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, ...]

\[ \langle u_i, s_i \mid u_i u_j = u_j u_i, s_i s_j = s_j s_i \rangle \]

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

**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

![Diagram showing $n$ processes $P_0, P_1, \ldots, P_{n-1}$ connected to a global memory and local memories.]

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

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

![Diagram of processes and memories]

- 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 \(\bot\) 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

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

such that there is an execution leading to a local memory \( I \).

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
- $\perp \in \mathcal{I} \cap \mathcal{O}$: 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 $\pi$ consists of, for $0 \leq i < n$,

- $\pi_{ui} : \mathcal{V} \rightarrow \mathcal{V}$
  the values it will write in its global memory cell depending on its local memory

- $\pi_{si} : \mathcal{V} \times \mathcal{V}^n \rightarrow \mathcal{V}$
  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

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

The set of possible actions is

\[ \mathcal{A} = \{ u_i, s_i, d_i \mid 0 \leq i < n \} \]
Execution traces

The set of possible **actions** is

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

The monoid \( \mathcal{A}^* \) acts on states \( \mathcal{V}^n \times \mathcal{V}^n \) as follows.
Execution traces

The set of possible actions is

\[ A = \{ u_i, s_i, d_i \mid 0 \leq i < n \} \]

An execution trace is a word in \( A^* \) which is well-bracketed:

\[ \text{proj}_i(T) \in (u_is_i)^*(\epsilon + u_id_i) \]
Execution traces

The set of possible actions is

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

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

\[ \text{proj}_i(T) \in (u_is_i)^*(\varepsilon + u_id_i) \]

Given a protocol \( \pi \), 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

\[ A = \{ u_i, s_i \mid 0 \leq i < n \} \]

An execution trace is a word in \( A^\ast \) which is well-bracketed:

\[ \text{proj}_i(T) \in (u_i s_i)^\ast (\epsilon + u_id_i) \]

Given a protocol \( \pi \), its semantics

\[ [T]_{\pi} : \mathcal{V}^n \times \mathcal{V}^n \rightarrow \mathcal{V}^n \times \mathcal{V}^n \]

is defined on a trace \( T \in A^\ast \) by

- \( [u_i]_{\pi}(l, m) = (l, m[i \leftarrow \pi_u(l_i)]) \)
- \( [s_i]_{\pi}(l, m) = (l[i \leftarrow \pi_s(l_i, m)], m) \)
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:

- \(u_0s_0u_1s_1\):
  - \(P_0\) does not see what \(P_1\) has written
  - \(P_1\) sees what \(P_0\) has written

- \(u_1s_1u_0s_0\):
  - \(P_0\) sees what \(P_1\) has written
  - \(P_1\) does not see what \(P_0\) has written

- \(u_0u_1s_0s_1 / u_0u_1s_1s_0 / u_1u_0s_0s_1 / u_1u_0s_1s_0\):
  - \(P_0\) sees what \(P_1\) has written
  - \(P_1\) sees what \(P_0\) has written
Execution traces

These execution traces can be represented geometrically by

\[ \begin{align*}
&u_0 s_0 u_1 s_1 \\
&u_0 u_1 s_0 s_1 \\
&u_1 s_1 u_0 s_0 
\end{align*} \]

We’ll get back to this representation later on.
Execution traces

These execution traces can be represented geometrically by

\[
\begin{array}{c}
0, 0 \\
\downarrow u_0 s_0 u_1 s_1 \\
1, 01
\end{array}
\quad
\begin{array}{c}
1, 01 \\
\downarrow u_0 u_1 s_0 s_1 \\
0, 01
\end{array}
\quad
\begin{array}{c}
1, \bot 1 \\
\downarrow u_1 s_1 u_0 s_0 \\
0, 01
\end{array}
\]

We’ll get back to this representation later on.
A **task** $\theta$ is a relation $\theta \subseteq I^n \times O^n$ such that for every $l, l' \in \Theta$

- $l_i = \bot$ if and only if $l'_i = \bot$,
- there exists $l'' \in O^n$ such that $(l, l'') \in \Theta$ and $(l[i \leftarrow \bot], l''[i \leftarrow \bot]) \in \Theta$.

