Summary

1. The language
2. Abstract machine
3. Higher dimensional control flow structure
4. Providing models with local pospace structure
5. Handling continuous models
6. Factoring
7. Directed topology
8. Perspectives
Summary

1. The language
2. Abstract machine
3. Higher dimensional control flow structure
4. Providing models with local pospace structure
5. Handling continuous models
6. Factoring
7. Directed topology
8. Perspectives
Summary

1. The language
2. Abstract machine
3. Higher dimensional control flow structure
4. Providing models with local pospace structure
5. Handling continuous models
6. Factoring
7. Directed topology
8. Perspectives
Summary

1. The language
2. Abstract machine
3. Higher dimensional control flow structure
4. Providing models with local pospace structure
5. Handling continuous models
6. Factoring
7. Directed topology
8. Perspectives
Summary

1. The language
2. Abstract machine
3. Higher dimensional control flow structure
4. Providing models with local pospace structure
5. Handling continuous models
6. Factoring
7. Directed topology
8. Perspectives
Summary

1. The language
2. Abstract machine
3. Higher dimensional control flow structure
4. Providing models with local pospace structure
5. Handling continuous models
6. Factoring
7. Directed topology
8. Perspectives
Summary

1. The language
2. Abstract machine
3. Higher dimensional control flow structure
4. Providing models with local pospace structure
5. Handling continuous models
6. Factoring
7. Directed topology
8. Perspectives
Summary

1. The language
2. Abstract machine
3. Higher dimensional control flow structure
4. Providing models with local pospace structure
5. Handling continuous models
6. Factoring
7. Directed topology
8. Perspectives
Summary

1. The language
2. Abstract machine
3. Higher dimensional control flow structure
4. Providing models with local pospace structure
5. Handling continuous models
6. Factoring
7. Directed topology
8. Perspectives
1. The Language
Targeted software and strategy

- Fine-grained parallel programs (e.g. asynchronous control command systems).
Targeted software and strategy

Program analysis
Targeted software and strategy

Control flow analysis
Targeted software and strategy

- Control flow analysis
- Value analysis
Targeted software and strategy

- Control flow analysis
- Coordination analysis
- Value analysis
Features of the language

Features of the language


- shared memory abstract machine (PRAM)
  concurrent read exclusive write (CREW)
- no pointer arithmetics
- no function
- no birth nor death of process at runtime
- tokens are *owned* by processes
- *conservative* processes
Standard examples

sem: \( 1 \ a \)
proc:
  \( p = P(a); V(a) \)
init: \( 2p \)

var: \( x = 0 \)
proc:
  \( p1 = x:=1 \),
  \( p2 = x:=2 \)
init: \( p1 \ p2 \)
Middle-end representation

- $\mathcal{P} = \{\text{process identifiers}\}$,
  $\mathcal{V} = \{\text{variables}\}$,
  $\mathcal{S} = \{\text{semaphores}\}$,
  $\mathcal{B} = \{\text{barriers}\}$
- $\text{init} : \mathcal{V} \rightarrow \mathbb{R}$
- $\text{arity} : \mathcal{S} \sqcup \mathcal{B} \rightarrow \mathbb{N} \cup \{\infty\}$
- $\text{proc} : \mathcal{P} \rightarrow \{\text{control flow graphs}\}$
Control flow graphs
Floyd, R. W., Assigning meanings to programs, 1967
Allen, F. E., Control flow analysis, 1970

var: x = 7
proc:
p = J(q)+[x<>1]+() ,
q = (x:=x/2; J(p))+[x % 2 = 0]+
    (x:=3*x+1; J(p))
init: p
An Execution Trace

on a control flow graph

entry point

\[ x := \frac{x}{2} \]

\[ x := 3x + 1 \]

\[ x \equiv 2 = 0 \]

\[ x = 1 \]

the current value of \( x \) is 7
An Execution Trace

on a control flow graph

the current value of $x$ is 7
An Execution Trace

on a control flow graph

the current value of \( x \) is 7
An Execution Trace

on a control flow graph

x := x / 2

x := 3 * x + 1

x % 2 = 0

the current value of x is 22
An Execution Trace

on a control flow graph

the current value of $x$ is 22
An Execution Trace

on a control flow graph

the current value of $x$ is 22
An Execution Trace

on a control flow graph

the current value of $x$ is 22
An Execution Trace
on a control flow graph

the current value of \( x \) is 11
An Execution Trace

on a control flow graph

x := x / 2
x := 3 * x + 1
x % 2 = 0
x = 1

the current value of x is 11
An Execution Trace

on a control flow graph

x := x/2
x := 3*x + 1
x % 2 = 0
x := 1

the current value of x is 7
the current value of x is 11

the current value of x is 11
An Execution Trace

on a control flow graph

the current value of x is 11
An Execution Trace
on a control flow graph

entry point

x := x / 2

x := 3 * x + 1

x \equiv 2 = 0

the current value of x is 34
An Execution Trace

on a control flow graph

the current value of $x$ is 34
An Execution Trace
on a control flow graph

the current value of $x$ is 34
An Execution Trace
on a control flow graph

the current value of $x$ is 34
An Execution Trace

on a control flow graph

the current value of $x$ is $17$
An Execution Trace

on a control flow graph

the current value of $x$ is 17
An Execution Trace
on a control flow graph

the current value of $x$ is 17
An Execution Trace
on a control flow graph

the current value of x is 17
An Execution Trace

on a control flow graph

the current value of $x$ is 52
An Execution Trace

on a control flow graph

the current value of $x$ is 52
An Execution Trace

on a control flow graph

$x := x/2$

$x := 3 \times x + 1$

$x \mod 2 = 0$

$x = 1$

the current value of $x$ is 7

the current value of $x$ is 52

the current value of $x$ is 52
An Execution Trace
on a control flow graph

the current value of $x$ is 52
An Execution Trace
on a control flow graph

the current value of $x$ is 26
An Execution Trace
on a control flow graph

entry point

the current value of $x$ is 26

the current value of $x$ is 7

the current value of $x$ is 26
An Execution Trace

on a control flow graph

the current value of $x$ is 26
An Execution Trace

on a control flow graph

\[ x := x / 2 \]
\[ x := 3 \times x + 1 \]
\[ x \mod 2 = 0 \]

