Parallélisme

Geometry and Distributed Systems

Eric Goubault

Commissariat à l'Energie Atomique & Chaire Ecole Polytechnique-Thalès

Saclay

16th of March 2011

Based on work by M. Herlihy, S. Rajsbaum, N. Shavit...

Aim of the talk

Can we implement some functions on some distributed architecture, even if there are some crashes?

> Example: consensus on an asynchronous system
> NO: FLP'85!

- There is a nice "geometrization" of the problem
- We will solve easy problems to make you understand
- But it has also solved some new problems!
- ... and this is an active research area!

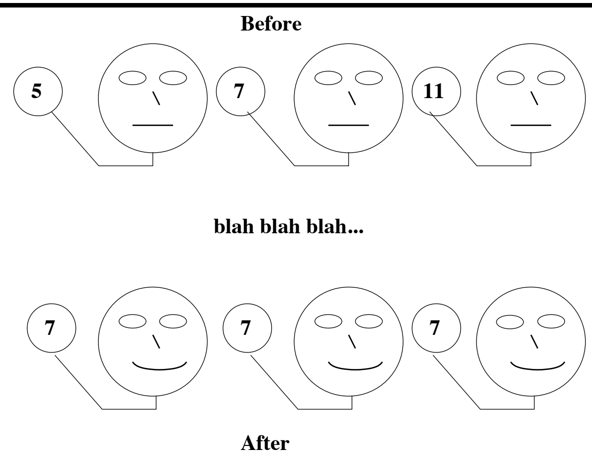
Decision tasks

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

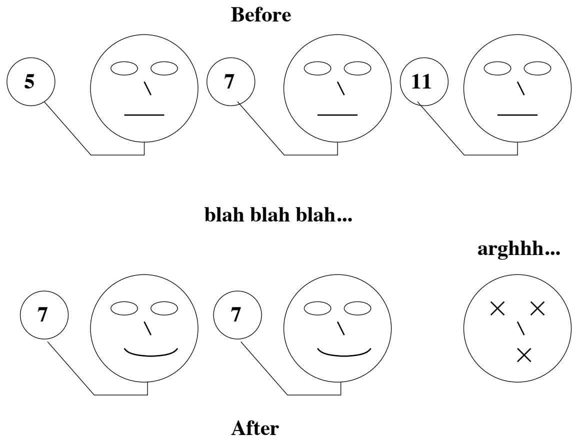
Each problem is given by:

- For each processor $P_0, \ldots, P_{n-1}$ a set of possible initial values (in a domain $\mathcal{K} = \mathbb{N}$ or $\mathbf{R}$ etc.), i.e. a subset $\mathcal{I}$ of $\mathcal{K}^n$: "*input*"
- Similarly, we are given a set of possible final values $\mathcal{J}$ in $\mathcal{K}^n$: "*output*"
- Finally, we are given a map, the "*decision map*" $\delta : \mathcal{I} \to \wp(\mathcal{J})$ associating to each possible initial value, the set of authorized output values

Example: consensus



Even if...



Example

- $\mathcal{K} = \mathbb{N}$, $\mathcal{I} = \mathbb{N}^n$,
- $\mathcal{J} = \{(n, n, \ldots, n) \mid n \in \mathbb{N}\}$,
- $\delta(x_0, x_1, \ldots, x_{n-1}) = \left\{ \begin{array}{c} \{(x_0, x_0, \ldots, x_0), \\ (x_1, x_1, \ldots, x_1), \\ \ldots, \\ (x_{n-1}, x_{n-1}, \ldots, x_{n-1})\} \end{array} \right.$

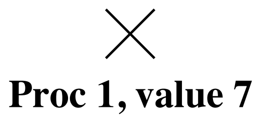
Main idea

- The *input* set and output sets have a geometrical structure (simplicial set)
- According to the architecture type, not all decision maps can be programmed
- There are geometrical constraints on the decision maps
- Very much like mainstream results in geometry, such as Brouwer's fixed point theorem...

