POINTS OF VIEW ON ASYNCHRONOUS COMPUTABILITY

Éric Goubault
Samuel Mimram
Christine Tasson
École Polytechnique / IRIF

Journées du GDR Topologie Algébrique
2 décembre 2016

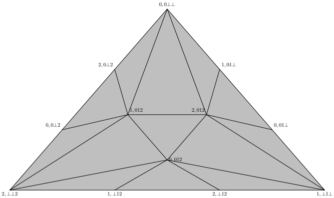
Asynchronous computability

I want to explain here our formulation of the major results obtained by Herlihy et al. in the 90s on asynchronous computability.

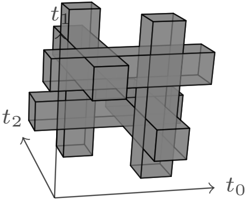
- What can a bunch of processes computing in parallel can compute in the presence of failures?
- For instance, they show that the consensus cannot be solved.
- Their proofs uses geometric arguments, they construct a simplicial complex encoding the possible states and
  - characterize those which can occur and their properties
  - obtain impossibility results from the fact that some maps should preserve (n-)connectivity
- The devil lies in the details.

Unifying points of view

We unify different points of view on executions:



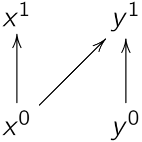
protocol complex
[Herlihy, . . .]



geometric semantics
[Goubault, . . .]

$$\left\langle u_i, s_i \;\middle|\; u_i u_j = u_j u_i, s_i s_j = s_j s_i \right\rangle$$

partially commutative traces



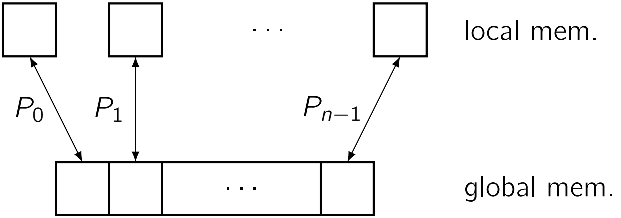
interval orders

ASYNCHRONOUS PROTOCOLS AND TASKS

Asynchronous protocols

We consider here a model with n processes P_i:

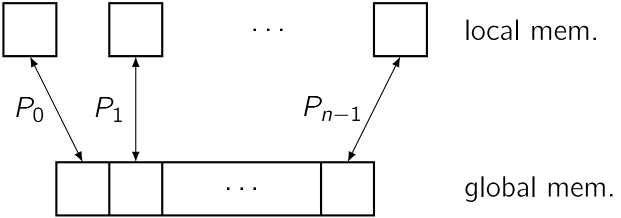
- each process has a local memory cell
- there is a global memory with n cells



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

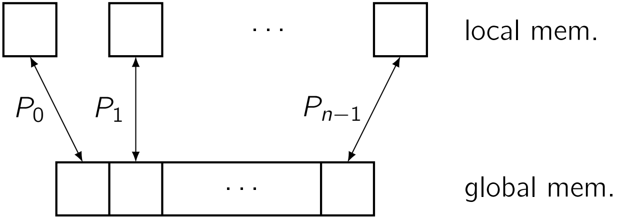


- each process alternatively does "rounds" made of
  - `update`: write in its global memory cell
  - `scan`: read the whole global memory and update its local cell (*immediate snapshot*)

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

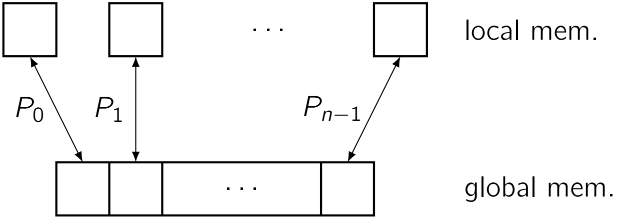


- each process alternatively does "rounds" made of
  - `update`: write in its global memory cell
  - `scan`: read the whole global memory and update its local cell (*immediate snapshot*)
- at any instant a process might die

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 “rounds” made of
  - `update`: write in its global memory cell
  - `scan`: read the whole global memory and update its local cell (*immediate snapshot*)
- at any instant a process might die
- and the question is: what we can compute in such a model? (for this question we are only interested in local memories)

Asynchronous protocols

Note that

- we do not know how the processes will be scheduled

Asynchronous protocols

Note that

- we do not know how the processes will be scheduled
- they might die: we cannot tell if a process is late or dead

Asynchronous protocols

Note that

- we do not know how the processes will be scheduled
- they might die: we cannot tell if a process is late or dead
- the local memory of each process is a partial information about the computation (called its **view**)

Asynchronous protocols

Note that

- we do not know how the processes will be scheduled
- they might die: we cannot tell if a process is late or dead
- the local memory of each process is a partial information about the computation (called its **view**)
- we are mostly interested in local memory: it contains the input and output values

Asynchronous protocols

Note that

- we do not know how the processes will be scheduled
- they might die: we cannot tell if a process is late or dead
- the local memory of each process is a partial information about the computation (called its **view**)
- we are mostly interested in local memory: it contains the input and output values
- the initial value for global memory is $\perp$ in every cell

Coherence between views

A set $X \subseteq \{(i, x) \mid i \in \mathbb{N}, x \in \mathcal{V}\}$ of local memories (= views) $(i, x) \in \mathbb{N} \times \mathcal{V}$ is **coherent** when

$$X = \{(i, l_i) \mid i \in I \subseteq \mathbb{N}\}$$

such that there is an execution leading to a local memory l.

We thus have a simplicial complex with

- vertices: views
- simplices: coherent views (result from a particular execution)

Coherence between views

A set $X \subseteq \{(i,x) \mid i \in \mathbb{N}, x \in \mathcal{V}\}$ of local memories (= views) $(i,x) \in \mathbb{N} \times \mathcal{V}$ is **coherent** when

$$X \quad = \quad \{(i, l_i) \mid i \in I \subseteq \mathbb{N}\}$$

such that there is an execution leading to a local memory l.

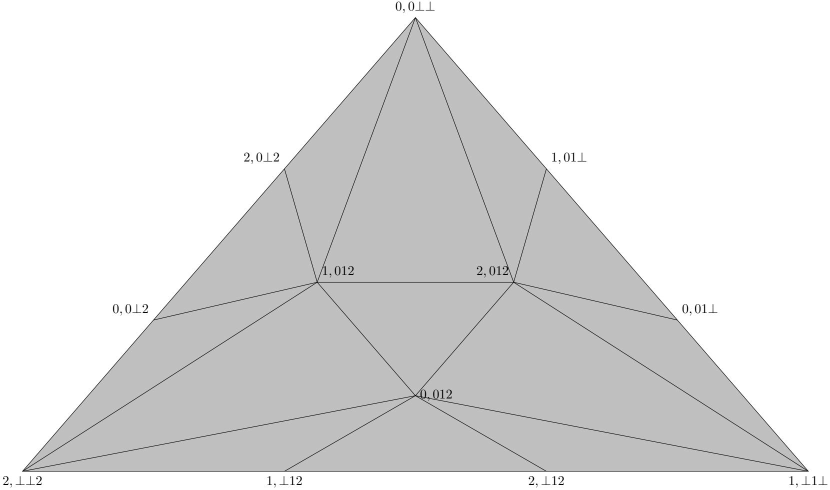
We thus have a simplicial complex with

- vertices: views
- simplices: coherent views (result from a particular execution)

Each vertex has a **color** $i \in \mathbb{N}$ and simplices have vertices of different colors.

Coherence between views

With 3 processes executing one round (update then scan), we typically obtain the following simplicial complex:



Notice that it is simply connected.

States

Formally, we suppose fixed a number $n \in \mathbb{N}$ of processes and a set $\mathcal{V}$ of **values** with

- $\mathcal{I} \subseteq \mathcal{V}$: *input values*
- $\mathcal{O} \subseteq \mathcal{V}$: *output values*
- $\perp \in \mathcal{I} \cap \mathcal{O}$: the *undefined value* / a *non-participating process*

States

Formally, we suppose fixed a number $n \in \mathbb{N}$ of processes and a set $\mathcal{V}$ of **values** with

- $\mathcal{I} \subseteq \mathcal{V}$: *input values*
- $\mathcal{O} \subseteq \mathcal{V}$: *output values*
- $\bot \in \mathcal{I} \cap \mathcal{O}$: the *undefined value* / a *non-participating process*

A **state** consists of

- $l \in \mathcal{V}^n$: the *local memories*
- $m \in \mathcal{V}^n$: the *global memories*

(always in this order)

States

Formally, we suppose fixed a number $n \in \mathbb{N}$ of processes and a set $\mathcal{V}$ of **values** with

- $\mathcal{I} \subseteq \mathcal{V}$: *input values*
- $\mathcal{O} \subseteq \mathcal{V}$: *output values*
- $\bot \in \mathcal{I} \cap \mathcal{O}$: the *undefined value* / a *non-participating process*

A **state** consists of

- $l \in \mathcal{V}^n$: the *local memories*
- $m \in \mathcal{V}^n$: the *global memories*

(always in this order)

The *standard* initial state has $l_i = i$ and $m_i = \bot$.

Protocols

A **protocol** π consists of, for $0 \leq i < n$,

- $\pi_{u_i} : \mathcal{V} \to \mathcal{V}$
  the values it will write in its global memory cell depending on its local memory
- $\pi_{s_i} : \mathcal{V} \times \mathcal{V}^n \to \mathcal{V}$
  the values it will write in its local memory depending on the values of its local memory and all the global memory cells

such that

- $\pi_{s_i}(x, m) = x$ for $x \in \mathcal{O}$
  once we decide an output we don't change our mind

Execution traces

The set of possible **actions** is

$$\mathcal{A} \quad = \quad \{u_i, s_i, d_i \mid 0 \leq i < n\}$$

Execution traces

The set of possible **actions** is

$$\mathcal{A} \quad = \quad \{u_i, s_i, d_i \mid 0 \leq i < n\}$$

The monoid $\mathcal{A}^*$ acts on states $\mathcal{V}^n \times \mathcal{V}^n$ as follows.

