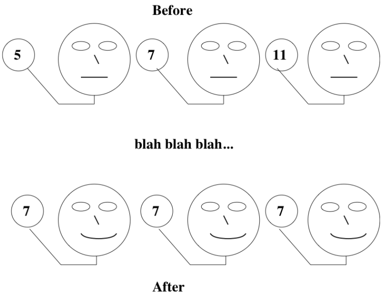
Contents of the talk

- An introduction to fault-tolerants protocols for distributed systems (à la Maurice Herlihy and Sergio Rajsbaum et al.)
- Trace spaces in the unlooping case (for loops, ask Lisbeth ;-)
- Links between the two approaches and future work

(ongoing work, with lots of inputs from Samuel Mimram, Emmanuel Haucourt, Christine Tasson, Lisbeth Fajstrup, Martin Raussen!)

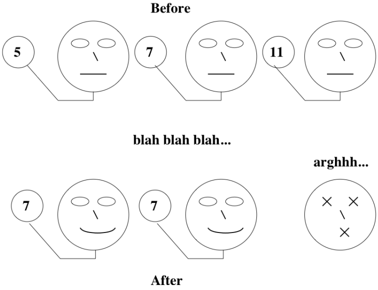
Can we implement a function...given an “architecture” (faults? shared memory / message passing, synchronous / semi-synchronous / asynchronous etc.)?

Example: consensus

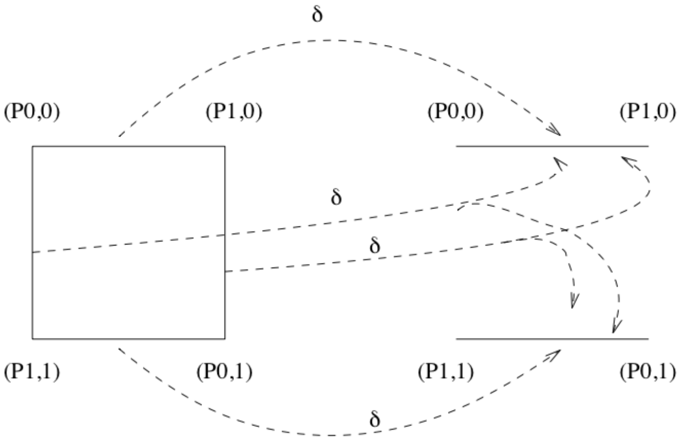


Can we implement a function...given an "architecture" (faults? shared memory / message passing, synchronous / semi-synchronous / asynchronous etc.)?

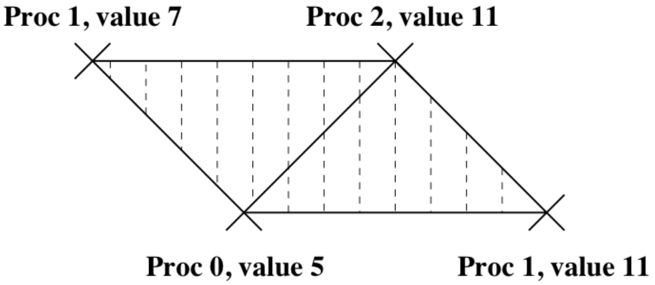
Even if...



Example: binary consensus specification

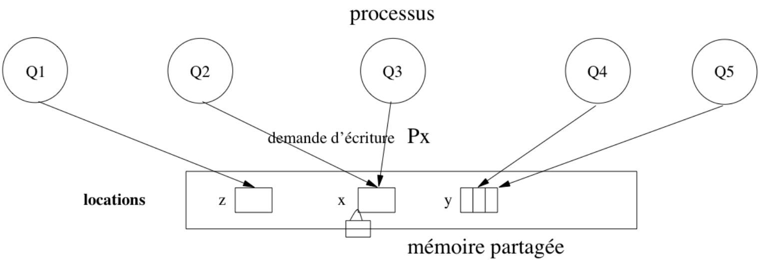


More generally: Simplicial model of states



Context of this talk

- We consider in this talk *concurrent programs interacting through shared memory*



- Synchronisation:
  - through semaphores (P for locking, V for unlocking), binary or "counting"
  - Or synchronisation through *scan*/*update*

Update-scan model

Update-scan model, very close to the PV model:

- The concurrent machine with n processes has a global memory (X_i, $i = 0, \ldots, n-1$)
- Each process P_j has a local mirror of the global memory and a local "decision value" x_j
- It can *update* the value of x_j in the *mirror* variable X_j in global memory: no conflict on writes!
- It can *scan* all of the global memory into its local memory ($x_i \leftarrow X_i$, $i = 0, \ldots, n-1$)
- It can perform local computations...

Processes are supposed to do (update; computation; scan)* in parallel

Protocol complex

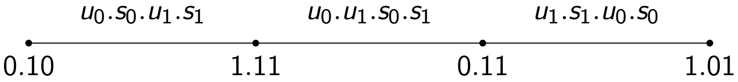
Main tool for proving impossibility results about decision tasks: represents reachable states at some round of execution

Generic protocol: all processes do $(update; scan)^*$ in parallel. This defines:

- a simplicial set (for all rounds r):
  - vertices: sequence of "values" scanned at a given round r
  - simplices: compound states at round r
- This is an operator on an input simplex
- A choice of model of computation entails some geometrical properties of the protocol complex

One-round protocol simplicial set (2D)

P_0 and P_1 do just *update*; *scan* each; there are only 3 possible end states when there is no failure (the 3 edges):



- First digit is the process number (identifying the local state)
- After the dot, for each round, we get a string of n bits, where n is the number of processes involved (here just one round, and $n = 2$)
- Bit 1 at position i for processor P_j means that P_j "knows" P_i (by convention, P_j knows itself)

One-round protocol simplicial set (3D)

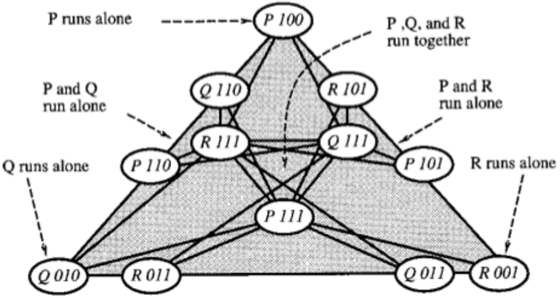


FIG. 25. A one-round protocol complex.

How can we find such pictures?

Theorems (Herlihy, Rajsbaum et al.)

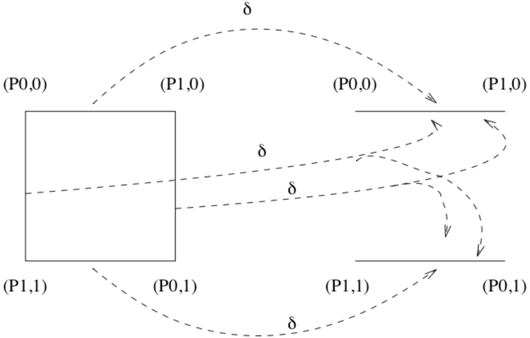
Or more precisely, how can we prove "from first principles"?:

- Wait-free read/write protocol complexes are:
  - $(n-1)$-connected
  - no matter how long the protocol runs

- From that topological characterization, the "computation power" of the scan/update models follows "easily":
- A task has a wait-free read/write protocol if and only if there exists a simplicial map μ:
  - from subdivided input complex
  - to output complex
  - that respects δ

Applications - among others

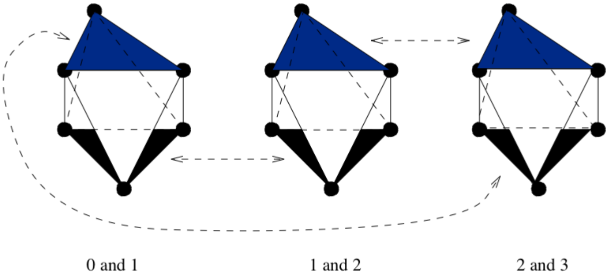
- Application: impossibility for consensus (in the presence of 1 fault already)
- Recall:



Computations at each round go through connected complexes...

