Concurrency and directed algebraic topology

Martin Raussen

Department of Mathematical Sciences
Aalborg University
Denmark

Research Seminar
Algebraic Topology and Invariant Theory
Göttingen, 27.9. – 29.9.2007

Outline

1. Motivations, mainly from Concurrency Theory
2. Directed topology: algebraic topology with a twist
3. A categorical framework (with examples)
4. “Compression” of d-topological categories: generalized congruences via homotopy flows

Main Collaborators:

- Lisbeth Fajstrup (Aalborg), Éric Goubault, Emmanuel Haucourt (CEA, France)

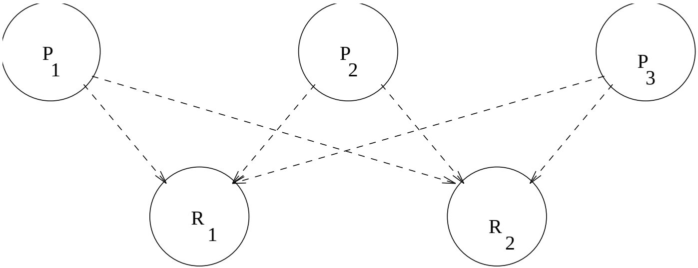
Outline

1. Motivations, mainly from Concurrency Theory
2. Directed topology: algebraic topology with a twist
3. A categorical framework (with examples)
4. “Compression” of d-topological categories: generalized congruences via homotopy flows

Main Collaborators:

- Lisbeth Fajstrup (Aalborg), Éric Goubault, Emmanuel Haucourt (CEA, France)

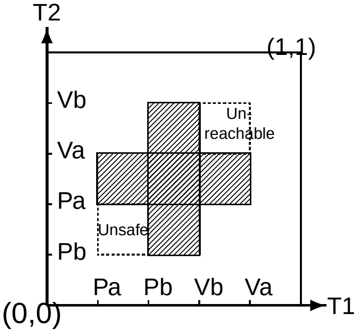
Mutual exclusion occurs, when n processes P_i compete for m resources R_j.



Only k processes can be served at any given time.
Semaphores!
Semantics: A processor has to lock a resource and relinquish the lock later on!
Description/abstraction $P_i : \dots PR_j \dots VR_j \dots$ (Dijkstra)

Schedules in "progress graphs"

The Swiss flag example



PV-diagram from

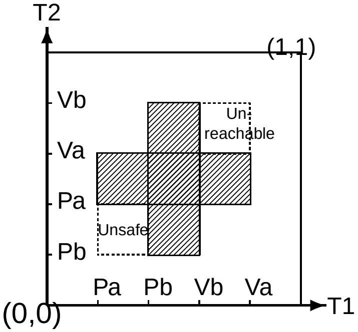
$P_1 : P_a P_b V_b V_a$

$P_2 : P_b P_a V_a V_b$

Executions are directed paths – since time flow is irreversible – avoiding a forbidden region (shaded).

Dipaths that are dihomotopic (through a 1-parameter deformation consisting of dipaths) correspond to equivalent executions.

Deadlocks, unsafe and unreachable regions may occur.



PV-diagram from

$P_1 : P_a P_b V_b V_a$

$P_2 : P_b P_a V_a V_b$

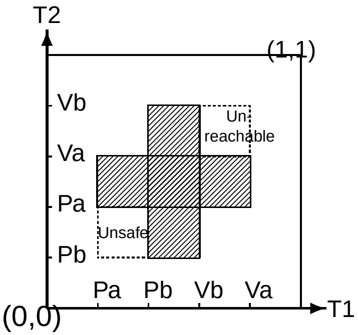
Executions are directed paths – since time flow is irreversible – avoiding a forbidden region (shaded).

Dipaths that are dihomotopic (through a 1-parameter deformation consisting of dipaths) correspond to equivalent executions.

Deadlocks, unsafe and unreachable regions may occur.

Schedules in ”progress graphs”

The Swiss flag example



PV-diagram from

$P_1 : P_a P_b V_b V_a$

$P_2 : P_b P_a V_a V_b$

Executions are directed paths – since time flow is irreversible – avoiding a forbidden region (shaded).

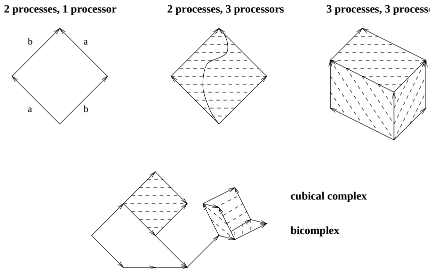
Dipaths that are dihomotopic (through a 1-parameter deformation consisting of dipaths) correspond to equivalent executions.

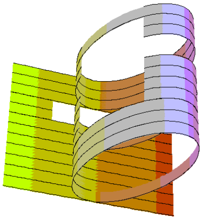
Deadlocks, unsafe and unreachable regions may occur.

Higher dimensional automata

seen as (geometric realizations of) cubical sets

Vaughan Pratt, Rob van Glabbeek, Eric Goubault...



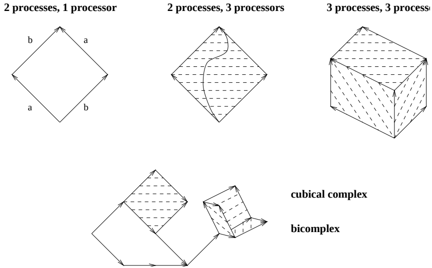


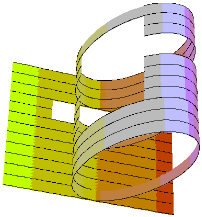
Squares/cubes/hypercubes are filled in iff actions on boundary are independent.

Higher dimensional automata are (pre)-cubical sets:

- like simplicial sets, but modelled on (hyper)cubes instead of simplices; glueing by face maps
- additionally: preferred directions – not all paths allowable.

Vaughan Pratt, Rob van Glabbeek, Eric Goubault...





Squares/cubes/hypercubes are filled in iff actions on boundary are independent.
Higher dimensional automata are (pre)-cubical sets:

- like simplicial sets, but modelled on (hyper)cubes instead of simplices; glueing by face maps
- additionally: preferred directions – not all paths allowable.

The state space explosion problem for discrete models for concurrency (transition graph models): The number of states (and the number of possible schedules) grows exponentially in the number of processors and/or the length of programs.

Need clever ways to find out which of the schedules yield equivalent results for general reasons – e.g., to check for correctness.

Alternative: Infinite continuous models allowing for well-known equivalence relations on paths (homotopy = 1-parameter deformations) – but with an important twist!

Analogy: Continuous physics as an approximation to (discrete) quantum physics.

Discrete versus continuous models

How to handle the state-space explosion problem?

The state space explosion problem for discrete models for concurrency (transition graph models): The number of states (and the number of possible schedules) grows exponentially in the number of processors and/or the length of programs. Need clever ways to find out which of the schedules yield equivalent results for general reasons – e.g., to check for correctness.