Execution traces

The set of possible **actions** is

$$\mathcal{A} \quad = \quad \{u_i, s_i, d_i \mid 0 \leq i < n\}$$

An **execution trace** is a word in $\mathcal{A}^*$ which is *well-bracketed*:

$$\mathrm{proj}_i(T) \quad \in \quad (u_i s_i)^*(\varepsilon + u_i d_i)$$

Execution traces

The set of possible **actions** is

$$\mathcal{A} \quad = \quad \{u_i, s_i, d_i \mid 0 \leq i < n\}$$

An **execution trace** is a word in $\mathcal{A}^*$ which is *well-bracketed*:

$$\text{proj}_i(T) \quad \in \quad (u_i s_i)^*(\varepsilon + u_i d_i)$$

Given a protocol π, its **semantics**

$$[\![T]\!]_\pi : \mathcal{V}^n \times \mathcal{V}^n \to \mathcal{V}^n \times \mathcal{V}^n$$

is defined on a trace $T \in \mathcal{A}^*$ by

- $[\![u_i]\!]_\pi(l, m) = (l, m[i \leftarrow \pi_{u_i}(l_i)])$
- $[\![s_i]\!]_\pi(l, m) = (l[i \leftarrow \pi_{s_i}(l_i, m)], m)$
- $[\![d_i]\!]_\pi(l, m) = (l, m)$

Execution traces

The set of possible **actions** is

$$\mathcal{A} \quad = \quad \{u_i, s_i \quad \mid 0 \leq i < n\}$$

An **execution trace** is a word in $\mathcal{A}^*$ which is *well-bracketed*:

$$\mathrm{proj}_i(T) \quad \in \quad (u_i s_i)^*(\varepsilon + u_i d_i)$$

Given a protocol π, its **semantics**

$$[\![T]\!]_\pi : \mathcal{V}^n \times \mathcal{V}^n \to \mathcal{V}^n \times \mathcal{V}^n$$

is defined on a trace $T \in \mathcal{A}^*$ by

- $[\![u_i]\!]_\pi(l, m) = (l, m[i \leftarrow \pi_{u_i}(l_i)])$
- $[\![s_i]\!]_\pi(l, m) = (l[i \leftarrow \pi_{s_i}(l_i, m)], m)$

Execution traces

With two processes executing one round each there are "essentially" three traces:

- $u_0 s_0 u_1 s_1$:
  - P_0 does not see what P_1 has written
  - P_1 sees what P_0 has written

Execution traces

With two processes executing one round each there are "essentially" three traces:

- $u_0 s_0 u_1 s_1$:
  - P_0 does not see what P_1 has written
  - P_1 sees what P_0 has written
- $u_1 s_1 u_0 s_0$:
  - P_0 sees what P_1 has written
  - P_1 does not see what P_0 has written

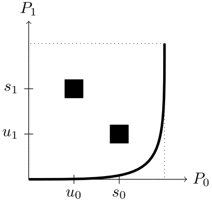
Execution traces

With two processes executing one round each there are "essentially" three traces:

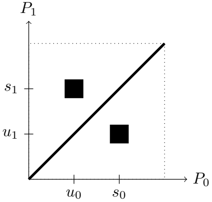
- $u_0 s_0 u_1 s_1$:
  - P_0 does not see what P_1 has written
  - P_1 sees what P_0 has written
- $u_1 s_1 u_0 s_0$:
  - P_0 sees what P_1 has written
  - P_1 does not see what P_0 has written
- $u_0 u_1 s_0 s_1$ / $u_0 u_1 s_1 s_0$ / $u_1 u_0 s_0 s_1$ / $u_1 u_0 s_1 s_0$:
  - P_0 sees what P_1 has written
  - P_1 sees what P_0 has written

Execution traces

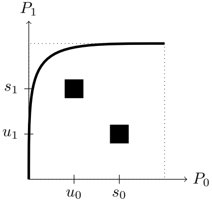
These execution traces can be represented geometrically by



$u_0 s_0 u_1 s_1$



$u_0 u_1 s_0 s_1$

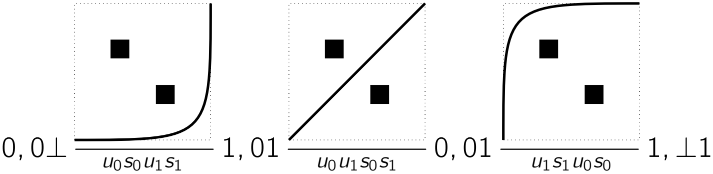


$u_1 s_1 u_0 s_0$

We'll get back to this representation later on.

Execution traces

These execution traces can be represented geometrically by



We'll get back to this representation later on.

Tasks

A **task** θ is a relation $\theta \subseteq \mathcal{I}^n \times \mathcal{O}^n$ such that for every $I, I' \in \Theta$

- $I_i = \bot$ if and only if $I'_i = \bot$,
- there exists $I'' \in \mathcal{O}^n$ such that $(I, I'') \in \Theta$ and $(I[i \leftarrow \bot], I''[i \leftarrow \bot]) \in \Theta$.

We write $\operatorname{dom}\Theta$ for the possible input values and $\operatorname{codom}\Theta$ for the possible output values.

The binary consensus

In the **binary consensus** problem each process

- starts with a value in $\{0, 1\}$
- end with the same value, among the initial values of the alive processes.

For instance, with $n = 2$, we have

$$\Theta \quad = \quad \left\{ (b\bot, b\bot), (\bot b, \bot b), (bb', bb), (b'b, bb) \;\middle|\; b, b' \in \{0, 1\} \right\}$$

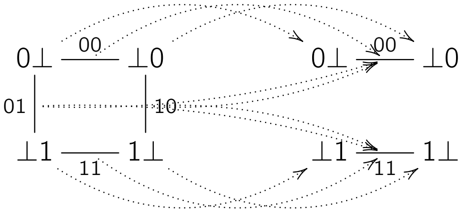
The binary consensus

In the **binary consensus** problem each process

- starts with a value in $\{0, 1\}$
- end with the same value, among the initial values of the alive processes.

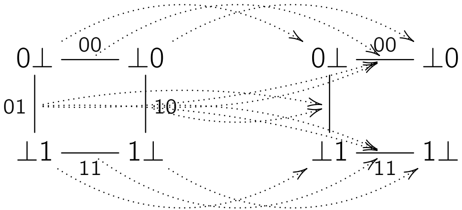
For instance, with $n = 2$, we have

$$\Theta \quad = \quad \left\{(b\bot, b\bot), (\bot b, \bot b), (bb', bb), (b'b, bb) \;\middle|\; b, b' \in \{0, 1\}\right\}$$



The binary quasi-consensus

In the case $n = 2$, we can also consider the **binary quasi-consensus**, which is similar but restricts the output so that it cannot happen that P_1 decides 0 and P_0 decide 1 at the same time:

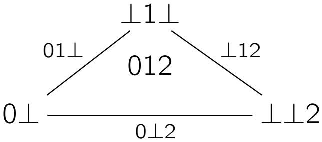


The way we draw tasks

Note that

- if $I \in \operatorname{dom}\Theta$ (the possible input values) then $I[i \leftarrow \bot]$ also belongs to $\operatorname{dom}\Theta$

$\operatorname{dom}\Theta$ can thus be pictured as a *simplicial complex* called the **input complex**:



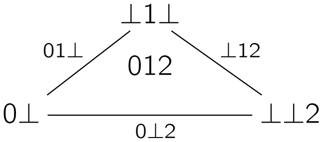
i.e. roughly a space made of triangles, tetrahedra, etc.
(and similarly $\operatorname{codom}\Theta$ gives rise to the **output complex**)

The way we draw tasks

Note that

- if $I \in \operatorname{dom}\Theta$ (the possible input values) then $I[i \leftarrow \bot]$ also belongs to $\operatorname{dom}\Theta$

$\operatorname{dom}\Theta$ can thus be pictured as a *simplicial complex* called the **input complex**:



i.e. roughly a space made of triangles, tetrahedra, etc.
(and similarly $\operatorname{codom}\Theta$ gives rise to the **output complex**)

Note also that the vertices are **colored** by $0 \leq i < n$:
the only active process

Tasks

A **task** θ is a relation $\theta \subseteq \mathcal{I}^n \times \mathcal{O}^n$ such that for every $I, I' \in \Theta$

1. $I_i = \bot$ if and only if $I'_i = \bot$,
2. there exists $I'' \in \mathcal{O}^n$ such that $(I, I'') \in \Theta$ and $(I[i \leftarrow \bot], I''[i \leftarrow \bot]) \in \Theta$.

which means

1. n-simplices are in relation with n-simplices
2. the relation is compatible with faces

Solving tasks

A protocol π **solves** a task Θ when

- for every initial local memory $l \in \operatorname{dom} \Theta$
- for every long enough and fair execution trace T

we have $l' \in \operatorname{codom} \Theta$, where

$$(l', m') \quad = \quad [\![T]\!]_\pi(l, \bot\bot \ldots \bot)$$

Solving tasks

A protocol π **solves** a task Θ when

- for every initial local memory $l \in \operatorname{dom}\Theta$
- for every long enough and fair execution trace T

we have $l' \in \operatorname{codom}\Theta$, where

$$(l', m') \quad = \quad [\![T]\!]_\pi(l, \bot\bot\ldots\bot)$$

For instance,

- the consensus cannot be solved
- the quasi-consensus can be solved

Let's understand why.

Solving tasks

A protocol π **solves** a task Θ when

- for every initial local memory $l \in \operatorname{dom} \Theta$
- for every long enough and fair execution trace T

we have $l' \in \operatorname{codom} \Theta$, where

$$(l', m') \quad = \quad [\![T]\!]_\pi(l, \bot\bot\ldots\bot)$$

For simplicity, we will suppose that $l_i = i$ initially (standard state) and thus write $[\![T]\!]_\pi$ instead of $[\![T]\!]_\pi(01\ldots(n-1), \bot\bot\ldots\bot)$.

For instance,

- the consensus cannot be solved
- the quasi-consensus can be solved

Let's understand why.

A more manageable setting

In order to study tasks which can be solved by protocols we should simplify as much as possible what we consider as

- protocols
- execution traces

Restricting executions

It can be shown that we can, without loss of generality, restrict to traces which are

- *well-bracketed*:

$$u_0\, u_1\, s_1\, u_2\, s_0\, s_2 \qquad \text{but not} \qquad u_0\, u_0\, s_1\, s_0$$

Restricting executions

It can be shown that we can, without loss of generality, restrict to traces which are

- *well-bracketed*:

$$u_0\, u_1\, s_1\, u_2\, s_0\, s_2 \qquad \text{but not} \qquad u_0\, u_0\, s_1\, s_0$$

- *layered*: a process does not start a round before all other have finished their or died

$$u_0\, s_0\, u_1\, s_1\, u_1\, u_0\, s_0\, s_1 \qquad \text{but not} \qquad u_0\, u_1\, s_0\, u_0\, s_1\, s_0$$

  In particular, we have a notion of *round*.

Restricting executions

It can be shown that we can, without loss of generality, restrict to traces which are

- *well-bracketed*:

$$u_0 u_1 s_1 u_2 s_0 s_2 \qquad \text{but not} \qquad u_0 \, u_0 \, s_1 \, s_0$$

- *layered*: a process does not start a round before all other have finished their or died

$$u_0 s_0 u_1 s_1 u_1 u_0 s_0 s_1 \qquad \text{but not} \qquad u_0 u_1 s_0 \, u_0 \, s_1 s_0$$

  In particular, we have a notion of *round*.

- *immediate snapshot*:

$$u_0 u_1 s_1 s_0 u_2 s_2 \qquad \text{but not} \qquad u_0 u_1 s_0 \, u_2 \, s_1 s_2$$

Full-information protocols

A protocol is **full-information** when

$$\pi_{u_i} \quad = \quad \mathrm{id}_{\mathcal{V}}$$

We can restrict to those without loss of generality (and we will).

