DIHOMOTOPY AND THE CUBE PROPERTY

SAMUEL MIMRAM
École Polytechnique

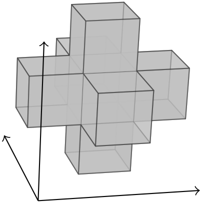
joint work with
ÉRIC GOUBAULT

GETCO conference

April 10, 2015

General idea

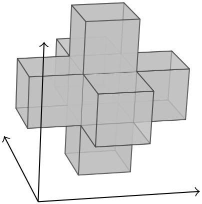
Concurrent programs can be interpreted as directed spaces, but methods in algebraic topology have been devised for non-directed spaces.



Dihomotopy and homotopy coincide for common programs!

General idea

Concurrent programs can be interpreted as directed spaces, but methods in algebraic topology have been devised for non-directed spaces.



Dihomotopy and homotopy coincide for common programs!

Here, I will focus on some algebraic and topological aspects.

PART I

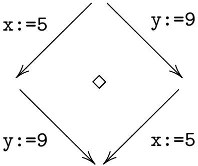
CUBICAL SEMANTICS
OF
CONCURRENT PROGRAMS

Commutation of actions concurrent programs

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

$$\texttt{x := 5} \quad \| \quad \texttt{y := 9}$$

in the sense that their order do not matter

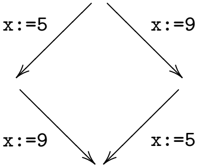


Commutation of actions concurrent programs

In concurrent programs, some actions do *not* **commute**

$$\texttt{x := 5} \quad \| \quad \texttt{x := 9}$$

in the sense that their order *does* matter



In fact, the resulting `x` could even be different from 5 and 9!

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.

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

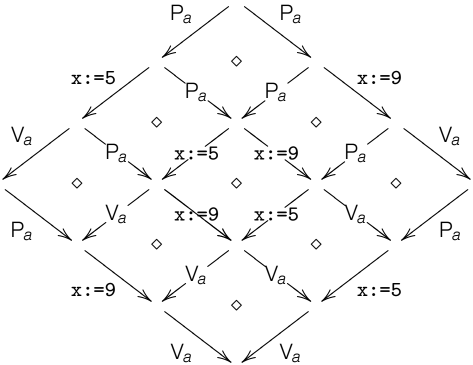
$$\mathrm{P}_a \,;\, \texttt{x:=5} \,;\, \mathrm{V}_a \qquad \| \qquad \mathrm{P}_a \,;\, \texttt{x:=9} \,;\, \mathrm{V}_a$$

Mutexes

Our earlier program should be rewritten as

$$\mathsf{P}_a\,;\,\texttt{x:=5}\,;\,\mathsf{V}_a \qquad \| \qquad \mathsf{P}_a\,;\,\texttt{x:=9}\,;\,\mathsf{V}_a$$

Possible executions are

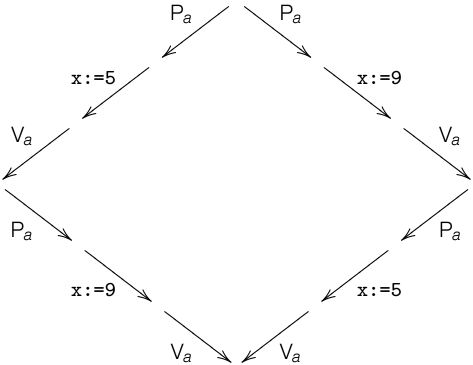


Mutexes

Our earlier program should be rewritten as

$$\mathsf{P}_a\,;\texttt{x:=5}\,;\mathsf{V}_a \qquad \| \qquad \mathsf{P}_a\,;\texttt{x:=9}\,;\mathsf{V}_a$$

Possible executions are



Concurrent programs

We consider **concurrent programs** defined by

$$p \quad ::= \quad A \quad | \quad p\,;p \quad | \quad p{+}p \quad | \quad p \parallel p \quad | \quad p^* \quad | \quad \mathsf{P}_a \quad | \quad \mathsf{V}_a$$

where

A	an *action* (e.g. `x:=5`)
$p\,;q$	do p then q
$p+q$	do p or q (`if` / `then` / `else`)
p^*	repeat p (`while`)
P_a	take mutex a
V_a	release mutex a

A **cubical graph** C consists of

- a set C_0 of *vertices*
- a set C_1 of *edges*
- source and target maps $\partial_0^-, \partial_0^+ : C_1 \to C_0$
- a set C_2 of *squares*
- source and target maps $\partial_0^-, \partial_0^+, \partial_1^-, \partial_1^+ : C_2 \to C_1$
- a transposition $\tau : C_2 \to C_2$

satisfying axioms so that

$$\begin{tikzcd}
 & {} \arrow[dl, "\partial_0^-(\alpha)"'] \arrow[dr, "\partial_1^-(\alpha)"] & \\
{} \arrow[dr, "\partial_1^+(\alpha)"'] & \alpha & {} \arrow[dl, "\partial_0^+(\alpha)"] \\
 & {} &
\end{tikzcd}$$

Cubical graphs

A **cubical graph** C consists of

- a set C_0 of *vertices*
- a set C_1 of *edges*
- source and target maps $\partial_0^-, \partial_0^+ : C_1 \to C_0$
- a set C_2 of *squares*
- source and target maps $\partial_0^-, \partial_0^+, \partial_1^-, \partial_1^+ : C_2 \to C_1$
- a transposition $\tau : C_2 \to C_2$

satisfying axioms so that

$$
\begin{tikzcd}
 & {} \arrow[dl, "\partial_0^-(\alpha)"'] \arrow[dr, "\partial_1^-(\alpha)"] & \\
{} \arrow[dr, "\partial_1^+(\alpha)"'] \arrow[rr, phantom, "\alpha"] & & {} \arrow[dl, "\partial_0^+(\alpha)"] \\
 & {} &
\end{tikzcd}
$$

We sometimes add **labels** on edges.