We write $\text{dom} \Theta$ for the possible input values and $\text{codom} \Theta$ 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

\[
\Theta = \{(b \perp, b \perp), (\perp b, \perp b), (bb', bb), (b' b, bb) \mid b, b' \in \{0, 1\}\}
\]
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

\[
\Theta = \{(b\perp, b\perp), (\perp b, \perp b), (bb', bb), (b' b, bb) \mid b, b' \in \{0, 1\}\}
\]
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 \( l \in \text{dom } \Theta \) (the possible input values) then \( l[i \leftarrow \bot] \) also belongs to \( \text{dom } \Theta \)

\( \text{dom } \Theta \) can thus be pictured as a *simplicial complex* called the **input complex**:

```
⊥ 1 ⊥
\( 01 \bot \) \( 012 \) \( \bot 12 \)
0 ⊥ \( 0 \bot 2 \) \( \bot \bot 2 \)
```

i.e. roughly a space made of triangles, tetrahedra, etc.

(and similarly \( \text{codom } \Theta \) gives rise to the **output complex**
The way we draw tasks

Note that

- if \( l \in \text{dom} \Theta \) (the possible input values) then \( l[i \leftarrow \perp] \) also belongs to \( \text{dom} \Theta \)

\( \text{dom} \Theta \) can thus be pictured as a *simplicial complex* called the **input complex**:

```
\[0\perp01\perp012\perp1\perp12\perp\perp2\perp\]
```

i.e. roughly a space made of triangles, tetrahedra, etc.

(and similarly \( \text{codom} \Theta \) gives rise to the **output complex**)

Note also that the vertices are **colored** by \( 0 \leq i < n \): the only active process
A task $\theta$ is a relation $\theta \subseteq I^n \times O^n$ such that for every $l, l' \in \Theta$

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

which means

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

A protocol $\pi$ solves a task $\Theta$ when

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

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

$$(l', m') = [T]_\pi(l, \perp \perp ... \perp)$$

For simplicity, we will suppose that $l_i = i$ initially (standard state) and thus write $[T]_\pi$ instead of $[T]_\pi(l, \perp \perp ... \perp)$.

For instance,

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

Let’s understand why.
Solving tasks

A protocol $\pi$ solves a task $\Theta$ when

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

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

$$(l', m') = [T]_\pi(l, \perp \perp \ldots \perp)$$

For instance,

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

Let’s understand why.
Solving tasks

A protocol $\pi$ **solves** a task $\Theta$ when

$\quad$ for every initial local memory $l \in \text{dom } \Theta$

$\quad$ for every long enough and fair execution trace $T$

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

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

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,

$\quad$ the consensus cannot be solved

$\quad$ the quasi-consensus can be solved

Let’s understand why.
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:

$$u_0 u_1 s_1 u_2 s_0 s_2$$  but not  $$u_0 u_0 s_1 s_0$$
Restricting executions

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

- **well-bracketed:**

  \[ u_0 u_1 s_1 u_2 s_0 s_2 \quad \text{but not} \quad u_0 u_0 s_1 s_0 \]

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

  \[ u_0 s_0 u_1 s_1 u_1 u_0 s_0 s_1 \quad \text{but not} \quad u_0 u_1 s_0 u_0 s_1 s_0 \]

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:

\[ u_0u_1s_1u_2s_0s_2 \quad \text{but not} \quad u_0u_0s_1s_0 \]

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

\[ u_0s_0u_1s_1u_1u_0s_0s_1 \quad \text{but not} \quad u_0u_1s_0u_0s_1s_0 \]

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

▶ immediate snapshot:

\[ u_0u_1s_1s_0u_2s_2 \quad \text{but not} \quad u_0u_1s_0u_2s_1s_2 \]
Full-information protocols

A protocol is **full-information** when

\[ \pi_{u_i} = \text{id}_\mathcal{V} \]

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

$$
\begin{array}{c}
\mathcal{V} \times \mathcal{V}^n \xrightarrow{\pi_{si}} \mathcal{V} \\
\phi_i \times \prod_i \phi_i & \downarrow \phi_i & \downarrow \\
\mathcal{V} \times \mathcal{V}^n \xrightarrow{\pi_{si}} \mathcal{V}
\end{array}
$$

We say that $\pi'$ simulates $\pi$. 
A category of protocols

A **morphism** \( \phi : \pi \rightarrow \pi' \) between protocols consists of functions

\[ \phi_i : \mathcal{V} \rightarrow \mathcal{V} \]

such that

\[ \phi_i(x) = x \text{ for } x \in \mathcal{I} \]

\[ \phi_i(x) \in \mathcal{O} \text{ for } x \in \mathcal{O} \]

\[ \text{and} \]

\[ \mathcal{V} \times \mathcal{V}^n \xrightarrow{\pi_{si}} \mathcal{V} \]

\[ \phi_i \times \prod_i \phi_i \]

\[ \mathcal{V} \times \mathcal{V}^n \xrightarrow{\pi_{si}} \mathcal{V} \]

We say that \( \pi' \) simulates \( \pi \).

