Formalising Topological Data Analysis in Cubical Agda

DANGER 4

London Institute for Mathematical Sciences, 8 August 2024

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

Formalising data science

DANGER: data science $\rightarrow$ pure maths

Formalising data science

DANGER: data science $\rightarrow$ pure maths

This talk: pure maths/CS $\rightarrow$ data science

Formalising data science

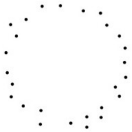
DANGER: data science $\rightarrow$ pure maths

This talk: pure maths/CS $\rightarrow$ data science

Goal: implement a fully verified tool for topological data analysis

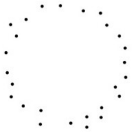
Topological data analysis

Study the *shape of data* using algebraic topology. Typical pipeline:

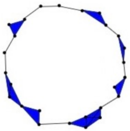


Topological data analysis

Study the *shape of data* using algebraic topology. Typical pipeline:



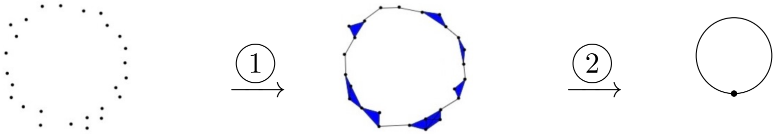
$\xrightarrow{①}$



① associate Vietoris-Rips complex with dataset

Topological data analysis

Study the *shape of data* using algebraic topology. Typical pipeline:

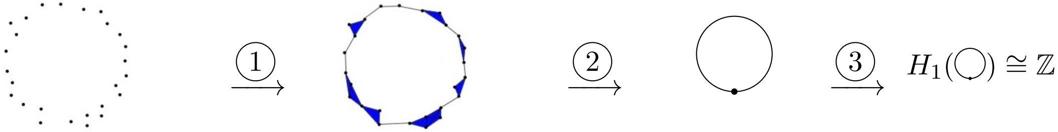


(1) associate Vietoris-Rips complex with dataset

(2) reduce size of complex using discrete Morse theory

Topological data analysis

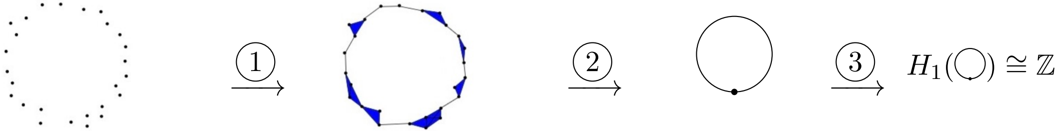
Study the *shape of data* using algebraic topology. Typical pipeline:



1. associate Vietoris-Rips complex with dataset
2. reduce size of complex using discrete Morse theory
3. compute (persistent) homology

Topological data analysis

Study the *shape of data* using algebraic topology. Typical pipeline:



(1) associate Vietoris-Rips complex with dataset

(2) reduce size of complex using discrete Morse theory

(3) compute (persistent) homology

→ implement this pipeline in a way that is *provably correct*

Writing provably correct programs

Why is $A \to (B \to A)$ a tautology?

Writing provably correct programs

Why is $A \to (B \to A)$ a tautology?

Because $\lambda x.\lambda y.x \quad : \quad A \to (B \to A)$

Writing provably correct programs

Why is $A \to (B \to A)$ a tautology?

Because $\underbrace{\lambda x.\lambda y.x}_{\text{proof } aka \text{ program}} : \underbrace{A \to (B \to A)}_{\text{proposition } aka \text{ type}}$

Writing provably correct programs

Why is $A \to (B \to A)$ a tautology?

Because $$\underbrace{\lambda x.\lambda y.x}_{\text{proof } \textit{aka} \text{ program}} \quad : \quad \underbrace{A \to (B \to A)}_{\text{proposition } \textit{aka} \text{ type}}$$

Dependent type theory (Martin-Löf 1984) extends this to predicates and induction.
Types can depend on variables, Π and Σ are dependent functions and products.

Writing provably correct programs

Why is $A \to (B \to A)$ a tautology?

Because $\underbrace{\lambda x.\lambda y.x}_{\text{proof } aka \text{ program}} \quad : \quad \underbrace{A \to (B \to A)}_{\text{proposition } aka \text{ type}}$

Dependent type theory (Martin-Löf 1984) extends this to predicates and induction. Types can depend on variables, Π and Σ are dependent functions and products.

