GEOMETRIC MODELS OF CONCURRENT COMPUTATIONS

SAMUEL MIMRAM
École Polytechnique

Habilitation à diriger des recherches

16 September 2016

Concurrent programs

We are interested in **concurrent** programs:

- they consist in multiple subprograms running in parallel,
- their scheduling is inherently non-deterministic.

$$A \parallel B \quad \rightsquigarrow \quad \begin{tikzcd} & {} \arrow[dl, "A"'] \arrow[dr, "B"] & \\ {} \arrow[dr, "B"'] & & {} \arrow[dl, "A"] \\ & {} & \end{tikzcd}$$

They raise specific problems:

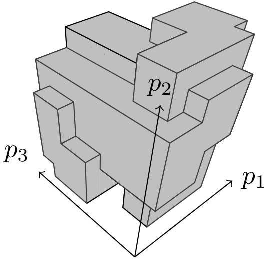
- how can we efficiently verify those programs?
- how can we represent the **state space** of those programs?
  (= all possible states the program can be)


We should understand
the structure of computations,
applications will follow.

The geometric approach

The general idea is that we are going to interpret the state space of programs as a **geometric space**

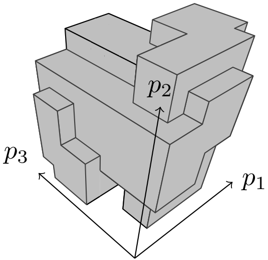
$$\begin{array}{ll} & P_a\,;P_b\,;P_c\,;V_a\,;P_f\,;V_c\,;V_b\,;V_f \\ \parallel & P_d\,;P_e\,;P_a\,;V_d\,;P_c\,;V_e\,;V_a\,;V_c \\ \parallel & P_b\,;P_f\,;V_b\,;P_d\,;V_f\,;P_e\,;V_d\,;V_e \end{array} \quad \rightsquigarrow$$



The geometric approach

The general idea is that we are going to interpret the state space of programs as a **geometric space**

$$
\begin{array}{ll}
 & P_a\,;P_b\,;P_c\,;V_a\,;P_f\,;V_c\,;V_b\,;V_f \\
\parallel & P_d\,;P_e\,;P_a\,;V_d\,;P_c\,;V_e\,;V_a\,;V_c \\
\parallel & P_b\,;P_f\,;V_b\,;P_d\,;V_f\,;P_e\,;V_d\,;V_e
\end{array}
\quad \rightsquigarrow
$$



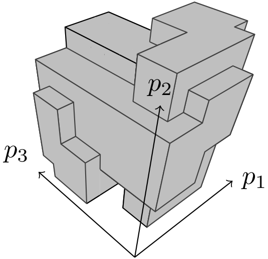
which helps programmers

- provides a way to visualize programs
- helps to come up with new methods
- allows the use of powerful **invariants**
  (fundamental category, curvature, homology, etc.)

The geometric approach

The general idea is that we are going to interpret the state space of programs as a **geometric space**

$$\begin{array}{ll} & P_a ; P_b ; P_c ; V_a ; P_f ; V_c ; V_b ; V_f \\ \parallel & P_d ; P_e ; P_a ; V_d ; P_c ; V_e ; V_a ; V_c \\ \parallel & P_b ; P_f ; V_b ; P_d ; V_f ; P_e ; V_d ; V_e \end{array} \quad \rightsquigarrow$$



which raises new questions

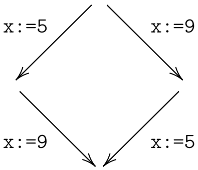
- requires adapting classical concepts in order to incorporate the **direction of time**
- provides interesting classes of spaces

Commutation of actions

In concurrent programs, some actions can be interleaved

`x := 5` $\quad\|\quad$ `x := 9`

which means that we have the following executions:



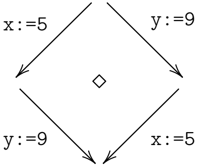
In fact, the resulting `x` could even be different from 5 and 9!
(we should ensure that the two actions are mutually exclusive)

Commutation of actions

In concurrent programs, some actions do **commute**

`x := 5      ‖      y := 9`

in the sense that their order do not matter



Commutation of actions

Two executions which are equivalent up to reordering of commuting actions give rise to the same result:

we can **reduce** the state-space!

$$
\begin{tikzcd}
x_{04} \arrow[r, "P_a"] & x_{14} \arrow[r, "P_b"] & x_{24} \arrow[r, "V_b"] & x_{34} \arrow[r, "V_a"] & x_{44} \\
x_{03} \arrow[u, "V_b"] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{13} \arrow[u] & & x_{33} \arrow[u] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{43} \arrow[u, "V_b"'] \\
x_{02} \arrow[u, "V_a"] & & & & x_{42} \arrow[u, "V_a"'] \\
x_{01} \arrow[u, "P_a"] \arrow[r] & x_{11} & & x_{31} \arrow[r] & x_{41} \arrow[u, "P_a"'] \\
x_{00} \arrow[u, "P_b"] \arrow[r, "P_a"'] \arrow[ur, phantom, "\diamond"] & x_{10} \arrow[u] \arrow[r, "P_b"'] & x_{20} \arrow[r, "V_b"'] & x_{30} \arrow[u] \arrow[r, "V_a"'] \arrow[ur, phantom, "\diamond"] & x_{40} \arrow[u, "P_b"']
\end{tikzcd}
\quad \rightsquigarrow \quad
\begin{tikzcd}
x_{04} \arrow[rr] & & x_{34} \arrow[r, "V_a"] & x_{44} \\
& & x_{33} \arrow[u] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{43} \arrow[u, "V_b"'] \\
x_{01} \arrow[uu] \arrow[r] & x_{11} & & \\
x_{00} \arrow[u, "P_b"] \arrow[r, "P_a"'] \arrow[ur, phantom, "\diamond"] & x_{10} \arrow[u] \arrow[rr] & & x_{40} \arrow[uu]
\end{tikzcd}
$$

Truly concurrent semantics

The control-flow graph should incorporate this information of commutation between actions:

$$
\begin{tikzcd}
& {} \arrow[dl, "A"'] \arrow[dr, "B"] \arrow[dd, phantom, "\diamond"] & \\
{} \arrow[dr, "B"'] & & {} \arrow[dl, "A"] \\
& {} &
\end{tikzcd}
$$

This is called **true concurrency**: Mazurkiewicz traces, trace monoids, asynchronous transition systems, transition systems with independence, automata with concurrency relations, ...

I

THE TITLE

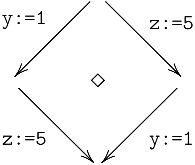
(Geometric Semantics for Concurrent Programs)

Concurrent programs

I am interested in various notions of "concurrent programs".

- An imperative programming language:

$$(\texttt{if } x = 3 \texttt{ then } y := 1 \texttt{ else } y := 2) \quad \| \quad z := 5$$



We want to verify programs, reduce the state-space, find invariants, find problematic code (deadlocks / dead code), etc.

Concurrent programs

(Herlihy, ...)

I am interested in various notions of "concurrent programs".

- Asynchronous protocols with a shared memory:

$$(U_1 ; S_1)^* \quad \| \quad (U_2 ; S_2)^* \quad \| \quad \ldots \quad \| \quad (U_n ; S_n)^*$$

$$\begin{tikzcd} & {} \arrow[dl, "U_i"'] \arrow[dr, "U_j"] & \\ {} \arrow[dr, "U_j"'] & \diamond & {} \arrow[dl, "U_i"] \\ & {} & \end{tikzcd}$$

Which tasks can be implemented in this model, in the presence of *failures*?

Concurrent programs

I am interested in various notions of "concurrent programs".

- Distributed version control systems:

$$\begin{tikzcd} & abc \arrow[dl] \arrow[dr] & \\ ab'c \arrow[dr] & \diamond & ab''c \arrow[dl] \\ & ??? & \end{tikzcd}$$

  What are the atomic operations and their rules?
  How to represent the states of the system in the case of conflicting operations?

Concurrent programs

I am interested in various notions of "concurrent programs".

- (String) rewriting systems:

$$sts \to tst \qquad ss \to 1 \qquad tt \to 1$$

$$\begin{tikzcd}
 & ststs \arrow[dl] \arrow[dr] & \\
tstts \arrow[d] \arrow[drr, phantom, "\diamond"] & & sttst \arrow[d] \\
tss \arrow[dr] & & sst \arrow[dl] \\
 & t &
\end{tikzcd}$$

We want to show confluence, reduce the number of rules, etc.