Squares

We write

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

or

$$A \cdot B \diamond B' \cdot A'$$

to indicate that there exists a square α with

$$\partial_0^-(\alpha) = A \qquad \partial_1^+(\alpha) = B \qquad \dots$$

Cubical graph associated to a program

To every every program p we can associate a cubical graph C_p, together with *beginning* vertex b_p and *end* vertex e_p, by induction:

- A:

$$C_A \quad = \quad b_A \bullet \xrightarrow{A} \bullet\, e_A$$

Cubical graph associated to a program

To every every program p we can associate a cubical graph C_p, together with *beginning* vertex b_p and *end* vertex e_p, by induction:

- A:

$$C_A \quad = \quad b_A \bullet \xrightarrow{A} \bullet\, e_A$$

- P_a:

$$C_{\mathrm{P}_a} \quad = \quad b_{\mathrm{P}_a} \bullet \xrightarrow{\mathrm{P}_a} \bullet\, e_{\mathrm{P}_a}$$

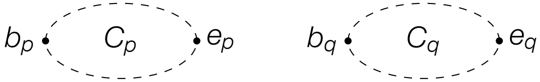
- V_a:

$$C_{\mathrm{V}_a} \quad = \quad b_{\mathrm{V}_a} \bullet \xrightarrow{\mathrm{V}_a} \bullet\, e_{\mathrm{V}_a}$$

Cubical graph associated to a program

To every every program p we can associate a cubical graph C_p, together with *beginning* vertex b_p and *end* vertex e_p, by induction:

- $p\,;q$:

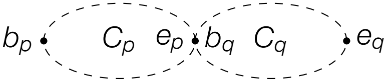


Cubical graph associated to a program

To every every program p we can associate a cubical graph C_p, together with *beginning* vertex b_p and *end* vertex e_p, by induction:

- $p\,;q$:

$$C_{p\,;\,q} \quad =$$

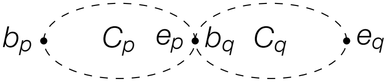


Cubical graph associated to a program

To every every program p we can associate a cubical graph C_p, together with *beginning* vertex b_p and *end* vertex e_p, by induction:

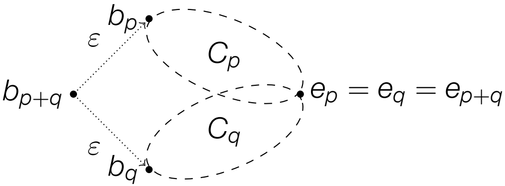
- $p\,;q$:

$C_{p\,;\,q} \quad =$



- $p+q$:

$C_{p+q} \quad =$

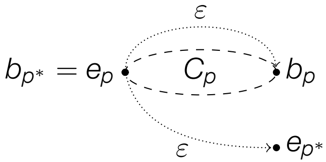


Cubical graph associated to a program

To every every program p we can associate a cubical graph C_p, together with *beginning* vertex b_p and *end* vertex e_p, by induction:

- p^*:

$p^* \quad =$

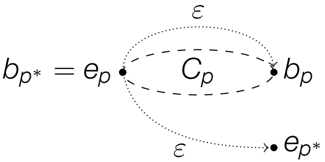


Cubical graph associated to a program

To every every program p we can associate a cubical graph C_p, together with *beginning* vertex b_p and *end* vertex e_p, by induction:

- p^*:

$$p^* \quad =$$



- $p \parallel q$:

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

Tensor product of cubical graphs

The **tensor product** $C \otimes D$ of two cubical graphs C and D has

- vertices: $(C \otimes D)_0 = C_0 \times D_0$
- edges: $(C \otimes D)_1 = (C_1 \times D_0) \sqcup (C_0 \times D_1)$
- squares are of the form

$$\begin{tikzcd}
 & (x,y) \arrow[dl, "{(f,y)}"'] \arrow[dr, "{(x,g)}"] & \\
(x',y) \arrow[dr, "{(x',g)}"'] \arrow[rr, phantom, "\alpha"] & & (x,y') \arrow[dl, "{(f,y')}"] \\
 & (x',y') &
\end{tikzcd}$$

for $f : x \to x'$ in C and $g : y \to y'$ in D.

Tensor product of cubical graphs

For instance:

$$\xrightarrow{P_a}\ \xrightarrow{\mathtt{x:=5}}\ \xrightarrow{V_a} \qquad \otimes \qquad \xrightarrow{P_a}\ \xrightarrow{\mathtt{x:=9}}\ \xrightarrow{V_a}$$

