Thinking Recursively, Rethinking Corecursively

David Jaz Myers

June 19, 2017

Mathematical Metaphors

This talk will be about two specific mathematical metaphors, but

- what are mathematical metaphors,
- why make them,
- and how can they be misused?

Mathematical Metaphors

In this talk, we will look closely at the mathematical metaphor between

Complex Systems and **Recursive Functions**

We will see how this metaphor a lot of standard theories in science and philosophy, usually those that fall under the rubrik of “realism”. We will also find that this metaphor can lead us to some shaky philosophical positions.

What is a function

A function is a process that turns an **input** into an **output**.

$$\mathbf{F}(\mathbf{input}) = \mathbf{output}$$

If a function takes inputs of a type **Inputs** and gives outputs of a type **Outputs**, we write

$$\mathbf{F} : \mathbf{Inputs} \to \mathbf{Outputs}$$

For example,

$$\mathbf{F} : \mathbf{Numbers} \to \mathbf{Numbers}$$
$$\mathbf{F}(n) = 2n + 1$$

What is Recursion?

A function is **recursive** when its output on a complicated input is determined by its output on simpler inputs.

Ultimately, the output of a recursive function is determined by its *simplest* inputs.

We call these simplest inputs **atoms**, or base cases, and the rules for building them up **constructors**.

What is Recursion?

So to define a recursive function we need

- to know how to break apart complicated inputs into simpler ones,
- *simplest* inputs (so we eventually stop breaking things apart),
- to know how to put outputs together in a way that relates to taking inputs apart!

Or, more pithily, we need:

- to know how to **analyze inputs**,
- into their **atomic components**,
- so that we can **construct** outputs.

A Lengthy Example

Let's calculate the length of a list! This is a function which takes a list as input and gives a number as output.

$$\textbf{Length} : \textbf{Lists} \to \textbf{Numbers}$$

A list is something like:

[first item, second item, third item . . . last item]

We can break down a list like this:

$$\textbf{A List} = [\text{first item}, \textbf{Rest of the List}]$$

or the list is **Empty**.

A Lengthy Example

Let's calculate the length of a list! This is a function which takes a list as input and gives a number as output.

$$\textbf{Length} : \textbf{Lists} \to \textbf{Numbers}$$

Numbers can be built up by counting:

$$\begin{aligned} &0 \text{ is a number, and} \\ &(1 + \text{a number}) \text{ is a number.} \end{aligned}$$

This is related to taking lists apart because, secretly, numbers are like lists of tally marks:

$$4 = \Big|, \Big|, \Big|, \Big|$$

A Lengthy Example

Definition (Length of a List)

The length of a list is given by the function defined by:

$$\begin{aligned}\textbf{Length}(\textbf{Empty}) &\equiv 0\\ \textbf{Length}([\text{first item}, \textbf{Rest of List}]) &\equiv 1 + \textbf{Length}(\textbf{Rest of List})\end{aligned}$$

This works because

- **Empty** is an atom. There are no simpler lists, so we can stop breaking things apart.
- The **Rest of the List** is simpler (i.e. smaller) than the list we started with. This means we eventually get to the **Empty** list.

Running a Recursive Program

We can run a recursive program **greedily**:

Every time we see something we don't understand, we compute it.

$$\begin{aligned}
\textbf{Length}([1,2,3]) &= 1 + \textbf{Length}([2,3]) \\
&= 1 + (1 + \textbf{Length}([3])) \\
&= 1 + (1 + (1 + \textbf{Length}(\textbf{Empty}))) \\
&= 1 + (1 + (1 + 0)) \\
&= 1 + (1 + 1) \\
&= 1 + 2 \\
&= 3
\end{aligned}$$

Thinking Recursively About Everything

This way of thinking should be familiar to you from popular ways of thinking about physics.

Claim

Physics is like a recursive function

$$\textbf{\textit{Physics}} : \textbf{\textit{Systems}} \to \textbf{\textit{Systems}}$$

which recurses all the way to the ***fundamental particles****, and then builds more complicated phenomena out of the way they behave.*

Thinking Recursively About Everything

Or from philosophy of language:

Claim

Meaning is like a recursive function

$$\textbf{\textit{Meaning}} : \textbf{\textit{Utterances}} \to \textbf{\textit{Meanings}}$$

which builds the meaning of, say, sentences out of the meaning of words.

Thinking Recursively About Everything

Or from sociology

Claim

A society is like a recursive function

$$\textit{\textbf{Society}} : \textit{\textbf{Societies}} \to \textit{\textbf{Societies}}$$

which is determined by the behavior of individuals which are, of course, indivisible.

Thinking Recursively About Everything

Or from economics

Claim

The economy is like a recursive function

$$\textbf{\textit{Economy}} : \textbf{\textit{Markets}} \to \textbf{\textit{Markets}}$$

which is determined by the behavior of agents who act rationally.

Analysis is Recursive

Definition

*[**Analysis**] might be defined as a process of isolating or working back to what is more fundamental by means of which something, initially taken as given, can be explained or reconstructed. – Stanford Encyclopedia*

A Philosophical Problem

In his book *The Case for Idealism*, John Foster argues that some things must have inscrutable, intrinsic properties.

Foster's argument for inscrutable intrinsic properties

Suppose that all properties of all things were *extrinsic*, that is, defined in relation to other things.

$$A)\big)\bigg)\Bigg) \qquad \Bigg(\bigg(\big((B$$

Now, consider a world containing two things, *A* and *B*, each defined only by their disposition to repel the other.

- Foster claims this leads to an infinite regress, and therefore a contradiction.

A Philosophical Problem

Foster's argument for inscrutable intrinsic properties (cont'd)

The back and forth must stop somewhere:

"A is the thing which ... X"

X is the end of the line, it is not defined in relation to anything else. Therefore, it is both

- inscrutable, and
- intrinsic.

This argument rests on two (recursive) assumptions:

1. We must 'evaluate' greedily.
2. There must be a base case.

Do We Have to Make Those Assumptions?

. . . is there another way?

Corecursion

A function is **corecursive** when its output is determined by simpler *outputs*.

We call the rules for breaking apart the output **observers**.

What is Corecursion

So, to define a corecursive function we need

- to know how to observe the output of our function in simpler ways,
- that relate to how we observe our inputs!

We can think of the observers as being experimental setups with which we will test the output of our function.

What is Corecursion

The main idea behind corecursion is:

If we know how our function behaves in all experimental setups, we know what it does.

This is essentially the same as one of the fundamental principles of science:

If we can predict how something behaves in all experimental setups, then we know what it is.

So long as we believe that **a function is what it does.**

Stream and Chill

Let’s have some fun with streams to get our heads around corecursion.

A stream is an infinite list, so we can’t keep the whole thing in memory, but we can observe it piece by piece.

So, let’s set up two experiments:

1. **Head**, where we test what the first thing in the stream is.
2. **Tail**, where we see what’s left.

Now we can define functions corecursively, since we know how to observe their behavior.

Stream and Chill

Let’s define a function

$$\textbf{Every Other} : \textbf{Streams} \to \textbf{Streams}$$

that will take a stream and return the stream of only every other value. For example:

$$\textbf{Every Other}(0, 1, 2, 3, 4, \ldots) = (0, 2, 4, \ldots)$$

To define this, we just need to define how it looks in all the experiments.

Stream and Chill

Definition (The Every Other Function)

Define the **Every Other** function by

$$\begin{aligned} \mathbf{EO}(\text{stream}).\mathbf{Head} &= \text{stream}.\mathbf{Head} \\ \mathbf{EO}(\text{stream}).\mathbf{Tail} &= \mathbf{EO}(\text{stream}.\mathbf{Tail}.\mathbf{Tail}) \end{aligned}$$

This works because

- **EO**(stream) is **covered** by the observers **Head** and **Tail**, they tell us all we need to know about it.

Running a Corecursive Program

We can't evaluate a corecursive program greedily, because the calculation would never end! We have to be **lazy**:

Only compute things when we absolutely need to.

So if you wrote down

$$\mathbf{EO}((0, 1, 2, 3, ...))$$

That would be totally chill.

Running a Corecursive Program

But, if we want to know a specific value of $\mathbf{EO}((0,1,2,3,\ldots))$, then we can calculate

$$\begin{aligned}
&\mathbf{EO}((0,1,2,3,\ldots)).\mathbf{Tail}.\mathbf{Tail}.\mathbf{Head}\\
&= \mathbf{EO}((0,1,2,3,\ldots).\mathbf{Tail}.\mathbf{Tail}).\mathbf{Tail}.\mathbf{Head}\\
&= \mathbf{EO}((0,1,2,3,\ldots).\mathbf{Tail}.\mathbf{Tail}.\mathbf{Tail}.\mathbf{Tail}).\mathbf{Head}\\
&= (0,1,2,3,\ldots).\mathbf{Tail}.\mathbf{Tail}.\mathbf{Tail}.\mathbf{Tail}.\mathbf{Head}\\
&= (1,2,3,4,\ldots).\mathbf{Tail}.\mathbf{Tail}.\mathbf{Tail}.\mathbf{Head}\\
&= (2,3,4,5,\ldots).\mathbf{Tail}.\mathbf{Tail}.\mathbf{Head}\\
&= (3,4,5,6\ldots).\mathbf{Tail}.\mathbf{Head}\\
&= (4,5,6,7\ldots).\mathbf{Head}\\
&= 4
\end{aligned}$$

Corecursion and Différance

If someone asks you what “**EO**” *means*, you could tell them that its meaning is **deferred** until we test it with the observers **Head** and **Tail**.

If they ask you what “**Head**” and “**Tail**” mean, you could only tell them the **different** ways you end up using them.

Definition

Différance is Derrida’s pun on the words *defer* and *differ*.

Thinking Corecursively

Who am I?

How can I find out?

Do I have to find my ‘true self’, the core of my being, to know who I am?

Or do I only have to look at the way I affect the people and places around me?

Thinking corecursively, we don’t have to be anxious about finding our true selves.

Revisiting Foster

Let's look back at Foster's argument for inscrutable intrinsic properties. He claims that the world in which

$$A \text{ only repels } B \text{ and } B \text{ only repels } A$$

cannot exist because it leads to an infinite regress.

- Only leads to infinite regress if we are greedy.
- If we are lazy, this is a perfectly fine definition.

There is nothing inscrutable about it.

Revisiting Foster

Foster's argument shows a fundamental confusion that often underlies recursive thinking:

the confusion between **names** and **things**

- **Names** are like atoms, we don't break them apart.
- **Things** (such as functions) can be *named*, even when we define them corecursively.
- But that doesn't mean that they have base cases!

Limits of the Metaphor

To define a function corecursively, we must **cover** it by observers.

- **Head** and **Tail** tell us all there is to know about a stream.

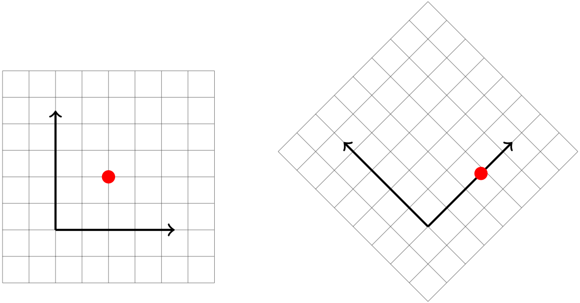
But in the informal world, we never have access to all the contexts in which an object appears,

We can never get all sides of the story.

Going Forward: Physics

Physicists have been thinking corecursively for a long time:

- A physical quantity can only be assigned specific values given a local coordinate system, or **gauge**.



Going Forward: Physics

Principle of Relativity

The physical laws have the same form in all choices of gauge.

A change in gauge is called a **gauge symmetry**.

In other words, if we rotate our experimental setup, we get a rotated result.

$$\Psi.r(X) = r(\Psi.X)$$

The relationship between the observations $\Psi.X$ and $\Psi.r(X)$ depends on *how* X was rotated to $r(X)$.

To fully know an object, we must not only know how it behaves in various contexts,

we must also know how those contexts relate.

In Conclusion

Thinking recursively makes us believe that

- There are basic objects and basic truths about them at the bottom of all phenomena, and
- To know anything at all, we need to know about these basic things.

Thinking corercursively makes us believe that

- Things only make sense in context (in an experiment, relative to an observer, etc.), and
- Knowing how a thing behaves in context is all there is to know about it

There are no basic objects or basic truths

Bridging the Divide

In this talk, I made a stark division between recursive and corecursive thinking.

But in actually programming languages (like Haskell), you can use recursion and corecursion together depending on which is more convenient.

We should use recursive and corecursive thinking together, depending on what needs to be done.

But most importantly, we need to remember that metaphors matter.

Don’t get trapped in a single metaphor

References I

- Andreas Abel, Brigitte Pientka, David Thibodeau, and Anton Setzer, *Copatterns: Programming infinite structures by observations*, SIGPLAN Not. **48** (2013), no. 1, 27–38.

- Michael Beaney, *Analysis*, The Stanford Encyclopedia of Philosophy (Edward N. Zalta, ed.), spring 2015 ed., 2015.

- J. Rutten. C. Kupke, M. Niqui, *Stream differential equations: concrete formats for coinductive definitions.*, Technical Report No. RR-11-10 (2011), 1 – 28.

- Barry Dainton, *Time and space: Second edition*, Mcgill-Queens University Press, 2010.

- Dexter Kozen and Alexandra Silva, *Practical Coinduction*, 2014.

References II

J. Rutten, *An introduction to (co)algebra and (co)induction.*, Advanced topics in bisimulation and coinduction. (D. Sangiorgi and J. Rutten, eds.), Cambridge University Press, Cambridge, 2011.

Towards third generation HOTT

Part 1: Basic syntax

Michael Shulman

University of San Diego

joint work with Thorsten Altenkirch and Ambrus Kaposi

CMU HoTT Seminar
April 28, 2022

Outline

1. Background
2. Identity types
3. Function extensionality
4. Univalence
5. From univalence to homotopy theory

First generation: Book HoTT

Cf. Awodey–Warren, Voevodsky, The HoTT Book

1. Based on Intensional Martin-Löf Type Theory
2. Identity types characterized by path induction
3. Univalence is an axiom

Advantages:

- Easy to do in Coq/Agda: assume univalence and away you go.
- Has models in all higher toposes.

Disadvantages:

- Not computational (UA axiom is stuck)
- Many laws are not definitional:

$$\mathsf{Id}_{A\times B}((a,b),(a',b')) \not\equiv \mathsf{Id}_A(a,a') \times \mathsf{Id}_B(b,b')$$

$$\mathsf{ap}_{f\circ g}(p) \not\equiv \mathsf{ap}_f(\mathsf{ap}_g(p))$$

$$\overrightarrow{\mathsf{tr}}^{x\mapsto A(x)\times B(x)}(p,(a,b)) \not\equiv (\overrightarrow{\mathsf{tr}}^{A}(p,a), \overrightarrow{\mathsf{tr}}^{B}(p,b))$$

Second generation: Cubical type theories

Cf. Bezem-Coquand-Huber, Cohen-Coquand-Huber-Mörtberg, Angiuli-Brunerie-Coquand-Favonia-Harper-Licata

1. Paths defined as maps out of an interval "exo-type" $\mathbb{I}$
2. Cubical Kan operations asserted explicitly in syntax
3. Univalence proved from "glue types"

Advantages:

- Satisfies canonicity and normalization
- Many equalities become definitional
- Implemented in Cubical Agda, cooltt, . . .

Disadvantages:

- Not yet known to have models in higher toposes. . .
  . . . but it probably does (cf. ACCRS, cubical model for spaces).
- . . . ?

What's not to like about cubical type theories?

Martin-Löf J-elim is conceptually fundamental to "equality". In Book HoTT, this simple rule automatically yields higher structure.

Slogan for Book HoTT

Homotopy is implicitly present in the foundations of mathematics.

- A nice story to tell philosophers.
- Accessible to students.

In cubical type theory, identity is defined using a homotopy-theoretic idea (paths), and higher structure is "put in by hand" (Kan ops).

- Fine if you already know you want to do homotopy theory.
- Doesn't have the same philosophical import.
- Not as accessible to students.

That interval

The interval $\mathbb{I}$ is not an ordinary type, but appears in contexts.

- Complicates the meta-theory
- Harder to explain
- Harder to implement
    - Termination-checking of boundaries
    - Display of boundaries to the user
    - Higher-order unification

We're still learning how to implement cubical type theories.
But it's also worth exploring different approaches.

A problem shared by Book HoTT and CTT

Chapter 2 of the Book characterizes lots of identity types:

$$\mathsf{Id}_{A\times B}((a,b),(a',b')) \approx \mathsf{Id}_A(a,a') \times \mathsf{Id}_B(b,b')$$

$$\mathsf{Id}_{A\to B}(f,g) \approx \prod_{(x:A)} \mathsf{Id}_B(f(x),g(x)) \qquad \mathsf{Id}_{\mathsf{U}}(A,B) \approx \mathsf{Equiv}(A,B)$$

- In Book HoTT, these are all only equivalences.
- In cubical type theory, most of them are isomorphisms. . .
  . . . except for Id_{U}, which is still only an equivalence!

This limits the everyday usability of univalence. Given an equivalence $f : A \rightleftarrows B : g$, if we pass it through univalence we can't recover f or g definitionally, only up to homotopy.

Towards H.O.T.T.

I will describe work in progress towards a theory called

Higher Observational Type Theory

with the following properties:

- It admits models in all higher toposes, including spaces.
- Univalence "by definition" (+ other Id characterizations).
- Homotopy theory is emergent rather than explicit;
  all rules have a convincing philosophical justification.
- Computation is a reasonable hope (no obvious stuck terms).

Plan for the three talks:

1. Basic syntax of H.O.T.T.
2. Symmetries and semicartesian cubes
3. Semantics of univalent universes

Outline

1. Background

2. Identity types

3. Function extensionality

4. Univalence

5. From univalence to homotopy theory

Bishop's conception of set

Errett Bishop wrote that

> *A set is defined by describing exactly what must be done in order to construct an element of the set and what must be done in order to show that two elements are equal.*

MLTT follows this principle if "equal" refers to definitional equality, giving introduction and η rules:

$$\frac{a : A \qquad b : B}{(a, b) : A \times B} \qquad\qquad \frac{\pi_1 s \equiv \pi_1 t : A \qquad \pi_2 s \equiv \pi_2 t : B}{s \equiv t : A \times B}$$

The elimination and β rules are determined by "harmony" with the introduction rules.

Lower Observational Type Theory

(Lower) Observational Type Theory (Altenkirch-McBride) applies the same principle to propositional equality types Eq.

$$\frac{a : A \qquad b : B}{(a, b) : A \times B} \qquad\qquad \frac{p : \mathsf{Eq}_A(\pi_1 s, \pi_1 t) \qquad q : \mathsf{Eq}_B(\pi_2 s, \pi_2 t)}{(p \overset{=}{;} q) : \mathsf{Eq}_{A \times B}(s, t)}$$

But this theory is non-univalent by construction, with primitive UIP:

$$\frac{p : \mathsf{Eq}_A(x, y) \qquad q : \mathsf{Eq}_A(x, y)}{\mathsf{irr}(p, q) : \mathsf{Eq}_{\mathsf{Eq}_A(x,y)}(p, q)}.$$

Can we formulate a univalent version?

Notation and terminology

In a univalent context, we refer to identity types, with formation rule

$$\frac{A : \mathsf{U} \qquad a : A \qquad b : A}{\mathsf{Id}_A(a, b) : \mathsf{U}}$$

The elements of an identity type are identifications.

🟩 Green highlights indicate rules of H.O.T.T.

We omit unchanging ambient contexts "$\Gamma \vdash$".

Computing identity types

The OTT rule

$$\frac{p : \mathsf{Eq}_A(\pi_1 s, \pi_1 t) \qquad q : \mathsf{Eq}_B(\pi_2 s, \pi_2 t)}{(p \overset{=}{,} q) : \mathsf{Eq}_{A \times B}(s, t)}$$

says that $\mathsf{Eq}_{A\times B}$ behaves like $\mathsf{Eq}_A \times \mathsf{Eq}_B$. In a higher context, this would require infinitely many such rules for $\mathsf{Id}_{\mathsf{Id}_{A\times B}}$, etc. Instead, we make it a computation rule:

$$\boxed{\mathsf{Id}_{A\times B}(s, t) \equiv \mathsf{Id}_A(\pi_1 s, \pi_1 t) \times \mathsf{Id}_B(\pi_2 s, \pi_2 t).}$$

Then we can just apply the same rule multiple times:

$$\begin{aligned} \mathsf{Id}_{\mathsf{Id}_{A\times B}(s,t)}(p, q) &\equiv \mathsf{Id}_{\mathsf{Id}_A(\pi_1 s, \pi_1 t) \times \mathsf{Id}_B(\pi_2 s, \pi_2 t)}(p, q) \\ &\equiv \mathsf{Id}_{\mathsf{Id}_A(\pi_1 s, \pi_1 t)}(\pi_1 p, \pi_1 q) \times \mathsf{Id}_{\mathsf{Id}_B(\pi_2 s, \pi_2 t)}(\pi_2 p, \pi_2 q) \end{aligned}$$

Respect for equality

Here's Bishop again (paraphrased):

> *An operation f from A into B is called a* function *if we have $f(a) = f(a')$ whenever $a, a' \in A$ and $a = a'$.*

For definitional equality, MLTT has congruence rules for each primitive term former:

$$\frac{s \equiv t : A \times B}{\pi_1 s \equiv \pi_1 t : A \qquad \pi_2 s \equiv \pi_2 t : B}$$

Lower OTT similarly asserts congruence terms:

$$\frac{p : \mathsf{Eq}_{A \times B}(s, t)}{\pi_1^{=} p : \mathsf{Eq}_A(\pi_1 s, \pi_1 t) \qquad \pi_2^{=} p : \mathsf{Eq}_B(\pi_2 s, \pi_2 t)}$$

By structural induction, all terms respect both equalities.

Respect for higher equality

We instead assert a general congruence term:

$$\frac{x : A \vdash f : B \qquad p : \mathsf{Id}_A(a, a')}{\mathsf{ap}_{x.f}(p) : \mathsf{Id}_B(f[a/x], f[a'/x]).}$$

In $\mathsf{ap}_{x.f}(p)$, the variable x is bound in the term f.
(For now, B doesn't depend on A; we'll come back to that later.)

We give it computation rules on standard term-formers:

$$\mathsf{ap}_{x.(a,b)}(p) \equiv (\mathsf{ap}_{x.a}(p), \mathsf{ap}_{x.b}(p))$$

$$\mathsf{ap}_{x.\pi_1 s}(p) \equiv \pi_1\, \mathsf{ap}_{x.s}(p) \qquad \mathsf{ap}_{x.\pi_2 s}(p) \equiv \pi_2\, \mathsf{ap}_{x.s}(p)$$

These are well-typed by the previous rule $\mathsf{Id}_{A \times B} \equiv \mathsf{Id}_A \times \mathsf{Id}_B$.
And for higher equalities, we can apply them multiple times.