Actually, we only require \( \phi_i \) 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 $\pi$: every execution of $\pi$ 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 $\pi^\triangleleft$ is called the **view protocol** and is defined by

- $\pi^\triangleleft_{ui}(x) = x$ for $x \in \mathcal{V}$ (full-information),
- $\pi^\triangleleft_{si}(x, m) = \langle x, \langle m \rangle \rangle$ for $(x, m) \in \mathcal{V} \times \mathcal{V}^n$.

Given a trace $T$, the local memory of the $i$-th process after executing the trace $T$ is called its **view**.
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 $\pi^<\downarrow$ is called the view protocol and is defined by

- $\pi^<\downarrow_{u_i}(x) = x$ for $x \in \mathcal{V}$ (full-information),
- $\pi^<\downarrow_{s_i}(x, m) = \langle x, \langle m \rangle \rangle$ 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

\[
x = l_i \quad \text{with} \quad (l, m) = \llbracket T \rrbracket_{\pi^{\triangleleft}}
\]

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

\[
\phi_i(x) = l'_i \quad \text{with} \quad (l', m') = \llbracket T \rrbracket_{\pi}
\]

It only remains to check that this definition is well-defined, i.e. it does not depend on the chosen trace \( T \ldots \)
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 $\pi^<$
- 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:

\[
0 \quad 1
\]

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:

\[
\begin{array}{c}
0 \\
\downarrow \\
\downarrow \\
1
\end{array}
\]

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

\[
0, 0\perp \quad 1, 01
\]

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

\[ u_0 s_0 u_1 s_1 : \]

\[
\begin{array}{c|c}
0 & 1 \\
\hline
\downarrow & \downarrow
\end{array}
\quad \xrightarrow{u_0} \quad
\begin{array}{c|c}
0 & 1 \\
\hline
0 & \perp
\end{array}
\quad \xrightarrow{s_0} \quad
\begin{array}{c|c}
0, 0\perp & 1 \\
\hline
0 & \perp
\end{array}
\]

\[
\begin{array}{c|c}
0, 0\perp & 1 \\
\hline
\downarrow & \downarrow
\end{array}
\quad \xrightarrow{u_1} \quad
\begin{array}{c|c}
0, 0\perp & 1 \\
\hline
0 & 1
\end{array}
\quad \xrightarrow{s_1} \quad
\begin{array}{c|c}
0, 0\perp & 1, 01 \\
\hline
0 & 1
\end{array}
\]

\[
\begin{array}{c}
0, 0\perp \\
\downarrow \\
\downarrow \\
1
\end{array}
\]

\[
\begin{array}{c|c}
0, 0\perp & 1 \\
\hline
0 & 1
\end{array}
\]
The protocol complex

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

\[
\begin{array}{c}
0 \\
\downarrow \\
\downarrow \\
1 \\
\end{array}
\]

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

\[
\begin{array}{c}
0, 0 \perp \\
\downarrow \\
\downarrow \\
1, 01 \\
\end{array}
\quad
\begin{array}{c}
0, 01 \\
\downarrow \\
\downarrow \\
1, \perp 1 \\
\end{array}
\]

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

\( u_1 s_1 u_0 s_0 \):

\[
\begin{array}{c|c}
0 & 1 \\
\hline
\perp & \perp \\
\end{array}
\quad
\begin{array}{c|c}
0 & 1 \\
\hline
\perp & 1 \\
\end{array}
\quad
\begin{array}{c|c}
0 & 1, \perp 1 \\
\hline
\perp & 1 \\
\end{array}
\]

\[
\begin{array}{c|c}
0, 1, \perp 1 \\
\hline
0 & 1 \\
\end{array}
\quad
\begin{array}{c|c}
0, 01 & 1, \perp 1 \\
\hline
0 & 1 \\
\end{array}
\]
The protocol complex