Applications - among others

- And more generally, impossibility of k-set agreement: processes must end up with at most k different values (taken from the initial values)
- Example: output simplicial set ($n = 3$, $k = 2$)



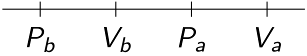
3 spheres glued together minus the simplex formed of all 3 values: not 2-connected

Partial conclusion, at this point

- Homotopy types of some combinatorial object (protocol complex) count!
- But where does the protocol complex comes from?
- Of course from the semantics of some programs! $\longrightarrow$ geometric semantics and directed algebraic topology
- How can we make the link?
- How can we generalize this to more intricate distributed models, than scan/update?

A program P_b;`x:=1`;V_b;P_a;`y:=2`;V_a | P_a;`y:=3`;V_a will be interpreted as a **directed space**:

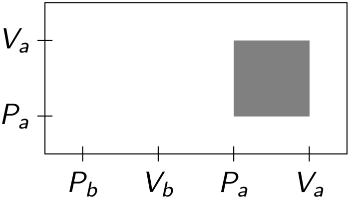
- $P_b.V_b.P_a.V_a$



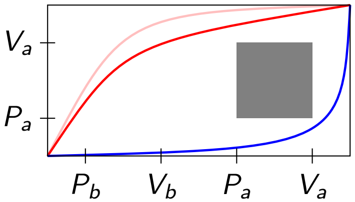
- $P_a.V_a$

- $P_b.V_b.P_a.V_a \quad | \quad P_a.V_a$

Forbidden regions



A **scheduling** is the (di-)homotopy class of a (di-)path.



We want to compute *a path in every scheduling*. Can be scheduled combinatorially:

`y:=3;x:=1;y:=2` $(x,y)=(1,2)$

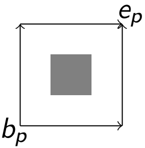
`x:=1;y:=3;y:=2` $(x,y)=(1,2)$

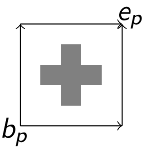
`x:=1;y:=2;y:=3` $(x,y)=(1,3)$

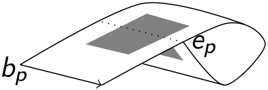
But just 2 (di-)homotopy classes.

To each program p we associate a directed space of some sort (d-space, stream etc.):

$P_a.V_a|P_a.V_a$ $\quad$ $P_a.P_b.V_b.V_a|P_b.P_a.V_a.V_b$ $\quad$ $P_a.(V_a.P_a)^*|P_a.V_a$

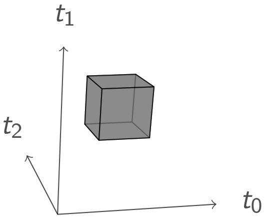




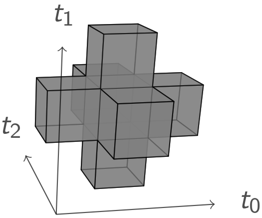


Examples of PV semantics

$$P_a.V_a|P_a.V_a|P_a.V_a$$
$$(\kappa_a = 2)$$

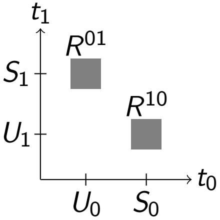


$$P_a.V_a|P_a.V_a|P_a.V_a$$
$$(\kappa_a = 1)$$

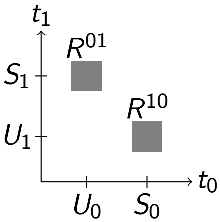


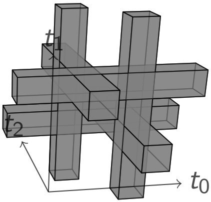
Examples of *scan*/*update* semantics

Only "obstructions" are between *scan* and *update*:



Only "obstructions" are between *scan* and *update*:





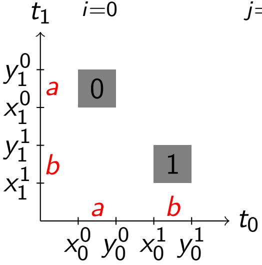
In dimension n, the forbidden region consists of n crosses with $n-1$ orthogonal branches.

For this talk - just po-spaces

Suppose given a program with n *threads* $p = p_0|p_1|\ldots|p_{n-1}$
Under mild assumptions, the geometric semantics is of the form

$$G_p = \vec{I}^n \setminus \bigcup_{i=0}^{l-1} R^i;\ R^i = \prod_{j=0}^{n-1}]x_j^i, y_j^i[$$

Example:



More formally

Basic definitions in directed algebraic topology

- Let X be a stream/d-space etc. (here we only consider a po-space, i.e. a topological space X together with a partial order $\leqslant \subseteq X \times X$, closed in the product topology)
- $p : \mathbf{I} \rightarrow X$ a continuous and increasing path from po-space $\mathbf{I} = ([0,1], \leqslant)$ (standard order) to X is a *directed path*
- Define the path space $\mathbf{P}(X)(a,b) = \{p : \mathbf{I} \rightarrow X \mod p(0) = a, p(1) = b, p$ is a directed path$\}$
- A *dihomotopy* on $\mathbf{P}(X)(a,b)$ is a continuous map $H : \mathbf{I} \times I \rightarrow X$ such that $H_t \in \mathbf{P}(X)(a,b)$ for all $t \in I$.

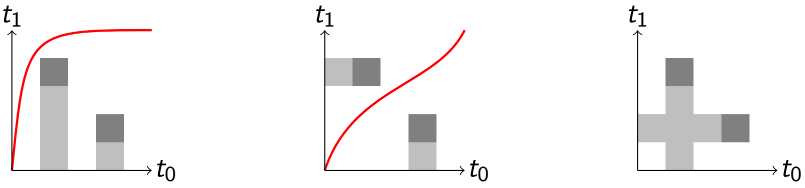
Formally

- Let X be a stream/d-space etc.
- Define the *trace space* $\mathbf{T}(X)(a,b)$ to be the path space between a and b modulo continuous and increasing reparametrizations
- Schedules are dihomotopy classes of dipaths ($\pi_0(\mathbf{T}(X))$)

- We wish to study the homotopy type of $\mathbf{T}(X)(a,b)$, since this has to be related to the protocol complex somehow
- There is a homotopy equivalence between $\mathbf{T}(X)(a,b)$ and a certain prodsimplicial complex (Martin Raussen), which can be calculated combinatorially, on our simple semantics...

Determining trace spaces, combinatorially

The main idea is to extend the forbidden cubes downwards in various directions and look whether there is a path from b to e in the resulting space.



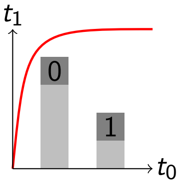
By combining those information, we will be able to compute traces modulo homotopy.

The directions in which to extend the holes will be coded by boolean matrices M.

The index poset

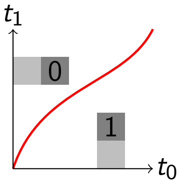
$\mathcal{M}_{l,n}$: boolean matrices with l rows and n columns.

X_M: space obtained by *extending*
for every (i, j) such that $M(i, j) = 1$
the forbidden cube i downwards
in every direction other than j



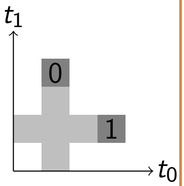
$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

alive



$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

alive

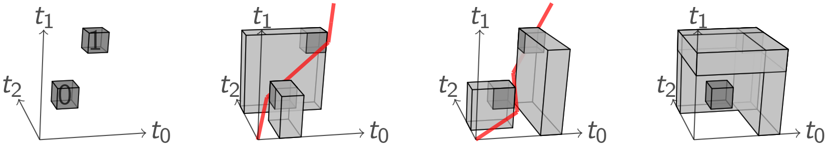


$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

dead

The index poset, combinatorially

$P_a.V_a.P_b.V_b \quad | \quad P_a.V_a.P_b.V_b \quad | \quad P_a.V_a.P_b.V_b$



$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ $\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$ $\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$

alive *alive* *alive* *dead*

Alive and dead?

Important matrices are

- the **dead poset** $D(X) = \{M \in \mathcal{M}^C_{l,n} \ / \ M \text{ dead}\}$.
- the **index poset** $\mathcal{C}(X) = \{M \in \mathcal{M}^R_{l,n} \ / \ M \text{ is alive}\}$.
- consider the entrywise ordering $(0 < 1)$ on matrices.

General results by Martin Raussen:

$D(X) \quad \rightsquigarrow \quad \mathcal{C}(X) \quad \rightsquigarrow \quad$ homotopy classes of traces

(and even more, but let us just start with that!)