Road map

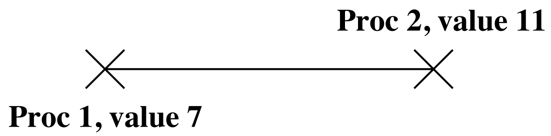
- Input and output sets as simplicial sets (examples)
- Some basic algebraic topology
- The dynamics as sets of simplicial sets (protocol simplicial set, or complex)
- Some results and references

Simplicial model of states



(local state)

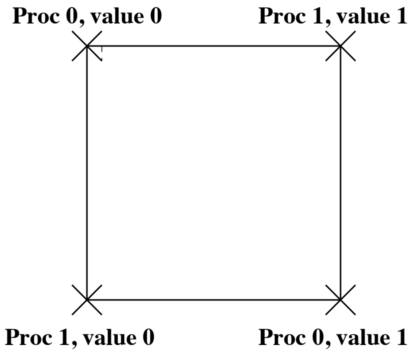
Simplicial model of states



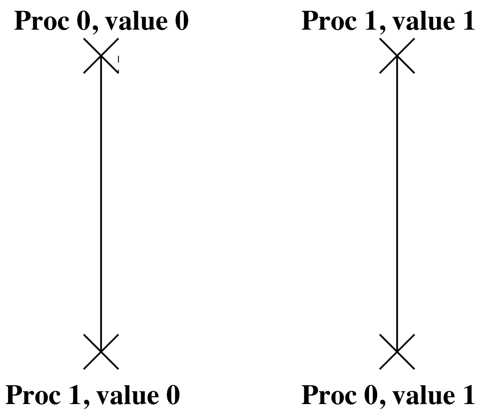
(compound state)

Initial states for (binary) consensus

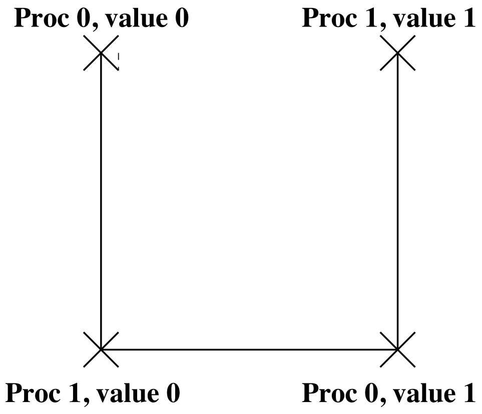
Here, 2 processors, i.e. dimension 2:



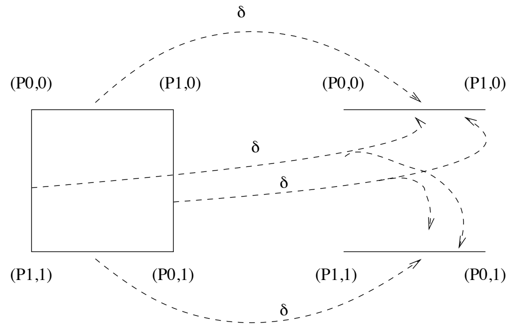
Final states for consensus



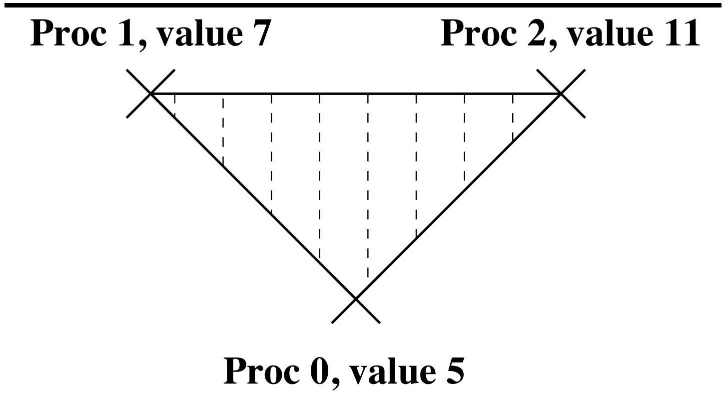
Final states for pseudo-consensus



Example: Consensus specification



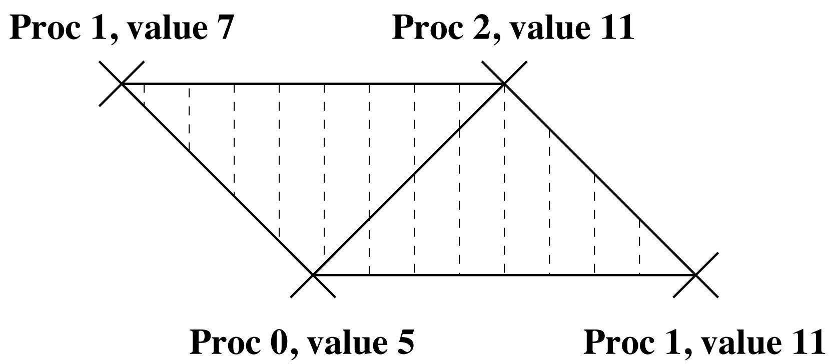
More generally: Simplicial model of states



(More generally [than a graph]: global state)

Example

Simplicial set=set of global states (with some common local states)



Back to *protocols*

- Finite program
- Starts with input values
- Fixed number of rounds
- Halts with decision value

The full-information protocol is the one where the local value is the full history of communications

Generic protocol

```
s = empty;
for (i=0; i<r; i++) {
  broadcast messages;
  s = s + messages received;
}
return delta(s);
```


Example

Synchronous message passing; notion of round:

- at each round, every processor broadcasts its own value to the others
- in any order
- then every processor receives the broadcasted values and computes a new local value

Failure models

- crash (fail-stop),
- byzantine etc.

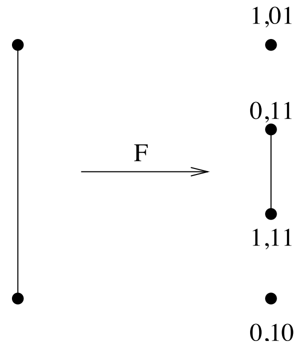
In what follows: crash failures only; can happen at any point of the broadcast, which can be done in any random order.

Protocol complex

Each protocol on some architecture defines:

- a simplicial set (for all rounds r):
  - vertices: sequence of messages received 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

Synchronous protocol complex

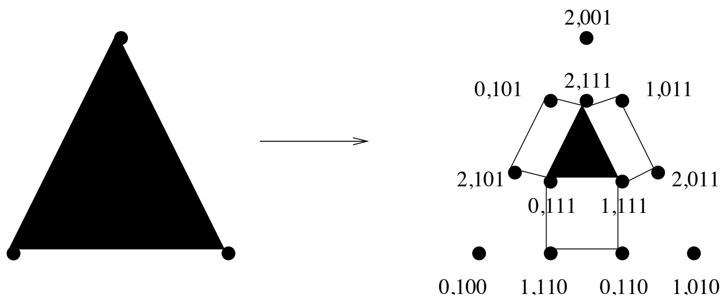


Explanation

In the synchronous model, at round 1:

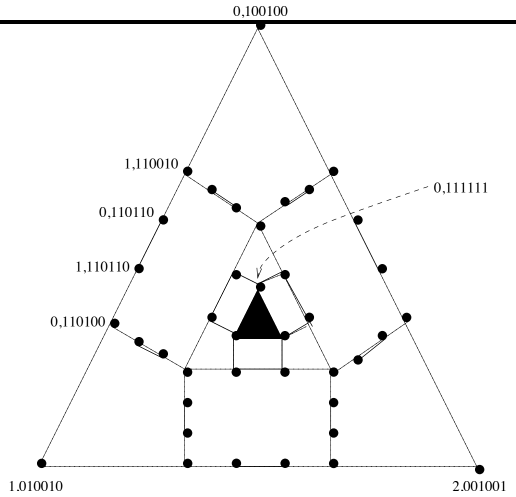
- no process has failed, hence everybody has received the message of the others (hence the central segment as global state)
- one process has failed, hence two points as possible states