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

\[
\begin{array}{c|c}
0 & 1 \\
\hline
\perp & \perp \\
\end{array}
\]

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

\[
0, 0\perp \quad 1, 01 \quad 0, 01 \quad 1, \perp 1
\]

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

\( u_1 u_1 s_0 s_1 \):

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
0 & 1 \\
\hline
\perp & \perp & u_0 \\
0 & \perp & \\
\hline
\hline
0, 01 & 1 & s_0 \\
0 & 1 & \\
\hline
\hline
0, 01 & 1, 01 & s_1 \\
0 & 1 & \\
\end{array}
\]
The protocol complex

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

\[ 0 \longrightarrow ^1 \rightarrow 1 \]

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

\[ 0, 01 \longrightarrow ^1 \rightarrow 1, 01 \]

\[ 1, \bot1 \longrightarrow ^1 \rightarrow 0, 01 \]
The protocol complex

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

\[
\begin{array}{c}
0 \\
1
\end{array}
\]

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

\[
\begin{array}{cccccc}
0, (0\bot)1 & \rightarrow & 1, (0\bot)(01) & \rightarrow & 0, (0\bot)(01) & \rightarrow & 1, 0(01) \\
& & & & & & 0, (01)(01) \\
& & & & & 1, (01)(01) \\
& & 1, 0(\bot1) & \rightarrow & 0, 0(\bot1) & \rightarrow & 1, (01)(\bot1) & \rightarrow & 0, (01)1
\end{array}
\]
The protocol complex

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

\[ \begin{array}{c}
0 \\
\end{array} \quad \begin{array}{c}
1 \\
\end{array} \]

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 $\Theta$ can be solved by a protocol $\pi$:

- it can be solved in $r$ rounds
Solvability

Suppose that a task $\Theta$ can be solved by a protocol $\pi$:

- it can be solved in $r$ rounds
- there is a map $\phi : \pi^< \to \pi$ such that, for every trace $T$,
  \[
  \phi([T]_{\pi^<}) = [T]_{\pi}
  \]
Solvability

Suppose that a task $\Theta$ can be solved by a protocol $\pi$:

- it can be solved in $r$ rounds
- there is a map $\phi : \pi^< \to \pi$ such that, for every trace $T$,
  \[
  \phi([T]_{\pi^<}) = [T]_{\pi}
  \]
- in particular, when the trace $T$ has $r$ rounds $[T] \in \mathcal{O}$
Solvability

Suppose that a task $\Theta$ can be solved by a protocol $\pi$:

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

\[
\phi([T]^<) = [T]^\pi
\]

- in particular, when the trace $T$ has $r$ rounds $[T] \in \mathcal{O}$
- [...] 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 $\text{codom } \Theta$ (and, in fact, conversely).
Solvability

Suppose that a task \( \Theta \) can be solved by a protocol \( \pi \):

- it can be solved in \( r \) rounds
- there is a map \( \phi : \pi^< \rightarrow \pi \) such that, for every trace \( T \),

\[
\phi([T]_{\pi^<}) = [T]_{\pi}
\]

- in particular, when the trace \( T \) has \( r \) rounds \([T] \in \mathcal{O}\)
- [...] 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 \( \text{codom} \Theta \) (and, in fact, conversely).*

NB: simplicial maps preserve contractibility!
Consider again the **binary consensus** task:

There can be no protocol solving it (even after some rounds).
Consider the **binary quasi-consensus**:

The binary quasi-consensus:
Consider the **binary quasi-consensus**: 

\[
\begin{align*}
0, 0 \downarrow & \rightarrow 1, 00 \rightarrow 0, 00 \rightarrow 1, \bot 0 \\
1, 01 & \rightarrow 0, 10 \\
0, 01 & \rightarrow 1, 10 \\
1, \bot 1 & \rightarrow 0, 11 \rightarrow 1, 11 \rightarrow 0, 1\downarrow \\
1, 1 & \rightarrow 0, 1
\end{align*}
\]
CONTRACTIBILITY OF THE PROTOCOL COMPLEX
Simplicial complex

Definition

A simplicial complex $K$ consists of

- a set $K$ of vertices,
- a set $K$ of finite subsets of $K$ called simplices,

such that

- $K$ is non-empty,
- for every $x \in K$, 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 $K$ of vertices,
- a set $K$ of finite subsets of $K$ called simplices,
such that
- $K$ is non-empty,
- for every $x \in K$, we have $\{x\} \in K$,
- for every $\sigma \in K$ and $\tau \subseteq \sigma$ we have $\tau \in K$.