The dead poset

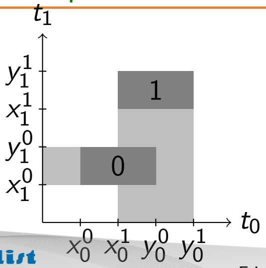
Proposition

A matrix $M \in \mathcal{M}^C_{l,n}$ is in $D(X)$ iff it satisfies

$$\forall (i,j) \in [0:l[\times[0:n[, \qquad M(i,j)=1 \quad \Rightarrow \quad x^i_j < \min_{i' \in R(M)} y^{i'}_j$$

where $R(M)$: indexes of non-null rows of M.

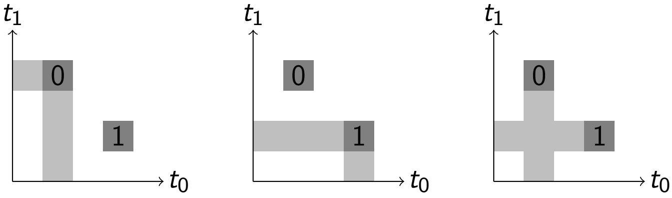
Example



$$M = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{matrix} x_1^0 = 1 < 2 = \min(y_1^0, y_1^1) \\ x_0^1 = 2 < 3 = \min(y_0^0, y_0^1) \end{matrix}$$

Example, scan/update in dimension 2

3 dead matrices



Proposition

A matrix M is in $\mathcal{C}(X)$ iff for every $N \in D(X)$, $N \nleq M$.

Remark

$N \nleq M$: *there exists* (i,j) *s.t.* $N(i,j) = 1$ *and* $M(i,j) = 0$.

Remark

Since $\mathcal{C}(X)$ is downward closed it will be enough to compute the set $\mathcal{C}_{\max}(X)$ of maximal alive matrices.

Some combinatorial considerations

Hypergraph transversal

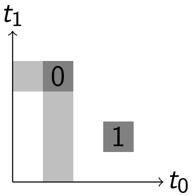
- An *hypergraph* $H = (V, E)$ consists of a set V of *vertices* and a set E of edges, where an *edge* is a subset of V
- A *transversal* T of H is a subset of V such that $T \cap e \neq \emptyset$ for every edge $e \in E$.

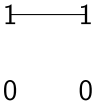
$D(X) \Rightarrow$ hypergraph H:

- vertices: $[0 : l[\times[0 : n[$
- hyperedges: $\{(i,j) \ / \ D(i,j) = 1\}$ (D is a matrix in $D(X)$)

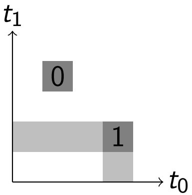
The sets $\{(i,j) \ / \ M(i,j) = 0\}$, where M is a maximal matrix of $\mathcal{C}(X)$, correspond to *minimal transversals* (wrt inclusion order) of H.

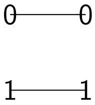
First dead matrix:



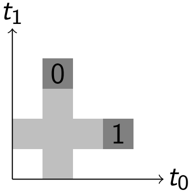


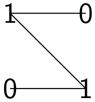
Second dead matrix:



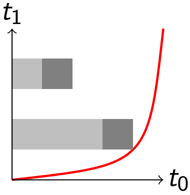


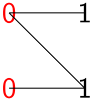
Third and last (minimal) dead matrix:





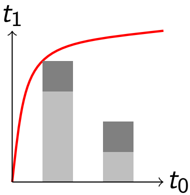
First (maximal) alive matrix:

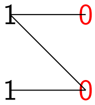




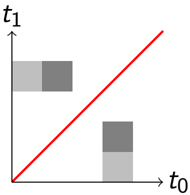
Some combinatorial considerations

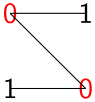
Second alive matrix:





Third (and last) maximal alive matrix:





Definition

Two matrices M and N are **connected** when $M \wedge N$ does not contain any null row. ($M \wedge N$: pointwise min of M and N)

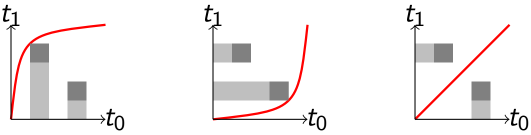
Proposition

The connected components of $\mathcal{C}(X)$ are in bijection with homotopy classes of traces $b \to e$ in X.

Scan/update in dimension 2 - 1 round

$$u.s \mid u.s$$

generates a trace space with 3 components:

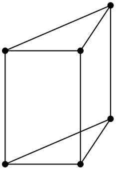


$$M_1 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \quad M_2 = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \quad M_3 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Trace spaces: prodsimplicial structure

In fact, there is more topological structure from the matrices...

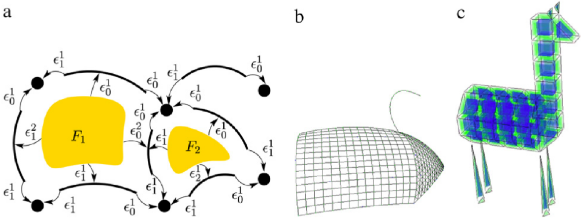
- A prod-simplicial space is just a space made up of simplices, and products of simplices, glued together along their faces (natural generalization of cubical and simplicial sets)



Trace spaces: prodsimplicial structure

In fact, there is more topological structure from the matrices...

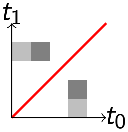
- A prod-simplicial space is just a space made up of simplices, and products of simplices, glued together along their faces (natural generalization of cubical and simplicial sets)
- Example:



Prodsimplicial structure of trace spaces

Each matrix of $\mathcal{C}$ represents a prod-simplex, product of one n-simplex per line, n=number of 1 per line minus 1...

Recall:



$$M_3 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

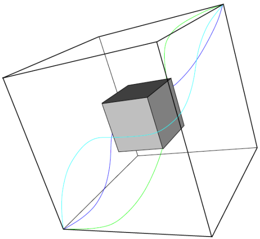
product of 2 0-simplices = point!

In fact, theorem:

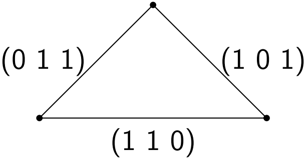
The prodsimplicial set corresponding to the scan/update model, in any dimension n, for one round, is discrete (all alive matrices have exactly one 1 on each line!).

Prodsimplicial structure of trace spaces

Each matrix of $\mathcal{C}$ represents a prod-simplex, product of one n-simplex per line, n=number of 1 per line minus 1...

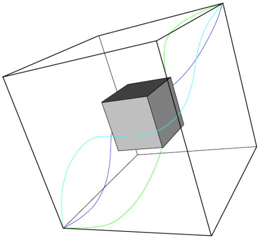


- $\mathcal{D}(X)(0,1) = \{(111)\}$
- $\mathcal{C}(X)(0,1) = \{(110),(101),(011)\}$
-

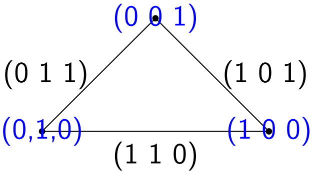


Prodsimplicial structure of trace spaces

Each matrix of $\mathcal{C}$ represents a prod-simplex, product of one n-simplex per line, n=number of 1 per line minus 1...

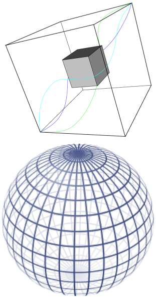


- $\mathcal{C}(X)(0,1) = \{(110),(101),(011)\}$
- and common faces are meet of matrices

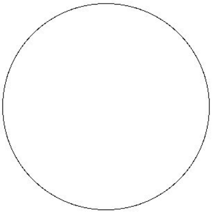


Prodsimplicial structure of trace spaces

Each matrix of $\mathcal{C}$ represents a prod-simplex, product of one n-simplex per line, n=number of 1 per line minus 1...

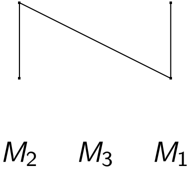


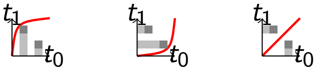
- $\mathcal{C}(X)(0,1) = \{(110),(101),(011)\}$
- connected, not simply-connected (reflecting the fact that $\pi_2(X) = \mathbb{Z}$)



Minimal transversal hypergraph

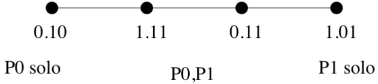
...is itself an hypergraph (same vertices, but hitting sets as hyper-edges):





$$\underset{M_1}{\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}} \quad \underset{M_2}{\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}} \quad \underset{M_3}{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}}$$

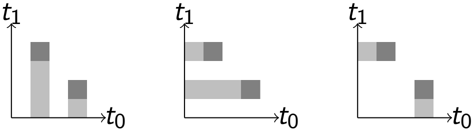
Looks like protocol complex already!