Alternative: Infinite continuous models allowing for well-known equivalence relations on paths (homotopy = 1-parameter deformations) – but with an important twist!

Analogy: Continuous physics as an approximation to (discrete) quantum physics.

X a topological space. $\vec{P}(X) \subseteq X^I = \{p : I = [0,1] \to X \text{ cont.}\}$ a space of d-paths (CO-topology; "directed" paths $\leftrightarrow$ executions) satisfying

- $\{$ constant paths $\} \subseteq \vec{P}(X)$
- $\varphi \in \vec{P}(X)(x,y), \psi \in \vec{P}(X)(y,z) \Rightarrow \varphi * \psi \in \vec{P}(X)(x,z)$
- $\varphi \in \vec{P}(X), \alpha \in I^I$ a nondecreasing reparametrization $\Rightarrow \varphi \circ \alpha \in \vec{P}(X)$

The pair $(X, \vec{P}(X))$ is called a d-space.

Observe: $\vec{P}(X)$ is in general not closed under reversal:

$$\alpha(t) = 1 - t,\ \varphi \in \vec{P}(X) \not\Rightarrow \varphi \circ \alpha \in \vec{P}(X)!$$

Examples:

- An HDA with directed execution paths.
- A space-time(relativity) with time-like or causal curves.

X a topological space. $\vec{P}(X) \subseteq X^I = \{p : I = [0,1] \to X \text{ cont.}\}$ a space of d-paths (CO-topology; "directed" paths $\leftrightarrow$ executions) satisfying

- $\{$ constant paths $\} \subseteq \vec{P}(X)$
- $\varphi \in \vec{P}(X)(x,y), \psi \in \vec{P}(X)(y,z) \Rightarrow \varphi * \psi \in \vec{P}(X)(x,z)$
- $\varphi \in \vec{P}(X), \alpha \in I^I$ a nondecreasing reparametrization $\Rightarrow \varphi \circ \alpha \in \vec{P}(X)$

The pair $(X, \vec{P}(X))$ is called a d-space.
Observe: $\vec{P}(X)$ is in general not closed under reversal:

$$\alpha(t) = 1 - t,\ \varphi \in \vec{P}(X) \not\Rightarrow \varphi \circ \alpha \in \vec{P}(X)!$$

Examples:

- An HDA with directed execution paths.
- A space-time(relativity) with time-like or causal curves.

X a topological space. $\vec{P}(X) \subseteq X^I = \{p : I = [0,1] \to X \text{ cont.}\}$ a space of d-paths (CO-topology; "directed" paths $\leftrightarrow$ executions) satisfying

- $\{$ constant paths $\} \subseteq \vec{P}(X)$
- $\varphi \in \vec{P}(X)(x,y), \psi \in \vec{P}(X)(y,z) \Rightarrow \varphi * \psi \in \vec{P}(X)(x,z)$
- $\varphi \in \vec{P}(X), \alpha \in I^I$ a nondecreasing reparametrization $\Rightarrow \varphi \circ \alpha \in \vec{P}(X)$

The pair $(X, \vec{P}(X))$ is called a d-space.
Observe: $\vec{P}(X)$ is in general not closed under reversal:

$$\alpha(t) = 1 - t,\ \varphi \in \vec{P}(X) \not\Rightarrow \varphi \circ \alpha \in \vec{P}(X)!$$

Examples:

- An HDA with directed execution paths.
- A space-time(relativity) with time-like or causal curves.

D-maps, Dihomotopy, d-homotopy

A d-map $f : X \to Y$ is a continuous map satisfying

- $f(\vec{P}(X)) \subseteq \vec{P}(Y)$.

special case: $\vec{P}(I) = \{\sigma \in I^I | \sigma$ nondecreasing reparametrization$\}, \vec{I} = (I, \vec{P}(I))$.
Then $\vec{P}(X) =$ space of d-maps from $\vec{I}$ to X.

- Dihomotopy $H : X \times I \to Y$, every H_t a d-map
- elementary d-homotopy = d-map $H : X \times \vec{I} \to Y$ – $H_0 = f \xrightarrow{H} g = H_1$
- d-homotopy: symmetric and transitive closure ("zig-zag")

L. Fajstrup, 05: In cubical models (for concurrency, e.g., HDAs), the two notions agree for d-paths ($X = \vec{I}$). In general, they do not.

D-maps, Dihomotopy, d-homotopy

A d-map $f : X \to Y$ is a continuous map satisfying

- $f(\vec{P}(X)) \subseteq \vec{P}(Y)$.

special case: $\vec{P}(I) = \{\sigma \in I^I | \sigma$ nondecreasing reparametrization$\}, \vec{I} = (I, \vec{P}(I))$.
Then $\vec{P}(X) =$ space of d-maps from $\vec{I}$ to X.

- Dihomotopy $H : X \times I \to Y$, every H_t a d-map
- elementary d-homotopy = d-map $H : X \times \vec{I} \to Y$ – $H_0 = f \xrightarrow{H} g = H_1$
- d-homotopy: symmetric and transitive closure ("zig-zag")

L. Fajstrup, 05: In cubical models (for concurrency, e.g., HDAs), the two notions agree for d-paths ($X = \vec{I}$). In general, they do not.

D-maps, Dihomotopy, d-homotopy

A d-map $f : X \to Y$ is a continuous map satisfying

- $f(\vec{P}(X)) \subseteq \vec{P}(Y)$.

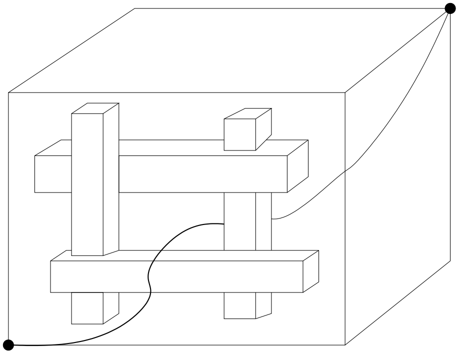
special case: $\vec{P}(I) = \{\sigma \in I^I | \sigma$ nondecreasing reparametrization$\}, \vec{I} = (I, \vec{P}(I))$.
Then $\vec{P}(X) =$ space of d-maps from $\vec{I}$ to X.

- Dihomotopy $H : X \times I \to Y$, every H_t a d-map
- elementary d-homotopy = d-map $H : X \times \vec{I} \to Y$ – $H_0 = f \xrightarrow{H} g = H_1$
- d-homotopy: symmetric and transitive closure (”zig-zag”)

L. Fajstrup, 05: In cubical models (for concurrency, e.g., HDAs), the two notions agree for d-paths ($X = \vec{I}$). In general, they do not.