Synchronous protocol complex

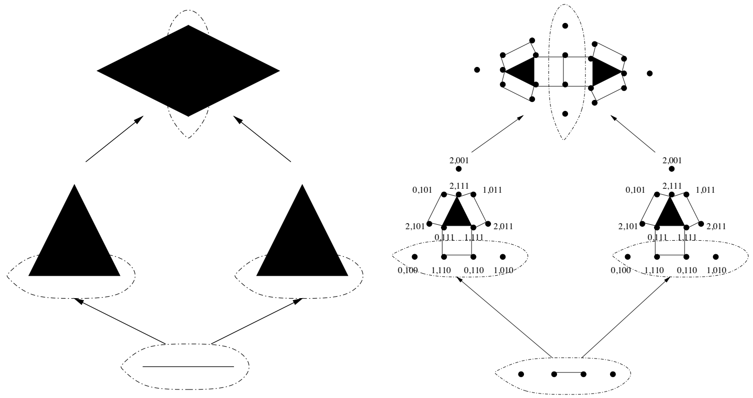


(wait-free - if up to 1 failure, forget the isolated points!)

Synchronous protocol complex - round 2



Synchronous protocol complex



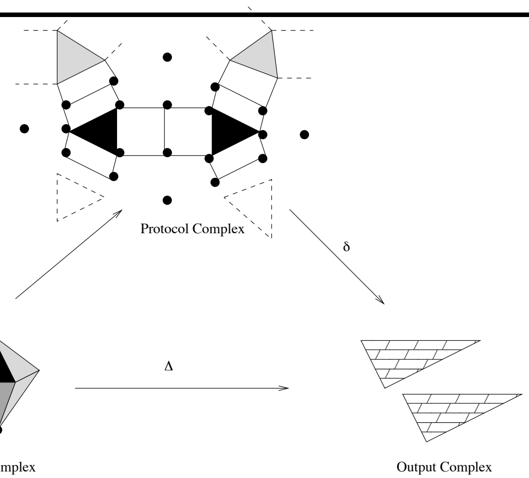
Decision map

The delta in the generic protocol is, mathematically speaking:

- is $\delta : P \rightarrow O$ (protocol to output complex)
- is a simplicial map (basically a function on vertices, extended on convex hulls)
- respects specification relation Δ, i.e. for all $x \in I$, for all $y \in P(I)$, $x\Delta(\delta(y))$

Proof strategy for impossibility/complexity results: find "topological obstruction" to the δ simplicial map (from protocol complex of any round/round up to k)

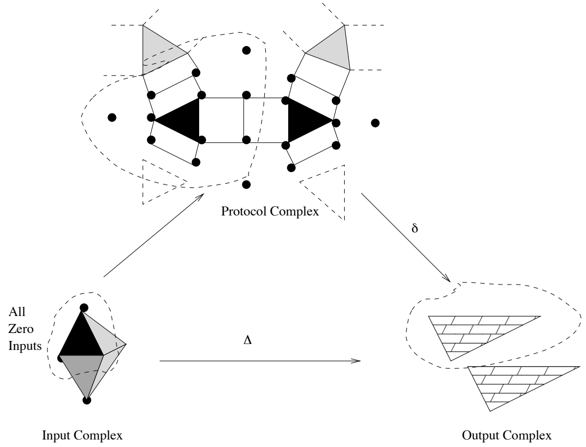
Main property



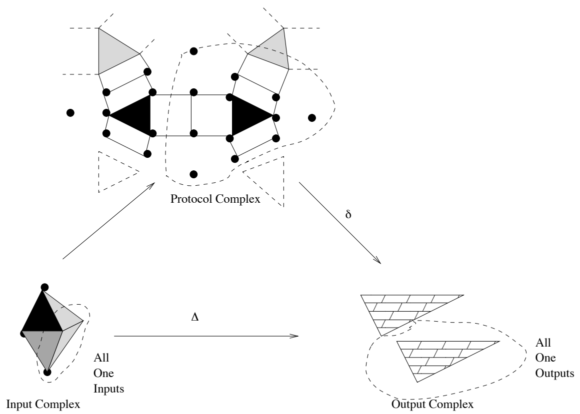
Easy application: consensus again...

