Categorical Proofs are Natural Proofs

David G. Berry j.w.w. Marcelo P. Fiore

University of Cambridge

Joint WG4 & WG5 Meeting
University of Cambridge
September 2023

Outline

Background Motivation

- STLC is (weakly) normalizing.
- Constructive setting: we want to compute algorithmically the normal form.

Background Motivation

- STLC is (weakly) normalizing.
- Constructive setting: we want to compute algorithmically the normal form.
- Standard Strategy:
  1. Construct Algorithm
  2. Prove Correctness

Background Motivation

- STLC is (weakly) normalizing.
- Constructive setting: we want to compute algorithmically the normal form.
- Standard Strategy:
  1. Construct Algorithm
  2. Prove Correctness
- Standard Approaches:
  1. Algorithm: Normalization by Evaluation
  2. Correctness: Logical Relations

Background Motivation

- STLC is (weakly) normalizing.
- Constructive setting: we want to compute algorithmically the normal form.
- Standard Strategy:
  1. Construct Algorithm
  2. Prove Correctness
- Standard Approaches:
  1. Algorithm: Normalization by Evaluation
  2. Correctness: Logical Relations
- Alternative Approach: Category Theory

The Challenge of Normalization

Problem

Extensionally, normalization does nothing! The output of nf is equal to the input. $\therefore$ nf $=$ id.

The Challenge of Normalization

Problem

Extensionally, normalization does nothing! The output of nf is equal to the input. $\therefore \mathsf{nf} = \mathsf{id}$.

Fraction Simplification

$$\mathsf{nf}\left(\frac{a}{b}\right) \triangleq \frac{a \div \gcd(a,b)}{b \div \gcd(a,b)}$$

$$\mathsf{nf}\left(\frac{4}{8}\right) \equiv \frac{1}{2} =_{\mathbb{Q}^+} \frac{4}{8} \neq_{\mathbb{N}\times\mathbb{N}^+} \frac{1}{2}$$

The Challenge of Normalization

Problem

Extensionally, normalization does nothing! The output of nf is equal to the input. $\therefore \mathsf{nf} = \mathsf{id}$.

Fraction Simplification

$$\mathsf{nf}\left(\frac{a}{b}\right) \triangleq \frac{a \div \gcd(a,b)}{b \div \gcd(a,b)}$$

$$\mathsf{nf}\left(\frac{4}{8}\right) \equiv \frac{1}{2} =_{\mathbb{Q}^+} \frac{4}{8} \neq_{\mathbb{N}\times\mathbb{N}^+} \frac{1}{2}$$

Observation

- Extensionally, normalization is the identity. Indeed, this is a correctness property!
- Intensionally, normalization is not the identity. Indeed, this is why it is algorithmically useful!

The Problem with Category Theory

- Standard Category Theory is a traditional Mathematical theory.
  - Easy to express extensional properties.
  - Difficult to express intensional computational; behaviour.
- "All properties and no computation!"

The Problem with Category Theory

- Standard Category Theory is a traditional Mathematical theory.
  - Easy to express extensional properties.
  - Difficult to express intensional computational; behaviour.
- "All properties and no computation!"

Desideratum

We want an alternate form of Category Theory which can *naturally* model:

- extensional properties;
- intensional computational behaviour; and
- partiality.

Category Theory Primer

A *category* is given by the following data:

- a collection of *objects*;
- for any two objects a collection of *morphisms*, $x \to y$;
- a composition operation, $f \circ g : x \xrightarrow{g} y \xrightarrow{f} z$; and
- an identity morphism, $\mathrm{id} : x \to x$;

such that

- composition is associative: $(f \circ g) \circ h = f \circ (g \circ h)$; and
- the identity morphism is both left and right neutral: $\mathrm{id} \circ f = f \wedge f = f \circ \mathrm{id}$.

Category Theory Primer

A *category* is given by the following data:

- a collection of *objects*;
- for any two objects a collection of *morphisms*, $x \to y$;
- a composition operation, $f \circ g : x \xrightarrow{g} y \xrightarrow{f} z$; and
- an identity morphism, $\mathrm{id} : x \to x$;

such that

- composition is associative: $(f \circ g) \circ h = f \circ (g \circ h)$; and
- the identity morphism is both left and right neutral: $\mathrm{id} \circ f = f \wedge f = f \circ \mathrm{id}$.

The use of equality ("=") means that often the collection of morphisms has to be quotiented. We therefore lose all computational behaviour when extensional properties are forced onto structure.