Dihomotopy is finer than homotopy with fixed endpoints

Example: Two wedges in the forbidden region

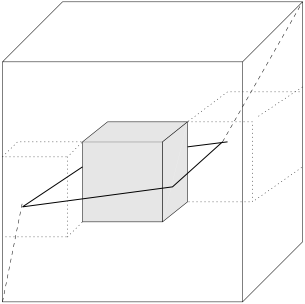


All dipaths from minimum to maximum are homotopic.
A dipath through the “hole” is not dihomotopic to a dipath on the boundary.

The twist has a price

Neither homogeneity nor cancellation nor group structure

Ordinary topology: Path space = loop space (within each path component).
A loop space is an H-space with concatenation, inversion, cancellation.



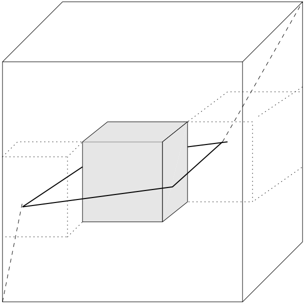
"Birth and death" of d-homotopy classes

Directed topology: Loops do not tell much; concatenation ok, cancellation not! Replace group structure by category structures!

The twist has a price

Neither homogeneity nor cancellation nor group structure

Ordinary topology: Path space = loop space (within each path component).
A loop space is an H-space with concatenation, inversion, cancellation.



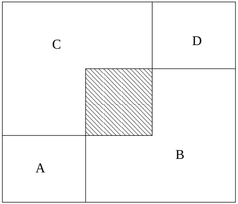
"Birth and death" of d-homotopy classes

Directed topology: Loops do not tell much; concatenation ok, cancellation not! Replace group structure by category structures!

A first remedy: the fundamental category

$\vec{\pi}_1(X)$ of a d-space X [Grandis:03, FGHR:04]:

- Objects: points in X
- Morphisms: d- or dihomotopy classes of d-paths in X
- Composition: from concatenation of d-paths

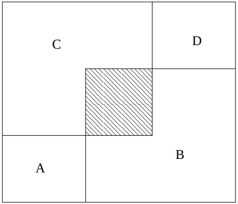


Property: van Kampen theorem (M. Grandis)
Drawbacks: Infinitely many objects. Calculations?
Question: How much does $\vec{\pi}_1(X)(x, y)$ depend on (x, y)?
Remedy: Localization, component category. [FGHR:04, GH:06]
Problem: This "compression" works only for loopfree categories (d-spaces)

A first remedy: the fundamental category

$\vec{\pi}_1(X)$ of a d-space X [Grandis:03, FGHR:04]:

- Objects: points in X
- Morphisms: d- or dihomotopy classes of d-paths in X
- Composition: from concatenation of d-paths



Property: van Kampen theorem (M. Grandis)
Drawbacks: Infinitely many objects. Calculations?
Question: How much does $\vec{\pi}_1(X)(x,y)$ depend on (x,y)?
Remedy: Localization, component category. [FGHR:04, GH:06]
Problem: This "compression" works only for loopfree categories (d-spaces)

Outline

- Spaces of d-paths and of traces
- Better bookkeeping: A zoo of categories and functors associated to a directed space – with a lot more animals than just the fundamental category
- Localization of categories with respect to (algebraic topological) functors via automorphic homotopy flows $\rightsquigarrow$ ”components”, compressing information, making calculations feasible.
- Directed homotopy equivalences – more than just the obvious generalization of the classical notion
  Definition? Automorphic homotopy flows! Properties?

X a (saturated) d-space.
$\varphi, \psi \in \vec{P}(X)(x,y)$ are called reparametrization equivalent if there are $\alpha, \beta \in \vec{P}(I)$ such that $\varphi \circ \alpha = \psi \circ \beta$ ("same oriented trace").
(Fahrenberg-R., 07): Reparametrization equivalence is an equivalence relation (transitivity).
$\vec{T}(X)(x,y) = \vec{P}(X)(x,y)/_{\simeq}$ makes $\vec{T}(X)$ into the (topologically enriched) trace category – composition associative.
A d-map $f : X \to Y$ induces a functor $\vec{T}(f) : \vec{T}(X) \to \vec{T}(Y)$.
On a pre-cubical set X, define the space of normalized d-paths $\vec{P}_n(X)$ with "arc length" parametrization (wrt. l^1).
The spaces $\vec{P}_n(X), \vec{P}(X)$ and $\vec{T}(X)$ are all homotopy equivalent.
D-homotopic paths in $\vec{P}_n(X)(x,y)$ have the same arc length.

X a (saturated) d-space.
$\varphi, \psi \in \vec{P}(X)(x,y)$ are called reparametrization equivalent if there are $\alpha, \beta \in \vec{P}(I)$ such that $\varphi \circ \alpha = \psi \circ \beta$ ("same oriented trace").
(Fahrenberg-R., 07): Reparametrization equivalence is an equivalence relation (transitivity).
$\vec{T}(X)(x,y) = \vec{P}(X)(x,y)/_{\simeq}$ makes $\vec{T}(X)$ into the (topologically enriched) trace category – composition associative.
A d-map $f : X \to Y$ induces a functor $\vec{T}(f) : \vec{T}(X) \to \vec{T}(Y)$.
On a pre-cubical set X, define the space of normalized d-paths $\vec{P}_n(X)$ with "arc length" parametrization (wrt. l^1).
The spaces $\vec{P}_n(X), \vec{P}(X)$ and $\vec{T}(X)$ are all homotopy equivalent.
D-homotopic paths in $\vec{P}_n(X)(x,y)$ have the same arc length.

X a (saturated) d-space.
$\varphi, \psi \in \vec{P}(X)(x, y)$ are called reparametrization equivalent if there are $\alpha, \beta \in \vec{P}(I)$ such that $\varphi \circ \alpha = \psi \circ \beta$ ("same oriented trace").
(Fahrenberg-R., 07): Reparametrization equivalence is an equivalence relation (transitivity).
$\vec{T}(X)(x, y) = \vec{P}(X)(x, y)/_{\simeq}$ makes $\vec{T}(X)$ into the (topologically enriched) trace category – composition associative.
A d-map $f : X \to Y$ induces a functor $\vec{T}(f) : \vec{T}(X) \to \vec{T}(Y)$.
On a pre-cubical set X, define the space of normalized d-paths $\vec{P}_n(X)$ with "arc length" parametrization (wrt. l^1).
The spaces $\vec{P}_n(X), \vec{P}(X)$ and $\vec{T}(X)$ are all homotopy equivalent.
D-homotopic paths in $\vec{P}_n(X)(x, y)$ have the same arc length.

X a (saturated) d-space.
$\varphi, \psi \in \vec{P}(X)(x,y)$ are called reparametrization equivalent if there are $\alpha, \beta \in \vec{P}(I)$ such that $\varphi \circ \alpha = \psi \circ \beta$ ("same oriented trace").
(Fahrenberg-R., 07): Reparametrization equivalence is an equivalence relation (transitivity).
$\vec{T}(X)(x,y) = \vec{P}(X)(x,y)/_{\simeq}$ makes $\vec{T}(X)$ into the (topologically enriched) trace category – composition associative.
A d-map $f : X \to Y$ induces a functor $\vec{T}(f) : \vec{T}(X) \to \vec{T}(Y)$.
On a pre-cubical set X, define the space of normalized d-paths $\vec{P}_n(X)$ with "arc length" parametrization (wrt. l^1).
The spaces $\vec{P}_n(X), \vec{P}(X)$ and $\vec{T}(X)$ are all homotopy equivalent.
D-homotopic paths in $\vec{P}_n(X)(x,y)$ have the same arc length.