- Binary consensus between 3 processes (synchronous message-passing model),
- Input complex is composed of 8 triangles: $(0,0,0)$, $(0,0,1)$, $(0,1,0)$, $(0,1,1)$, $(1,0,0)$, $(1,0,1)$, $(1,1,0)$ and $(1,1,1)$,
- Input complex is homeomorphic to a sphere (one connected component); the first four determine a "north" hemisphere, the last four create a "south" hemisphere
- Output complex is composed of 2 triangles: $(0,0,0)$ and $(1,1,1)$ (hence two connected components),
- Here: just one round.

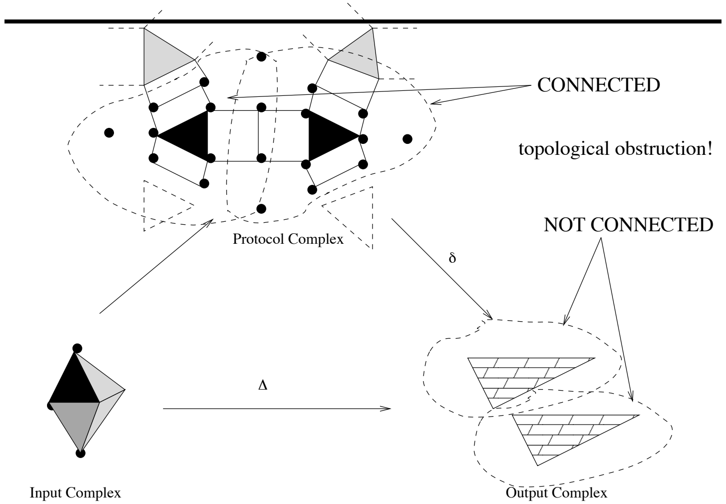
Easy application



Easy application



Easy application - for at most $n-2$ failures only!



More generally

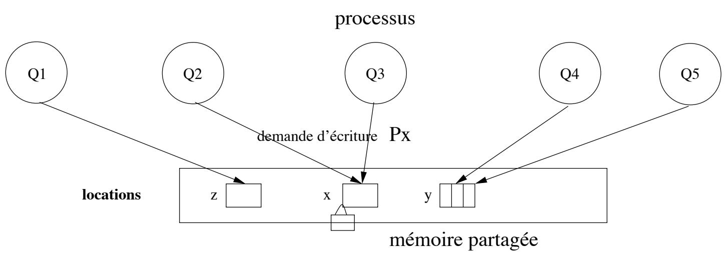
- In any such $(n-2)$-round protocol complex, the all-zero subcomplex and the all-one subcomplex are connected
- Corollary: no $(n-2)$-round consensus protocol

Easy and not new... but gives the idea...

Even more generally...

- Synchronous message-passing model with r rounds, and at most k failures
- $P(S^{n-1})$ is $(n-rk-2)$-connected: implies $(n-1)$-round consensus bound (for $k=1$).

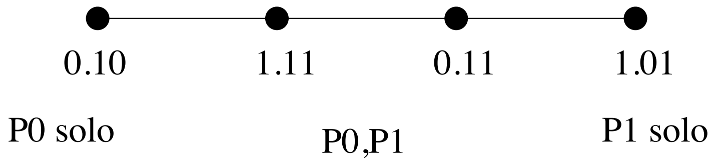
Shared-memory model



Asynchronous wait-free protocols

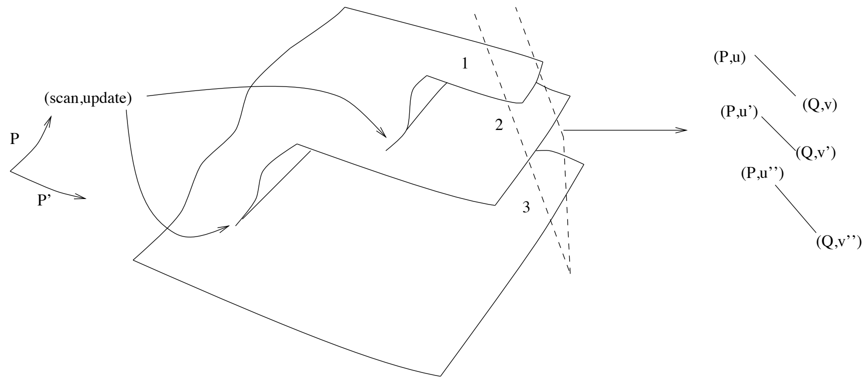
- n processes share memory (unbounded size), partitioned: one private chunk for each process
- Each process can:
  - atomically write to its location (update)
  - atomically scan (read) all of the memory into its local memory
- Equivalent to the usual read/write models
- We want *wait-free* protocols, i.e. robust to up to $n-1$ crash failures