Correct-by-construction programming

Give specification as type T, then program $p : T$ provably meets the specification.

Writing provably correct programs

Why is $A \to (B \to A)$ a tautology?

Because $\underbrace{\lambda x.\lambda y.x}_{\text{proof } aka \text{ program}} : \underbrace{A \to (B \to A)}_{\text{proposition } aka \text{ type}}$

Dependent type theory (Martin-Löf 1984) extends this to predicates and induction.
Types can depend on variables, Π and Σ are dependent functions and products.

Correct-by-construction programming

Give specification as type T, then program $p : T$ provably meets the specification.

(1) associate Vietoris-Rips complex with dataset

Write a program of this type:

$\Pi(D : \text{Dataset}).\Sigma(K : \text{Complex}).\Pi(x\ y : \text{points}(D)).d(x, y) < \epsilon \to \text{line}(K, x, y)$

Showing Morse reductions correct

(2) reduce size of complex using discrete Morse theory

Discrete Morse theory removes cells irrelevant for the topology of a complex K, using an *acyclic partial matching* μ that says how to collapse cells.

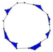
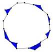
Morse complex M contains only cells not part of μ. *Morse theorem*: $| K | \simeq | M |$

Showing Morse reductions correct

(2) reduce size of complex using discrete Morse theory

Discrete Morse theory removes cells irrelevant for the topology of a complex K, using an *acyclic partial matching* μ that says how to collapse cells.

Morse complex M contains only cells not part of μ. *Morse theorem*: $|\, K \,| \simeq |\, M \,|$

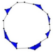
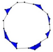
Example: given  and apt matching, Morse theorem gives  $\simeq$ 

Showing Morse reductions correct

(2) reduce size of complex using discrete Morse theory

Discrete Morse theory removes cells irrelevant for the topology of a complex K, using an *acyclic partial matching* μ that says how to collapse cells.

Morse complex M contains only cells not part of μ. *Morse theorem*: $|\, K \,| \simeq |\, M \,|$

Example: given  and apt matching, Morse theorem gives  $\simeq$ 

Specification for step (2):

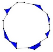
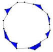
$\Pi(K : \text{Complex}).\Pi(\mu : \text{Matching}(K)).\Sigma(M : \text{Complex}').|\, K \,| \simeq |\, M \,|$

Showing Morse reductions correct

(2) reduce size of complex using discrete Morse theory

Discrete Morse theory removes cells irrelevant for the topology of a complex K, using an *acyclic partial matching* μ that says how to collapse cells.

Morse complex M contains only cells not part of μ. *Morse theorem*: $|\ K\ | \simeq |\ M\ |$

Example: given  and apt matching, Morse theorem gives  $\simeq$ 

Specification for step (2):

$\Pi(K : \mathrm{Complex}).\Pi(\mu : \mathrm{Matching}(K)).\Sigma(M : \mathrm{Complex}').|\ K\ | \simeq |\ M\ |$

Original work[1] up to homology, modern approach[2] up to homotopy equivalence. Intricate to formalise head-on...

[1] Forman 1998, *Morse Theory for Cell Complexes*
[2] Nanda 2019, *Discrete Morse Theory and Localization*

Homotopy Type Theory and Cubical Agda

Idea for dealing with equality in type theory: there can be different proofs of an equality, and it's meaningful to study equalities between equalities.

Example: For $a, b : A$ have $p, q : a \equiv b$ and $\alpha, \beta : p \equiv q$ and $\phi : \alpha \equiv \beta$ etc...

Homotopy Type Theory and Cubical Agda

Idea for dealing with equality in type theory: there can be different proofs of an equality, and it's meaningful to study equalities between equalities.

Example: For $a, b : A$ have $p, q : a \equiv b$ and $\alpha, \beta : p \equiv q$ and $\phi : \alpha \equiv \beta$ etc...

Homotopy type theory by Awodey, Voevodsky, ..., 2013

$\equiv$ in type theory *aka* $\simeq$ in topology *aka* $\cong$ in higher category theory

Homotopy Type Theory and Cubical Agda

Idea for dealing with equality in type theory: there can be different proofs of an equality, and it's meaningful to study equalities between equalities.

Example: For $a, b : A$ have $p, q : a \equiv b$ and $\alpha, \beta : p \equiv q$ and $\phi : \alpha \equiv \beta$ etc...