GEOMETRIC MODELS

Geometric models

The model of **asynchronous graphs** is already quite geometric:

$$\begin{tikzcd}[column sep=small] & x \arrow[dl] \arrow[dr] & \\ y_1 \arrow[dr] & \diamond & y_2 \arrow[dl] \\ & z & \end{tikzcd}$$

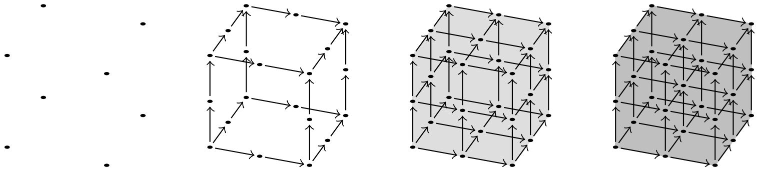
we have:

- 0-dimensional objects: the vertices
- 1-dimensional objects: the edges
- 2-dimensional objects: the commutation squares

Geometric models

(Pratt, van Glabbeek, ...)

In fact we could also take in account commutation of n actions:



$\kappa(a) = 0$ $\kappa(a) = 1$ $\kappa(a) = 2$ $\kappa(a) = 3$

and more. In **precubical sets**, we have:

- 0-dimensional cubes: points
- 1-dimensional cubes: edges
- 2-dimensional cubes: squares
- ...

These can be defined as a suitable presheaf category $\hat{\square}$: each n-cube has a source and target in dimension i for $0 \leq i < n$.

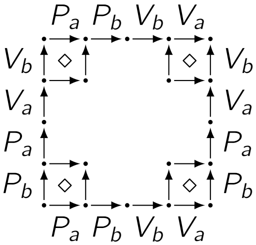
Geometric models

(Pratt, Goubault, Raussen, Haucourt, ...)

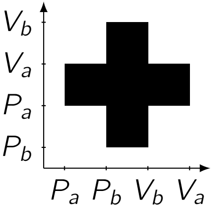
These models are still very algebraic in nature. However, we can realize them as **topological spaces**:

$$|-| \quad : \quad \hat{\square} \to \mathbf{Top}$$

For instance:



$\rightsquigarrow$



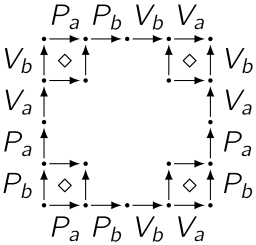
Geometric models

(Pratt, Goubault, Raussen, Haucourt, . . .)

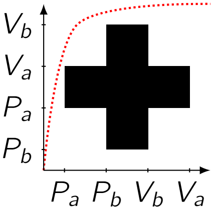
These models are still very algebraic in nature. However, we can realize them as **topological spaces**:

$$|-| \quad : \quad \hat{\square} \to \mathbf{Top}$$

For instance:



$\rightsquigarrow$



dipath	=	execution
dihomotopy	=	equivalence

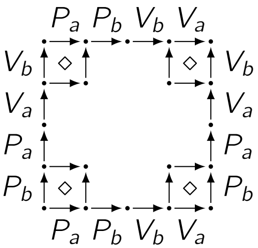
Geometric models

(Pratt, Goubault, Raussen, Haucourt, . . .)

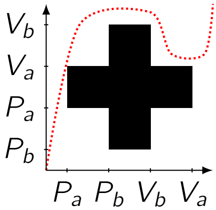
These models are still very algebraic in nature. However, we can realize them as **directed topological spaces**:

$$|-| \quad : \quad \hat{\square} \to \mathbf{dTop}$$

For instance:



$\rightsquigarrow$

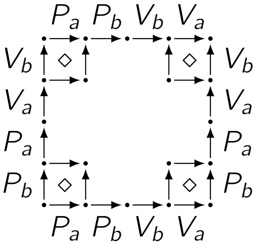


dipath	=	execution
dihomotopy	=	equivalence

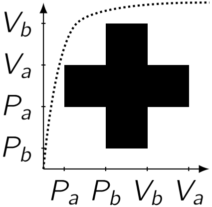
In order to be able to speak about quantitative properties, one would also like to consider **metric** models

$$|-| \quad : \quad \hat{\square} \to \mathbf{Met}$$

For instance:



$\rightsquigarrow$



Geometric models

(Burroni, ...)

Presentations of (higher-)categories, called **polygraphs**, involve relations which are not necessarily squares:

$$\langle s, t \mid st \Rightarrow ts, sts \Rightarrow tst \rangle$$

can be depicted as

$$
\begin{tikzcd}
 & \cdot \arrow[dl, "s"'] \arrow[dr, "t"] & \\
\cdot \arrow[dr, "t"'] \arrow[rr, phantom, "\Rightarrow"] & & \cdot \arrow[dl, "s"] \\
 & \cdot &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & \cdot \arrow[dl, "s"'] \arrow[dr, "t"] & \\
\cdot \arrow[d, "t"'] & & \cdot \arrow[d, "s"] \\
\cdot \arrow[dr, "s"'] \arrow[rr, phantom, "\Rightarrow"] & & \cdot \arrow[dl, "t"] \\
 & \cdot &
\end{tikzcd}
$$

I will present *some* aspects of these models and focus on some important results:

- give an overview of the methods in the field
- give an overview of their possible applications

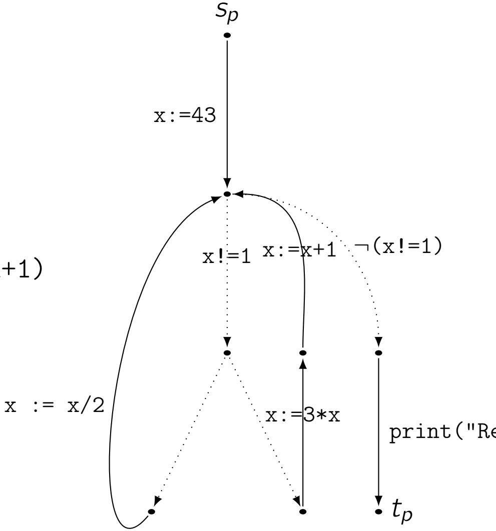
II

MUTEXES AND THE CUBE PROPERTY

Control flow graphs

To any program one can associate a **control flow graph**:

```
x := 43;
while x != 1 do (
    if x mod 2 != 0 then
        (x := 3*x; x := x+1)
    else
        x := x/2
);
print("Reached 1!")
```



How can we extend these to concurrent programs?

Mutexes

In order to prevent incompatible actions from running in parallel, one uses **mutexes**, which are *resources* on which two actions are available

- P_a: *take* the resource a
- V_a: *release* the resource a

and implementation

- guarantees that a resource has been taken at most once at any moment,
- forbids releasing a resource which as not been taken.

Mutexes

In order to prevent incompatible actions from running in parallel, one uses **mutexes**, which are *resources* on which two actions are available

- P_a: *take* the resource a
- V_a: *release* the resource a

and implementation

- guarantees that a resource has been taken at most once at any moment,
- forbids releasing a resource which as not been taken.

Our earlier program should be rewritten as

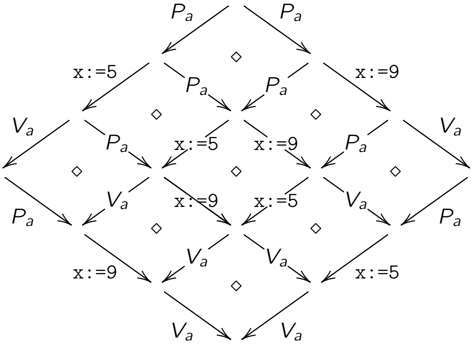
$$P_a\,;\texttt{x:=5}\,;V_a \qquad \| \qquad P_a\,;\texttt{x:=9}\,;V_a$$

Mutexes

In the program

$$P_a\,;\texttt{x:=5}\,;V_a \qquad \parallel \qquad P_a\,;\texttt{x:=9}\,;V_a$$

the possible executions are

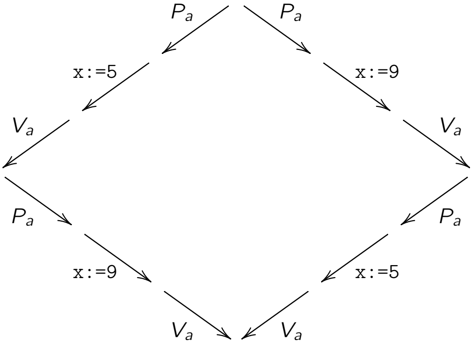


Mutexes

In the program

$$P_a\,;\texttt{x:=5}\,;V_a \qquad \| \qquad P_a\,;\texttt{x:=9}\,;V_a$$

the possible executions are



The cubical semantics

Definition

The **cubical semantics** $\check{C}_p$ of a program p is

- the precubical set C_p associated to p by induction:

$$C_{p \parallel q} \quad = \quad C_p \otimes C_q$$

- with forbidden vertices removed (and iterated cofaces)

