(Automatically) verifying Morse reductions in Cubical Agda

Proglog meeting
Chalmers University, 13 March 2024

Maximilian Doré
maximilian.dore@cs.ox.ac.uk

Background

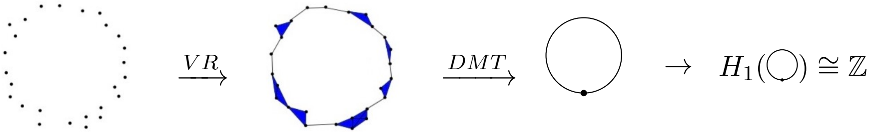
Project: Formalisation in applied topology, *using* and *automating* Cubical Agda's logic for homotopy types.

This talk:

- Overview of results.
- Convey "style" of proofs carried out.
- Outlook on improvements that'd be nice for CTT tools.

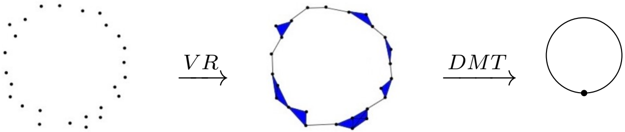
Topological data analysis

Study the *shape of data* using tools from algebraic topology:



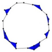
Topological data analysis

Study the *shape of data* using tools from algebraic topology:



$\rightarrow \quad H_1(\bigcirc) \cong \mathbb{Z}$

Discrete Morse theory (DMT):
Reduce size of complex while retaining its topology.

$\rightarrow$ *establishes that*  $\simeq$ 

Discrete Morse theory for graphs

Input: Graph G and acyclic partial matching (APM) μ
Output: Equivalent Morse complex M of critical cells of G

Discrete Morse theory for graphs

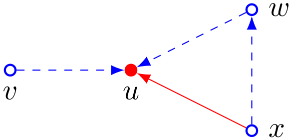
Input: Graph G and acyclic partial matching (APM) μ
Output: Equivalent Morse complex M of critical cells of G

- APM: pairs of vertices and edges such that every gradient path $u \rightsquigarrow c$ ends at some $c \in M_0$
- $M_0 \triangleq \Sigma u \notin \mu, \;\; M_1(c, d) \triangleq \Sigma(uv \notin \mu).(u \rightsquigarrow c) \times (v \rightsquigarrow d)$

Discrete Morse theory for graphs

Input: Graph G and acyclic partial matching (APM) μ
Output: Equivalent Morse complex M of critical cells of G

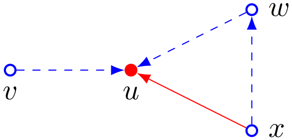
- APM: pairs of vertices and edges such that every gradient path $u \rightsquigarrow c$ ends at some $c \in M_0$
- $M_0 \triangleq \Sigma u \notin \mu, \;\; M_1(c,d) \triangleq \Sigma(uv \notin \mu).(u \rightsquigarrow c) \times (v \rightsquigarrow d)$



Discrete Morse theory for graphs

Input: Graph G and acyclic partial matching (APM) μ
Output: Equivalent Morse complex M of critical cells of G

- APM: pairs of vertices and edges such that every gradient path $u \rightsquigarrow c$ ends at some $c \in M_0$
- $M_0 \triangleq \Sigma u \notin \mu, \;\; M_1(c, d) \triangleq \Sigma(uv \notin \mu).(u \rightsquigarrow c) \times (v \rightsquigarrow d)$

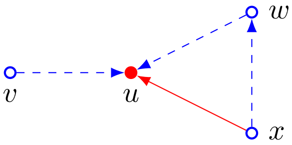


$\mu \triangleq (v, vu), (w, wu), (x, xw)$

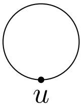
Discrete Morse theory for graphs

Input: Graph G and acyclic partial matching (APM) μ
Output: Equivalent Morse complex M of critical cells of G

- APM: pairs of vertices and edges such that every gradient path $u \rightsquigarrow c$ ends at some $c \in M_0$
- $M_0 \triangleq \Sigma u \notin \mu, \;\; M_1(c, d) \triangleq \Sigma(uv \notin \mu).(u \rightsquigarrow c) \times (v \rightsquigarrow d)$



$\mu \triangleq (v, vu), (w, wu), (x, xw)$

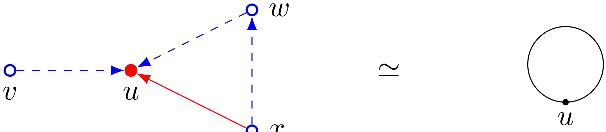


$M_0 \triangleq u$
$M_1(u, u) \triangleq (xu, [(x, xw), (w, wu)], [])$

Discrete Morse theory for graphs

Input: Graph G and acyclic partial matching (APM) μ
Output: Equivalent Morse complex M of critical cells of G