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

\[
\begin{align*}
\Delta^2 & = \\
0 & \quad 01 \quad 02 \quad 2 \\
& \quad 1 \quad 012 \quad 12
\end{align*}
\]
Simplicial complex

Definition
A simplicial complex $K$ consists of

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

such that

- $K$ is non-empty,
- for every $x \in K$, 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 $f : K \to K'$ which

- preserves simplices: for $\sigma \in K$, we have $f(\sigma) \in K'$,
- is locally injective: for $\sigma \in K$, $f$ restricted to $\sigma$ 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.
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
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 $El(K)$ of a simplicial complex has
- the non-empty simplices of $K$ as vertices,
- an edge $\tau \rightarrow \sigma$ whenever $\tau \subsetneq \sigma$. 
Example
For $\Delta^1$

\[
\begin{array}{c}
0 \quad \longrightarrow \quad 01 \quad \longrightarrow \quad 1
\end{array}
\]

the graph of elements is

\[
\begin{array}{c}
0 \quad \longrightarrow \quad 01 \quad \longleftrightarrow \quad 1
\end{array}
\]
The graph of elements

Example
For $\Delta^2$

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 $x_i \rightarrow x_j$

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 $x_i \rightarrow x_j$ for every $i < j$.

**Example**

The nerve of the graph

```
0 → 01 ← 1
```

is

```
0 —— 01 —— 1
```
The barycentric subdivision

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

\[ \chi = N \circ E_! \]
The barycentric subdivision

Example
For $\Delta^1$

```
0 -----01----- 1
```

the barycentric subdivision is

```
0 ----- 01 ----- 1
```
The barycentric subdivision

Example
For $\Delta^2$

The barycentric subdivision is
Colored complexes

Definition
The category of colored simplicial complexes is

$$\text{SC/} \! \mathbb{N}$$

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:

$$\ell \ : \ K \ \rightarrow \ \mathbb{N}$$

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

We write $\coloredGraph N$ for the graph with

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

We write $\mathcal{G}$ 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

$$\text{Graph} / \mathcal{G}$$

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

\[ \text{El} : \text{SC/} ! \mathbb{N} \to \text{Graph/} ! \mathbb{N} \]

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

- vertices are \((\sigma, i)\) 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 $\Delta^1$

the chromatic graph of elements is

```
0   01   1
   ↗   ↗   ↗
0,0 01,1 01,0 1,1
```
The chromatic graph of elements

Example
For $\Delta^2$

the chromatic graph of elements is
The chromatic nerve

Definition

The functor

\[ N : \text{Graph} / !N \to \text{SC} / !N \]

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

- the elements of \(G\) as vertices, colored by \(\ell\),
- 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

\[ \chi = N \circ E_l \]
The standard chromatic subdivision

Example

For $\Delta^1$

\[
0 \quad 01 \quad 1
\]

the standard chromatic subdivision is

\[
0, 0 \quad 01, 1 \quad 01, 0 \quad 1, 1
\]
The standard chromatic subdivision

Example
For $\Delta^2$

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 $\tau$ is a **free face** of a simplex $\sigma$ when

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

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

**Definition**
A simplex $\tau$ is a **free face** of a simplex $\sigma$ when

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

In this case, the monomorphism

$$K \leftarrow K \setminus \tau$$

is called a **collapse step**.
Collapsibility

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

**Definition**
A simplex $\tau$ is a **free face** of a simplex $\sigma$ when

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

In this case, the monomorphism

$$K \leftrightarrow K \setminus \tau$$

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 $\tau$ is a **free face** of a simplex $\sigma$ when

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

In this case, the monomorphism

$$K \hookrightarrow K \setminus \tau$$

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

The simplex $\Delta^2$ is collapsible:
Collapsibility

Example
The simplex $\Delta^2$ is collapsible:

```
1
/|
| |
1--2
| |
0--2
| |
0--1
```

$\Delta^2$ is collapsible because it can be reduced to a single point by collapsing the edges in a specific sequence.
Collapsibility