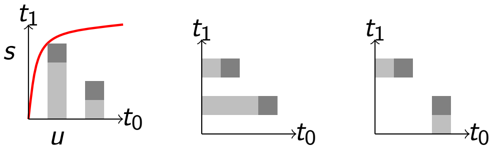
Link with protocol complexes

- Where does this "coincidence" comes from?
- What about in higher dimension?
- What about with more rounds? with differents models?

What is the *meaning* of traces?



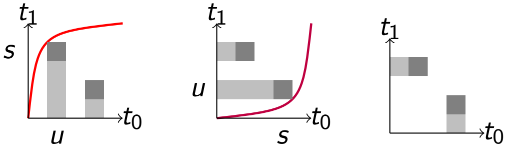
$$\underset{M_1}{\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}} \qquad \underset{M_2}{\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}} \qquad \underset{M_3}{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}}$$



$$M_1 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \qquad M_3 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- M_1: P_1 does its scan before P_0 does its update
- M_1: P_1 *does not know* the current value of P_0 but P_0 does

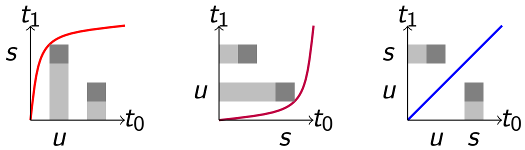
What is the *meaning* of traces?



$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

M_1 M_2 M_3

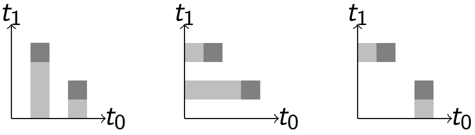
- M_1: P_1 does its scan before P_0 does its update
- M_2: P_0 does its scan before P_1 does its update
- M_1: P_1 *does not know* the current value of P_0 but P_0 does
- M_2: P_0 *does not know* the current value of P_1 but P_1 does



$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$M_1 \qquad M_2 \qquad M_3$$

- M_1: P_1 does its scan before P_0 does its update
- M_2: P_0 does its scan before P_1 does its update
- M_3: P_0 and P_1 do update, then do there scan together
- M_1: P_1 *does not know* the current value of P_0 but P_0 does
- M_2: P_0 *does not know* the current value of P_1 but P_1 does
- M_3: P_0 and P_1 *know* their values



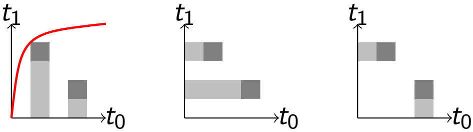
$$\begin{pmatrix}1 & 0\\ 1 & 0\end{pmatrix} \quad \begin{pmatrix}0 & 1\\ 0 & 1\end{pmatrix} \quad \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$$

Protocol complex:

$$1.01 \longrightarrow 0.11 \longrightarrow 1.11 \longrightarrow 0.10$$

(1.01 (resp. 0.10) means P_1 (resp. P_0) knows only its own value; 1.11 (resp. 0.11) means P_1 (resp. P_0) knows all values)

Link with the protocol complex



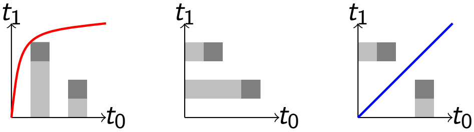
$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

M_1

Protocol complex:

$$1.01 \xrightarrow{M_1} 0.11 \longrightarrow 1.11 \longrightarrow 0.10$$

Link with the protocol complex



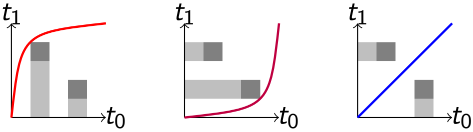
$$\underset{\color{red}{M_1}}{\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}} \qquad \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \qquad \underset{\color{blue}{M_3}}{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}}$$

Protocol complex:

$$1.01 \xrightarrow{M_1} 0.11 \xrightarrow{M_3} 1.11 \longrightarrow 0.10$$

M_3 differs from M_1 by just a 1 (connected)

Link with the protocol complex



$$\underbrace{\begin{pmatrix}1 & 0\\ 1 & 0\end{pmatrix}}_{M_1} \qquad \underbrace{\begin{pmatrix}0 & 1\\ 0 & 1\end{pmatrix}}_{M_2} \qquad \underbrace{\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}}_{M_3}$$

Protocol complex:

$$1.01 \xrightarrow{M_1} 0.11 \xrightarrow{M_3} 1.11 \xrightarrow{M_2} 0.10$$

M_3 differs from M_2 by just a 1 (connected)

Interval posets and schedules

Interval posets

- Let S be a set of closed intervals in $\mathbb{R}$ (i.e. of elements of the form $[a, b]$, a, b in $\mathbb{R}$)
- We define the partial order:

$$[a, b] \leqslant [c, d] \Leftrightarrow b \leqslant c$$

  (plus reflexivity)
- $(S, \leqslant)$ is called an interval poset

- Are very well described, combinatorially
- For instance Fishburn's theorem (equivalence with (2+2)-free posets)
- And number of such posets on n elements is well known, example: 1,3,19,207,3451,... (this is A079144 on OEIS)

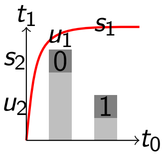
The dihomotopy classes of maximal paths, for the 1-round scan/update model for n processes, is in bijection with the interval posets on n elements.

The bijection associates to each dihomotopy class $[p]$ the set of intervals in $[0,1]$

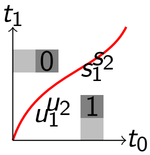
$$(p \circ \pi_i)^{-1}([u_i, s_i])$$

$(i = 1, \ldots, n)$

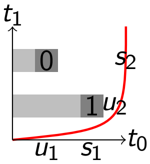
Proof relies on the characterization of dihomotopy classes through alive matrices, hence dead matrices - recall condition on being dead, as some interval inequalities!



$[u_2, s_2] < [u_1, s_1]$



$[u_1, s_1], [u_2, s_2]$



$[u_2, s_2] < [u_1, s_1]$

19 alive matrices (for 53 dead ones)

0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	0	1	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	1	0	0	1	0
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0	0	1	0
1	0	0	1	0	0	0	0	1	0	0	1	0	0	1
1	0	0	0	1	0	1	0	0	0	1	0	0	1	0
0	1	0	0	1	0	0	1	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	1	0	0	0	0	1
0	1	0	0	1	0	0	1	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0	0	1			
1	0	0	0	0	1	0	0	1	0	0	1			
0	0	1	0	0	1	0	0	1	0	0	1			
1	0	0	1	0	0	1	0	0	0	1	0			
0	0	1	0	0	1	0	0	1	0	0	1			
0	1	0	1	0	0	0	1	0	0	1	0			

Reorganizing things a bit...