The cubical semantics

Definition

The **cubical semantics** $\check{C}_p$ of a program p is

- the precubical set C_p associated to p by induction:

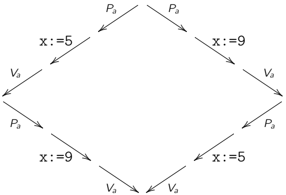
$$C_{p \parallel q} \quad = \quad C_p \otimes C_q$$

- with forbidden vertices removed (and iterated cofaces)

Lemma

The dipaths starting from the initial vertex are in bijection with possible executions of the program.

$$\check{C}_{\dots} \quad =$$



Dihomotopy between dipaths

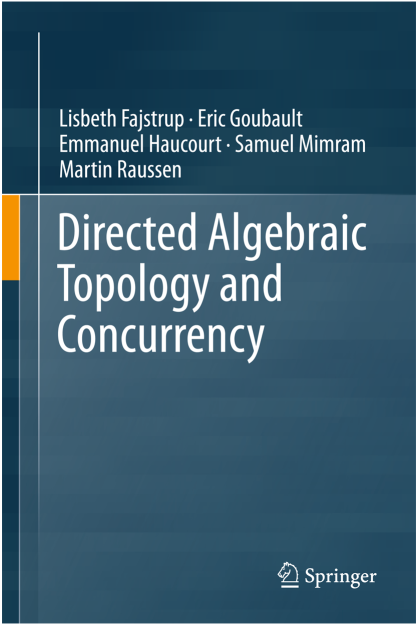
Definition

The **dihomotopy** relation $\hat{\sim}$ between dipaths is the smallest congruence such that $A \,.\, B \;\hat{\sim}\; B' \,.\, A'$ whenever $A \,.\, B \diamond B' \,.\, A'$:

$$\begin{tikzcd}
 & {} \arrow[dl, "A"'] \arrow[dr, "B'"] & \\
{} \arrow[dr, "B"'] & \diamond & {} \arrow[dl, "A'"] \\
 & {} &
\end{tikzcd}$$

Proposition

For "coherent" programs, two dihomotopic executions lead to the same state.


Lisbeth Fajstrup · Eric Goubault
Emmanuel Haucourt · Samuel Mimram
Martin Raussen
Directed Algebraic Topology and Concurrency
Springer

III

HOMOTOPY
VS
DIHOMOTOPY

Dipaths

From a topological point of view, instead of considering directed paths (or **dipaths**)

$$\xrightarrow{A}\ \xrightarrow{B}\ \xrightarrow{C} \qquad \text{or} \qquad A\,.\,B\,.\,C$$

it is more natural to consider **paths**

$$\xrightarrow{A}\ \xleftarrow{B}\ \xrightarrow{C} \qquad \text{or} \qquad A\,.\,\overline{B}\,.\,C$$

where arrows can occur backward.

Homotopy

Similarly, one would usual consider **homotopy** $\sim$ between paths: the smallest congruence, containing dihomotopy $\hat{\sim}$ and such that for every edge

$$x \xrightarrow{A} y$$

we have

$$\mathrm{id}_x \sim A \,.\, \overline{A} \qquad\qquad \overline{A} \,.\, A \sim \mathrm{id}_y$$

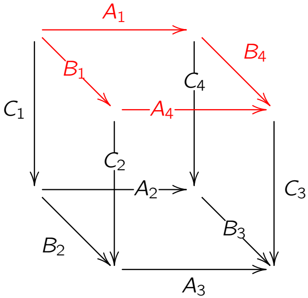
Remark

Clearly $f \mathrel{\hat{\sim}} g$ implies $f \sim g$, but converse is *not* generally true.

Homotopy vs dihomotopy

(Fahrenberg)

Consider the following “matchbox”:



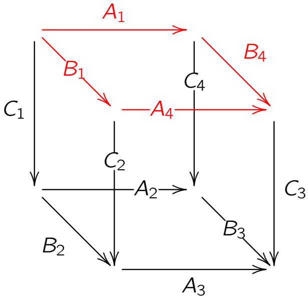
where every square is filled excepting the top one:

$$\xcancel{A_1 \,.\, B_4 \diamond B_1 \,.\, A_4}$$

Homotopy vs dihomotopy

(Fahrenberg)

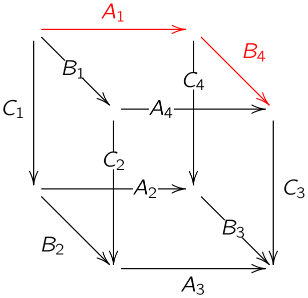
Consider the following “matchbox”:



We have

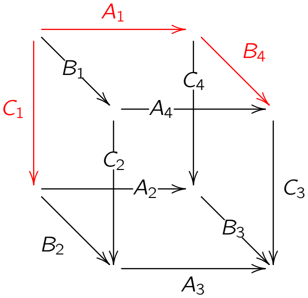
$$A_1 \,.\, B_4 \sim B_1 \,.\, A_4 \qquad \text{but not} \qquad A_1 \,.\, B_4 \mathrel{\hat{\sim}} B_1 \,.\, A_4$$

Homotopy vs dihomotopy



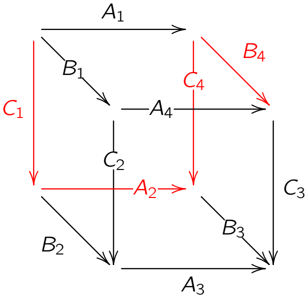
$A_1 . B_4$

Homotopy vs dihomotopy



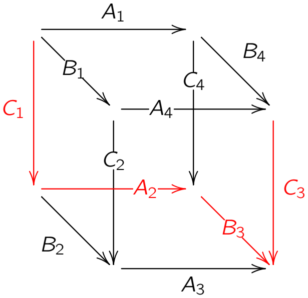
$$A_1 \,.\, B_4 \quad \sim \quad C_1 \,.\, \overline{C_1} \,.\, A_1 \,.\, B_4$$

Homotopy vs dihomotopy



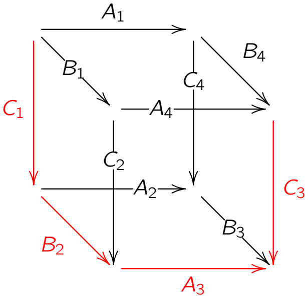
$$\begin{aligned} A_1 \,.\, B_4 &\sim C_1 \,.\, \overline{C_1} \,.\, A_1 \,.\, B_4 \\ &\sim C_1 \,.\, A_2 \,.\, \overline{C_4} \,.\, B_4 \end{aligned}$$

Homotopy vs dihomotopy



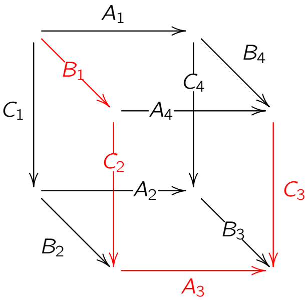
$$
\begin{aligned}
A_1 \,.\, B_4 \quad & \sim \quad C_1 \,.\, \overline{C_1} \,.\, A_1 \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, \overline{C_4} \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, B_3 \,.\, \overline{C_3}
\end{aligned}
$$

Homotopy vs dihomotopy



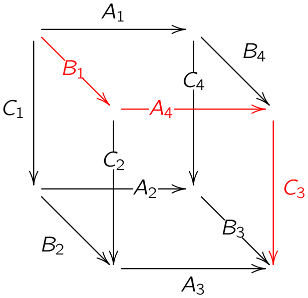
$$
\begin{aligned}
A_1 \,.\, B_4 \quad & \sim \quad C_1 \,.\, \overline{C_1} \,.\, A_1 \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, \overline{C_4} \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, B_3 \,.\, \overline{C_3} \\
& \sim \quad {\color{red} C_1 \,.\, B_2 \,.\, A_3 \,.\, \overline{C_3}}
\end{aligned}
$$

