

Combinatorial Species and Labelled Structures

Brent Yorgey

PhD dissertation defense
October 14, 2014



Introduction

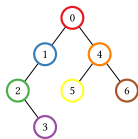
A Tale of Two Cities Worlds

A Tale of Two Cities Worlds

Combinatorics

1, 2, 3, 4, 5, ...

$$\sum_{n \geq 0} f_n \frac{x^n}{n!}$$

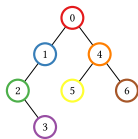


A Tale of Two Cities Worlds

Combinatorics

1, 2, 3, 4, 5, ...

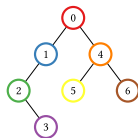
$$\sum_{n \geq 0} f_n \frac{x^n}{n!}$$



Programming Languages

$\lambda f : \tau \rightarrow \tau. \lambda a : \tau. f a$

data Tree a
= Empty | ...

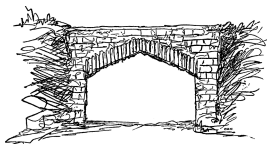
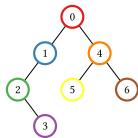


A Tale of Two Cities Worlds

Combinatorics

1, 2, 3, 4, 5, ...

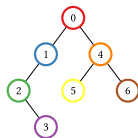
$$\sum_{n \geq 0} f_n \frac{x^n}{n!}$$



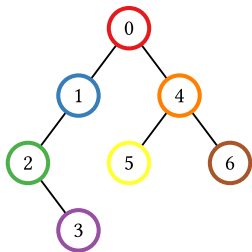
Programming Languages

$\lambda f : \tau \rightarrow \tau. \lambda a : \tau. f a$

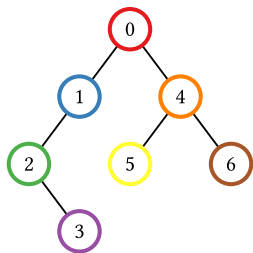
data Tree a
= Empty | ...



Example: trees



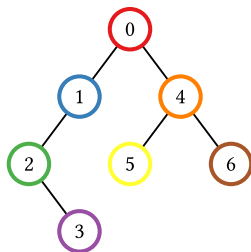
Example: trees



$$T = 1 + X \cdot T^2$$

$$T(x) = 1 + x + 2x^2 + 5x^3 + \dots$$

Example: trees

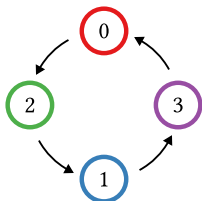


$$T = 1 + X \cdot T^2$$
$$T(x) = 1 + x + 2x^2 + 5x^3 + \dots$$

data *Tree a*
= *Empty*
| *Node a (Tree a) (Tree a)*

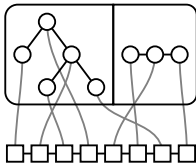
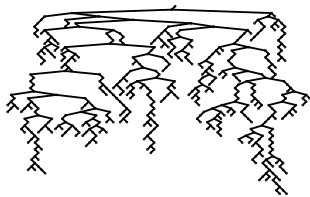
Why is this interesting?

- Species that are not algebraic data types



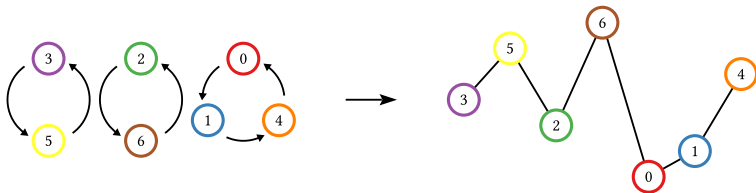
Why is this interesting?

- Tools for working with existing algebraic data types



Why is this interesting?

- Additional insight into the mathematics



Contributions

- **Port** species to PL-world
- **Categorical requirements** for operations on generalized species
- “Labelled structures” via **analytic functors**
- Exposition of species for a PL audience
- ... and many smaller things!

Outline

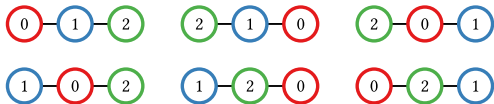
- Species—examples and formal definition
- “Bringing species across the bridge” —species in homotopy type theory
- Labelled structures
- Conclusions and future work

Species

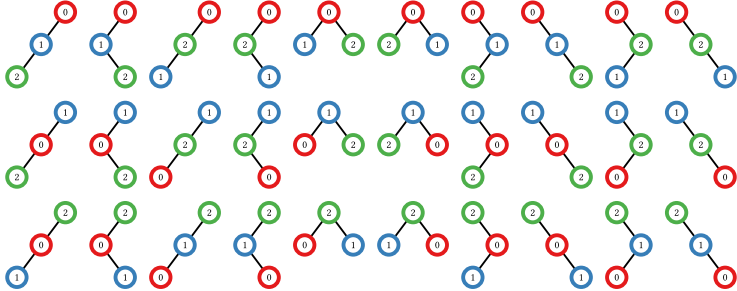
Intuition and examples

A species is a **family of labelled shapes**.