Theorem (Whitehead)

A *collapsible simplicial complex* is contractible.
Collapsibility

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**

  \[ K \star L = K \cup L \]

- **simplices**

  \[ K \star L = \{ \sigma \subseteq K \cup L | \sigma \cap K \in K \text{ and } \sigma \cap L \in L \} \]

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

Example
\[ \Delta^m \star \Delta^n = \Delta^{m+n+1}. \]
Definition
Given colored simplicial complexes $K$ and $L$, their **colored join** $K \star L$ is the complex with

- vertices

  $$K \star L = K \cup L$$

- simplices

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

Definition
The \textbf{basic chromatic subdivision} of $\Delta^n$ is

$$\partial \Delta^n \star \Delta^n$$
The basic chromatic subdivision

Definition
The **basic chromatic subdivision** of $\Delta^n$ is

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

Example
For $\Delta^2$, we have:

![Diagram](image-url)
The basic chromatic subdivision

**Definition**

The **basic chromatic subdivision** of $\Delta^n$ 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

\[ \Delta' \hookrightarrow \partial \Delta' \star \Delta' = K' \]

\[ \sigma \mapsto \emptyset | \sigma \]

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 $\Delta^2$ 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

\[
\begin{array}{ccc}
\Lambda_i^n & \rightarrow & K \\
\downarrow & & \downarrow \\
\Delta^n & \rightarrow & L
\end{array}
\]

- the image of $\chi$ is characterized by its action on representables

\[
\chi(K) = \colim(El(K) \xrightarrow{\pi} \Delta \xrightarrow{\gamma} \hat{\Delta} \xrightarrow{\chi} \hat{\Delta})
\]
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

\[
\begin{array}{ccc}
\Lambda_i^n & \longrightarrow & K \\
\downarrow & & \downarrow \\
\Delta^n & \longrightarrow & L
\end{array}
\]

- the image of $\chi$ is characterized by its action on representables

\[\chi(K) = \text{colim}(\text{El}(K)) \xrightarrow{\pi} \Delta \xrightarrow{\gamma} \hat{\Delta} \xrightarrow{\chi} \hat{\Delta})\]

**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

\[
u_j u_i \approx u_i u_j \quad \text{and} \quad s_j s_i \approx s_i s_j
\]

which means that

\[
T \approx T' \quad \text{implies} \quad [T]_\pi = [T']_\pi
\]
Execution traces

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

\[
\begin{align*}
  u_j u_i & \approx u_i u_j \\
  s_j s_i & \approx s_i s_j
\end{align*}
\]

which means that

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

e.g.
Execution traces

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

$$u_j u_i \approx u_i u_j \quad s_j s_i \approx s_i s_j$$

which means that

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

e.g. $u_0$
Execution traces

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

\[
u_j u_i \approx u_i u_j \quad \quad s_j s_i \approx s_i s_j
\]

which means that

\[
T \approx T' \quad \quad \text{implies} \quad \quad \langle T \rangle_\pi = \langle T' \rangle_\pi
\]
Execution traces

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

\[
    u_j u_i \approx u_i u_j \quad \quad \quad s_j s_i \approx s_i s_j
\]

which means that

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

![Diagram](image)

e.g. \( u_0 u_1 \approx u_1 \)

local mem.

global mem.
Execution traces

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

\[
\begin{align*}
u_j u_i & \approx u_i u_j & s_j s_i & \approx s_i s_j
\end{align*}
\]

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:

$u_0 u_1 u_2 s_1 s_0 s_2 u_1 s_1 \leadsto u_2 s_2 u_1 s_1 u_1 s_1 u_0 s_0$
Interval orders

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

$$u_0u_1u_2s_1s_0s_2u_1s_1 \implies u_0 \overline{s_0} \quad u_1 \overline{s_1} \quad u_2 \overline{s_2}$$

$u_0u_1u_2s_1s_0s_2u_1s_1 \implies x_0 \xymatrix{\ar[r] & x_1' \ar[r] & x_1 \ar[r] & x_2}$

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

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

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

\[
\begin{align*}
   u_0 u_1 u_2 s_1 s_0 s_2 u_1 s_1 & \leadsto & u_2 s_2 \\
   u_0 u_1 s_1 & \leadsto & u_1 s_1 \\
   & \leadsto & u_0 s_0 \\
   & \leadsto & x_1' \\
   & \leadsto & x_0 \rightarrow x_1 \rightarrow x_2
\end{align*}
\]