the current value of \( x \) is 26
An Execution Trace
on a control flow graph

the current value of $x$ is 13
An Execution Trace
on a control flow graph

the current value of \( x \) is 13
An Execution Trace
on a control flow graph

The current value of $x$ is 13
An Execution Trace

on a control flow graph

the current value of $x$ is 13
An Execution Trace

on a control flow graph

the current value of $x$ is 40
An Execution Trace

on a control flow graph

entry point

the current value of x is 40
An Execution Trace

on a control flow graph

the current value of $x$ is 40
An Execution Trace

on a control flow graph

entry point

\[ x := x / 2 \]

\[ x := 3 \times x + 1 \]

\[ x \mod 2 = 0 \]

the current value of \( x \) is 40

the current value of \( x \) is 7

the current value of \( x \) is 40
An Execution Trace

on a control flow graph

x := x/2
x := 3*x + 1
x % 2 = 0
x = 1

the current value of x is 20
An Execution Trace
on a control flow graph

Entry point $x := 1$

$x := x/2$

$x \% 2 = 0$

$x := 3x + 1$

The current value of $x$ is 20
An Execution Trace
on a control flow graph

The current value of $x$ is 20
An Execution Trace
on a control flow graph

x := x/2

x := 3*x + 1

x mod 2 = 0

the current value of \( x \) is 20
An Execution Trace
on a control flow graph

The current value of $x$ is 10
An Execution Trace

on a control flow graph

the current value of $x$ is 10
An Execution Trace
on a control flow graph

the current value of $x$ is 10
An Execution Trace
on a control flow graph

the current value of \( x \) is 10
An Execution Trace

on a control flow graph

the current value of $x$ is 5
An Execution Trace
on a control flow graph

the current value of $x$ is 5
An Execution Trace

on a control flow graph

the current value of $x$ is 5
An Execution Trace on a control flow graph

the current value of $x$ is 5
An Execution Trace
on a control flow graph

the current value of $x$ is 16
An Execution Trace
on a control flow graph

x := x/2
x := 3*x + 1

x ≡ 0

the current value of x is 16
An Execution Trace

on a control flow graph

the current value of $x$ is 16
An Execution Trace

on a control flow graph

the current value of $x$ is 16
An Execution Trace

on a control flow graph

entry point

x := x/2

x := 3*x + 1

x%2 = 0

the current value of x is 8

the current value of x is 7

the current value of x is 8
An Execution Trace
on a control flow graph

the current value of $x$ is 8
An Execution Trace

on a control flow graph

x := \( \frac{x}{2} \)

x := \( 3x + 1 \)

x \equiv 0 \pmod{2}

x = 1

the current value of x is 8
An Execution Trace

on a control flow graph

the current value of $x$ is 8
An Execution Trace

on a control flow graph

the current value of $x$ is 4
An Execution Trace
on a control flow graph

the current value of $x$ is 4
An Execution Trace

on a control flow graph

the current value of $x$ is 4
An Execution Trace

on a control flow graph

The current value of $x$ is 4
An Execution Trace

on a control flow graph

the current value of $x$ is 2
An Execution Trace

on a control flow graph

the current value of $x$ is 2
An Execution Trace
on a control flow graph

the current value of \( x \) is 2
An Execution Trace

on a control flow graph

x := x/2
x := 3*x + 1
x \equiv 2 = 0
x = 1

the current value of x is 2
An Execution Trace on a control flow graph

the current value of $x$ is 1
An Execution Trace
on a control flow graph

the current value of $x$ is 1
An Execution Trace
on a control flow graph

x := x/2
x := 3*x + 1
x%2 = 0
x = 1

the current value of x is 1
An Execution Trace

on a control flow graph

the current value of $x$ is 1
An Execution Trace
on a control flow graph

the current value of $x$ is 1
2. Abstract Machine
State (Assume that $\mathcal{P} = \{1, \ldots, N\}$)

**point:** a tuple $(p_1, \ldots, p_N)$ s.t. each $p_n$ is either a vertex or an arrow of the $n^{th}$ control flow graph.
State (Assume that $\mathcal{P} = \{1, \ldots, N\}$)

**point:** a tuple $(p_1, \ldots, p_N)$ s.t. each $p_n$ is either a vertex or an arrow of the $n^{th}$ control flow graph.

**context:** mapping $\sigma$ defined over $\mathcal{V} \cup \mathcal{S}$ s.t.
- for all $v \in \mathcal{V}$, $\sigma(v) \in \mathbb{R}$, and
- for all $s \in \mathcal{S}$, $\sigma(s)$ is a multiset over $\mathcal{P}$.

We denote by $\sigma_0$ the initial context of a program.

**state:** a point and a context.
Multi-instruction
Assume $\mathcal{P} = \{1, \ldots, N\}$

**multi-instruction**: a partial map $\mu$ from $\mathcal{P}$ to \{single instructions\}

Multi-instruction
Assume $\mathcal{P} = \{1, \ldots, N\}$

**multi-instruction**: a partial map $\mu$ from $\mathcal{P}$ to \{single instructions\}

$\mu$ **admissible** in the context $\sigma$:
- $\mu(i)$ and $\mu(j)$ do not conflict
- for all $s \in S$, $|\sigma(s)| + \text{card}\{i \in M \mid \mu(i) = \mathcal{P}(s)\} \leq \text{arity}(s)$
- for all $b \in \mathcal{B}$, $\text{card}\{i \in M \mid \mu(i) = \mathcal{W}(b)\} \notin \{1, \ldots, \text{arity}(b)\}$

The context $\sigma \cdot \mu$ is the result of the execution of $\mu$ in the context $\sigma$. 
**Paths**

**path:** a sequence of points \( p(0), \ldots, p(K) \) s.t. \( \forall k \in \{1, \ldots, K\} \) one has

**execution:** \( \forall n \in D_k \ \partial^+ p_n(k - 1) = p_n(k) \)

or

**branching:** \( \forall n \in D_k \ p_n(k - 1) = \partial^- p_n(k) \)

where \( D_k = \{ n \in \{1, \ldots, N\} \mid p_n(k - 1) \neq p_n(k) \} \)
Each path is associated with a sequence of multi-instructions \( (\mu_k) \) where 
\[ \mu_k(n) = \lambda_n(p_n(k)) \]
for all \( n \) such that 
\[ p_n(k) = \partial^+ p_n(k - 1) \text{ or } \lambda_n(p_n(k)) = W(\_). \]

The path is said to be **admissible** when \( \mu_{k+1} \) is admissible in the context \( \sigma_0 \cdot \mu_1 \cdots \mu_k \) for all \( k \).