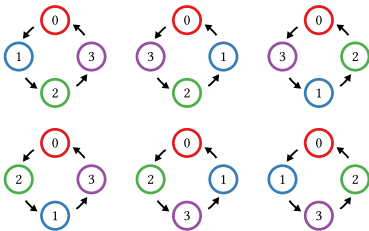
Examples



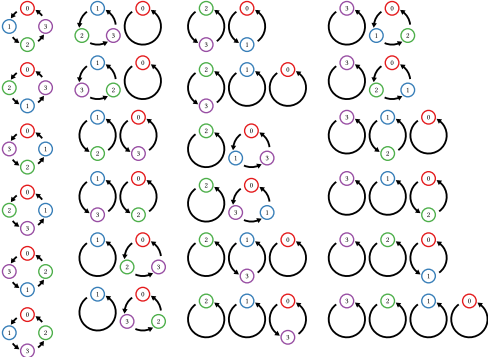
Examples



Examples

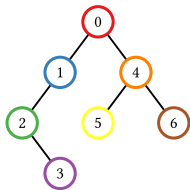


Examples



Labels

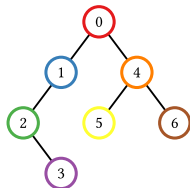
A species is a family of **labelled** shapes.



Labels

A species is a family of **labelled** shapes.

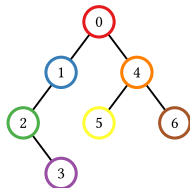
- Labels are names for **locations**



Labels

A species is a family of **labelled** shapes.

- Labels are names for **locations**
- Labels are taken from a **finite** set



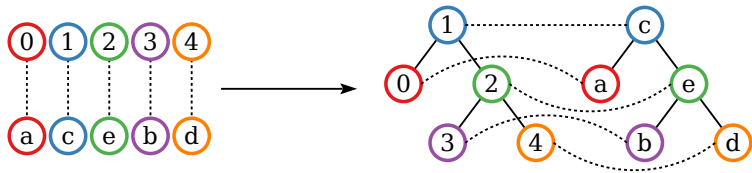
Relabelling

A species is a **family** of labelled shapes.

Relabelling

A species is a **family** of labelled shapes.

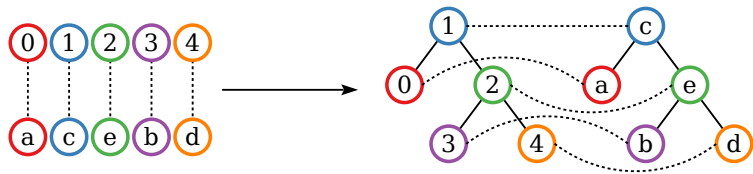
We must be able to **relabel**:



Relabelling

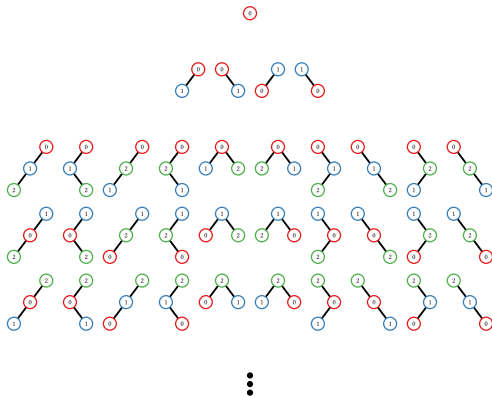
A species is a **family** of labelled shapes.

We must be able to **relabel**:



Relabelling must be **functorial**.

Size



Size (*i.e.* number of labels) is preserved by relabelling.

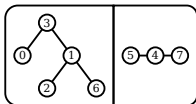
The algebra of species

The algebra of species

- Sum: $(F + G) L = F L \uplus G L$

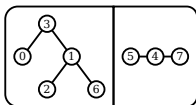
The algebra of species

- Sum: $(F + G) L = F L \uplus G L$
- Product:



The algebra of species

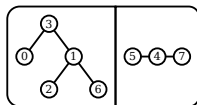
- Sum: $(F + G) L = F L \uplus G L$
- Product:



- 0, 1, X, E, C, ...

