FROM GEOMETRIC SEMANTICS TO ASYNCHRONOUS COMPUTABILITY

ÉRIC GOUBAULT

SAMUEL MIMRAM

CHRISTINE TASSON

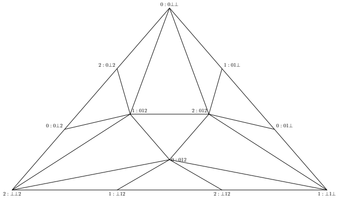
École Polytechnique / PPS

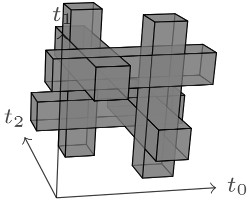
DISC conference

October 9, 2015

General idea

Given an asynchronous protocol, we relate two associated geometric constructions:





the **protocol complex** [Herlihy, . . .]

the **geometric semantics** [Goubault, . . .]

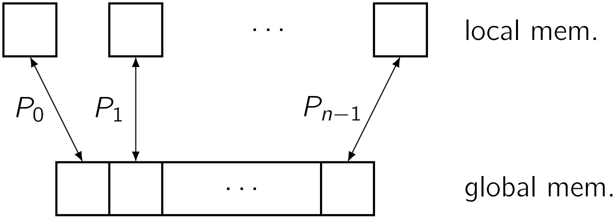
Aim: show impossibility results!
This work should help generalizing to more communication primitives.

ASYNCHRONOUS COMPUTABILITY

Asynchronous protocols

We consider here a model with n processes P_i:

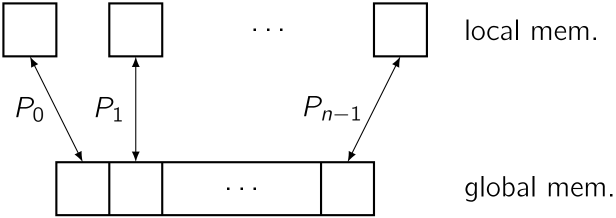
- each process has a local memory cell
- there is a global memory with n cells



Asynchronous protocols

We consider here a model with n processes P_i:

- each process has a local memory cell
- there is a global memory with n cells

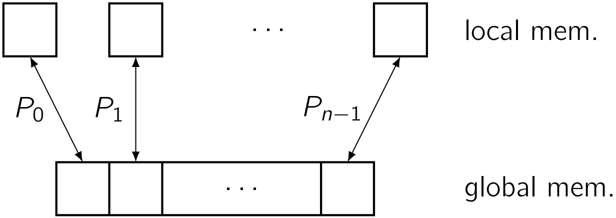


- each process alternatively does
  - `update`: write in its global memory cell
  - `scan`: read the whole global memory and update its local cell

Asynchronous protocols

We consider here a model with n processes P_i:

- each process has a local memory cell
- there is a global memory with n cells



- each process alternatively does
  - `update`: write in its global memory cell
  - `scan`: read the whole global memory and update its local cell

A **protocol** = what processes compute depending on values.

Decision tasks

We write $\mathcal{V}$ for the set of values, where $\perp$ denotes a dead process.

The question is whether a protocol can solve a **task** $\Theta \subseteq \mathcal{V}^n \times \mathcal{V}^n$ (where $\mathcal{V}$ is the set of values) in the presence of faults.

We suppose here that the initial value of a process is its process number (for simplicity).

Example (*Consensus*)

All processes must end with the same value, which is among the input values of the alive processes, i.e.

$$\Theta \quad = \quad \left\{ \begin{array}{c} (01, 00) \\ (01, 11) \\ (0\perp, 0\perp) \\ (\perp 1, \perp 1) \end{array} \right\}$$

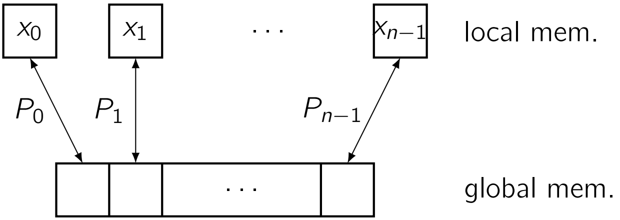
Execution traces

An **execution trace** is determined by a word in $\{u_i, s_i\}^*$.