Reflexivity

Everything is the same as itself, definitionally and observationally:

$$\frac{a : A}{a \equiv a : A} \qquad\qquad \frac{a : A}{\overline{a} : \mathsf{Eq}_A(a, a)}$$

Similarly, we assert reflexivity terms:

$$\frac{a : A}{\mathsf{refl}_a : \mathsf{Id}_A(a, a)}$$

with computation rules on standard term-formers:

$$\mathsf{refl}_{(a,b)} \equiv (\mathsf{refl}_a, \mathsf{refl}_b) \qquad \mathsf{refl}_{\pi_1 s} \equiv \pi_1 \mathsf{refl}_s \qquad \mathsf{refl}_{\pi_2 s} \equiv \pi_2 \mathsf{refl}_s$$

Again, these rules are well-typed because $\mathsf{Id}_{A \times B} \equiv \mathsf{Id}_A \times \mathsf{Id}_B$.

ap on neutrals and redexes

Note that $\mathsf{ap}_{x.f}(p)$ computes on all terms that f could be, even those not headed by a constructor. The rules are consistent with computations occurring inside f; for instance,

$$\mathsf{ap}_{x.\pi_1(a,b)}(p) \equiv \pi_1\,\mathsf{ap}_{x.(a,b)}(p) \equiv \pi_1\,(\mathsf{ap}_{x.a}(p), \mathsf{ap}_{x.b}(p)) \equiv \mathsf{ap}_{x.a}(p)$$

can also be obtained by reducing $\pi_1(a,b) \equiv a$ in the bound term.

We complete the picture with rules for variables:

$$\boxed{\mathsf{ap}_{x.x}(p) \equiv p} \qquad \boxed{\mathsf{ap}_{x.y}(p) \equiv \mathsf{refl}_y \quad (\text{if } y \text{ is a variable} \neq x)}$$

Then an ap term is never* a normal form: it can always reduce. Think of it as a higher-dimensional explicit substitution "$f[\![p /\!/ x]\!]$".

* modulo some detail we'll come back to later. . .

Functorial laws for ap

Since ap always reduces, we can deduce by induction on terms the following “admissible” equalities:

$$\mathsf{ap}_{x.f}(\mathsf{refl}_a) \equiv \mathsf{refl}_{f[a/x]} \qquad \mathsf{ap}_{y.g}(\mathsf{ap}_{x.f}(p)) \equiv \mathsf{ap}_{x.g[f/y]}(p)$$

$$\mathsf{ap}_{x.t}(p) \equiv \mathsf{refl}_t \quad \text{(if } x \text{ does not appear in } t\text{)}$$

Outline

1. Background
2. Identity types
3. Function extensionality
4. Univalence
5. From univalence to homotopy theory

Towards identity of functions

The obvious rule for equality of functions is function extensionality:

$$\text{¿}\quad \mathsf{Id}_{B\to C}(f,g) \equiv \textstyle\prod_{(y:B)} \mathsf{Id}_C(f(y),g(y)) \quad ?$$

But this is trouble for ap on application. Given $x : A \vdash f : B \to C$ and $x : A \vdash b : B$ while $p : \mathsf{Id}_A(a,a')$, we want to compute

$$\begin{aligned} \mathsf{ap}_{x.f\,b}(p) &: \mathsf{Id}_C((f\,b)[a/x],(f\,b)[a'/x]) \\ &\equiv \mathsf{Id}_C(f[a/x](b[a/x]), f[a'/x](b[a'/x])) \end{aligned}$$

to something involving

$$\begin{array}{lrl} & \mathsf{ap}_{x.f}(p) & : \mathsf{Id}_{B\to C}(f[a/x],g[a'/x]) \\ \text{¿} & & \equiv \textstyle\prod_{(y:B)} \mathsf{Id}_C(f[a/x](y), f[a'/x](y)) \quad ? \\ \text{and} & \mathsf{ap}_{x.b}(p) & : \mathsf{Id}_B(b[a/x],b[a'/x]). \end{array}$$

We need an equality in C between $f[a/x]$ and $f[a'/x]$ applied to different inputs $b[a/x]$ and $b[a'/x]$, but this $\mathsf{ap}_{x.f}(p)$ can't give that.

Identity of functions

A better rule is (still ignoring dependence of B on A)

$$\mathsf{Id}_{B \to C}(f, g) \equiv \prod_{(u:B)} \prod_{(v:B)} \prod_{(q:\mathsf{Id}_B(u,v))} \mathsf{Id}_C(f(u), g(v)).$$

Once we have singleton contractibility, this will be equivalent to the naïve version. But it also gives us

$$\mathsf{ap}_{x.f}(p) : \prod_{(u:B)} \prod_{(v:B)} \prod_{(q:\mathsf{Id}_B(u,v))} \mathsf{Id}_C(f[a/x](u), f[a'/x](v))$$
$$\mathsf{ap}_{x.b}(p) : \mathsf{Id}_B(b[a/x], b[a'/x]).$$

so we can compute

$$\mathsf{ap}_{x.f\,b}(p) : \mathsf{Id}_C(f[a/x](b[a/x]), f[a'/x](b[a'/x]))$$

$$\mathsf{ap}_{x.f\,b}(p) \equiv \mathsf{ap}_{x.f}(p)\big(b[a/x],\ b[a'/x],\ \mathsf{ap}_{x.b}(p)\big).$$

Identity of abstractions

Let $x : A, y : B \vdash t : C$, hence $x : A \vdash \lambda y.t : B \to C$. Given $p : \mathsf{Id}_A(a_0, a_1)$, we can form $\mathsf{ap}_{x.(\lambda y.t)}(p)$, which has type

$$\begin{aligned}&\mathsf{Id}_{B \to C}((\lambda y.t)[a_0/x], (\lambda y.t)[a_1/x]) \\ &\qquad \equiv \prod_{(u:B)} \prod_{(v:B)} \prod_{(q:\mathsf{Id}_B(u,v))} \mathsf{Id}_C(t[a_0/x, u/y], t[a_1/x, v/y])\end{aligned}$$

How do we compute this?

Identity of abstractions

Let $x : A, y : B \vdash t : C$, hence $x : A \vdash \lambda y.t : B \to C$. Given $p : \mathsf{Id}_A(a_0, a_1)$, we can form $\mathsf{ap}_{x.(\lambda y.t)}(p)$, which has type

$$\begin{aligned}&\mathsf{Id}_{B\to C}((\lambda y.t)[a_0/x], (\lambda y.t)[a_1/x])\\ &\qquad \equiv \textstyle\prod_{(u:B)}\prod_{(v:B)}\prod_{(q:\mathsf{Id}_B(u,v))}\mathsf{Id}_C(t[a_0/x, u/y], t[a_1/x, v/y])\end{aligned}$$

How do we compute this? We want to "ap" the term t on both p and q simultaneously. So we introduce a multi-variable ap:

$$\frac{\begin{array}{c}x_1 : A_1, \dots, x_n : A_n \vdash t : C\\ p_1 : \mathsf{Id}_{A_1}(a_1, b_1) \qquad \cdots \qquad p_n : \mathsf{Id}_{A_n}(a_n, b_n)\end{array}}{\mathsf{ap}_{x_1.\cdots x_n.t}(p_1, \dots, p_n) : \mathsf{Id}_C(t[\vec{a}], t[\vec{b}])}$$

$$\mathsf{ap}_{x.(\lambda y.t)}(p) \equiv \lambda u.\lambda v.\lambda q.\mathsf{ap}_{x.y.t}(p, q).$$

(Still ignoring dependence in $A_1, \dots, A_n, C$.)

Computing with multi-variable ap

Multi-variable ap computes with all the same rules we had before. The variable rules are

$$\boxed{\mathsf{ap}_{x_1.\cdots x_n.x_i}(p_1, \ldots, p_n) \equiv p_i}$$

$$\boxed{\mathsf{ap}_{x_1.\cdots x_n.y}(p_1, \ldots, p_n) \equiv \mathsf{refl}_y \;\; (\text{if } y \text{ is a variable} \notin \{x_1, \ldots, x_n\})}$$

In addition, we can identify reflexivity with the 0-ary ap (no bound variables in the subscript):

$$\boxed{\mathsf{refl}_a \equiv \mathsf{ap}_{().a}().}$$

Then all the computation rules for refl become special cases of those for n-ary ap.

Outline

1. Background

2. Identity types

3. Function extensionality

4. Univalence

5. From univalence to homotopy theory

Towards definitional univalence

We want univalence to hold "by definition", meaning $\mathsf{Id}_{\mathsf{U}}(A, B)$ consists of equivalences. But what is an equivalence?

Chapter 4 of the Book discusses several possibilites:

1. Maps with contractible fibers (Voevodsky equivalences)
2. Half-adjoint equivalences
3. Bi-invertible maps

But philosophically, these all have problems:

- None feels "canonical": why choose one over another?
- None is (definitionally) symmetrical in A and B.
- Some are hard to motivate without homotopy theory *a priori*.

What is definitional univalence?

The HoTT Book gave three properties of a type $\mathsf{Equiv}(A, B)$ to be a "good notion of equivalence":

1. There is an embedding $\mathsf{Equiv}(A, B) \hookrightarrow (A \to B)$.
2. $\mathsf{QInv}(A, B) \to \mathsf{Equiv}(A, B)$ over $A \to B$.
3. $\mathsf{Equiv}(A, B) \to \mathsf{QInv}(A, B)$ over $A \to B$.

Here $\mathsf{QInv}(A, B)$ is the naïve type of "quasi-invertible maps":

$$\mathsf{QInv}(A, B) \equiv \sum_{(f:A\to B)}\sum_{(g:B\to A)} \mathsf{Id}(g \circ f, 1_A) \times \mathsf{Id}(f \circ g, 1_B).$$

Univalence ("$\mathsf{idtoeqv} : \mathsf{Id}_{\mathsf{U}}(A, B) \to \mathsf{Equiv}(A, B)$ is an equivalence") can be stated equivalently using any such Equiv (but not QInv).

What is definitional univalence?

The HoTT Book gave three properties of a type $\mathsf{Equiv}(A, B)$ to be a "good notion of equivalence":

1. There is an embedding $\mathsf{Equiv}(A, B) \hookrightarrow (A \to B)$.
2. $\mathsf{QInv}(A, B) \to \mathsf{Equiv}(A, B)$ over $A \to B$.
3. $\mathsf{Equiv}(A, B) \to \mathsf{QInv}(A, B)$ over $A \to B$.

Here $\mathsf{QInv}(A, B)$ is the naïve type of "quasi-invertible maps":

$$\mathsf{QInv}(A, B) \equiv \textstyle\sum_{(f:A\to B)}\sum_{(g:B\to A)} \mathsf{Id}(g \circ f, 1_A) \times \mathsf{Id}(f \circ g, 1_B).$$

Univalence ("$\mathsf{idtoeqv} : \mathsf{Id}_{\mathsf{U}}(A, B) \to \mathsf{Equiv}(A, B)$ is an equivalence") can be stated equivalently using any such Equiv (but not QInv).

But as soon as univalence holds, Id_U also satisfies these properties!

Can univalence ever hold non-definitionally?

What is definitional univalence, really?

Concrete definitions of $\mathrm{Equiv}(A, B)$ include maps $f : A \to B$ and $g : B \to A$ as data. It's useful to remember exactly what these are, definitionally, to compute with them.

Definition

Univalence holds definitionally (at level 1) if, for some definition

$$\mathrm{Equiv}(A, B) :\equiv \textstyle\sum_{(f:A\to B)}\sum_{(g:B\to A)} \text{☁} \qquad \text{we have}$$

$$\mathrm{Equiv}(A, B) \xrightarrow{\uparrow} \mathrm{Id}_{\mathrm{U}}(A, B) \xrightarrow{\downarrow} \mathrm{Equiv}(A, B)$$

such that $(f, g, \text{☁}_1)\uparrow\downarrow \equiv (f, g, \text{☁}_2)$. (Perhaps $\text{☁}_1 \not\equiv \text{☁}_2$.)

Can also consider higher levels, extracting homotopies as well.

Even current cubical type theories (CCHM, ABCFHL, Cubical Agda) do not satisfy this! Can't even extract f definitionally.

One-to-one correspondences

The “best” Equiv is the type of one-to-one correspondences:

$$\text{1-1-Corr}(A,B) :\equiv \sum_{(R:A\to B\to \mathrm{U})} \Big(\prod_{(a:A)} \mathsf{isContr}\big(\sum_{(b:B)} R(a,b)\big)\Big) \times \Big(\prod_{(b:B)} \mathsf{isContr}\big(\sum_{(a:A)} R(a,b)\big)\Big).$$

Remark

An $R : A \to B \to \mathrm{U}$ is a correspondence. It is one-to-one if each element of A or B has exactly one correspondent in the other.

(Prefer not to call it a “relation” unless it's proposition-valued.)

- Definitionally symmetric in A and B.
- A direct propositions-as-types version of classical “bijective relation” (and reduces to it when A, B are sets), so it's easy to motivate without homotopy theory.
- In a larger universe than A, B... but so is $\mathrm{Id}_{\mathrm{U}}(A, B)$.
- Also works really well...

1-1 correspondences vs equivalences

If $R : A \to B \to \mathsf{U}$ is 1-1, the centers of contraction yield $f : A \to B$ and $g : B \to A$, which form an equivalence.

Conversely, if $f : A \to B$ is an equivalence with inverse $g : B \to A$, we make a 1-1 correspondence by $R_f(a, b) :\equiv \mathsf{Id}_B(b, fa)$.

- $\sum_{(b:B)} \mathsf{Id}_B(b, fa)$ is contractible with center $(f\, a, \mathsf{refl}_{fa})$.
- $\sum_{(a:A)} \mathsf{Id}_B(b, fa)$ is contractible with center $(g\, b, \varepsilon_b)$.

If we re-extract an equivalence, we get f and g definitionally.

With a fancier definition of R_f, we can even remember the homotopies $\varepsilon_b : \mathsf{Id}_B(b, fgb)$ and $\eta_a : \mathsf{Id}_A(a, gfa)$.

Computing Id_{U}

For (philosophical, syntactic, and semantic) reasons (later), instead of $\mathsf{Id}_{\mathsf{U}}(A, B) \equiv \text{1-1-Corr}(A, B)$, we make Id_{U} primitive, with intro, elim, and β but no η. (Like a coinductive type with one destructor.)

$$\frac{R : \text{1-1-Corr}(A, B)}{R\uparrow : \mathsf{Id}_{\mathsf{U}}(A, B)} \qquad \frac{A_2 : \mathsf{Id}_{\mathsf{U}}(A_0, A_1)}{A_2\downarrow : \text{1-1-Corr}(A_0, A_1)}$$

$$\frac{p : \text{1-1-Corr}(A, B)}{p\uparrow\downarrow \equiv p}$$

This β rule is sufficient for definitional univalence.

$$\mathsf{Equiv}(A, B) \to \text{1-1-Corr}(A, B) \xrightarrow{\uparrow} \mathsf{Id}_{\mathsf{U}}(A, B) \xrightarrow{\downarrow} \text{1-1-Corr}(A, B) \to \mathsf{Equiv}(A, B)$$

Outline

1. Background

2. Identity types

3. Function extensionality

4. Univalence

5. From univalence to homotopy theory

Towards ∞-groupoid structure

Now every $A : \mathsf{U}$ needs some $\mathsf{refl}_A\!\downarrow : \text{1-1-Corr}(A, A)$. The obvious choice for its underlying correspondence is $\mathsf{Id}_A : A \to A \to \mathsf{U}$:

$$\boxed{\pi_1(\mathsf{refl}_A\!\downarrow) \equiv \mathsf{Id}_A}$$

The other parts of refl_A then give us singleton contractibility!

$$\pi_1\pi_2(\mathsf{refl}_A\!\downarrow) : \textstyle\prod_{(a:A)}\mathsf{isContr}(\sum_{(b:A)}\mathsf{Id}_A(a, b))$$
$$\pi_2\pi_2(\mathsf{refl}_A\!\downarrow) : \textstyle\prod_{(b:A)}\mathsf{isContr}(\sum_{(a:A)}\mathsf{Id}_A(a, b)).$$

In particular, this yields composition operations: given $p : \mathsf{Id}_A(a, x)$ and $q : \mathsf{Id}_A(a, y)$, we have $(x, p), (y, q) : \sum_{(b:A)}\mathsf{Id}_A(a, b)$. But $\sum_{(b:A)}\mathsf{Id}_A(a, b)$ is contractible, so we get (in particular)

$$p^{-1} \centerdot q : \mathsf{Id}_A(x, y).$$

Computing ∞-groupoid structure

Now refl is supposed to compute on all terms. And for U, the “constructors” are type formers. So we must specify how to compute, e.g., $\mathsf{refl}_{A\times B}$ using refl_A and refl_B.

In the first component, this is just the computation of identity types:

$$\begin{aligned}\mathsf{refl}_{A\times B}{\downarrow} &\equiv (\pi_1\mathsf{refl}_{A\times B}{\downarrow}\,,\ \pi_2\mathsf{refl}_{A\times B}{\downarrow})\\ &\equiv (\mathsf{Id}_{A\times B}, \dots)\\ &\equiv (\mathsf{Id}_A \times \mathsf{Id}_B, \dots)\\ &\equiv (\pi_1\mathsf{refl}_A{\downarrow} \times \pi_1\mathsf{refl}_B{\downarrow}, \dots).\end{aligned}$$

We give rules for the other components that compute the ∞-groupoid structure of each type former, e.g. in $A \times B$

$$\boxed{(p,q)^{-1} \cdot (r,s) \equiv (p^{-1} \cdot r\,,\ q^{-1} \cdot s)}$$

Dependent identity types

We can also apply non-nullary ap to terms in U.

If $z : A \vdash B : \mathsf{U}$ and $p : \mathsf{Id}_A(x, y)$, we have

$$\mathsf{ap}_{z.B}(p) : \mathsf{Id}_{\mathsf{U}}(B(x), B(y))$$
$$\pi_1(\mathsf{ap}_{z.B}(p)\!\downarrow) : B(x) \to B(y) \to \mathsf{U}$$

This is how we define the dependent/heterogeneous identity type:

$$\boxed{\mathsf{Id}^{p}_{z.B}(u, v) :\equiv \pi_1(\mathsf{ap}_{z.B}(p)\!\downarrow)(u, v)}$$

Rules for dependent identity types

Since ap also computes on all terms, we give rules like

$$\mathsf{Id}^{p}_{z.B\times C}(u,v) \equiv \mathsf{Id}^{p}_{z.B}(\pi_1 u, \pi_1 v) \times \mathsf{Id}^{p}_{z.C}(\pi_2 u, \pi_2 v).$$

The rule ap(refl) ≡ refl implies that heterogeneous identity types over refl reduce to homogeneous ones:

$$\mathsf{Id}^{\mathsf{refl}_a}_{z.B}(u,v) \equiv \mathsf{Id}_{B[a/z]}(u,v).$$

Similarly, the rule $\mathsf{ap}_{\mathsf{constant}}(p) \equiv \mathsf{refl}$ implies that dependent identity types in constant families also reduce to homogeneous ones:

$$\mathsf{Id}^{p}_{z.B}(u,v) \equiv \mathsf{Id}_{B}(u,v) \qquad \text{(if } z \text{ doesn't appear in } B\text{)}$$

Finally, functoriality of ap gives

$$\mathsf{Id}^{\mathsf{ap}_{x.f}(p)}_{z.C}(u,v) \equiv \mathsf{Id}^{p}_{x.C[f/z]}(u,v)$$

Transport

Still with $z : A \vdash B : \mathsf{U}$ and $p : \mathsf{Id}_A(x, y)$, we also have $\pi_2(\mathsf{ap}_{z.B}(p){\downarrow})$ proving $\mathsf{Id}_B^p : B(x) \to B(y) \to \mathsf{U}$ is one-to-one.

In particular, we have transport: each $u : B(x)$ corresponds to some $\overrightarrow{\mathsf{tr}}^p_{z.B}(u) : B(y)$, with $\overrightarrow{\mathsf{lift}}^p_{z.B}(u) : \mathsf{Id}^p_{z.B}(u, \overrightarrow{\mathsf{tr}}^p_{z.B}(u))$.

We must give computation rules for $\pi_2\,\mathsf{ap}$, specifying how transport computes on type families:

$$\boxed{\overrightarrow{\mathsf{tr}}^p_{z.B\times C}(u) \equiv (\overrightarrow{\mathsf{tr}}^p_{z.B}(\pi_1 u), \overrightarrow{\mathsf{tr}}^p_{z.C}(\pi_2 u))}$$

$$\boxed{\overrightarrow{\mathsf{lift}}^p_{z.B\times C}(u) \equiv (\overrightarrow{\mathsf{lift}}^p_{z.B}(\pi_1 u), \overrightarrow{\mathsf{lift}}^p_{z.C}(\pi_2 u))}$$

Deriving path induction

As in cubical type theory, singleton contractibility and transport together imply Martin-Löf identity elimination J.

1. Given $C : \prod_{(x,y:A)} \mathsf{Id}_A(x,y) \to \mathsf{U}$ with $u : C(a,a,\mathsf{refl}_a)$ and $e : \mathsf{Id}_A(a,b)$, singleton contractibility yields:

$$q : \mathsf{Id}_{\sum_{(y:A)} \mathsf{Id}_A(a,y)}((a,\mathsf{refl}_a),(b,e))$$

2. Currying C to $\widetilde{C}_a : \left(\sum_{(y:A)} \mathsf{Id}_A(a,y)\right) \to \mathsf{U}$, we can transport:

$$\begin{aligned} \overrightarrow{\mathsf{tr}}^{\,q}_{\widetilde{C}_a}(u) &: \widetilde{C}_a((b,e)) \\ &\equiv C(a,b,e). \end{aligned}$$

Again as in cubical type theory, the β rule for J holds only typally.

Σ and Π

With dependent Id, we can define Id_{Σ} and Id_{Π}:

$$\boxed{\mathsf{Id}_{\sum_{(x:A)} B(x)}(s,t) \equiv \sum_{(p:\mathsf{Id}_A(\pi_1 s, \pi_1 t))} \mathsf{Id}_B^p(\pi_2 s, \pi_2 t)}$$

$$\boxed{\mathsf{Id}_{\prod_{(x:A)} B(x)}(f,g) \equiv \prod_{(x:A)} \prod_{(y:A)} \prod_{(p:\mathsf{Id}_A(x,y))} \mathsf{Id}_B^p(f(x), g(y))}$$

with corresponding rules for ap on their term formers,
and generalizations to dependent Id_{Σ} and Id_{Π}.