A category of protocols

A **morphism** $\phi : \pi \to \pi'$ between protocols consists of functions

- $\phi_i : \mathcal{V} \to \mathcal{V}$ translating memory

such that

- $\phi_i(x) = x$ for $x \in \mathcal{I}$
- $\phi_i(x) \in \mathcal{O}$ for $x \in \mathcal{O}$
- and

$$\begin{array}{ccc} \mathcal{V} \times \mathcal{V}^n & \xrightarrow{\pi_{s_i}} & \mathcal{V} \\ {\scriptstyle \phi_i \times \prod_i \phi_i}\downarrow & & \downarrow{\scriptstyle \phi_i} \\ \mathcal{V} \times \mathcal{V}^n & \xrightarrow[\pi_{s_i}]{} & \mathcal{V} \end{array}$$

We say that π' *simulates* π.

A category of protocols

A **morphism** $\phi : \pi \to \pi'$ between protocols consists of functions

- $\phi_i : \mathcal{V} \to \mathcal{V}$ translating memory

such that

- $\phi_i(x) = x$ for $x \in \mathcal{I}$
- $\phi_i(x) \in \mathcal{O}$ for $x \in \mathcal{O}$
- and

$$\begin{array}{ccc} \mathcal{V} \times \mathcal{V}^n & \xrightarrow{\pi_{s_i}} & \mathcal{V} \\ {\scriptstyle \phi_i \times \prod_i \phi_i} \downarrow & & \downarrow {\scriptstyle \phi_i} \\ \mathcal{V} \times \mathcal{V}^n & \xrightarrow[\pi_{s_i}]{} & \mathcal{V} \end{array}$$

We say that π' *simulates* π.

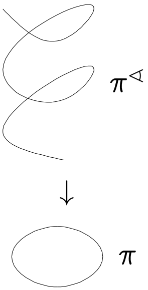
Actually, we only require ϕ_i to be defined on *reachable* values for a given task.

The view protocol

Theorem (GMT)

The category of protocols admits an initial object $\pi^{\triangleleft}$.

Morally, the space of executions of $\pi^{\triangleleft}$ is the “universal cover” of the space of executions of any process π: every execution of π corresponds to a unique execution of $\pi^{\triangleleft}$.



The view protocol

We suppose that $\mathcal{V}$ is countable so that we have an encoding $\langle x, y \rangle$ of pairs (and uples).

The view protocol

We suppose that $\mathcal{V}$ is countable so that we have an encoding $\langle x, y \rangle$ of pairs (and uples).

The initial object $\pi^{\lhd}$ is called the **view protocol** and is defined by

- $\pi^{\lhd}_{u_i}(x) = x$ for $x \in \mathcal{V}$ (full-information),
- $\pi^{\lhd}_{s_i}(x, m) = \langle x, \langle m \rangle \rangle$ for $(x, m) \in \mathcal{V} \times \mathcal{V}^n$.

The view protocol

We suppose that $\mathcal{V}$ is countable so that we have an encoding $\langle x, y \rangle$ of pairs (and uples).

The initial object $\pi^{\lhd}$ is called the **view protocol** and is defined by

- $\pi^{\lhd}_{u_i}(x) = x$ for $x \in \mathcal{V}$ (full-information),
- $\pi^{\lhd}_{s_i}(x, m) = \langle x, \langle m \rangle \rangle$ for $(x, m) \in \mathcal{V} \times \mathcal{V}^n$.

Given a trace T, the local memory of i-th process after executing the trace T is called its **view**.

The view protocol

Theorem (GMT)

The category of protocols admits an initial object $\pi^{\triangleleft}$ with $\pi^{\triangleleft}_{s_i}(x, m) = \langle x, \langle m \rangle \rangle$.

Proof.

Suppose given a reachable memory

$$x = l_i \qquad \text{with} \qquad (l, m) = [\![T]\!]_{\pi^{\triangleleft}}$$

Because of the definition of morphisms, we are forced to define

$$\phi_i(x) = l'_i \qquad \text{with} \qquad (l', m') = [\![T]\!]_{\pi}$$

It only remains to check that this definition is well-defined, i.e. it does not depend on the chosen trace T... □

THE PROTOCOL COMPLEX

The protocol complex

Given a number r of rounds for each process, the **protocol complex** $\chi^r(\Theta)$ is the abstract simplicial complex whose

- vertices are $x \in \mathcal{V}$ such that x is the view (= local memory) of i-th process after executing a trace with $\pi^{\lhd}$
- simplices are sets of vertices occurring together after a same execution.

The protocol complex

Suppose that we have 2 processes and the input is the standard one:

$$0 \text{ ———————— } 1$$

The protocol complex $\chi^1(\Theta)$ for 1 round is as follows:

The protocol complex

Suppose that we have 2 processes and the input is the standard one:

0 ———————————— 1

The protocol complex $\chi^1(\Theta)$ for 1 round is as follows:

$0, 0\bot$ ——— $1, 01$

After executing 1 round for each process, we have the executions

- $u_0 s_0 u_1 s_1$:

0	1
$\bot$	$\bot$

$\xrightarrow{u_0}$

0	1
0	$\bot$

$\xrightarrow{s_0}$

$0, 0\bot$	1
0	$\bot$

$\xrightarrow{u_1}$

$0, 0\bot$	1
0	1

$\xrightarrow{s_1}$

$0, 0\bot$	$1, 01$
0	1

The protocol complex

Suppose that we have 2 processes and the input is the standard one:

$$0 \text{ ——————————— } 1$$

The protocol complex $\chi^1(\Theta)$ for 1 round is as follows:

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

After executing 1 round for each process, we have the executions

- $u_1 s_1 u_0 s_0$:

0	1
$\bot$	$\bot$

$\xrightarrow{u_1}$

0	1
$\bot$	1

$\xrightarrow{s_1}$

0	1, $\bot$1
$\bot$	1

$\xrightarrow{u_0}$

0	1, $\bot$1
0	1

$\xrightarrow{s_0}$

0, 01	1, $\bot$1
0	1

The protocol complex

Suppose that we have 2 processes and the input is the standard one:

$$0 \text{ ———————— } 1$$

The protocol complex $\chi^1(\Theta)$ for 1 round is as follows:

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

After executing 1 round for each process, we have the executions

- $u_1 u_1 s_0 s_1$:

0	1
$\bot$	$\bot$

$\xrightarrow{u_0}$

0	1
0	$\bot$

$\xrightarrow{u_1}$

0	1
0	1

$\xrightarrow{s_0}$

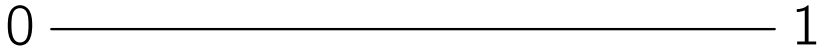
0, 01	1
0	1

$\xrightarrow{s_1}$

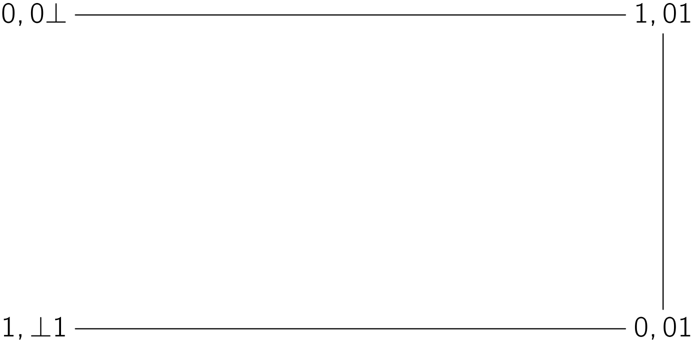
0, 01	1, 01
0	1

The protocol complex

Suppose that we have 2 processes and the input is the standard one:



The protocol complex $\chi^1(\Theta)$ for 1 round is as follows:

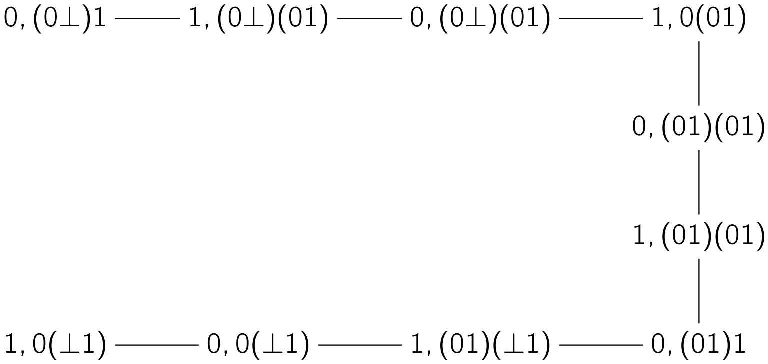


The protocol complex

Suppose that we have 2 processes and the input is the standard one:

$$0 \text{ ——— } 1$$

The protocol complex $\chi^2(\Theta)$ for 2 rounds is as follows:

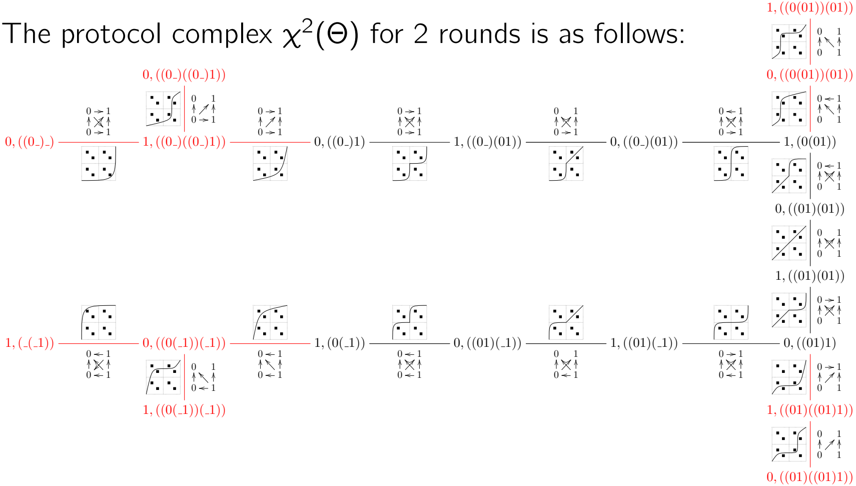


The protocol complex

Suppose that we have 2 processes and the input is the standard one:

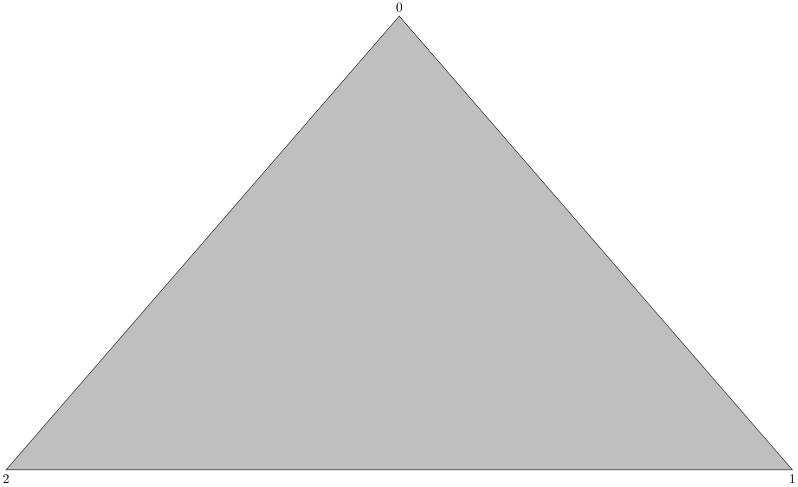
$$0 \text{ ——————— } 1$$

The protocol complex $\chi^2(\Theta)$ for 2 rounds is as follows:



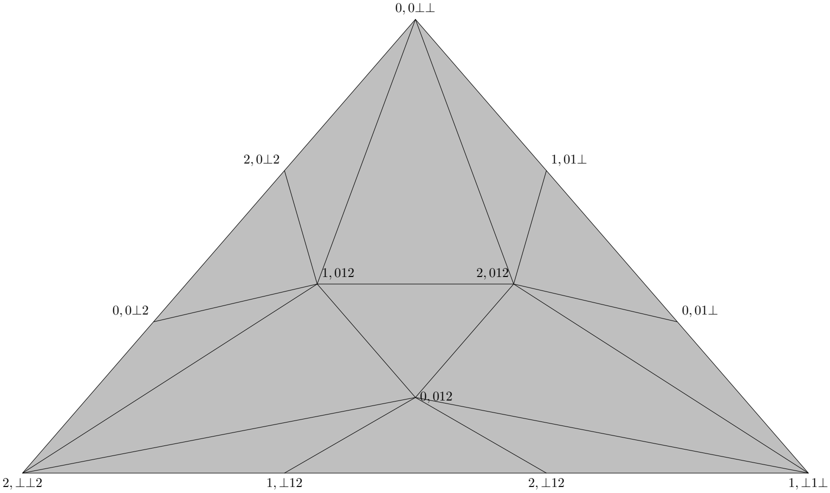
The protocol complex

With 3 processes and 1 one round, starting from the input complex



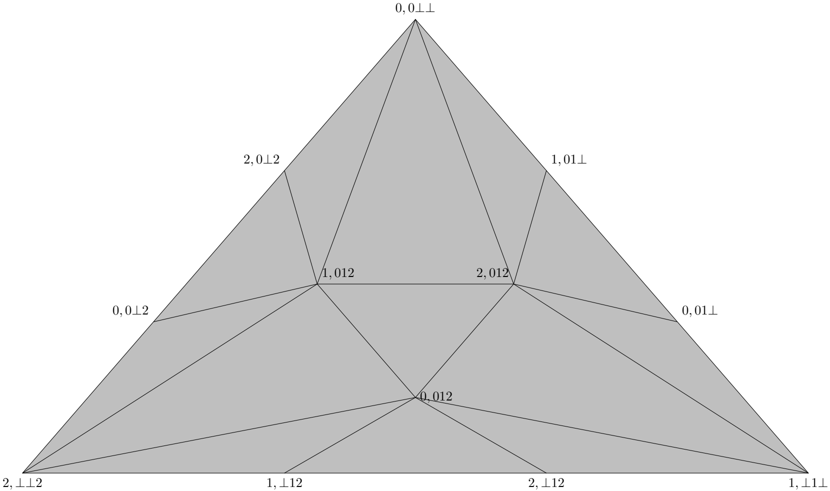
The protocol complex

With 3 processes and 1 one round, starting from the input complex we obtain the protocol complex



The protocol complex

With 3 processes and 1 one round, starting from the input complex we obtain the protocol complex



Notice that this is a particular subdivision of the original complex.

The chromatic subdivision

In general, the protocol complex on r rounds is obtained by

- starting from the input complex
- performing a **chromatic subdivision** of it r times

and this subdivision can be defined and studied independently.

The chromatic subdivision

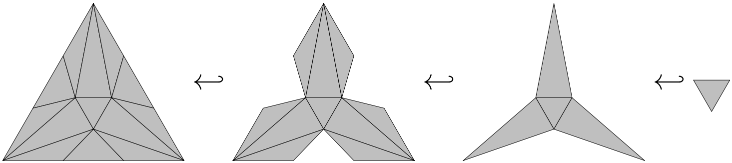
In general, the protocol complex on r rounds is obtained by

- starting from the input complex
- performing a **chromatic subdivision** of it r times

and this subdivision can be defined and studied independently.

Theorem (Herliy-Shavit, GMT, Koszlov)

If the input complex is contractible then the protocol complex is.



Solvability

Suppose that a task Θ can be solved by a protocol π:

- it can be solved in r rounds

Solvability

Suppose that a task Θ can be solved by a protocol π:

- it can be solved in r rounds
- there is a map $\phi : \pi^{\triangleleft} \to \pi$ such that, for every trace T,

$$\phi(\llbracket T \rrbracket_{\pi^{\triangleleft}}) \quad = \quad \llbracket T \rrbracket_{\pi}$$

Solvability

Suppose that a task Θ can be solved by a protocol π:

- it can be solved in r rounds
- there is a map $\phi : \pi^{\triangleleft} \to \pi$ such that, for every trace T,

$$\phi(\llbracket T \rrbracket_{\pi^{\triangleleft}}) \quad = \quad \llbracket T \rrbracket_{\pi}$$

- in particular, when the trace T has r rounds $\llbracket T \rrbracket \in \mathcal{O}$

Solvability

Suppose that a task Θ can be solved by a protocol π:

- it can be solved in r rounds
- there is a map $\phi : \pi^{\triangleleft} \to \pi$ such that, for every trace T,

$$\phi(\llbracket T \rrbracket_{\pi^{\triangleleft}}) \quad = \quad \llbracket T \rrbracket_{\pi}$$

- in particular, when the trace T has r rounds $\llbracket T \rrbracket \in \mathcal{O}$
- [...] therefore there is a simplicial map from the r-iterated protocol complex to the output complex:

Theorem

If a task can be solved then there is r and a simplicial map from $\chi^r(\Theta)$ to $\operatorname{codom}\Theta$ (and, in fact, conversely).

Solvability

Suppose that a task Θ can be solved by a protocol π:

- it can be solved in r rounds
- there is a map $\phi : \pi^{\triangleleft} \to \pi$ such that, for every trace T,

$$\phi(\llbracket T \rrbracket_{\pi^{\triangleleft}}) \quad = \quad \llbracket T \rrbracket_{\pi}$$

- in particular, when the trace T has r rounds $\llbracket T \rrbracket \in \mathcal{O}$
- [...] therefore there is a simplicial map from the r-iterated protocol complex to the output complex:

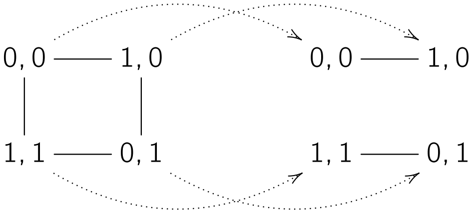
Theorem

If a task can be solved then there is r and a simplicial map from $\chi^r(\Theta)$ *to* $\operatorname{codom}\Theta$ *(and, in fact, conversely).*

NB: simplicial maps preserve contractibility!

The binary consensus

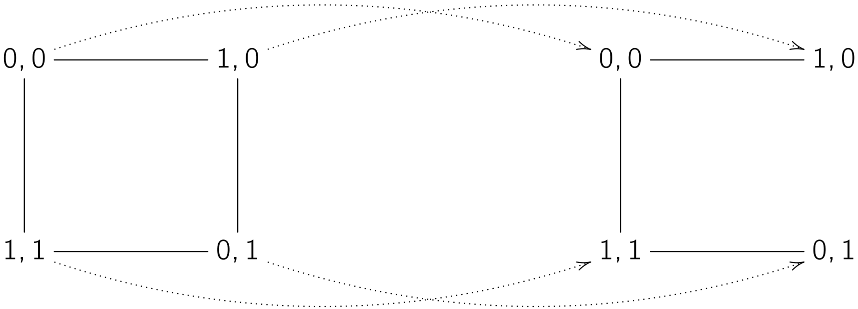
Consider again the **binary consensus** task:



There can be no protocol solving it (even after some rounds).

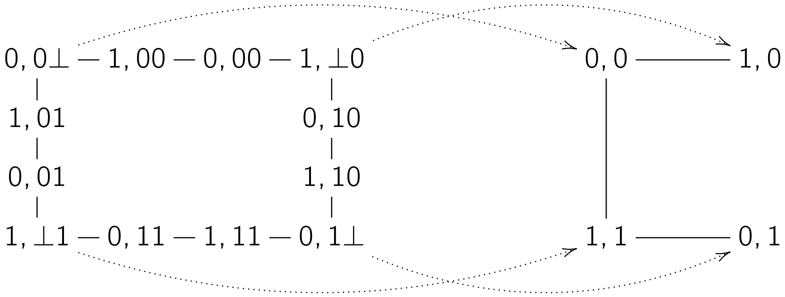
The binary quasi-consensus

Consider the **binary quasi-consensus**:



The binary quasi-consensus

Consider the **binary quasi-consensus**:



CONTRACTIBILITY OF THE PROTOCOL COMPLEX

Simplicial complex

Definition

A **simplicial complex** K consists of

- a set $\underline{K}$ of *vertices*,
- a set K of finite subsets of $\underline{K}$ called *simplices*,

such that

- K is non-empty,
- for every $x \in \underline{K}$, we have $\{x\} \in K$,
- for every $\sigma \in K$ and $\tau \subseteq \sigma$ we have $\tau \in K$.

Simplicial complex

Definition

A **simplicial complex** K consists of

- a set $\underline{K}$ of *vertices*,
- a set K of finite subsets of $\underline{K}$ called *simplices*,

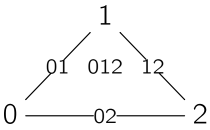
such that

- K is non-empty,
- for every $x \in \underline{K}$, we have $\{x\} \in K$,
- for every $\sigma \in K$ and $\tau \subseteq \sigma$ we have $\tau \in K$.

Example

The **standard simplicial complex** Δ^n has $\{0, \ldots, n\}$ as vertices and all possible simplices.

$\Delta^2 \quad =$



Simplicial complex

Definition

A **simplicial complex** K consists of

- a set $\underline{K}$ of *vertices*,
- a set K of finite subsets of $\underline{K}$ called *simplices*,

such that

- K is non-empty,
- for every $x \in \underline{K}$, we have $\{x\} \in K$,
- for every $\sigma \in K$ and $\tau \subseteq \sigma$ we have $\tau \in K$.

A morphism

$$f : K \to K'$$

is a function $f : \underline{K} \to \underline{K'}$ which

- preserves simplices: for $\sigma \in K$, we have $f(\sigma) \in K'$,
- is locally injective: for $\sigma \in K$, f restricted to σ is injective.

Towards the standard chromatic subdivision

Before defining the standard chromatic subdivision, we will first recall the barycentric subdivision.

For this, we need to introduce:

- the graph of elements of a simplicial complex,
- the nerve of a graph,
- the chromatic variants of these notions.

The graph of elements

Definition

A **graph** $G = (V, E)$ consists here of

- a set V of *vertices*,
- a set $E \subseteq V \times V$ of *edges*,

such that $(x, y) \in E$ implies $x \neq y$.

The graph of elements

Definition

A **graph** $G = (V, E)$ consists here of

- a set V of *vertices*,
- a set $E \subseteq V \times V$ of *edges*,

such that $(x, y) \in E$ implies $x \neq y$.