$$=$$

$$\begin{tikzcd}
 & & & {} \arrow[dl, "P_a"'] \arrow[dr, "P_a"] \arrow[dd, phantom, "\diamond"] & & & \\
 & & {} \arrow[dl, "\mathtt{x:=5}"'] \arrow[dr, "P_a"] \arrow[dd, phantom, "\diamond"] & & {} \arrow[dl, "P_a"'] \arrow[dr, "\mathtt{x:=9}"] \arrow[dd, phantom, "\diamond"] & & \\
 & {} \arrow[dl, "V_a"'] \arrow[dr, "P_a"] \arrow[dd, phantom, "\diamond"] & & {} \arrow[dl, "\mathtt{x:=5}"'] \arrow[dr, "\mathtt{x:=9}"] \arrow[dd, phantom, "\diamond"] & & {} \arrow[dl, "P_a"'] \arrow[dr, "V_a"] \arrow[dd, phantom, "\diamond"] & \\
{} \arrow[dr, "P_a"'] & & {} \arrow[dl, "V_a"] \arrow[dr, "\mathtt{x:=9}"] \arrow[dd, phantom, "\diamond"] & & {} \arrow[dl, "\mathtt{x:=5}"'] \arrow[dr, "V_a"] \arrow[dd, phantom, "\diamond"] & & {} \arrow[dl, "P_a"] \\
 & {} \arrow[dr, "\mathtt{x:=9}"'] & & {} \arrow[dl, "V_a"] \arrow[dr, "V_a"] \arrow[dd, phantom, "\diamond"] & & {} \arrow[dl, "\mathtt{x:=5}"] & \\
 & & {} \arrow[dr, "V_a"'] & & {} \arrow[dl, "V_a"] & & \\
 & & & {} & & &
\end{tikzcd}$$

Definition

The **cubical semantics** $\check{C}_p$ of a program p is the cubical graph obtained from C_p by removing vertices (as well as adjacent vertices and squares) which are **forbidden** because some resource is taken more than once.

Definition

The **cubical semantics** $\check{C}_p$ of a program p is the cubical graph obtained from C_p by removing vertices (as well as adjacent vertices and squares) which are **forbidden** because some resource is taken more than once.

Remark

This supposes that the resource consumption is unambiguously defined for a vertex. A program for which this is the case is called *conservative*, e.g. not

$$\begin{tikzcd}
& {} \arrow[dl, "\mathsf{P}_a"'] \arrow[dr, "\mathsf{P}_a"] & \\
{} \arrow[dr, "\mathsf{P}_a"'] & & {} \arrow[dl, "\mathsf{P}_b"] \\
& ? &
\end{tikzcd}$$

Proposition

Paths in $\check{C}_p$ starting from b_p are in bijection with executions of the program p.

$$\check{C}_{(\mathsf{P}_a\,;\,\texttt{x:=5}\,;\,\mathsf{V}_a)\,\|\,(\mathsf{P}_a\,;\,\texttt{x:=9}\,;\,\mathsf{V}_a)}$$
$$=$$
$$\begin{tikzcd}
 & & & {} \arrow[dl, "\mathsf{P}_a"'] \arrow[dr, "\mathsf{P}_a"] & & & \\
 & & {} \arrow[dl, "\texttt{x:=5}"'] & & {} \arrow[dr, "\texttt{x:=9}"] & & \\
 & {} \arrow[dl, "\mathsf{V}_a"'] & & & & {} \arrow[dr, "\mathsf{V}_a"] & \\
{} \arrow[dr, "\mathsf{P}_a"'] & & & & & & {} \arrow[dl, "\mathsf{P}_a"] \\
 & {} \arrow[dr, "\texttt{x:=9}"'] & & & & {} \arrow[dl, "\texttt{x:=5}"] & \\
 & & {} \arrow[dr, "\mathsf{V}_a"'] & & {} \arrow[dl, "\mathsf{V}_a"] & & \\
 & & & {} & & &
\end{tikzcd}$$

Homotopy between paths

Definition

The **homotopy** relation $\sim$ between paths is the smallest congruence such that $A \cdot B \sim B' \cdot A'$ whenever $A \cdot B \diamond B' \cdot A'$:

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

Homotopy between paths

Definition

The **homotopy** relation $\sim$ between paths is the smallest congruence such that $A \cdot B \sim B' \cdot A'$ whenever $A \cdot B \diamond B' \cdot A'$:

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

Proposition

For "reasonable" programs, two homotopic executions lead to the same state.

Homotopy between paths

Definition

The **homotopy** relation $\sim$ between paths is the smallest congruence such that $A \cdot B \sim B' \cdot A'$ whenever $A \cdot B \diamond B' \cdot A'$:

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

Proposition

For "reasonable" programs, two homotopic executions lead to the same state.

It seems interesting to study the space of paths up to homotopy.

PART II

HOMOTOPY
VS
DIHOMOTOPY

Path direction

In classical topology paths are not *directed*: given a path $p : I \to X$ we also have a reverse path $\overline{p} : I \to X$ defined by

$$\overline{p}(t) \quad = \quad p(1 - t)$$

and most constructions in algebraic topology depend on this (the fundamental *group*, etc.)

Path direction

In classical topology paths are not *directed*: given a path $p : I \to X$ we also have a reverse path $\overline{p} : I \to X$ defined by

$$\overline{p}(t) \quad = \quad p(1-t)$$

and most constructions in algebraic topology depend on this (the fundamental *group*, etc.)

On the contrary our paths must follow the directions indicated by arrows.

How can we compare the two?

Dipaths

We call a **dipath** what we have been calling a path, i.e. a sequence of composable arrows:

$$\xrightarrow{A} \; \xrightarrow{B} \; \xrightarrow{C} \qquad \text{or} \qquad A \cdot B \cdot C$$

Dipaths

We call a **dipath** what we have been calling a path, i.e. a sequence of composable arrows:

$$\xrightarrow{A} \; \xrightarrow{B} \; \xrightarrow{C} \qquad\qquad \text{or} \qquad\qquad A \cdot B \cdot C$$

We call a **path** a sequence of *possibly reversed* composable arrows:

$$\xrightarrow{A} \; \xleftarrow{B} \; \xrightarrow{C} \qquad\qquad \text{or} \qquad\qquad A \cdot \overline{B} \cdot C$$

We call **dihomotopy** between paths, the smallest congruence $\leftrightsquigarrow$ such that for every square

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

we have

$$A \cdot B \leftrightsquigarrow B' \cdot A' \qquad \overline{A} \cdot B' \leftrightsquigarrow B \cdot \overline{A'} \qquad \overline{B} \cdot \overline{A} \leftrightsquigarrow \overline{A'} \cdot \overline{B'}$$

We call **dihomotopy** between paths, the smallest congruence $\leftrightsquigarrow$ such that for every square

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

we have

$$A \cdot B \leftrightsquigarrow B' \cdot A' \qquad \overline{A} \cdot B' \leftrightsquigarrow B \cdot \overline{A'} \qquad \overline{B} \cdot \overline{A} \leftrightsquigarrow \overline{A'} \cdot \overline{B'}$$

Remark

A path dihomotopic to a dipath is necessarily a dipath.