One-round protocol simplicial set (2D)

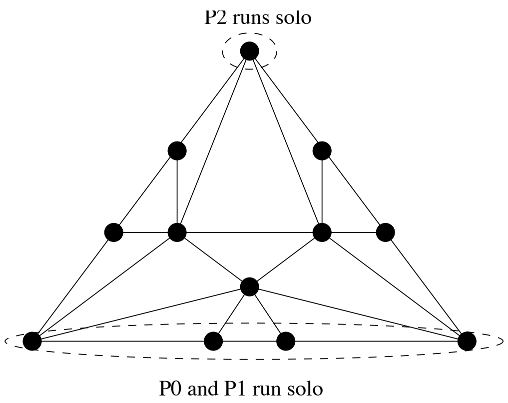


Semantics

Dynamics (and its cut up to time r=protocol complex):



One-round protocol simplicial set (3D)

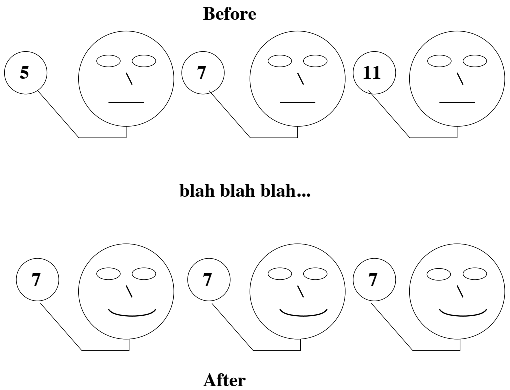


Theorem

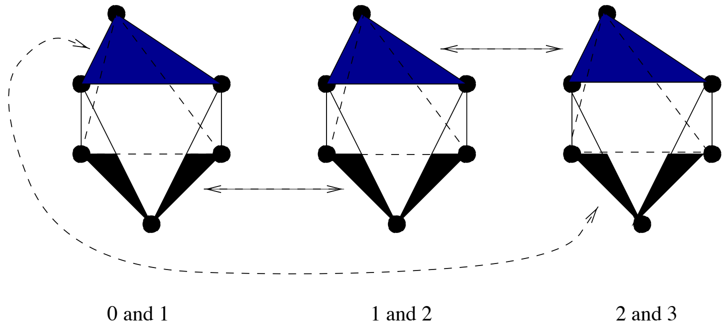
- Wait-free read/write protocol complexes are:
  - $(n-1)$-connected (no holes in any dimension)
  - no matter how long the protocol runs
- Application: k-set agreement

k-set agreement task

Generalization of consensus; processes must end up with at most k different values (taken from the initial values):



Output simplicial set $(n = 3,\ k = 2)$



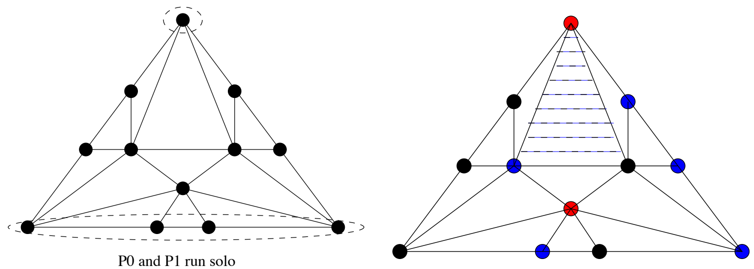
3 spheres glued together minus the simplex formed of all 3 values: not 1-connected

Sketch of a proof

A tool from algebraic topology (Sperner's lemma):

- Subdivide a simplex
- Give each "corner" a distinct "color"
- Give each vertex a corner color
- Giver interior vertices any corner color