E-Category Theory Primer

An *E-category* is given by the following data:

- a collection of *objects*;
- for any two objects a collection of *morphisms*, $x \to y$, equipped with an *equivalence relation*;
- a composition operation: $f \circ g : x \xrightarrow{g} y \xrightarrow{f} z$; and
- an identity morphism: $\mathrm{id} : x \to x$;

such that

- composition respects the ER: $f \sim f' \wedge g \sim g' \Rightarrow (f \circ g) \sim (f' \circ g')$;
- composition is associative: $(f \circ g) \circ h \sim f \circ (g \circ h)$; and
- the identity morphism is both left and right neutral: $\mathrm{id} \circ f \sim f \wedge f \sim f \circ \mathrm{id}$.

E-Category Theory Primer

An *E-category* is given by the following data:

- a collection of *objects*;
- for any two objects a collection of *morphisms*, $x \to y$, equipped with an *equivalence relation*;
- a composition operation: $f \circ g : x \xrightarrow{g} y \xrightarrow{f} z$; and
- an identity morphism: $\mathrm{id} : x \to x$;

such that

- composition respects the ER: $f \sim f' \wedge g \sim g' \Rightarrow (f \circ g) \sim (f' \circ g')$;
- composition is associative: $(f \circ g) \circ h \sim f \circ (g \circ h)$; and
- the identity morphism is both left and right neutral: $\mathrm{id} \circ f \sim f \wedge f \sim f \circ \mathrm{id}$.

The use of an equivalence relation ("$\sim$") means that the collection of morphisms does not need to be quotiented. We regain computational behaviour.

E-Category Theory Primer

An *E-category* is given by the following data:

- a collection of *objects*;
- for any two objects a collection of *morphisms*, $x \to y$, equipped with an *equivalence relation*;
- a composition operation: $f \circ g : x \xrightarrow{g} y \xrightarrow{f} z$; and
- an identity morphism: $\mathrm{id} : x \to x$;

such that

- composition respects the ER: $f \sim f' \wedge g \sim g' \Rightarrow (f \circ g) \sim (f' \circ g')$;
- composition is associative: $(f \circ g) \circ h \sim f \circ (g \circ h)$; and
- the identity morphism is both left and right neutral: $\mathrm{id} \circ f \sim f \wedge f \sim f \circ \mathrm{id}$.

The use of an equivalence relation ("$\sim$") means that we need to prove that all operations respect the appropriate ER. We enter "setoid hell". Fails to model partiality with much elegance.

P-Category Theory Primer

A *P-category* is given by the following data:

- a collection of *objects*;
- for any two objects a collection of *morphisms*, $x \to y$, equipped with a ***partial*** *equivalence relation*;
- a composition operation: $f \circ g : x \xrightarrow{g} y \xrightarrow{f} z$; and
- an identity morphism: $\mathrm{id} : x \to x$;

such that

- composition respects the PER: $f \sim f' \wedge g \sim g' \Rightarrow (f \circ g) \sim (f' \circ g')$;
- the identity morphism is self-related: $\mathrm{id} \sim \mathrm{id}$;
- composition is associative: $f \sim f' \wedge g \sim g' \wedge h \sim h' \Rightarrow (f \circ g) \circ h \sim f' \circ (g' \circ h')$; and
- the identity morphism is both left and right neutral: $f \sim f' \Rightarrow \mathrm{id} \circ f \sim f' \wedge f' \sim f \circ \mathrm{id}$.

P-Category Theory Primer

A *P-category* is given by the following data:

- a collection of *objects*;
- for any two objects a collection of *morphisms*, $x \to y$, equipped with a ***partial*** *equivalence relation*;
- a composition operation: $f \circ g : x \xrightarrow{g} y \xrightarrow{f} z$; and
- an identity morphism: $\mathrm{id} : x \to x$;

such that

- composition respects the PER: $f \sim f' \wedge g \sim g' \Rightarrow (f \circ g) \sim (f' \circ g')$;
- the identity morphism is self-related: $\mathrm{id} \sim \mathrm{id}$;
- composition is associative: $f \sim f' \wedge g \sim g' \wedge h \sim h' \Rightarrow (f \circ g) \circ h \sim f' \circ (g' \circ h')$; and
- the identity morphism is both left and right neutral: $f \sim f' \Rightarrow \mathrm{id} \circ f \sim f' \wedge f' \sim f \circ \mathrm{id}$.

The use of a **partial** ER ("$\sim$") means that we need to prove that all operations respect the appropriate PER, but now we have more hypotheses and lack reflexivity. We enter "subsetoid hell".

What about formalizing P-category theory?