Homotopy vs dihomotopy



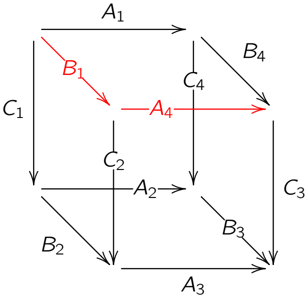
$$
\begin{aligned}
A_1 \,.\, B_4 \quad & \sim \quad C_1 \,.\, \overline{C_1} \,.\, A_1 \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, \overline{C_4} \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, B_3 \,.\, \overline{C_3} \\
& \sim \quad C_1 \,.\, B_2 \,.\, A_3 \,.\, \overline{C_3} \\
& \sim \quad B_1 \,.\, C_2 \,.\, A_3 \,.\, \overline{C_3}
\end{aligned}
$$

Homotopy vs dihomotopy



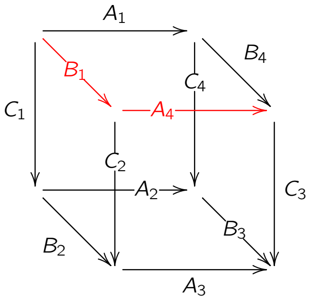
$$
\begin{aligned}
A_1 \,.\, B_4 \quad & \sim \quad C_1 \,.\, \overline{C_1} \,.\, A_1 \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, \overline{C_4} \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, B_3 \,.\, \overline{C_3} \\
& \sim \quad C_1 \,.\, B_2 \,.\, A_3 \,.\, \overline{C_3} \\
& \sim \quad B_1 \,.\, C_2 \,.\, A_3 \,.\, \overline{C_3} \\
& \sim \quad B_1 \,.\, A_4 \,.\, C_3 \,.\, \overline{C_3}
\end{aligned}
$$

Homotopy vs dihomotopy



$$
\begin{aligned}
A_1 \,.\, B_4 \quad & \sim \quad C_1 \,.\, \overline{C_1} \,.\, A_1 \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, \overline{C_4} \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, B_3 \,.\, \overline{C_3} \\
& \sim \quad C_1 \,.\, B_2 \,.\, A_3 \,.\, \overline{C_3} \\
& \sim \quad B_1 \,.\, C_2 \,.\, A_3 \,.\, \overline{C_3} \\
& \sim \quad B_1 \,.\, A_4 \,.\, C_3 \,.\, \overline{C_3} \\
& \sim \quad B_1 \,.\, A_4
\end{aligned}
$$

Homotopy vs dihomotopy



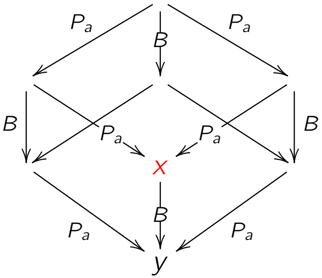
$$
\begin{aligned}
A_1 \,.\, B_4 \quad & \sim \quad C_1 \,.\, \overline{C_1} \,.\, A_1 \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, \overline{C_4} \,.\, B_4 \\
& \sim \quad C_1 \,.\, A_2 \,.\, B_3 \,.\, \overline{C_3} \\
& \sim \quad C_1 \,.\, B_2 \,.\, A_3 \,.\, \overline{C_3} \\
& \sim \quad B_1 \,.\, C_2 \,.\, A_3 \,.\, \overline{C_3} \\
& \sim \quad B_1 \,.\, A_4 \,.\, C_3 \,.\, \overline{C_3} \\
& \sim \quad B_1 \,.\, A_4
\end{aligned}
$$

This example cannot be obtained as the semantics of a program!

Binary conflicts

In a situation such as

$$P_a \parallel P_a \parallel B \qquad =$$

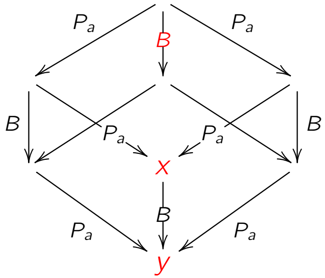


the vertex x is forbidden (and has to be removed).

Binary conflicts

In a situation such as

$$P_a \parallel P_a \parallel B \qquad = $$



the vertex x is forbidden (and has to be removed).

In this case, the vertex y has to be removed too, because $B \neq V_a$!

Proposition

Semantics of programs satisfy the **cube property**:

$$
\begin{tikzcd}
& {} \arrow[dl, "A"'] \arrow[d] \arrow[dr, "C'"] & \\
{} \arrow[d, "B"'] \arrow[r, phantom, "\diamond"] & {} \arrow[dl] \arrow[dr] \arrow[r, phantom, "\diamond"] & {} \arrow[d, "B'"] \\
{} \arrow[dr, "C"'] & \diamond & {} \arrow[dl, "A'"] \\
& {} &
\end{tikzcd}
\quad \Leftrightarrow \quad
\begin{tikzcd}
& {} \arrow[dl, "A"'] \arrow[dr, "C'"] & \\
{} \arrow[d, "B"'] \arrow[dr] & \diamond & {} \arrow[d, "B'"] \arrow[dl] \\
{} \arrow[dr, "C"'] \arrow[r, phantom, "\diamond"] & {} \arrow[d] \arrow[r, phantom, "\diamond"] & {} \arrow[dl, "A'"] \\
& {} &
\end{tikzcd}
$$

(and other more minor properties).

Homotopy vs dihomotopy

Theorem

In a precubical set satisfying the cube property, two dipaths are dihomotopic if and only if they are homotopic: the inclusion functor

$$\vec{\Pi}_1(C) \quad \hookrightarrow \quad \Pi_1(C)$$

is faithful.

Proof.

Uses 2-dimensional rewriting techniques! $\square$

Some consequences

From this follows many interesting properties:

- normal forms for dihomotopy classes of paths,
- an easy definition of universal covers,
- ...

IV

NON-POSITIVELY CURVED PRECUBICAL SETS

Metric semantics

A semantics in metric spaces is desirable:

- we want to have a notion of length of paths (corresponding to the duration of an execution),
- interesting notions, such as *curvature* are available in this context.

Geometric realization

In order to define a geometric realization $|-|$ in metric spaces, we should use the usual formula

$$|C| \quad = \quad \int^{n \in \square} C_n \cdot I^n$$

However, the category of metric spaces does not have enough colimits.

We should consider Lawvere's generalized metric spaces!

Generalizing metric spaces

Definition

A **metric space** is a space X equipped with a metric $d : X \times X \to [0, \infty]$ such that, given $x, y, z \in X$,

$$\begin{array}{lrl} (1) & \text{point equality:} & d(x,x) = 0 \\ (2) & \text{triangle inequality:} & d(x,z) \leq d(x,y) + d(y,z) \\ (3) & \text{finite distances:} & d(x,y) < \infty \\ (4) & \text{separation:} & d(x,y) = 0 \text{ implies } x = y \\ (5) & \text{symmetry:} & d(x,y) = d(y,x) \end{array}$$

We consider contracting maps $f : X \to Y$:

$$d_Y(f(x), f(x')) \qquad \leq \qquad d_X(x, x')$$

Unfortunately, the resulting category is not cocomplete!

Generalizing metric spaces

Definition

A **metric space** is a space X equipped with a metric $d : X \times X \to [0, \infty]$ such that, given $x, y, z \in X$,

$$
\begin{array}{rrl}
(1) & \text{point equality:} & d(x,x) = 0 \\
(2) & \text{triangle inequality:} & d(x,z) \leq d(x,y) + d(y,z) \\
(3) & \text{finite distances:} & d(x,y) < \infty \\
(4) & \text{separation:} & d(x,y) = 0 \text{ implies } x = y \\
(5) & \text{symmetry:} & d(x,y) = d(y,x)
\end{array}
$$

Intuitively, $X + Y$ should be such that

$$d(x,y) \quad = \quad \infty$$

for $x \in X$ and $y \in Y$.

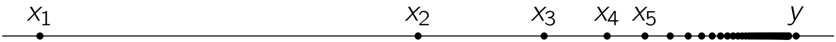
Generalizing metric spaces

Definition

A **metric space** is a space X equipped with a metric $d : X \times X \to [0, \infty]$ such that, given $x, y, z \in X$,

$$
\begin{array}{lrl}
(1) & \text{point equality:} & d(x,x) = 0 \\
(2) & \text{triangle inequality:} & d(x,z) \leq d(x,y) + d(y,z) \\
(3) & \text{finite distances:} & d(x,y) < \infty \\
(4) & \text{separation:} & d(x,y) = 0 \text{ implies } x = y \\
(5) & \text{symmetry:} & d(x,y) = d(y,x)
\end{array}
$$

Consider the relation $\approx$ on X identifying a family of points $(x_i)_{i \in \mathbb{N}}$ such that $d(x_i, y) = 1/i$ for some y



Intuitively, in $X/\approx$, we should have $d([x_i], [y]) = 0$.