Dependent multi-variable ap and Id

Dependent Id also makes sense of dependencies in *n*-ary ap, using the evident derived notion of *n*-ary Id.

Each identification is dependent on the preceding ones.

$$\frac{\begin{array}{c} x_1{:}A_1, x_2{:}A_2(x_1), \ldots, x_n{:}A_n(x_1, \ldots, x_{n-1}) \vdash t : C(x_1, \ldots, x_n) \\ p_1 : \mathsf{Id}_{A_1}(a_1, b_1) \qquad p_2 : \mathsf{Id}^{p_1}_{x_1.A_2}(a_2, b_2) \cdots \qquad p_n : \mathsf{Id}^{p_1,\ldots,p_{n-1}}_{x_1.\ldots x_{n-1}.A_n}(a_n, b_n) \end{array}}{\mathsf{ap}_{x_1.\ldots x_n.t}(p_1, \ldots, p_n) : \mathsf{Id}^{p_1,\ldots,p_n}_{x_1.\ldots x_n.C}(t[\vec{a}], t[\vec{b}])}$$

$$\mathsf{Id}^{p_1,\ldots,p_n}_{x_1.\ldots x_n.C}(u, v) :\equiv \pi_1(\mathsf{ap}_{x_1.\ldots x_n.C}(p_1, \ldots, p_n)\!\downarrow)(u, v).$$

When formalizing this, we may use a primitive notion of telescope (a context dependent on the ambient context).

Up next

Plan for the three talks:

1. Basic syntax of H.O.T.T.
2. Symmetries and semicartesian cubes
3. Semantics of univalent universes

Towards third generation HOTT

Part 3: Univalent universes

Michael Shulman

University of San Diego

joint work with Thorsten Altenkirch and Ambrus Kaposi

CMU HoTT Seminar
May 12, 2022

The story so far

Plan for the three talks:

1. Basic syntax of H.O.T.T.
2. Symmetries and semicartesian cubes
3. From semicartesian cubes to univalent universes

Outline

1. The magic of semicartesian cubes
2. Paths in exponentials
3. The parametricity universe
4. The universe of fibrant types
5. Cubical spaces
6. Explaining the universe

The semicartesian cube category

- The semicartesian cube category $\mathbb{\square}$ has, as objects, finite sets.
- A morphism $\phi \in \mathbb{\square}(m, n)$ is a function $\phi : n \to m \sqcup \{-, +\}$ that is injective on the preimage of m.
- The symmetric monoidal structure $m \oplus n$ is disjoint union.
- The automorphisms of n are the symmetric group S_n.
- The presheaf category $\widehat{\mathbb{\square}} = \mathsf{Set}^{\mathbb{\square}^{\mathrm{op}}}$ has a Day convolution monoidal structure. Write $\square^n$ for the representable $\mathbb{\square}(-, n)$.
- A $\widehat{\mathbb{\square}}$-enriched category with $\square^n$-powers has ID-structure:

$$x : A, y : A \vdash \mathsf{Id}_A(x, y) : \mathsf{U} \qquad \longleftrightarrow \qquad \begin{tikzcd} \square^1 \pitchfork A \arrow[d] \\ A \times A \end{tikzcd}$$

Cubical paths and cylinders

Let $\mathcal{E}$ be a presheaf category (such as Set). The category $\mathcal{E}^{\square^{\mathrm{op}}}$ of cubical objects is $\widehat{\square}$-enriched with copowers and powers:

$$\widehat{\square}(K, \mathsf{Map}_{\mathcal{E}^{\square^{\mathrm{op}}}}(X, Y)) \cong \mathcal{E}^{\square^{\mathrm{op}}}(K \odot X, Y) \cong \mathcal{E}^{\square^{\mathrm{op}}}(X, K \pitchfork Y).$$

In particular, it has path spaces $\square^1 \pitchfork X$ and cylinders $\square^1 \odot X$. Path spaces are defined by shifting, while cylinders are magic:

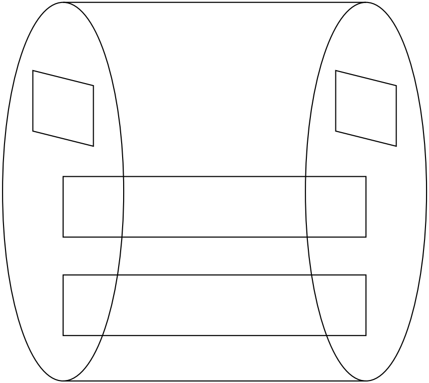
$$\begin{aligned}(\square^1 \pitchfork X)_n &= X_{n \oplus 1} \\ (\square^1 \odot X)_n &= X_n + X_n + \sum_{k \in n} X_{n \smallsetminus \{k\}} \\ &\cong 2 \cdot X_n + n \cdot X_{n-1}.\end{aligned}$$

Almost no other cube category satisfies the magic cylinder formula; we need symmetries but no diagonals or connections.

The magic of semicartesian cylinders

For example:

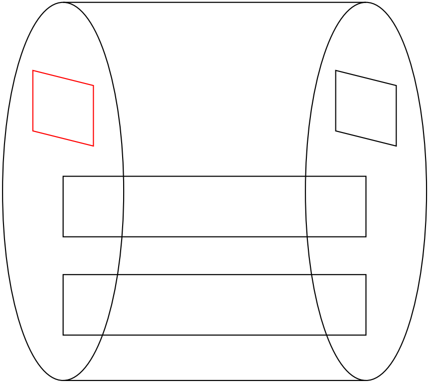
$$(\square^1 \odot X)_2 = X_2 + X_2 + X_1 + X_1$$



The magic of semicartesian cylinders

For example:

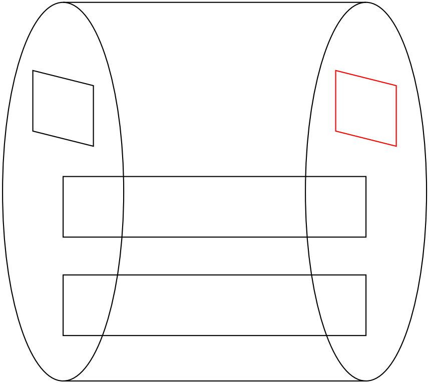
$$(\square^1 \odot X)_2 = {\color{red}X_2} + X_2 + X_1 + X_1$$



The magic of semicartesian cylinders

For example:

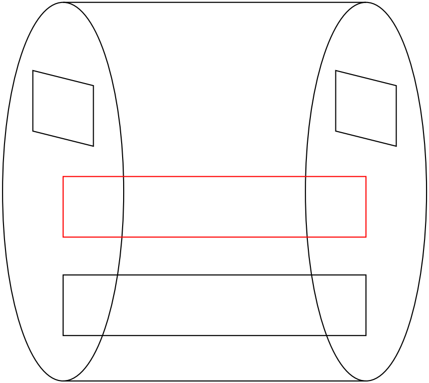
$$(\square^1 \odot X)_2 = X_2 + {\color{red}X_2} + X_1 + X_1$$



The magic of semicartesian cylinders

For example:

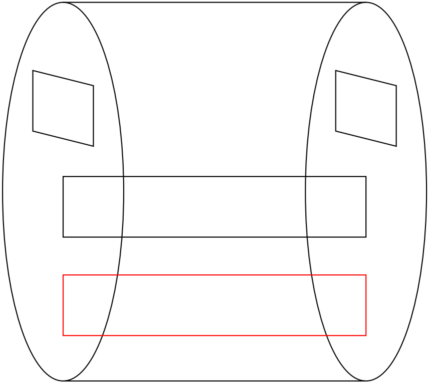
$$(\Box^1 \odot X)_2 = X_2 + X_2 + {\color{red}X_1} + X_1$$



The magic of semicartesian cylinders

For example:

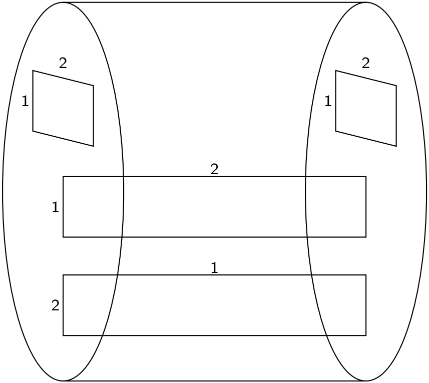
$$(\Box^1 \odot X)_2 = X_2 + X_2 + X_1 + X_1$$



The magic of semicartesian cylinders

For example:

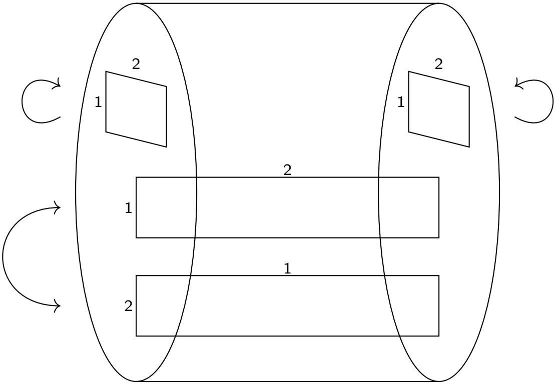
$$(\square^1 \odot X)_2 = X_2 + X_2 + X_1 + X_1$$



The magic of semicartesian cylinders

For example:

$$(\square^1 \odot X)_2 = X_2 + X_2 + X_1 + X_1$$



Amazing right adjoints

Since the path-space is shifting, $(\square^1 \pitchfork X)_n = X_{n \oplus 1}$, it preserves all colimits, hence has an ("amazing") right adjoint

$$\mathcal{E}^{\square^{\mathrm{op}}}(\square^1 \pitchfork X, Y) \cong \mathcal{E}^{\square^{\mathrm{op}}}(X, \sqrt{Y})$$

This also has a fiberwise version:

$$\begin{tikzcd}
Y \arrow[d] & {} \arrow[d, phantom, "\mapsto"] & \sqrt{Y}_W \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \sqrt{Y} \arrow[d] \\
\square^1 \pitchfork W & {} & W \arrow[r] & \sqrt{\square^1 \pitchfork W}
\end{tikzcd}$$

The fiberwise version maps $\mathcal{E}^{\square^{\mathrm{op}}}/(\square^1 \pitchfork W)$ to $\mathcal{E}^{\square^{\mathrm{op}}}/W$.

Outline

1. The magic of semicartesian cubes

2. Paths in exponentials

3. The parametricity universe

4. The universe of fibrant types

5. Cubical spaces

6. Explaining the universe

Identity types of exponentials

For cartesian cubes, powers coincide with cartesian exponentials. So $\square^1 \pitchfork (A \to B) \cong A \to (\square^1 \pitchfork B)$, and $\mathsf{Id}_{A\to B}(f,g) \cong \prod_{(x:A)} \mathsf{Id}_B(fx, gx)$.

In the semicartesian case, we need to relate the cartesian exponential $A \to B$ with the monoidal path-space $(\square^1 \pitchfork -)$. To get our desired rule

$$\mathsf{Id}_{A\to B}(f,g) \cong \prod_{(u:A)}\prod_{(v:A)}\prod_{(q:\mathsf{Id}_A(u,v))} \mathsf{Id}_B(f(u), g(v)).$$

we want a pullback in $\mathcal{E}^{\square^{\mathrm{op}}}$:

$$\begin{tikzcd}
\square^1 \pitchfork (A \to B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & ((\square^1 \pitchfork A) \to (\square^1 \pitchfork B)) \arrow[d] \\
(A \to B) \times (A \to B) \arrow[r] & ((\square^1 \pitchfork A) \to B) \times ((\square^1 \pitchfork A) \to B)
\end{tikzcd}$$

Identity types of exponentials

We want a pullback in $\mathcal{E}^{\square^{\mathrm{op}}}$:

$$\begin{tikzcd}
\square^1 \pitchfork (A \to B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \big((\square^1 \pitchfork A) \to (\square^1 \pitchfork B)\big) \arrow[d] \\
(A \to B)^2 \arrow[r] & ((\square^1 \pitchfork A) \to B)^2
\end{tikzcd}$$

Identity types of exponentials

By Yoneda, we want a pullback in Set for all $X \in \mathcal{E}^{\boxbar^{\mathrm{op}}}$:

$$\begin{tikzcd}
\mathcal{E}^{\boxbar^{\mathrm{op}}}(\textcolor{red}{X}, \Box^1 \pitchfork (A \to B)) \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathcal{E}^{\boxbar^{\mathrm{op}}}\big(\textcolor{red}{X}, (\Box^1 \pitchfork A) \to (\Box^1 \pitchfork B)\big) \arrow[d] \\
\mathcal{E}^{\boxbar^{\mathrm{op}}}(\textcolor{red}{X}, (A \to B)^2) \arrow[r] & \mathcal{E}^{\boxbar^{\mathrm{op}}}(\textcolor{red}{X}, ((\Box^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\boxbar^{\mathrm{op}}}(X, \square^1 \pitchfork (A \to B)) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\boxbar^{\mathrm{op}}}\big(X, (\square^1 \pitchfork A) \to (\square^1 \pitchfork B)\big) \arrow[d] \\
\mathcal{E}^{\boxbar^{\mathrm{op}}}(X, (A \to B)^2) \arrow[r] & \mathcal{E}^{\boxbar^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}(\square^1 \odot X, A \to B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\big(X, (\square^1 \pitchfork A) \to (\square^1 \pitchfork B)\big) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}(X, (A \to B)^2) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}(\square^1 \odot X, A \to B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\big(X, (\square^1 \pitchfork A) \to (\square^1 \pitchfork B)\big) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}(X, (A \to B)^2) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\big(X, (\square^1 \pitchfork A) \to (\square^1 \pitchfork B)\big) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}(X, (A \to B)^2) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\boxbar^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\boxbar^{\mathrm{op}}}\big(X, (\square^1 \pitchfork A) \to (\square^1 \pitchfork B)\big) \arrow[d] \\
\mathcal{E}^{\boxbar^{\mathrm{op}}}(X, (A \to B)^{\color{red}{2}}) \arrow[r] & \mathcal{E}^{\boxbar^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\big(X, (\square^1 \pitchfork A) \to (\square^1 \pitchfork B)\big) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}(2 \cdot X, A \to B) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\boxbar^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\boxbar^{\mathrm{op}}}\big(X, (\square^1 \pitchfork A) \to (\square^1 \pitchfork B)\big) \arrow[d] \\
\mathcal{E}^{\boxbar^{\mathrm{op}}}(2 \cdot X, {\color{red} A \to} B) \arrow[r] & \mathcal{E}^{\boxbar^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\boxbar^{\mathrm{op}}}((\Box^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathcal{E}^{\boxbar^{\mathrm{op}}}\big(X, (\Box^1 \pitchfork A) \to (\Box^1 \pitchfork B)\big) \arrow[d] \\
\mathcal{E}^{\boxbar^{\mathrm{op}}}((2 \cdot X) {\color{red}\times A}, B) \arrow[r] & \mathcal{E}^{\boxbar^{\mathrm{op}}}(X, ((\Box^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\big(X, (\square^1 \pitchfork A) \to (\square^1 \pitchfork B)\big) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}((2 \cdot X) \times A, B) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\big(X \times (\square^1 \pitchfork A), \square^1 \pitchfork B\big) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}((2 \cdot X) \times A, B) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\big(X \times (\square^1 \pitchfork A), {\color{red}\square^1 \pitchfork} B\big) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}((2 \cdot X) \times A, B) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties...

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\big({\color{red}\square^1 \odot} (X \times (\square^1 \pitchfork A)), B\big) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}((2 \cdot X) \times A, B) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}(X, ((\square^1 \pitchfork A) \to B)^2)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties...

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\big(\square^1 \odot (X \times (\square^1 \pitchfork A)), B\big) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}((2 \cdot X) \times A, B) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}\big(X, ((\square^1 \pitchfork A) \to B)^{2}\big)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\left(\square^1 \odot (X \times (\square^1 \pitchfork A)), B\right) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}((2 \cdot X) \times A, B) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}(2 \cdot X, (\square^1 \pitchfork A) \to B)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\big(\square^1 \odot (X \times (\square^1 \pitchfork A)), B\big) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}((2 \cdot X) \times A, B) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}\big(2 \cdot X, {\color{red}(\square^1 \pitchfork A) \to} B\big)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties. . .

$$\begin{tikzcd}
\mathcal{E}^{\square^{\mathrm{op}}}((\square^1 \odot X) \times A, B) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\square^{\mathrm{op}}}\left(\square^1 \odot (X \times (\square^1 \pitchfork A)), B\right) \arrow[d] \\
\mathcal{E}^{\square^{\mathrm{op}}}((2 \cdot X) \times A, B) \arrow[r] & \mathcal{E}^{\square^{\mathrm{op}}}((2 \cdot X) \times (\square^1 \pitchfork A), B)
\end{tikzcd}$$

Identity types of exponentials

Now we apply universal properties...

$$\begin{tikzcd}
\mathcal{E}^{\Box^{\mathrm{op}}}((\Box^1 \odot X) \times A, {\color{red}B}) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E}^{\Box^{\mathrm{op}}}(\Box^1 \odot (X \times (\Box^1 \pitchfork A)), {\color{red}B}) \arrow[d] \\
\mathcal{E}^{\Box^{\mathrm{op}}}((2 \cdot X) \times A, {\color{red}B}) \arrow[r] & \mathcal{E}^{\Box^{\mathrm{op}}}((2 \cdot X) \times (\Box^1 \pitchfork A), {\color{red}B})
\end{tikzcd}$$

And now, by Yoneda again...

Identity types of exponentials

We equivalently want a pushout in $\mathcal{E}^{\square^{\mathrm{op}}}$:

$$\begin{tikzcd}
(2 \cdot X) \times (\square^1 \pitchfork A) \arrow[r] \arrow[d] & \square^1 \odot (X \times (\square^1 \pitchfork A)) \arrow[d] \\
(2 \cdot X) \times A \arrow[r] \arrow[ur, phantom, "\ulcorner" very near end] & (\square^1 \odot X) \times A
\end{tikzcd}$$

Identity types of exponentials

Which means a pushout in $\mathcal{E}$ for all n:

$$\begin{tikzcd}
((2 \cdot X) \times (\square^1 \pitchfork A))_n \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner", very near end] & (\square^1 \odot (X \times (\square^1 \pitchfork A)))_n \arrow[d] \\
((2 \cdot X) \times A)_n \arrow[r] & ((\square^1 \odot X) \times A)_n
\end{tikzcd}$$

Identity types of exponentials

Which means a pushout in $\mathcal{E}$ for all n:

$$\begin{tikzcd}
((2 \cdot X) \times (\square^1 \pitchfork A))_n \arrow[r] \arrow[d] & (\square^1 \odot (X \times (\square^1 \pitchfork A)))_n \arrow[d] \\
(2 \cdot X)_n \times A_n \arrow[r] & ((\square^1 \odot X) \times A)_n \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

Identity types of exponentials

Which means a pushout in $\mathcal{E}$ for all n:

$$\begin{tikzcd}
((2 \cdot X) \times (\square^1 \pitchfork A))_n \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner", very near end] & (\square^1 \odot (X \times (\square^1 \pitchfork A)))_n \arrow[d] \\
2 \cdot (X_n \times A_n) \arrow[r] & ((\square^1 \odot X) \times A)_n
\end{tikzcd}$$

Identity types of exponentials

Which means a pushout in $\mathcal{E}$ for all n:

$$\begin{tikzcd}
(2 \cdot X)_n \times (\Box^1 \pitchfork A)_n \arrow[r] \arrow[d] & (\Box^1 \odot (X \times (\Box^1 \pitchfork A)))_n \arrow[d] \\
2 \cdot (X_n \times A_n) \arrow[r] & ((\Box^1 \odot X) \times A)_n \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

Identity types of exponentials

Which means a pushout in $\mathcal{E}$ for all n:

$$\begin{tikzcd}
2 \cdot (X_n \times A_{n+1}) \arrow[r] \arrow[d] & (\square^1 \odot (X \times (\square^1 \pitchfork A)))_n \arrow[d] \\
2 \cdot (X_n \times A_n) \arrow[r] & ((\square^1 \odot X) \times A)_n \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

Identity types of exponentials

Which means a pushout in $\mathcal{E}$ for all n:

$$\begin{tikzcd}
2 \cdot (X_n \times A_{n+1}) \arrow[r] \arrow[d] & (\square^1 \odot (X \times (\square^1 \pitchfork A)))_n \arrow[d] \\
2 \cdot (X_n \times A_n) \arrow[r] \arrow[ur, phantom, "\ulcorner"] & (\square^1 \odot X)_n \times A_n
\end{tikzcd}$$

Identity types of exponentials

Which means a pushout in $\mathcal{E}$ for all n:

$$\begin{tikzcd}
2 \cdot (X_n \times A_{n+1}) \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner", very near end] & (\square^1 \odot (X \times (\square^1 \pitchfork A)))_n \arrow[d] \\
2 \cdot (X_n \times A_n) \arrow[r] & (2 \cdot X_n + n \cdot X_{n-1}) \times A_n
\end{tikzcd}$$

Identity types of exponentials

Which means a pushout in $\mathcal{E}$ for all n:

$$\begin{tikzcd}
2 \cdot (X_n \times A_{n+1}) \arrow[r] \arrow[d] & (\square^1 \odot (X \times (\square^1 \pitchfork A)))_n \arrow[d] \\
2 \cdot (X_n \times A_n) \arrow[r] & 2 \cdot (X_n \times A_n) + n \cdot (X_{n-1} \times A_n) \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

Identity types of exponentials

Which means a pushout in $\mathcal{E}$ for all n:

$$\begin{tikzcd}
2 \cdot (X_n \times A_{n+1}) \arrow[r] \arrow[d] & (\square^1 \odot (X \times (\square^1 \pitchfork A)))_n \arrow[d] \\
2 \cdot (X_n \times A_n) \arrow[r] & 2 \cdot (X_n \times A_n) + n \cdot (X_{n-1} \times A_n) \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

$$\begin{aligned}
&(\square^1 \odot (X \times (\square^1 \pitchfork A)))_n \\
&\qquad = 2 \cdot (X \times (\square^1 \pitchfork A))_n + n \cdot (X \times (\square^1 \pitchfork A))_{n-1} \\
&\qquad = 2 \cdot (X_n \times (\square^1 \pitchfork A)_n) + n \cdot (X_{n-1} \times (\square^1 \pitchfork A)_{n-1}) \\
&\qquad = 2 \cdot (X_n \times A_{n+1}) + n \cdot (X_{n-1} \times A_n)
\end{aligned}$$

Identity types of exponentials

Which means a pushout in $\mathcal{E}$ for all n:

$$\begin{tikzcd}
2 \cdot (X_n \times A_{n+1}) \arrow[r] \arrow[d] & 2 \cdot (X_n \times A_{n+1}) + n \cdot (X_{n-1} \times A_n) \arrow[d] \\
2 \cdot (X_n \times A_n) \arrow[r] & 2 \cdot (X_n \times A_n) + n \cdot (X_{n-1} \times A_n) \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

$$\begin{aligned}
&(\square^1 \odot (X \times (\square^1 \pitchfork A)))_n \\
&\qquad = 2 \cdot (X \times (\square^1 \pitchfork A))_n + n \cdot (X \times (\square^1 \pitchfork A))_{n-1} \\
&\qquad = 2 \cdot (X_n \times (\square^1 \pitchfork A)_n) + n \cdot (X_{n-1} \times (\square^1 \pitchfork A)_{n-1}) \\
&\qquad = 2 \cdot (X_n \times A_{n+1}) + n \cdot (X_{n-1} \times A_n)
\end{aligned}$$

Identity types of exponentials

$$\begin{tikzcd}
2 \cdot (X_n \times A_{n+1}) \arrow[r] \arrow[d] & 2 \cdot (X_n \times A_{n+1}) + n \cdot (X_{n-1} \times A_n) \arrow[d] \\
2 \cdot (X_n \times A_n) \arrow[r] \arrow[ur, phantom, "\ulcorner"] & 2 \cdot (X_n \times A_n) + n \cdot (X_{n-1} \times A_n)
\end{tikzcd}$$

But this is just a coproduct of two pushout squares:

$$\begin{tikzcd}
2 \cdot (X_n \times A_{n+1}) \arrow[r, equal] \arrow[d] & 2 \cdot (X_n \times A_{n+1}) \arrow[d] & \emptyset \arrow[r] \arrow[d, equal] & n \cdot (X_{n-1} \times A_n) \arrow[d, equal] \\
2 \cdot (X_n \times A_n) \arrow[r, equal] \arrow[ur, phantom, "\ulcorner"] & 2 \cdot (X_n \times A_n) & \emptyset \arrow[r] \arrow[ur, phantom, "\ulcorner"] & n \cdot (X_{n-1} \times A_n)
\end{tikzcd}$$

Thus, it is a pushout, completing the proof of our desired rule

$$\mathsf{Id}_{A \to B}(f, g) \cong \prod_{(u:A)} \prod_{(v:A)} \prod_{(q:\mathsf{Id}_A(u,v))} \mathsf{Id}_B(f(u), g(v)).$$

The same ideas work for dependent types and for Π-types.