Theorem

X a pre-cubical set; $x, y \in X$. Then $\vec{T}(X)(x, y)$ is

- *metrizable and locally contractible.*

Hope: Applications of Vietoris-Begle theorem for "inductive calculations".

Examples

I^n the unit cube, ∂I^n its boundary.

- $\vec{T}(I^n; \mathbf{x}, \mathbf{y})$ is contractible for all $x \preceq y \in I^n$;
- $\vec{T}(\partial I^n; \mathbf{0}, \mathbf{1})$ is homotopy equivalent to S^{n-2}.

Theorem

X a pre-cubical set; $x, y \in X$. Then $\vec{T}(X)(x,y)$ is

- *metrizable and locally contractible.*

Hope: Applications of Vietoris-Begle theorem for “inductive calculations”.

Examples

I^n the unit cube, ∂I^n its boundary.

- $\vec{T}(I^n; \mathbf{x}, \mathbf{y})$ is contractible for all $x \preceq y \in I^n$;
- $\vec{T}(\partial I^n; \mathbf{0}, \mathbf{1})$ is homotopy equivalent to S^{n-2}.

Preorder categories

Getting organised with indexing categories

A d-space structure on X induces the preorder $\preceq$:

$$x \preceq y \Leftrightarrow \vec{T}(X)(x,y) \neq \emptyset$$

and an indexing preorder category $\vec{D}(X)$ with

- Objects: (end point) pairs $(x,y), x \preceq y$
- Morphisms:
  $\vec{D}(X)((x,y),(x',y')) := \vec{T}(X)(x',x) \times \vec{T}(X)(y,y')$:

$$\begin{tikzcd} x' \arrow[r, bend left] \arrow[r, bend right] & x \arrow[r, "\preceq"] & y \arrow[r, bend left] \arrow[r, bend right] & y' \end{tikzcd}$$

- Composition: by pairwise contra-, resp. covariant concatenation.

A d-map $f : X \to Y$ induces a functor $\vec{D}(f) : \vec{D}(X) \to \vec{D}(Y)$.

Preorder categories

Getting organised with indexing categories

A d-space structure on X induces the preorder $\preceq$:

$$x \preceq y \Leftrightarrow \vec{T}(X)(x,y) \neq \emptyset$$

and an indexing preorder category $\vec{D}(X)$ with

- Objects: (end point) pairs $(x,y), x \preceq y$
- Morphisms:
  $\vec{D}(X)((x,y),(x',y')) := \vec{T}(X)(x',x) \times \vec{T}(X)(y,y')$:

$$\begin{tikzcd} x' \arrow[r, bend left] \arrow[r, bend right] & x \arrow[r, "\preceq"] & y \arrow[r, bend left] \arrow[r, bend right] & y' \end{tikzcd}$$

- Composition: by pairwise contra-, resp. covariant concatenation.

A d-map $f : X \to Y$ induces a functor $\vec{D}(f) : \vec{D}(X) \to \vec{D}(Y)$.

Preorder categories

Getting organised with indexing categories

A d-space structure on X induces the preorder $\preceq$:

$$x \preceq y \Leftrightarrow \vec{T}(X)(x,y) \neq \emptyset$$

and an indexing preorder category $\vec{D}(X)$ with

- Objects: (end point) pairs $(x,y), x \preceq y$
- Morphisms:
  $\vec{D}(X)((x,y),(x',y')) := \vec{T}(X)(x',x) \times \vec{T}(X)(y,y')$:

$$\begin{tikzcd} x' \arrow[r, bend left] \arrow[r, bend right] & x \arrow[r, "\preceq"] & y \arrow[r, bend left] \arrow[r, bend right] & y' \end{tikzcd}$$

- Composition: by pairwise contra-, resp. covariant concatenation.

A d-map $f : X \to Y$ induces a functor $\vec{D}(f) : \vec{D}(X) \to \vec{D}(Y)$.

The trace space functor

Preorder categories organise the trace spaces

The preorder category organises X via the trace space functor $\vec{T}^X : \vec{D}(X) \to Top$

- $\vec{T}^X(x,y) := \vec{T}(X)(x,y)$
- $\vec{T}^X(\sigma_x, \sigma_y)$:

$$\begin{array}{rcl} \vec{T}(X)(x,y) & \longrightarrow & \vec{T}(X)(x',y') \\ [\sigma] & \longmapsto & [\sigma_x * \sigma * \sigma_y] \end{array}$$

Homotopical variant $\vec{D}_\pi(X)$ with morphisms $\vec{D}_\pi(X)((x,y),(x',y')) := \vec{\pi}_1(X)(x',x) \times \vec{\pi}_1(X)(y,y')$ and trace space functor $\vec{T}^X_\pi : \vec{D}_\pi(X) \to Ho-Top$ (with homotopy classes as morphisms).

The preorder category organises X via the
trace space functor $\vec{T}^X : \vec{D}(X) \to Top$

- $\vec{T}^X(x,y) := \vec{T}(X)(x,y)$
- $\vec{T}^X(\sigma_x,\sigma_y):$ $$\begin{array}{ccc} \vec{T}(X)(x,y) & \longrightarrow & \vec{T}(X)(x',y') \\ [\sigma] & \longmapsto & [\sigma_x * \sigma * \sigma_y] \end{array}$$

Homotopical variant $\vec{D}_\pi(X)$ with morphisms
$\vec{D}_\pi(X)((x,y),(x',y')) := \vec{\pi}_1(X)(x',x) \times \vec{\pi}_1(X)(y,y')$
and trace space functor $\vec{T}_\pi^X : \vec{D}_\pi(X) \to Ho-Top$ (with homotopy classes as morphisms).

D-homology $\vec{H}_*$

For every d-space X, there are homology functors

$$\vec{H}_{*+1}(X) = H_* \circ \vec{T}^X_\pi : \vec{D}_\pi(X) \to Ab, \ (x, y) \mapsto H_*(\vec{T}(X)(x, y))$$

capturing homology of all relevant d-path spaces in X
and the effects of the concatenation structure maps.
A d-map $f : X \to Y$ induces a natural transformation $\vec{H}_{*+1}(f)$ from $\vec{H}_{*+1}(X)$ to $\vec{H}_{*+1}(Y)$.
Properties? Calculations? Not much known in general.
A master's student has studied this topic for X a cubical complex (its geometric realization) by constructing a cubical model for d-path spaces.
Similarly for other algebraic topological functors; a bit more complicated for homotopy groups: base points!