0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	0	1	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	1	0	0	1	0
4	2	0	3	3	0	2	3	1	2	4	0	1	4	1

0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0	0	1	0
1	0	0	1	0	0	0	0	1	0	0	1	0	0	1
1	0	0	0	1	0	1	0	0	0	1	0	0	1	0
4	1	1	3	2	1	3	1	2	2	2	2	1	3	2

0	1	0	0	1	0	0	1	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	1	0	0	0	0	1
0	1	0	0	1	0	0	1	0	1	0	0	1	0	0
0	4	2	1	2	3	0	3	3	4	0	2	3	0	3

0	0	1	0	0	1	0	0	1	0	0	1			
1	0	0	0	0	1	0	0	1	0	0	1			
0	0	1	0	0	1	0	0	1	0	0	1			
1	0	0	1	0	0	1	0	0	0	1	0			
0	0	1	0	0	1	0	0	1	0	0	1			
0	1	0	1	0	0	0	1	0	0	1	0			
2	1	3	2	0	4	1	1	4	0	2	4			

1 of symmetry type (2,2,2)

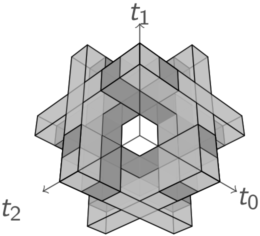
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	0	1	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	1	0	0	1	0
4	2	0	3	3	0	2	3	1	2	4	0	1	4	1

0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0	0	1	0
1	0	0	1	0	0	0	0	1	0	0	1	0	0	1
1	0	0	0	1	0	1	0	0	0	1	0	0	1	0
4	1	1	3	2	1	3	1	2	2	2	2	1	3	2

0	1	0	0	1	0	0	1	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	1	0	0	0	0	1
0	1	0	0	1	0	0	1	0	1	0	0	1	0	0
0	4	2	1	2	3	0	3	3	4	0	2	3	0	3

0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	0	1	0
2	1	3	2	0	4	1	1	4	0	2	4

Example, in dimension 3



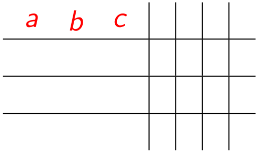
Hasse diagram of the corresponding interval poset:

$$[u_0, s_0] \quad [u_1, s_1] \quad [u_2, s_2]$$

(let us call $a = [u_0, s_0]$, $b = [u_1, s_1]$, $c = [u_2, s_2]$ for the sequel)

Corresponding the labelled interval posets on 3 elements (19 of them)

(2,2,2)



6 of symmetry type (3,2,1)

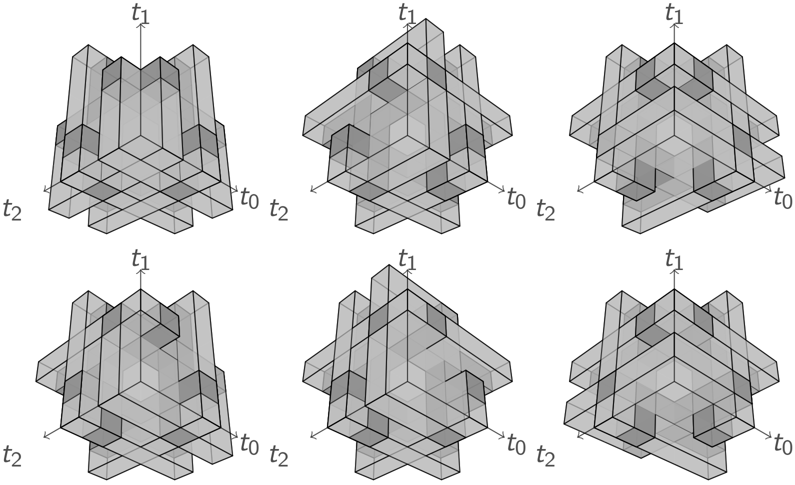
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	0	1	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	1	0	0	1	0
4	2	0	3	3	0	2	3	1	2	4	0	1	4	1

0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0	0	1	0
1	0	0	1	0	0	0	0	1	0	0	1	0	0	1
1	0	0	0	1	0	1	0	0	0	1	0	0	1	0
4	1	1	3	2	1	3	1	2	2	2	2	1	3	2

0	1	0	0	1	0	0	1	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	1	0	0	0	0	1
0	1	0	0	1	0	0	1	0	1	0	0	1	0	0
0	4	2	1	2	3	0	3	3	4	0	2	3	0	3

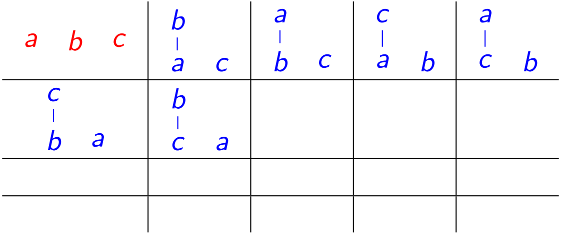
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	0	1	0
2	1	3	2	0	4	1	1	4	0	2	4

In pictures



Corresponding the labelled interval posets on 3 elements (19 of them)

(3,2,1)



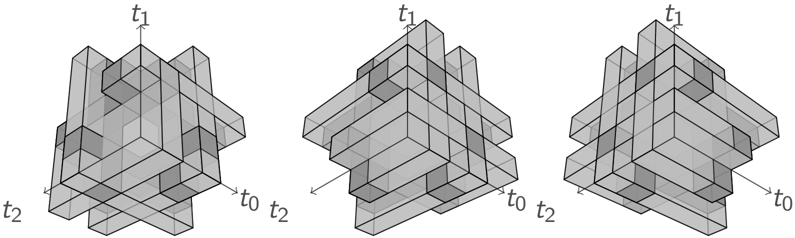
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	0	1	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	1	0	0	1	0
4	2	0	3	3	0	2	3	1	2	4	0	1	4	1

0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0	0	1	0
1	0	0	1	0	0	0	0	1	0	0	1	0	0	1
1	0	0	0	1	0	1	0	0	0	1	0	0	1	0
4	1	1	3	2	1	3	1	2	2	2	2	1	3	2

0	1	0	0	1	0	0	1	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	1	0	0	0	0	1
0	1	0	0	1	0	0	1	0	1	0	0	1	0	0
0	4	2	1	2	3	0	3	3	4	0	2	3	0	3

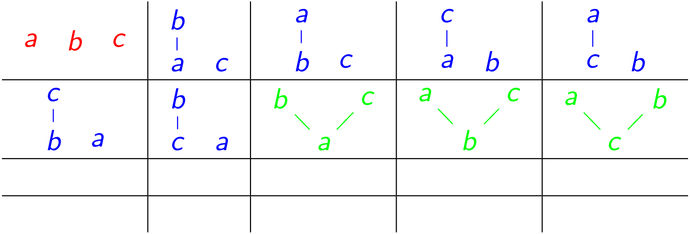
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	0	1	0
2	1	3	2	0	4	1	1	4	0	2	4

In picture



Corresponding the labelled interval posets on 3 elements (19 of them)

(3,3,0)



3 of symmetry type (4,1,1)

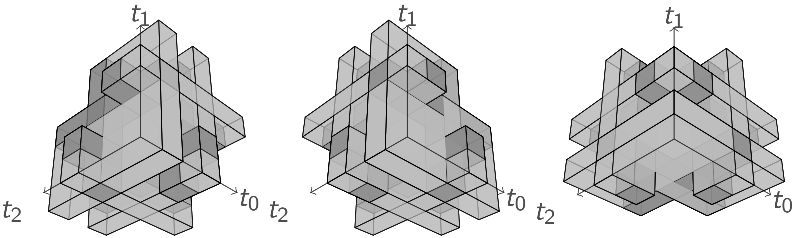
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	0	1	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	1	0	0	1	0
4	2	0	3	3	0	2	3	1	2	4	0	1	4	1

0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0	0	1	0
1	0	0	1	0	0	0	0	1	0	0	1	0	0	1
1	0	0	0	1	0	1	0	0	0	1	0	0	1	0
4	1	1	3	2	1	3	1	2	2	2	2	1	3	2

0	1	0	0	1	0	0	1	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	1	0	0	0	0	1
0	1	0	0	1	0	0	1	0	1	0	0	1	0	0
0	4	2	1	2	3	0	3	3	4	0	2	3	0	3

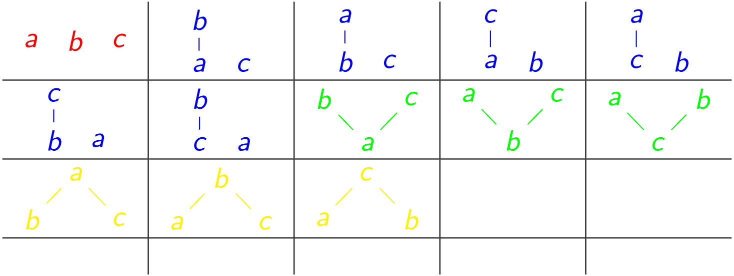
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	0	1	0
2	1	3	2	0	4	1	1	4	0	2	4

In pictures



Corresponding the labelled interval posets on 3 elements (19 of them)

(4,1,1)



6 of symmetry type (4,2,0)