Homotopy

The **homotopy** relation on paths $\sim$ is the smallest congruence containing dihomotopy and such that for every edge

$$x \xrightarrow{A} y$$

we have

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

Homotopy

The **homotopy** relation on paths $\sim$ is the smallest congruence containing dihomotopy and such that for every edge

$$x \xrightarrow{A} y$$

we have

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

Remark

Clearly $f \leftrightsquigarrow g$ implies $f \sim g$, but converse is *not* generally true.

Homotopy vs dihomotopy

Consider the following “matchbox”:

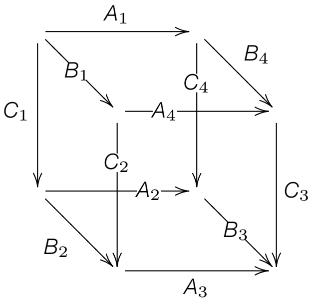
$$
\begin{tikzcd}
\bullet \arrow[rr, "A_1"] \arrow[dr, "B_1"] \arrow[dd, "C_1"'] & & \bullet \arrow[dr, "B_4"] \arrow[dd, "C_4"] & \\
& \bullet \arrow[rr, "A_4"] \arrow[dd, "C_2"] & & \bullet \arrow[dd, "C_3"] \\
\bullet \arrow[rr, "A_2"] \arrow[dr, "B_2"'] & & \bullet \arrow[dr, "B_3"] & \\
& \bullet \arrow[rr, "A_3"'] & & \bullet
\end{tikzcd}
$$

where every square is filled excepting the top one:

$$\xcancel{A_1 \cdot B_4 \diamond B_1 \cdot A_4}$$

Homotopy vs dihomotopy

Consider the following “matchbox”:



We have

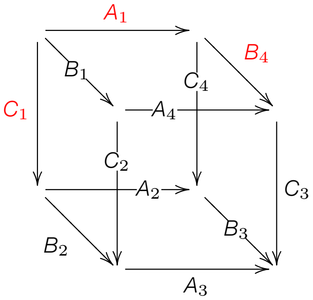
$$A_1 \cdot B_4 \sim B_1 \cdot A_4 \qquad \text{but not} \qquad A_1 \cdot B_4 \leftrightsquigarrow B_1 \cdot A_4$$

Homotopy vs dihomotopy

$$\begin{tikzcd}
\bullet \arrow[rr, "A_1"] \arrow[dr, "B_1"] \arrow[dd, "C_1"'] & & \bullet \arrow[dr, "B_4"] \arrow[dd, "C_4"] & \\
& \bullet \arrow[rr, "A_4"] \arrow[dd, "C_2"] & & \bullet \arrow[dd, "C_3"] \\
\bullet \arrow[rr, "A_2"] \arrow[dr, "B_2"'] & & \bullet \arrow[dr, "B_3"] & \\
& \bullet \arrow[rr, "A_3"'] & & \bullet
\end{tikzcd}$$

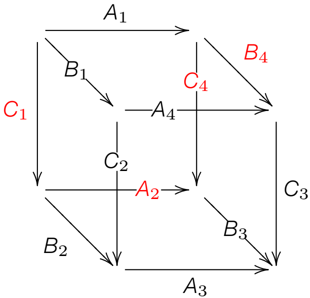
$A_1 \cdot B_4$

Homotopy vs dihomotopy



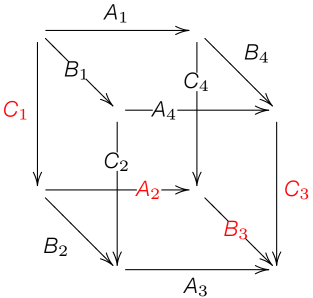
$$A_1 \cdot B_4 \quad \sim \quad C_1 \cdot \overline{C_1} \cdot A_1 \cdot B_4$$

Homotopy vs dihomotopy



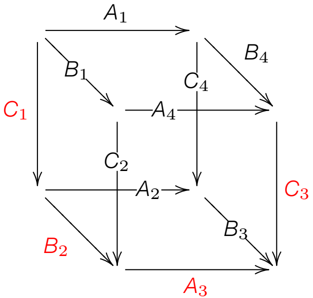
$$\begin{aligned} A_1 \cdot B_4 \quad &\sim \quad C_1 \cdot \overline{C_1} \cdot A_1 \cdot B_4 \\ &\sim \quad C_1 \cdot A_2 \cdot \overline{C_4} \cdot B_4 \end{aligned}$$

Homotopy vs dihomotopy



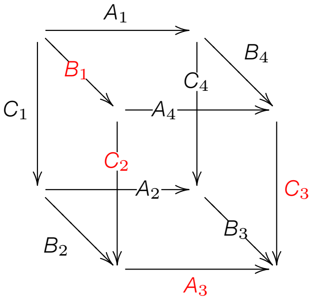
$$\begin{aligned} A_1 \cdot B_4 &\sim C_1 \cdot \overline{C_1} \cdot A_1 \cdot B_4 \\ &\sim C_1 \cdot A_2 \cdot \overline{C_4} \cdot B_4 \\ &\sim C_1 \cdot A_2 \cdot B_3 \cdot \overline{C_3} \end{aligned}$$