D-homology $\vec{H}_*$

For every d-space X, there are homology functors

$$\vec{H}_{*+1}(X) = H_* \circ \vec{T}_\pi^X : \vec{D}_\pi(X) \to Ab,\ (x, y) \mapsto H_*(\vec{T}(X)(x, y))$$

capturing homology of all relevant d-path spaces in X
and the effects of the concatenation structure maps.
A d-map $f : X \to Y$ induces a natural transformation $\vec{H}_{*+1}(f)$ from $\vec{H}_{*+1}(X)$ to $\vec{H}_{*+1}(Y)$.
Properties? Calculations? Not much known in general.
A master's student has studied this topic for X a cubical complex (its geometric realization) by constructing a cubical model for d-path spaces.
Similarly for other algebraic topological functors; a bit more complicated for homotopy groups: base points!

D-homology $\vec{H}_*$

For every d-space X, there are homology functors

$$\vec{H}_{*+1}(X) = H_* \circ \vec{T}_\pi^X : \vec{D}_\pi(X) \to Ab,\ (x,y) \mapsto H_*(\vec{T}(X)(x,y))$$

capturing homology of all relevant d-path spaces in X
and the effects of the concatenation structure maps.
A d-map $f : X \to Y$ induces a natural transformation $\vec{H}_{*+1}(f)$ from $\vec{H}_{*+1}(X)$ to $\vec{H}_{*+1}(Y)$.
Properties? Calculations? Not much known in general.
A master's student has studied this topic for X a cubical complex (its geometric realization) by constructing a cubical model for d-path spaces.
Similarly for other algebraic topological functors; a bit more complicated for homotopy groups: base points!

D-homology $\vec{H}_*$

For every d-space X, there are homology functors

$$\vec{H}_{*+1}(X) = H_* \circ \vec{T}_\pi^X : \vec{D}_\pi(X) \to Ab, \ (x,y) \mapsto H_*(\vec{T}(X)(x,y))$$

capturing homology of all relevant d-path spaces in X and the effects of the concatenation structure maps.
A d-map $f : X \to Y$ induces a natural transformation $\vec{H}_{*+1}(f)$ from $\vec{H}_{*+1}(X)$ to $\vec{H}_{*+1}(Y)$.
Properties? Calculations? Not much known in general.
A master's student has studied this topic for X a cubical complex (its geometric realization) by constructing a cubical model for d-path spaces.
Similarly for other algebraic topological functors; a bit more complicated for homotopy groups: base points!

Sensitivity with respect to variations of end points

Questions from a persistence point of view

- How much does (the homotopy type of) $\vec{T}^X(x,y)$ depend on (small) changes of x,y?
- Which concatenation maps $\vec{T}^X(\sigma_x,\sigma_y):\vec{T}^X(x,y)\to\vec{T}^X(x',y'),\ [\sigma]\mapsto[\sigma_x*\sigma*\sigma_y]$ are homotopy equivalences, induce isos on homotopy, homology groups etc.?
- The persistence point of view: Homology classes etc. are born (at certain branchings/mergings) and may die (analogous to the framework of G. Carlsson etal.)
- Are there “components” with (homotopically/homologically) stable dipath spaces (between them)? Are there borders (“walls”) at which changes occur?

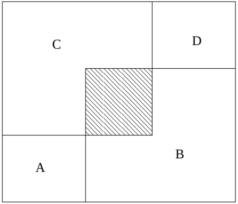
- How much does (the homotopy type of) $\vec{T}^X(x,y)$ depend on (small) changes of x,y?
- Which concatenation maps $\vec{T}^X(\sigma_x,\sigma_y) : \vec{T}^X(x,y) \to \vec{T}^X(x',y'),\ [\sigma] \mapsto [\sigma_x * \sigma * \sigma_y]$ are homotopy equivalences, induce isos on homotopy, homology groups etc.?
- The persistence point of view: Homology classes etc. are born (at certain branchings/mergings) and may die (analogous to the framework of G. Carlsson etal.)
- Are there "components" with (homotopically/homologically) stable dipath spaces (between them)? Are there borders ("walls") at which changes occur?

- How much does (the homotopy type of) $\vec{T}^X(x,y)$ depend on (small) changes of x,y?
- Which concatenation maps $\vec{T}^X(\sigma_x,\sigma_y): \vec{T}^X(x,y) \to \vec{T}^X(x',y'),\ [\sigma] \mapsto [\sigma_x * \sigma * \sigma_y]$ are homotopy equivalences, induce isos on homotopy, homology groups etc.?
- The persistence point of view: Homology classes etc. are born (at certain branchings/mergings) and may die (analogous to the framework of G. Carlsson etal.)
- Are there "components" with (homotopically/homologically) stable dipath spaces (between them)? Are there borders ("walls") at which changes occur?

- How much does (the homotopy type of) $\vec{T}^X(x,y)$ depend on (small) changes of x,y?
- Which concatenation maps $\vec{T}^X(\sigma_x,\sigma_y):\vec{T}^X(x,y)\to\vec{T}^X(x',y'),\ [\sigma]\mapsto[\sigma_x*\sigma*\sigma_y]$ are homotopy equivalences, induce isos on homotopy, homology groups etc.?
- The persistence point of view: Homology classes etc. are born (at certain branchings/mergings) and may die (analogous to the framework of G. Carlsson etal.)
- Are there "components" with (homotopically/homologically) stable dipath spaces (between them)? Are there borders ("walls") at which changes occur?

Examples of component categories

Example 1: No nontrivial d-loops



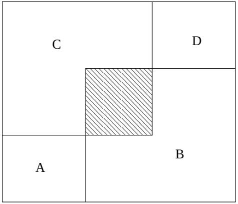
$$
\begin{tikzcd}
AA \arrow[drr] \arrow[ddr] & & & & BB \arrow[dll] \arrow[ddl] \\
& & AB \arrow[d] & \# & \\
& AC \arrow[r] & AD & BD \arrow[l] & \\
& \# & CD \arrow[u] & & \\
CC \arrow[uur] \arrow[urr] & & & & DD \arrow[ull] \arrow[uul]
\end{tikzcd}
$$

Figure: Deleted square with component category

Components A, B, C, D – or rather $AA, AB, AC, AD, BB, BD, CC, CD, DD \subseteq X \times X$.
$\#$: diagram commutes.

Examples of component categories

Example 1: No nontrivial d-loops

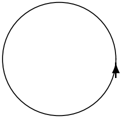


$$\begin{tikzcd}
AA \arrow[drr] \arrow[ddr] & & & & BB \arrow[dll] \arrow[ddl] \\
& & AB \arrow[d] & \# & \\
& AC \arrow[r] & AD & BD \arrow[l] & \\
& \# & CD \arrow[u] & & \\
CC \arrow[uur] \arrow[urr] & & & & DD \arrow[ull] \arrow[uul]
\end{tikzcd}$$

Figure: Deleted square with component category

Components A, B, C, D – or rather $AA, AB, AC, AD, BB, BD, CC, CD, DD \subseteq X \times X$.
$\#$: diagram commutes.