It is an **execution path** when for all \( k, n : \)
\[ \partial^- p_n(k) = p_n(k - 1) \text{ implies } \llbracket \lambda_n(p_n(k)) \rrbracket_{\sigma \cdot \mu_0 \cdots \mu_{k-1}} \neq 0 \]
3. Higher Dimensional Control Flow Structure

Encoding admissibility in a model
Race condition

write-write conflict

var:  x = 0
proc:  
   p1 = x := 1,
   p2 = x := 2
init:  p1 p2
Exhaustive model

tensor product of graphs

x := 2

\[ x := \frac{18}{75} \]
Exhaustive model

tensor product of graphs

x := 2

T := x

\frac{18}{75}
Exhaustive model
tensor product of graphs
Exhaustive model

tensor product of graphs

$x := 2$

$\otimes$

$\frac{18}{75}$
Exhaustive model

tensor product of graphs

\( x := 2 \)

\( I := x \)
Exhaustive model

tensor product of graphs

$x := 2$

$T := x$
Not admissible path
due to race condition

\[ x := 2 \]

\[ I := \lnot x \]

the value of \( x \) is 0
Not admissible path
due to race condition

the value of $x$ is 0
Not admissible path
due to race condition

the value of x is 0
Not admissible path

due to race condition

\[ x := 2 \]

\[ 1 \rightarrow 2 \]

the value of \( x \) is ?
Admissible path
that however meets a forbidden point

the value of $x$ is 0
Admissible path
that however meets a forbidden point

\[ x := 2 \]

\[ \Gamma = \cdot x \]

the value of \( x \) is 0
Admissible path
that however meets a forbidden point

the value of $x$ is $0$
Admissible path
that however meets a forbidden point

the value of $x$ is 1
Admissible path that however meets a forbidden point

the value of $x$ is 2
Admissible path
that however meets a forbidden point

the value of $x$ is 2
Admissible path

that however meets a forbidden point

the value of \( x \) is \( 2 \)
Admissible path
that however meets a forbidden point

the value of $x$ is 2
Admissible path
avoiding forbidden points

the value of $x$ is 0
Admissible path
avoiding forbidden points

the value of $x$ is 0
Admissible path

avoiding forbidden points

the value of $x$ is 0
Admissible path
avoiding forbidden points

the value of $x$ is 1
Admissible path
avoiding forbidden points

the value of $x$ is 1
Admissible path
avoiding forbidden points

the value of $x$ is 2
Admissible path

avoiding forbidden points

the value of $x$ is 2

$x := 2$

$y := 1$
Admissible path
avoiding forbidden points

the value of $x$ is 2
Admissible path
avoiding forbidden points

the value of $x$ is 2
The replacement property
for admissible paths

Replacement:

Any admissible path that meets a race condition is “equivalent” to an admissible path which avoids all of them.
One token for two processes

sem: 1 a
proc:
   p = P(a); V(a)
init: 2p
Discrete model

sem: 1 a
Discrete model

sem: 1 a
Discrete model

\text{sem: 1 a}
Discrete model

$\text{sem: 1 a}$
Discrete model

sem: 1 a
Discrete model

$\text{sem: } 1 \ a$
Discrete model

sem: 1 a
A process $\pi$ is conservative when for all paths and all semaphores $s$, the amount of tokens of type $s$ held by the process at the end of the execution trace only depends on its arrival point.

In that case the process $\pi$ comes with a potential function $F_{\pi}$

$$F_{\pi} : \{\text{semaphores}\} \times \{\text{points}\} \to \mathbb{N}$$

A program $\Pi$ is conservative when so are its processes $\pi_1, \ldots, \pi_d$ and its potential function is given by

$$F_{\Pi}(s, (p_1, \ldots, p_d)) = \sum_{k=1}^{d} F_{\pi_k}(s, p_k)$$

If $F_{\Pi}(s, p) > \text{arity}(s)$ for some semaphore $s$, then $p$ is forbidden.
Conservative process

example
Conservative process

example
Conservative process

example
Conservative process

example
Conservative process

example
Conservative process

example
Conservative process

example
Conservative process

e.example
Conservative process

example
Conservative process

example
Conservative process

example
Conservative process
example
Conservative process

example
Conservative process

example
Conservative process

example
Conservative process

equation

\[ P(s) \]

\[ V(s) \]
Conservative process

example
Conservative process

example
Not conservative process

eexample
Not conservative process

degenerate example

\[ P(s) \]

Diagram:

- A labeled circle labeled \( P(s) \)
- Arrows pointing clockwise around the circle
- Arrows pointing from the circle to the outside
Not conservative process

example
Not conservative process

example
Not conservative process

example
Not conservative process
example
Not conservative process

e
c

\[ P(s) \]
Not conservative process

e
c

P(s)
Not conservative process

example
Not conservative process

example
Not conservative process

example
Not conservative process

example
Not conservative process

example
Not conservative process

Example
Not conservative process

example
Not conservative process

example
Not conservative process

example
Not conservative process
example
A synchronization barrier

sync: 1 b
proc:
  p = W(b)
init: 2p
Discrete Model

\[ W(b) \]

sync: 1 b
Discrete Model

\[ \text{sync: } 1 \ b \]
Discrete Model

\[ \text{sync: } 1 \ b \]
Discrete Model

\[
\begin{array}{ccccc}
1 & W(b) & & & \\
& & & & \\
0 & 0 & 1 & 0 & 0
\end{array}
\]
Discrete Model

$\text{sync: } 1 \ b$
Discrete Model

sync: 1 b
Discrete Model

sync: 1 b
4. Providing Models with Local Pospace Structure
Locally ordered spaces

Directed atlas $\mathcal{U}$ For all points $p$, for all directed neighborhoods $A$ and $B$ of $p$, there exists a directed neighborhood $C$ of $p$ such that $C \subseteq A \cap B$ and $\preceq_A |_C = \preceq_C = \preceq_B |_C$. 