Remark

The effect of execution traces on memory is invariant under the smallest congruence $\approx$ such that

$$u_j u_i \approx u_i u_j \qquad s_j s_i \approx s_i s_j$$



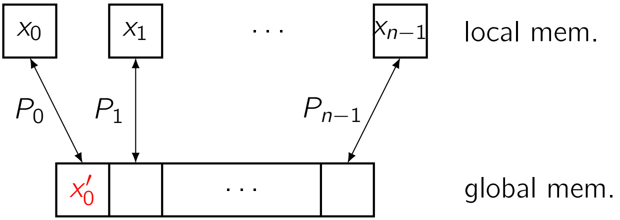
e.g.

An **execution trace** is determined by a word in $\{u_i, s_i\}^*$.

Remark

The effect of execution traces on memory is invariant under the smallest congruence $\approx$ such that

$$u_j u_i \approx u_i u_j \qquad s_j s_i \approx s_i s_j$$



e.g. u_0

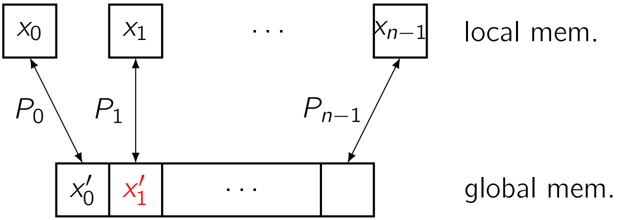
Execution traces

An **execution trace** is determined by a word in $\{u_i, s_i\}^*$.

Remark

The effect of execution traces on memory is invariant under the smallest congruence $\approx$ such that

$$u_j u_i \approx u_i u_j \qquad\qquad s_j s_i \approx s_i s_j$$



e.g. $u_0 u_1$

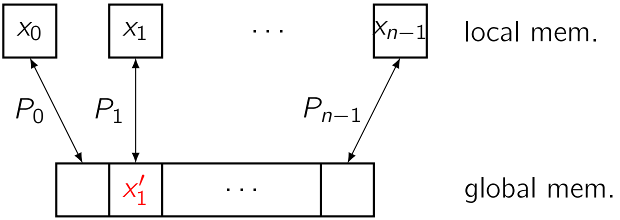
Execution traces

An **execution trace** is determined by a word in $\{u_i, s_i\}^*$.

Remark

The effect of execution traces on memory is invariant under the smallest congruence $\approx$ such that

$$u_j u_i \approx u_i u_j \qquad\qquad s_j s_i \approx s_i s_j$$



e.g. $\qquad u_0 u_1 \approx u_1$

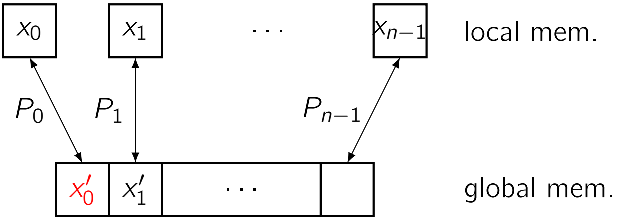
Execution traces

An **execution trace** is determined by a word in $\{u_i, s_i\}^*$.

Remark

The effect of execution traces on memory is invariant under the smallest congruence $\approx$ such that

$$u_j u_i \approx u_i u_j \qquad s_j s_i \approx s_i s_j$$



e.g. $\quad u_0 u_1 \approx u_1 u_0$

GEOMETRIC SEMANTICS

Directed geometric semantics

The idea of geometric semantics is to formalize the dictionary:

$$\textbf{program} \qquad \Leftrightarrow \qquad \textbf{topological space}$$

so that we can import tools from (algebraic) topology in order to study concurrent programs.