Homotopy vs dihomotopy



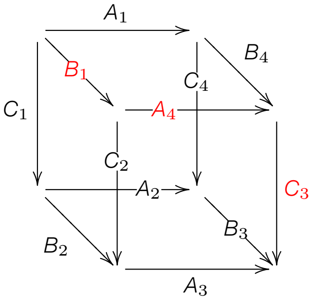
$$
\begin{aligned}
A_1 \cdot B_4 \quad &\sim \quad C_1 \cdot \overline{C_1} \cdot A_1 \cdot B_4 \\
&\sim \quad C_1 \cdot A_2 \cdot \overline{C_4} \cdot B_4 \\
&\sim \quad C_1 \cdot A_2 \cdot B_3 \cdot \overline{C_3} \\
&\sim \quad C_1 \cdot B_2 \cdot A_3 \cdot \overline{C_3}
\end{aligned}
$$

Homotopy vs dihomotopy



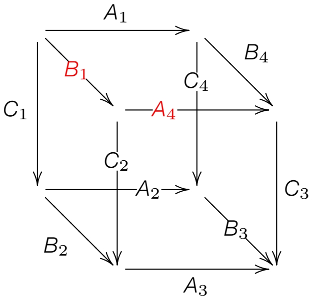
$$
\begin{aligned}
A_1 \cdot B_4 \quad & \sim \quad C_1 \cdot \overline{C_1} \cdot A_1 \cdot B_4 \\
& \sim \quad C_1 \cdot A_2 \cdot \overline{C_4} \cdot B_4 \\
& \sim \quad C_1 \cdot A_2 \cdot B_3 \cdot \overline{C_3} \\
& \sim \quad C_1 \cdot B_2 \cdot A_3 \cdot \overline{C_3} \\
& \sim \quad {\color{red}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

Homotopy vs dihomotopy



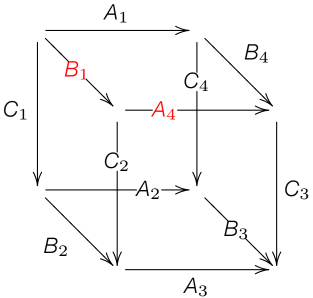
$$
\begin{aligned}
A_1 \cdot B_4 \quad & \sim \quad C_1 \cdot \overline{C_1} \cdot A_1 \cdot B_4 \\
& \sim \quad C_1 \cdot A_2 \cdot \overline{C_4} \cdot B_4 \\
& \sim \quad C_1 \cdot A_2 \cdot B_3 \cdot \overline{C_3} \\
& \sim \quad C_1 \cdot B_2 \cdot A_3 \cdot \overline{C_3} \\
& \sim \quad B_1 \cdot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

Homotopy vs dihomotopy



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

Homotopy vs dihomotopy



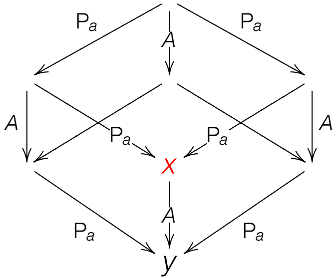
$$
\begin{aligned}
A_1 \cdot B_4 \quad & \sim \quad C_1 \cdot \overline{C_1} \cdot A_1 \cdot B_4 \\
& \sim \quad C_1 \cdot A_2 \cdot \overline{C_4} \cdot B_4 \\
& \sim \quad C_1 \cdot A_2 \cdot B_3 \cdot \overline{C_3} \\
& \sim \quad C_1 \cdot B_2 \cdot A_3 \cdot \overline{C_3} \\
& \sim \quad B_1 \cdot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

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

Binary conflicts

In a situation such as

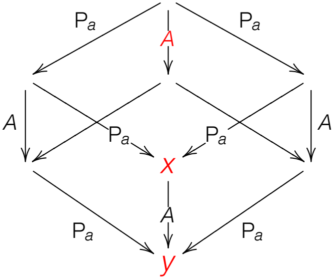
$$\mathrm{P}_a \parallel \mathrm{P}_a \parallel A \qquad =$$



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

In a situation such as

$$\mathsf{P}_a \parallel \mathsf{P}_a \parallel A \qquad =$$



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

In this case, the vertex y has to be removed too, because $A \neq \mathsf{V}_a$!

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

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

The cube property

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

$$
\begin{tikzcd}[column sep=small]
 & & {} \arrow[dll, "A"'] \arrow[d] \arrow[drr, "C'"] & & \\
{} \arrow[d, "B"'] & \diamond & {} \arrow[dll] \arrow[drr] & \diamond & {} \arrow[d, "B'"] \\
{} \arrow[drr, "C"'] & & \diamond & & {} \arrow[dll, "A'"] \\
 & & {} & &
\end{tikzcd}
\quad \Leftrightarrow \quad
\begin{tikzcd}[column sep=small]
 & & {} \arrow[dll, "A"'] \arrow[drr, "C'"] & & \\
{} \arrow[d, "B"'] \arrow[drr] & & \diamond & & {} \arrow[d, "B'"] \arrow[dll] \\
{} \arrow[drr, "C"'] & \diamond & {} \arrow[d] & \diamond & {} \arrow[dll, "A'"] \\
 & & {} & &
\end{tikzcd}
$$

and other more minor properties, e.g.