Outline

1. The magic of semicartesian cubes

2. Paths in exponentials

3. The parametricity universe

4. The universe of fibrant types

5. Cubical spaces

6. Explaining the universe

Paths in the universe

If U "classifies" small maps, then

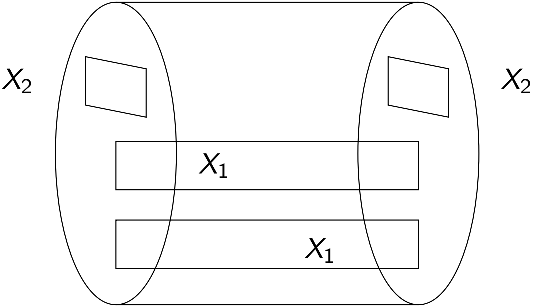
$$\mathcal{E}^{\square^{\mathrm{op}}}(X, \square^1 \pitchfork \mathsf{U}) \cong \mathcal{E}^{\square^{\mathrm{op}}}(\square^1 \odot X, \mathsf{U})$$

so $\square^1 \pitchfork \mathsf{U}$ "classifies" small maps over cylinders.

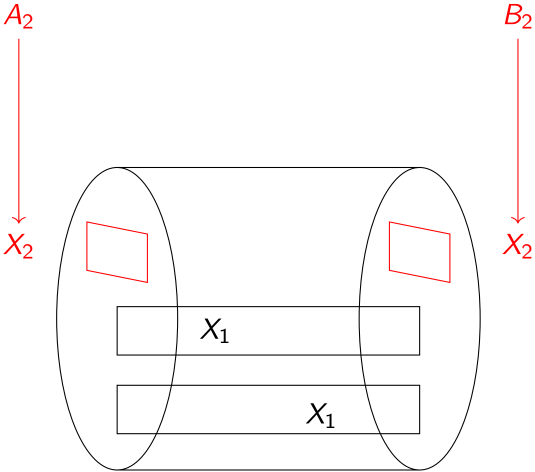
By "extensivity", a map $Y \to (\square^1 \odot X)$ decomposes Y_n as a coproduct too:

$$\begin{tikzcd}
Y_n \arrow[r, "\cong"] \arrow[d] & A_n + B_n + \sum_{k \in n} C_{n,k} \arrow[d] \\
(\square^1 \odot X)_n \arrow[r, "\cong"'] & X_n + X_n + \sum_{k \in n} X_{n \setminus \{k\}}.
\end{tikzcd}$$

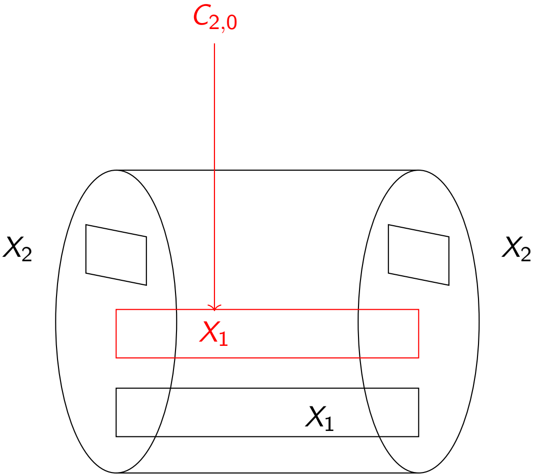
Cubes over cylinders



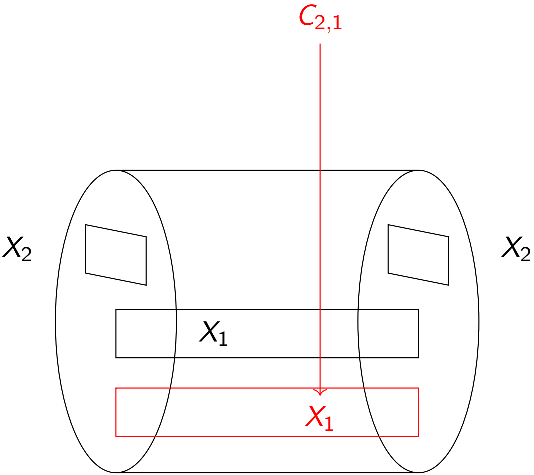
Cubes over cylinders



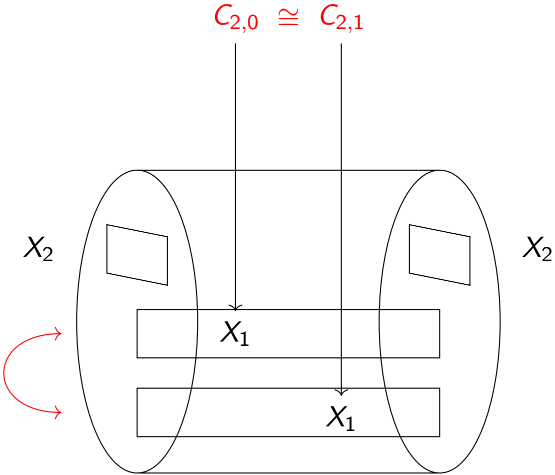
Cubes over cylinders



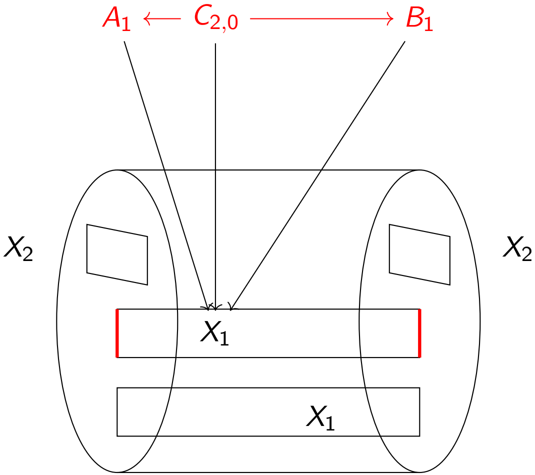
Cubes over cylinders



Cubes over cylinders



Cubes over cylinders



Cubical operators over cylinders

Let $\phi \in \square(m,n)$, so $\phi : n \to m \sqcup \{-,+\}$.

$$\begin{tikzcd}
A_n + B_n + \sum_{k\in n} C_{n,k} \arrow[r, "\phi^*"] \arrow[d] & A_m + B_m + \sum_{\ell\in m} C_{m,\ell} \arrow[d] \\
X_n + X_n + \sum_{k\in n} X_{n\setminus\{k\}} \arrow[r, "\phi^*"] & X_m + X_m + \sum_{\ell\in m} X_{m\setminus\{\ell\}}
\end{tikzcd}$$

- Any ϕ preserves the first two summands, so we have $A, B \in \mathcal{E}^{\square^{\mathrm{op}}}$ with maps $A \to X$ and $B \to X$.
- $S_n \subseteq \square(n,n)$ permutes the summands $X_{n\setminus\{k\}}$, and its subgroup $S_{n\setminus\{k\}}$ acts on $X_{n\setminus\{k\}}$. Thus, $C_{n,k} \cong C_{n,k'}$ $\forall k, k'$.
- If $\phi(k) = \ell \in m$, then ϕ maps $X_{n\setminus\{k\}}$ to $X_{m\setminus\{\ell\}}$. These assemble the $C_{n,k}$ into $C \in \mathcal{E}^{\square^{\mathrm{op}}}$ with a map $C \to X$.
- If $\phi(k) \in \{-,+\}$, then ϕ maps $X_{n\setminus\{k\}}$ to one of the first X_m's. These assemble into maps $C \to A$ and $C \to B$ over X.

Cubical operators over cylinders

Let $\phi \in \square(m, n)$, so $\phi : n \to m \sqcup \{-, +\}$.

$$\begin{tikzcd}
A_n + B_n + \sum_{k \in n} C_{n,k} \arrow[r, "\phi^*"] \arrow[d] & A_m + B_m + \sum_{\ell \in m} C_{m,\ell} \arrow[d] \\
X_n + X_n + \sum_{k \in n} X_{n \smallsetminus \{k\}} \arrow[r, "\phi^*"] & X_m + X_m + \sum_{\ell \in m} X_{m \smallsetminus \{\ell\}}
\end{tikzcd}$$

- Any ϕ preserves the first two summands, so we have $A, B \in \mathcal{E}^{\square^{\mathrm{op}}}$ with maps $A \to X$ and $B \to X$.
- $S_n \subseteq \square(n, n)$ permutes the summands $X_{n \smallsetminus \{k\}}$, and its subgroup $S_{n \smallsetminus \{k\}}$ acts on $X_{n \smallsetminus \{k\}}$. Thus, $C_{n,k} \cong C_{n,k'}$ $\forall k, k'$.
- If $\phi(k) = \ell \in m$, then ϕ maps $X_{n \smallsetminus \{k\}}$ to $X_{m \smallsetminus \{\ell\}}$. These assemble the $C_{n,k}$ into $C \in \mathcal{E}^{\square^{\mathrm{op}}}$ with a map $C \to X$.
- If $\phi(k) \in \{-, +\}$, then ϕ maps $X_{n \smallsetminus \{k\}}$ to one of the first X_m's. These assemble into maps $C \to A$ and $C \to B$ over X.

Cubical operators over cylinders

Let $\phi \in \Box(m,n)$, so $\phi : n \to m \sqcup \{-,+\}$.

$$\begin{tikzcd}
A_n + B_n + \sum_{k\in n} C_{n,k} \arrow[r, "\phi^*"] \arrow[d] & A_m + B_m + \sum_{\ell\in m} C_{m,\ell} \arrow[d] \\
X_n + X_n + \sum_{k\in n} X_{n\smallsetminus\{k\}} \arrow[r, "\phi^*"] & X_m + X_m + \sum_{\ell\in m} X_{m\smallsetminus\{\ell\}}
\end{tikzcd}$$

- Any ϕ preserves the first two summands, so we have $A, B \in \mathcal{E}^{\Box^{\mathrm{op}}}$ with maps $A \to X$ and $B \to X$.
- $S_n \subseteq \Box(n,n)$ permutes the summands $X_{n\smallsetminus\{k\}}$, and its subgroup $S_{n\smallsetminus\{k\}}$ acts on $X_{n\smallsetminus\{k\}}$. Thus, $C_{n,k} \cong C_{n,k'}$ $\forall k, k'$.
- If $\phi(k) = \ell \in m$, then ϕ maps $X_{n\smallsetminus\{k\}}$ to $X_{m\smallsetminus\{\ell\}}$. These assemble the $C_{n,k}$ into $C \in \mathcal{E}^{\Box^{\mathrm{op}}}$ with a map $C \to X$.
- If $\phi(k) \in \{-,+\}$, then ϕ maps $X_{n\smallsetminus\{k\}}$ to one of the first X_m's. These assemble into maps $C \to A$ and $C \to B$ over X.

Cubical operators over cylinders

Let $\phi \in \square(m, n)$, so $\phi : n \to m \sqcup \{-, +\}$.

$$\begin{tikzcd}
A_n + B_n + \sum_{k \in n} C_{n,k} \arrow[r, "\phi^*"] \arrow[d] & A_m + B_m + \sum_{\ell \in m} C_{m,\ell} \arrow[d] \\
X_n + X_n + \sum_{k \in n} X_{n \smallsetminus \{k\}} \arrow[r, "\phi^*"] & X_m + X_m + \sum_{\ell \in m} X_{m \smallsetminus \{\ell\}}
\end{tikzcd}$$

- Any ϕ preserves the first two summands, so we have $A, B \in \mathcal{E}^{\square^{\mathrm{op}}}$ with maps $A \to X$ and $B \to X$.
- $S_n \subseteq \square(n, n)$ permutes the summands $X_{n \smallsetminus \{k\}}$, and its subgroup $S_{n \smallsetminus \{k\}}$ acts on $X_{n \smallsetminus \{k\}}$. Thus, $C_{n,k} \cong C_{n,k'}$ $\forall k, k'$.
- If $\phi(k) = \ell \in m$, then ϕ maps $X_{n \smallsetminus \{k\}}$ to $X_{m \smallsetminus \{\ell\}}$. These assemble the $C_{n,k}$ into $C \in \mathcal{E}^{\square^{\mathrm{op}}}$ with a map $C \to X$.
- If $\phi(k) \in \{-, +\}$, then ϕ maps $X_{n \smallsetminus \{k\}}$ to one of the first X_m's. These assemble into maps $C \to A$ and $C \to B$ over X.

Type families over cylinders

Thus, from $Y \to (\Box^1 \odot X)$, we extract a span in $\mathcal{E}^{\Box^{\mathrm{op}}}/X$:

$$\begin{tikzcd}
A \arrow[dr] & C \arrow[l] \arrow[r] \arrow[d] & B \arrow[dl] \\
& X &
\end{tikzcd}$$

Theorem

For $X \in \mathcal{E}^{\Box^{\mathrm{op}}}$, we have an equivalence of categories

$$\mathcal{E}^{\Box^{\mathrm{op}}}/(\Box^1 \odot X) \;\simeq\; (\mathcal{E}^{\Box^{\mathrm{op}}}/X)^{(\cdot \leftarrow \cdot \rightarrow \cdot)}$$

(Also generalizes to many other $K \in \widehat{\Box}$ replacing $\Box^1$.)

Identity types of the parametricity universe

Corollary

There is a $\mathsf{U}^0 \in \mathcal{E}^{\square^{\mathrm{op}}}$ *that classifies small maps, and such that there is a trivial fibration (i.e. a map with RLP against monos)*

$$(\square^1 \pitchfork \mathsf{U}^0) \overset{\sim}{\twoheadrightarrow} \sum\nolimits_{(A,B:\mathsf{U}^0)}(A \to B \to \mathsf{U}^0)$$

This is not an isomorphism: the isomorphic copies $C_{n,k}$ have to be classified separately.

Identity types of the parametricity universe

Corollary

There is a $\mathrm{U}^0 \in \mathcal{E}^{\square^{\mathrm{op}}}$ *that classifies small maps, and such that there is a trivial fibration (i.e. a map with RLP against monos)*

$$(\square^1 \pitchfork \mathrm{U}^0) \overset{\sim}{\twoheadrightarrow} \textstyle\sum_{(A,B:\mathrm{U}^0)}(A \to B \to \mathrm{U}^0)$$

This is not an isomorphism: the isomorphic copies $C_{n,k}$ have to be classified separately.

Trivial fibrations have sections, so we interpret a syntactic retraction

$$(A \to B \to \mathrm{U}^0) \xrightarrow{\uparrow} \mathrm{Id}_{\mathrm{U}^0}(A,B) \xrightarrow{\downarrow} (A \to B \to \mathrm{U}^0) \qquad p{\uparrow}{\downarrow} \equiv p$$

Thus $\widehat{\square}$ with U^0 models a theory of internal parametricity, whose “identity types” consist of arbitrary correspondences.

Outline

1. The magic of semicartesian cubes

2. Paths in exponentials

3. The parametricity universe

4. The universe of fibrant types

5. Cubical spaces

6. Explaining the universe

Towards a universe of fibrant types

We'd like to define U to be

"the subtype of U^0 whose identity type
correspondences are one-to-one."

For any span $A \leftarrow C \rightarrow B$, i.e. correspondence $C : A \rightarrow B \rightarrow \mathsf{U}^0$, we have the type of assertions that it is one-to-one:

$$\mathsf{is11}(C) :\equiv \left(\prod_{(a:A)} \mathsf{isContr}(\textstyle\sum_{(b:B)} C(a,b))\right) \\ \times \left(\prod_{(b:B)} \mathsf{isContr}(\textstyle\sum_{(a:A)} C(a,b))\right)$$

If A, B, C lie in a slice $\mathcal{E}^{\square^{\mathrm{op}}}/X$, so does $\mathsf{is11}(C) \in \mathcal{E}^{\square^{\mathrm{op}}}/X$.

The universal correspondence

We pull back the universal type family along the adjunction counit:

$$\begin{tikzcd}
\bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \widetilde{\mathsf{U}^0} \arrow[d] \\
\square^1 \odot (\square^1 \pitchfork \mathsf{U}^0) \arrow[r] & \mathsf{U}^0
\end{tikzcd}$$

This yields a type family over the cylinder $\square^1 \odot (\square^1 \pitchfork \mathsf{U}^0)$, hence a universal correspondence over $\square^1 \pitchfork \mathsf{U}^0$:

$$\begin{tikzcd}
A^0 \arrow[dr] & C^0 \arrow[l] \arrow[r] \arrow[d] & B^0 \arrow[dl] \\
& \square^1 \pitchfork \mathsf{U}^0 &
\end{tikzcd}$$

Thus we have the classifying object $\mathsf{is11}(C^0) \in \mathcal{E}^{\square^{\mathrm{op}}}/(\square^1 \pitchfork \mathsf{U}^0)$.

BUT: this is a predicate on $\square^1 \pitchfork \mathsf{U}^0$, not U^0 itself.

A first-order approximation

We can fix this with the fiberwise amazing right adjoint:

$$\color{red}{\mathsf{U}^1} = \sqrt{\mathsf{is11}(C^0)}_{\mathsf{U}^0}.$$

Theorem

The classifying map $\Delta \to \mathsf{U}^0$ *of a type family* $\Delta \vdash P : \mathsf{U}^0$ *lifts to* U^1 *if and only if the correspondence* $\mathsf{Id}^{\varrho}_{\Delta.P}$ *is one-to-one.*

A first-order approximation

We can fix this with the fiberwise amazing right adjoint:

$$\mathsf{U}^1 = \sqrt{\mathsf{is11}(C^0)}_{\mathsf{U}^0}.$$

Theorem

The classifying map $\Delta \to \mathsf{U}^0$ *of a type family* $\Delta \vdash P : \mathsf{U}^0$ *lifts to* U^1 *if and only if the correspondence* $\mathsf{Id}^{\varrho}_{\Delta.P}$ *is one-to-one.*

BUT: This correspondence is still U^0-valued: even for $\Delta \vdash P : \mathsf{U}^1$,

$$\mathsf{Id}^{\varrho}_{\Delta.P} : P[\delta] \to P[\delta'] \to \mathsf{U}^0.$$

So we can't consistently use U^1 as "the" universe.

A second-order approximation

For any $A \leftarrow C \rightarrow B$ we have a type $\mathsf{classif}(C, \mathsf{U}^1)$ of U^1-valued classifying maps for C, i.e. pullback squares

$$\begin{tikzcd}
C \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \widetilde{\mathsf{U}^1} \arrow[d] \\
A \times B \arrow[r] & \mathsf{U}^1
\end{tikzcd}$$

Then we define a further improved universe:

$$\mathsf{U}^2 = \sqrt{\mathsf{is11}(C^1) \times \mathsf{classif}(C^1, \mathsf{U}^1)}_{\mathsf{U}^1}$$

The identity types of $\Delta \vdash P : \mathsf{U}^2$ are one-to-one and U^1-valued.

A limit construction

We continue inductively and take a limit:

$$\mathsf{U}^{n+1} = \sqrt{\mathsf{is11}(C^n) \times \mathsf{classif}(C^n, \mathsf{U}^n)}_{\mathsf{U}^n}$$

$$\mathsf{U} = \lim_n (\cdots \to \mathsf{U}^n \to \cdots \to \mathsf{U}^1 \to \mathsf{U}^0)$$

Theorem

The identity types of $\Delta \vdash P : \mathsf{U}$ are one-to-one and U-valued.

U classifies maps with contractible spaces of uniform Kan fillers.

Higher coinduction for Id_{U}

U is a higher coinductive type: the terminal coalgebra of a functor involving $\sqrt{}$.

- Its higher destructors assemble into

$$\downarrow : \mathsf{Id}_{\mathsf{U}}(A, B) \to \text{1-1-Corr}(A, B)$$

- The magic cylinder formula implies a formula for paths in $\sqrt{}$. Thus, Id_{U} is also a higher coinductive type.
- By higher coinduction (univ. prop. of lim and $\sqrt{}$) we define

$$\uparrow : \text{1-1-Corr}(A, B) \to \mathsf{Id}_{\mathsf{U}}(A, B)$$

such that $p{\uparrow}{\downarrow} \equiv p$.

Even if $\uparrow/\downarrow$ for U^0 were an isomorphism, this wouldn't be: Id_{U} contains more data than 1-1-Corr.