Homotopy type theory by Awodey, Voevodsky, ..., 2013

$\equiv$ in type theory *aka* $\simeq$ in topology *aka* $\cong$ in higher category theory

Cubical Agda 2019[1] implements these ideas:

- types are really homotopy types, and $\equiv$ behaves like $\simeq$
- *higher inductive types* allow for introducing higher equalities
- properties of cubical sets become principles of logic

[1] based on ideas of Abel, Bezem, Coquand, Cohen, Huber, Mörtberg, Vezzosi, ...

Discrete Morse theory for graphs

Given a (directed) graph G and an acyclic matching μ, we want to construct its Morse reduct M and show that $|\,G\,| \equiv |\,M\,|$.

Discrete Morse theory for graphs

Given a (directed) graph G and an acyclic matching μ, we want to construct its Morse reduct M and show that $\mid G \mid \; \equiv \; \mid M \mid$.

A *gradient path* $\gamma : u \rightsquigarrow v$ is a sequence of matched pairs between u and v.

$M_0 \triangleq \Sigma u \notin \mu, \;\; M_1(c,d) \triangleq \Sigma(uv \notin \mu).(u \rightsquigarrow c) \times (v \rightsquigarrow d)$

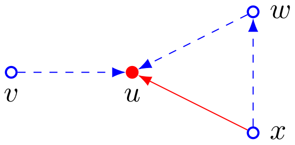
Discrete Morse theory for graphs

Given a (directed) graph G and an acyclic matching μ, we want to construct its Morse reduct M and show that $|\, G \,| \;\equiv\; |\, M \,|$.

A *gradient path* $\gamma : u \rightsquigarrow v$ is a sequence of matched pairs between u and v.

$M_0 \triangleq \Sigma u \notin \mu, \;\; M_1(c, d) \triangleq \Sigma(uv \notin \mu).(u \rightsquigarrow c) \times (v \rightsquigarrow d)$

Example:



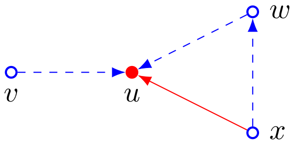
Discrete Morse theory for graphs

Given a (directed) graph G and an acyclic matching μ, we want to construct its Morse reduct M and show that $\mid G \mid \; \equiv \; \mid M \mid$.

A *gradient path* $\gamma : u \rightsquigarrow v$ is a sequence of matched pairs between u and v.

$M_0 \triangleq \Sigma u \notin \mu, \;\; M_1(c, d) \triangleq \Sigma(uv \notin \mu).(u \rightsquigarrow c) \times (v \rightsquigarrow d)$

Example:



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

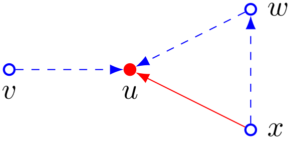
Discrete Morse theory for graphs

Given a (directed) graph G and an acyclic matching μ, we want to construct its Morse reduct M and show that $|\ G\ | \equiv |\ M\ |$.

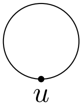
A *gradient path* $\gamma : u \rightsquigarrow v$ is a sequence of matched pairs between u and v.

$M_0 \triangleq \Sigma u \notin \mu,\ \ M_1(c,d) \triangleq \Sigma(uv \notin \mu).(u \rightsquigarrow c) \times (v \rightsquigarrow d)$

Example:



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



$M_0 \triangleq u$

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

The Morse theorem in Cubical Agda

Higher inductive type allows us to take the *geometric realisation* 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

Higher inductive type allows us to take the *geometric realisation* 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
```

Ex. cont'd: $|\ M\ |$ has point $|\ u\ |_0$ with circle $|\ u \Rightarrow u \ni (xu, [(x, xw), (w, wu)], [])\ |_1$

The Morse theorem in Cubical Agda

Higher inductive type allows us to take the *geometric realisation* 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
```

Ex. cont'd: $|\ M\ |$ has point $|\ u\ |_0$ with circle $|\ u \Rightarrow u \ni (xu, [(x, xw), (w, wu)], [])\ |_1$

Morse theorem for graphs: $|\ G\ |\ \equiv\ |\ M\ |$ formalised in Cubical Agda:

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

Computing invariants of the space

③ compute (persistent) homology

Computing invariants involves loads of linear algebra. Already partially formalised.[1]

WIP: how to integrate this with my approach.