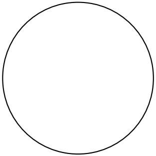
Generalizing metric spaces

Definition

A **metric space** is a space X equipped with a metric $d : X \times X \to [0, \infty]$ such that, given $x, y, z \in X$,

$$\begin{array}{rrl}
(1) & \text{point equality:} & d(x,x) = 0 \\
(2) & \text{triangle inequality:} & d(x,z) \leq d(x,y) + d(y,z) \\
(3) & \text{finite distances:} & d(x,y) < \infty \\
(4) & \text{separation:} & d(x,y) = 0 \text{ implies } x = y \\
(5) & \text{symmetry:} & d(x,y) = d(y,x)
\end{array}$$

We can encode direction in the distance!



$$d(x,y) = \bigwedge \left\{ \rho - \theta \;\middle|\; x = \mathrm{e}^{\mathrm{i}2\pi\theta}, y = \mathrm{e}^{\mathrm{i}2\pi\rho}, \rho \geq \theta \right\}$$

Generalized metric spaces

(Lawvere)

Definition

A **generalized metric space** is a space X equipped with a metric $d : X \times X \to [0, \infty]$ such that, given $x, y, z \in X$,

$$\begin{array}{lrl} (1) & \text{point equality:} & d(x,x) = 0 \\ (2) & \text{triangle inequality:} & d(x,z) \leq d(x,y) + d(y,z) \end{array}$$

Generalized metric spaces

(Lawvere)

Definition

A **generalized metric space** is a space X equipped with a metric $d : X \times X \to [0, \infty]$ such that, given $x, y, z \in X$,

$$\begin{array}{lrl} (1) & \text{point equality:} & d(x,x) = 0 \\ (2) & \text{triangle inequality:} & d(x,z) \leq d(x,y) + d(y,z) \end{array}$$

Proposition

The category **GMet** enjoys the following:

- the category **GMet** is complete and cocomplete,
- the forgetful functor **GMet** $\to$ **Set** has left and right adjoints,
- the forgetful functor **GMet** $\to$ **Top** preserves <u>finite</u> (co)limits.

Metric realization

We can define a metric realization functor

$$|-| \quad : \quad \hat{\square} \quad \to \quad \mathbf{GMet}$$

Theorem

For a locally finite precubical set C, the space $|C|$

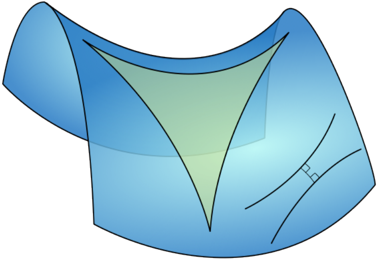
- *has the usual geometric real. as underlying topological space*
- *is a geodesic length space.*

Non-positively curved spaces

The cube property for precubical sets is analogous to Gromov's condition for characterizing NPC cubical complexes:

Theorem

Given a finite dimensional geometric precubical set C satisfying the cube property, it metric realization $|C|$ is **non-positively curved**, *i.e.* **locally CAT(0)**.



In fact, many definitions and properties of
NPC cubical complexes
can be carried on directly in the setting of
precubical sets!

Some consequences

We have the properties of NPC spaces:

- $|C|$ is locally uniquely geodesic
- the universal cover is NPC,
- the fundamental group is automatic,
- links with geometric group theory,
- ...

This suggests generalizations of the cube property:

- what are the corresponding notion of "NPC"?

V

DISTRIBUTED VERSION CONTROL SYSTEMS

DVCS

Distributed Version Control Systems are used when working collaboratively on files and import modifications from other people

SUBVERSION git darcs . . .

A **patch** stores the difference between two files (i.e. the list of inserted and deleted lines).

Users can perform two actions:

- **commit** the difference between the current version and the last committed version as a patch to a server
- **update** its current version by importing all the new patches on the server

Intuitively, we have a category of files and patches.

Using DVCS

Alice Bob

b

Using DVCS

Alice Bob

b

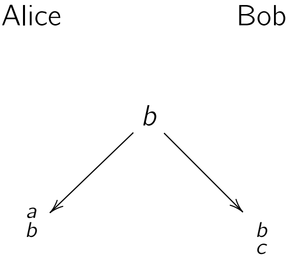
Using DVCS

Alice Bob

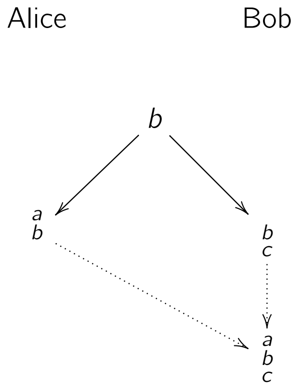
b

a
b

Using DVCS

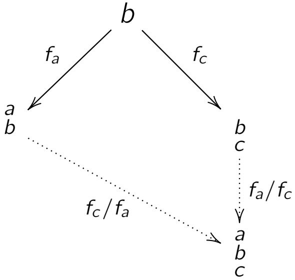


Using DVCS



Using DVCS

Alice Bob



Merging modifications is naturally modeled by pushouts.

In particular, this provides residual patches!

Conflicts

However, not every pair of coinitial morphisms has a pushout!

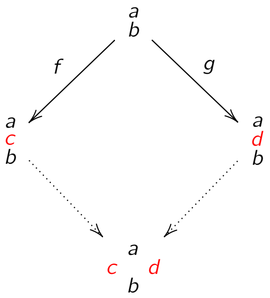
Conflicts

However, not every pair of coinitial morphisms has a pushout!

$$
\begin{tikzcd}
& \begin{matrix}a\\b\end{matrix} \arrow[dl, "f"'] \arrow[dr, "g"] & \\
\begin{matrix}a\\c\\b\end{matrix} \arrow[dr, dotted] & & \begin{matrix}a\\d\\b\end{matrix} \arrow[dl, dotted] \\
& ? &
\end{tikzcd}
$$

Conflicts

However, not every pair of coinitial morphisms has a pushout!




```
a
<<<<<<< HEAD
c
=======
d
>>>>>>> 5c55f7c1c4ad1e02be6d0474e858bd8ad712e22b
b
```

Handling conflicts

We should extend our model to account for "files with conflicts" and their handling.

There were many proposals for this. Instead of discussing the relative merits of each of those, we instead look for a

universal property

that this extension should satisfy: we want to

formally add pushouts.

Handling conflicts

We should extend our model to account for "files with conflicts" and their handling.

There were many proposals for this. Instead of discussing the relative merits of each of those, we instead look for a

universal property

that this extension should satisfy: we want to

formally add finite colimits.

Handling conflicts

We should extend our model to account for “files with conflicts” and their handling.

There were many proposals for this. Instead of discussing the relative merits of each of those, we instead look for a

universal property

that this extension should satisfy: we want to

formally add finite colimits which do not already exist.

Handling conflicts

We should extend our model to account for “files with conflicts” and their handling.

There were many proposals for this. Instead of discussing the relative merits of each of those, we instead look for a

universal property

that this extension should satisfy: we want to

compute a conservative finite cocompletion.

Enforcing confluence

This is a different perspective on concurrency:

- usually, we want to check the **confluence** property:

$$\begin{tikzcd} & {} \arrow[dl, "*"'] \arrow[dr, "*"] & \\ {} \arrow[dr, dotted, "*"'] \arrow[rr, phantom, "\hat{\sim}"] & & {} \arrow[dl, dotted, "*"] \\ & {} & \end{tikzcd}$$

- here, we want to <u>enforce</u> confluence

Let's begin with a simplified model:

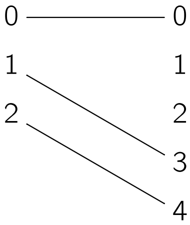
- one file
- lines can only be inserted
- lines don't have contents

The category of files and insertions

Definition

We write $\mathcal{L}$ for the category whose

- objects are sets $\{0, \ldots, n-1\}$ for some $n \in \mathbb{N}$,
- morphisms are injective increasing functions:



Remark

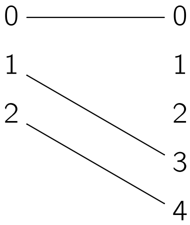
This is also known as the augmented presimplicial category.