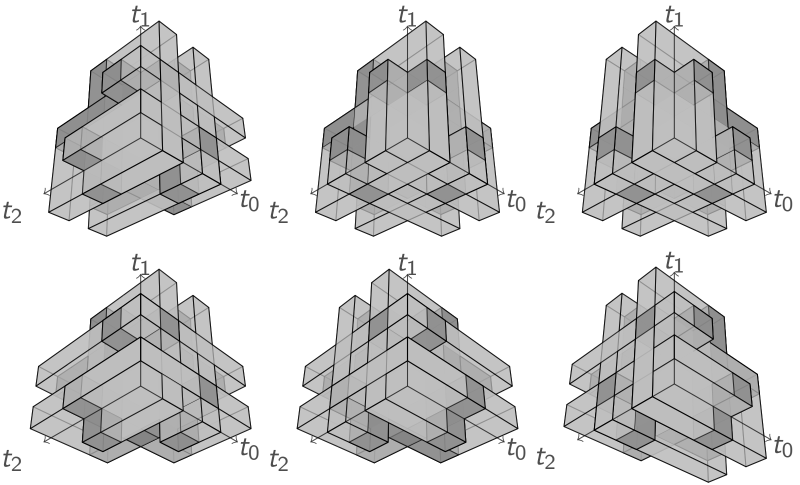
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	0	1	1	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	1	0	0	1	0
4	2	0	3	3	0	2	3	1	2	4	0	1	4	1

0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0	0	1	0
1	0	0	1	0	0	0	0	1	0	0	1	0	0	1
1	0	0	0	1	0	1	0	0	0	1	0	0	1	0
4	1	1	3	2	1	3	1	2	2	2	2	1	3	2

0	1	0	0	1	0	0	1	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	1	0	0	0	0	1
0	1	0	0	1	0	0	1	0	1	0	0	1	0	0
0	4	2	1	2	3	0	3	3	4	0	2	3	0	3

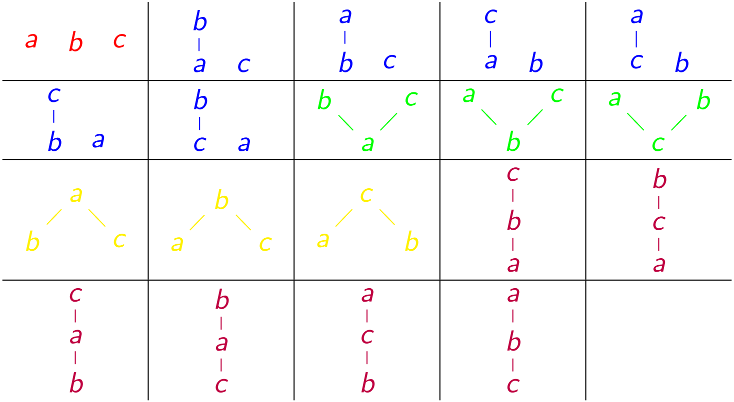
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1	0	0	1	0
2	1	3	2	0	4	1	1	4	0	2	4

In pictures



Corresponding the labelled interval posets on 3 elements (19 of them)

(4,2,0)



What is the structure of the protocol complex now?

Extension order on posets

Let $(S_1 \leqslant_1)$ and $(S_2, \leqslant_2)$ be two partial order on some sets $S_1 \subseteq S_2$. We say that $\leqslant_1 \Rightarrow \leqslant_2$ if $\forall s, t \in S_1$, $s \leqslant_1 t \Rightarrow s \leqslant_2 t$.

When $S_1 = S_2$, this is the linearization order.

Importance of the extension order for our purpose

Let $\leqslant_1$ and $\leqslant_2$ be interval orders on the same set of cardinal $n+1$. If $\leqslant_1$ is a linearization of $\leqslant_2$ then the corresponding n-simplexes share a common $(n-1)$ face.

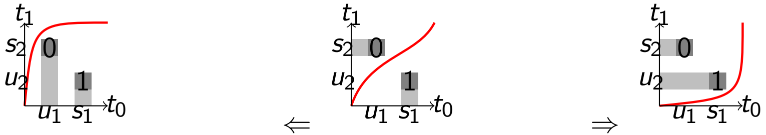
In fact, the face poset of the protocol complex is given by the extension order on interval posets up to n elements

Logical interpretation

Each interval can be interpreted in terms of “knowledge”, hence the structure of the protocol complex...

$a \quad b \quad c$ $\{0.111, 1.111, 2.111\}$	b $\vert$ $a \quad c$ $\{0.101, 1.111, 2.111\}$
a $\vert$ $b \quad c$ $\{0.111, 1.011, 2.111\}$	c $\vert$ $a \quad b$ $\{0.110, 1.111, 2.111\}$
a $\vert$ $c \quad b$ $\{0.111, 1.111, 2.011\}$ $\ldots$	c $\vert$ $b \quad a$ $\{0.111, 1.110, 2.111\}$ $\ldots$

Example, in dimension 2



$[u_2, s_2] < [u_1, s_1]$ $\quad$ $[u_1, s_1], [u_2, s_2]$ $\quad$ $[u_2, s_2] < [u_1, s_1]$

M_1 $\quad$ M_3 $\quad$ M_2

Protocol complex:

$$1.01 \xrightarrow{M_1} 0.11 \xrightarrow{M_3} 1.11 \xrightarrow{M_2} 0.10$$

Structure of the protocol complex

Corollary

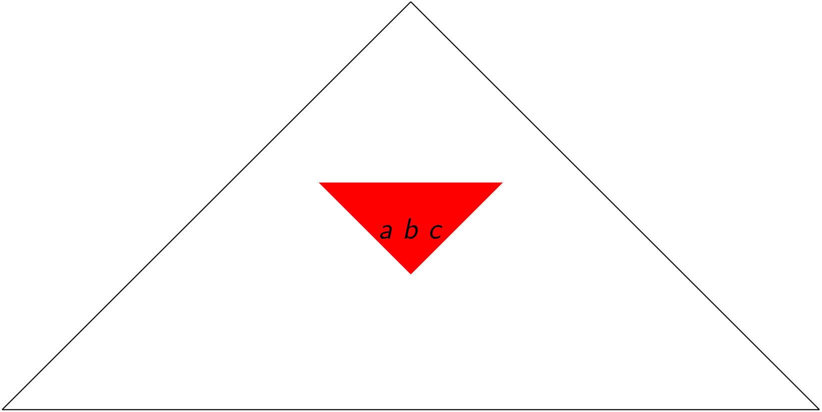
The protocol complex for scan/update in dimension n, for one round, is homotopy equivalent to the order complex for the extension order on interval posets up to n elements.

(since the order complex of the face poset is just the barycentric subdivision)

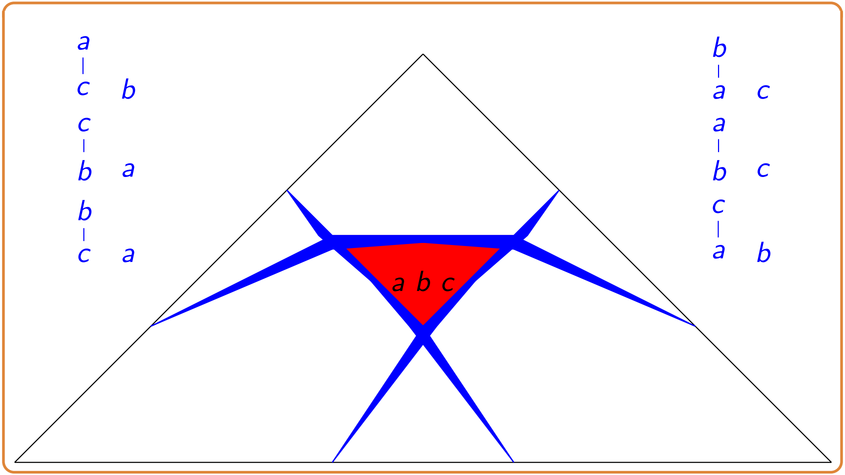
Theorem

The protocol complex for the scan/update model, in dimension n, for one round, is an $(n-1)$-connected simplicial set. It is a subdivision of $\Delta[n]$ plus some extra contractible "flares".

