Model categories in Lean

Reid Barton

Lean Together, January 2019

Source code

- `https://github.com/rwbarton/lean-model-categories`
  Branch `top-dev` (under construction!)
- `https://github.com/rwbarton/lean-homotopy-theory`

What is a model category?

- (Quillen, *Homotopical Algebra*, 1967)
  A *model category* is a category M equipped with three classes of morphisms
  - weak equivalences (W)
  - cofibrations (C)
  - fibrations (F)

  satisfying axioms **M1**, **M2**, **M3**, **M4**, **M5**.
- A model category is one kind of presentation of an underlying "homotopy theory" (or $(\infty, 1)$-category) $L_W M$, which depends only on the weak equivalences W.
- The extra structure on M allows us to do certain calculations in $L_W M$ within the framework of ordinary categories.

Examples

- Classical homotopy theory of spaces
  - Topological spaces ($\mathrm{Top}_{\mathrm{Quillen}}$)

Examples

- Classical homotopy theory of spaces
  - Topological spaces ($\mathsf{Top}_{\mathsf{Quillen}}$)
  - Simplicial sets ($\mathsf{sSet}_{\mathsf{Quillen}}$)

Examples

- Classical homotopy theory of spaces
  - Topological spaces ($\mathsf{Top}_{\mathsf{Quillen}}$)
  - Simplicial sets ($\mathsf{sSet}_{\mathsf{Quillen}}$)
  - ...

Examples

- Classical homotopy theory of spaces
  - Topological spaces ($\mathrm{Top}_{\mathrm{Quillen}}$)
  - Simplicial sets ($\mathrm{sSet}_{\mathrm{Quillen}}$)
  - ...
- Chain complexes (homological algebra)
  - $\mathrm{Ch}(R\mathrm{Mod})_{\mathrm{proj}}$, $\mathrm{Ch}(R\mathrm{Mod})_{\mathrm{inj}}$

Examples

- Classical homotopy theory of spaces
  - Topological spaces ($\mathrm{Top}_{\mathrm{Quillen}}$)
  - Simplicial sets ($\mathrm{sSet}_{\mathrm{Quillen}}$)
  - ...
- Chain complexes (homological algebra)
  - $\mathrm{Ch}(R\mathrm{Mod})_{\mathrm{proj}}$, $\mathrm{Ch}(R\mathrm{Mod})_{\mathrm{inj}}$
- Stable homotopy theory
- Equivariant homotopy theory
- Higher category theory, e.g., $(\infty, 1)$-categories
- $\mathbb{A}^1$-homotopy theory
- ...

Examples

- Classical homotopy theory of spaces
  - Topological spaces ($\mathrm{Top}_{\mathrm{Quillen}}$)
  - Simplicial sets ($\mathrm{sSet}_{\mathrm{Quillen}}$)
  - ...
- Chain complexes (homological algebra)
  - $\mathrm{Ch}(R\mathrm{Mod})_{\mathrm{proj}}$, $\mathrm{Ch}(R\mathrm{Mod})_{\mathrm{inj}}$
- Stable homotopy theory
- Equivariant homotopy theory
- Higher category theory, e.g., $(\infty, 1)$-categories
- $\mathbb{A}^1$-homotopy theory
- ...

468 questions on MathOverflow with tag [model-categories]

Examples

- Classical homotopy theory of spaces
  - Topological spaces ($\mathrm{Top}_{\mathrm{Quillen}}$)
  - Simplicial sets ($\mathrm{sSet}_{\mathrm{Quillen}}$)
  - ...
- Chain complexes (homological algebra)
  - $\mathrm{Ch}(R\mathrm{Mod})_{\mathrm{proj}}$, $\mathrm{Ch}(R\mathrm{Mod})_{\mathrm{inj}}$
- Stable homotopy theory
- Equivariant homotopy theory
- Higher category theory, e.g., $(\infty, 1)$-categories
- $\mathbb{A}^1$-homotopy theory
- ...

468 questions on MathOverflow with tag [model-categories]
(23 questions containing the word "perfectoid")

Project goals

- Formalize the definition of a model category (easy)
- Construct some of the central examples (hard)
  - Current focus: topological spaces ($\mathsf{Top}_{\mathsf{Quillen}}$)
  - Future aims: combinatorial model categories, simplicial sets