![Diagram of directed atlas](image-url)
From discrete models to continuous ones

\[ G : A \xrightarrow{\partial^+} V \quad \downarrow G \downarrow = V \sqcup A \times ]0, 1[ \]

\[ \downarrow G_1 \downarrow \times \cdots \times \downarrow G_N \downarrow = \bigsqcup_{\text{points } p \text{ of } G_1, \ldots, G_N} \{p\} \times ]0, 1[^{\dim(p_1, \ldots, p_N)} \]

where \( p = (p_1, \ldots, p_N) \) and \( \dim p = \#\{n \in \{1, \ldots, N\} \mid p_n \in A_n\} \)

The directed topological model is then

\[ \bigsqcup_{\text{not forbidden points } p \text{ of } G_1, \ldots, G_N} \{p\} \times ]0, 1[^{\dim(p_1, \ldots, p_N)} \]
From discrete to continuous

sem: 1 a  sync: 1 b

\[
\begin{align*}
y &:= 0 \\
W(b) & P(a) \\
x &:= y \\
V(a) & \\
z &:= 1 \\
W(b) & P(a) \\
0 & y := 0 \\
\otimes & \times \\
\end{align*}
\]
From discrete to continuous

sem: 1 a  sync: 1 b

\[ y := 0 \]

\[ W(b) \]

\[ P(a) \]

\[ x := z \]

\[ V(a) \]

\[ z := 1 \]

\[ W(b) \]

\[ P(a) \]

\[ z := x \]

\[ V(a) \]

\[ \otimes \]

\[ \times \]
From discrete to continuous

sem: 1 a
sync: 1 b

z:=1

x:=y

P(a)

W(b)

V(a)
From discrete to continuous

\[
\begin{align*}
\text{sem: } & \quad 1 \ a \\
\text{sync: } & \quad 1 \ b
\end{align*}
\]
From discrete to continuous

\( \text{sem: 1 a sync: 1 b} \)
From discrete to continuous

sem: 1 a  sync: 1 b

\[ y := 0 \]
\[ W(b) \]
\[ P(a) \]
\[ x := z \]
\[ V(a) \]
\[ z := 1 \]
\[ W(b) \]
\[ P(a) \]
From discrete to continuous

sem: 1 a  sync: 1 b
From discrete to continuous

sem: 1 a  sync: 1 b
From discrete to continuous

sem: 1 a  sync: 1 b

\[ y := 0 \]

\[ W(b) \]

\[ P(a) \]

\[ x := y \]

\[ V(a) \]

\[ z := 1 \]

\[ W(b) \]

\[ P(a) \]

\[ z := x \]

\[ V(a) \]
Weakly directed homotopy: A homotopy of paths whose intermediate paths are directed.

Strongly directed homotopy: A morphism of local pospace whose underlying map is a homotopy of paths.

The dihomotopy classes of a local pospace $X$ are the morphisms of its fundamental category usually denoted by $\pi_1 X$. 
Adequacy

Theorem
Adequacy

Theorem

1. Each directed path on a continuous model gives rise to an admissible path on the corresponding discrete model. Hence directed paths act on valuations.
1. Each directed path on a continuous model gives rise to an admissible path on the corresponding discrete model. Hence directed paths act on valuations.

2. The output valuations of weakly dihomotopic directed paths are the same. Hence weak dihomotopy classes (i.e. the fundamental category of the model) act on valuations.
Adequacy

Theorem

1. Each directed path on a continuous model gives rise to an admissible path on the corresponding discrete model. Hence directed paths act on valuations.

2. The output valuations of weakly dihomotopic directed paths are the same. Hence weak dihomotopy classes (i.e. the fundamental category of the model) act on valuations.

3. The weak dihomotopy class of an execution path only contains execution paths.
Directed homotopy

sem: 1a  sync: 1b

\[ y := 0 \]
\[ W(b) \]
\[ P(a) \]
\[ x := y \]
\[ V(a) \]
\[ z := 1 \]
Directed homotopy

sem: 1a  sync: 1b

\[
\begin{align*}
y := 0 & \quad W(b) \\
x := y & \quad P(a) \\
\end{align*}
\]

\[
\begin{align*}
z := 1 & \quad W(b) \\
\end{align*}
\]
Directed homotopy

sem: 1 a sync: 1 b
Directed homotopy

sem: 1 a sync: 1 b
Directed homotopy

sem: 1 a    sync: 1 b

\[ y := 0 \]
\[ W(b) \]
\[ P(a) \]
\[ x := y \]
\[ V(a) \]
\[ z := 1 \]
Directed homotopy

sem: 1 a     sync: 1 b
Directed homotopy

sem: 1a  sync: 1b
Directed homotopy

sem: 1a  sync: 1b

\[
\begin{align*}
&y := 0 \\
&W(b) \\
&P(a) \\
&x := y \\
&W(b) \\
&P(a) \\
&z := 1
\end{align*}
\]
Directed homotopy

\[ \text{sem: 1 a sync: 1 b} \]
Directed homotopy

sem: 1 a  sync: 1 b

\[ y := 0 \]
\[ W(b) \]
\[ P(a) \]
\[ x := y \]
\[ V(a) \]
\[ z := 1 \]
\[ W(b) \]
\[ P(a) \]
\[ z := x \]
\[ V(a) \]
Directed homotopy

sem: 1 a sync: 1 b

\[ x := y \]
\[ z := 1 \]

\[ W(b) \]
\[ P(a) \]
\[ V(a) \]
Directed homotopy

sem: 1 a sync: 1 b
Directed homotopy

$\text{sem: 1 a} \quad \text{sync: 1 b}$
Directed homotopy

sem: 1 a  sync: 1 b
Directed homotopy

sem: 1 a  sync: 1 b
Independence of programs

Model independence

Theorem [Haucourt - not published yet]: The following chain of implications is strict.

syntactic independence ⇓ model independence ⇓ observational independence
Independence of programs

Observational independence

\[ P_1 \{ \begin{array}{c}
\mu \\
\mu'
\end{array} \} \quad P_2 \{ \begin{array}{c}
\mu \\
\mu'
\end{array} \} \]

\[ J_{P_1|P_2} = J_{P_1} \times J_{P_2} \]

Theorem [Haucourt - not published yet]: The following chain of implications is strict.