Formalization Decisions

- In Coq (I already had experience with it)
- Universe Polymorphism (We need to perform constructions at various size levels)
- Proof-Relevant PERs (Allows extraction of both program for computation, and correctness proofs for certification)

Formalization Experience

- Most simple constructions are not that much more complicated than E-category theory, or even rigorous pen-and-paper category theory.
- Some arguments even become more elegant and/or efficient:
  - associativity law combines two E-steps into one; and
  - unit laws combine two E-steps into one.
- Naturality (in the categorical sense) conditions become cumbersome and tedious pretty quickly.

Functor Categories

- Functor Categories are a standard categorical construction.
- They are the categorical generalization of "a set of functions between two sets".
  - "a category of functors between two categories"
- Objects are functors ($\approx$ structure preserving maps between categories).
- Morphisms are natural transformations ($\approx$ objectwise-family of morphisms between functors).
- Crucially they must satisfy a *naturality* condition.

Functor Categories

- Functor Categories are a standard categorical construction.
- They are the categorical generalization of "a set of functions between two sets".
  - "a category of functors between two categories"
- Objects are functors ($\approx$ structure preserving maps between categories).
- Morphisms are natural transformations ($\approx$ objectwise-family of morphisms between functors).
- Crucially they must satisfy a *naturality* condition.

P-Functor Category

- Objects are P-functors
- Morphisms are:
  - represented by not-necessarily-natural transformations; and
  - related when *both* are natural, and are objectwise related.*

Formalizing the Presheaf Exponential

- Presheaf categories are an instance of a functor category.
- They have an exponential ($\approx$ the collection of morphisms between two objects can be modelled by an object of the category).
- This is an important construction for our work, and is somewhat non-trivial.
- The definition of an exponential utilises natural transformations between functors valued in functor categories.
- This means that:

Subsetoid Hell?

```
forall x : PFun (POppCat C) PSet, (forall a a' : forall x0 : C, x x0 -> forall x1 : C, PCat_hom x1 x0 -> F x1 -> G x1, IsPNatural (
POppCat C) PSet x (PPshfExp G F) a * IsPNatural (POppCat C) PSet x (PPshfExp G F) a' * (forall (x0 : C) (a0 a'0 : x x0), a0 ~ a'0 -> (
forall (x1 y : C) (f f' : PCat_hom x1 y), f ~ f' -> forall k k' : PCat_hom y x0, k ~ k' -> forall s s' : F y, s ~ s' -> PFun_hom_of G f
(a x0 a0 y (PCat_comp (PCat_comp (PCat_id_mor x0) k) (PCat_id_mor y)) (PFun_hom_of F (PCat_id_mor y) s)) ~ PFun_hom_of G (PCat_id_mor x1
) (a x0 a0 x1 (PCat_comp (PCat_comp (PCat_id_mor x0) k') f') (PFun_hom_of F f' s'))) * (forall (x1 y : C) (f f' : PCat_hom x1 y), f ~ f'
 -> forall k k' : PCat_hom y x0, k ~ k' -> forall s s' : F y, s ~ s' -> PFun_hom_of G f (a' x0 a'0 y (PCat_comp (PCat_comp (PCat_id_mor
x0) k) (PCat_id_mor y)) (PFun_hom_of F (PCat_id_mor y) s)) ~ PFun_hom_of G (PCat_id_mor x1) (a' x0 a'0 x1 (PCat_comp (PCat_comp (
PCat_id_mor x0) k') f') (PFun_hom_of F f' s'))) * (forall (x1 : C) (k k' : PCat_hom x1 x0), k ~ k' -> forall s s' : F x1, s ~ s' -> a x0
 a0 x1 k s ~ a' x0 a'0 x1 k' s')) -> IsPNatural (POppCat C) PSet (PCompFun PCartProdFun (PPairFun x F)) G (fun (x0 : C) (X : x x0 * F x0
) => a x0 (fst X) x0 (PCat_id_mor x0) (snd X)) * IsPNatural (POppCat C) PSet (PCompFun PCartProdFun (PPairFun x F)) G (fun (x0 : C) (X :
 x x0 * F x0) => a' x0 (fst X) x0 (PCat_id_mor x0) (snd X)) * (forall (x0 : C) (a0 a'0 : x x0 * F x0), (fst a0 ~ fst a'0) * (snd a0 ~
snd a'0) -> a x0 (fst a0) x0 (PCat_id_mor x0) (snd a0) ~ a' x0 (fst a'0) x0 (PCat_id_mor x0) (snd a'0))) * (forall a a' : forall x0 : C,
 x x0 * F x0 -> G x0, IsPNatural (POppCat C) PSet (PCompFun PCartProdFun (PPairFun x F)) G a * IsPNatural (POppCat C) PSet (PCompFun
PCartProdFun (PPairFun x F)) G a' * (forall (x0 : C) (a0 a'0 : x x0 * F x0), (fst a0 ~ fst a'0) * (snd a0 ~ snd a'0) -> a x0 a0 ~ a' x0
a'0) -> IsPNatural (POppCat C) PSet x (PPshfExp G F) (fun (x0 : C) (s : x x0) (z : C) (f : PCat_hom z x0) (r : F z) => a z (PFun_hom_of
x f s, r)) * IsPNatural (POppCat C) PSet x (PPshfExp G F) (fun (x0 : C) (s : x x0) (z : C) (f : PCat_hom z x0) (r : F z) => a' z (
PFun_hom_of x f s, r)) * (forall (x0 : C) (a0 a'0 : x x0), a0 ~ a'0 -> (forall (x1 y : C) (f f' : PCat_hom x1 y), f ~ f' -> forall k k'
: PCat_hom y x0, k ~ k' -> forall s s' : F y, s ~ s' -> PFun_hom_of G f (a y (PFun_hom_of x (PCat_comp (PCat_comp (PCat_id_mor x0) k) (
PCat_id_mor y)) a0, PFun_hom_of F (PCat_id_mor y) s)) ~ PFun_hom_of G (PCat_id_mor x1) (a x1 (PFun_hom_of x (PCat_comp (PCat_comp (
PCat_id_mor x0) k') f') a0, PFun_hom_of F f' s'))) * (forall (x1 y : C) (f f' : PCat_hom x1 y), f ~ f' -> forall k k' : PCat_hom y x0, k
 ~ k' -> forall s s' : F y, s ~ s' -> PFun_hom_of G f (a' y (PFun_hom_of x (PCat_comp (PCat_comp (PCat_id_mor x0) k) (PCat_id_mor y)) a
'0, PFun_hom_of F (PCat_id_mor y) s)) ~ PFun_hom_of G (PCat_id_mor x1) (a' x1 (PFun_hom_of x (PCat_comp (PCat_comp (PCat_id_mor x0) k')
f') a'0, PFun_hom_of F f' s'))) * (forall (x1 : C) (k k' : PCat_hom x1 x0), k ~ k' -> forall s s' : F x1, s ~ s' -> a x1 (PFun_hom_of x
k a0, s) ~ a' x1 (PFun_hom_of x k' a'0, s'))))
```