Definition

The **graph of elements** $\mathrm{El}(K)$ of a simplicial complex has

- the non-empty simplices of K as vertices,
- an edge $\tau \to \sigma$ whenever $\tau \subsetneq \sigma$.

The graph of elements

Example

For Δ^1

$$0 \;\text{———}\; 01 \;\text{———}\; 1$$

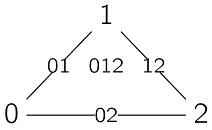
the graph of elements is

$$0 \longrightarrow 01 \longleftarrow 1$$

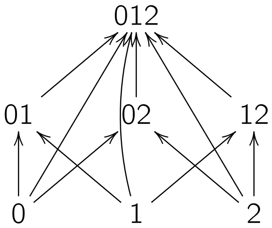
The graph of elements

Example

For Δ^2



the graph of elements is



The nerve of a graph

Definition

The **nerve** $N(G)$ of a graph $G = (V, E)$ has

- the elements of G as vertices,
- simplices are sets $\{x_0, \ldots, x_n\} \subseteq G$ such that there is an edge

$$x_i \to x_j$$

for every $i < j$.

The nerve of a graph

Definition

The **nerve** $N(G)$ of a graph $G = (V, E)$ has

- the elements of G as vertices,
- simplices are sets $\{x_0, \ldots, x_n\} \subseteq G$ such that there is an edge

$$x_i \to x_j$$

  for every $i < j$.

Example

The nerve of the graph

$$0 \longrightarrow 01 \longleftarrow 1$$

is

$$0 \text{ ------ } 01 \text{ ------ } 1$$

The barycentric subdivision

Definition

The **barycentric subdivision** of a simplicial complex is

$$\chi \quad = \quad N \circ \mathrm{El}$$

The barycentric subdivision

Example

For Δ^1

$$0 \longrightarrow 01 \longrightarrow 1$$

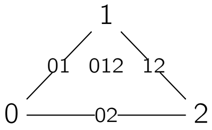
the barycentric subdivision is

$$0 \longrightarrow 01 \longrightarrow 1$$

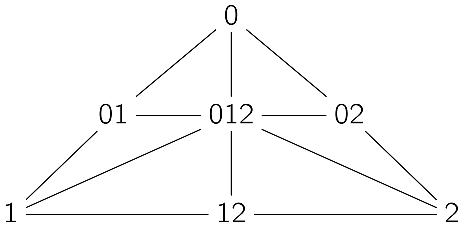
The barycentric subdivision

Example

For Δ^2



the barycentric subdivision is



Colored complexes

Definition

The category of **colored simplicial complexes** is

$$\mathbf{SC} / \, !\,\mathbb{N}$$

where $!\,\mathbb{N}$ has $\mathbb{N}$ as vertices and all finite subsets as simplices.

Remark

- The coloring of a simplicial complex K is uniquely determined by a coloring of vertices:

$$\ell \quad : \quad \underline{K} \quad \to \quad \mathbb{N}$$

- In a simplex, every vertex has a different color.

Colored graphs

We write $!\mathbb{N}$ for the graph with

- $\mathbb{N}$ as vertices,
- pairs $(x, y) \in \mathbb{N} \times \mathbb{N}$ with $x \neq y$ as edges.

We write $!\,\mathbb{N}$ for the graph with

- $\mathbb{N}$ as vertices,
- pairs $(x, y) \in \mathbb{N} \times \mathbb{N}$ with $x \neq y$ as edges.

Definition

The category of **colored graphs** is

$$\mathbf{Graph} / \, !\,\mathbb{N}$$

We thus color vertices by natural numbers in a way such that two vertices of an edge have a distinct color.

The chromatic graph of elements

Definition

The functor

$$\mathrm{El} \quad : \quad \mathbf{SC}/\,!\,\mathbb{N} \quad \to \quad \mathbf{Graph}/\,!\,\mathbb{N}$$

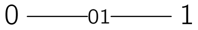
associates to each colored simplicial complex (K,ℓ) the graph where

- vertices are (σ, i) with $\sigma \in K$ and $i \in \ell(\sigma)$
- there is an edge $(\tau, i) \to (\sigma, j)$ whenever
  1. $i \neq j$
  2. $\tau \subseteq \sigma$
  3. $\tau = \sigma$ or $j \notin \ell(\tau)$

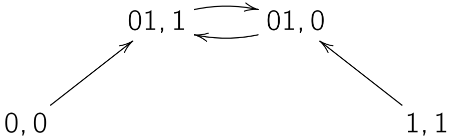
The chromatic graph of elements

Example

For Δ^1



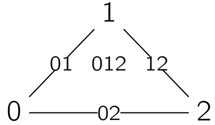
the chromatic graph of elements is



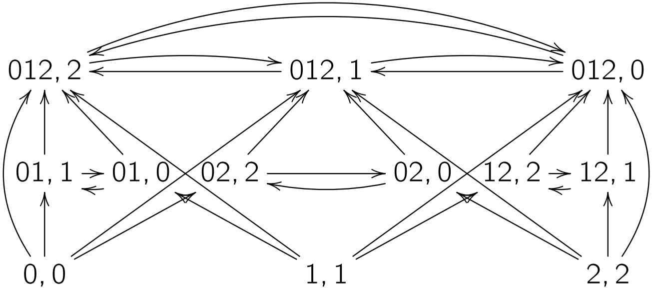
The chromatic graph of elements

Example

For Δ^2



the chromatic graph of elements is



The chromatic nerve

Definition
The functor

$$N \quad : \quad \mathbf{Graph}/\,!\,\mathbb{N} \quad \to \quad \mathbf{SC}/\,!\,\mathbb{N}$$

associates to a colored graph $(G = (V, E), \ell)$ the simplicial complex with

- the elements of G as vertices, colored by ℓ,
- simplices are sets $\{x_0, \ldots, x_n\} \subseteq G$ such that there is an edge

  $$x_i \to x_j$$

  for every $i < j$.

The standard chromatic subdivision

Definition

The **standard chromatic subdivision** is

$$\chi \quad = \quad N \circ \mathrm{El}$$

The standard chromatic subdivision

Example

For Δ^1

$$0 \overset{}{-\!\!-\!\!-\!\!-} 01 -\!\!-\!\!-\!\!- 1$$

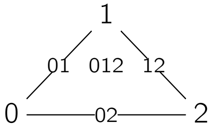
the standard chromatic subdivision is

$$0,0 -\!\!-\!\!-\!\!- 01,1 -\!\!-\!\!-\!\!- 01,0 -\!\!-\!\!-\!\!- 1,1$$

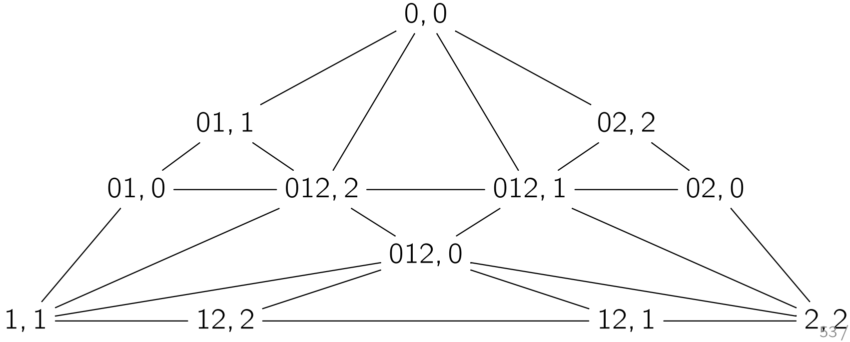
The standard chromatic subdivision

Example

For Δ^2



the standard chromatic subdivision is



We want to show that $\chi^r(K)$ is n-connected when K is.

This will be deduced from the fact that $\chi(\Delta^n)$ is contractible.

Which we prove by showing that $\chi(\Delta^n)$ is collapsible.

Collapsibility

From now on, we consider simplicial complexes K of finite dimension.

Definition

A simplex τ is a **free face** of a simplex σ when

1. $\tau \subseteq \sigma$ and $\tau \neq \sigma$,
2. σ is a maximal simplex of K,
3. no other maximal simplex of K contains τ.

Collapsibility

From now on, we consider simplicial complexes K of finite dimension.

Definition

A simplex τ is a **free face** of a simplex σ when

1. $\tau \subseteq \sigma$ and $\tau \neq \sigma$,
2. σ is a maximal simplex of K,
3. no other maximal simplex of K contains τ.

In this case, the monomorphism

$$K \quad \hookleftarrow \quad K \setminus \tau$$

is called a **collapse step**.

From now on, we consider simplicial complexes K of finite dimension.

Definition

A simplex τ is a **free face** of a simplex σ when

1. $\tau \subseteq \sigma$ and $\tau \neq \sigma$,
2. σ is a maximal simplex of K,
3. no other maximal simplex of K contains τ.

In this case, the monomorphism

$$K \quad \hookleftarrow \quad K \setminus \tau$$

is called a **collapse step**. A **collapse** is a composite of collapse steps.

Collapsibility

From now on, we consider simplicial complexes K of finite dimension.

Definition

A simplex τ is a **free face** of a simplex σ when

1. $\tau \subseteq \sigma$ and $\tau \neq \sigma$,
2. σ is a maximal simplex of K,
3. no other maximal simplex of K contains τ.

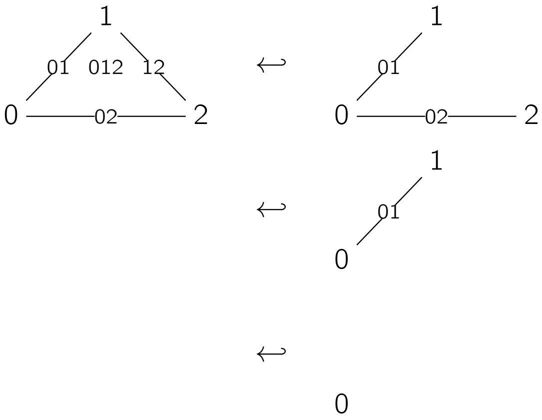
In this case, the monomorphism

$$K \quad \hookleftarrow \quad K \setminus \tau$$

is called a **collapse step**. A **collapse** is a composite of collapse steps. K is **collapsible** if it can be collapsed to Δ^0.

Example

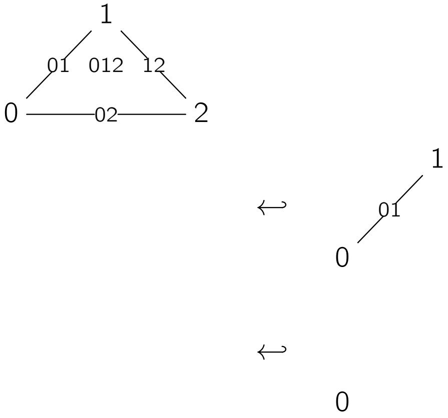
The simplex Δ^2 is collapsible:



Collapsibility

Example

The simplex Δ^2 is collapsible:



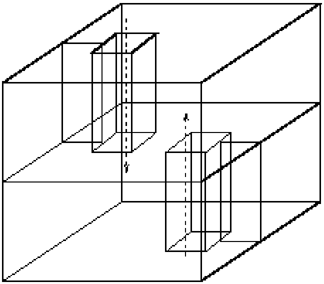
Theorem (Whitehead)

A collapsible simplicial complex is contractible.