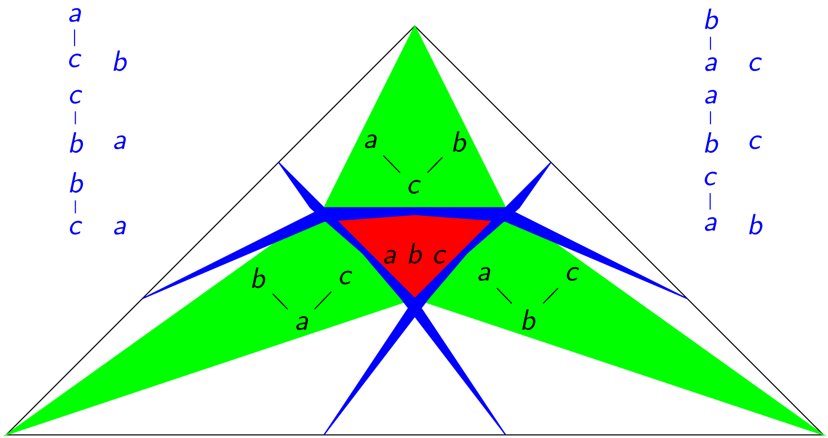
Construction of the protocol complex



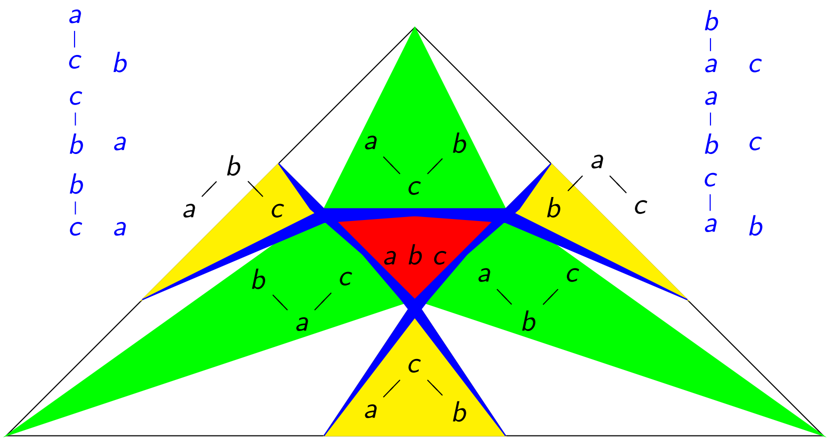
Construction of the protocol complex



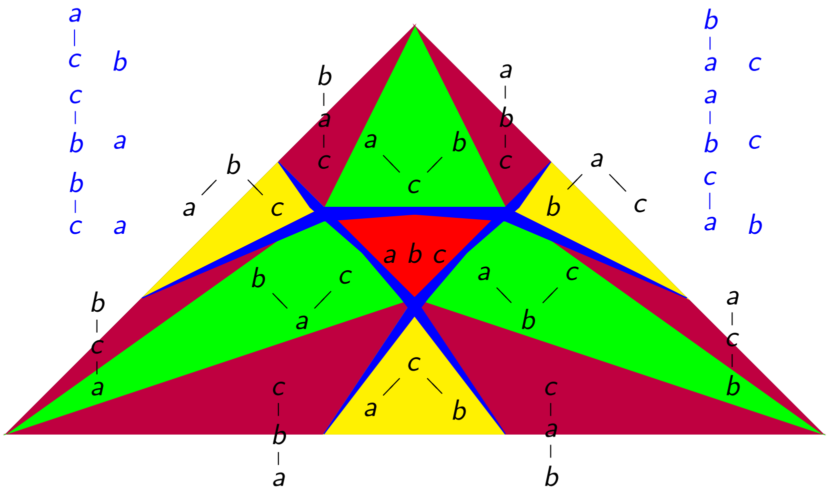
Construction of the protocol complex



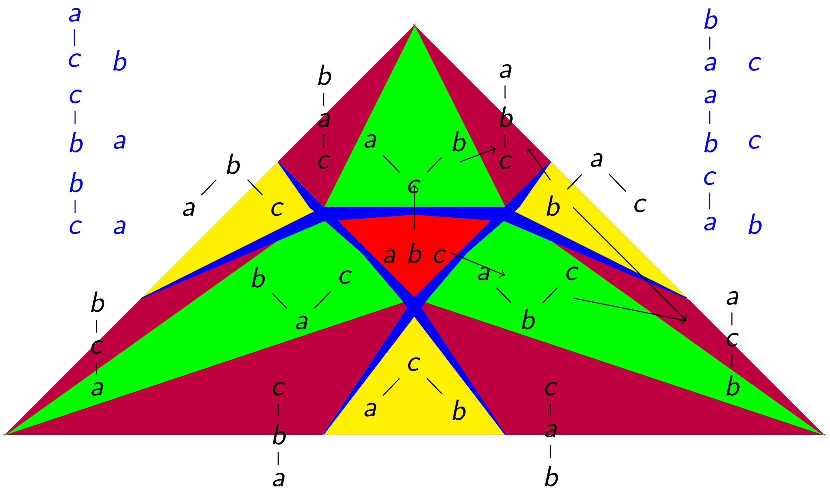
Construction of the protocol complex



Construction of the protocol complex



Construction of the protocol complex



Theorem

The prodsimplicial set corresponding to the scan/update model, in any dimension n, for one round, is discrete. Its cardinal is the number of interval posets on n elements.

Compare with:

Theorem

The protocol complex for the scan/update model, in dimension n, for one round, is an $(n-1)$-connected simplicial set. It is a subdivision of $\Delta[n]$ plus some extra contractible "flares".

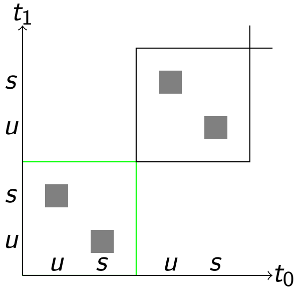
Conjectural construction of protocol complexes

The protocol complex is homotopy equivalent to the transversal hypergraph made of dead matrices (a hypergraph is in particular a simplicial set).

For $n = 2$ we saw that; for $n = 3$, the transversal hypergraph is a 11 dimensional simplicial set; for any n it is of dimension $n(n-1)^2 - 1$.

Sort of duality between prodsimplicial representation and the protocol complex one?

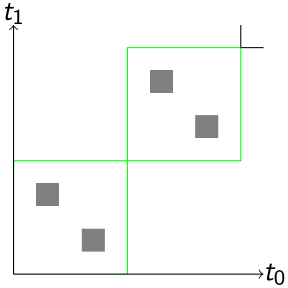
More rounds? clean-memory model first...



Iterated subdivision (fractal) of the protocol complex (round 1):

$$1.01 \xrightarrow{M_1} 0.11 \xrightarrow{M_3} 1.11 \xrightarrow{M_2} 0.10$$

Clean-memory model

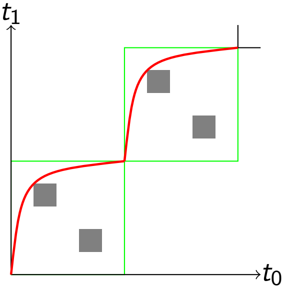


Iterated subdivision (fractal) of the protocol complex (round 2):

1.0101 ⟶ 0.1111) ⟶ 1.0111 ⟶ 0.1101 ⟶ 1.1101

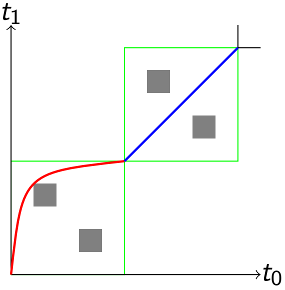
↓

0.0101 ⟵ 1.1111 ⟵ 0.0111 ⟵ 1.1101 ⟵ 0.1111



Iterated subdivision (fractal) of the protocol complex (round 2):

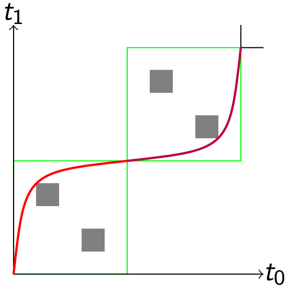
1.0101 —11→ 0.1111) —1→ 1.0111 —1→ 0.1101 ——→ 1.1101
↓
0.0101 ←—— 1.1111 ←—— 0.0111 ←—— 1.1101 ←—— 0.1111



Iterated subdivision (fractal) of the protocol complex (round 2):

1.0101 —11→ 0.1111) —13→ 1.0111 —1→ 0.1101 ——→ 1.1101
↓
0.0101 ←—— 1.1111 ←—— 0.0111 ←—— 1.1101 ←—— 0.1111