The algebra of species

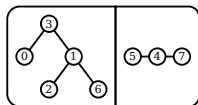
- Sum: $(F + G) L = F L \uplus G L$
- Product:



- 0, 1, X, E, C, ...
- Cartesian product, arithmetic product, composition, derivative, ...

The algebra of species

- Sum: $(F + G) L = F L \uplus G L$
- Product:



- $0, 1, X, E, C, \dots$
- Cartesian product, arithmetic product, composition, derivative, \dots

$$T = 1 + X \cdot T^2$$

Species, formally

A species F consists of

- a function sending finite sets of labels L to sets of shapes $F L$

Species, formally

A species F consists of

- a function sending finite sets of labels L to sets of shapes $F L$
- a function sending bijections $\sigma : L \xrightarrow{\sim} L'$ to relabellings $F L \rightarrow F L'$ (satisfying laws)

Species, even more formally

A species F is a functor $F : \mathbf{B} \rightarrow \mathbf{Set}$.

Species, even more formally

A species F is a functor $F : \mathbf{B} \rightarrow \mathbf{Set}$.

- \mathbf{B} : category of finite sets and bijections
- \mathbf{Set} : category of sets and total functions

Generalized species

... a **generalized species** is a functor $\mathfrak{L} \rightarrow \mathfrak{S}$?

What properties must \mathfrak{L} and \mathfrak{S} have?

Generalized species

... a **generalized species** is a functor $\mathfrak{L} \rightarrow \mathfrak{S}$?

What properties must \mathfrak{L} and \mathfrak{S} have?

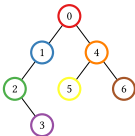
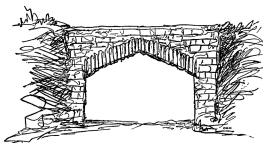
See Chapter 3!

Porting species to type theory

Combinatorics

1, 2, 3, 4, 5, ...

$$\sum_{n \geq 0} f_n \frac{x^n}{n!}$$



Programming Languages

$\lambda f : \tau \rightarrow \tau. \lambda a : \tau. f a$

data Tree a
= Empty | ...



The problem

Like most mathematics, the theory of species

- uses classical logic
- is based on set theory
- is untyped

The problem

Like most mathematics, the theory of species

- uses classical logic
- is based on set theory
- is untyped

... but PL World

- uses constructive logic
- is based on type theory
- is typed.

The problem

Like most mathematics, the theory of species

- uses classical logic
- is based on set theory
- is untyped

... but PL World

- uses constructive logic
- is based on type theory
- is typed.

Goal: encode species into a constructive type theory.

Homotopy type theory (HoTT)

Recent variant type theory developed by Voevodsky *et al.*

Homotopy type theory (HoTT)

Recent variant type theory developed by Voevodsky *et al.*

Claim: HoTT makes a good (constructive!) foundation for mathematics.

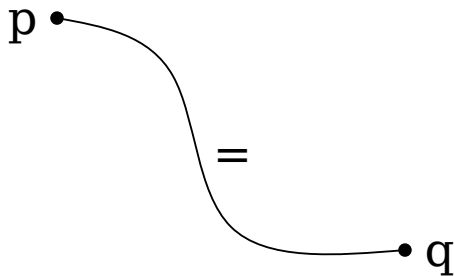
Homotopy type theory (HoTT)

Recent variant type theory developed by Voevodsky *et al.*

Claim: HoTT makes a good (constructive!) foundation for mathematics.

... Let's try it!

Equality in HoTT



Topological intuition: equality is like a path (homotopy) in a space. Equality can have **computational content**.

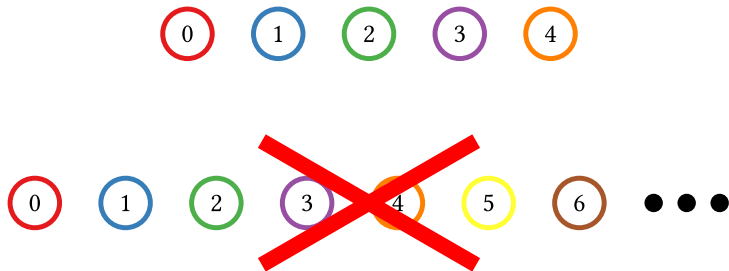
Univalence

$$(A = B) \simeq (A \simeq B)$$

Allows unifying equality and isomorphism—just what the doctor ordered for species!

Finiteness

Species map **finite sets** of labels to sets of shapes. How to model finite sets in constructive type theory?