Sperner's lemma



⇒ At least one simplex has all colors

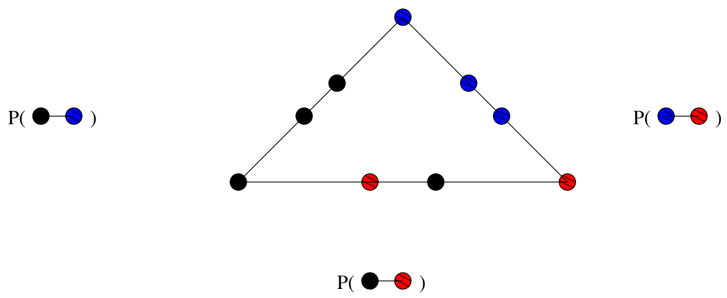
Input and protocol simplicial set

- Each process colored with distinct input
- Each vertex colored with decision

Protocol complex

- For a one-process execution: same vertex and same color (cannot decide anything else)
- For a two-process execution:
  - the protocol complex is connected
  - all vertices are of one of the two colors

Protocol complex - for all 2 process executions



Full protocol complex

- Because complex is simply-connected
- We can “fill-in” edge-paths
- Vertices colored with input colors

End of proof

Apply Sperner's Lemma:

- Some simplex has all three colors
- That simplex is a protocol execution that decides three values!

Converse

- In fact, even more:
- 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

Principle of the proof

$\Rightarrow$

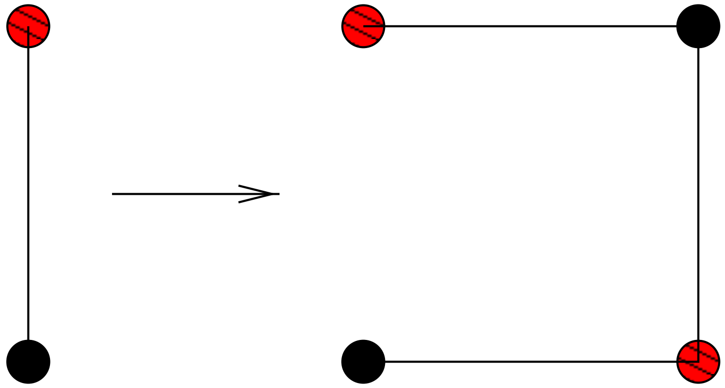
- Protocol complex is $(n-1)$-connected (using Mayer-Vietoris)
- Exploit connectivity to
  - embed subdivided input complex into protocol complex
  - map protocol complex to output complex
  - just like k-set agreement proof

Principle of the proof

$\Leftarrow$

- We can reduce any task to "simplex agreement" [using the participating set algorithm of Borowsky and Gafni 1993]
- Start out at corners of subdivided simplex
- Must rendez-vous on vertices of single simplex in subdivision

Example



Subdivision of a segment into three segments

Protocol

$$\begin{array}{lll} P & = & update; \\ & & scan; \\ & & case\ (u,v)\ of \\ & & \ (x,y'):\ u=x'; update; [] \\ & & \ default:\ update \end{array} \qquad \begin{array}{lll} P' & = & update; \\ & & scan; \\ & & case\ (u,v)\ of \\ & & \ (x,y'):\ v=y; update; [] \\ & & \ default:\ update \end{array}$$

Proof

Using the semantics, we have the following three possible 1-schedules (up to homotopy), since the only possible interactions are between the *scan* and *update* statements,

(i) Suppose the *scan* operation of P is completed before the *update* operation of P' is started: P does not know y so it chooses to write x. *Prog* ends up with $((P,x),(P',y))$.

(ii) Symmetric case: *Prog* ends up with $((P,x'),(P',y'))$.

(iii) The *scan* operation of P is after the *update* of P' and the *scan* of P' is after the *update* of P. *Prog* ends up with $((P,x'),(P',y))$.