The category of files and insertions

Definition

We write $\mathcal{L}$ for the category whose

- objects are sets $\{0, \ldots, n-1\}$ for some $n \in \mathbb{N}$,
- morphisms are injective increasing functions:



Proposition

This is the free monoidal category containing an object 1 and a morphism

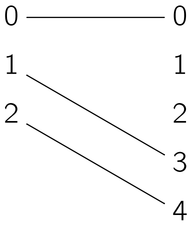
$$\eta \quad : \quad 0 \to 1$$

The category of files and insertions

Definition

We write $\mathcal{L}$ for the category whose

- objects are sets $\{0, \ldots, n-1\}$ for some $n \in \mathbb{N}$,
- morphisms are injective increasing functions:



Remark

We can add labels to the lines by using a slice category construction.

The conservative finite cocompletion

The category $\mathcal{P}$ of "files with conflicts" should be the **free conservative finite cocompletion** of $\mathcal{L}$:

$$\begin{tikzcd} \mathcal{L} \arrow[r, "F"] \arrow[d, "Y"'] & \mathcal{C} \\ \mathcal{P} \arrow[ur, dotted, "\tilde{F}"'] & \end{tikzcd}$$

with

- $\mathcal{P}, \mathcal{C}$ with finite colimits
- $Y, F, \tilde{F}$ preserving finite colimits

Computing $\mathcal{P}$

It is well-known that $\hat{\mathcal{L}}$ is the free cocompletion of $\mathcal{L}$:

$$\begin{tikzcd} \mathcal{L} \arrow[r, "F"] \arrow[d, "Y"'] & \mathcal{C} \\ \hat{\mathcal{L}} \arrow[ur, dotted, "\tilde{F}"'] & \end{tikzcd}$$

Theorem (Kelly)

The free conservative cocompletion of $\mathcal{L}$ is the full subcategory of $\hat{\mathcal{L}}$ whose objects are continuous presheaves.

The finite cocompletion can be obtained by further restricting to "finite" presheaves.

Computing $\mathcal{P}$

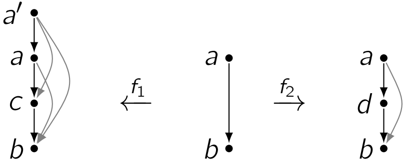
From this one can extract an explicit description of $\mathcal{P}$.

Theorem

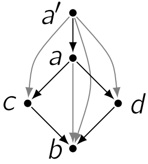
The free conservative finite cocompletion $\mathcal{P}$ of $\mathcal{L}$ is the category:

- *objects $(A, <)$ are finite sets equipped with a transitive relation $<$,*
- *a morphism $f : A \to B$ is a function respecting the relation.*

We have all pushouts, e.g. the pushout of

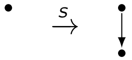


is

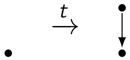


A free cocompletion of $\mathcal{L}$

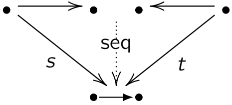
Every object in $\mathcal{P}$ can be obtained as a colimit of objects in $\mathcal{L}$.
For instance, consider the morphisms



and

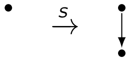


By coproduct, we get a "sequentialization" morphism

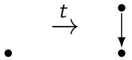


A free cocompletion of $\mathcal{L}$

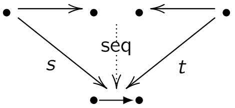
Every object in $\mathcal{P}$ can be obtained as a colimit of objects in $\mathcal{L}$.
For instance, consider the morphisms



and

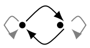


By coproduct, we get a "sequentialization" morphism



The pushout of

is



Computing in $\mathcal{P}$

Notice that we get a way of identifying two independent lines, which can be used to solve a conflict.

$$
\begin{tikzcd}
\bullet \arrow[r] \arrow[drr, "\mathsf{id}_\bullet"'] & \bullet & {} \arrow[d, dotted, "\mathrm{merge}"] & \bullet & \bullet \arrow[l] \arrow[dll, "\mathsf{id}_\bullet"] \\
 & & \bullet & &
\end{tikzcd}
$$

Extensions

There are many possible extensions of this work:

- patches with deletions,
- structured files,
- many files,
- ...

Pijul

(Meunier, Becker, ...)

http://pijul.org/

VI

PRESENTATIONS MODULO OF CATEGORIES

Presentations of n-categories

(Burroni, ...)

An n-category can be *presented* by a **polygraph**, i.e. described as

- the free n-category on given 0-, 1, . . . , n-generators,
- quotiented by a congruence generated by relations on n-cells.

$$
\begin{tikzcd}
P_0 \arrow[d, "i_0"] & P_1 \arrow[d, "i_1"] \arrow[dl, shift right, "s_0"'] \arrow[dl, shift left, "t_0"] & P_2 \arrow[d, "i_2"] \arrow[dl, shift right, "s_1"'] \arrow[dl, shift left, "t_1"] & {} \arrow[dl, shift right, "s_2"'] \arrow[dl, shift left, "t_2"] & \cdots & & P_{n-1} \arrow[d, "i_{n-1}"] \arrow[dl, shift right, "s_{n-2}"'] \arrow[dl, shift left, "t_{n-2}"] & P_n \arrow[dl, shift right, "s_{n-1}"'] \arrow[dl, shift left, "t_{n-1}"] \\
P_0^* & P_1^* \arrow[l, shift right, "s_0^*"'] \arrow[l, shift left, "t_0^*"] & P_2^* \arrow[l, shift right, "s_1^*"'] \arrow[l, shift left, "t_1^*"] & {} \arrow[l, shift right, "s_2^*"'] \arrow[l, shift left, "t_2^*"] & \cdots & {} & P_{n-1}^* \arrow[l, shift right, "s_{n-2}^*"'] \arrow[l, shift left, "t_{n-2}^*"]
\end{tikzcd}
$$

Presentations of n-categories

(Burroni, ...)

An n-category can be *presented* by a **polygraph**, i.e. described as

- the free n-category on given 0-, 1, . . . , n-generators,
- quotiented by a congruence generated by relations on n-cells.

$$
\begin{tikzcd}
P_0 \arrow[d, "i_0"] & P_1 \arrow[d, "i_1"] \arrow[dl, shift right, "s_0"'] \arrow[dl, shift left, "t_0"] & P_2 \arrow[d, "i_2"] \arrow[dl, shift right, "s_1"'] \arrow[dl, shift left, "t_1"] & \cdots \arrow[dl, shift right, "s_2"'] \arrow[dl, shift left, "t_2"] & P_{n-1} \arrow[d, "i_{n-1}"] \arrow[dl, shift right, "s_{n-2}"'] \arrow[dl, shift left, "t_{n-2}"] & P_n \arrow[dl, shift right, "s_{n-1}"'] \arrow[dl, shift left, "t_{n-1}"] \\
P_0^* & P_1^* \arrow[l, shift right, "s_0^*"'] \arrow[l, shift left, "t_0^*"] & P_2^* \arrow[l, shift right, "s_1^*"'] \arrow[l, shift left, "t_1^*"] & \cdots \arrow[l, shift right, "s_2^*"'] \arrow[l, shift left, "t_2^*"] & P_{n-1}^* \arrow[l, shift right, "s_{n-2}^*"'] \arrow[l, shift left, "t_{n-2}^*"] &
\end{tikzcd}
$$

Remark

Since the quotient is performed on n-cells the underlying $(n-1)$-category is free!

Question

Can we coherently quotient in lower dimensions?

We provide an answer for $n = 1$ and $n = 2$.

Presentations of categories.

Graphs

Definition

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

- a set V of *vertices*
- a set E of *edges*
- *source* and *target* functions $s, t : E \to V$

$$\begin{tikzcd} & E \arrow[dl, shift right, "s"'] \arrow[dl, shift left, "t"] \\ V & \end{tikzcd}$$

Graphs

Definition

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

- a set V of *vertices*
- a set E of *edges*
- *source* and *target* functions $s, t : E \to V$

The **free category** generated by G has

- objects: vertices V
- morphisms: paths E^* (with concatenation as composition)

$$
\begin{tikzcd}
 & E \arrow[dl, shift right, "s"'] \arrow[dl, shift left, "t"] \arrow[d, "i"] \\
V & E^* \arrow[l, shift right, "s^*"'] \arrow[l, shift left, "t^*"]
\end{tikzcd}
$$

Presentations of categories

Definition

A **presentation** P of category consists of

- a graph (the *signature*)
- a set of rules relating a path with another path with same source and target

$$\begin{tikzcd}
& y_1 \arrow[r, "f_2"] & y_2 \arrow[r, "f_3"] \arrow[d, phantom, "\rho\Downarrow"] & y_3 \arrow[dr, "f_4"] & \\
x \arrow[ur, "f_1"] \arrow[drr, "g_1"'] & & {} & & z \\
& & y \arrow[urr, "g_2"'] & &
\end{tikzcd}$$

The *presented category* $\|P\|$ is the free category on the graph with paths taken modulo the congruence generated by rules.