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

implies $A' = A''$ and $B' = B''$.

Homotopy vs dihomotopy

Theorem
In a cubical graph satisfying the cube property, two dipaths are dihomotopic if and only if they are homotopic.

PART III

PRESENTING THE FUNDAMENTAL CATEGORY AND GROUPOID

Fundamental groupoid and category

To every cubical graph *C*, we can associate

1. a **fundamental groupoid** $\Pi_1(C)$ of vertices and paths up to homotopy,
2. a **fundamental category** $\vec{\Pi}_1(C)$ of vertices and *di*paths up to *di*homotopy.

Fundamental groupoid and category

To every cubical graph C, we can associate

1. a **fundamental groupoid** $\Pi_1(C)$ of vertices and paths up to homotopy,
2. a **fundamental category** $\vec{\Pi}_1(C)$ of vertices and *di*paths up to *di*homotopy.

Notice that previous theorem can be reformulated as

Theorem

If C satisfies the cube property, then the inclusion functor

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

is faithful.

The fundamental 2-category

In order to study the relationships between the two categories, we in introduce:

Definition

The **fundamental 2-category** $\vec{\Pi}_2(C)$ is the 2-category whose

- 0-cells are vertices of C,
- 1-cells are paths in C,
- 2-cells are generated by

$$\gamma^{A,B}_{B',A'} : A \cdot B \Rightarrow B' \cdot A' \qquad \text{whenever} \qquad \begin{tikzcd} & \cdot \arrow[dl, "A"'] \arrow[dr, "B'"] \arrow[dd, phantom, "\diamond"] & \\ \cdot \arrow[dr, "B"'] & & \cdot \arrow[dl, "A'"] \\ & \cdot & \end{tikzcd}$$

$$\eta_A : \mathrm{id}_x \Rightarrow A \cdot \overline{A} \qquad \varepsilon_A : \overline{A} \cdot A \Rightarrow \mathrm{id}_y \qquad \text{for} \qquad x \xrightarrow{A} y$$

- *quotiented by relations on 2-cells*
- horizontal composition is concatenation of paths

Towards a proof

Notice that

- two paths f, g are *homotopic* if and only if there is a 2-cell

$$\alpha \quad : \quad f \quad \Rightarrow \quad g$$

- the paths f, g are *dihomotopic* if and only if there is such a 2-cell constructed without generators η_A and ε_A:

$$\eta_A : \mathrm{id}_x \Rightarrow A \cdot \overline{A} \qquad\qquad \varepsilon_A : \overline{A} \cdot A \Rightarrow \mathrm{id}_y$$

Towards a proof

Notice that

- two paths f, g are *homotopic* if and only if there is a 2-cell

$$\alpha \quad : \quad f \quad \Rightarrow \quad g$$

- the paths f, g are *dihomotopic* if and only if there is such a 2-cell constructed without generators η_A and ε_A:

$$\eta_A : \mathrm{id}_x \Rightarrow A \cdot \overline{A} \qquad\qquad \varepsilon_A : \overline{A} \cdot A \Rightarrow \mathrm{id}_y$$

Remark
Notice that this does not depend on the relations on 2-cells.

Towards a proof

Notice that

- two paths f, g are *homotopic* if and only if there is a 2-cell

$$\alpha \quad : \quad f \quad \Rightarrow \quad g$$

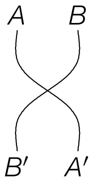
- the paths f, g are *dihomotopic* if and only if there is such a 2-cell constructed without generators η_A and ε_A:

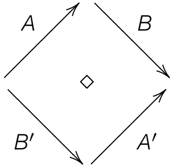
$$\eta_A : \mathrm{id}_x \Rightarrow A \cdot \overline{A} \qquad\qquad \varepsilon_A : \overline{A} \cdot A \Rightarrow \mathrm{id}_y$$

Theorem
Any 2-cell $\alpha : f \Rightarrow g$ between f and g is equal to one without the bad generators (with the right relations!).

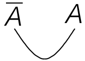
String diagrams

For the 2-cells I will use the string-diagrammatic notation:





for $\gamma^{A,B}_{B',A'}$, and

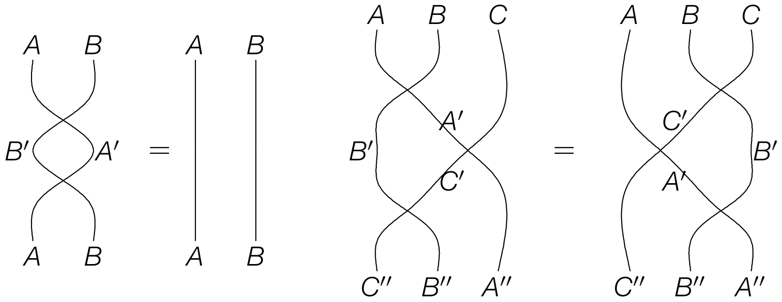


for η_A and ε_A.