Theorem (Whitehead)

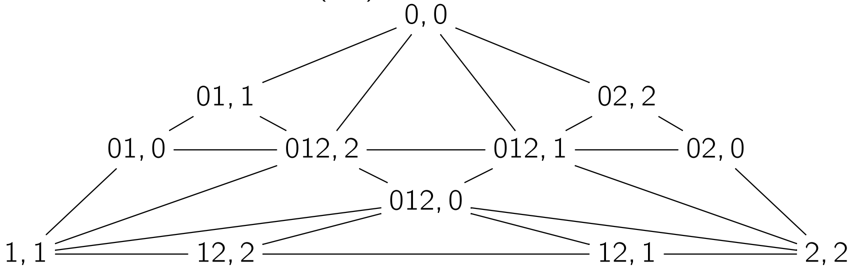
A collapsible simplicial complex is contractible.

The converse is not true, e.g. *Bing's house with two rooms*:

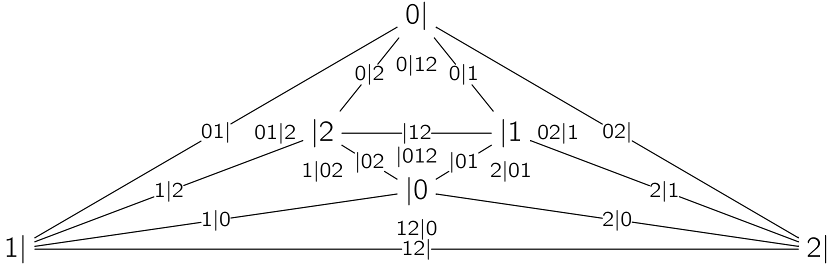


A simpler example

Instead of showing that $\chi(\Delta^n)$ is collapsible



we are going to show the result on $\partial\Delta^n \star \Delta^n$:



The join

Definition

Given simplicial complexes K and L, their **join** $K \star L$ is the complex with

- vertices

$$\underline{K \star L} \quad = \quad \underline{K} \uplus \underline{L}$$

- simplices

$$K \star L \quad = \quad \{\sigma \subseteq \underline{K} \uplus \underline{L} \mid \sigma \cap \underline{K} \in K \quad \text{and} \quad \sigma \cap \underline{L} \in L\}$$

A simplex in $K \star L$ is thus of the form $\sigma|\tau$ with $\sigma \in K$ and $\tau \in L$.

Example

$\Delta^m \star \Delta^n = \Delta^{m+n+1}$.

The colored join

Definition

Given colored simplicial complexes K and L, their **colored join** $K \star L$ is the complex with

- vertices

$$\underline{K \star L} \quad = \quad \underline{K} \uplus \underline{L}$$

- simplices

$$K \star L \quad = \quad \{\sigma | \tau \in K \times L \mid \ell_K(\sigma) \cap \ell_L(\tau) = \emptyset\}$$

The basic chromatic subdivision

Definition

The **basic chromatic subdivision** of Δ^n is

$$\partial\Delta^n \star \Delta^n$$

The basic chromatic subdivision

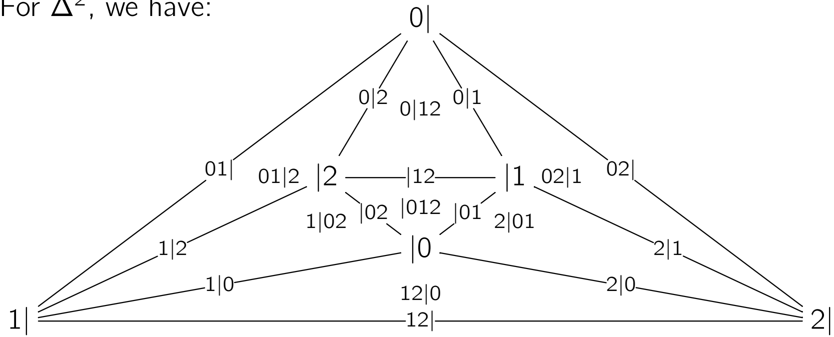
Definition

The **basic chromatic subdivision** of Δ^n is

$$\partial\Delta^n \star \Delta^n$$

Example

For Δ^2, we have:



The basic chromatic subdivision

Definition

The **basic chromatic subdivision** of Δ^n is

$$\partial\Delta^n \star \Delta^n$$

Its simplices are of the form $\sigma|\tau$ with

- $\sigma, \tau \subseteq \{0, \ldots, n\}$
- $\sigma \neq \{0, \ldots, n\}$
- $\sigma \cap \tau = \emptyset$

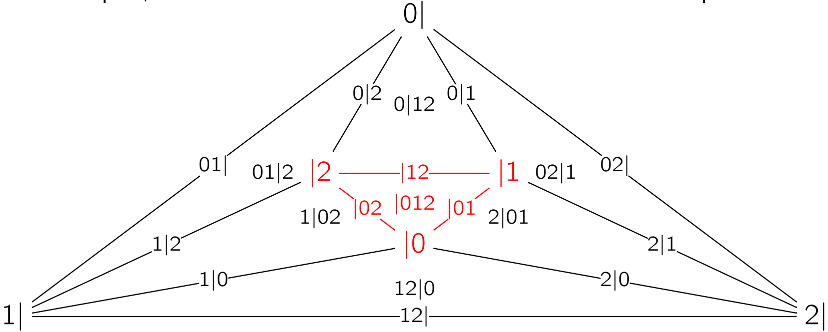
Collapsibility of the basic subdivision

Proposition

The canonical inclusion

$$\begin{array}{ccc} \Delta^I & \hookrightarrow & \partial\Delta^I \star \Delta^I = K^I \\ \sigma & \mapsto & \emptyset|\sigma \end{array}$$

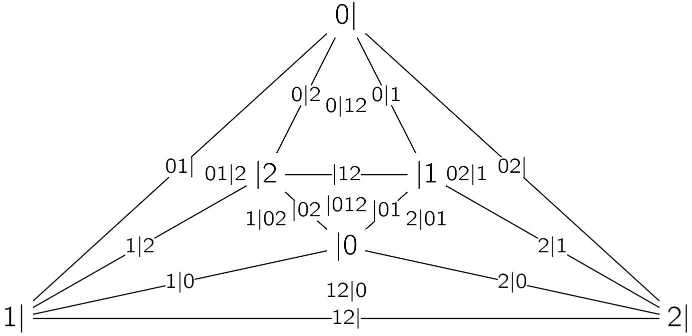
is a collapse, thus the basic chromatic subdivision is collapsible.



Proof: remove $\sigma|\emptyset$ with $\dim(\sigma)$ decreasing.

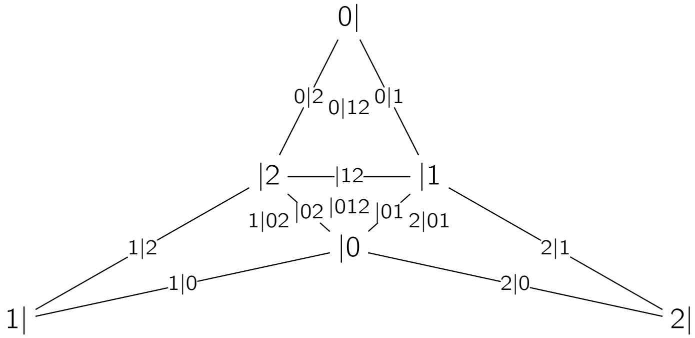
Collapsibility of the basic subdivision

We consider the following sequence of collapse steps:



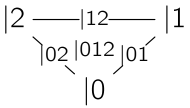
Collapsibility of the basic subdivision

We consider the following sequence of collapse steps:



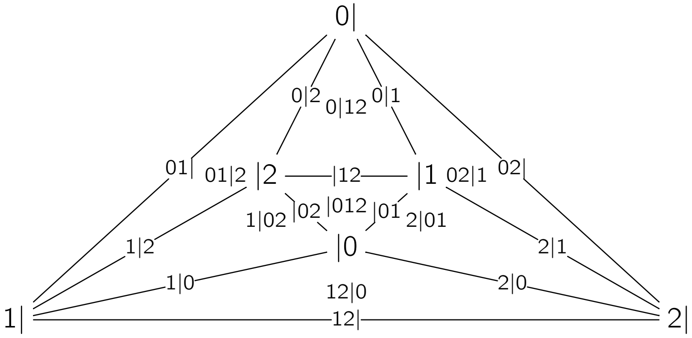
Collapsibility of the basic subdivision

We consider the following sequence of collapse steps:



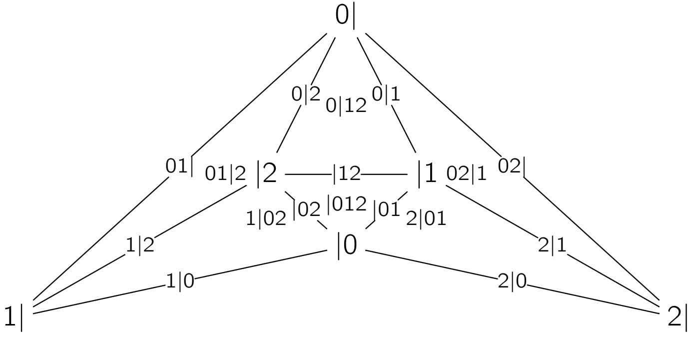
Collapsibility of the basic subdivision

Note that other sequences could have been used in order to show collapsibility:



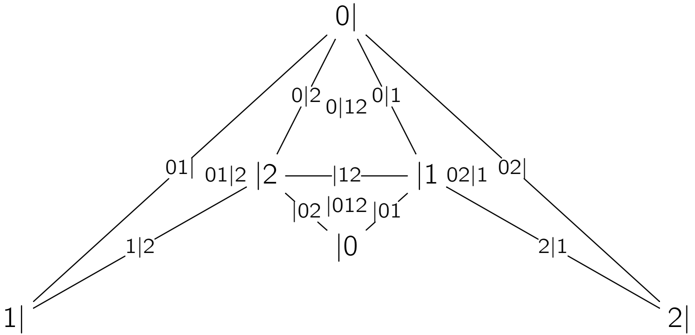
Collapsibility of the basic subdivision

Note that other sequences could have been used in order to show collapsibility:



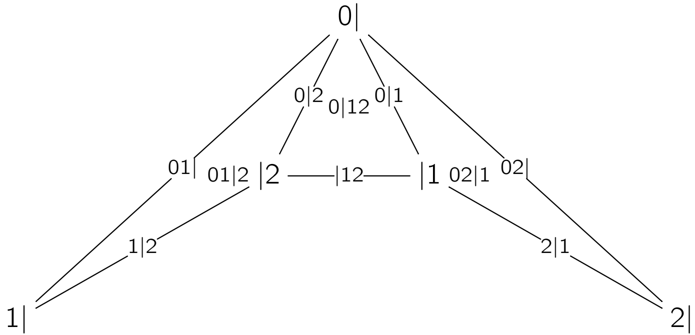
Collapsibility of the basic subdivision

Note that other sequences could have been used in order to show collapsibility:



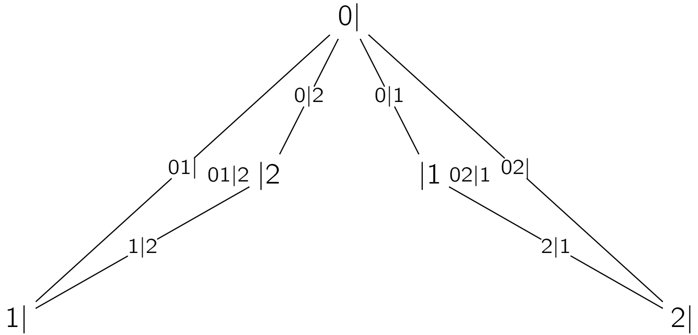
Collapsibility of the basic subdivision

Note that other sequences could have been used in order to show collapsibility:



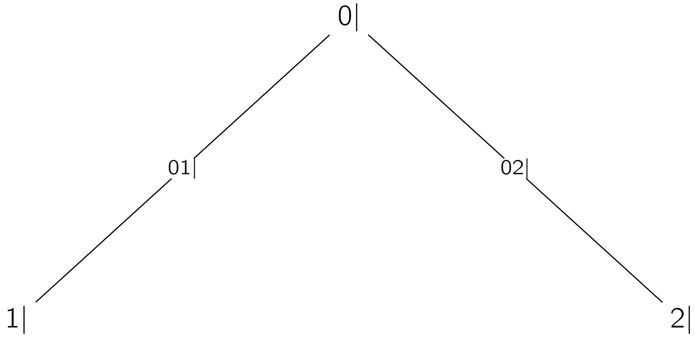
Collapsibility of the basic subdivision

Note that other sequences could have been used in order to show collapsibility:



Collapsibility of the basic subdivision

Note that other sequences could have been used in order to show collapsibility:



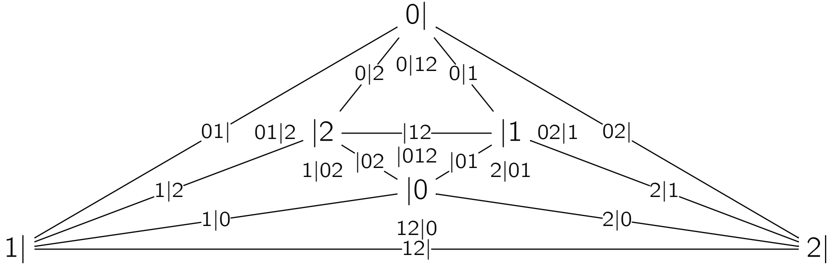
Collapsibility of the basic subdivision

Note that other sequences could have been used in order to show collapsibility:

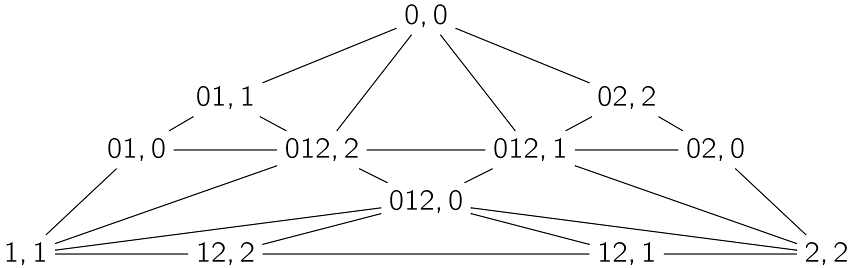
$$0|$$

Collapsibility of the chromatic subdivision

Comparing the basic chromatic subdivision



and the standard chromatic subdivision



we see that some Δ^2 are replaced by $\partial\Delta^1 \star \Delta^0$.

Collapsibility of the chromatic subdivision

Theorem

$\chi(\Delta^n)$ *is collapsible and thus contractible.*

Proof.

Show a bunch of lemmas showing that collapsing is compatible with join and simulate the previous sequence of collapse steps. □

The iterated subdivision

In order to show that the iterated subdivision is contractible, it is simpler to work with (colored) presimplicial sets:

- every elementary collapse step $K \hookrightarrow L$ can be obtained as a pushout

$$\begin{array}{ccc} \Lambda^n_i & \longrightarrow & K \\ \downarrow & & \downarrow \\ \Delta^n & \longrightarrow & L \end{array}$$

- the image of χ is characterized by its action on representables

$$\chi(K) \quad = \quad \operatorname{colim}(\mathsf{El}(K) \xrightarrow{\pi} \Delta \xrightarrow{Y} \hat{\Delta} \xrightarrow{\chi} \hat{\Delta})$$

The iterated subdivision

In order to show that the iterated subdivision is contractible, it is simpler to work with (colored) presimplicial sets:

- every elementary collapse step $K \hookrightarrow L$ can be obtained as a pushout

$$\begin{array}{ccc} \Lambda^n_i & \longrightarrow & K \\ \downarrow & & \downarrow \\ \Delta^n & \longrightarrow & L \end{array}$$

- the image of χ is characterized by its action on representables

$$\chi(K) \quad = \quad \operatorname{colim}(\mathsf{El}(K) \xrightarrow{\pi} \Delta \xrightarrow{Y} \hat{\Delta} \xrightarrow{\chi} \hat{\Delta})$$

Theorem

$\chi^r(\Delta^n)$ *is collapsible and thus contractible.*

EQUIVALENCE BETWEEN TRACES

Execution traces

The (well-bracketed) execution traces in $\{u_i, s_i\}^*$ are semantically invariant under the congruence $\approx$ generated by

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

which means that

$$T \approx T' \qquad \text{implies} \qquad [\![T]\!]_\pi = [\![T']\!]_\pi$$

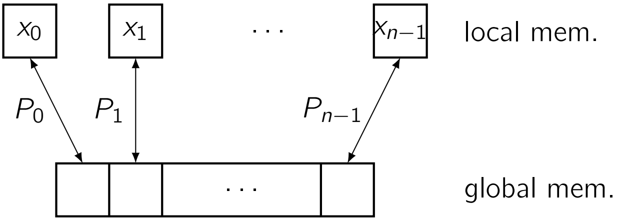
Execution traces

The (well-bracketed) execution traces in $\{u_i, s_i\}^*$ are semantically invariant under the congruence $\approx$ generated by

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

which means that

$$T \approx T' \qquad \text{implies} \qquad [\![T]\!]_\pi = [\![T']\!]_\pi$$



e.g.

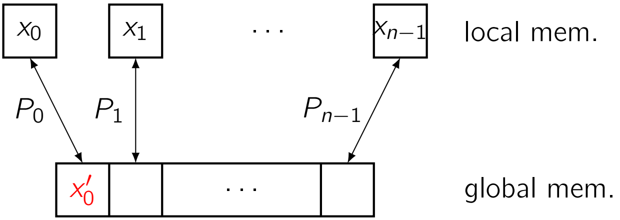
Execution traces

The (well-bracketed) execution traces in $\{u_i, s_i\}^*$ are semantically invariant under the congruence $\approx$ generated by

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

which means that

$$T \approx T' \qquad \text{implies} \qquad [\![T]\!]_\pi = [\![T']\!]_\pi$$



e.g. u_0

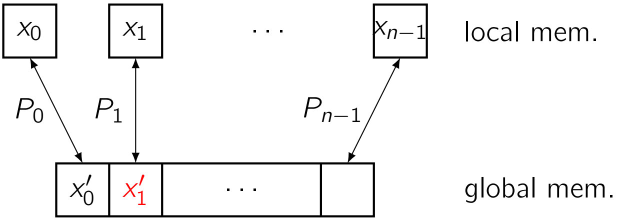
Execution traces

The (well-bracketed) execution traces in $\{u_i, s_i\}^*$ are semantically invariant under the congruence $\approx$ generated by

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

which means that

$$T \approx T' \qquad \text{implies} \qquad [\![T]\!]_\pi = [\![T']\!]_\pi$$



e.g. $u_0 u_1$

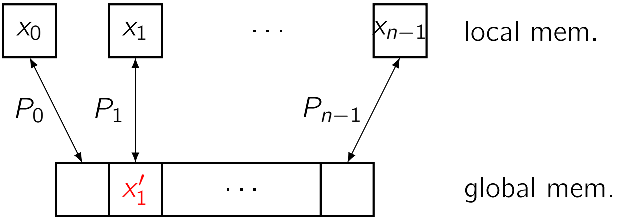
Execution traces

The (well-bracketed) execution traces in $\{u_i, s_i\}^*$ are semantically invariant under the congruence $\approx$ generated by

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

which means that

$$T \approx T' \qquad \text{implies} \qquad [\![T]\!]_\pi = [\![T']\!]_\pi$$



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

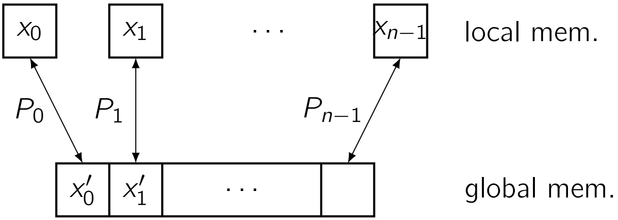
Execution traces

The (well-bracketed) execution traces in $\{u_i, s_i\}^*$ are semantically invariant under the congruence $\approx$ generated by

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

which means that

$$T \approx T' \quad \text{implies} \quad [\![T]\!]_\pi = [\![T']\!]_\pi$$

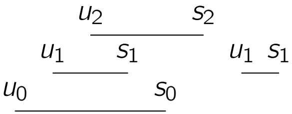


e.g. $u_0 u_1 \approx u_1 u_0$

Interval orders

In a well-bracketed trace, the u_i and s_i form intervals:

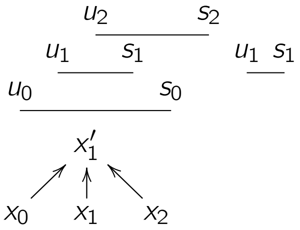
$u_0 u_1 u_2 s_1 s_0 s_2 u_1 s_1 \quad \rightsquigarrow$



Interval orders

In a well-bracketed trace, the u_i and s_i form intervals:

$u_0 u_1 u_2 s_1 s_0 s_2 u_1 s_1 \quad \rightsquigarrow$



$\rightsquigarrow$

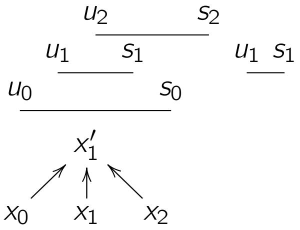
An **interval order** $(X, \preceq)$ is a poset such that there exists a function $I : X \to \wp(\mathbb{R})$ associating an interval I_x to each x in such a way that

$$x \prec y \qquad \text{if and only if} \qquad \forall s \in I_x, \forall t \in I_y,\ s < t$$

Interval orders

In a well-bracketed trace, the u_i and s_i form intervals:

$u_0 u_1 u_2 s_1 s_0 s_2 u_1 s_1$ $\rightsquigarrow$ $\rightsquigarrow$



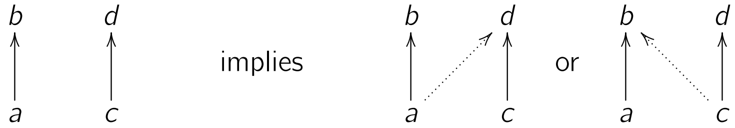
An **interval order** $(X, \preceq)$ is a poset such that there exists a function $I : X \to \wp(\mathbb{R})$ associating an interval I_x to each x in such a way that

$$x \prec y \qquad \text{if and only if} \qquad \forall s \in I_x, \forall t \in I_y,\ s < t$$

There is a *colored variant* with $\ell : X \to \mathbb{N}$ such that $\ell(x) = \ell(y)$ implies that x and y are comparable.