Presentations of categories (formally)

Definition

A **presentation** P of category consists of

$$\begin{tikzcd}
 & P_1 \arrow[dl, shift right, "s_0"'] \arrow[dl, shift left, "t_0"] \arrow[d, "i_1"] & P_2 \arrow[dl, shift right, "s_1"'] \arrow[dl, shift left, "t_1"] \\
P_0 & P_1^* \arrow[l, shift right, "s_0^*"'] \arrow[l, shift left, "t_0^*"] &
\end{tikzcd}$$

- a set P_0 of *object generators*
- a set P_1 of *morphism generators*
- a set P_2 of *relations*
  with $s_0^* \circ s_1 = s_0^* \circ t_1$ and $t_0^* \circ s_1 = t_0^* \circ t_1$

Question

How do we add a quotient on objects too?

Presentations modulo

Definition

A **presentation modulo** $(P, \tilde{P}_1)$ of category consists of

- a presentation of category P,
- a set $\tilde{P}_1 \subseteq P_1$ of *equational generators*.

$$
\begin{tikzcd}
 & x_1 \arrow[dl, "f_1"'] \arrow[dd, "g_1"] \arrow[dr, "f_4"] & \\
x_2 \arrow[dr, "f_2"'] & {} \arrow[r, phantom, "\overset{\rho}{\Longrightarrow}"] & x_4 \arrow[dl, "f_3"] \arrow[loop right, "g_2"] \\
 & x_3 \arrow[uu, bend left=90, "g_3"] &
\end{tikzcd}
$$

$$
\begin{aligned}
P_0 &= \{x_i\} \\
P_1 &= \{f_i, g_i\} \\
\tilde{P}_1 &= \{f_i\} \\
P_2 &= \{\rho\}
\end{aligned}
$$

Presentations modulo

Definition

A **presentation modulo** $(P, \tilde{P}_1)$ of category consists of

- a presentation of category P,
- a set $\tilde{P}_1 \subseteq P_1$ of *equational generators*.

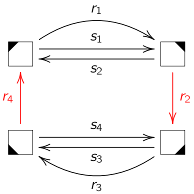
$$
\begin{tikzcd}
 & x_1 \arrow[dl, "f_1"'] \arrow[dr, "f_4"] \arrow[dd, "g_1"'] \arrow[dd, phantom, shift left=6ex, "\overset{\rho}{\Longrightarrow}"] & \\
x_2 \arrow[dr, "f_2"'] & & x_4 \arrow[dl, "f_3"] \arrow[loop right, "g_2"] \\
 & x_3 \arrow[uu, bend left=90, looseness=2, "g_3"] &
\end{tikzcd}
\qquad
\begin{aligned}
P_0 &= \{x_i\} \\
P_1 &= \{f_i, g_i\} \\
\tilde{P}_1 &= \{f_i\} \\
P_2 &= \{\rho\}
\end{aligned}
$$

Since, we want to consider objects modulo relations in $\tilde{P}_1$, it is natural to suppose that

Assumption

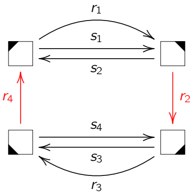
The abstract rewriting system $(P_0, \tilde{P}_1)$ is convergent.

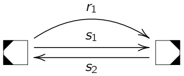
The category presented modulo



Given a presentation modulo $(P, \tilde{P}_1)$, we have three possible ways of defining the presented category from $\|P\|$:

The category presented modulo

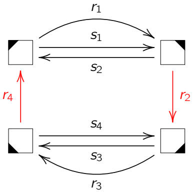




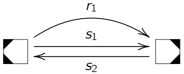
Given a presentation modulo $(P, \tilde{P}_1)$, we have three possible ways of defining the presented category from $\|P\|$:

1. **quotient** by equational generators: turn them into identities,

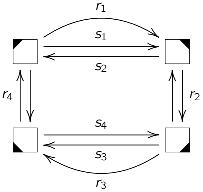
The category presented modulo



Given a presentation modulo $(P, \tilde{P}_1)$, we have three possible ways of defining the presented category from $\|P\|$:

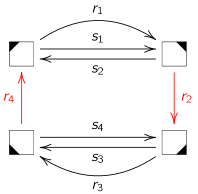


1. **quotient** by equational generators: turn them into identities,

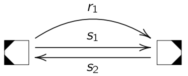


2. **localize** by equational generators: turn them into isomorphisms,

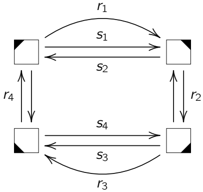
The category presented modulo



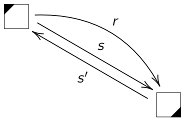
Given a presentation modulo $(P, \tilde{P}_1)$, we have three possible ways of defining the presented category from $\|P\|$:



1. **quotient** by equational generators: turn them into identities,



2. **localize** by equational generators: turn them into isomorphisms,



3. **restrict** to objects which are normal forms wrt equational generators.

The main result

Theorem

Given a presentation modulo $(P, \tilde{P}_1)$ satisfying suitable assumptions, the three constructions are related by

$$\begin{tikzcd}
\|P\|{\downarrow}\tilde{P}_1 \arrow[rr, "iso"] \arrow[dr, "equiv"'] & & \|P\|/\tilde{P}_1 \arrow[dl, "equiv"] \\
& P[\tilde{P}_1{}^{-1}] &
\end{tikzcd}$$

normal forms — *quotient* — *localization*

Theorem

Given a presentation modulo $(P, \tilde{P}_1)$ satisfying suitable assumptions, the three constructions are related by

$$
\begin{tikzcd}
\text{normal forms} & \|P\|{\downarrow}\tilde{P}_1 \arrow[rr, "iso"] \arrow[dr, "equiv"'] & & \|P\|/\tilde{P}_1 \arrow[dl, "equiv"] & \text{quotient} \\
& \text{localization} & P[\tilde{P}_1{}^{-1}] & &
\end{tikzcd}
$$

Remark

This generalizes Ore-Dehornoy conditions ensuring that a (presented) category embeds into its enveloping groupoid.

Remarks

- One of these assumptions is a variant of the cube property

Remarks

- One of these assumptions is a variant of the cube property
- We need the notion of residual for rewriting paths

Remarks

- One of these assumptions is a variant of the cube property
- We need the notion of residual for rewriting paths
- The motivations (and tools) are the same as for defining the **category of components** (Goubault, Haucourt, ...):

$$
\begin{tikzcd}
x_{04} \arrow[r, "P_a"] & x_{14} \arrow[r, "P_b"] & x_{24} \arrow[r, "V_b"] & x_{34} \arrow[r, "V_a"] & x_{44} \\
x_{03} \arrow[u, "V_b"] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{13} \arrow[u] & & x_{33} \arrow[u] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{43} \arrow[u, "V_b"'] \\
x_{02} \arrow[u, "V_a"] & & & & x_{42} \arrow[u, "V_a"'] \\
x_{01} \arrow[u, "P_a"] \arrow[r] & x_{11} & & x_{31} \arrow[r] & x_{41} \arrow[u, "P_a"'] \\
x_{00} \arrow[u, "P_b"] \arrow[r, "P_a"'] \arrow[ur, phantom, "\diamond"] & x_{10} \arrow[u] \arrow[r, "P_b"'] & x_{20} \arrow[r, "V_b"'] & x_{30} \arrow[u] \arrow[r, "V_a"'] \arrow[ur, phantom, "\diamond"] & x_{40} \arrow[u, "P_b"']
\end{tikzcd}
\rightsquigarrow
\begin{tikzcd}
x_{04} \arrow[rrr] & & & x_{34} \arrow[r, "V_a"] & x_{44} \\
& & & x_{33} \arrow[u] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{43} \arrow[u, "V_b"'] \\
& & & & \\
x_{01} \arrow[uuu] \arrow[r] & x_{11} & & & \\
x_{00} \arrow[u, "P_b"] \arrow[r, "P_a"'] \arrow[ur, phantom, "\diamond"] & x_{10} \arrow[u] \arrow[rrr] & & & x_{40} \arrow[uuu]
\end{tikzcd}
$$