Relations on 2-cells

We relations on 2-cells so that

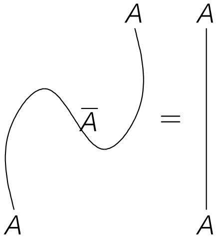
- $\gamma^{A,B}_{B',A'}$ acts like a symmetry:

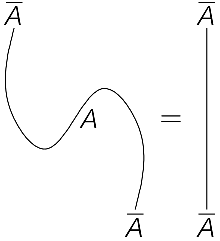


Relations on 2-cells

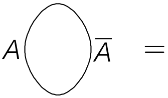
We relations on 2-cells so that

- η_A and ε_A act as (co)units of an adjunction:





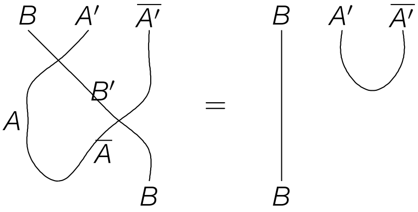
and

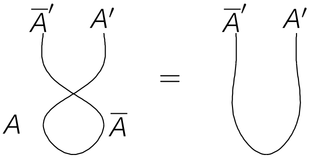


Relations on 2-cells

We relations on 2-cells so that

- the two are “naturally” compatible:

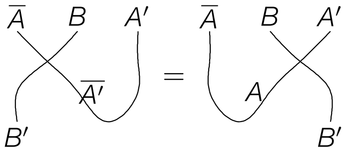


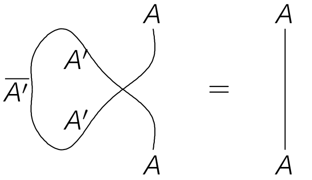


+ dual and symmetric relations

Derivable relations

Some other relations are derivable:





Well-definedness

Notice that “not every diagram makes sense”: if we cannot commute some actions for instance.

Lemma

If the left member of a relation is well-defined then the right member too.

Well-definedness

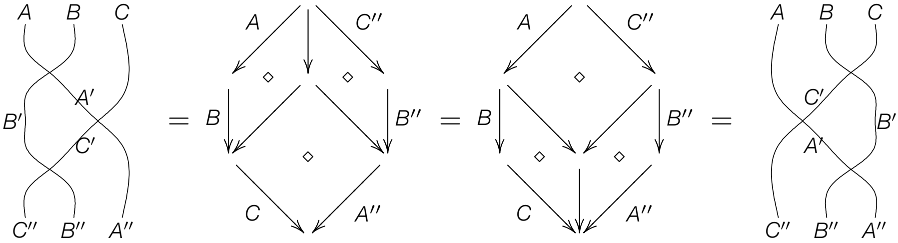
Notice that “not every diagram makes sense”: if we cannot commute some actions for instance.

Lemma

If the left member of a relation is well-defined then the right member too.

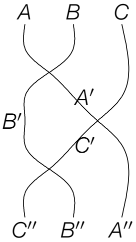
Proof.

This is where we use our properties on the cubical graph:

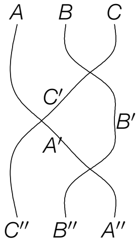


A rewriting system

We can turn our relations into a rewriting system (from left to right), e.g.



$\Rrightarrow$



Conjecture

The rewriting system is convergent, thus normal forms are canonical representatives of equivalence classes.

A proof for our theorem

Suppose given a 2-cell between dipaths $\alpha : f \Rightarrow g$. This 2-cell is equal to a normal form, so we suppose that we are in this case.

Proposition

The 2-cell α does not contain η_A or ε_A generators.

A proof for our theorem

Suppose given a 2-cell between dipaths $\alpha : f \Rightarrow g$. This 2-cell is equal to a normal form, so we suppose that we are in this case.

Proposition

The 2-cell α does not contain η_A or ε_A generators.

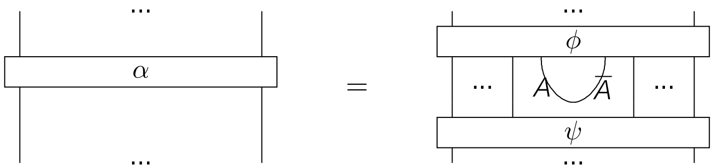
Proof.

Suppose that it "contains"

$$\varepsilon_A : \overline{A} \cdot A \Rightarrow \mathrm{id}_x$$

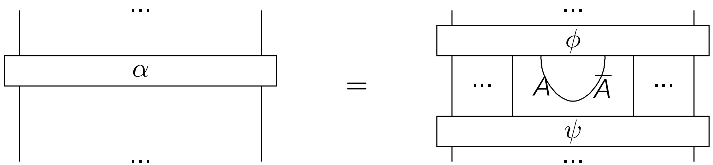
i.e.

$$\alpha \quad = \quad \psi \circ (\mathrm{id}_f \cdot \varepsilon_A \cdot \mathrm{id}_g) \circ \phi$$



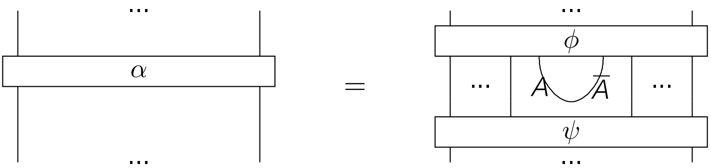
A proof for our theorem

What can ϕ be?



A proof for our theorem

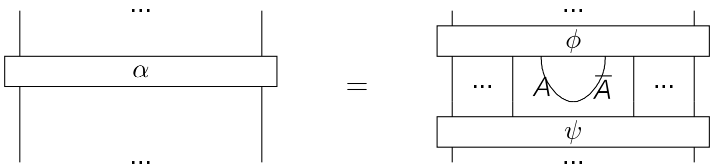
What can ϕ be?



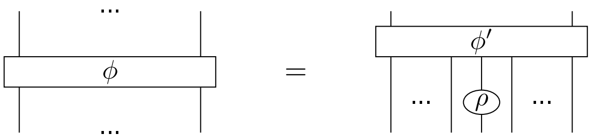
- Notice that ϕ cannot be an identity, otherwise α would contain $\overline{A}$ in its source (a reversed edge), which would not be a dipath.

A proof for our theorem

What can ϕ be?