Algebraic invariants can be captured differently inside Cubical Agda[2], need to see how this can be used for computation.

[1] Heras, Coquand, Mörtberg 2013, *Computing Persistent Homology Within Coq/SSReflect*
[2] work by Brunerie, Cavallo, Lamiaux, Ljungström, Mörtberg on "synthetic" cohomology (rings)

Next steps

- Formalise DMT for 2-dim complexes to compute topology of grayscale images[3]

$H_1($  $) \cong \mathbb{Z}$

- Implement full pipeline in Cubical Agda: turn grayscale image into complex; compute APM; compute cohomology of reduced complex
- Refine pipeline: filtered complexes for persistent homology, cellular sheaves, ...

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

Conclusions

- Type theory is a natural language for verification of mathematical software: we can write down programs and mathematics in the same language
- Cubical Agda is a powerful (but intricate) logic for homotopy types

Conclusions

- Type theory is a natural language for verification of mathematical software: we can write down programs and mathematics in the same language
- Cubical Agda is a powerful (but intricate) logic for homotopy types

Thank you for your attention!

The Morse theorem in Cubical Agda

Morse theorem: $| \ G \ | \ \equiv \ | \ M \ |$

- establish maps back and forth, for example 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

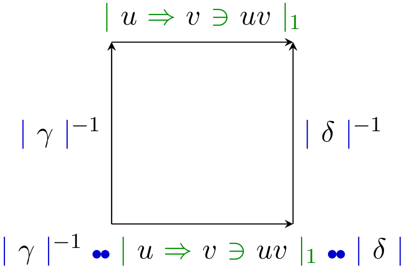
E.g., for any critical edge $\mid u \Rightarrow v \ni uv \mid_1$ show that going along $\mid E \mid \to \mid M_1 \mid \to \mid E \mid$ is coherent:

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

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

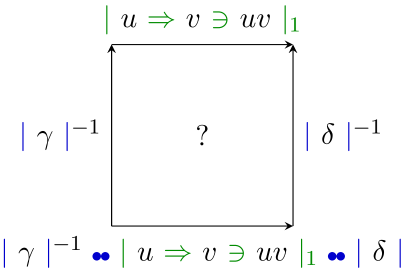
Proving the Morse Theorem

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



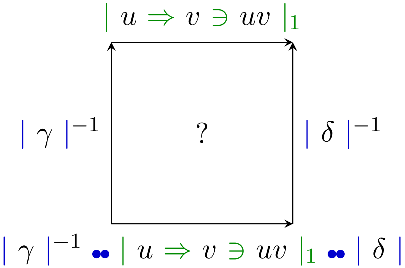
Proving the Morse Theorem

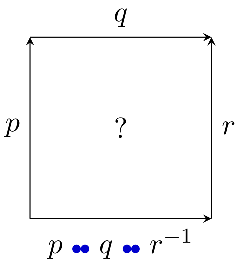
E.g., for any critical edge $\mid u \Rightarrow v \ni uv \mid_1$ show that going along $\mid E \mid \to \mid M_1 \mid \to \mid E \mid$ is coherent:



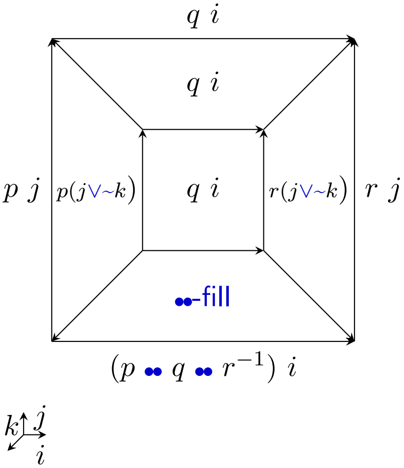
Proving the Morse Theorem

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





Kan compositions in Cubical Agda




```
lemma : (p : w ≡ x) (q : x ≡ z) (r : y ≡ z)
  → PathP (λ i → (p ∙∙ q ∙∙ sym r) i ≡ q i) p r
lemma p q r i j = hcomp (λ k → λ {
                      (i = i0) → p (j ∨ ∼ k)
                    ; (i = i1) → r (j ∨ ∼ k)
                    ; (j = i1) → q i
                    }) (q i)
```