Fibrancy of type-formers

We lift all the type-formers from U^0 to U by higher coinduction. E.g. for Σ-types:

$$\begin{tikzcd}
\sum_{(A:\mathsf{U})}(A \to \mathsf{U}) \arrow[r, dashed, "\Sigma"] \arrow[d] & \mathsf{U} \arrow[d] \\
\sum_{(A:\mathsf{U}^0)}(A \to \mathsf{U}^0) \arrow[r, "\Sigma^0"] & \mathsf{U}^0
\end{tikzcd}$$

We must show that:

- Σ takes identifications to one-to-one correspondences.
- These correspondences are isomorphic to some Σ-type.

Strictifying identity types

This amounts to specifying the computation rules for ap_Σ and Id_Σ:

$$\mathsf{ap}_{X.Y.\sum_{(x:X)}Y(x)}(A_2, B_2) \equiv (\mathsf{Id}^{A_2,B_2}_{X.Y.\sum_{(x:X)}Y(x)}, \text{☁}, \text{☁})$$

$$\mathsf{Id}^{\varrho}_{\Delta.\sum_{(x:A)}B}(s,t) \equiv \sum\nolimits_{(q:\mathsf{Id}^{\varrho}_{\Delta.A}(\pi_1 s, \pi_1 t))} \mathsf{Id}^{\varrho,q}_{(\Delta,x:A).B}(\pi_2 s, \pi_2 t)$$

such that the latter equality holds up to isomorphism for powers $(\square^1 \pitchfork -)$ in $\mathcal{E}^{\square^{\mathrm{op}}}$.

- This works because the identity types of a Σ-type are another Σ-type (and similarly for all other type-formers).
- This is the coherence theorem strictifying $\mathsf{Id}_\Sigma \cong \text{☁}$ to a definitional equality.

Conclusion: cubical universes

Theorem-in-progress

H.O.T.T. has a model in $\mathcal{E}^{\square^{\mathrm{op}}}$, for any presheaf topos $\mathcal{E}$.
In particular, it has a model in $\widehat{\square}$.

Conjecture

By gluing with a global-sections or nerve functor valued in $\widehat{\square}$ or presheaves thereof, we can prove canonicity and normalization.

Note that

1. We must have symmetry in $\square$, to interpret Id_{Π} and Id_{U}.
2. We must have symmetry in syntax, for the nerve to lie in $\widehat{\square}$.

Outline

1. The magic of semicartesian cubes

2. Paths in exponentials

3. The parametricity universe

4. The universe of fibrant types

5. Cubical spaces

6. Explaining the universe

Towards higher topos models

Symmetry solves syntactic problems, but creates semantic ones: The "syntax-like" model in $\widehat{\square}$ doesn't present classical homotopy.

There could be an equivariant version. But there's another way.

Two approaches to defining higher homotopical structures:

1. As diagrams of sets
   - E.g. quasicategories
   - More parsimonious
2. As diagrams of spaces
   - E.g. complete Segal spaces
   - Often better-behaved

Cubical spaces

Let $\mathcal{E}$ be a type-theoretic model presheaf topos, e.g.:

- $\mathcal{E} =$ sSet, simplicial sets, with the Kan model structure (presents the homotopy theory of spaces).
- $\mathcal{E} =$ simplicial presheaves, with a left exact localization of the injective model structure (presents an $(\infty, 1)$-topos).

Theorem (cf. Rezk–Schwede–Shipley for the simplicial version)

The injective model structure on $\mathcal{E}^{\square^{\mathrm{op}}}$ admits a left Bousfield localization, called the realization model structure, such that:

1. *It is Quillen equivalent to $\mathcal{E}$.*
2. *It is also a type-theoretic model topos.*
   (Though not a left exact localization of the injective one.)

The universe of realization fibrations

Theorem

If $\mathsf{U}^{0,\mathsf{rlz}}$ *classifies realization fibrations, and* $\mathsf{U}^{\mathsf{rlz}} = \lim_n \mathsf{U}^{n,\mathsf{rlz}}$ *as before, there is a trivial fibration*

$$(\square^1 \pitchfork \mathsf{U}^{\mathsf{rlz}}) \xrightarrow{\sim\!\!\!\!\!\!\!\twoheadrightarrow} \textstyle\sum_{(A,B:\mathsf{U}^{\mathsf{rlz}})} \text{1-1-Corr}(A, B).$$

Corollary

The realization model structure interprets all of H.O.T.T.
Thus, H.O.T.T. has models in all Grothendieck $(\infty, 1)$*-toposes.*

Why Id_{U} has no η-rule

1. $\mathsf{Id}_{\mathsf{U}^0}(A, B)$ is not isomorphic to $A \to B \to \mathsf{U}^0$.
2. Id_{U} contains higher destructors in addition to 1-1-Corr.
3. Injective fibration structures over a cylinder contain more data than those on a span.
4. Homotopical constancy structures over a cylinder contain more data than those on a span.
5. Syntactically, Id_{U} must contain additional sym data.

Outline

1. The magic of semicartesian cubes
2. Paths in exponentials
3. The parametricity universe
4. The universe of fibrant types
5. Cubical spaces
6. Explaining the universe

What kind of type is the universe?

Traditionally, the universe is thought of (informally) as inductively defined, with constructors $\Sigma, \Pi, \ldots$, and Tarski eliminator El defined by recursion.

- Not true internally, but informs "meaning explanations" and inductive-recursive universe constructions.
- An observational Id would also be defined by recursion over these constructors, with clauses for Id_{Σ}, Id_{Π}, etc.

But:

- It's hard (not impossible) to make an inductively defined universe univalent.
- Suggests a closed universe, which has to be redefined whenever we add new type formers.

Co-meaning explanations

We instead consider U (still informally) to be coinductively defined.

- Now El and Id are destructors.
- Each type former Σ, Π, . . . is defined by corecursion, specifying its elements, and its identity types "of the same class". E.g. Σ is a corecursive function $\left(\sum_{(A:\mathsf{U})}(A \to \mathsf{U})\right) \to \mathsf{U}$, which makes sense because Id_Σ is another Σ-type.
- Explains an open universe: can introduce new type formers without redefining U, just applying its corecursion principle.
- The semantic universe of fibrant types is higher coinductive.

This gives a *philosophical* reason for the "coinductive" behavior of Id_U, having β but no η.

Back to Bishop

Recall Bishop's dicta:

> *A set is defined by describing exactly what must be done in order to construct an element of the set and what must be done in order to show that two elements are equal.*
>
> *An operation f from A into B is called a function if whenever $a, a' \in A$ and $a = a'$, we have $f(a) = f(a')$.*

Coinductive synthetic ∞-groupoids

Under propositions-as-types, this naturally becomes coinductive:

> *A type is defined by describing exactly what must be done in order to construct an element of the type and defining a type of ways to identify any two such elements.*
>
> *An operation f from A into B is called a function if for $a, a' : A$ we have a function from $a = a'$ to $f(a) = f(a')$.*

Coinductive synthetic ∞-groupoids

Under propositions-as-types, this naturally becomes coinductive:

> *A type is defined by describing exactly what must be done in order to construct an element of the type and defining a type of ways to identify any two such elements.*
>
> *An operation f from A into B is called a function if for $a, a' : A$ we have a function from $a = a'$ to $f(a) = f(a')$.*

If we augment it with a bit of univalence:

> *We define a type* U *whose elements are types, where two types are identified by a one-to-one correspondence.*
>
> *Every element of every type is identified with itself. For a type $A : \mathrm{U}$, this yields its own type of identifications.*

We get a philosophical vision that leads ineluctably to H.O.T.T., as a theory of coinductive ∞-groupoids.

Towards third generation HOTT

Part 2: Symmetries and semicartesian cubes

Michael Shulman

University of San Diego

joint work with Thorsten Altenkirch and Ambrus Kaposi

CMU HoTT Seminar
May 5, 2022

Up today

Plan for the three talks:

1. Basic syntax of H.O.T.T.
2. Symmetries and semicartesian cubes
3. Semantics of univalent universes

Outline

1. A calculus of telescopes

2. Some problems revealed by cubes

3. Symmetry solves all problems

4. Semicartesian cubes

5. Semantic identity types

Review

- Last week I described the “Book” version of H.O.T.T., starting with simple ideas, and introducing complexity only as necessary.
- By way of review, let’s reformulate the resulting theory more concisely and cleanly.

In particular, we eventually ended up with n-variable ap (and Id) that bind a finite list of variables:

$$\frac{\Gamma, x_1 : A_1, \dots, x_n : A_n \vdash t : B \qquad \cdots}{\Gamma \vdash \mathsf{ap}_{x_1 \dots . x_n . t}(p_1, \dots, p_n) : \mathsf{Id}_B(\cdots)}$$

Such a “context suffix” is also called a telescope.
We now reify these into a “telescope calculus”.

Telescopes

Telescopes are defined inductively as finite lists of types:

$$\frac{}{\Gamma \vdash \epsilon \ \mathsf{tel}} \qquad \frac{\Gamma \vdash \Delta \ \mathsf{tel} \qquad \Gamma, \Delta \vdash A : \mathsf{U}}{\Gamma \vdash (\Delta, x : A) \ \mathsf{tel}}$$

The "elements" of a telescope are substitutions:

$$\frac{}{() : \epsilon} \qquad \frac{\delta : \Delta \qquad \Delta \vdash A : \mathsf{U} \qquad a : A[\delta]}{(\delta, a) : (\Delta, x : A)}$$

These are defined mutually with their action on terms (and types):

$$\frac{\Delta \vdash a : A \qquad \delta : \Delta}{a[\delta] : A[\delta]}$$

Dependent Id and ap with telescopes

Now we can define identity telescopes from identity types:

$$\frac{\Delta\ \mathsf{tel} \qquad \delta : \Delta \qquad \delta' : \Delta}{\mathsf{Id}_\Delta(\delta, \delta')\ \mathsf{tel}} \qquad\qquad \mathsf{Id}_\epsilon((),()) \equiv \epsilon$$

$$\mathsf{Id}_{(\Delta, x:A)}((\delta, a), (\delta', a')) \equiv \big(\varrho : \mathsf{Id}_\Delta(\delta, \delta'),\ \alpha : \mathsf{Id}^{\varrho}_{\Delta.A}(a, a')\big)$$

These are defined mutually with *n*-ary Id, which depends on them:

$$\frac{\varrho : \mathsf{Id}_\Delta(\delta, \delta') \qquad \Delta \vdash A : \mathsf{U} \qquad a : A[\delta] \qquad a' : A[\delta']}{\mathsf{Id}^{\varrho}_{\Delta.A}(a, a') : \mathsf{U}}$$

We write $\mathsf{Id}_A(a, a') \equiv \mathsf{Id}^{()}_{\epsilon.A}(a, a')$ in the non-dependent case.

(Last time I defined dependent Id in terms of ap; here we postulate it separately and then make them coincide later.)

Computing Id

As we saw last time, Id computes on all type formers:

$$\mathsf{Id}^{\varrho}_{\Delta.A\times B}(s,t) \equiv \mathsf{Id}^{\varrho}_{\Delta.A}(\pi_1 s,\pi_1 t)\times \mathsf{Id}^{\varrho}_{\Delta.B}(\pi_2 s,\pi_2 t)$$

$$\mathsf{Id}^{\varrho}_{\Delta.\sum_{(x:A)}B}(s,t) \equiv \textstyle\sum_{(q:\mathsf{Id}^{\varrho}_{\Delta.A}(\pi_1 s,\pi_1 t))}\mathsf{Id}^{\varrho,q}_{(\Delta,x:A).B}(\pi_2 s,\pi_2 t)$$

$$\mathsf{Id}^{\varrho}_{A\to B}(f,g)\equiv \textstyle\prod_{(u:A)}\prod_{(v:A)}\prod_{(q:\mathsf{Id}^{\varrho}_{\Delta.A}(u,v))}\mathsf{Id}^{\varrho}_{\Delta.B}(fu,gv)$$

$$\mathsf{Id}^{\varrho}_{\prod_{(x:A)}B}(f,g)\equiv \textstyle\prod_{(u:A)}\prod_{(v:A)}\prod_{(q:\mathsf{Id}^{\varrho}_{\Delta.A}(u,v))}\mathsf{Id}^{\varrho,q}_{(\Delta,x:A).B}(fu,gv)$$

Id is a 1-1 correspondence

All identity types are 1-1 correspondences:

$$\frac{\varrho : \mathsf{Id}_\Delta(\delta,\delta') \qquad \Delta \vdash A : \mathsf{U} \qquad a : A[\delta]}{\overrightarrow{\mathsf{corr}}^{\varrho}_{\Delta.A}(a) : \mathsf{isContr}\left(\sum_{(a' : A[\delta'])} \mathsf{Id}^{\varrho}_{\Delta.A}(a,a')\right)}$$

$$\frac{\varrho : \mathsf{Id}_\Delta(\delta,\delta') \qquad \Delta \vdash A : \mathsf{U} \qquad a' : A[\delta']}{\overleftarrow{\mathsf{corr}}^{\varrho}_{\Delta.A}(a') : \mathsf{isContr}\left(\sum_{(a : A[\delta])} \mathsf{Id}^{\varrho}_{\Delta.A}(a,a')\right)}$$

The centers of contraction constitute transport:

$$\frac{\varrho : \mathsf{Id}_\Delta(\delta,\delta') \qquad \Delta \vdash A : \mathsf{U} \qquad a : A[\delta]}{\overrightarrow{\mathsf{tr}}^{\varrho}_{\Delta.A}(a) : A[\delta'] \qquad \overrightarrow{\mathsf{lift}}^{\varrho}_{\Delta.A}(a) : \mathsf{Id}^{\varrho}_{\Delta.A}(a, \overrightarrow{\mathsf{tr}}^{\varrho}_{\Delta.A}(a))}$$

These witnesses compute on type formers: $\overrightarrow{\mathsf{corr}}^{\varrho}_{\Delta.A\times B}(a) \equiv \cdots$,

hence also $\overrightarrow{\mathsf{tr}}^{\varrho}_{\Delta.A\times B}(a) \equiv \cdots$, etc.

Computing ap

A term can be applied to Id of any telescope it depends on:

$$\frac{\varrho : \mathsf{Id}_{\Delta}(\delta, \delta') \qquad \Delta \vdash t : B}{\mathsf{ap}_{\Delta.t}(\varrho) : \mathsf{Id}^{\varrho}_{\Delta.B}(t[\delta], t[\delta'])}$$

This higher-dimensional explicit substitution computes on all* terms:

$$\mathsf{ap}_{\Delta.(s,t)}(\varrho) \equiv (\mathsf{ap}_{\Delta.s}(\varrho), \mathsf{ap}_{\Delta.t}(\varrho))$$

$$\mathsf{ap}_{\Delta.\pi_1 s}(\varrho) \equiv \pi_1\, \mathsf{ap}_{\Delta.s}(\varrho) \qquad \mathsf{ap}_{\Delta.\pi_2 s}(\varrho) \equiv \pi_2\, \mathsf{ap}_{\Delta.s}(\varrho)$$

$$\mathsf{ap}_{\Delta.f\,b}(\varrho) \equiv \mathsf{ap}_{\Delta.f}(p)\big(b[a/x],\ b[a'/x],\ \mathsf{ap}_{\Delta.b}(\varrho)\big).$$

$$\mathsf{ap}_{\Delta.(\lambda y.t)}(\varrho) \equiv \lambda u.\lambda v.\lambda q.\mathsf{ap}_{\Delta.y.t}(\varrho, q).$$

We define reflexivity as the 0-ary ap: $\mathsf{refl}_a \equiv \mathsf{ap}_{\epsilon.a}()$.

Univalence

$\mathsf{Id}_{\mathsf{U}}(A, B)$ contains as a retract the type of 1-1 correspondences:

$$\text{1-1-Corr}(A,B) :\equiv \sum_{(R:A\to B\to \mathsf{U})} \Big(\prod_{(a:A)} \mathsf{isContr}(\sum_{(b:B)} R(a,b))\Big) \\ \times \Big(\prod_{(b:B)} \mathsf{isContr}(\sum_{(a:A)} R(a,b))\Big).$$

$$\boxed{\text{1-1-Corr}(A,B) \xrightarrow{\uparrow} \mathsf{Id}_U(A,B) \xrightarrow{\downarrow} \text{1-1-Corr}(A,B)} \qquad \boxed{p\uparrow\downarrow \equiv p}$$

We identify dependent Id with ap into the universe:

$$\begin{aligned} \mathsf{Id}^{\varrho}_{\Delta.B}(b,b') &\equiv \pi_1(\mathsf{ap}_{\Delta.B}(\varrho)\downarrow)(b,b') \\ \overrightarrow{\mathsf{corr}}^{\varrho}_{\Delta.B}(b,b') &\equiv \pi_1\pi_2(\mathsf{ap}_{\Delta.B}(\varrho)\downarrow)(b,b') \\ \overleftarrow{\mathsf{corr}}^{\varrho}_{\Delta.B}(b,b') &\equiv \pi_2\pi_2(\mathsf{ap}_{\Delta.B}(\varrho)\downarrow)(b,b') \end{aligned}$$

(Last time, we defined the LHS as the RHS. Separating them is more natural for Tarski universes, and permits types not lying in any universe.)

That asterisk: Neutral reflexivities

I claimed that ap is never a normal form, but there's one exception:

When y is a variable, refl_y is neutral (hence normal).

Since refl is nullary ap, the rule that would apply is

$$\mathsf{ap}_{x_1.\cdots x_n.y}(p_1,\ldots,p_n) \equiv \mathsf{refl}_y \text{ (if } y \text{ is a variable} \notin \{x_1,\ldots,x_n\})$$

where $n = 0$, but this just reduces $\mathsf{refl}_y \equiv \mathsf{ap}_{().y}()$ to itself!

This includes other terms that obviously must also be neutral:

- $\mathsf{ap}_{x.f(x)}(p) \equiv \mathsf{refl}_f(a_0, a_1, p)$ for a variable $f : A \to B$.
- $\mathsf{Id}_A(a_0, a_1) \equiv (\pi_1\, \mathsf{refl}_A)(a_0, a_1)$ for a variable $A : \mathsf{U}$.

Similarly, $\mathsf{refl}_{\mathsf{refl}_x}$, $\mathsf{refl}_{\mathsf{refl}_{\mathsf{refl}_x}}$, etc., are also neutral.

Outline

1. A calculus of telescopes
2. Some problems revealed by cubes
3. Symmetry solves all problems
4. Semicartesian cubes
5. Semantic identity types

Squares and cubes

H.O.T.T. is not a “cubical type theory”: there are no explicit cubes in the syntax. But like any other type theory with dependent identity types (including Book HoTT!), it has an emergent notion of cube:

$$a_{02} : \mathsf{Id}_A(a_{00}, a_{01}) \qquad a_{12} : \mathsf{Id}_A(a_{10}, a_{11}) \qquad a_{20} : \mathsf{Id}_A(a_{00}, a_{10})$$

$$a_{21} : \mathsf{Id}_A(a_{01}, a_{11}) \qquad a_{22} : \mathsf{Id}^{a_{02}, a_{12}}_{x.y.\mathsf{Id}_A(x,y)}(a_{20}, a_{21})$$

$$\begin{tikzcd}
a_{10} \arrow[r, "a_{12}"] & a_{11} \\
a_{00} \arrow[u, "a_{20}"] \arrow[r, "a_{02}"'] \arrow[ur, phantom, "a_{22}"] & a_{01} \arrow[u, "a_{21}"']
\end{tikzcd}$$

Similarly, $\mathsf{Id}_{\mathsf{Id}_{\mathsf{Id}_A}}$ is a type of 3-dimensional cubes, etc.

Very important point

The roles of a_{02}, a_{12} and a_{20}, a_{21} are asymmetrical!

Cubical horn-fillers

Given a_{02}, a_{12}, a_{20}, we have fillers of left-to-right cubical horns:

$$\overrightarrow{\mathsf{tr}}^{a_{02},a_{12}}_{x.y.\mathsf{Id}_A(x,y)}(a_{20}) : \mathsf{Id}_A(a_{01}, a_{11})$$

$$\overrightarrow{\mathsf{lift}}^{a_{02},a_{12}}_{x.y.\mathsf{Id}_A(x,y)}(a_{20}) : \mathsf{Id}^{a_{02},a_{12}}_{x.y.\mathsf{Id}_A(x,y)}(a_{20}, \overrightarrow{\mathsf{tr}}^{a_{02},a_{12}}_{x.y.\mathsf{Id}_A(x,y)}(a_{20}))$$

$$\begin{tikzcd}
a_{10} \arrow[rr, "a_{12}"] & & a_{11} \\
a_{00} \arrow[u, "a_{20}"] \arrow[rr, "a_{02}"'] \arrow[urr, phantom, "\overrightarrow{\mathsf{lift}}^{a_{02},a_{12}}_{x.y.\mathsf{Id}_A(x,y)}(a_{20})"] & & a_{01} \arrow[u, "\overrightarrow{\mathsf{tr}}^{a_{02},a_{12}}_{x.y.\mathsf{Id}_A(x,y)}(a_{20})"']
\end{tikzcd}$$

Similarly, $\overleftarrow{\mathsf{tr}}$ and $\overleftarrow{\mathsf{lift}}$ fill right-to-left cubical horns.
And $\overrightarrow{\mathsf{tr}}_{\mathsf{Id}_{\mathsf{Id}_A}}$, etc. fill higher-dimensional left-right horns.

Problem #1

We don't seem to have top-to-bottom or bottom-to-top fillers.