Syntax independence \(\Downarrow\) Model independence \(\Downarrow\) Observational independence
Independence of programs

Observational independence

Model independence

\[
\begin{bmatrix}
P_1 \\
\mid \\
P_2
\end{bmatrix}
= \begin{bmatrix}
P_1
\end{bmatrix} \times \begin{bmatrix}
P_2
\end{bmatrix}
\]
Independence of programs

Observational independence

\[ \begin{align*}
P_1 \setminus \{ \mu \} & \quad \rightarrow \\
\mu' \quad \rightarrow & \\
\begin{align*}
P_1 \setminus \{ \mu' \} & \quad \rightarrow \\
\mu & \quad \rightarrow
\end{align*}
\end{align*} \]

Model independence

\[
\begin{align*}
\left[ P_1 \mid P_2 \right] & = \left[ P_1 \right] \times \left[ P_2 \right]
\end{align*}
\]

Theorem [Haucourt - not published yet]: The following chain of implications is strict.

- syntactic independence
- \rightarrow
- model independence
- \rightarrow
- observational independence
Parallelizing a program

sem: 1 a
sem: 2 c

proc:
  p = P(a);P(c);V(c);V(a)

  q = P(c);V(c)

init: 2p q
## Parallelizing a program

### Code Snippet

<table>
<thead>
<tr>
<th>sem: 1 a</th>
<th>sem: 1 a</th>
</tr>
</thead>
<tbody>
<tr>
<td>sem: 2 c</td>
<td>sem: 2 c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>proc:</th>
<th>proc:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p = \text{P}(a); \text{P}(c); \text{V}(c); \text{V}(a)$</td>
<td>$q = \text{P}(c); \text{V}(c)$</td>
</tr>
</tbody>
</table>

| init: 2p | init: q |
Parallelizing a program

sem: 1 a

proc:
p = P(a); V(a)

init: 2p

sem: 1 a

proc:
q = ()

init: q
5. Handling Continuous Models
Almost finite graphs

A graph $G$ is said to be linear when $|G|$ is an interval of $\mathbb{R}$.
Almost finite graphs

A graph $G$ is said to be linear when $|G|$ is an interval of $\mathbb{R}$. 
Almost finite graphs

A graph $G$ is said to be linear when $|G|$ is an interval of $\mathbb{R}$. 
Characterizing almost finite graphs

Theorem [Haucourt - Ninin not published yet]

Given a graph \( G \), the following are equivalent:

- \( G \) is almost finite,
- The collection \( R_G \) forms a Boolean subalgebra of \( 2^{|G|} \),
- The following sum is finite
  \[
  \sum_{v \text{ vertex}} |\text{deg}(v) - 2| + \#\{\text{connected components}\} < \infty
  \]
- The Freudenthal extension of \( |G| \) is homeomorphic with the geometric realization of some finite graph.

When the preceding statements are satisfied, the number of ends of \( |G| \) is the number of infinite connected components of \( L \).
Characterizing almost finite graphs

$\mathcal{R}_G = \{\text{finite union of connected subsets of } |G|\}$

Theorem [Haucourt - Ninin not published yet]

Given a graph $G$, the following are equivalent:

- $G$ is almost finite,
- The collection $\mathcal{R}_G$ forms a Boolean subalgebra of $2^{|G|}$,
- The following sum is finite
  \[ \sum_{\text{vertex } v} \left| \deg(v) - 2 \right| + \# \{\text{connected components}\} < \infty \]
- The Freudenthal extension of $|G|$ is homeomorphic with the geometric realization of some finite graph.

When the preceding statements are satisfied, the number of ends of $|G|$ is the number of infinite connected components of $L$. 

43 / 75
Characterizing almost finite graphs

\[ \mathcal{R}_G = \{ \text{finite union of connected subsets of } |G| \} \]

**Theorem [Haucourt - Ninin not published yet]**

Given a graph \( G \), the following are equivalent:

- \( G \) is almost finite,
- The collection \( \mathcal{R}_G \) forms a Boolean subalgebra of \( 2^{|G|} \)
- The following sum is finite
  \[
  \sum_{v \text{ vertex}} |\deg(v) - 2| + \#\{\text{connected components}\} < \infty
  \]
- The **Freudenthal extension** of \( |G| \) is homeomorphic with the geometric realization of some finite graph.

When the preceding statements are satisfied, the number of ends of \( |G| \) is the number of infinite connected components of \( L \).
Isothetic regions (1)
block: $B_1 \times \cdots \times B_N$ with $B_n \neq \emptyset$ and $B_n \in \mathcal{R}_{G_n}$ for $1 \leq n \leq N$. 
Isothetic regions (1)

block: $B_1 \times \cdots \times B_N$ with $B_n \neq \emptyset$ and $B_n \in \mathcal{R}_{G_n}$ for $1 \leq n \leq N$.

**Theorem [Haucourt - not published yet]**

The (nonempty) graphs $G_1, \ldots, G_N$ are almost finite, iff

$$\mathcal{R}_{G_1,\ldots,G_N} = \{\text{finite unions of blocks}\}$$

is a Boolean subalgebra of $2^{|G_1| \times \cdots \times |G_N|}$.