Cardinal-finiteness: first try

A set A is **cardinal-finite** if there exists some $n \in \mathbb{N}$ and a bijection $\{0, 1, \dots, n - 1\} \xrightarrow{\sim} A$.

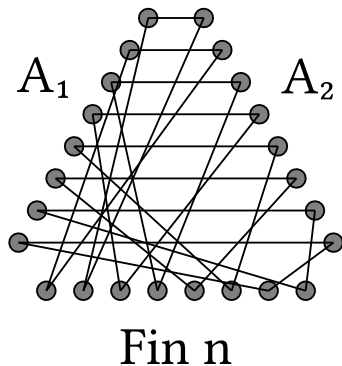
Cardinal-finiteness: first try

A set A is **cardinal-finite** if there exists some $n \in \mathbb{N}$ and a bijection $\{0, 1, \dots, n - 1\} \xrightarrow{\sim} A$.

$$\mathcal{U}_{\text{Fin}} := (A : \mathcal{U}) \times (n : \mathbb{N}) \times (\text{Fin } n \simeq A)$$

Cardinal-finiteness: first try

But this is too strong! There is **at most one** isomorphism between two finite types.



Propositional truncation to the rescue

Given a type A , its **propositional truncation** $\|A\|$ is like A , except (by fiat) all inhabitants are considered equal.

Propositional truncation to the rescue

Given a type A , its **propositional truncation** $\|A\|$ is like A , except (by fiat) all inhabitants are considered equal.

$$\mathcal{U}_{\|\text{Fin}\|} \equiv (A : \mathcal{U}) \times \|(n : \mathbb{N}) \times (\text{Fin } n \simeq A)\|$$

Propositional truncation to the rescue

Given a type A , its **propositional truncation** $\|A\|$ is like A , except (by fiat) all inhabitants are considered equal.

$$\mathcal{U}_{\|\text{Fin}\|} \equiv (A : \mathcal{U}) \times \|(n : \mathbb{N}) \times (\text{Fin } n \simeq A)\|$$

Akin to using an existential type:

$$(A : \mathcal{U}) \times (\exists n : \mathbb{N}. \text{Fin } n \simeq A)$$

Propositional truncation to the rescue

Given a type A , its **propositional truncation** $\|A\|$ is like A , except (by fiat) all inhabitants are considered equal.

$$\mathcal{U}_{\|\text{Fin}\|} \equiv (A : \mathcal{U}) \times \|(n : \mathbb{N}) \times (\text{Fin } n \simeq A)\|$$

Akin to using an existential type:

$$(A : \mathcal{U}) \times (\exists n : \mathbb{N}. \text{Fin } n \simeq A)$$

... but the **induction principle** for $\|A\|$ still allows the contents to be used in ways that do not “leak” information.

Propositional truncation to the rescue

We can use $\mathcal{U}_{\|\text{Fin}\|}$ to make a HoTT analogue of \mathbf{B} , which

- seems to have all the right properties
- yields insight into the mathematics (equipotence, \mathbf{L} -species).

In HoTT, it is **impossible to write down invalid species**.

More...

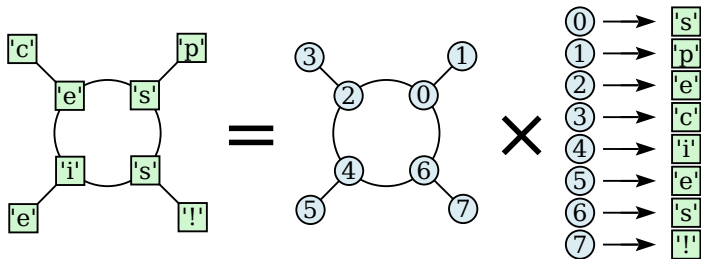
See dissertation for:

- How HoTT solves other issues (e.g. axiom of choice)
- More on category theory, finiteness, and related issues in HoTT

Labelled structures

Data structure = shape + data

A species gives us **labelled shapes**. To make a **data structure** we pair a shape with a mapping from labels to data values.



$$F L \times (L \rightarrow A)$$

Analytic functors

The labels “shouldn't matter” so we quotient by relabelling:

$$\widehat{F} A := \exists L. F L \times (L \rightarrow A)$$

Analytic functors

The labels “shouldn't matter” so we quotient by relabelling:

$$\widehat{F} A := \exists L. F L \times (L \rightarrow A)$$

\widehat{F} is an **analytic functor** (Joyal, 1986).