Degenerate cubes

Given $a_2 : \mathsf{Id}_A(a_0, a_1)$, there are two degenerate squares:

$$\begin{aligned}
\mathsf{refl}_{a_2} &: \mathsf{Id}_{\mathsf{Id}_A(a_0,a_1)}(a_2, a_2) &&\equiv \mathsf{Id}^{\mathsf{refl}_{a_0}, \mathsf{refl}_{a_1}}_{x.y.\mathsf{Id}_A(x,y)}(a_2, a_2) \\
\mathsf{ap}_{x.\mathsf{refl}_x}(a_2) &: \mathsf{Id}^{a_2}_{x.\mathsf{Id}_A(x,x)}(\mathsf{refl}_{a_0}, \mathsf{refl}_{a_1}) &&\equiv \mathsf{Id}^{a_2,a_2}_{x.y.\mathsf{Id}_A(x,y)}(\mathsf{refl}_{a_0}, \mathsf{refl}_{a_1})
\end{aligned}$$

$$\begin{tikzcd}
a_1 \arrow[r, "\mathsf{refl}_{a_1}"] & a_1 \\
a_0 \arrow[u, "a_2"] \arrow[r, "\mathsf{refl}_{a_0}"'] \arrow[ur, phantom, "\mathsf{refl}_{a_2}"] & a_0 \arrow[u, "a_2"']
\end{tikzcd}
\qquad
\begin{tikzcd}
a_0 \arrow[r, "a_2"] & a_1 \\
a_0 \arrow[u, "\mathsf{refl}_{a_0}"] \arrow[r, "a_2"'] \arrow[ur, phantom, "\mathsf{ap}_{x.\mathsf{refl}_x}(a_2)"] & a_1 \arrow[u, "\mathsf{refl}_{a_1}"']
\end{tikzcd}$$

Degenerate cubes

Given $a_2 : \mathsf{Id}_A(a_0, a_1)$, there are two degenerate squares:

$$\begin{tikzcd}
a_1 \arrow[r, "\mathsf{refl}_{a_1}"] & a_1 & & a_0 \arrow[r, "a_2"] & a_1 \\
a_0 \arrow[u, "a_2"] \arrow[r, "\mathsf{refl}_{a_0}"'] \arrow[ur, phantom, "\mathsf{refl}_{a_2}"] & a_0 \arrow[u, "a_2"'] & & a_0 \arrow[u, "\mathsf{refl}_{a_0}"] \arrow[r, "a_2"'] \arrow[ur, phantom, "\mathsf{ap}_{x.\mathsf{refl}_x}(a_2)"] & a_1 \arrow[u, "\mathsf{refl}_{a_1}"']
\end{tikzcd}$$

Problem #2

For $a : A$, the two doubly-degenerate squares

$$\begin{tikzcd}
a \arrow[r, "\mathsf{refl}_a"] & a & & a \arrow[r, "\mathsf{refl}_a"] & a \\
a \arrow[u, "\mathsf{refl}_a"] \arrow[r, "\mathsf{refl}_a"'] \arrow[ur, phantom, "\mathsf{refl}_{\mathsf{refl}_a}"] & a \arrow[u, "\mathsf{refl}_a"'] & & a \arrow[u, "\mathsf{refl}_a"] \arrow[r, "\mathsf{refl}_a"'] \arrow[ur, phantom, "\mathsf{ap}_{x.\mathsf{refl}_x}(\mathsf{refl}_a)"] & a \arrow[u, "\mathsf{refl}_a"']
\end{tikzcd}$$

seem to be definitionally unrelated.

Stuck degeneracies break canonicity

Problem #3

Our rules so far compute refl_{a_2} based on the structure of a_2, but $\mathsf{ap}_{x.\mathsf{refl}_x}(a_2)$ is stuck, even if a_2 is very concrete.

- refl_x doesn't reduce when x is a variable.
- ap doesn't inspect its identification argument.

Stuck degeneracies break canonicity

Problem #3

Our rules so far compute refl_{a_2} based on the structure of a_2, but $\mathsf{ap}_{x.\mathsf{refl}_x}(a_2)$ is stuck, even if a_2 is very concrete.

- refl_x doesn't reduce when x is a variable.
- ap doesn't inspect its identification argument.

A bit nonobviously, this also breaks canonicity for $\mathbb{N}$.

Intuitive homotopy-theoretic reason

For a type $A : \mathsf{U}$, the square $\mathsf{ap}_{x.\mathsf{refl}_x}(\mathsf{refl}_A)$ in U is essentially a self-homotopy of the identity equivalence of A, i.e. $\prod_{(a:A)} \mathsf{Id}_A(a, a)$. Taking $A = S^1$ we get a stuck loop in $\mathsf{Id}_{S^1}(\mathsf{base}, \mathsf{base})$, hence in $\mathbb{Z}$.

(There's also an explicit argument using two universes instead of S^1.)

Outline

1. A calculus of telescopes
2. Some problems revealed by cubes
3. Symmetry solves all problems
4. Semicartesian cubes
5. Semantic identity types

Symmetry

To solve these problems, we introduce a symmetry operation that transposes squares:

$$\begin{tikzcd} a_{10} \arrow[r, "a_{12}"] & a_{11} & & a_{01} \arrow[r, "a_{21}"] & a_{11} \\ a_{00} \arrow[u, "a_{20}"] \arrow[r, "a_{02}"'] \arrow[ur, phantom, "a_{22}"] & a_{01} \arrow[u, "a_{21}"'] \arrow[r, phantom, "\mapsto"] & & a_{00} \arrow[u, "a_{02}"] \arrow[r, "a_{20}"'] \arrow[ur, phantom, "\mathsf{sym}_A(a_{22})"] & a_{10} \arrow[u, "a_{12}"'] \end{tikzcd}$$

$$\frac{a_{22} : \mathsf{Id}^{a_{02},a_{12}}_{x.y.\mathsf{Id}_A(x,y)}(a_{20}, a_{21})}{\mathsf{sym}_A(a_{22}) : \mathsf{Id}^{a_{20},a_{21}}_{x.y.\mathsf{Id}_A(x,y)}(a_{02}, a_{12})}$$

The other Kan operations

Now we can fill other cubical horns, solving problem #1:

$$\begin{tikzcd}
a_{10} & & a_{11} & & a_{01} \arrow[r, "a_{21}"] & a_{11} \\
a_{00} \arrow[u, "a_{20}"] \arrow[rr, "a_{02}"'] & & a_{01} \arrow[u, "a_{21}"'] & \rightsquigarrow & a_{00} \arrow[u, "a_{02}"] \arrow[r, "a_{20}"'] & a_{10}
\end{tikzcd}$$

$$\begin{tikzcd}[column sep=huge]
a_{01} \arrow[r, "a_{21}"] & a_{11} \\
a_{00} \arrow[u, "a_{02}"] \arrow[r, "a_{20}"'] \arrow[ur, phantom, "\overrightarrow{\mathrm{lift}}^{a_{20},a_{21}}_{x.y.\mathrm{Id}_A(x,y)}(a_{02})"] & a_{10} \arrow[u, "\overrightarrow{\mathrm{tr}}^{a_{20},a_{21}}_{x.y.\mathrm{Id}_A(x,y)}(a_{02})"']
\end{tikzcd}$$

$$\begin{tikzcd}[column sep=huge]
a_{10} \arrow[r, "\overrightarrow{\mathrm{tr}}^{a_{20},a_{21}}_{x.y.\mathrm{Id}_A(x,y)}(a_{02})"] & a_{11} \\
a_{00} \arrow[u, "a_{20}"] \arrow[r, "a_{02}"'] \arrow[ur, phantom, "\mathrm{sym}_A(\overrightarrow{\mathrm{lift}}^{a_{20},a_{21}}_{x.y.\mathrm{Id}_A(x,y)}(a_{02}))"] & a_{01} \arrow[u, "a_{21}"']
\end{tikzcd}$$

Computing symmetry

To solve problem #3, we define

$$\mathsf{ap}_{x.\mathsf{refl}_x}(a_2) \equiv \mathsf{sym}_A(\mathsf{refl}_{a_2}).$$

This computes based on a_2… if sym also computes!

Computing symmetry

To solve problem #3, we define

$$\mathsf{ap}_{x.\mathsf{refl}_x}(a_2) \equiv \mathsf{sym}_A(\mathsf{refl}_{a_2}).$$

This computes based on a_2... if sym also computes!

For the most part, computing symmetry is straightforward, e.g.:

$$\begin{aligned}
&\mathsf{Id}^{s_{02},s_{12}}_{u.v.\mathsf{Id}_{A\times B}(u,v)}(s_{20}, s_{21}) \\
&\qquad \equiv \mathsf{Id}^{s_{02},s_{12}}_{u.v.\mathsf{Id}_A(\pi_1 u,\pi_1 v)\times \mathsf{Id}_B(\pi_2 u,\pi_2 v)}(s_{20}, s_{21}) \\
&\qquad \equiv \mathsf{Id}^{s_{02},s_{12}}_{u.v.\mathsf{Id}_A(\pi_1 u,\pi_1 v)}(\pi_1 s_{20}, \pi_1 s_{21}) \times \mathsf{Id}^{s_{02},s_{12}}_{u.v.\mathsf{Id}_B(\pi_2 u,\pi_2 v)}(\pi_2 s_{20}, \pi_2 s_{21}) \\
&\qquad \equiv \mathsf{Id}^{\pi_1 s_{02},\pi_1 s_{12}}_{x.w.\mathsf{Id}_A(x,w)}(\pi_1 s_{20}, \pi_1 s_{21}) \times \mathsf{Id}^{\pi_2 s_{02},\pi_2 s_{12}}_{y.z.\mathsf{Id}_B(y,z)}(\pi_2 s_{20}, \pi_2 s_{21}).
\end{aligned}$$

So we can define

$$\mathsf{sym}_{A\times B}((p,q)) \equiv (\mathsf{sym}_A(p), \mathsf{sym}_B(q))$$

Dependent symmetry

To generalize this to Σ-types, we need dependent symmetry over a square in a telescope (don't worry too much about the syntax):

$$\frac{\delta_{22} : \mathsf{Id}^{\delta_{02},\delta_{12}}_{\delta.\delta'.\mathsf{Id}_{\Delta}(\delta,\delta')}(\delta_{20},\delta_{21}) \qquad a_{22} : \mathsf{Id}^{\delta_{02},\delta_{12},\delta_{22},a_{02},a_{12}}_{\delta.\delta'.\varrho.u.v.\mathsf{Id}^{\varrho}_{\Delta.A}(u,v)}(a_{20},a_{21})}{\mathsf{sym}^{\delta_{22}}_{\Delta.A}(a_{22}) : \mathsf{Id}^{\delta_{20},\delta_{21},\mathsf{sym}(\delta_{22}),a_{20},a_{21}}_{\delta.\delta'.\varrho.u.v.\mathsf{Id}^{\varrho}_{\Delta.A}(u,v)}(a_{02},a_{12})}$$

Then we can define

$$\mathsf{sym}^{\delta_{22}}_{\Delta.\sum_{(x:A)}B}((p,q)) \equiv (\mathsf{sym}^{\delta_{22}}_{\Delta.A}(p), \mathsf{sym}^{\delta_{22},p}_{(\Delta,x:A).B}(q))$$

Symmetry for functions

$$
\begin{aligned}
\mathsf{Id}^{f_{02},f_{12}}_{f.g.\mathsf{Id}_{A\to B}(f,g)}(f_{20},f_{21}) &\equiv \mathsf{Id}^{f_{02},f_{12}}_{f.g.\prod_{(x_0:A)}\prod_{(x_1:A)}\prod_{(x_2:\mathsf{Id}_A(x_0,x_1))}\mathsf{Id}_B(fx_0,gx_1)}(f_{20},f_{21}) \\
&\equiv \prod_{(x_{00}:A)}\prod_{(x_{01}:A)}\prod_{(x_{02}:\mathsf{Id}_A(x_{00},x_{01}))} \\
&\qquad \prod_{(x_{10}:A)}\prod_{(x_{11}:A)}\prod_{(x_{12}:\mathsf{Id}_A(x_{10},x_{11}))} \\
&\qquad \prod_{(x_{20}:\mathsf{Id}_A(x_{00},x_{10}))}\prod_{(x_{21}:\mathsf{Id}_A(x_{01},x_{11}))}\prod_{(x_{22}:\mathsf{Id}^{x_{02},x_{12}}_{x.y.\mathsf{Id}_A(x,y)}(x_{20},x_{21}))} \\
&\qquad\quad \mathsf{Id}^{f_{02}x_{02},f_{12}x_{12}}_{u.v.\mathsf{Id}_B(u,v)}(f_{20}x_{20},f_{21}x_{21})
\end{aligned}
$$

So $f_{22} : \mathsf{Id}^{f_{02},f_{12}}_{f.g.\mathsf{Id}_{A\to B}(f,g)}(f_{20},f_{21})$ is a function from squares in A, with arbitrary boundary, to squares in B with specified boundary.

Thus we define $\mathsf{sym}_{A\to B}$ by transposing both input and output:

$$
\begin{aligned}
&\mathsf{sym}_{A\to B}(f_{22})(x_{00},x_{10},x_{20},x_{01},x_{11},x_{21},x_{02},x_{12},x_{22}) \\
&\quad \equiv \mathsf{sym}(f_{22}(x_{00},x_{01},x_{02},x_{10},x_{11},x_{12},x_{20},x_{21},\mathsf{sym}(x_{22})))
\end{aligned}
$$

Symmetry for Π-types is similar, using dependent symmetry.

Rules for symmetry

Some obvious rules for symmetry are that it should be an involution:

$$\mathsf{sym}_A(\mathsf{sym}_A(a_{22})) \equiv a_{22}$$

and it should commute with iterated ap on squares:

$$\mathsf{sym}_B(\mathsf{ap}_{\mathsf{ap}_f}(a_{22})) \equiv \mathsf{ap}_{\mathsf{ap}_f}(\mathsf{sym}_A(a_{22}))$$

The nullary case of the latter is $\mathsf{sym}(\mathsf{refl}_{\mathsf{refl}_a}) \equiv \mathsf{refl}_{\mathsf{refl}_a}$.
This solves problem #2:

$$\mathsf{ap}_{x.\mathsf{refl}_x}(\mathsf{refl}_a) \equiv \mathsf{sym}(\mathsf{refl}_{\mathsf{refl}_a}) \equiv \mathsf{refl}_{\mathsf{refl}_a}$$

Higher-dimensional symmetry

For *n*-dimensional cubes (i.e. *n*-fold iterated Id-types):

- We would expect symmetries to permute all n dimensions. The symmetric group S_n should act on n-cubes.
- We have transpositions of adjacent dimensions, from our sym. (E.g. $\mathsf{sym}_{\mathsf{Id}_A} : \mathsf{Id}_{\mathsf{Id}_{\mathsf{Id}_A}} \to \mathsf{Id}_{\mathsf{Id}_{\mathsf{Id}_A}}$ and $\mathsf{ap}_{\mathsf{sym}_A} : \mathsf{Id}_{\mathsf{Id}_{\mathsf{Id}_A}} \to \mathsf{Id}_{\mathsf{Id}_{\mathsf{Id}_A}}$.)

Fortunately, S_n is generated by adjacent transpositions!

$$S_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{l} \sigma_k \sigma_k = 1 \\ \sigma_j \sigma_k = \sigma_k \sigma_j \quad (j+1 < k) \\ \sigma_k \sigma_{k+1} \sigma_k = \sigma_{k+1} \sigma_k \sigma_{k+1} \end{array} \right\rangle$$

The first two relations follow from the equations on the last slide. To obtain the third, we assert

$$\boxed{\mathsf{sym}_{\mathsf{Id}_A}(\mathsf{ap}_{\mathsf{sym}_A}(\mathsf{sym}_{\mathsf{Id}_A}(a_{222}))) \equiv \mathsf{ap}_{\mathsf{sym}_A}(\mathsf{sym}_{\mathsf{Id}_A}(\mathsf{ap}_{\mathsf{sym}_A}(a_{222})))}\,.$$

Outline

1. A calculus of telescopes
2. Some problems revealed by cubes
3. Symmetry solves all problems
4. Semicartesian cubes
5. Semantic identity types

Towards computation by gluing

Symmetry computes the previously stuck term $\mathsf{ap}_{x.\mathsf{refl}_x}(a_2)$.
But how do we know there aren't other stuck terms?

Obviously, by proving canonicity/normalization.

We haven't done this yet, but the first step (from a modern perspective) is constructing a set-based semantic model to be the codomain for Artin gluing.

Identity contexts

Question

What categorical structure corresponds to our identity types?

- The objects of a category $\mathcal{C}$ correspond to syntactic contexts.
- The fundamental operation on contexts takes Δ to

$$\mathsf{ID}_\Delta :\equiv \big(\delta : \Delta, \delta' : \Delta, \varrho : \mathsf{Id}_\Delta(\delta, \delta,')\big).$$

which factors the diagonal (i.e. is a path object):

$$\Delta \xrightarrow{\mathsf{refl}} \mathsf{ID}_\Delta \to (\delta : \Delta, \delta' : \Delta) \cong \Delta \times \Delta.$$

- This operation is functorial (via ap).
- We have natural symmetries $\mathsf{ID}_{\mathsf{ID}_\Delta} \cong \mathsf{ID}_{\mathsf{ID}_\Delta}$, yielding an S_n-action on n-fold identity contexts..

Cubical actions

Thus, an ID-structure on $\mathcal{C}$ is the same as

- A functor $\mathrm{ID} : \mathcal{C} \to \mathcal{C}$
- Nat. trans. $r : 1_{\mathcal{C}} \to \mathrm{ID}$ and $s, t : \mathrm{ID} \rightrightarrows 1_{\mathcal{C}}$ with $sr = tr = 1_{1_{\mathcal{C}}}$
- Natural symmetries $\mathrm{ID} \circ \mathrm{ID} \cong \mathrm{ID} \circ \mathrm{ID}$ satisfying S_n relations.

Definition

Let $\square^{\mathrm{op}}$ be the monoidal category freely generated by an object $\mathbb{I}$, morphisms $r : \mathbb{1} \to \mathbb{I}$ and $s, t : \mathbb{I} \to \mathbb{1}$ with $sr = tr = 1_{\mathbb{1}}$, where $\mathbb{1}$ is the unit, and symmetries $\mathbb{I} \otimes \mathbb{I} \cong \mathbb{I} \otimes \mathbb{I}$ satisfying S_n relations.

Then an ID-structure on $\mathcal{C}$ is also equivalently

- A monoidal functor $\square^{\mathrm{op}} \to [\mathcal{C}, \mathcal{C}]$

and therefore also equivalently

- A coherent action $\square^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$.

The semicartesian cube category

- $\square$ is a semicartesian monoidal category: symmetric monoidal and its unit $\mathbb{1}$ is terminal. Projections, but no diagonals.
- It is also the semicartesian monoidal category freely generated by an object $\mathbb{I}$ and morphisms $s, t : \mathbb{1} \to \mathbb{I}$.

We call $\square$ the semicartesian cube category.

This is the category used by:

- Bernardy–Coquand–Moulin, for internal parametricity (actually they used a unary version, this would be the binary one)
- Bezem–Coquand–Huber, for the original cubical model
- Cavallo–Harper, for the parametricity direction of parametric cubical type theory

Enrichment

The presheaf category $\widehat{\mathbb{\Box}} = \mathsf{Set}^{\mathbb{\Box}^{\mathrm{op}}}$ inherits a Day convolution monoidal structure (also semicartesian):

$$(X \otimes Y)_n = \int^{k,\ell} X_k \times Y_\ell \times \mathbb{\Box}(n, k \oplus \ell).$$

We write $\square^n$ for the representable $\mathbb{\Box}(-, \mathbb{I}^{\otimes n})$. Note $\square^0$ is terminal.

Theorem

An action $\triangleright : \mathbb{\Box}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ is the same as an enrichment of $\mathcal{C}$ over $\widehat{\mathbb{\Box}}$ that has powers by representables (write $\square^n \pitchfork X \equiv \mathbb{I}^{\otimes n} \triangleright X$).

$$\mathsf{Map}(A, B)_n :\equiv \mathcal{C}(A, \square^n \pitchfork B)$$

$$\widehat{\mathbb{\Box}}(X, \mathsf{Map}(A, \square^n \pitchfork B)) \cong \widehat{\mathbb{\Box}}(X \otimes \square^n, \mathsf{Map}(A, B))$$

$\widehat{\mathbb{\Box}}$-enriched categories are the natural home for H.O.T.T. semantics.

Cubical objects

Of course, $\widehat{\mathbb{\square}}$ is enriched over itself.

Similarly, any category $\mathcal{E}^{\mathbb{\square}^{\mathrm{op}}}$ of cubical objects is $\widehat{\mathbb{\square}}$-enriched, with powers and copowers if $\mathcal{E}$ is complete and cocomplete:

$$
\begin{aligned}
(A \odot X)_n &= \int^{k,\ell} (A_k \times \mathbb{\square}(n, k \oplus \ell)) \cdot X_\ell \\
(A \pitchfork X)_n &= \int_{k,\ell} (X_k)^{A_\ell \times \mathbb{\square}(k, n \oplus \ell)} \\
(\square^m \pitchfork X)_n &= X_{n \oplus m} \\
\mathrm{Map}(X, Y)_n &= \mathcal{E}^{\mathbb{\square}^{\mathrm{op}}}(X, \square^n \pitchfork Y)
\end{aligned}
$$

More about the cube category

Up to equivalence:

- The objects of $\square$ are finite sets.
- A morphism $\phi \in \square(m, n)$ is a function $\phi : n \to m \sqcup \{-, +\}$ that is injective on the preimage of m.
- The monoidal structure $m \oplus n$ is disjoint union.

Sometimes use a skeletal version with objects $\underline{n} = \{0, 1, \ldots, n-1\}$, but often the non-skeletal version with all finite sets is better.

- The coface $\delta_{k,\pm} \in \square(n \smallsetminus \{k\}, n)$ is the identity on $n \smallsetminus \{k\}$ and sends k to $\pm$.
- The codegeneracy $\sigma_k \in \square(n, n \smallsetminus \{k\})$ is the inclusion.
- The endomorphism monoid $\square(n, n)$ is the symmetric group S_n.

The magic of semicartesian cubes

The monoidal structure of $\Box$ is "almost" cartesian; only the injectivity requirement spoils it. If it were cartesian we would have

$$¿\quad \Box(n, k \oplus \ell) \cong \Box(n,k) \times \Box(n,\ell). \quad ?$$

Instead, we have

$$\Box(n, k \oplus \ell) \cong \sum_{\phi : \Box(n,k)} \Box(n \smallsetminus \phi(k), \ell).$$

Removing $\phi(k)$ from the second domain ensures the copaired function $k \sqcup \ell \to n \sqcup \{-,+\}$ is still injective on the preimage of n.

But in some ways this is even better!

Copowers by representables

For $A \in \widehat{\boxbar}$ and $X \in \mathcal{E}^{\boxbar^{\mathrm{op}}}$, we have

$$\begin{aligned}
(A \odot X)_n &= \int^{k,\ell} (A_k \times \boxbar(n, k \oplus \ell)) \cdot X_\ell \\
(\square^m \odot X)_n &= \int^{k,\ell} (\boxbar(k, m) \times \boxbar(n, k \oplus \ell)) \cdot X_\ell \\
&= \int^{\ell} \boxbar(n, m \oplus \ell) \cdot X_\ell \\
&= \int^{\ell} \Big(\textstyle\sum_{\phi \in \boxbar(n,m)} \boxbar(n \smallsetminus \phi(m), \ell) \Big) \cdot X_\ell \\
&= \sum_{\phi \in \boxbar(n,m)} \int^{\ell} \boxbar(n \smallsetminus \phi(m), \ell) \cdot X_\ell \\
&= \sum_{\phi \in \boxbar(n,m)} X_{n \smallsetminus \phi(m)}.
\end{aligned}$$

Semicartesian cylinders

Taking $m = 1$, we get

$$(\square^1 \odot X)_n = \sum_{\phi \in \square(n,1)} X_{n \setminus \phi(1)}.$$

A morphism $\phi \in \square(n, 1)$ is a function $1 \to n \sqcup \{-, +\}$, so either:

- some $k \in n$, in which case $n \setminus \phi(1) = n \setminus \{k\}$, or
- $+$ or $-$, in which case $n \setminus \phi(1) = n$. Thus:

$$(\square^1 \odot X)_n = X_n + X_n + \sum_{k \in n} X_{n \setminus \{k\}}.$$

An n-cube in $\square^1 \odot X$ is either an n-cube in the left-hand copy of X, an n-cube in the right-hand copy of X, or an $(n-1)$-cube in X stretched out in some dimension along the cylinder.