$X \quad = \quad \vec{S}^1$



oriented circle

$\vec{T}(\vec{S}^1)(x, y) \simeq \mathbf{N}_0$.

$$\mathcal{C} : \begin{tikzcd} \Delta \arrow[r, bend left, "a"] & \bar{\Delta} \arrow[l, bend left, "b"] \end{tikzcd}$$

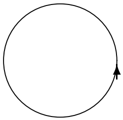
Δ the diagonal, $\bar{\Delta}$ its complement. $\mathcal{C}$ is the free category generated by a, b.

- Remark that the components are no longer products!
- In order to get a discrete component category, it is essential to use an indexing category taking care of pairs (source, target).

Examples of component categories

Example 2: Oriented circle

$X \quad = \quad \vec{S}^1$



oriented circle

$$\vec{T}(\vec{S}^1)(x,y) \simeq \mathbf{N}_0.$$

$$\mathcal{C} : \Delta \underset{b}{\overset{a}{\rightleftarrows}} \bar{\Delta}$$

Δ the diagonal, $\bar{\Delta}$ its complement.
$\mathcal{C}$ is the free category generated by a, b.

- Remark that the components are no longer products!
- In order to get a discrete component category, it is essential to use an indexing category taking care of pairs (source, target).

Tool: Homotopy flows

in particular: Automorphic homotopy flows

A d-map $H : X \times \vec{I} \to X$ is called a (f/p) homotopy flow if

future $H_0 = id_X \xrightarrow{H} f = H_1$

past $H_0 = g \xrightarrow{H} id_X = H_1$

H_t is not a homeomorphism, in general; the flow is irreversible.

H and f are called

automorphic if $\vec{T}(H_t) : \vec{T}(X)(x, y) \to \vec{T}(X)(H_t x, H_t y)$ is a homotopy equivalence for all $x \preceq y, t \in I$.

Automorphisms are closed under composition – concatenation of homotopy flows!

$Aut_+(X), Aut_-(X)$ monoids of automorphisms.

Variations: $\vec{T}(H_t)$ induces isomorphisms on homology groups, homotopy groups....

Tool: Homotopy flows
in particular: Automorphic homotopy flows

A d-map $H : X \times \vec{I} \to X$ is called a (f/p) homotopy flow if

future $H_0 = id_X \xrightarrow{H} f = H_1$

past $H_0 = g \xrightarrow{H} id_X = H_1$

H_t is not a homeomorphism, in general; the flow is irreversible.
H and f are called

automorphic if $\vec{T}(H_t) : \vec{T}(X)(x, y) \to \vec{T}(X)(H_t x, H_t y)$ is a homotopy equivalence for all $x \preceq y, t \in I$.

Automorphisms are closed under composition – concatenation of homotopy flows!

$Aut_+(X), Aut_-(X)$ monoids of automorphisms.

Variations: $\vec{T}(H_t)$ induces isomorphisms on homology groups, homotopy groups....

Tool: Homotopy flows

in particular: Automorphic homotopy flows

A d-map $H : X \times \vec{I} \to X$ is called a (f/p) homotopy flow if

future $H_0 = id_X \xrightarrow{H} f = H_1$

past $H_0 = g \xrightarrow{H} id_X = H_1$

H_t is not a homeomorphism, in general; the flow is irreversible.
H and f are called

automorphic if $\vec{T}(H_t) : \vec{T}(X)(x, y) \to \vec{T}(X)(H_t x, H_t y)$ is a homotopy equivalence for all $x \preceq y, t \in I$.

Automorphisms are closed under composition – concatenation of homotopy flows!
$Aut_+(X), Aut_-(X)$ monoids of automorphisms.
Variations: $\vec{T}(H_t)$ induces isomorphisms on homology groups, homotopy groups....

Compression: Generalized congruences and quotient categories

Bednarczyk, Borzyszkowski, Pawlowski, TAC 1999

How to identify morphisms in a category between different objects in an organised manner?

Start with an equivalence relation $\simeq$ on the objects.

A generalized congruence is an equivalence relation on non-empty sequences $\varphi = (f_1 \dots f_n)$ of morphisms with $cod(f_i) \simeq dom(f_{i+1})$ ($\simeq$-paths) satisfying

1. $\varphi \simeq \psi \Rightarrow dom(\varphi) \simeq dom(\psi), codom(\varphi) \simeq codom(\psi)$
2. $a \simeq b \Rightarrow id_a \simeq id_b$
3. $\varphi_1 \simeq \psi_1, \varphi_2 \simeq \psi_2, cod(\varphi_1) \simeq dom(\varphi_2) \Rightarrow \varphi_2\varphi_1 \simeq \psi_2\psi_1$
4. $cod(f) = dom(g) \Rightarrow f \circ g \simeq (fg)$

Quotient category $\mathcal{C}/\simeq$: Equivalence classes of objects and of $\simeq$-paths; composition: $[\varphi] \circ [\psi] = [\varphi\psi]$.

Compression: Generalized congruences and quotient categories

Bednarczyk, Borzyszkowski, Pawlowski, TAC 1999

How to identify morphisms in a category between different objects in an organised manner?
Start with an equivalence relation $\simeq$ on the objects.
A generalized congruence is an equivalence relation on non-empty sequences $\varphi = (f_1 \dots f_n)$ of morphisms with $cod(f_i) \simeq dom(f_{i+1})$ ($\simeq$-paths) satisfying

1. $\varphi \simeq \psi \Rightarrow dom(\varphi) \simeq dom(\psi), codom(\varphi) \simeq codom(\psi)$
2. $a \simeq b \Rightarrow id_a \simeq id_b$
3. $\varphi_1 \simeq \psi_1,\ \varphi_2 \simeq \psi_2,\ cod(\varphi_1) \simeq dom(\varphi_2) \Rightarrow \varphi_2\varphi_1 \simeq \psi_2\psi_1$
4. $cod(f) = dom(g) \Rightarrow f \circ g \simeq (fg)$

Quotient category $\mathcal{C}/\simeq$: Equivalence classes of objects and of $\simeq$-paths; composition: $[\varphi] \circ [\psi] = [\varphi\psi]$.

Compression: Generalized congruences and quotient categories

Bednarczyk, Borzyszkowski, Pawlowski, TAC 1999

How to identify morphisms in a category between different objects in an organised manner?
Start with an equivalence relation $\simeq$ on the objects.
A generalized congruence is an equivalence relation on non-empty sequences $\varphi = (f_1 \dots f_n)$ of morphisms with $cod(f_i) \simeq dom(f_{i+1})$ ($\simeq$-paths) satisfying

1. $\varphi \simeq \psi \Rightarrow dom(\varphi) \simeq dom(\psi), codom(\varphi) \simeq codom(\psi)$
2. $a \simeq b \Rightarrow id_a \simeq id_b$
3. $\varphi_1 \simeq \psi_1$, $\varphi_2 \simeq \psi_2$, $cod(\varphi_1) \simeq dom(\varphi_2) \Rightarrow \varphi_2\varphi_1 \simeq \psi_2\psi_1$
4. $cod(f) = dom(g) \Rightarrow f \circ g \simeq (fg)$

Quotient category $\mathcal{C}/\simeq$: Equivalence classes of objects and of $\simeq$-paths; composition: $[\varphi] \circ [\psi] = [\varphi\psi]$.