Analytic functors

The labels “shouldn't matter” so we quotient by relabelling:

$$\widehat{F} A := \exists L. F L \times (L \rightarrow A)$$

\widehat{F} is an **analytic functor** (Joyal, 1986).

Analytic functors:

Analytic functors

The labels “shouldn't matter” so we quotient by relabelling:

$$\widehat{F} A := \exists L. F L \times (L \rightarrow A)$$

\widehat{F} is an **analytic functor** (Joyal, 1986).

Analytic functors:

- correspond to generating functions

Analytic functors

The labels “shouldn't matter” so we quotient by relabelling:

$$\widehat{F} A := \exists L. F L \times (L \rightarrow A)$$

\widehat{F} is an **analytic functor** (Joyal, 1986).

Analytic functors:

- correspond to generating functions
- are closed under many operations of interest (sum, product, fixed point . . .)

Analytic functors

The labels “shouldn't matter” so we quotient by relabelling:

$$\widehat{F} A := \exists L. F L \times (L \rightarrow A)$$

\widehat{F} is an **analytic functor** (Joyal, 1986).

Analytic functors:

- correspond to generating functions
- are closed under many operations of interest (sum, product, fixed point . . .)
- can be defined via **Kan extensions**

Analytic functors

The labels “shouldn't matter” so we quotient by relabelling:

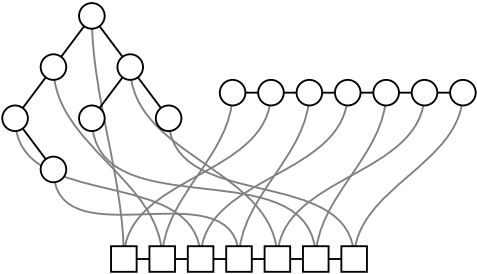
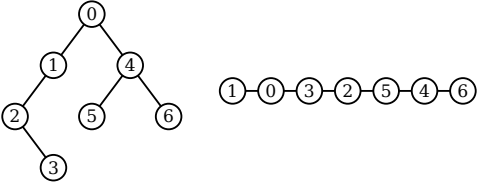
$$\widehat{F} A := \exists L. F L \times (L \rightarrow A)$$

\widehat{F} is an **analytic functor** (Joyal, 1986).

Analytic functors:

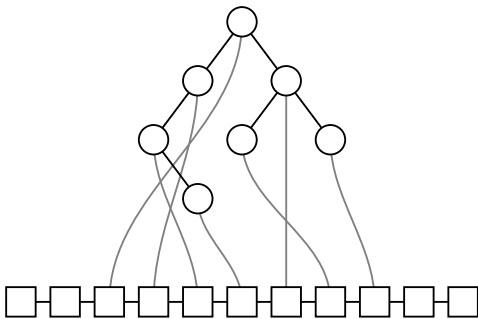
- correspond to generating functions
- are closed under many operations of interest (sum, product, fixed point . . .)
- can be defined via **Kan extensions**
- correspond to Σ -types in HoTT

Labels as memory addresses



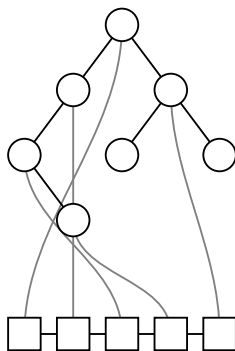
Labels as memory addresses

Promise: instantiate framework with right functor category $\mathfrak{L} \rightarrow \mathfrak{G}$, and we get species operations and data structures (analytic functors) for free.



\mathcal{B}_{\subseteq} —variant of \mathbf{B} where morphisms are **injections** rather than bijections.

Labels as memory addresses

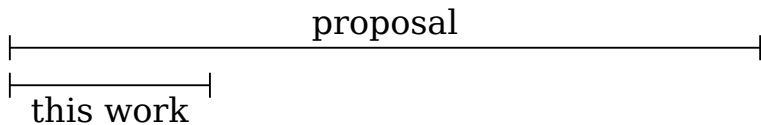


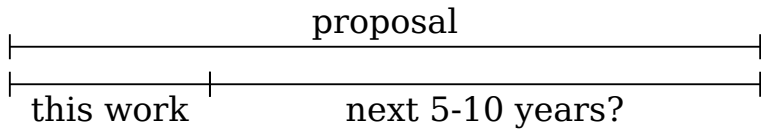
$$\mathcal{B}_{\subseteq}^{\text{op}}$$

Conclusions and future work



proposal





Future work

- Computational aspects of generating functions
- Formalizing theory of molecular and atomic species
- Applications to generic programming
- Applications to memory layout
- Connections to linear logic?

Thank you!