- Notice that ϕ cannot be an identity, otherwise α would contain $\overline{A}$ in its source (a reversed edge), which would not be a dipath.
- Thus ϕ is thus of the form

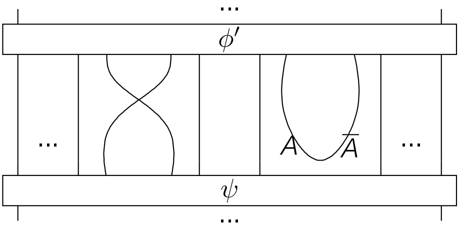


where ρ is a generator.

A proof for our theorem

We then proceed on case analysis on ρ and its position, keeping in mind that α must be in normal form. For instance, if $\rho = \gamma$,

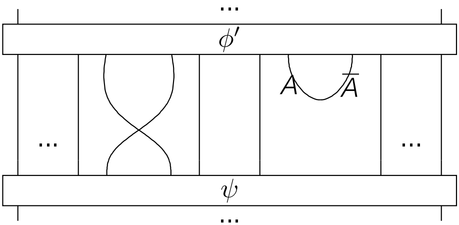
- in a case such as



we can use the exchange law to "put the γ down in the ψ" and reason by induction on ϕ'.

We then proceed on case analysis on ρ and its position, keeping in mind that α must be in normal form. For instance, if $\rho = \gamma$,

- in a case such as

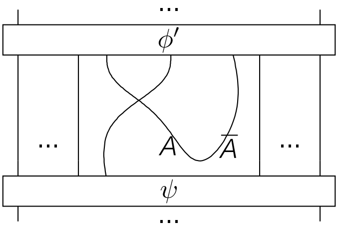


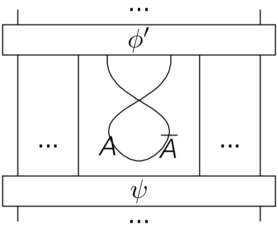
we can use the exchange law to "put the γ down in the ψ" and reason by induction on ϕ'.

A proof for our theorem

We then proceed on case analysis on ρ and its position, keeping in mind that α must be in normal form. For instance, if $\rho = \gamma$,

- the following cannot happen



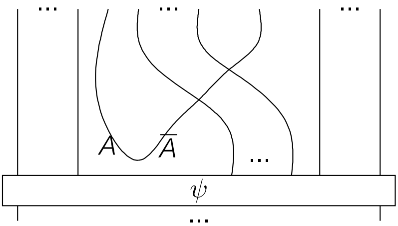


otherwise α would not be normal.

A proof for our theorem

We then proceed on case analysis on ρ and its position, keeping in mind that α must be in normal form. For instance, if $\rho = \gamma$,

- we can show that α is of the form



and thus the morphism would contain $\overline{A}$ (a reversed transition in its source). $\square$

A real proof

Showing that the rewriting system is convergent is difficult:

- there is an infinite number of critical pairs even though there is a finite number of rules (they can however be grouped in a finite number of families),
- there is an awful lot of cases to be checked.

A real proof

Showing that the rewriting system is convergent is difficult:

- there is an infinite number of critical pairs even though there is a finite number of rules (they can however be grouped in a finite number of families),
- there is an awful lot of cases to be checked.

In practice, we only need a representative (not necessarily unique), which can be defined by hand, and the proof goes on roughly as indicated before. So we actually have a proof here.

Notes on the axioms

In the category **Vect** we have bjiections

$$\frac{A \otimes B \to C}{A \to C \otimes B^*} \qquad \frac{A \to B \otimes C}{B^* \otimes A \to C}$$

Notes on the axioms

In the category **Vect** we have bjiections

$$\frac{A \otimes B \to C}{A \to C \otimes B^*} \qquad \frac{A \to B \otimes C}{B^* \otimes A \to C}$$

In particular, consider the morphisms associated to $\mathrm{id}_A : A \to A$,

$$\eta : \mathbb{k} \to A \otimes A^* \qquad \varepsilon : A^* \otimes A \to \mathbb{k}$$

Together with the symmetry $\gamma : A \otimes A \to A \otimes A$, these satisfy the axioms before, i.e. these correspond to cubical graph with one vertex, one edge and one square.

Notes on the axioms

In the category **Vect** we have bjections

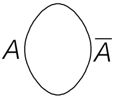
$$\frac{A \otimes B \to C}{A \to C \otimes B^*} \qquad\qquad \frac{A \to B \otimes C}{B^* \otimes A \to C}$$

In particular, consider the morphisms associated to $\mathrm{id}_A : A \to A$,

$$\eta : \Bbbk \to A \otimes A^* \qquad\qquad \varepsilon : A^* \otimes A \to \Bbbk$$

Together with the symmetry $\gamma : A \otimes A \to A \otimes A$, these satisfy the axioms before, i.e. these correspond to cubical graph with one vertex, one edge and one square.

To be precise, we also have to satisfy the axiom

$(\dim A)\,\mathrm{id}_k \quad = \quad \mathrm{tr}(\mathrm{id}_A) \quad = \quad$ 

$\quad = \qquad\qquad = \quad \mathrm{id}_k$

i.e. $\dim A = 1$.

CONCLUSION

Going further

For a cubical graph satisfying the cube property:

- universal **dicovering** has a simple definition,
- its unfolding corresponds to the configuration space of an **event structure** (Chepoi, Ardilla et al., …)
- its trace space can be computed thanks to (traditional) **homology**
- metric geometric realization is **non-positively curved** (= locally CAT(0))

Also:

- **Relations** on 2-cells are meaningful?
- Variants for n**-semaphores**, etc.
- Links with motion planning (Ghrist et al.)
- Links with geometric group theory (Dehornoy, …)