In that case $\mathcal{R}_{G_1,\ldots,G_N}$ is stable under interior, closure, forward and backward operators, and its elements are the subsets of $|G_1| \times \cdots \times |G_N|$ with finitely many maximal subblocks. They are called isothetic regions.

$$\text{frw}(A, B) = \bigcup \{\text{img}(\delta) \mid \delta \text{ a dipath of } A \cup B \text{ starting in } A\}$$

$$\text{bck}(A, B) = \bigcup \{\text{img}(\delta) \mid \delta \text{ a dipath of } A \cup B \text{ ending in } B\}$$
Isothetic regions (2)

Main motivation

The continuous model of a program is an isothetic region.
Maximal subblocks of the state space

#mtx a b
proc:
q = P(b).P(a).V(a).V(b)
init: p q
Swiss Flag
Maximal subblocks of the state space

#mtx a b
proc:
q = P(b).P(a).V(a).V(b)
init:  p  q
Swiss Flag
Maximal subblocks of the state space

```
#mtx a b
proc:
q = P(b).P(a).V(a).V(b)
init: p q
```
Swiss Flag

Maximal subblocks of the state space

#mtx a b
proc:
p = P(a) . P(b) . V(b) . V(a)
q = P(b) . P(a) . V(a) . V(b)
init:  p q
3D Swiss cross

Tetrahemihexacron
The Lipski algorithm

No deadlock
Applications

Past attractor and deadlocks

\[ \Omega = |G_1| \times \cdots \times |G_N| \]

\[ \text{cone}^p A = \{ p \in \Omega \text{ from which } A \text{ can be reached} \} = \text{bck}(A, \Omega) \]

\[ \text{escape}^f A = \{ p \in \Omega \text{ from which } A \text{ cannot be reached} \} \]

\[ \text{escape}^f A = (\text{cone}^p A)^c \]

\[ \text{att}^p A = \{ p \in \Omega \text{ from which } A \text{ cannot be avoided} \} \]

\[ \text{att}^p A = \text{escape}^f(\text{escape}^f A) \]
Swiss Flag
Past attractor of a deadlock point

```plaintext
#mtx a b
proc:
q = P(b).P(a).V(a).V(b)
init:  p q
```
Swiss Flag
Past attractor of a deadlock point

#mtx a b
proc:
q = P(b).P(a).V(a).V(b)
init:  p q
Swiss Flag
Past attractor of a deadlock point

#mtx a b
proc:
q = P(b).P(a).V(a).V(b)
init: p q
Swiss Flag

Past attractor of a deadlock point

#mtx a b
proc:
q = P(b).P(a).V(a).V(b)
init: p q
Swiss Flag
Past attractor of a deadlock point

```
#mtx a b
proc:
q = P(b).P(a).V(a).V(b)
init:  p q
```
Three dining philosophers

Deadlock
Tensor product of Boolean algebras

Blocks are pure tensors
Tensor product of Boolean algebras
Blocks are pure tensors

For $\Omega \in \mathcal{R}_{G_1, \ldots, G_n}$ define

$$\mathcal{R}_\Omega = \{ X \in \mathcal{R}_{G_1, \ldots, G_n} \mid A \subseteq \Omega \}$$

For all elements $A \in \mathcal{R}_{\Omega_1}$, and $B, C \in \mathcal{R}_{\Omega_2}$:

$$(A \times B) \cup (A \times C) = A \times (B \cup C)$$

$$(A \times B) \cap (A \times C) = A \times (B \cap C)$$

$$A \times \emptyset = \emptyset$$

but

$$A \times \Omega_2 \neq \Omega_1 \times \Omega_2$$
## Tensor product of Boolean algebras

### Semilattices and some other algebraic theories

<table>
<thead>
<tr>
<th>Structure</th>
<th>Signature</th>
<th>Axioms</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>semilattice</td>
<td>∨</td>
<td>commutative idempotent semigroup</td>
<td>SLat</td>
</tr>
<tr>
<td>semilattice with zero</td>
<td>∨, 0</td>
<td>commutative idempotent monoid</td>
<td>SLat₀</td>
</tr>
<tr>
<td>lattice</td>
<td>∨, ∧</td>
<td>two semilattices with $\sqcap = \sqcup^\text{op}$</td>
<td>Lat</td>
</tr>
<tr>
<td>distributive lattice</td>
<td>∨, ∧</td>
<td>lattice in which $\sqcap$ distributes over $\sqcup$</td>
<td>DLat</td>
</tr>
<tr>
<td>distributive lattice with zero</td>
<td>∨, 0, ∧</td>
<td>distributive lattice in which $\lor$ has a neutral element</td>
<td>DLat₀</td>
</tr>
<tr>
<td>distributive lattice with difference</td>
<td>∨, 0, ∧, \</td>
<td>distributive lattice with zero s.t. $(x \setminus y) \lor (x \land y) = x$ $(x \setminus y) \land y = 0$</td>
<td>DLatₓ</td>
</tr>
<tr>
<td>bounded distributive lattice</td>
<td>∨, 0, ∧, 1</td>
<td>lattice in which both $\lor$ and $\land$ have a neutral element</td>
<td>DLatₓ</td>
</tr>
<tr>
<td>Boolean algebra</td>
<td>∨, 0, ∧, 1, _c</td>
<td>bounded distributive lattice s.t. $x^c \land x = 0$ and $x^c \lor x = 1$</td>
<td>BoolAlg</td>
</tr>
<tr>
<td></td>
<td>∨, 0, ∧, 1, \</td>
<td>bounded distributive lattice with difference</td>
<td></td>
</tr>
</tbody>
</table>
Tensor product of Boolean algebras

Fraser, G. A., *The semilattice tensor product of distributive lattices*, 1976
Tensor product of Boolean algebras
Fraser, G. A., *The semilattice tensor product of distributive lattices*, 1976

\[
\begin{array}{cccc}
\text{BoolAlg} & \longrightarrow & \text{DLat}_d & \longrightarrow \text{DLat}_0 & \longrightarrow \text{SLat}_0 & \longrightarrow \text{SLat}
\end{array}
\]

**Theorem [Haucourt - Ninin (2014)]**

The universal tensor products of (finitely many) Boolean algebras in *SLat*$_0$, *DLat*$_0$, and *DLat*$_d$ are isomorphic Boolean algebras. Moreover

\[
\mathcal{R}_{\Omega_1} \times \cdots \times \mathcal{R}_{\Omega_N} \cong \mathcal{R}_{\Omega_1} \otimes \cdots \otimes \mathcal{R}_{\Omega_N}
\]
6. Factoring
Example of factorization
Example of factorization
Example of factorization
Example of factorization
Example of factorization

\[
\begin{array}{cccc}
\bullet & \diamond & \circ & \diamond \\
\star & \pentagon & \star & \pentagon & \circ & \diamond \\
\star & \pentagon & \star & \pentagon & \circ & \diamond \\
\end{array}
\]
Homogeneous languages

Main results

**Theorem [Balabonki - Haucourt (2010)]**
The collection \( \mathcal{H}(\mathbb{A}) \) forms a free commutative monoid under the product induced by word concatenation and the zero language.