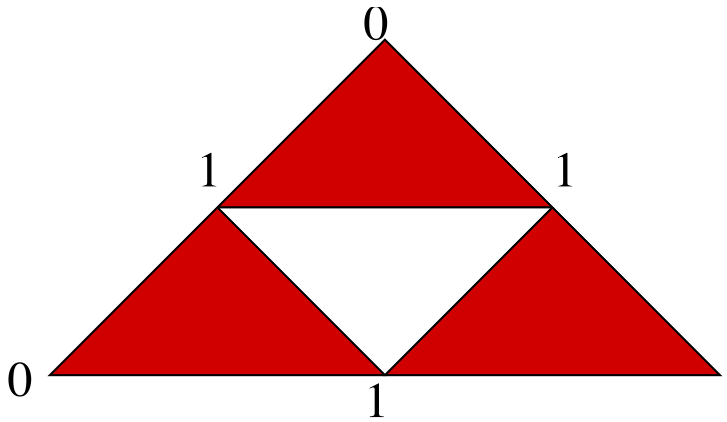
Other communication primitives

Real multiprocessors provide additional atomic synchronization:

- test&set
- fetch&add
- compare&swap
- queues...

Other protocol complexes...other results

Example: test&set protocol complex



Test&Set

- Wait-free Test&Set protocol complexes
  - are all $(n-3)$-connected
  - more powerful than read/write (2-process consensus)
  - but still no 3-process consensus
- Similar results hold for other synchronization operations

References and main results

- Begins with Fisher-Lynch-Patterson ("FLP") in 1985: there exists a simple task that cannot be solved in a (simple) message-passing system with at most one potential crash
- Created a very active research area, see for instance Nancy Lynch's book "Distributed Algorithms" (1996)

References and main results

- Later developed by Biran-Moran-Zaks in PoDC'88: characterization of the tasks that can be solved by a (simple) message-passing system in the presence of one failure
- The argument uses a "similarity chain", which could be seen as a 1-dimensional version of what we just developed
- Revealed to be difficult to extend to models with more failures

References and main results

Then, in PoDC'1993, independently,

- Borowsky-Gafni, Saks-Zaharoglou and Herlihy-Shavit derived lower bounds for the k-set agreement problem of Chaudhuri (proposed in 1990)
  [at least $\lfloor \frac{f}{k} \rfloor + 1$ steps in synchronous model]
- Saks-Zaharoglou and Herlihy-Shavit exploited topological properties to derive this lower bound

References and main results

- Renaming: Attiya-BarNoy-Dolev-Peleg JACM 1990,
- The $(n+1, K)$-renaming task starts with $n+1$ processes being given a unique input name in $0, \ldots, N$ and are required to choose unique output name in $0, \ldots, K$ with $n \leq K < N$ (independently of a "process id" - i.e. "anonymous renaming" in fact).
- Showed that (message-passing model) there is a wait-free solution for $K \geq 2n+1$, none when $K \leq n+2$
- Using these geometrical techniques: it has been shown that there is no renaming when $K \leq 2n$
- Herlihy and Shavit STOC'93: same result holds for the wait-free asynchronous model (using homology explicitely).

References and main results

Later results, on the same line, include:

- Full characterization of wait-free asynchronous tasks with atomic read/writes on registers, see "The topological structure of asynchronous computability", M. Herlihy and N. Shavit, J. of the ACM, jan. 2000
- Use of algebraic spans in "Algebraic Spans", M. Herlihy and S. Rajsbaum as a unified methods for renaming, k-set agreement problems etc.
- Use of pseudo-spheres...

References and research directions

- Consensus numbers (see M. Herlihy and then E. Ruppert SIAM J. Comput. vol 30, No 4, 2000 for instance). Importance based on the remark (M. Herlihy): an object which solves the consensus problem for n processes can simulate in a wait-free manner (together with read/write registers) any object for n or fewer processes.
- Example: R/W registers have consensus number 1, test&set, queues, stacks, fetch and add have consensus number 2 etc.
- Example: There is no wait-free $(n+1, 2j)$-renaming protocol if processes share a read/write memory and $(n+1, j)$-consensus objects.

References and research directions

- Afek et Strup: characterization of the effect of the register size in the power of synchronization primitives
- Characterization of complexity and not only computability, see for instance “Towards a Topological Characterization of Asynchronous Complexity”, G. Hoest and N. Shavit
- Links with (geometric) semantics [potential for more realistic models of distributed systems?], for instance my paper in CAAP’97 “Optimal Implementation of Wait-Free Binary Relations” ?
- Extension of this model for randomized algorithms etc.?