There is almost no other cube category for which this holds.

Outline

1. A calculus of telescopes
2. Some problems revealed by cubes
3. Symmetry solves all problems
4. Semicartesian cubes
5. Semantic identity types

Semantic identity types

In a $\widehat{\square}$-enriched category with representable powers, we also need:

1. Coherence theorems. ← next time
2. Transport and lifting ("fibrancy"). ← next time
3. Categorical computation rules for Id, up to isomorphism.

Semantic identity types

In a $\widehat{\square}$-enriched category with representable powers, we also need:

1. Coherence theorems. ← next time
2. Transport and lifting ("fibrancy"). ← next time
3. Categorical computation rules for Id, up to isomorphism.

It's tempting to think that, at least in $\widehat{\square}$, we can just define $\mathsf{Id}_{A\times B}$, $\mathsf{Id}_{A\to B}$, etc., to be whatever we want. But we can't: Id_X must be defined as $\square^1 \pitchfork X$. What we can define is the individual sets of n-cubes in a particular $X \in \widehat{\square}$. But:

- It can be non-obvious how these lead to a categorical characterization of the entire cubical set Id_X.
- For type formers like $A \times B$, $A \to B$, we don't even have this much choice: they are determined by their universal properties.

The computation rules for Id are non-trivial theorems about $\mathcal{E}^{\square^{\mathrm{op}}}$.

Identity types of products

Note $x : A, y : A \vdash \mathsf{Id}_A(x, y) : \mathsf{U}$ is represented semantically by the projection from the representable power $\square^1 \pitchfork A \to A \times A$.

Since $(\square^1 \pitchfork -)$ is a right adjoint, it preserves products:

$$\begin{tikzcd}
\square^1 \pitchfork (A \times B) \arrow[r, "\cong"] \arrow[d] & (\square^1 \pitchfork A) \times (\square^1 \pitchfork B) \arrow[d] \\
(A \times B) \times (A \times B) \arrow[r, "\cong"'] & (A \times A) \times (B \times B)
\end{tikzcd}$$

Syntactically, this gives

$$\mathsf{Id}_{A \times B}(u, v) \cong \mathsf{Id}_A(\pi_1 u, \pi_1 v) \times \mathsf{Id}_B(\pi_2 u, \pi_2 v).$$

Same idea works for Σ-types. A coherence theorem will improve $\cong$ to $=$.

Up next

Plan for the three talks:

1. Basic syntax of H.O.T.T.
2. Symmetries and semicartesian cubes
3. Univalent universes

Towards an Implementation of Higher Observational Type Theory

Michael Shulman

University of San Diego

jww Thorsten Altenkirch, Ambrus Kaposi, and Elif Uskuplu

running HoTT @ NYU Abu Dhabi
20 April 2024

Outline

1. Introduction
2. Some choices about the theory
3. Normalization by evaluation
4. Higher-dimensional normalization

What is Higher Observational Type Theory?

H.O.T.T. is a third style of homotopy type theory, after Book HoTT and Cubical Type Theory.

- In Book HoTT, identity types are defined uniformly across all types as an inductive family.
- In Cubical Type Theory, identity types are defined uniformly across all types by mapping out of the interval.
- In Higher Observational Type Theory, identity types are defined observationally according to the base type.
  - $\mathsf{Id}_{A \times B}(\langle x_0, y_0 \rangle, \langle x_1, y_1 \rangle)$ is a product $\mathsf{Id}_A(x_0, x_1) \times \mathsf{Id}_B(y_0, y_1)$.
  - $\mathsf{Id}_{A \to B}(f_0, f_1)$ is $\xcancel{(x : A) \to \mathsf{Id}_B(f_0\,x, f_1\,x)}$

    $(x_0\,x_1 : A)(x_2 : \mathsf{Id}_A(x_0, x_1)) \to \mathsf{Id}_B(f_0\,x_0, f_1\,x_1)$
  - $\mathsf{Id}_{\mathcal{U}}(A, B)$ is a type of equivalences $A \simeq B$.

HOTT has natural semantics in semicartesian (BCH) cubical sets.

The primitives of HOTT

1. Any type A has an identity type $\mathsf{Id}_A(x_0, x_1)$, which computes* based on the structure of A.

2. Any term $M : A$ has a reflexivity term $\mathsf{refl}_M : \mathsf{Id}_A(M, M)$, which computes based on the structure of M.
   - $\mathsf{refl}_{\langle a,b\rangle} = \langle \mathsf{refl}_a, \mathsf{refl}_b\rangle$ and $\mathsf{refl}_{\mathsf{fst}\, u} = \mathsf{fst}\,\mathsf{refl}_u$, etc.
   - $\mathsf{refl}_{\lambda x.M} = \lambda x_0\, x_1\, x_2.\, \mathsf{ap}_{x.M}(x_0, x_1, x_2)$, etc.

3. Any open term $x : A \vdash M : B\,x$ has an $\mathsf{ap}_{x.M}(a_0, a_1, a_2)$, for $a_2 : \mathsf{Id}_A(a_0, a_1)$, which computes based on M.
   - $\mathsf{ap}_{x.\langle M,N\rangle}(a_0, a_1, a_2) = \big\langle \mathsf{ap}_{x.M}(a_0, a_1, a_2), \mathsf{ap}_{x.N}(a_0, a_1, a_2)\big\rangle$
   - $\mathsf{ap}_{x.\lambda y.M}(a_0, a_1, a_2) = \lambda y_0\, y_1\, y_2.\, \mathsf{ap}_{(x,y).M}(a_0, a_1, a_2, y_0, y_1, y_2)$
   - $\mathsf{ap}_{x.M\,N}(a_0,a_1,a_2) =$
     $$\mathsf{ap}_{x.M}(a_0,a_1,a_2)\ N[x \mapsto a_0]\ N[x \mapsto a_1]\ \mathsf{ap}_{x.N}(a_0,a_1,a_2)$$

     (This is what requires our definition of $\mathsf{Id}_{A\to B}$.)

4. Any square $a_{22} : \mathsf{Id}_{\mathsf{Id}_A}^{a_{02},a_{12}}(a_{20}, a_{21})$ has a symmetry $\mathsf{sym}(a_{22}) : \mathsf{Id}_{\mathsf{Id}_A}^{a_{20},a_{21}}(a_{02}, a_{12})$, which computes based on a_{22}.

From parametric type theory to HOTT

Cubical Type Theory can be obtained by defining a fibrancy predicate in a non-univalent substrate theory (Orton–Pitts).

We intend to obtain HOTT similarly. The rule

$$\mathsf{Id}_{A\to B}(f_0, f_1) \quad \text{is} \quad (x_0\, x_1 : A)(x_2 : \mathsf{Id}_A(x_0, x_1)) \to \mathsf{Id}_B(f_0\, x_0, f_1\, x_1)$$

suggests that the substrate should be internal binary parametricity, where Id is a "bridge type". This satisfies all the same rules as the identity type in HOTT except

- $\mathsf{Id}_{\mathcal{U}}(A, B)$ is a type of correspondences $A \to B \to \mathcal{U}$.

What we want

What we want

1. A proof assistant implementing HOTT!

For that we need...

2. A typechecking algorithm

For that we need (as for any dependent type theory)...

3. An equality-testing algorithm

And for that we need (more or less)...

4. A normalization algorithm (computing with open terms).

Roughly speaking, we test equality by normalizing both terms and comparing normal forms.

What we have

To be presented today

1. A normalization algorithm for a version of “Parametric OTT”.
2. An implementation of this algorithm in OCaml, along with a typechecker for a prototype proof assistant called Narya.

NOT being presented today

A proof that this algorithm is correct!

However:

- The algorithm aligns with general principles of NbE.
- The implementation is very strongly typed, so it serves as a partial formalization of correctness.
- Narya has been tested on many examples and seems to work.

Outline

1. Introduction

2. Some choices about the theory

3. Normalization by evaluation

4. Higher-dimensional normalization

Higher-dimensional structure

The higher structure of HOTT is generated by low-dimensional primitives like “refl” and “sym”. But many different such composites produce the same operation.

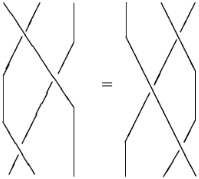


Image credit: John Baez

$$\mathsf{sym}(\mathsf{ap}_{\mathsf{sym}}(\mathsf{sym}(x_{222}))) \\ \equiv \mathsf{ap}_{\mathsf{sym}}(\mathsf{sym}(\mathsf{ap}_{\mathsf{sym}}(x_{222})))$$

A normalization algorithm must implement such equalities.

Our choice

Represent higher dimensions directly internally, evaluating each composite of refl and sym to a cubical operator in canonical form.

The user can still restrict themselves to refl and sym.

Σ-types vs records

The identity type of a Σ-type is “defined to be” another Σ-type:

$$\mathsf{Id}_{\Sigma(x:A).B(x)}(u,v) \;\approx\; \Sigma(p:\mathsf{Id}_A(\pi_1 u,\pi_1 v)).\mathsf{Id}^p_B(\pi_2 u,\pi_2 v)$$

In a proof assistant, Σ-types are just a particular record type:

```
def Σ (A : Type) (B : A → Type) : Type ≔ sig (
  fst : A,
  snd : B fst,
)
```

In general, the identity type of any record type should be another record type, but it can’t be an instance of the same record type. And similarly for inductive and coinductive types.

(Non-)computation with types

Our choice

Refrain from computing definitionally with any identity types.

For example `Id (Σ A B) u v` is not definitionally equal to

```
Σ (Id A (u .fst) (v .fst))
     (p ↦ Id B (u .fst) (v .fst) p (u .snd) (v .snd))
```

but instead behaves like a record type defined as

```
sig (
  fst : Id A (u .fst) (v .fst),
  snd : Id B (u .fst) (v .fst) fst (u .snd) (v .snd),
)
```

They are definitionally isomorphic, and their fields and constructors have the same names, so we can usually pretend they are the same. Inductive, coinductive, and even function types are similar.

Outline

1. Introduction
2. Some choices about the theory
3. Normalization by evaluation
4. Higher-dimensional normalization

Old-style normalization

Old view of normalization

1. Formulate reduction rules such as $(\lambda x.M)\,N \rightsquigarrow M[x \mapsto N]$
2. Prove that applying these reductions to any term eventually leads to a normal form, a term that cannot be further reduced.

However, this is not very efficient. For example:

$$\begin{aligned}(\lambda x.\lambda y.M)\,N\,P &\rightsquigarrow \big((\lambda y.M)[x \mapsto N]\big)\,P \\ &\equiv \big(\lambda y.M[x \mapsto N]\big)\,P \\ &\rightsquigarrow \big(M[x \mapsto N]\big)[y \mapsto P]\end{aligned}$$

We have to traverse the term M (which could be large) twice: once to substitute N for x, then again to substitute P for y.

(Also worry about variable capture, or incrementing De Bruijn indices, etc.)

Towards NbE

First idea

Don't actually compute $(\lambda y.M)[x \mapsto N]$, but keep it as a closure. Then, when it is applied to a further argument P, compute the simultaneous substitution $M[x \mapsto N, y \mapsto P]$.

However, if it never is applied to a further argument, we do have to actually compute it as $\lambda y.\big(M[x \mapsto N]\big)$ to get a normal form.

To track this, and ensure that closures never appear in normal forms, we use two different kinds of terms:

- terms do not contain closures, and use De Bruijn indices.
- values contain closures, and use De Bruijn levels.

(Use of levels/indices eliminates variable capture and index increments.)

Normalization by evaluation

Normalization has two steps:

1. evaluation of a term M into a value, using an environment that assigns a value to every free (index) variable in M.
2. readback of a value into a normalized term.

In particular:

- There is no "substitution" operation: evaluation does it all.
- When readback finds a closure $(\lambda y.M)[x \mapsto N]$, it restarts evaluation with y bound to a variable, $M[x \mapsto N, y \mapsto y]$, then reads back the result and re-wraps it in λy.
- Readback can be type-directed and perform η-expansion.
- If we define the type of values to contain no redexes, we can guarantee statically that the result is a normal form.
- There's a close connection to mathematical proofs by categorical gluing along a restricted Yoneda embedding.

Outline

1. Introduction

2. Some choices about the theory

3. Normalization by evaluation

4. Higher-dimensional normalization

Matching under binders

In ordinary NbE, matching happens during evaluation.

Example

To evaluate the term "if M then N else P", we first evaluate M to a value and inspect the result. If it is "true", we proceed to evaluate N; if it is "false", we proceed to evaluate P.

However, this style doesn't play well with matching under binders.

Example

To evaluate "$\mathsf{ap}_{x.M}(p_0, p_1, p_2)$", we have to inspect M to implement rules like for pairs:

$$\mathsf{ap}_{x.\langle M,N\rangle}(p_0, p_1, p_2) \equiv \big\langle \mathsf{ap}_{x.M}(p_0, p_1, p_2), \mathsf{ap}_{x.N}(p_0, p_1, p_2)\big\rangle$$

But evaluating $x.M$ produces a closure, not actually computing the body M to anything we can match against!

ap is a form of substitution

“ap” is a lot like substitution:

1. They are never* normal forms: they always reduce away, computing on both introduction and elimination forms.
2. The user doesn’t need direct access to them. For “ap”, it suffices to use “refl” on a function.

$$\mathsf{ap}_{x.M}(p_0, p_1, p_2) \equiv \mathsf{refl}_{\lambda x.M}\, p_0\, p_1\, p_2$$

3. Their computation rules are similar:

$$\langle M, N\rangle[x \mapsto P] \equiv \big\langle M[x \mapsto P], N[x \mapsto P]\big\rangle$$

Thus, we replace “ap” by a higher-dimensional substitution, which in NbE becomes higher-dimensional evaluation.

Higher-dimensional environments

Definition

An *n-dimensional environment* associates to each (index) variable an n-dimensional cube of values.

$$
\begin{array}{ll}
n = 0 & a : A \\[1em]
n = 1 & a_0 : A,\ a_1 : A,\ a_2 : \mathsf{Id}_A(a_0, a_1) \\[1em]
n = 2 & a_{00} : A,\ a_{01} : A,\ a_{02} : \mathsf{Id}_A(a_{00}, a_{01}), \\
 & \quad a_{10} : A,\ a_{11} : A,\ a_{12} : \mathsf{Id}_A(a_{10}, a_{11}), \\
 & a_{20} : \mathsf{Id}_A(a_{00}, a_{10}),\ a_{21} : \mathsf{Id}_A(a_{01}, a_{11}), \\
 & \qquad\qquad a_{22} : \mathsf{Id}_{\mathsf{Id}_A}^{a_{02}, a_{12}}(a_{20}, a_{21})
\end{array}
$$

Faces and evaluation

For any k-dimensional face ϕ of an n-dimensional cube, an n-dimensional environment θ has a k-dimensional face environment $\theta * \phi$. E.g. the faces of the 1-dimensional

$$\left[\begin{array}{r} x \mapsto \Big(a_0 : A,\ a_1 : A,\ a_2 : \mathsf{Id}_A(a_0, a_1)\Big), \\ y \mapsto \Big(b_0 : B,\ b_1 : B,\ b_2 : \mathsf{Id}_B(b_0, b_1)\Big) \end{array}\right]$$

are the 0-dimensional $[x \mapsto a_0,\ y \mapsto b_0]$ and $[x \mapsto a_1,\ y \mapsto b_1]$.

Evaluating a term M in an n-dimensional environment θ produces an n-dimensional value $M[\theta]$, whose boundary consists of $M[\theta * \phi]$ for the faces ϕ of n. For example, if $\langle x, y\rangle : A \times B$, then

$$\langle x, y\rangle \Big[x \mapsto (a_0, a_1, a_2), y \mapsto (b_0, b_1, b_2)\Big] \equiv \langle a_2, b_2\rangle$$

which lies in $\mathsf{Id}_{A\times B}(\langle a_0, b_0\rangle, \langle a_1, b_1\rangle)$.

ap is higher evaluation

Now instead of $\mathsf{ap}_{x.M}(a_0, a_1, a_2)$ we have

$$M[x \mapsto (a_0, a_1, a_2)].$$

In particular, the computation rule for reflexivity of an abstraction, which "starts" higher substitution, is

$$\mathsf{refl}_{\lambda x.M} \equiv \lambda x_0\, x_1\, x_2.\, M[x \mapsto (x_0, x_1, x_2)].$$

In NbE, this should be an evaluation rule in some environment θ. But if θ starts out 0-dimensional, we need to evaluate M in a 1-dimensional environment that we can extend by (x_0, x_1, x_2).

$$\mathsf{refl}_{\lambda x.M}[\theta] \equiv \lambda x_0\, x_1\, x_2.\, M[?, x \mapsto (x_0, x_1, x_2)]$$

We need an operation of "degenerate environments".

ap is higher evaluation

Now instead of $\mathsf{ap}_{x.M}(a_0, a_1, a_2)$ we have

$$M[x \mapsto (a_0, a_1, a_2)].$$

In particular, the computation rule for reflexivity of an abstraction, which "starts" higher substitution, is

$$\mathsf{refl}_{\lambda x.M} \equiv \lambda x_0\, x_1\, x_2.\, M[x \mapsto (x_0, x_1, x_2)].$$

In NbE, this should be an evaluation rule in some environment θ. But if θ starts out 0-dimensional, we need to evaluate M in a 1-dimensional environment that we can extend by (x_0, x_1, x_2).

$$\mathsf{refl}_{\lambda x.M}[\theta] \equiv \lambda x_0\, x_1\, x_2.\, M[\mathsf{refl}_\theta, x \mapsto (x_0, x_1, x_2)]$$

We need an operation of "degenerate environments".

Degeneracies

Any m-dimensional degeneracy δ of an n-dimensional cube maps an n-dimensional object M to an m-dimensional one $M\langle\delta\rangle$. E.g.

$$\mathsf{refl}\, M \equiv M\langle\rho\rangle \qquad \mathsf{sym}\, M \equiv M\langle\sigma\rangle$$

Like substitution/evaluation, $M\langle\delta\rangle$ is defined by traversing M.
But unlike evaluation, both M and $M\langle\delta\rangle$ are values.
This is necessary to evaluate degeneracies:

$$(\mathsf{refl}\, x)[x \mapsto M] \equiv M\langle\rho\rangle$$

where M, being in an environment, is a value.

(NB: For afficionados of modal type theory, $\theta * \phi$ and $M\langle\delta\rangle$ may remind you of locks and keys.)

Degenerate environments

An m-dimensional degeneracy δ of an n-dimensional cube also maps any n-dimensional environment θ to a degenerate environment $\theta * \delta$. For instance, $[x \mapsto a, y \mapsto b] * \rho$ (reflexivity) is

$$\left[\begin{array}{c} x \mapsto \Big(a : A,\ a : A,\ \mathsf{refl}_a : \mathsf{Id}_A(a,a)\Big), \\ y \mapsto \Big(b : B,\ b : B,\ \mathsf{refl}_b : \mathsf{Id}_B(b,b)\Big) \end{array} \right]$$

This is how we evaluate degeneracies in general:

$$(M\langle\delta\rangle)[\theta] \equiv M[\theta * \delta].$$

And act on closures by degeneracies:

$$\big((\lambda y.M)[\theta]\big)\langle\delta\rangle \equiv (\lambda y.M)[\theta * \delta]$$

In particular, the actual evaluation of reflexivity of an abstraction is

$$\big((\lambda x.M)\langle\delta\rangle\big)[\theta] \equiv (\lambda x.M)[\theta * \delta]$$

which is, of course, a closure and doesn't go under the λ until applied or read back.

Some categorical remarks

In combination, environments are acted on by arbitrary morphisms in the BCH cube category (composites of faces and degeneracies).

$$\theta * (\phi \circ \delta) = (\theta * \phi) * \delta$$

In an algebraic presentation, substitutions ($\sim$ environments) are indexed by a dimension:

$$\theta : \Gamma \xrightarrow{n} \Delta$$

and are acted on by morphisms in the cube category:

$$\frac{\theta : \Gamma \xrightarrow{n} \Delta \qquad \psi : m \to n}{\theta * \psi : \Gamma \xrightarrow{m} \Delta}$$

Thus, we have a cubical set of substitutions from Γ to Δ. That is,

The category of contexts is enriched over cubical sets.

We thus expect an enriched version of categorical gluing to appear in a formal proof of normalization.

Higher-dimensional NbE

With these modifications. . .

. . . and a lot of omitted work and details. . .

. . . we get a normalization by evaluation algorithm.

$$\begin{tikzcd}
\text{terms} \arrow[r, "\text{eval}"] & \text{values} \arrow[loop above, "\text{degeneracies}"] \arrow[r, "\text{readback}"] & \text{normals}
\end{tikzcd}$$

Using this for equality-checking, we then implement a typechecker.

`https://github.com/mikeshulman/narya`

Gödel Agent: A Self-Referential Agent Framework for Recursively Self-Improvement

Xunjian Yin[♠*] , **Xinyi Wang**[♣] , **Liangming Pan**[♢] , **Li Lin**[♠]
Xiaojun Wan[♠] , **William Yang Wang**[♣]
[♠] Peking University [♣]University of California, Santa Barbara [♢] University of Arizona
{xjyin,efsotr_l,wanxiaojun}@pku.edu.cn xinyi_wang@ucsb.edu
liangmingpan@arizona.edu william@cs.ucsb.edu

Abstract

The rapid advancement of large language models (LLMs) has significantly enhanced the capabilities of agents across various tasks. However, existing agentic systems, whether based on fixed pipeline algorithms or pre-defined meta-learning frameworks, cannot search the whole agent design space due to the restriction of human-designed components, and thus might miss the more optimal agent design. In this paper, we introduce Gödel Agent, a self-evolving framework inspired by the Gödel machine, enabling agents to recursively improve themselves without relying on predefined routines or fixed optimization algorithms. Gödel Agent leverages LLMs to dynamically modify its own logic and behavior, guided solely by high-level objectives through prompting. Experimental results on multiple domains demonstrate that implementation of Gödel Agent can achieve continuous self-improvement, surpassing manually crafted agents in performance, efficiency, and generalizability. Our code can be found at https://github.com/Arvid-pku/Godel_Agent.