Clean-memory model



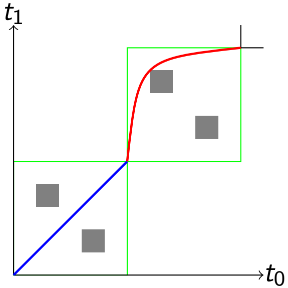
Iterated subdivision (fractal) of the protocol complex (round 2):

1.0101 —11→ 0.1111) —13→ 1.0111 —12→ 0.1101 ——→ 1.1101

↓

0.0101 ←—— 1.1111 ←—— 0.0111 ←—— 1.1101 ←—— 0.1111

Clean-memory model



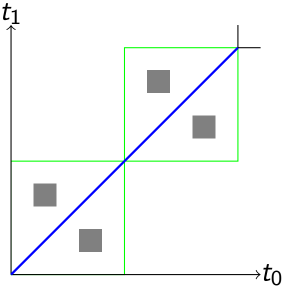
Iterated subdivision (fractal) of the protocol complex (round 2):

1.0101 ⟶ 0.1111) ⟶ 1.0111 ⟶ 0.1101 —31⟶ 1.1101

1.1101 —3⟶ 0.1111

0.0101 ⟵ 1.1111 ⟵ 0.0111 ⟵ 1.1101 ⟵3— 0.1111

Clean-memory model

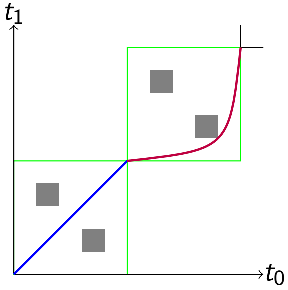


Iterated subdivision (fractal) of the protocol complex (round 2):

1.0101 ⟶ 0.1111) ⟶ 1.0111 ⟶ 0.1101 —31→ 1.1101

1.1101 —33→ 0.1111

0.0101 ⟵ 1.1111 ⟵ 0.0111 ⟵ 1.1101 ←3— 0.1111



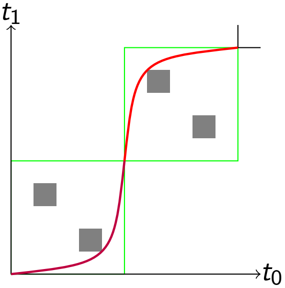
Iterated subdivision (fractal) of the protocol complex (round 2):

1.0101 → 0.1111) → 1.0111 → 0.1101 —31→ 1.1101

↓ 33

0.0101 ← 1.1111 ← 0.0111 ← 1.1101 ←32— 0.1111

Clean-memory model



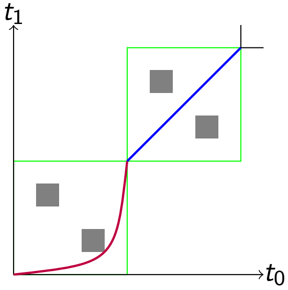
Iterated subdivision (fractal) of the protocol complex (round 2):

1.0101 ⟶ 0.1111) ⟶ 1.0111 ⟶ 0.1101 ⟶ 1.1101

↓

0.0101 ←2— 1.1111 ←2— 0.0111 ←21— 1.1101 ⟵ 0.1111

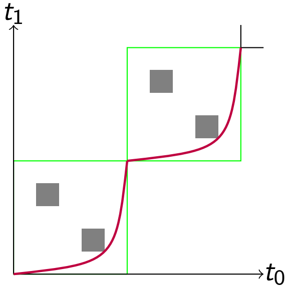
Clean-memory model



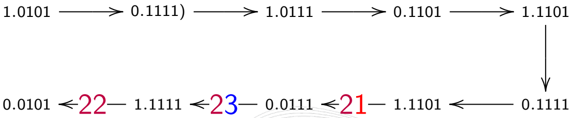
Iterated subdivision (fractal) of the protocol complex (round 2):

1.0101 ⟶ 0.1111) ⟶ 1.0111 ⟶ 0.1101 ⟶ 1.1101 ↓

0.0101 ←2— 1.1111 ←23— 0.0111 ←21— 1.1101 ⟵ 0.1111



Iterated subdivision (fractal) of the protocol complex (round 2):



Hence

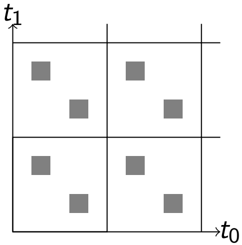
Theorem

The clean memory model for n processes at round r produces a subdivided n simplex (up to some "flares" which do not affect $(n-1)$-connectedness)

- Clear relation with underlying geometric semantics
- All is fine, but is there a new result here? Not yet...

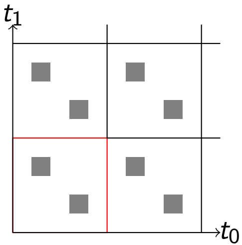
Example: same-memory model

Much more complicated! But fits in our framework perfectly



Example: same-memory model

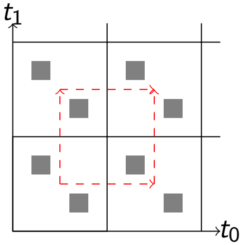
Much more complicated! But fits in our framework perfectly



→ each block (1 unfolding) creates an $(n-1)$-connected complex

Example: same-memory model

Much more complicated! But fits in our framework perfectly

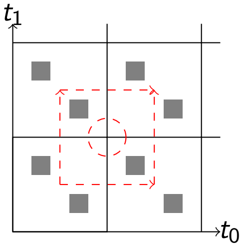


→ each block (1 unfolding) creates an $(n-1)$-connected complex

→ glued under some recurrence relation

Example: same-memory model

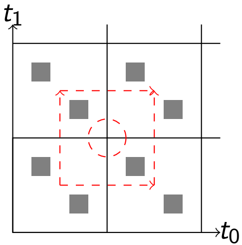
Much more complicated! But fits in our framework perfectly



$\rightarrow$ each block (1 unfolding) creates an $(n-1)$-connected complex
$\rightarrow$ glued under some recurrence relation
$\rightarrow$ whose relations make it a contractible scheme for pasting blocks

Example: same-memory model

Much more complicated! But fits in our framework perfectly

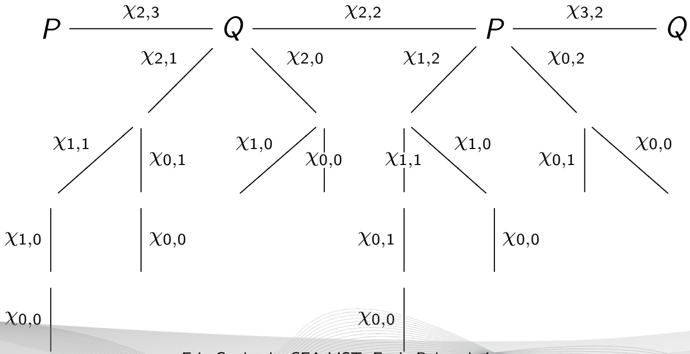


→ each block (1 unfolding) creates an $(n-1)$-connected complex
→ glued under some recurrence relation
→ whose relations make it a contractible scheme for pasting blocks
→ hence (nerve lemma), creates an $(n-1)$-connected protocol complex! (not previously described, as this does not create an iterated subdivided simplex)

...are fairly complex and not well known in general (yet!)

We call $\chi_{p,q}$ the protocol complex for p unrollings on the 1st coordinate, q on the 2nd ($n = 2$).

For instance, for 3×3, $\chi_{3,3}$ is:



Conjecture

The protocol complex for scan/update, same memory model, in any dimension n, and any number of rounds, is $(n-1)$-connected.

(so the same memory model has the same "computing power" as the clean memory model - which is folk knowledge...only)

This is a theorem in low dimension (2 and 3).

- Lots of experiments and lots of mathematics to be investigated yet on trace spaces...
- Applications to more subtle (and less combinatorial) models for protocols, in particular the "same memory model", and more intricate synchronisation primitives (test&set, fetch&add etc.)
- Extension to randomized algorithms: random simplicial sets! Application: possibility of consensus now!
- Logical interpretation of these 2 frameworks, simplicial, and directed...

Conclusion and future work

- Lots of experiments and lots of mathematics to be investigated yet on trace spaces...
- Applications to more subtle (and less combinatorial) models for protocols, in particular the "same memory model", and more intricate synchronisation primitives (test&set, fetch&add etc.)
- Extension to randomized algorithms: random simplicial sets! Application: possibility of consensus now!
- Logical interpretation of these 2 frameworks, simplicial, and directed...

Thanks for your attention!