- APM: pairs of vertices and edges such that every gradient path $u \rightsquigarrow c$ ends at some $c \in M_0$
- $M_0 \triangleq \Sigma u \notin \mu, \;\; M_1(c,d) \triangleq \Sigma(uv \notin \mu).(u \rightsquigarrow c) \times (v \rightsquigarrow d)$



$\mu \triangleq (v, vu), (w, wu), (x, xw)$

$M_0 \triangleq u$
$M_1(u,u) \triangleq (xu, [(x,xw),(w,wu)], [])$

The Morse theorem in Cubical Agda

Use HIT to introduce homotopy type of a relation:

```
data |_| {c₀ : Type} (c₁ : c₀ → c₀ → Type) : Type where
  |_|₀ : c₀ → | c₁ |
  |_⇒_∋_|₁ : (x y : c₀) → c₁ x y → | x |₀ ≡ | y |₀
```

The Morse theorem in Cubical Agda

Use HIT to introduce homotopy type of a relation:

```
data |_| {c₀ : Type} (c₁ : c₀ → c₀ → Type) : Type where
  |_|₀ : c₀ → | c₁ |
  |_⇒_∋_|₁ : (x y : c₀) → c₁ x y → | x |₀ ≡ | y |₀
```

Example: for $M_0 \triangleq u$, $M_1(u,u) \triangleq (xu, [(x, xw), (w, wu)], [])$, it follows with simple pattern matching that $|\ M\ | \equiv S^1$.

The Morse theorem in Cubical Agda

Use HIT to introduce homotopy type of a relation:

$$\mathsf{data}\ |\_|\ \{c_0 : \mathsf{Type}\}\ (c_1 : c_0 \to c_0 \to \mathsf{Type}) : \mathsf{Type}\ \mathsf{where}$$
$$|\_|_0 : c_0 \to |\ c_1\ |$$
$$|\_\Rightarrow\_\ni\_|_1 : (x\ y : c_0) \to c_1\ x\ y \to |\ x\ |_0 \equiv |\ y\ |_0$$

Example: for $M_0 \triangleq u$, $M_1(u, u) \triangleq (xu, [(x, xw), (w, wu)], [])$, it follows with simple pattern matching that $|\ M\ | \equiv S^1$.

Central result: $|\ G\ |\ \equiv\ |\ M\ |$

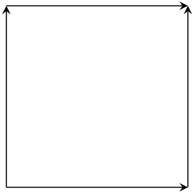
- establish maps back and forth, e.g., define $|\ M_1\ | \to |\ E\ |$: $|\ c \Rightarrow d \ni (uv, \gamma, \delta)\ |_1 \mapsto |\ \gamma\ |^{-1} \bullet\bullet\ |\ u \Rightarrow v \ni uv\ |_1 \bullet\bullet\ |\ \delta\ |$
- show that these maps are mutually inverse.

https://cs.ox.ac.uk/people/maximilian.dore/thesis/html/Morse.Morse.html

Proving the Morse Theorem

E.g., for any critical edge $\mid u \Rightarrow v \ni uv \mid_1$ show that going along $\mid E \mid \to \mid M_1 \mid \to \mid E \mid$ is coherent:

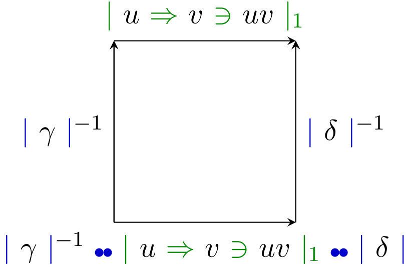
$\mid u \Rightarrow v \ni uv \mid_1$



$\mid \gamma \mid^{-1} \bullet\bullet \mid u \Rightarrow v \ni uv \mid_1 \bullet\bullet \mid \delta \mid$

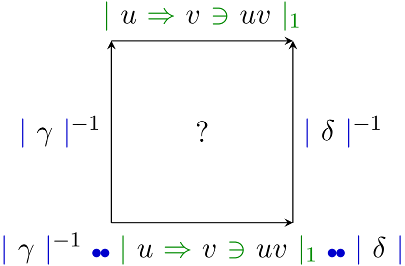
Proving the Morse Theorem

E.g., for any critical edge $| u \Rightarrow v \ni uv |_1$ show that going along $| E | \to | M_1 | \to | E |$ is coherent:



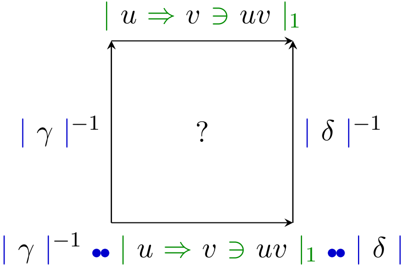
Proving the Morse Theorem

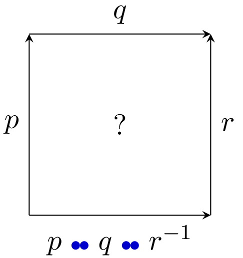
E.g., for any critical edge $|\ u \Rightarrow v \ni uv\ |_1$ show that going along $|\ E\ | \to |\ M_1\ | \to |\ E\ |$ is coherent:



Proving the Morse Theorem

E.g., for any critical edge $| u \Rightarrow v \ni uv |_1$ show that going along $| E | \to | M_1 | \to | E |$ is coherent:





Automating Boundary Filling in Cubical Agda[1]

Search problem: Given cell context Γ and boundary $\Gamma \vdash \phi$, give a cell $\Gamma \vdash t : \phi$ constructed from

- **Contortions**: interval substitutions with $\wedge$, $\vee$, $\sim$.
- **Kan compositions**: gives completion of open cube with ϕ on missing side.

[1] jww with Evan Cavallo & Anders Mörtberg, arXiv:2402.12169

Automating Boundary Filling in Cubical Agda[1]

Search problem: Given cell context Γ and boundary $\Gamma \vdash \phi$, give a cell $\Gamma \vdash t : \phi$ constructed from

- **Contortions**: interval substitutions with $\wedge$, $\vee$, $\sim$.
- **Kan compositions**: gives completion of open cube with ϕ on missing side.

Example: $q : x \equiv y,\ r : y \equiv z\ \vdash\ ? : x \equiv z$

$$x \overset{?}{\dashrightarrow} z$$

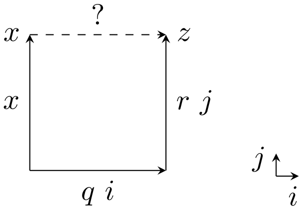
[1] jww with Evan Cavallo & Anders Mörtberg, arXiv:2402.12169

Automating Boundary Filling in Cubical Agda[1]

Search problem: Given cell context Γ and boundary $\Gamma \vdash \phi$, give a cell $\Gamma \vdash t : \phi$ constructed from

- **Contortions**: interval substitutions with $\wedge$, $\vee$, $\sim$.
- **Kan compositions**: gives completion of open cube with ϕ on missing side.

Example: $q : x \equiv y,\ r : y \equiv z\ \vdash\ ? : x \equiv z$



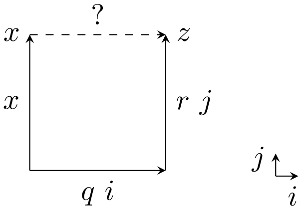
[1] jww with Evan Cavallo & Anders Mörtberg, arXiv:2402.12169

Automating Boundary Filling in Cubical Agda[1]

Search problem: Given cell context Γ and boundary $\Gamma \vdash \phi$, give a cell $\Gamma \vdash t : \phi$ constructed from

- **Contortions**: interval substitutions with $\wedge$, $\vee$, $\sim$.
- **Kan compositions**: gives completion of open cube with ϕ on missing side.

Example: $q : x \equiv y,\ r : y \equiv z \ \vdash\ ? : x \equiv z$



$(q \bullet r)\ i = \mathsf{hcomp}\ (\lambda\ j \to \lambda\ \{$
$\quad (i = \mathsf{i0}) \to x$
$\quad ;\ (i = \mathsf{i1}) \to r\ j$
$\quad \})\ (q\ i)$

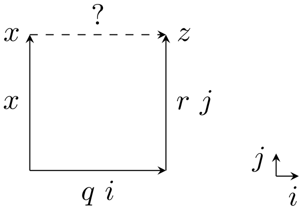
[1] jww with Evan Cavallo & Anders Mörtberg, arXiv:2402.12169

Automating Boundary Filling in Cubical Agda[1]

Search problem: Given cell context Γ and boundary $\Gamma \vdash \phi$, give a cell $\Gamma \vdash t : \phi$ constructed from

- **Contortions**: interval substitutions with $\wedge$, $\vee$, $\sim$.
- **Kan compositions**: gives completion of open cube with ϕ on missing side.