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

\[
x \prec y \text{ if and only if } \forall s \in l_x, \forall t \in l_y, \ s < t
\]

There is a **colored variant** with $\ell : X \rightarrow \mathbb{N}$ 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:

\[
\begin{array}{cc}
  b & d \\
  \uparrow & \uparrow \\
  a & c \\
\end{array}
\]

implies

\[
\begin{array}{cc}
  b & d \\
  \uparrow & \uparrow \\
  a & c \\
\end{array}
\]

or

\[
\begin{array}{cc}
  b & d \\
  \uparrow & \uparrow \\
  a & c \\
\end{array}
\]

or
Theorem

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

\[ u_0 u_1 u_2 s_1 s_0 s_2 u_1 s_1 \sim \rightarrow \]

\[ x_0 \quad x_1' \quad x_2 \]

\[ x_0 \quad x_1 \quad x_2 \]
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:

$u_is_iu_js_j$  $u_is_is_js_j$  $u_js_js_i$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:

$$u_i s_i u_j s_j \quad u_i u_j s_i s_j \quad u_j s_j u_i s_i$$

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 \(x_i^k\)

\[\text{Theorem} \quad \text{an interval order can be reconstructed from all the } i\text{-views} \]

\[\text{the execution of the } i\text{-th process in the view protocol } \pi \]

\[\text{is uniquely determined by the } i\text{-view} \]
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 $x^k_i$ labeled by $i$
2. remove dependencies from $x^k_i$

**Theorem**

- an interval order can be reconstructed from all the $i$-views
- the execution of the $i$-th process in the view protocol $\pi^{<}$ 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

\[
\begin{array}{c}
\uparrow \\
\leftrightarrow \\
\uparrow \\
\end{array}
\]
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

\[
x_0 \leftarrow x_1^1 \leftarrow x_1^1 \leftarrow x_0^0 \leftarrow x_0^0 \leftarrow x_0^1 \leftarrow x_1^0 \leftarrow x_1^0
\]

- The views are

\[
x_0 \leftarrow x_1^1 \leftarrow x_1^1 \leftarrow x_0^0 \leftarrow x_0^0 \leftarrow x_0^1 \leftarrow x_1^0 \leftarrow x_1^0\]
Views of interval orders

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

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

From this we deduce:

**Theorem**

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

\[ t \approx t' \quad \text{iff} \quad [t]_{\pi^<} = [t']_{\pi^<} \]

**Theorem**

$\pi^<$ 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.

\[ u_0.s_0 \parallel u_1.s_1 \]

The geometric semantics of this program will be

i.e. a square \([0, 1] \times [0, 1]\) minus two holes, which is directed componentwise.
Consider two processes executing one round of update/scan, i.e.

\[ u_0 . s_0 \parallel u_1 . s_1 \]

The geometric semantics of this program will be

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

\[ \text{directed path} : u_1 u_0 s_0 s_1 \]
An example

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

\[ u_0 \cdot s_0 \parallel u_1 \cdot s_1 \]

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.

\[ u_0.s_0 \parallel u_1.s_1 \]

The geometric semantics of this program will be

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

\[
\text{homotopy between paths} \quad : \quad u_1u_0s_0s_1 \approx u_0u_1s_0s_1
\]
Consider two processes executing one round of update.scan, i.e. 

\[ u_0.s_0 \parallel u_1.s_1 \]

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,

\[
\begin{align*}
    u_0.s_0.u_0.s_0 & \parallel u_1.s_1.u_1.s_1.u_1.s_1.u_1.s_1
\end{align*}
\]

The geometric semantics of this program will be

[Diagram of two timelines with overlapping updates and stacks]

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

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

\[
\begin{align*}
&u_0.s_0 \parallel u_1.s_1 \parallel u_2.s_2
\end{align*}
\]

The geometric semantics of this program will be

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

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
Direct 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 ≈

$$u_1u_0s_0s_1 \approx u_0u_1s_0s_1$$
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* $\approx$

\[
[u_0, s_0] \prec [u_1, s_1] \quad [u_0, s_0] \parallel [u_1, s_1] \quad [u_0, s_0] \succ [u_1, s_1]
\]
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!