**Theorem [Balabonki - Haucourt (2010)]**
The collection of isothetic regions \( \bigcup_{n \in \mathbb{N}} R_{G, \ldots, G} \) forms a free commutative monoid that is isomorphic to \( \mathcal{H}(\mathbb{A}) \) with \( \mathbb{A} \) the collection of connected subsets of \( |G| \).
Application to program factoring

If $X$ is the model of a program $P$, a factorization of $X$ induces a family of model independent programs whose parallel compound is $P$. 
Parallelizing a program

sem: 1 a b
sem: 2 c

proc:
  p = P(a);P(c);V(c);V(a)
  q = P(b);P(c);V(c);V(b)

init: p q p q
Factoring the space of states

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[0,+∞]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[4,+∞]</td>
<td>[0,+∞]</td>
<td>[0,+∞]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[0,+∞]</td>
<td>[0,1]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[0,+∞]</td>
<td>[4,+∞]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[4,+∞]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[4,+∞]</td>
<td>[0,1]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[4,+∞]</td>
<td>[4,+∞]</td>
<td>[0,+∞]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[0,1]</td>
<td>[0,+∞]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[4,+∞]</td>
<td>[0,+∞]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[0,1]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[4,+∞]</td>
<td>[0,1]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[0,1]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[4,+∞]</td>
<td>[0,1]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[0,1]</td>
<td>[4,+∞]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[0,1]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[4,+∞]</td>
</tr>
<tr>
<td>[0,1]</td>
<td>[0,+∞]</td>
<td>[4,+∞]</td>
<td>[4,+∞]</td>
</tr>
</tbody>
</table>
Factoring the space of states
Factoring the space of states
Factoring the space of states
Factoring the space of states
Factoring the space of states
Factoring the space of states
Factoring the space of states

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>★</td>
</tr>
</tbody>
</table>
Factoring the space of states

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>★</td>
<td>★</td>
<td>☆</td>
<td>☆</td>
</tr>
<tr>
<td>☆</td>
<td>★</td>
<td>☆</td>
<td>★</td>
</tr>
<tr>
<td>☆</td>
<td>☆</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>☆</td>
<td>★</td>
<td>☆</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>☆</td>
<td>☆</td>
</tr>
<tr>
<td>☆</td>
<td>★</td>
<td>☆</td>
<td>★</td>
</tr>
<tr>
<td>☆</td>
<td>☆</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>☆</td>
<td>★</td>
<td>☆</td>
<td>★</td>
</tr>
</tbody>
</table>

**Diagram Description**: The diagram represents a 4x4 grid of states, factoring the space of states. Each cell contains symbols representing different states or categories, with ★ and ☆ used to denote specific conditions or attributes within the space.
### Parallelizing a program

<table>
<thead>
<tr>
<th>sem:</th>
<th>1 a b</th>
<th>sem:</th>
<th>1 a b</th>
</tr>
</thead>
<tbody>
<tr>
<td>sem:</td>
<td>2 c</td>
<td>sem:</td>
<td>2 c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>proc:</th>
<th>p = P(a);P(c);V(c);V(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>init:</td>
<td>2p</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>proc:</th>
<th>q = P(b);P(c);V(c);V(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>init:</td>
<td>2q</td>
</tr>
</tbody>
</table>
## Parallelizing a program

<table>
<thead>
<tr>
<th>sem: 1 a</th>
<th></th>
<th>sem: 1 b</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>proc:</strong></td>
<td><strong>proc:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p = P(a); V(a)</td>
<td></td>
<td>q = P(b); V(b)</td>
<td></td>
</tr>
<tr>
<td><strong>init:</strong> 2p</td>
<td></td>
<td><strong>init:</strong> 2q</td>
<td></td>
</tr>
</tbody>
</table>
1. Is the following decomposition unique?

\[ \mathcal{R}_{\Omega_1} \times \cdots \times \Omega_N \cong \mathcal{R}_{\Omega_1} \otimes \cdots \otimes \mathcal{R}_{\Omega_N} \]

2. In that case, does the Boolean algebra decomposition matches that of isothetic regions?

R. S. Pierce (Tensor Product of Boolean Algebras, 1983) proved that for all \( n \in \mathbb{N} \) there exists a countable Boolean algebra \( b \) such that the tensor products \( b, b^2, \ldots, b^n \) are distinct though \( b^n = b^{n+1} \). Yet, Boolean algebras of the form \( \mathcal{R}_{\Omega} \) are very specific.
Unique decomposition conjectures

Metrics

Any isothetic region can be turned into a finite affine rank length metric space in a natural way.

T. Foertsch and A. Lytchak (The De Rham Decomposition Theorem for Metric Spaces, 2008) proved a unique decomposition property for finite affine rank geodesic metric spaces.

1. Does the result extend to length metrics?

2. In that case, does the metric decomposition matches that of isothetic regions?
Unique decomposition conjectures
Categories of components vs Isothetic regions

category of components $\pi_0(C)$: a generalized notion of skeleton that fits with categories $C$ with no isomorphisms but identities. Well-defined for all loop-free categories.

E.g.: $\pi_1(X)$ for some isothetic region $X$.

Property (Haucourt 2006): For $C$ loop-free, $\pi_0(C) = 1$ iff $C$ is a lattice.

Property: $\pi_1$ and $\pi_0$ preserves products.

Theorem (Balabonski 2006, unpublished): the collection of (isomorphism classes of) nonempty connected finite loop-free categories with Cartesian product form a free commutative monoid $M$.

Problem: relate the decomposition of an isothetic region to that of its category of components.