Directed geometric semantics

The idea of geometric semantics is to formalize the dictionary:

$$\begin{array}{rcl} \textbf{program} & \Leftrightarrow & \textbf{topological space} \\ \text{state} & \Leftrightarrow & \text{point of the space} \end{array}$$

so that we can import tools from (algebraic) topology in order to study concurrent programs.

Directed geometric semantics

The idea of geometric semantics is to formalize the dictionary:

program	$\Leftrightarrow$	**topological space**
state	$\Leftrightarrow$	point of the space
execution trace	$\Leftrightarrow$	path

so that we can import tools from (algebraic) topology in order to study concurrent programs.

Directed geometric semantics

The idea of geometric semantics is to formalize the dictionary:

program	$\Leftrightarrow$	**topological space**
state	$\Leftrightarrow$	point of the space
execution trace	$\Leftrightarrow$	path
equivalent traces	$\Leftrightarrow$	homotopic paths

so that we can import tools from (algebraic) topology in order to study concurrent programs.

Directed geometric semantics

The idea of geometric semantics is to formalize the dictionary:

program	$\Leftrightarrow$	**topological space**
state	$\Leftrightarrow$	point of the space
execution trace	$\Leftrightarrow$	path
equivalent traces	$\Leftrightarrow$	homotopic paths

so that we can import tools from (algebraic) topology in order to study concurrent programs.

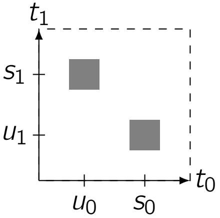
We actually need to use spaces equipped with a notion of direction in order to take in account irreversible time.

An example

Consider two processes executing one round of update/scan, i.e.

$$u_0.s_0 \quad \| \quad u_1.s_1$$

The geometric semantics of this program will be



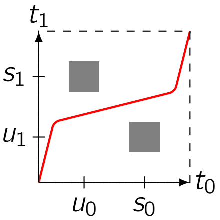
i.e. a square $[0,1] \times [0,1]$ minus two holes, which is directed componentwise.

An example

Consider two processes executing one round of update/scan, i.e.

$$u_0.s_0 \quad \| \quad u_1.s_1$$

The geometric semantics of this program will be



i.e. a square $[0,1] \times [0,1]$ minus two holes, which is directed componentwise.

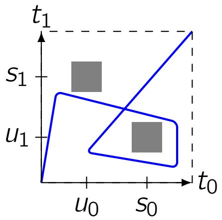
directed path : $u_1 u_0 s_0 s_1$

An example

Consider two processes executing one round of update/scan, i.e.

$$u_0.s_0 \quad \| \quad u_1.s_1$$

The geometric semantics of this program will be



i.e. a square $[0,1] \times [0,1]$ minus two holes, which is directed componentwise.

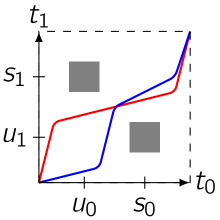
non directed path : ???

An example

Consider two processes executing one round of update/scan, i.e.

$$u_0.s_0 \quad \| \quad u_1.s_1$$

The geometric semantics of this program will be



i.e. a square $[0,1] \times [0,1]$ minus two holes, which is directed componentwise.

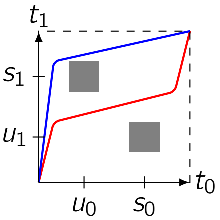
homotopy between paths : $u_1 u_0 s_0 s_1 \approx u_0 u_1 s_0 s_1$

An example

Consider two processes executing one round of update/scan, i.e.

$$u_0.s_0 \quad \| \quad u_1.s_1$$

The geometric semantics of this program will be



i.e. a square $[0,1] \times [0,1]$ minus two holes, which is directed componentwise.