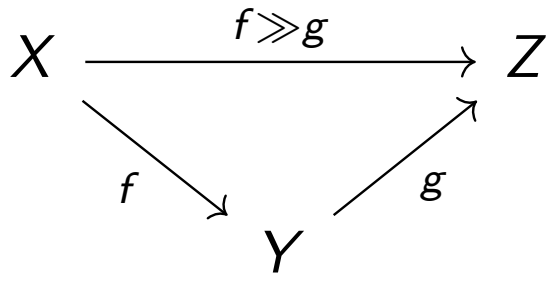
Definition: preliminaries

```
universe u
variables (M : Type (u+1)) [large_category M]

def morphism_class : Type* :=
Π {{a b : M}}, (a → b) -> Prop

instance : has_subset (morphism_class M) :=
{ subset := λ I J,
    ∀ {{a b : M}} {{f : a → b}}, I f -> J f }

instance : has_inter (morphism_class M) :=
{ inter := λ I J, λ a b f, I f ∧ J f }
```

Definition: weak equivalences

```
structure is_weq (W : morphism_class M) : Prop :=
(weq_of_iso : ∀ {x y} (i : iso x y), W i.hom)
(weq_comp : ∀ {x y z} {f : x → y} {g : y → z},
  W f -> W g -> W (f ≫ g))
(weq_cancel_left : ∀ {x y z} {f : x → y} {g : y → z},
  W f -> W (f ≫ g) -> W g)
(weq_cancel_right : ∀ {x y z} {f : x → y} {g : y → z},
  W g -> W (f ≫ g) -> W f)
```

The last three conditions together are the "two-out-of-three property".

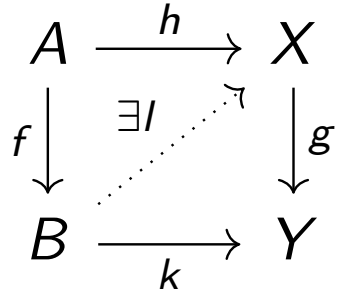


In $\mathrm{Top}_{\mathrm{Quillen}}$, W = maps inducing isomorphisms on π_*.

Definition: lifting property

```
def lp {a b x y : M} (f : a → b) (g : x → y) : Prop :=
∀ (h : a → x) (k : b → y), h ≫ g = f ≫ k ->
∃ (l : b → x), f ≫ l = h ∧ l ≫ g = k

def llp (R : morphism_class M) : morphism_class M :=
λ a b f, ∀ {{x y}} {{g : x → y}}, R g -> lp f g

def rlp (L : morphism_class M) : morphism_class M :=
λ x y g, ∀ {{a b}} {{f : a → b}}, L f -> lp f g
```

$$\mathsf{lp}(f,g) \iff$$



Definition: weak factorization system

```
structure is_wfs (L R : morphism_class M) : Prop :=
(llp : L = llp R)
(rlp : R = rlp L)
(fact : ∀ {x y} (g : x → y),
  ∃ z (l : x → z) (r : z → y), L l ∧ R r ∧ l ≫ r = g)
```

Factorization axiom: Any $g : X \to Y$ can be written as a composition $X \xrightarrow{l} Z \xrightarrow{r} Y$ with $l \in L$ and $r \in R$.

Definition: model category

```
class model_category (M : Type (u+1)) [large_category M] :=
(complete : has_limits M)
(cocomplete : has_colimits M)
(W C F : morphism_class M)
(weq : is_weq W)
(caf : is_wfs C (F ∩ W))
(acf : is_wfs (C ∩ W) F)
```

Constructing model categories

Construction of a model category structure on a category M usually proceeds along the following lines.

- Write down a class of weak equivalences W on M and check that W satisfies the two-out-of-three property.
- Construct two weak factorization systems (C, AF) and (AC, F) on M. There is a standard procedure for this called the small object argument.
- Verify that $AC = C \cap W$ and $AF = F \cap W$. This step generally requires some specialized argument.

Here, we'll focus exclusively on the second step.

Constructing weak factorization systems

- Recall that a weak factorization system (L, R) is required to satisfy $L = \mathrm{llp}(R)$ and $R = \mathrm{rlp}(L)$.
- Easy lemma: $\mathrm{rlp}(\mathrm{llp}(\mathrm{rlp}(I))) = \mathrm{rlp}(I)$.
  (Same as constructive proof of $\neg\neg\neg P \iff \neg P$.)
- Thus, we can choose any collection of morphisms I and set $L = \mathrm{llp}(\mathrm{rlp}(I))$ and $R = \mathrm{rlp}(I)$, and then the conditions $L = \mathrm{llp}(R)$ and $R = \mathrm{rlp}(L)$ will be satisfied. We say that (L, R) is "cofibrantly generated" by I.