Automorphic homotopy flows give rise to generalized congruences

Let X be a d-space and $Aut_{\pm}(X)$ the monoid of all (future/past) automorphisms.
"Flow lines" are used to identify objects (pairs of points) and morphisms (classes of d-paths) in an organized manner.
$Aut_{\pm}(X)$ gives rise to a generalized congruence on the (homotopy) preorder category $\vec{D}_{\pi}(X)$ as the symmetric and transitive congruence closure of:

- $f_+ : (x,y) \overset{\simeq}{\leftrightarrow} (x',y') : f_-, \qquad f_\pm \in Aut_\pm(X)$
-
$$(x,y) \xrightarrow{(\sigma_1,\sigma_2)} (u,v) \simeq (x',y') \xrightarrow{(\tau_1,\tau_2)} (u',v'),$$
$f_+ : (x,y,u,v) \leftrightarrow (x',y',u',v') : f_-, \qquad f_\pm \in Aut_\pm(X)$, and
$$\begin{tikzcd} \vec{T}(X)(x',y') \arrow[r, "(\tau_1,\tau_2)"] \arrow[d, shift left, "\vec{T}(f_-)"] & \vec{T}(X)(u',v') \arrow[d, shift left, "\vec{T}(f_-)"] \\ \vec{T}(X)(x,y) \arrow[r, "(\sigma_1,\sigma_2)"] \arrow[u, shift left, "\vec{T}(f_+)"] & \vec{T}(X)(u,v) \arrow[u, shift left, "\vec{T}(f_+)"] \end{tikzcd}$$
commutes (up to ...).
- $(x,y) \xrightarrow{(c_x,H_y)} (x,fy) \simeq (fx,fy) \xrightarrow{(H_x,c_{fy})} (x,fy), \; H : id_X \to f.$
Likewise for $H : g \to id_X$.

The component category $\vec{D}_\pi(X)/\simeq$ identifies pairs of points on the same “homotopy flow line” and (chains of) morphisms.

Congruences and component categories

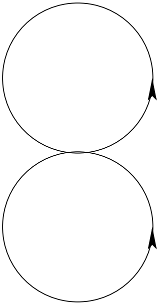
- $f_+ : (x,y) \overset{\simeq}{\leftrightarrow} (x',y') : f_-, \qquad f_\pm \in Aut_\pm(X)$
-
$$(x,y) \xrightarrow{(\sigma_1,\sigma_2)} (u,v) \simeq (x',y') \xrightarrow{(\tau_1,\tau_2)} (u',v'),$$
$f_+ : (x,y,u,v) \leftrightarrow (x',y',u',v') : f_-, \qquad f_\pm \in Aut_\pm(X)$, and
$$\begin{tikzcd} \vec{T}(X)(x',y') \arrow[r, "(\tau_1,\tau_2)"] \arrow[d, bend left, "\vec{T}(f_-)"] & \vec{T}(X)(u',v') \arrow[d, bend left, "\vec{T}(f_-)"] \\ \vec{T}(X)(x,y) \arrow[r, "(\sigma_1,\sigma_2)"] \arrow[u, bend left, "\vec{T}(f_+)"] & \vec{T}(X)(u,v) \arrow[u, bend left, "\vec{T}(f_+)"] \end{tikzcd}$$
commutes (up to ...).
- $(x,y) \xrightarrow{(c_x,H_y)} (x,fy) \simeq (fx,fy) \xrightarrow{(H_x,c_{fy})} (x,fy), \; H : id_X \to f.$
Likewise for $H : g \to id_X$.

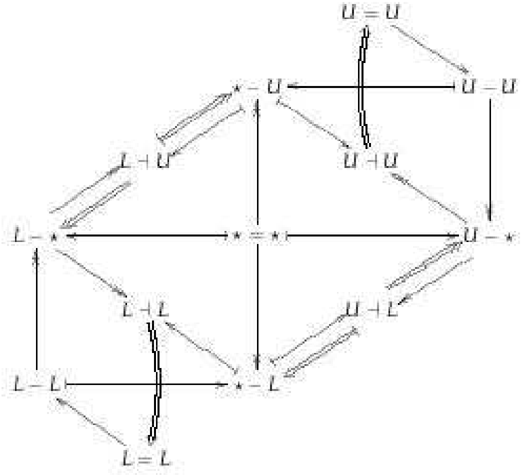
The component category $\vec{D}_\pi(X)/\simeq$ identifies pairs of points on the same "homotopy flow line" and (chains of) morphisms.

Examples of component categories

Example 3: The component category of a wedge of two oriented circles

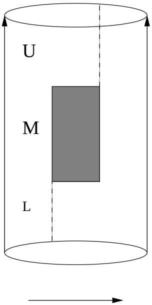
$$X = \vec{S}^1 \vee \vec{S}^1$$
$$\vec{T}(X)(x,y) \qquad \simeq$$
$$\mathbf{N}_0 * \mathbf{N}_0$$

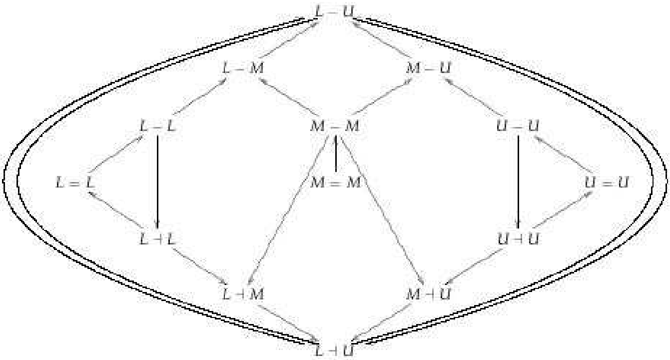




Examples of component categories

Example 4: The component category of an oriented cylinder with a deleted rectangle





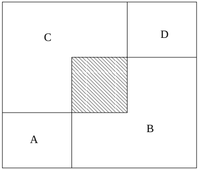
Dihomotopy equivalence – a naive definition

Definition

A d-map $f : X \to Y$ is a dihomotopy equivalence if there exists a d-map $g : Y \to X$ such that $g \circ f \simeq id_X$ and $f \circ g \simeq id_Y$.

But this does not imply an obvious property wanted for:
A dihomotopy equivalence $f : X \to Y$ should induce (ordinary) homotopy equivalences

$$\vec{T}(f) : \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy)!$$



A map d-homotopic to the identity does not preserve homotopy types of trace spaces? Need to be more restrictive!