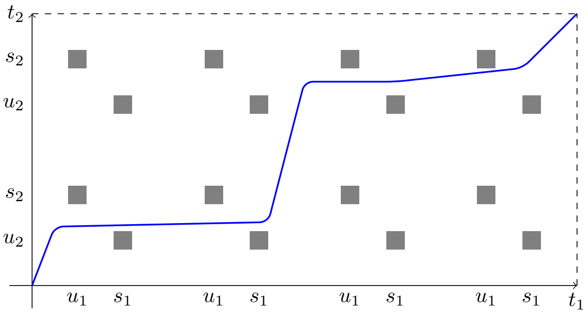
some paths are not homotopic

More examples

This generalizes to *more rounds*:
consider two processes executing 2 and 4 rounds of update/scan,

$$u_0.s_0.u_0.s_0 \quad \| \quad u_1.s_1.u_1.s_1.u_1.s_1.u_1.s_1$$

The geometric semantics of this program will be

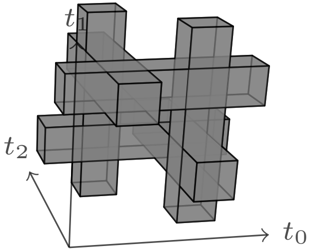


More examples

This generalizes to *more processes*:
consider three processes executing one round of update/scan,

$$u_0.s_0 \quad \| \quad u_1.s_1 \quad \| \quad u_2.s_2$$

The geometric semantics of this program will be



NB: we will illustrate in dimension 2, where things are simpler

Formally,

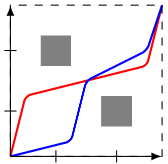
Definition

A **pospace** $(X, \leq)$ consists of a topological space X equipped with a partial order $\leq \subseteq X \times X$, which is closed.

A **dipath** p is a continuous non-decreasing map $p : [0, 1] \to X$.

A **dihomotopy** H from a path p to a path q is a continuous map $H : [0, 1] \times [0, 1] \to X$ such that

- $H(0, t) = p(t)$ for every t
- $H(1, t) = q(t)$ for every t
- $t \mapsto H(s, t)$ is a dipath for every s
- $t \mapsto H(0, t)$ and $t \mapsto H(1, t)$ are constant



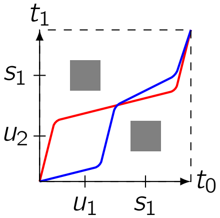
Directed paths vs traces

Theorem

Fixing a number of rounds for each process, there is a bijection between

(i) *directed paths up to directed homotopy in the geometric semantics*

(iii) *execution traces up to* $\approx$



$\Leftrightarrow$ $\quad$ $u_1\,u_0\,s_0\,s_1 \approx u_0\,u_1\,s_0\,s_1$

Directed paths vs traces

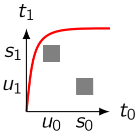
Theorem

Fixing a number of rounds for each process, there is a bijection between

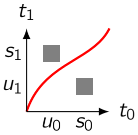
(i) *directed paths up to directed homotopy in the geometric semantics*

(ii) *colored interval orders*

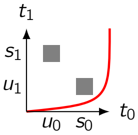
(iii) *execution traces up to* $\approx$



$[u_0, s_0] \succ [u_1, s_1]$



$[u_0, s_0] \parallel [u_1, s_1]$



$[u_0, s_0] \prec [u_1, s_1]$

Definition

A family (I_j) of intervals $I_j = [a_j, b_j]$ of $\mathbb{R}$ is partially ordered by $I_j \prec I_k$ whenever $x < y$ for every $x \in I_j$ and $y \in I_k$, e.g.

$$[1,3] \prec [5,6] \qquad\qquad [1,3] \parallel [2,6]$$

We write $I_j \parallel I_k$ when two elements are not comparable.
A poset isomorphic to such a poset is called an **interval order**.

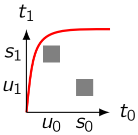
Definition

A family (I_j) of intervals $I_j = [a_j, b_j]$ of $\mathbb{R}$ is partially ordered by $I_j \prec I_k$ whenever $x < y$ for every $x \in I_j$ and $y \in I_k$, e.g.