  For $\mathrm{Top}_{\mathrm{Quillen}}$, we will take
  - (C, AF) to be generated by $I = \{\, \partial D^n \to D^n \mid n \geq 0 \,\}$
  - (AC, F) to be generated by $J = \{\, D^n \times \{0\} \to D^n \times [0,1] \mid n \geq 0 \,\}$

Constructing weak factorization systems

However, we still need to verify the last condition for a weak factorization system, namely that every morphism g can be factored as a morphism of L followed by a morphism of R.

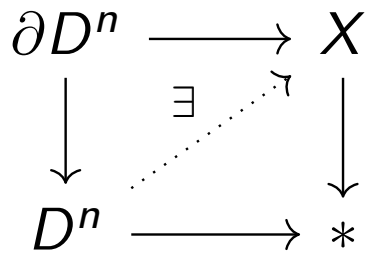
We'll specialize to this situation:

- $M = \mathrm{Top}$
- $I = \{ \partial D^n \to D^n \mid n \geq 0 \}$
- $Y = *$ (so $g : X \to *$)

Need to express g as a composition of $j : X \to X'$ and $g' : X' \to *$ with $j \in \mathrm{llp}(\mathrm{rlp}(I))$ and $g' \in \mathrm{rlp}(I)$.

The small object argument

- Suppose g already has the property that every map $\partial D^n \to X$ admits an extension to D^n.



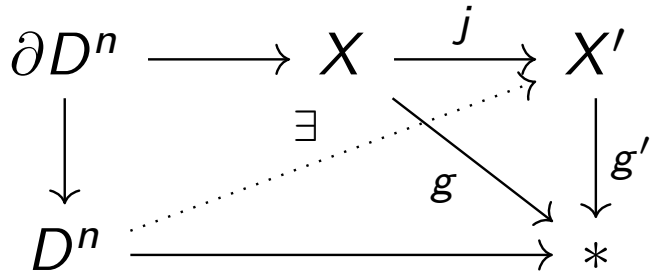
  Then $g \in \mathrm{rlp}(I)$ and we can factor g as the identity followed by g.
- In general, this will not be the case. Idea: replace $g : X \to *$ with a new map $g' : X' \to *$ which is "closer" to having this property.

The small object argument

- Form X' by attaching a cell D^n along every possible attaching map $\partial D^n \to X$. This yields a factorization of $g : X \to *$ as

$$X \xrightarrow{j} X' \xrightarrow{g'} *$$

  By construction, any map $\partial D^n \to X$ admits an extension landing in X':



- General fact: $j \in \mathrm{llp}(\mathrm{rlp}(I)) = L$.
- Done?

The small object argument

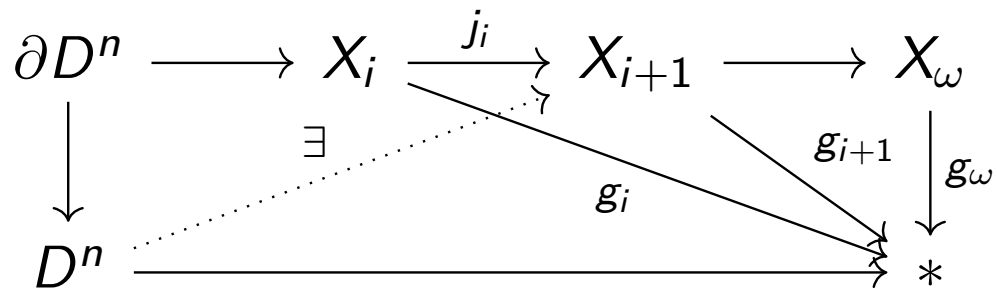
- There will be new maps $\partial D^n \to X'$, so that $g' \notin \mathrm{rlp}(I)$.
- Repeat! Set $X_{i+1} = X_i'$, obtaining a sequence

$$X = X_0 \xrightarrow{j_0} X_1 \xrightarrow{j_1} X_2 \xrightarrow{j_2} \cdots$$

- Define $X_\omega = \mathrm{colim}_i X_i$.
- General fact: the composition $j : X \to X_\omega$ still belongs to $\mathrm{llp}(\mathrm{rlp}(I)) = L$.

The small object argument

- Fact (not general!): Any map $\partial D^n \to X_\omega$ factors through X_i for some $i < \omega$.
- That means any lifting problem for $g_\omega : X_\omega \to *$ factors through $g_i : X_i \to *$ for some i, and therefore we attached a solution to this lifting problem at stage $i + 1$.



- Therefore, $g_\omega \in \mathrm{rlp}(I) = R$ and we're done.

The transfinite case

- In general, countably many steps may not be sufficient.
- We can extend this construction transfinitely:
  - At a successor stage, construct $X_{\alpha+1}$ from X_α as before.
  - At a limit stage, define X_β to be the colimit of the entire sequence $(X_\alpha)_{\alpha<\beta}$ defined so far.
- In broad generality, one can show that there exists an ordinal γ such that every map $A_i \to X_\gamma$ factors through X_α for some $\alpha < \gamma$. Then we can end the construction at stage γ.