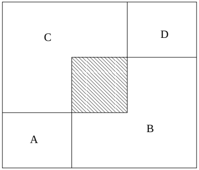
Dihomotopy equivalence – a naive definition

Definition

A d-map $f : X \to Y$ is a dihomotopy equivalence if there exists a d-map $g : Y \to X$ such that $g \circ f \simeq id_X$ and $f \circ g \simeq id_Y$.

But this does not imply an obvious property wanted for:
A dihomotopy equivalence $f : X \to Y$ should induce (ordinary) homotopy equivalences

$$\vec{T}(f) : \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy)!$$



A map d-homotopic to the identity does not preserve homotopy types of trace spaces? Need to be more restrictive!

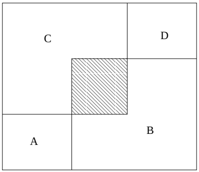
Dihomotopy equivalence – a naive definition

Definition

A d-map $f : X \to Y$ is a dihomotopy equivalence if there exists a d-map $g : Y \to X$ such that $g \circ f \simeq id_X$ and $f \circ g \simeq id_Y$.

But this does not imply an obvious property wanted for:
A dihomotopy equivalence $f : X \to Y$ should induce (ordinary) homotopy equivalences

$$\vec{T}(f) : \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy)!$$



A map d-homotopic to the identity does not preserve homotopy types of trace spaces? Need to be more restrictive!

Dihomotopy equivalences

using automorphic homotopy flows

Definition

A d-map $f: X \to Y$ is called a future dihomotopy equivalence if there are maps $f_+: X \to Y, g_+: Y \to X$ with $f \to f_+$ and automorphic homotopy flows $id_X \to g_+ \circ f_+, id_Y \to f_+ \circ g_+$.

Property of dihomotopy class!

likewise: past dihomotopy equivalence $f_- \to f, g_- \to g$

dihomotopy equivalence = both future and past dhe

(g_-, g_+ are then d-homotopic).

Theorem

A (future/past) d-homotopy equivalence $f: X \to Y$ induces homotopy equivalences

$$\vec{T}(f)(x,y): \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy) \text{ for all } x \preceq y.$$

Moreover: (All sorts of) Dihomotopy equivalences are closed under composition.

Definition

A d-map $f : X \to Y$ is called a future dihomotopy equivalence if there are maps $f_+ : X \to Y, g_+ : Y \to X$ with $f \to f_+$ and automorphic homotopy flows $id_X \to g_+ \circ f_+, id_Y \to f_+ \circ g_+$.
Property of dihomotopy class!

likewise: past dihomotopy equivalence $f_- \to f, g_- \to g$
dihomotopy equivalence = both future and past dhe
(g_-, g_+ are then d-homotopic).

Theorem

A (future/past) d-homotopy equivalence $f : X \to Y$ induces homotopy equivalences

$$\vec{T}(f)(x,y) : \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy) \text{ for all } x \preceq y.$$

Moreover: (All sorts of) Dihomotopy equivalences are closed under composition

Definition

A d-map $f : X \to Y$ is called a future dihomotopy equivalence if there are maps $f_+ : X \to Y, g_+ : Y \to X$ with $f \to f_+$ and automorphic homotopy flows $id_X \to g_+ \circ f_+, id_Y \to f_+ \circ g_+$.
Property of dihomotopy class!

likewise: past dihomotopy equivalence $f_- \to f, g_- \to g$
dihomotopy equivalence = both future and past dhe
(g_-, g_+ are then d-homotopic).

Theorem

A (future/past) d-homotopy equivalence $f : X \to Y$ induces homotopy equivalences

$$\vec{T}(f)(x,y) : \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy) \text{ for all } x \preceq y.$$

Moreover: (All sorts of) Dihomotopy equivalences are closed under composition

Dihomotopy equivalences

using automorphic homotopy flows

Definition

A d-map $f : X \to Y$ is called a future dihomotopy equivalence if there are maps $f_+ : X \to Y, g_+ : Y \to X$ with $f \to f_+$ and automorphic homotopy flows $id_X \to g_+ \circ f_+, id_Y \to f_+ \circ g_+$.
Property of dihomotopy class!

likewise: past dihomotopy equivalence $f_- \to f, g_- \to g$
dihomotopy equivalence = both future and past dhe
(g_-, g_+ are then d-homotopic).

Theorem

A (future/past) d-homotopy equivalence $f : X \to Y$ induces homotopy equivalences

$$\vec{T}(f)(x,y) : \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy) \text{ for all } x \preceq y.$$

Moreover: (All sorts of) Dihomotopy equivalences are closed under composition.

Concluding remarks

- Component categories contain the essential information given by (algebraic topological invariants of) path spaces
- Compression via component categories is an antidote to the state space explosion problem
- Some of the ideas (for the fundamental category) are implemented and have been tested for huge industrial software from EDF (Éric Goubault & Co., CEA)
- Dihomotopy equivalence: Definition uses automorphic homotopy flows to ensure homotopy equivalences

$$\vec{T}(f)(x,y) : \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy) \text{ for all } x \preceq y.$$

- Much more theoretical and practical work remains to be done!

Concluding remarks

- Component categories contain the essential information given by (algebraic topological invariants of) path spaces
- Compression via component categories is an antidote to the state space explosion problem
- Some of the ideas (for the fundamental category) are implemented and have been tested for huge industrial software from EDF (Éric Goubault & Co., CEA)
- Dihomotopy equivalence: Definition uses automorphic homotopy flows to ensure homotopy equivalences

$$\vec{T}(f)(x,y) : \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy) \text{ for all } x \preceq y.$$

- Much more theoretical and practical work remains to be done!

Concluding remarks

- Component categories contain the essential information given by (algebraic topological invariants of) path spaces
- Compression via component categories is an antidote to the state space explosion problem
- Some of the ideas (for the fundamental category) are implemented and have been tested for huge industrial software from EDF (Éric Goubault & Co., CEA)
- Dihomotopy equivalence: Definition uses automorphic homotopy flows to ensure homotopy equivalences

$$\vec{T}(f)(x,y) : \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy) \text{ for all } x \preceq y.$$

- Much more theoretical and practical work remains to be done!

Concluding remarks

- Component categories contain the essential information given by (algebraic topological invariants of) path spaces
- Compression via component categories is an antidote to the state space explosion problem
- Some of the ideas (for the fundamental category) are implemented and have been tested for huge industrial software from EDF (Éric Goubault & Co., CEA)
- Dihomotopy equivalence: Definition uses automorphic homotopy flows to ensure homotopy equivalences

$$\vec{T}(f)(x,y) : \vec{T}(X)(x,y) \to \vec{T}(Y)(fx,fy) \text{ for all } x \preceq y.$$

- Much more theoretical and practical work remains to be done!

Thanks!

The end!

Thanks to

Larry Smith for the invitation!

you all for listening to this talk!