Example: $q : x \equiv y,\ r : y \equiv z \ \vdash\ ? : x \equiv z$

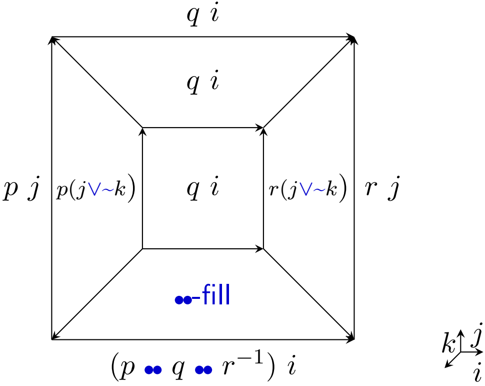


$$
\begin{aligned}
&(q \bullet r)\ i = \mathsf{hcomp}\ (\lambda\ j \to \lambda\ \{\\
&\quad (i = \mathsf{i0}) \to x\\
&\quad ; (i = \mathsf{i1}) \to r\ j\\
&\quad \})\ (q\ i)
\end{aligned}
$$

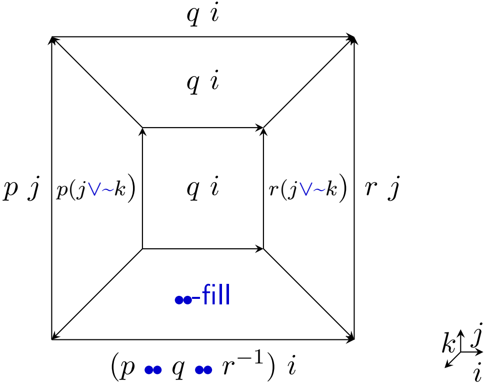
Find Kan compositions using finite domain constraint solving.

[1] jww with Evan Cavallo & Anders Mörtberg, arXiv:2402.12169

Higher-dimensional Kan composition



Higher-dimensional Kan composition



Two issues:

- With growing dimension, there are *a lot* of contortions.
- Sides of the cube can also be the result of Kan composition.

Dedekind contortions

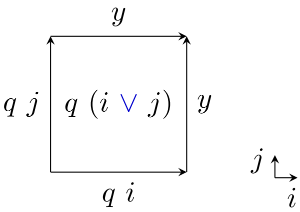
Contortions built only with $\wedge$,$\vee$ can also be seen as poset maps:

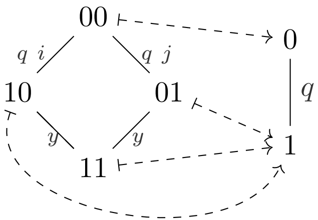
$$\text{Dedekind contortions built from } \Psi$$
$$\leftrightsquigarrow$$
$$\text{poset maps } \mathbf{I}^{|\Psi|} \to \mathbf{I} \text{ for } \mathbf{I} := \{0 < 1\}.$$

Dedekind contortions

Contortions built only with $\wedge$,$\vee$ can also be seen as poset maps:

$$\text{Dedekind contortions built from } \Psi$$
$$\leftrightsquigarrow$$
$$\text{poset maps } \mathbf{I}^{|\Psi|} \to \mathbf{I} \text{ for } \mathbf{I} := \{0 < 1\}.$$

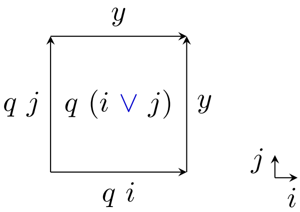


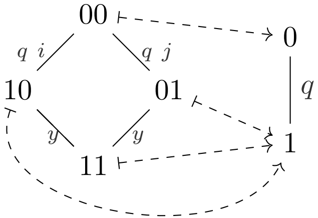


Dedekind contortions

Contortions built only with $\wedge$,$\vee$ can also be seen as poset maps:

$$\text{Dedekind contortions built from } \Psi$$
$$\leftrightsquigarrow$$
$$\text{poset maps } \mathbf{I}^{|\Psi|} \to \mathbf{I} \text{ for } \mathbf{I} := \{0 < 1\}.$$

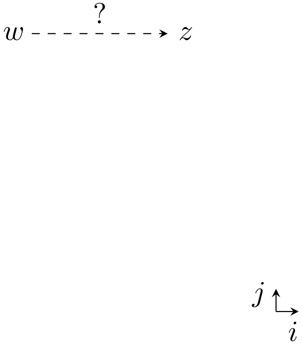




Construct contortions gradually by restricting poset map.