E.g.: $\pi_0(\pi_1([0, 1])) = 1$ though $[0, 1]$ is not the neutral isothetic region.
7. Directed Topology
Beyond locally ordered spaces

S. Krishnan, *Convenient Category of Locally Preordered Spaces*, 2009

**Theorem (Haucourt 2012)**

All the categories on the diagram are *complete* and *cocomplete*. 
Realization of (pre)cubical sets

Glabbeek (van), R.J., *Bisimulations for Higher Dimensional Automata*, 1991

Face maps:

\[ xx01 \equiv (2, x_1 x_2 01) : (a, b) \in [0, 1]^2 \mapsto (a, b, 0, 1) \in [0, 1]^4 \]
Realization of (pre)cubical sets

Glabbeek (van), R.J., *Bisimulations for Higher Dimensional Automata*, 1991

Face maps:

\[ xx01 \equiv (2, x_1 x_2 01) : (a, b) \in [0, 1]^2 \mapsto (a, b, 0, 1) \in [0, 1]^4 \]

Degeneracy maps:

\[ (4, x_1 x_3) : (a, b, c, d) \in [0, 1]^4 \mapsto (a, c) \in [0, 1]^2 \]
Realizations in streams and d-spaces
and their fundamental categories

**Theorem (Haucourt 2012)**

For any cubical set $K$, the fundamental categories of the following objects are isomorphic: $D(\lhd K\rhd_{\mathbf{Strm}})$, $\lhd K\rhd_{\mathbf{Strm}}$, $\lhd K\rhd_{\mathbf{Strm}_d}$, $S(\lhd K\rhd_{\mathbf{dTop}_f})$, $\lhd K\rhd_{\mathbf{dTop}_f}$.

But they may differ from the fundamental category of $\lhd K\rhd_{\mathbf{dTop}}$.

**Conjecture**

If $K$ is a precubical set the preceding pathology vanishes.
The downward spiral

A directed path on the directed complex plane
Fundamental category vs fundamental groupoid

$\pi_1$ and $\Pi_1$ are the fundamental category and the fundamental groupoid functors.

$G : \textbf{Cat} \to \textbf{Grd}$ is the enveloping groupoid functor (i.e. left adjoint to $\textbf{Cat} \hookrightarrow \to \textbf{Grd}$)

$U$ is the forgetful functor to $\textbf{Top}$.

There is a natural transformation $g : G \circ \pi_1 \to \Pi_1 \circ U$
Fundamental category vs fundamental groupoid

\( \pi_1 \) and \( \Pi_1 \) are the fundamental category and the fundamental groupoid functors.

\( G : \text{Cat} \to \text{Grd} \) is the enveloping groupoid functor (i.e. left adjoint to \( \text{Cat} \hookrightarrow \text{Grd} \))

\( U \) is the forgetful functor to \( \text{Top} \).

There is a natural transformation \( g : G \circ \pi_1 \to \Pi_1 \circ U \)

**Conjecture:** The groupoid morphism \( g_X \) is an isomorphism when \( X \) is:

- the directed realization of a precubical set
- an isothetic region
Direction generated by vector fields on a manifold

Given some tuple of vector fields \( f_1, \ldots, f_k \) over a manifold \( \mathcal{M} \), the **forward cone** of \( \mathcal{M} \) at \( x \) is the set

\[
F_x := \left\{ \sum_{i=1}^{k} \lambda_i \cdot f_i(x) \mid \lambda_i \geq 0 \text{ for } i = 1, \ldots, k \right\}
\]

A curve \( \gamma \) is said to be **forward** (with respect to \( f_1, \ldots, f_k \)) when its derivative at time \( t \) belongs to \( F_{\gamma(t)} \) for all \( t \in \text{dom} \gamma \):

\[
\frac{\partial \gamma}{\partial t}(t) \in F_{\gamma(t)}
\]

Example: \( \mathbb{R}^n \) with the constant vector fields \( f_k(x) = (\ldots, 0, 1, 0, \ldots) \)
A parallelization of a manifold $\mathcal{M}$ of dimension $n$ is an tuple of vector fields $(f_1, \ldots, f_n)$ s.t. for all $x \in \mathcal{M}$, $(f_1(x), \ldots, f_n(x))$ is a vector basis of the tangent space of $\mathcal{M}$ at $x$ namely $T_x \mathcal{M}$.

A manifold $\mathcal{M}$ is said to be parallelizable when it admits a parallelization. All parallelizations of a given manifold are “isomorphic” (the frame manifold acts transitively on the set of parallelizations).

**Conjecture:** Any parallelization induces a local pospace structure on its underlying manifold. That local pospace structure does not depend on the parallelization.

**Conjecture:** Given a manifold $\mathcal{M}$ equipped with the local order induced by some parallelization, there exists a precubical set $K$ whose local pospace realization is the local pospace $\mathcal{M}$.

**Example:** Every Lie group is parallelizable.

**Example:** It works for the circle! What about the spheres $S^3$ and $S^7$?
8. Conclusion
1. Connect a value analysis to the backend of the static analyzer ALCOOL
2. Prove that all precubical sets can be realized in the category of local pospaces
3. Extend the notion of category of components to realization of precubical sets and isothetic regions
4. Directed version of the Gelfand-Naimark-Segal theorem
1. Connect a value analysis to the backend of the static analyzer ALCOOL

2. Prove that all precubical sets can be realized in the category of local pospaces

3. Extend the notion of category of components to realization of precubical sets and isothetic regions

4. Directed version of the Gelfand-Naimark-Segal theorem
1. Connect a value analysis to the backend of the static analyzer ALCOOL

2. Prove that all precubical sets can be realized in the category of local pospaces

3. Extend the notion of category of components to realization of precubical sets and isothetic regions

4. Directed version of the Gelfand-Naimark-Segal theorem
1. Connect a value analysis to the backend of the static analyzer ALCOOL
2. Prove that all precubical sets can be realized in the category of local pospaces
3. Extend the notion of category of components to realization of precubical sets and isothetic regions
4. Directed version of the Gelfand-Naimark-Segal theorem
1. Connect a value analysis to the backend of the static analyzer ALCOOL
2. Prove that all precubical sets can be realized in the category of local pospaces
3. Extend the notion of category of components to realization of precubical sets and isothetic regions
4. Directed version of the Gelfand-Naimark-Segal theorem