Remarks

- One of these assumptions is a variant of the cube property
- We need the notion of residual for rewriting paths
- The motivations (and tools) are the same as for defining the **category of components** (Goubault, Haucourt, ...):

$$
\begin{tikzcd}
x_{04} \arrow[r, "P_a"] & x_{14} \arrow[r, "P_b"] & x_{24} \arrow[r, "V_b"] & x_{34} \arrow[r, "V_a"] & x_{44} \\
x_{03} \arrow[u, "V_b"] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{13} \arrow[u] & & x_{33} \arrow[u] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{43} \arrow[u, "V_b"'] \\
x_{02} \arrow[u, "V_a"] & & & & x_{42} \arrow[u, "V_a"'] \\
x_{01} \arrow[u, "P_a"] \arrow[r] & x_{11} & & x_{31} \arrow[r] & x_{41} \arrow[u, "P_a"'] \\
x_{00} \arrow[u, "P_b"] \arrow[r, "P_a"'] \arrow[ur, phantom, "\diamond"] & x_{10} \arrow[u] \arrow[r, "P_b"'] & x_{20} \arrow[r, "V_b"'] & x_{30} \arrow[u] \arrow[r, "V_a"'] \arrow[ur, phantom, "\diamond"] & x_{40} \arrow[u, "P_b"']
\end{tikzcd}
\quad \rightsquigarrow \quad
\begin{tikzcd}
x_{04} \arrow[rrr] & & & x_{34} \arrow[r, "V_a"] & x_{44} \\
& & & x_{33} \arrow[u] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{43} \arrow[u, "V_b"'] \\
& & & & \\
x_{01} \arrow[uuu] \arrow[r] & x_{11} & & & \\
x_{00} \arrow[u, "P_b"] \arrow[r, "P_a"'] \arrow[ur, phantom, "\diamond"] & x_{10} \arrow[u] \arrow[rrr] & & & x_{40} \arrow[uuu]
\end{tikzcd}
$$

- . . . which is itself closely related to **partial order reduction**.

In dimension 2

Presentations of 1-categories is a toy case, this has since then been extended to monoidal categories (= 2-categories) to obtain a presentation of

$$\triangle \times \triangle$$

as a product of monoidal categories (with similar properties).

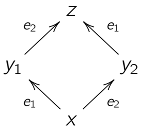
VII

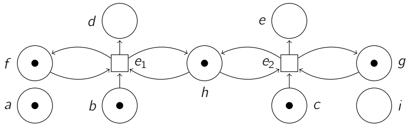
CONCLUSION

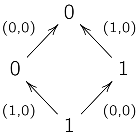
Omitted work

There is more work in the manuscript:

- adj. between cubical models and classical ones (Winskel)







Omitted work

There is more work in the manuscript:

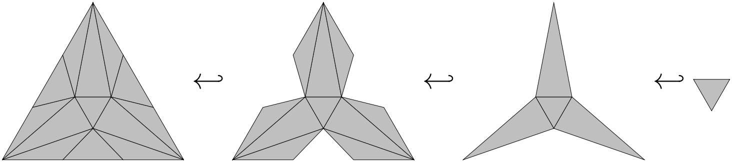
- adj. between cubical models and classical ones (Winskel)
- relating partial order reduction and the cat. of components

$$
\begin{tikzcd}
x_{04} \arrow[r, "P_a"] & x_{14} \arrow[r, "P_b"] & x_{24} \arrow[r, "V_b"] & x_{34} \arrow[r, "V_a"] & x_{44} \\
x_{03} \arrow[u, "V_b"] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{13} \arrow[u] & & x_{33} \arrow[u] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{43} \arrow[u, "V_b"'] \\
x_{02} \arrow[u, "V_a"] & & & & x_{42} \arrow[u, "V_a"'] \\
x_{01} \arrow[u, "P_a"] \arrow[r] & x_{11} & & x_{31} \arrow[r] & x_{41} \arrow[u, "P_a"'] \\
x_{00} \arrow[u, "P_b"] \arrow[r, "P_a"'] \arrow[ur, phantom, "\diamond"] & x_{10} \arrow[u] \arrow[r, "P_b"'] & x_{20} \arrow[r, "V_b"'] & x_{30} \arrow[u] \arrow[r, "V_a"'] \arrow[ur, phantom, "\diamond"] & x_{40} \arrow[u, "P_b"']
\end{tikzcd}
\quad \rightsquigarrow \quad
\begin{tikzcd}
x_{04} \arrow[rrr] & & & x_{34} \arrow[r, "V_a"] & x_{44} \\
& & & x_{33} \arrow[u] \arrow[r] \arrow[ur, phantom, "\diamond"] & x_{43} \arrow[u, "V_b"'] \\
& & & & \\
x_{01} \arrow[uuu] \arrow[r] & x_{11} & & & \\
x_{00} \arrow[u, "P_b"] \arrow[r, "P_a"'] \arrow[ur, phantom, "\diamond"] & x_{10} \arrow[u] \arrow[rrr] & & & x_{40} \arrow[uuu]
\end{tikzcd}
$$

Omitted work

There is more work in the manuscript:

- adj. between cubical models and classical ones (Winskel)
- relating partial order reduction and the cat. of components
- geometric models and asynchronous computability



Omitted work

There is more work in the manuscript:

- adj. between cubical models and classical ones (Winskel)
- relating partial order reduction and the cat. of components
- geometric models and asynchronous computability
- homotopical completion of rewriting systems

$$
\begin{tikzcd}[column sep=small]
 & aa \\
sta \arrow[ur, Rightarrow, "\beta a"] \arrow[dr, Rightarrow, "s\alpha"'] \arrow[r, phantom, "\Downarrow A"] & \\
 & sas \arrow[uu, Rightarrow, bend right, "\gamma"']
\end{tikzcd}
\quad
\begin{tikzcd}[column sep=small]
 & aat \\
sast \arrow[ur, Rightarrow, "\gamma t"] \arrow[dr, Rightarrow, "sa\beta"'] \arrow[r, phantom, "\Downarrow B"] & \\
 & saa \arrow[uu, Rightarrow, bend right, "\delta"']
\end{tikzcd}
\quad
\begin{tikzcd}[column sep=small]
 & aaas & \\
sasas \arrow[ur, Rightarrow, "\gamma as"] \arrow[dr, Rightarrow, "sa\gamma"'] \arrow[r, phantom, "\Downarrow C"] & & aata \arrow[ul, Rightarrow, "aa\alpha"'] \\
 & saaa \arrow[ur, Rightarrow, "\delta a"'] &
\end{tikzcd}
\quad
\begin{tikzcd}[column sep=small]
 & aaaa & aaast \arrow[l, Rightarrow, "aaa\beta"'] \\
sasaa \arrow[ur, Rightarrow, "\gamma aa"] \arrow[dr, Rightarrow, "sa\delta"'] \arrow[r, phantom, "\Downarrow D"] & & \\
 & saaat \arrow[r, Rightarrow, "\delta at"'] & aatat \arrow[uu, Rightarrow, "aa\alpha t"']
\end{tikzcd}
$$

Omitted work

There is more work in the manuscript:

- adj. between cubical models and classical ones (Winskel)
- relating partial order reduction and the cat. of components
- geometric models and asynchronous computability
- homotopical completion of rewriting systems
- preliminary implementation of polygraphs

$$
\begin{tikzcd}
 & x_0 \arrow[r, bend left=60, "f_0"] & y_1 \arrow[r, "g_4"] & y_2 \arrow[r, "g_5"] & y_3 & \\
x_0 \arrow[ur, dotted] \arrow[dr, dotted] & x_0 \arrow[r, bend left=60, "f_0"] \arrow[r, bend left=30, phantom, "\Downarrow\alpha_0"] \arrow[r, "f_1" description] \arrow[r, bend right=30, phantom, "\Downarrow\alpha_1"] \arrow[r, bend right=60, "f_2"'] & y_1 \arrow[r, "g_4"] & y_2 \arrow[r, "g_5"] & y_3 & y_3 \arrow[ul, dotted] \arrow[dl, dotted] \\
 & x_0 \arrow[r, bend right=60, "f_2"'] & y_1 \arrow[r, "g_4"'] & y_2 \arrow[r, "g_5"'] & y_3 &
\end{tikzcd}
$$

Unifying viewpoints and techniques

$$
\begin{tikzcd}
 & \text{Computation} \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\text{Algebra} \arrow[rr, leftrightarrow] & & \text{Geometry}
\end{tikzcd}
$$

Perspectives:

- higher-dimensional categories and rewriting:
  - computational properties
  - homotopical properties
  - presentations of weak ∞-categories
  - directed homotopy equivalence
  - ...
- geometric invariants for asynchronous computations
  - first links with geom. sem. and partially commutative monoids
  - more invariants (Squier?)
  - ...

QUESTIONS?