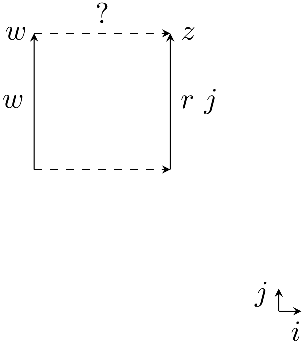
Searching for nested Kan compositions

$$p : w \equiv x,\ q : x \equiv y,\ r : y \equiv z \ \vdash\ ? : w \equiv z$$



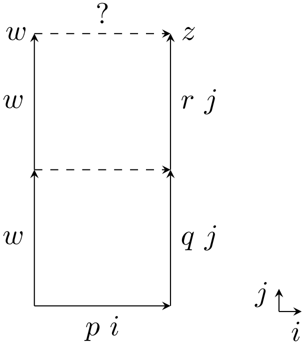
Searching for nested Kan compositions

$$p : w \equiv x,\ q : x \equiv y,\ r : y \equiv z \vdash ? : w \equiv z$$



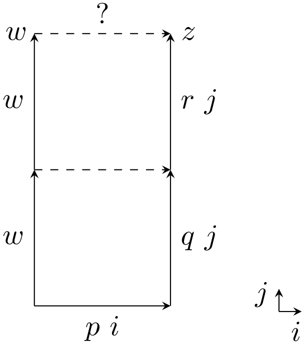
Searching for nested Kan compositions

$$p : w \equiv x,\ q : x \equiv y,\ r : y \equiv z\ \vdash\ ? : w \equiv z$$



Searching for nested Kan compositions

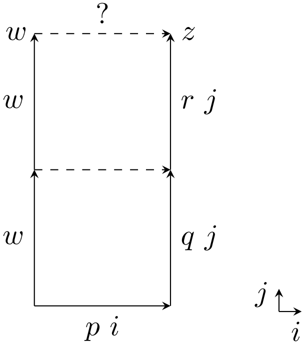
$$p : w \equiv x,\ q : x \equiv y,\ r : y \equiv z\ \vdash\ ? : w \equiv z$$




```
(p ∙∙ q ∙∙ r) i = hcomp (λ j → λ {
      (i = i0) → w
    ; (i = i1) → r j
    }) (hcomp (λ j → λ {
      (i = i0) → w
    ; (i = i1) → q j
    }) (p i))
```

Searching for nested Kan compositions

$$p : w \equiv x,\ q : x \equiv y,\ r : y \equiv z\ \vdash\ ? : w \equiv z$$




```
(p ∙∙ q ∙∙ r) i = hcomp (λ j → λ {
      (i = i0) → w
   ; (i = i1) → r j
   }) (hcomp (λ j → λ {
      (i = i0) → w
   ; (i = i1) → q j
   }) (p i))
```

Use as many contortions as possible, fill the remaining sides with Kan compositions afterwards.

Demo

`https://github.com/maxdore/dedekind`

Summary

- Applied topology is a natural application for Cubical Agda.
- Being able to directly reason about homotopy types is neat, difficult combinatorial steps can be carried out by solver.

Outlook

Developing a tool for TDA fully in Cubical Agda:

- Formalise DMT for 2-dimensional complexes
  $\to$ compute topology of grayscale images[2]
- Implement full pipeline: turn grayscale image into complex; compute APM; compute cohomology of reduced complex.

[2] *Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images*, Robins, Wood & Sheppard, 2010

Outlook

Developing a tool for TDA fully in Cubical Agda:

- Formalise DMT for 2-dimensional complexes
  → compute topology of grayscale images[2]
- Implement full pipeline: turn grayscale image into complex; compute APM; compute cohomology of reduced complex.

Improve tools along the way:

- Cubical compiler necessary to get executable.
- Incorporate solver into Cubical Agda; refine heuristics; incorporate heterogeneous equality, `transp`, etc.
- Combine with synthesis of dependent type theory.

[2] *Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images*, Robins, Wood & Sheppard, 2010

Outlook

Developing a tool for TDA fully in Cubical Agda:

- Formalise DMT for 2-dimensional complexes
  → compute topology of grayscale images[2]
- Implement full pipeline: turn grayscale image into complex; compute APM; compute cohomology of reduced complex.

Improve tools along the way:

- Cubical compiler necessary to get executable.
- Incorporate solver into Cubical Agda; refine heuristics; incorporate heterogeneous equality, `transp`, etc.
- Combine with synthesis of dependent type theory.

Thank you for your attention!

[2] *Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images*, Robins, Wood & Sheppard, 2010