Remark (Fishburn)

A poset is an interval order if it is "$(2+2)$-free":

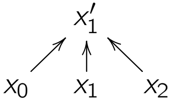


Interval orders

Theorem

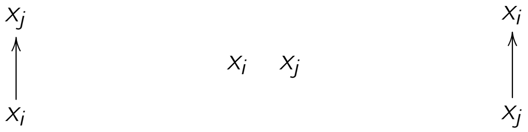
Well-bracketed traces up to equivalence are in bijection with colored interval orders.

$u_0 u_1 u_2 s_1 s_0 s_2 u_1 s_1 \quad \rightsquigarrow$



Views of interval orders

Suppose given two elements x_i and x_j of an interval order. We have the following possible situations:

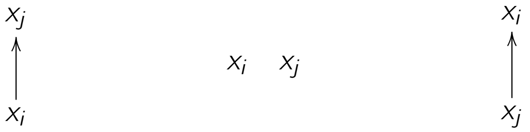


which correspond to the following traces:

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

Views of interval orders

Suppose given two elements x_i and x_j of an interval order. We have the following possible situations:



which correspond to the following traces:

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

In the two first cases, s_j sees u_i.

Views of interval orders

This suggests defining the *i***-view** of a colored interval order $(X, \preceq)$ by

1. restricting to elements which are below or independent from the maximum element x_i^k labeled by i
2. remove dependencies from x_i^k

Views of interval orders

This suggests defining the i-**view** of a colored interval order $(X, \preceq)$ by

1. restricting to elements which are below or independent from the maximum element x_i^k labeled by i
2. remove dependencies from x_i^k

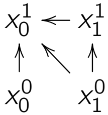
Theorem

- *an interval order can be reconstructed from all the i-views*
- *the execution of the i-th process in the view protocol $\pi^{\lhd}$ is uniquely determined by the i-view*

Views of interval orders

For instance, with two processes, consider $u_0 u_1 s_1 u_1 s_0 s_1 u_0 s_0$:

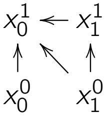
- it corresponds to the colored interval order



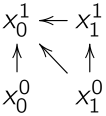
Views of interval orders

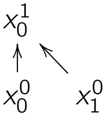
For instance, with two processes, consider $u_0 u_1 s_1 u_1 s_0 s_1 u_0 s_0$:

- it corresponds to the colored interval order



- the views are





Views of interval orders

For instance, with two processes, consider $u_0 u_1 s_1 u_1 s_0 s_1 u_0 s_0$:

0	1
$\bot$	$\bot$

$\xrightarrow{u_0}$

0	1
0	$\bot$

$\xrightarrow{u_1}$

0	1
0	1

$\xrightarrow{s_1}$

0	$\langle 1, 01 \rangle$
0	1

$\xrightarrow{u_1}$

0	$\langle 1, 01 \rangle$
0	$\langle 1, 01 \rangle$

$\xrightarrow{s_0}$

$\langle 0, 0\langle 1, 01 \rangle\rangle$	$\langle 1, 01 \rangle$
0	$\langle 1, 01 \rangle$

$\xrightarrow{s_1}$

$\langle 0, 0\langle 1, 01 \rangle\rangle$	$\langle \langle 1, 01 \rangle, 0\langle 1, 01 \rangle\rangle$
0	$\langle 1, 01 \rangle$

$\xrightarrow{u_0}$

$\langle 0, 0\langle 1, 01 \rangle\rangle$	$\langle \langle 1, 01 \rangle, 0\langle 1, 01 \rangle\rangle$
$\langle 0, 0\langle 1, 01 \rangle\rangle$	$\langle 1, 01 \rangle$

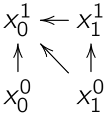
$\xrightarrow{s_0}$

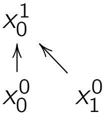
$\langle \langle 0, 0\langle 1, 01 \rangle\rangle, \langle 0, 0\langle 1, 01 \rangle\rangle \langle 1, 01 \rangle\rangle$	$\langle \langle 1, 01 \rangle, 0\langle 1, 01 \rangle\rangle$
$\langle 0, 0\langle 1, 01 \rangle\rangle$	$\langle 1, 01 \rangle$

Views of interval orders

For instance, with two processes, consider $u_0 u_1 s_1 u_1 s_0 s_1 u_0 s_0$:

- we have a correspondence:





$\langle\langle 0, 0\langle 1, 01\rangle\rangle\,, \langle 0, 0\langle 1, 01\rangle\rangle\,\langle 1, 01\rangle\rangle$ $\qquad$ $\langle\langle 1, 01\rangle\,, 0\langle 1, 01\rangle\rangle$

Completeness results

From this we deduce:

Theorem

The equivalence is complete: given two traces t and t'

$$t \approx t' \qquad \textit{iff} \qquad [\![t]\!]_{\pi^{\lhd}} = [\![t']\!]_{\pi^{\lhd}}$$

Theorem

$\pi^{\lhd}$ *is actually initial in the category of protocols.*

The interval order complex

Definition

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

- *vertices* are (i, V_i) where V_i is an i-view
- *maximal simplices* are $\{(0, V_0), \ldots, (n, V_n)\}$ such that there is an interval order $(X, \prec)$ (with given number of rounds) whose i-view is V_i.

Theorem

The interval order complex is isomorphic to the protocol complex.

DIRECTED GEOMETRIC SEMANTICS

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.

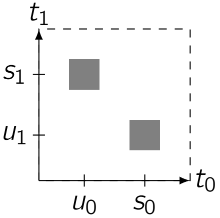
We actually need to use spaces equipped with a notion of direction in order to take in account irreversible time.

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



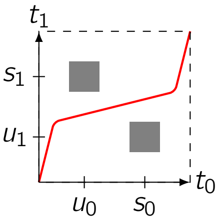
i.e. a square $[0, 1] \times [0, 1]$ minus two holes, which is directed componentwise.

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.

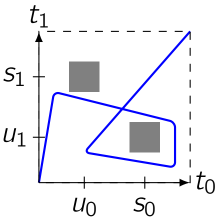
directed path : $u_1 u_0 s_0 s_1$

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.

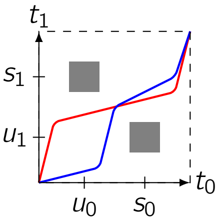
non directed path : ???

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.

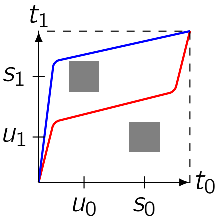
homotopy between paths : $u_1 u_0 s_0 s_1 \approx u_0 u_1 s_0 s_1$

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.

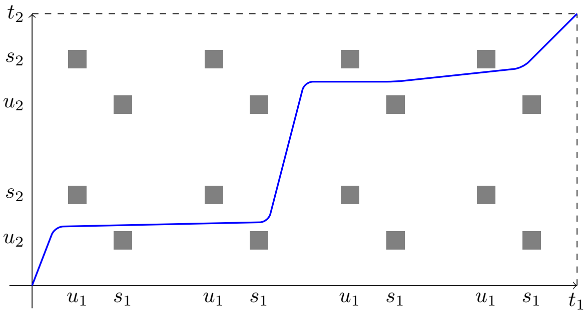
some paths are not homotopic

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

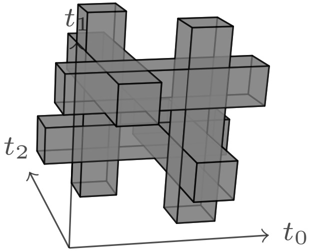


More examples

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,

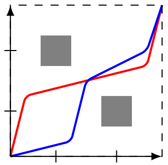
Definition

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
- $s \mapsto H(s, 0)$ and $s \mapsto H(s, 1)$ are constant



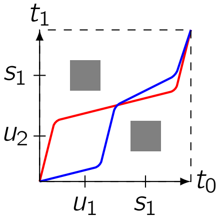
Directed paths vs traces

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* $\approx$



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

Directed paths vs traces

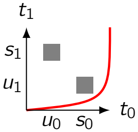
Theorem

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

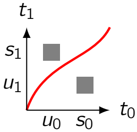
(i) *directed paths up to directed homotopy in the geometric semantics*

(ii) *colored interval orders*

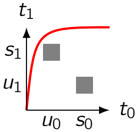
(iii) *execution traces up to* $\approx$



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



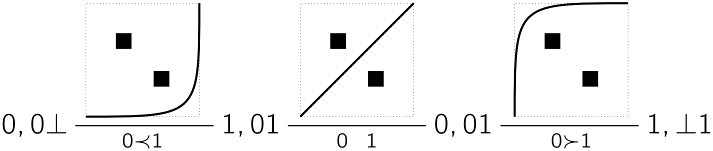
$[u_0, s_0] \parallel [u_1, s_1]$



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

From geometry to the complex

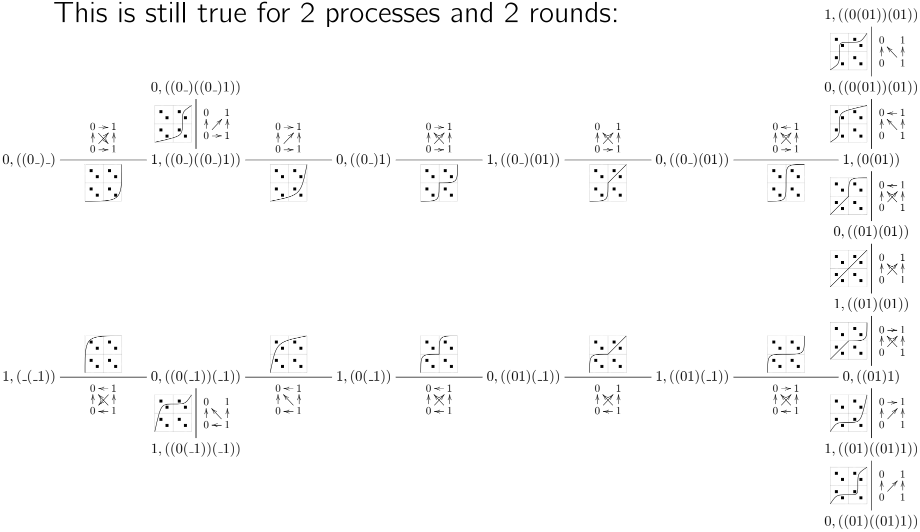
One can notice in the last example that edges are in bijection with directed paths up to homotopy (and with interval orders):



(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:



Thanks!