times ??? (This is but a small glimpse!)

A Categorical Proof

$$
\begin{aligned}
\widehat{\mathbb{C}}\left(K, G^F\right) &\cong \int_c \mathbf{Set}\left(Kc, G^F c\right) \equiv \int_c \mathbf{Set}\left(Kc, \int_{c'} \mathbb{C}(c', c) \Rightarrow Fc' \Rightarrow Gc'\right) \\
&\cong \int_c \int_{c'} \mathbf{Set}\Big(Kc, \mathbb{C}(c', c) \Rightarrow Fc' \Rightarrow Gc'\Big) \\
&\cong \int_{c'} \int_c \mathbf{Set}\Big(Kc, \mathbb{C}(c', c) \Rightarrow Fc' \Rightarrow Gc'\Big) \\
&\cong \int_{c'} \int_c \mathbf{Set}\big(Kc \times \mathbb{C}(c', c), Fc' \Rightarrow Gc'\big) \\
&\cong \int_{c'} \mathbf{Set}\left(\int^c Kc \times \mathbb{C}(c', c), Fc' \Rightarrow Gc'\right) \\
&\cong \int_{c'} \mathbf{Set}(Kc', Fc' \Rightarrow Gc') \\
&\cong \int_{c'} \mathbf{Set}(Kc' \times Fc', Gc') \cong \widehat{\mathbb{C}}(K \times F, G)
\end{aligned}
$$

Advantages

- High-level, compositional, abstract proof: good for properties!
- Each categorical step is a useful categorical building block.
  - More generally useful results justify complexity;
  - Although, often simpler/more mechanical.
- Sheds more light on P-category theory.
- Easy to communicate to category theorists (*i.e., natural* proof).
- More resilient to small definition changes due to isolation.

Disadvantages

- High-level, compositional, abstract proof: bad for computational behaviour!
- Lack of strictness for functors with identities and composition, results in gratuitous indirections.
- Requires more formalization effort.
- Not so straightforward to translate into a combinatorial presentation for formalization.

Solution: Mild Hell

1. Formalize the high-level categorical proof (extensional properties favourable).
2. Construct the low-level solution by hand (intensional computational behaviour favourable).
3. Prove that the two implementations are equivalent.
   - For computation: dispatch to the low-level construction.
   - For properties: dispatch to the high-level proof.