Structural Coends and Bisimulations

MGS Participant Talks
5 April 2023

What is a cake?

What is a cake?

Heat the oven to 180C/160C fan/gas 4 and line the base of two 20cm sandwich tins with baking parchment.

What is a cake?

Heat the oven to 180C/160C fan/gas 4 and line the base of two 20cm sandwich tins with baking parchment. Beat the butter and sugar together for 3 mins using an electric whisk until smooth and fluffy.

What is a cake?

Heat the oven to 180C/160C fan/gas 4 and line the base of two 20cm sandwich tins with baking parchment. Beat the butter and sugar together for 3 mins using an electric whisk until smooth and fluffy. Add the eggs, one at a time, beating well between each addition and scraping down the sides of the bowl.

What is a cake?

Heat the oven to 180C/160C fan/gas 4 and line the base of two 20cm sandwich tins with baking parchment. Beat the butter and sugar together for 3 mins using an electric whisk until smooth and fluffy. Add the eggs, one at a time, beating well between each addition and scraping down the sides of the bowl. Fold in the flour and baking powder until well incorporated, then fold in the yogurt, vanilla and lemon zest.

What is a cake?

Heat the oven to 180C/160C fan/gas 4 and line the base of two 20cm sandwich tins with baking parchment. Beat the butter and sugar together for 3 mins using an electric whisk until smooth and fluffy. Add the eggs, one at a time, beating well between each addition and scraping down the sides of the bowl. Fold in the flour and baking powder until well incorporated, then fold in the yogurt, vanilla and lemon zest. Divide between the tins and bake for 30-35 mins until golden and a skewer inserted into the middles comes out clean.

What is a cake?

Heat the oven to 180C/160C fan/gas 4 and line the base of two 20cm sandwich tins with baking parchment. Beat the butter and sugar together for 3 mins using an electric whisk until smooth and fluffy. Add the eggs, one at a time, beating well between each addition and scraping down the sides of the bowl. Fold in the flour and baking powder until well incorporated, then fold in the yogurt, vanilla and lemon zest. Divide between the tins and bake for 30-35 mins until golden and a skewer inserted into the middles comes out clean. Meanwhile, make the drizzle.

What is a cake?

Heat the oven to 180C/160C fan/gas 4 and line the base of two 20cm sandwich tins with baking parchment. Beat the butter and sugar together for 3 mins using an electric whisk until smooth and fluffy. Add the eggs, one at a time, beating well between each addition and scraping down the sides of the bowl. Fold in the flour and baking powder until well incorporated, then fold in the yogurt, vanilla and lemon zest. Divide between the tins and bake for 30-35 mins until golden and a skewer inserted into the middles comes out clean. Meanwhile, make the drizzle. Tip the sugar, lemon juice and 100ml water into a small pan set over a medium heat and stir until dissolved.

What is a cake?

Heat the oven to 180C/160C fan/gas 4 and line the base of two 20cm sandwich tins with baking parchment. Beat the butter and sugar together for 3 mins using an electric whisk until smooth and fluffy. Add the eggs, one at a time, beating well between each addition and scraping down the sides of the bowl. Fold in the flour and baking powder until well incorporated, then fold in the yogurt, vanilla and lemon zest. Divide between the tins and bake for 30-35 mins until golden and a skewer inserted into the middles comes out clean. Meanwhile, make the drizzle. Tip the sugar, lemon juice and 100ml water into a small pan set over a medium heat and stir until dissolved. Add the lemon zest, bring to the boil and simmer for 2-3 mins until the zest has softened and the liquid is syrupy.

What is a cake?

Heat the oven to 180C/160C fan/gas 4 and line the base of two 20cm sandwich tins with baking parchment. Beat the butter and sugar together for 3 mins using an electric whisk until smooth and fluffy. Add the eggs, one at a time, beating well between each addition and scraping down the sides of the bowl. Fold in the flour and baking powder until well incorporated, then fold in the yogurt, vanilla and lemon zest. Divide between the tins and bake for 30-35 mins until golden and a skewer inserted into the middles comes out clean. Meanwhile, make the drizzle. Tip the sugar, lemon juice and 100ml water into a small pan set over a medium heat and stir until dissolved. Add the lemon zest, bring to the boil and simmer for 2-3 mins until the zest has softened and the liquid is syrupy. Remove the zest to a sheet of baking parchment using a slotted