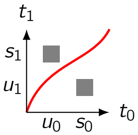
$$[1,3] \prec [5,6] \qquad\qquad [1,3] \parallel [2,6]$$

We write $I_j \parallel I_k$ when two elements are not comparable.
A poset isomorphic to such a poset is called an **interval order**.

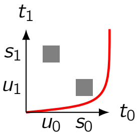
Here, we consider a colored version, where elements are of the form (i, k) with $0 \le i < n$ a process number, such that two elements with the same labels are comparable.



$[u_0, s_0] \succ [u_1, s_1]$



$[u_0, s_0] \parallel [u_1, s_1]$



$[u_0, s_0] \prec [u_1, s_1]$

THE PROTOCOL COMPLEX

The protocol complex

The **protocol complex** is a simplicial complex associated to a protocol, introduced by Herlihy et al. as a central tool in order to characterize tasks which are solvable.

We show here how to reconstruct it from the geometric semantics.

Generic protocols

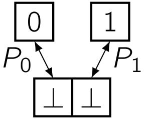
Given a protocol solving a given task, we can without loss of generality suppose that it is

- *full-information*:
  u_i writes the exact local memory in the global one
- *generic*:
  s_i adds the contents of global memory to local one

(except at the end where a choice is made).

Example

With two processes, we have for instance



Generic protocols

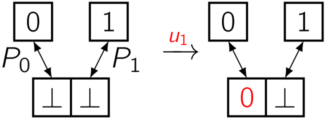
Given a protocol solving a given task, we can without loss of generality suppose that it is

- *full-information*:
  u_i writes the exact local memory in the global one
- *generic*:
  s_i adds the contents of global memory to local one

(except at the end where a choice is made).

Example

With two processes, we have for instance



Generic protocols

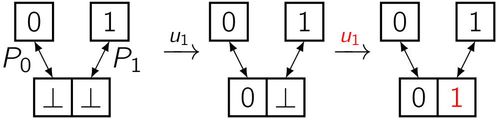
Given a protocol solving a given task, we can without loss of generality suppose that it is

- *full-information*:
  u_i writes the exact local memory in the global one
- *generic*:
  s_i adds the contents of global memory to local one

(except at the end where a choice is made).

Example

With two processes, we have for instance



Generic protocols

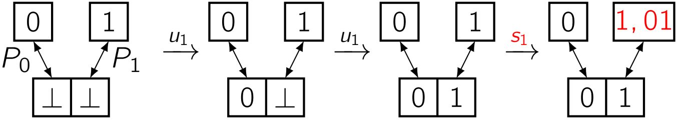
Given a protocol solving a given task, we can without loss of generality suppose that it is

- *full-information*:
  u_i writes the exact local memory in the global one
- *generic*:
  s_i adds the contents of global memory to local one

(except at the end where a choice is made).

Example

With two processes, we have for instance



Generic protocols

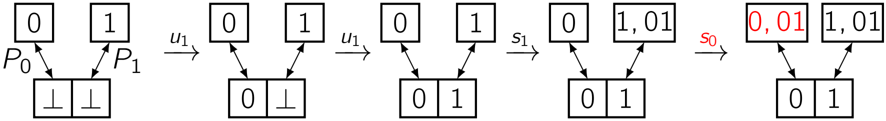
Given a protocol solving a given task, we can without loss of generality suppose that it is

- *full-information*:
  u_i writes the exact local memory in the global one
- *generic*:
  s_i adds the contents of global memory to local one

(except at the end where a choice is made).

Example

With two processes, we have for instance



Generic protocols

Given a protocol solving a given task, we can without loss of generality suppose that it is

- *full-information*:
  u_i writes the exact local memory in the global one
- *generic*:
  s_i adds the contents of global memory to local one

(except at the end where a choice is made).

Definition

The local memory of a process is called its **view**.
Two (or more) views are **coherent** when they can occur at the same time in some execution.

The protocol complex

Definition

Given a number of rounds, the protocol complex is the simplicial complex such that

- vertices are the possible views,
- two vertices are linked by an edge when they are coherent,
- three vertices bound a triangle when they are coherent,
- etc.