Formalizing the small object argument

- We need to construct the transfinite sequence $X_0 \to X_1 \to \cdots \to X_\omega \to X_{\omega+1} \cdots \to X_\gamma$.
- Obvious idea: some kind of recursive construction.
- Choose a well-ordered type α with order type $\gamma + 1$.
- `def well_founded.fix {α : Sort u} {C : α -> Sort v}`
  `{r : α -> α -> Prop} (hwf : well_founded r)`
  `(F : Π x, (Π y, r y x -> C y) -> C x)`
  `(x : α) : C x`
- How to choose `C`? Maybe take `C i` to be the type of sequences $X_0 \to X_1 \to \cdots \to X_i$?

Formalizing the small object argument

- At a limit stage β, we need to form the "union" $(X_\alpha)_{\alpha<\beta}$ of all the sequences defined earlier, so that we can define X_β to be its colimit. In order to make sense of this union, we must know that each sequence extends the earlier ones.
- However, the induction hypothesis `Π y, y < x -> C y` of `well_founded.fix` does not give us any information about how the previous values were defined.
- *If* we could perform this recursive construction, then we could use the defining equations for `well_founded.fix` to prove that each sequence extends the earlier ones...

Recursive constructions with `roption`

- Idea: Recursively construct a sequence of values, each of which is either the desired thing, or an "undefined" value.

  ```
  structure roption (α : Type u) : Type u :=
  (dom : Prop)
  (get : dom -> α)
  ```

  Classically, `dom` is either `true`, in which case `get` provides a value of type α, or `false`, in which case `get` provides nothing. So `roption` α is isomorphic to `option` α.
- After constructing these `roption` values recursively, prove by induction that all the `dom` fields are actually `true`.

General setup

```
parameters {α : Type u} {r : α -> α -> Prop}
  (hwf : well_founded r)
local infix `<` := r

parameters {C : α -> Type u}
parameters (q : Π {{i j}}, C i -> C j -> Prop)
local infix `∼` := q

parameters
  (F : Π (a : α) (f : Π (i : α), i < a -> C i)
         (hf : ∀ i j ria rja, i < j -> f i ria ∼ f j rja),
       {x : C a // ∀ i (ria : i < a), f i ria ∼ x})
```

- `C` is the major premise, i.e., what we want to construct at each step.
- `q` is a compatibility condition on the values of `C` at each pair of steps `i`, `j` with `i < j`.
- `F` describes how to perform each inductive step.

The recursive definition

```
def crec_core : Π (a : α), roption (C a) :=
hwf.fix $ λ a I,
{ dom := ∃ (t : ∀ i (ria : i < a), (I i ria).dom),
    ∀ i j ria rja, i < j ->
    (I i ria).get (t i ria) ∼ (I j rja).get (t j rja),
  get := λ h,
    F a (λ i ria, (I i ria).get (h.fst i ria)) h.snd }
```

- The value at step `a` is defined if the values at all steps `i < a` are defined and consistent.
- In that case, we can use the inductive step `F` to produce the new value.

The definition is actually total

```
lemma crec_lemma : ∀ (a : α),
  ∃ (t : (crec_core a).dom)
    (t' : ∀ i (ria : i < a), (crec_core i).dom)
    (hq : ∀ {{i j ria rja}} {{rij : i < j}},
      (crec_core i).get (t' i ria) ∼
      (crec_core j).get (t' j rja)),
  (crec_core a).get t =
    (F a (λ m rma, (crec_core m).get (t' m rma)) hq).val ∧
  ∀ i (ria : i < a),
    (crec_core i).get (t' i ria) ∼ (crec_core a).get t :=
...
```

- Straightforward inductive argument.
- Key point: `(t : (crec_core a).dom)`, so `crec_core a` has a value.

Extracting the value

```
def crec (a : α) : C a :=
(crec_core a).get (crec_lemma a).fst

lemma crec_consistent {i j} (h : i < j) :
  crec i ∼ crec j :=
by apply (crec_lemma j).snd.snd.snd.right; exact h
```

Generic way to carry out an inductive construction while maintaining a consistency invariant.

Current status of $\mathrm{Top}_{\mathrm{Quillen}}$

```
axiom A : rlp serre_I = rlp serre_J ∩ @is_weak_equivalence

def quillen_serre : model_category.{1 0} Top.{0} :=
model_category.mk' W_is_weq serre_caf serre_acf A AC_sub_W
```

Thank you!