Moral: Saying what something *is* necessarily involves saying what it's *for*

Example: natural numbers

A natural number is either zero or successor of some natural number

Example: natural numbers

A natural number is either zero or successor of some natural number

A natural number tells you how many times to iterate a given function to produce a new output

Impredicative Encodings

Impredicative Encodings :

Define a type entirely in terms of how it's used

Encoding Nat (Awodey-Frey-Speight 2018)

$$\mathbb{N} :\equiv \sum_{\phi:(X:\mathsf{Set})\to(X\to X)\to X\to X}$$

Encoding Nat (Awodey-Frey-Speight 2018)

$$\mathbb{N} :\equiv \sum_{\phi:(X:\mathrm{Set})\to(X\to X)\to X\to X} \prod_{f:X\to Y} (f(x) = y) \to (f \circ \gamma = \delta \circ f) \to f(\phi\,\gamma\,x) = \phi\,\gamma\,y$$

Goal: Impredicative encodings of *coinductive* types

F-Struct

For a profunctor $F : \mathsf{Set}^{\mathsf{op}} \times \mathsf{Set} \to \mathsf{Set}$,

F-Struct

For a profunctor $F : \mathsf{Set}^{\mathsf{op}} \times \mathsf{Set} \to \mathsf{Set}$, define the category F-Struct as

$$|F\text{-Struct}| :\equiv \sum_{C:\mathsf{Set}} F(C, C)$$

$$(C, \gamma) \to (D, \delta) :\equiv \sum_{f:C \to D} F(C, f)\, \gamma = F(f, D)\, \delta$$

Defn For $F : \mathsf{Set}^{\mathsf{op}} \times \mathsf{Set} \to \mathsf{Set}$, the **costructure integral** is defined as

$$\int^{C:\mathsf{Set}} F(C, C) \,\mathbf{p} C :\equiv \left(\sum_{(C,\gamma):F\text{-}\mathsf{Struct}} C \right) \Big/ \mathsf{Sim}_F$$

Defn For $F : \mathrm{Set}^{\mathrm{op}} \times \mathrm{Set} \to \mathrm{Set}$, the **costructure integral** is defined as

$$\int^{C:\mathrm{Set}} F(C, C)\, \mathbf{p} C :\equiv \left(\sum_{(C,\gamma):F\text{-Struct}} C \right) \Big/ \mathrm{Sim}_F$$

where Sim_F is the least equivalence relation such that

$$\prod_{(C,\gamma),(D,\delta):F\text{-Struct}} \quad \prod_{f:(C,\gamma)\to(D,\delta)} \quad \prod_{c_0:C} \mathrm{Sim}_F\ (C, \gamma, c_0)\ (D, \delta, f\ c_0)$$

Defn For $F : \mathrm{Set}^{\mathrm{op}} \times \mathrm{Set} \to \mathrm{Set}$, the **costructure integral** is defined as

$$\int^{C:\mathrm{Set}} F(C,C)\,\mathbf{p}C :\equiv \left(\sum_{(C,\gamma):F\text{-Struct}} C\right) \Big/ \mathrm{Sim}_F$$

where Sim_F is the least equivalence relation such that

$$\prod_{(C,\gamma),(D,\delta):F\text{-Struct}} \quad \prod_{f:(C,\gamma)\to(D,\delta)} \quad \prod_{c_0:C} \mathrm{Sim}_F\ (C,\gamma,c_0)\ (D,\delta,f\ c_0)$$

For each F-Struct (C,γ), there is a map

$$\mathrm{colt}\ \gamma : \quad C \to \int^{C:\mathrm{Set}} F(C,C)\,\mathbf{p}C$$

sending c to the Sim_F-equivalence class of (C,γ,c_0).

Let T : Set $\to$ Set be a functor. Then there is a profunctor F_T : Set$^{\mathrm{op}}$ $\times$ Set $\to$ Set given by

$$F_T(X, Y) :\equiv X \to T(Y)$$

The category of F_T-structures is given by

$$|F_T\text{-Struct}| :\equiv \sum_{C:\mathrm{Set}} C \to T(C)$$

$$(C, \gamma) \to (D, \delta) :\equiv \sum_{f:C \to D} T(f) \circ \gamma = \delta \circ f$$

Let $T : \mathsf{Set} \to \mathsf{Set}$ be a functor. Then there is a profunctor $F_T : \mathsf{Set}^{\mathsf{op}} \times \mathsf{Set} \to \mathsf{Set}$ given by

$$F_T(X, Y) :\equiv X \to T(Y)$$

The category of F_T-structures is given by

$$|F_T\text{-Struct}| :\equiv \sum_{C:\mathsf{Set}} C \to T(C)$$

$$(C, \gamma) \to (D, \delta) :\equiv \sum_{f:C\to D} T(f) \circ \gamma = \delta \circ f$$

F_T-structures are better known as T-**coalgebras**.