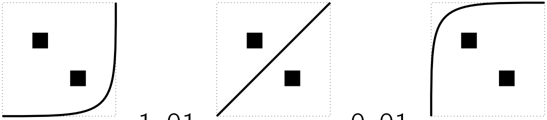
Example

With two processes, after 0 and 1 rounds, the complexes are

- $0 \text{ —— } 1$
- $0, 0\bot \text{ —— } 1, 01 \text{ —— } 0, 01 \text{ —— } 1, \bot 1$

From geometry to the complex

One can notice in the last example that edges are in bijection with directed paths up to homotopy (and with interval orders):

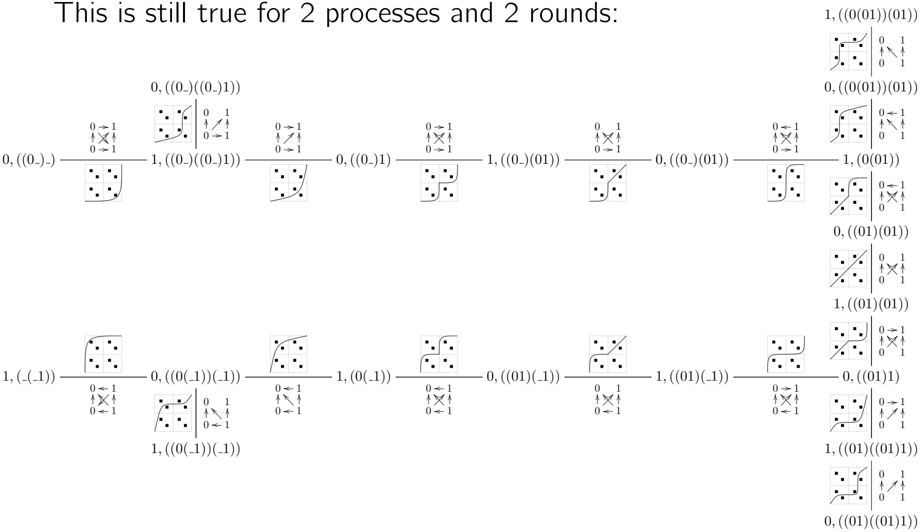


$$0,0\bot \xrightarrow[0\prec 1]{} 1,01 \xrightarrow[0\quad 1]{} 0,01 \xrightarrow[0\succ 1]{} 1,\bot 1$$

(more generally maximal simplices are in bijection with maximal directed paths up to homotopy).

From geometry to the complex

This is still true for 2 processes and 2 rounds:



From interval orders to the complex

Since dipaths up to dihomotopy are the same as interval orders, we can start from the latter.

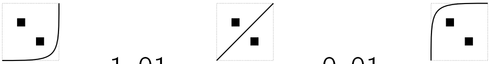
From interval orders to the complex

Since dipaths up to dihomotopy are the same as interval orders, we can start from the latter.

Proposition

Given a colored interval order $(X, \preceq)$, the **view** V_i^k of the i-th process at round k is given by its restriction to the k-th scan of the i-th process

$$V_i^k \quad = \quad \{(j,l) \mid (i,k) \parallel (j,l) \text{ or } (j,l) \prec (i,k)\}$$



$$0,0\bot \xrightarrow[0\prec 1]{\hspace{3em}} 1,01 \xrightarrow[0\quad 1]{\hspace{3em}} 0,01 \xrightarrow[0\succ 1]{\hspace{3em}} 1,\bot 1$$