Terminal coalgebras

For any $T : \mathsf{Set} \to \mathsf{Set}$, the set

$$\nu_T :\equiv \int^{C:\mathsf{Set}} C \to T(C) \,\mathbf{p}\, C.$$

Terminal coalgebras

For any $T : \mathsf{Set} \to \mathsf{Set}$, the set

$$\nu_T :\equiv \int^{C:\mathsf{Set}} C \to T(C) \, \mathbf{p} C.$$

comes equipped with a T-coalgebra structure

$$\mathsf{out}_T : \quad \nu_T \to T(\nu_T)$$

sending the equivalence class of (C, γ, c_0) to $T(\mathsf{colt}\ \gamma)(\gamma\ c_0)$.

For any $T : \mathsf{Set} \to \mathsf{Set}$, the set

$$\nu_T :\equiv \int^{C:\mathsf{Set}} C \to T(C) \, \mathbf{p} C.$$

comes equipped with a T-coalgebra structure

$$\mathsf{out}_T : \quad \nu_T \to T(\nu_T)$$

sending the equivalence class of (C, γ, c_0) to $T(\mathsf{colt}\ \gamma)(\gamma\ c_0)$.

Claim For any T-coalgebra (C, γ), the map $\mathsf{colt}\ \gamma : C \to \nu_T$ is the unique T-coalgebra morphism $(C, \gamma) \to (\nu_T, \mathsf{out}_T)$.

Terminal coalgebras

For any $T : \mathsf{Set} \to \mathsf{Set}$, the set

$$\nu_T :\equiv \int^{C:\mathsf{Set}} C \to T(C)\,\mathbf{p}\,C.$$

comes equipped with a T-coalgebra structure

$$\mathsf{out}_T : \quad \nu_T \to T(\nu_T)$$

sending the equivalence class of (C, γ, c_0) to $T(\mathsf{colt}\ \gamma)(\gamma\ c_0)$.

Claim For any T-coalgebra (C, γ), the map $\mathsf{colt}\ \gamma : C \to \nu_T$ is the unique T-coalgebra morphism $(C, \gamma) \to (\nu_T, \mathsf{out}_T)$. Thus, (ν_T, out_T) is the **terminal T-coalgebra**.

Examples

- If $T(X) :\equiv A \times X$, then ν_T is the set $\mathsf{Stream}(A)$ of streams of elements of A

Examples

- If $T(X) :\equiv A \times X$, then ν_T is the set $\mathsf{Stream}(A)$ of streams of elements of A
  - If $\gamma : C \to A \times C$, then $\mathsf{colt}\ \gamma\ c_0$ is the infinite stream of elements of type A obtained from repeatedly applying γ to c_0

Examples

- If $T(X) :\equiv A \times X$, then ν_T is the set $\mathsf{Stream}(A)$ of streams of elements of A
  - If $\gamma : C \to A \times C$, then colt γ c_0 is the infinite stream of elements of type A obtained from repeatedly applying γ to c_0
  - $\mathsf{out}_T : \mathsf{Stream}(A) \to A \times \mathsf{Stream}(A)$ sends a stream to its head and tail

- If $T(X) :\equiv A \times X$, then ν_T is the set $\mathsf{Stream}(A)$ of streams of elements of A
  - If $\gamma : C \to A \times C$, then colt $\gamma\ c_0$ is the infinite stream of elements of type A obtained from repeatedly applying γ to c_0
  - $\mathsf{out}_T : \mathsf{Stream}(A) \to A \times \mathsf{Stream}(A)$ sends a stream to its head and tail
- If $T(X) :\equiv 1 + X$, then $\nu_T \equiv \mathbb{N}^\infty$, the set of conatural numbers
- If $T(X) :\equiv 1 + A \times X$ for some set A, then ν_T is the set of co-lists of elements of A