The interval order complex

Definition

The **interval order complex** is the simplicial complex whose

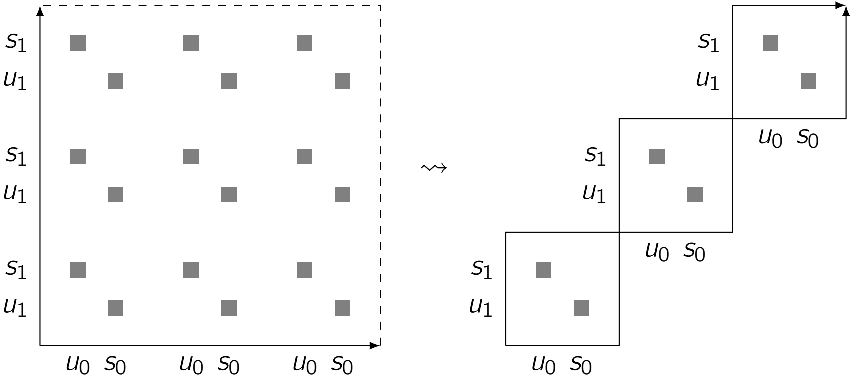
- *vertices* are $((i,k), V_i^k)$ where i stands for the i-th process, k for the round number and V_i^k for an interval order such that for all $(j,l) \in V_i^k$, either $(i,k) \parallel (j,l)$ or $(j,l) \prec (i,k)$,
- *maximal simplices* are $\left\{((0,r_0), V_0^{r_0}), \ldots, ((n,r_n), V_n^{r_n})\right\}$ such that there is an interval order $(X_{(r)}^n, \prec)$ whose restriction to (i, r_i) is $V_i^{r_i}$.

Theorem

The interval order complex is isomorphic to the protocol complex.

The layered protocol complex

Generally, one considers executions which are **layered**: all processes must have finished round n (or died) before process can start round $n+1$.



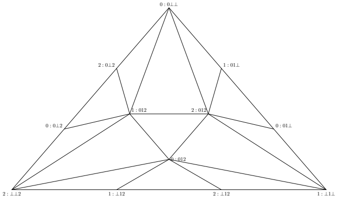
The layered protocol complex

Generally, one considers executions which are **layered**: all processes must have finished round n (or died) before process can start round $n+1$.

Proposition

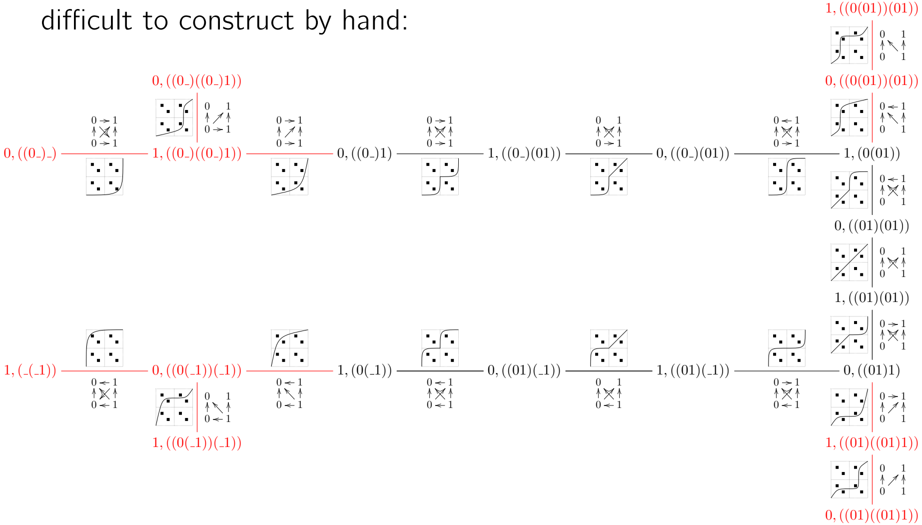
Layered immediate snapshot executions correspond to the interval orders such that $J \prec K$ and $I \parallel J$ implies $I \prec K$.

Moreover, we can recover the fact that layered protocol complexes are iterated chromatic subdivisions of the standard simplex.



The layered protocol complex

Here, we can compute non-layered protocols, which would be difficult to construct by hand:



Conclusion

We have linked geometric semantics and asynchronous computability.

The geometric semantics of many more primitives than update/scan is known (e.g. test/set, compare/swap, etc.) the next step is to try to start from the geometric semantics in order to invent the corresponding “protocol complex”
(NB: interval orders were not really crucial in this work).