Defn. A **bisimulation** between F-structures (C, γ) to (D, δ) is a binary relation $\mathcal{R} \colon C \to D \to \mathrm{Prop}$

Defn. A **bisimulation** between F-structures (C, γ) to (D, δ) is a binary relation $\mathcal{R} \colon C \to D \to \mathrm{Prop}$ equipped with an F-structure $\alpha_{\mathcal{R}} : F(\overline{\mathcal{R}}, \overline{\mathcal{R}})$, where

$$\overline{\mathcal{R}} :\equiv \sum_{c:C} \sum_{d:D} \mathcal{R}\ c\ d$$

Defn. A **bisimulation** between F-structures (C, γ) to (D, δ) is a binary relation $\mathcal{R} \colon C \to D \to \mathrm{Prop}$ equipped with an F-structure $\alpha_{\mathcal{R}} : F(\overline{\mathcal{R}}, \overline{\mathcal{R}})$, where

$$\overline{\mathcal{R}} :\equiv \sum_{c:C} \sum_{d:D} \mathcal{R}\ c\ d$$

such that the projection maps $\mathrm{pr}_1 : \overline{\mathcal{R}} \to C$ and $\mathrm{pr}_2 : \overline{\mathcal{R}} \to D$ are F-structure morphisms.

Defn. A **bisimulation** between F-structures (C, γ) to (D, δ) is a binary relation $\mathcal{R} \colon C \to D \to \text{Prop}$ equipped with an F-structure $\alpha_{\mathcal{R}} : F(\overline{\mathcal{R}}, \overline{\mathcal{R}})$, where

$$\overline{\mathcal{R}} :\equiv \sum_{c:C} \sum_{d:D} \mathcal{R}\ c\ d$$

such that the projection maps $\text{pr}_1 : \overline{\mathcal{R}} \to C$ and $\text{pr}_2 : \overline{\mathcal{R}} \to D$ are F-structure morphisms.

Observe If $c : C$ and $d : D$ are related by a bisimulation $\mathcal{R}$ (i.e. $\mathcal{R}\ c\ d$ is inhabited), then (C, γ, c) and (D, δ, d) are in the same equivalence class of $\nu :\equiv \int^{X:\text{Set}} F(X, X)\ \mathbf{p}X$.

Defn. A **bisimulation** between F-structures (C, γ) to (D, δ) is a binary relation $\mathcal{R} \colon C \to D \to \mathsf{Prop}$ equipped with an F-structure $\alpha_{\mathcal{R}} : F(\overline{\mathcal{R}}, \overline{\mathcal{R}})$, where

$$\overline{\mathcal{R}} :\equiv \sum_{c:C} \sum_{d:D} \mathcal{R}\ c\ d$$

such that the projection maps $\mathrm{pr}_1 : \overline{\mathcal{R}} \to C$ and $\mathrm{pr}_2 : \overline{\mathcal{R}} \to D$ are F-structure morphisms.

Observe If $c : \nu$ and $d : \nu$ are related by a bisimulation $\mathcal{R}$ (i.e. $\mathcal{R}\ c\ d$ is inhabited), then c and d are equal elements of $\nu :\equiv \int^{X:\mathsf{Set}} F(X, X)\ \mathbf{p}X$.

Defn. A **bisimulation** between F-structures (C, γ) to (D, δ) is a binary relation $\mathcal{R} \colon C \to D \to \text{Prop}$ equipped with an F-structure $\alpha_{\mathcal{R}} : F(\overline{\mathcal{R}}, \overline{\mathcal{R}})$, where

$$\overline{\mathcal{R}} :\equiv \sum_{c:C} \sum_{d:D} \mathcal{R}\ c\ d$$

such that the projection maps $\mathrm{pr}_1 : \overline{\mathcal{R}} \to C$ and $\mathrm{pr}_2 : \overline{\mathcal{R}} \to D$ are F-structure morphisms.

Principle of Coinduction If $c : \nu$ and $d : \nu$ are related by a bisimulation $\mathcal{R}$ (i.e. $\mathcal{R}\ c\ d$ is inhabited), then c and d are equal elements of $\nu :\equiv \int^{X:\mathrm{Set}} F(X, X)\ \mathbf{p}X$.

Reason for the extra generality

Reason for the extra generality : Can also use to encode 'existential' types

Thank you!