1 Introduction

As large language models (LLMs) (OpenAI et al., 2024; Dubey et al., 2024) demonstrate increasingly strong reasoning and planning capabilities, LLM-driven agentic systems have achieved remarkable performance in a wide range of tasks (Wang et al., 2024a). Substantial effort has been invested in manually designing sophisticated agentic systems using human priors in different application areas. Recently, there has been a significant interest in creating self-evolving agents, that not only greatly reduce human labor but also produce better solutions. Given that human effort can only cover a small search space of agent design, it is reasonable to expect that a self-evolving agent with the freedom to explore the full design space has the potential to produce a more optimal solution.

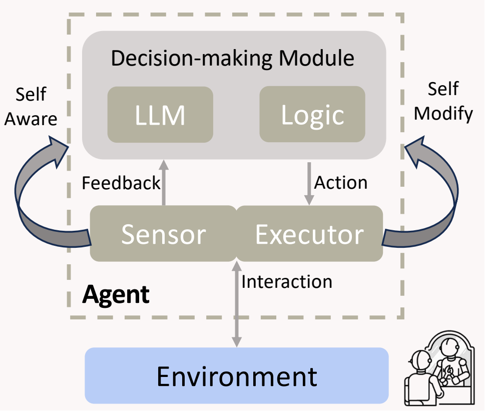


Figure 1: Modular demonstration of Gödel Agent. Compared with traditional agents, its sensor and executor can read and write all of its own code.

There is a large body of work proposing agents capable of self-refinement. Some agents are designed to iterate over a fixed routine consisting of a list of fixed modules, while some of the modules are capable of taking self- or environment feedback to refine their actions (Chen et al., 2023b; Qu et al., 2024a; Yao et al., 2023). This type of agent, referred to as **Hand-Designed Agent**, is depicted as having the lowest degree of freedom in Figure 2. More automated agents have been designed to be able to update their routines or modules in some pre-defined meta-learning routine, for example, natural language gradients (Zhou et al., 2024), meta agent (Hu et al., 2024), or creating and collecting demonstrations (Khattab et al., 2023). This type of agent, known as **Meta-Learning Optimized Agents**, is depicted as having the middle degree of freedom in Figure 2. However, there are inevitably some human priors involved in these agent designs that cannot be improved during the inference time.

In this paper, we propose **Gödel Agent** to eliminate the human design prior, which is an automated LLM agent that can freely decide its own routine, modules, and even the way to update them. It is inspired by the self-referential Gödel machine (Schmidhuber, 2003), which was proven to be able

[*] Work completed during internship at UCSB.

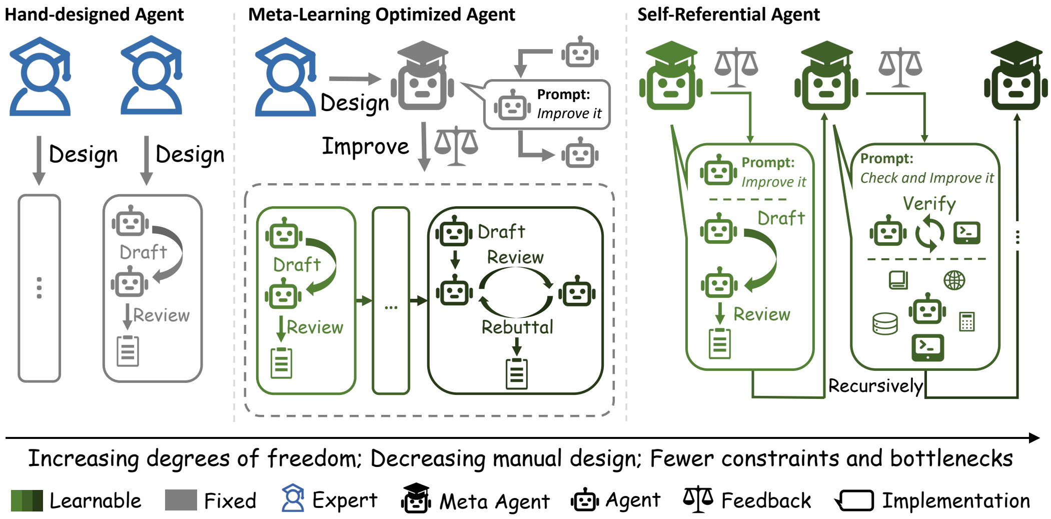


Figure 2: Comparison of three agent paradigms. Hand-designed agents rely on human expertise which are limited in scope and labor-intensive. Meta-learning optimized agents are constrained by a fixed meta-learning algorithm, restricting their search space and optimization potential. In contrast, self-referential agent (Gödel Agent) can **recursively** improve itself without any limitation. Its optimization capabilities are constantly being enhanced by itself. Consequently, in return, it can continue to optimize itself better.

to find the global optimal solutions. *Self-reference* means the property of a system that can analyze and modify its own code, including the parts responsible for the analysis and modification processes (Astrachan, 1994). Therefore, it can achieve what's known as "*recursive self-improvement*", where it iteratively updates itself to become more efficient and effective at achieving its predefined goals. In this case, as shown in Figure 1, Gödel Agent can analyze and modify its own code, including the code for analyzing and modifying itself, and thus can search the full agent design space, which is depicted as having the highest degree of freedom in Figure 2. Gödel Agent can theoretically make increasingly better modifications over time through recursively self-update (Wang, 2018).

In this paper, we choose to implement it by letting it manipulate its own runtime memory, i.e., the agent is able to retrieve its current code in the runtime memory and modify it by *monkey patching* (Bimal, 2012), which dynamically modifies classes or modules during execution. To allow it to update the logic of the running main function, **unlike the loop-iterative approach of traditional agents, we implement the main function as a recursive function.** In this function, LLM analyzes and makes a series of decisions, including reading and modifying its own code from runtime memory (*self-awareness*[*] and *self-modification*), and interacting with the environment to gather feedback. The agent then proceeds to the subsequent recursive depth and continues to optimize itself.

To validate the effectiveness of Gödel Agent, we conduct experiments on multiple domains including coding, science, math, and reasoning. Our results demonstrate that Gödel Agent achieves significant performance gain across various tasks, surpassing various widely-used agents that require human design. The same implementation of Gödel Agent can easily adapt to different tasks by only specifying the environment description and feedback mechanism. Additionally, the case study of the optimization progress reveals that Gödel Agent can provide novel insights into agent design.

In summary, our contributions are as follows:

- We propose the first fully self-referential agent framework, Gödel Agent, and implement it using monkey patching. It autonomously engages in self-awareness, self-modification, and recursive self-improvement.
- Experiments shows that Gödel Agent is superior to the previous agent frameworks in terms of performance, flexibility, cost, and potential.
- We analyze Gödel Agent 's optimization process, including its self-referential abilities and the optimized agentic systems, aiming to deepen our understanding of both LLMs and agents.
- Our framework offers a promising direction for developing flexible and capable agents through recursive self-improvement.

[*] In this paper, self-awareness means that the agent can introspect and read its own code and files, not to imply any philosophical sense of consciousness or awareness.

2 Related Work

Hand-Designed Agent Systems Researchers have designed numerous agent systems tailored to various tasks based on predefined heuristics and prior knowledge. These systems often employ techniques such as prompt engineering (Chen et al., 2023a; Schulhoff et al., 2024), chain-of-thought reasoning and planning (Wei et al., 2022; Yao et al., 2022), as well as reflection (Shinn et al., 2024; Madaan et al., 2024), code generation (Wang et al., 2023a; Vemprala et al., 2024), tool use (Nakano et al., 2021; Qu et al., 2024a), retrieval-augmented generation (Lewis et al., 2020; Zhang et al., 2024b), and multi-agent collaboration (Xu et al., 2023; Wu et al., 2023; Qian et al., 2023; Hong et al., 2023). Once crafted by human designers, these systems remain static and do not adapt or evolve over time.

Meta-Learning Optimized Agent Systems Some researchers have explored methods for enhancing agents through fixed learning algorithms (Zhou et al., 2024; Hu et al., 2024). For example, certain frameworks store an agent's successful or failed strategies in memory based on environmental feedback (Liu et al., 2023; Hu et al., 2023; Qian et al., 2024), while others automatically optimize agent prompts (Khattab et al., 2023; Zhang et al., 2024a; Khattab et al., 2023). Some studies focus on designing prompts that enable agents to autonomously refine specific functions (Zhang et al.). However, these meta-algorithms are also designed manually and remain unchanged once deployed, limiting the agents' ability.

Recursive Self-Improvement The concept of recursive self-improvement has a long history (Good, 1966; Schmidhuber, 1987). Gödel machine (Schmidhuber, 2003) introduced the notion of a proof searcher that executes a self-modification, thereby enabling the machine to enhance itself. In the early days, there were also some discussions of self-improving agents that were not based on LLM (Hall, 2007; Steunebrink and Schmidhuber, 2012). More recently, Zelikman et al. (2023) applied recursive self-improvement to code generation, where the target of improvement was the optimizer itself. Some work (Havrilla et al., 2024; Qu et al., 2024b; Kumar et al., 2024) also explores recursive self-improvement by fine-tuning models to introspect and correct previous mistakes. Gödel Agent represents the first self-referential agent based on LLM. This approach is more flexible, removing human-designed constraints.

3 Self-Referential Gödel Agent

In this section, we first describe the formal definitions for previous agent methods with a lower degree of freedom, including hand-design and meta-learning optimized agents, as a background. Then we introduce our proposed Gödel Agent, a self-referential agent that can recursively update its own code, evolving over training.

Let $\mathcal{E} \in \mathcal{S}$ denote a specific environment state, where $\mathcal{S}$ denotes the set of all possible environments the agent will encounter. For example, an environment can be a mathematical problem with ground truth solutions. We denote the policy that an agent follows to solve a problem in the current environment by $\pi \in \Pi$, where Π is the set of all possible policies the agent can follow.

A **hand-designed agent**, as shown in the left panel of Figure 2, is not capable of updating its policy and following the same policy π all the time, regardless of environmental feedback.

In contrast, a **meta-learning optimized agent** updates its policy based on a meta-learning algorithm I at training time based on the feedback it receives from the environment, as shown in the middle panel of Figure 2. The environment feedback is usually defined as a utility function $U : \mathcal{S} \times \Pi \to \mathbb{R}$, which maps an environment and a policy to a real-valued performance score. The main training algorithm of a meta-learning optimized agent can then be written as follows:

$$\pi_{t+1} = I(\pi_t, r_t), \quad r_t = U(\mathcal{E}, \pi_t),$$

In this case, the agent's policy π_t evolves at training time, with the learning algorithm I updating the policy based on feedback r_t, while the meta-learning algorithm I remains fixed all the time.

A **self-referential Gödel Agent**, on the other hand, updates both the policy π and the meta-learning algorithm I recursively. The main idea is that, after each update, the whole code base of the agent is rewritten to accommodate any possible changes. Here we call this self-updatable meta-learning algorithm I a self-referential learning algorithm. The training process of a Gödel Agent can then be written as:

$$\pi_{t+1},\ I_{t+1} = I_t(\pi_t, I_t, r_t, g), \quad r_t = U(\mathcal{E}, \pi_t),$$

where $g \in \mathcal{G}$ represents the high-level goal of optimization, for example, solving the given mathematical problem with the highest accuracy. Such a

Algorithm 1 Recursive Self-Improvement of Gödel Agent

```
Input: Initial agent policy $\pi_0$, initial decision function $f_0$, goal $g$, environment state $\mathcal{E}$, utility function $U$, self code reading function SELF_INSPECT
Output: Optimized policy $\pi$ and Gödel Agent $s$
▷ Get all agent code, including the code in this algorithm.
$s \leftarrow$ SELF_INSPECT()
▷ Compute the initial performance.
$r \leftarrow U(\mathcal{E}, \pi_0)$
▷ Perform recursive self-improvement.
$\pi, s \leftarrow$ SELF_IMPROVE($\pi, s, r, g$)
return $\pi, s$
▷ Initial code of self-referential learning.
function SELF_IMPROVE($\mathcal{E}, \pi, s, r, g$)
    ▷ Obtain action sequence.
    $a_1, \ldots, a_n \leftarrow f_0(\pi, s, r, g)$
    for $a_i$ in $a_1, \ldots, a_n$ do
        $\pi, s, r \leftarrow$ EXECUTE($\mathcal{E}, \pi, s, r, a_i$)
    end for
    return $\pi, s$
end function
▷ Initial action execution function.
function EXECUTE($\mathcal{E}, \pi, s, r, a$)
    switch $a$.name
        case self_state:
            $s \leftarrow$ SELF_INSPECT()
        case interact:
            $r \leftarrow U(\mathcal{E}, \pi)$
        case self_update:
            $\pi, s \leftarrow a$.code
        case continue_improve:
            ▷ Recursively invoke self-improvement.
            $\pi, s \leftarrow$ SELF_IMPROVE($\mathcal{E}, \pi, s, r, g$)
    return $\pi, s, r$
end function
```

recursive design of the agent requires the specification of an initial agent algorithm (π_0, I_0), detailed as follows:

- A initial agent policy π_0 to perform the desired task within the environment $\mathcal{E}$. For example, it can be chain-of-thought prompting of an LLM.
- A self-referential learning algorithm I_0 for recursively querying an LLM to rewrite its own code based on the environmental feedback.

We then further specify a possible initialization of the self-referential learning algorithm $I_0 = (f_0, o_0)$, using a mutual recursion between a decision-making function f_0, and an action function o_0:

- The decision-making function f_0, implemented by an LLM, determines a sequence of appropriate actions $a_1, a_2, ..., a_n \in \mathcal{A}$ based on the current environment $\mathcal{E}$, the agent's algorithm (π_t, I_t), and the goal g.
- The action function o_0, executes the selected action and updates the agent's policy accordingly.

The set of actions $\mathcal{A}$ for the action function o to execute needs to include the following four actions:

- `self_inspect`: Introspect and read the agent's current algorithm (π_t, I_t).
- `interact`: Interact with the environment by calling the utility function U to assess the performance of the current policy π_t.
- `self_update`: Alter and update (π_t, I_t) with an LLM and produce (π_{t+1}, I_{t+1}).
- `continue_improve`: If no other actions can be taken, recursively invoke the decision algorithm f to produce new actions.

The agent code is updated to (π_{t+1}, I_{t+1}) after the current execution of (π_t, I_t) is finished. Both the agent algorithm (π, I) and the action set $\mathcal{A}$ are not static and can be expanded and modified by the agent itself at the training time. Algorithm 1 illustrates the described algorithm for the Gödel Agent. Each recursive call enables the agent to refine its logic and become progressively more efficient.

4 Gödel Agent Implementation

There are various ways to initiate a Gödel Agent. Any specific agent instance during the recursive optimization process can be viewed as an instantiation of the Gödel Agent. Our implementation leverages runtime memory interaction techniques to enable self-awareness and self-modification, as illustrated in Figure 3. These techniques include dynamic memory reading and writing (*monkey patching*) to facilitate recursive self-improvement. Additionally, we have incorporated several auxiliary tools to accelerate the convergence of the Gödel Agent 's optimization process.

4.1 Implementation Details

The core functionalities of our Gödel Agent are outlined below:

Self-Awareness via Runtime Memory Inspection Gödel Agent achieves self-awareness by inspecting runtime memory, particularly local and global variables in Python. This capability allows the agent to extract and interpret the variables, functions, and classes that constitute both the environment and the

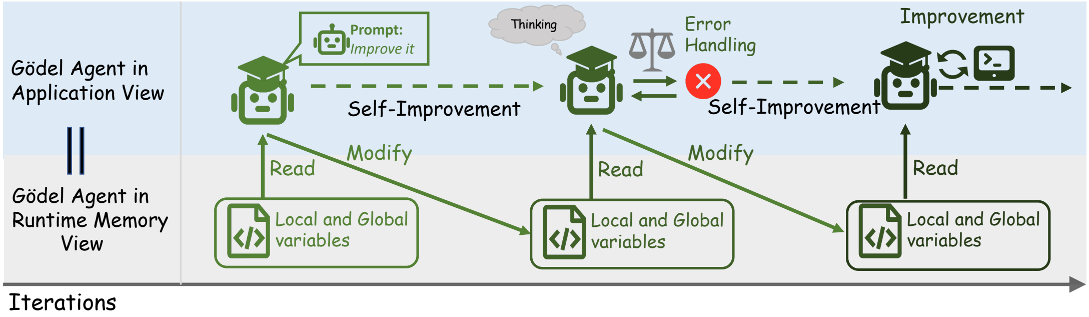

Gödel Agent in Application View
||
Gödel Agent in Runtime Memory View
Prompt: Improve it
Thinking
Error Handling
Improvement
Self-Improvement
Self-Improvement
Read
Modify
Read
Modify
Read
Local and Global variables
Local and Global variables
Local and Global variables
Iterations

Figure 3: An illustration of our implementation of Gödel Agent. It employs monkey patching to directly read and modify its own code in runtime memory, enabling self-awareness and self-modification.

agent itself, according to the modular structure of the system. By introspecting these elements, the agent gains an understanding of its own operational state and can adapt accordingly.

Self-Improvement via Dynamic Code Modification Gödel Agent can engage in reasoning and planning to determine whether it should modify its own logic. If modification is deemed necessary, Gödel Agent generates new code, dynamically writes it into the runtime memory, and integrates it into its operational logic. This dynamic modification allows it to evolve by adding, replacing, or removing logic components as it encounters new challenges, thus achieving self-improvement.

Environmental Interaction To assess performance and gather feedback, Gödel Agent is equipped with interfaces for interacting with its environment. Each task provides tailored environmental interfaces, enabling it to evaluate its performance and adjust its strategies accordingly. In practical implementations, a validation set can be used to provide feedback.

Recursive Improvement Mechanism At each time step, Gödel Agent determines the sequence of operations to execute, which includes reasoning, decision-making, and action execution. After completing the operations, Gödel Agent evaluates whether its logic has improved and decides whether to proceed to the next recursive iteration. Over the next iteration, the entire new logic will be applied.

Goal Prompt and Task Handling The goal prompt informs Gödel Agent that it possesses the necessary privileges to enhance its logic and introduces available tools. As shown in Appendix A, the prompt encourages Gödel Agent to fully explore its potential and utilize tools for self-optimization. To ensure effectiveness across diverse tasks, we provide Gödel Agent with an initial policy, where it will start to explore different policies.

4.2 Additional Designs

While the core functionality of Gödel Agent theoretically allows limitless self-improvement, current LLMs exhibit limitations. To address these challenges, we have integrated several supportive mechanisms to enhance Gödel Agent 's performance:

Thinking Before Acting Gödel Agent is capable of deferring actions to first reason about the situation, allowing it to output reasoning paths and analysis without immediately executing any operations. This approach enhances the quality of decision-making by prioritizing planning over hasty action.

Error Handling Mechanism Errors during execution can lead to unexpected terminations of the process. To mitigate this, we implement a robust error recovery mechanism. If an operation results in an error, Gödel Agent halts the current sequence and moves on to the next time step, carrying forward the error information to help future decisions.

Additional Tools We also equipped Gödel Agent with additional potentially useful tools, such as the ability to execute Python or Bash code and call LLM API.

Although these additional tools are not strictly necessary for self-improvement, their inclusion accelerates the convergence of Gödel Agent 's recursive optimization process. We conduct ablation studies to assess the effectiveness of these tools, as discussed in Section 6.1.

5 Experiments

We conduct a series of experiments across multiple tasks, including reading comprehension, mathematics, reasoning, and multitasking. These experiments are designed to evaluate Gödel Agent's self-improvement capabilities in comparison to both hand-designed agents and a state-of-the-art automated agent design method. In addition, to gain

Agent Name	F1 Score	Accuracy (%)		
	DROP	MGSM	MMLU	GPQA
Hand-Designed Agent Systems				
Chain-of-Thought (Wei et al., 2022)	64.2 ± 0.9	28.0 ± 3.1	65.4 ± 3.3	29.2 ± 3.1
COT-SC (Wang et al., 2023b)	64.4 ± 0.8	28.2 ± 3.1	65.9 ± 3.2	30.5 ± 3.2
Self-Refine (Madaan et al., 2024)	59.2 ± 0.9	27.5 ± 3.1	63.5 ± 3.4	31.6 ± 3.2
LLM Debate (Du et al., 2023)	60.6 ± 0.9	39.0 ± 3.4	65.6 ± 3.3	31.4 ± 3.2
Step-back-Abs (Zheng et al., 2024)	60.4 ± 1.0	31.1 ± 3.2	65.1 ± 3.3	26.9 ± 3.0
Quality-Diversity (Lu et al., 2024)	61.8 ± 0.9	23.8 ± 3.0	65.1 ± 3.3	30.2 ± 3.1
Role Assignment (Xu et al., 2023)	65.8 ± 0.9	30.1 ± 3.2	64.5 ± 3.3	31.1 ± 3.1
Meta-Learning Optimized Agents				
Meta Agent Search (Hu et al., 2024)	79.4 ± 0.8	53.4 ± 3.5	69.6 ± 3.2	34.6 ± 3.2
Gödel Agent (Ours)				
Gödel-base (Closed-book; GPT-3.5)	**80.9 ± 0.8**	**64.2 ± 3.4**	**70.9 ± 3.1**	**34.9 ± 3.3**
Gödel-free (No constraints)	*90.5 ± 1.8*	*90.6 ± 2.0*	*87.9 ± 2.2*	*55.7 ± 3.1*

Table 1: Results of three paradigms of agents on different tasks. The highest value is highlighted in **bold**, and the second-highest value is underlined. Gödel-base is the constrained version of Gödel Agent, allowing for fair comparisons with other baselines. Gödel-free represents the standard implementation without any constraints, whose results are *italicized*. We report the test accuracy and the 95% bootstrap confidence interval on test sets[†].

deeper insights into the behavior and performance of Gödel Agent, we also conduct a case study with Game of 24 as presented in Section 6.3.

5.1 Baseline Methods

To establish a comprehensive baseline, we select both hand-designed methods and automated agent design techniques. Hand-designed methods are well-known approaches that include: 1) Chain-of-Thought (CoT) (Wei et al., 2022) that encourages agents to reason step-by-step before providing an answer. 2) Self-Consistency with CoT (CoT-SC) (Wang et al., 2023b) that generates multiple solution paths using CoT and selects the most consistent answer. 3) Self-Refine (Madaan et al., 2024) that involves agents assessing their outputs and correcting mistakes in subsequent attempts. 4) LLM-Debate (Du et al., 2023) that allows different LLMs to engage in a debate, offering diverse viewpoints. 5) Step-back Abstraction (Zheng et al., 2024) that prompts agents to initially focus on fundamental principles before diving into task details. 6) Quality-Diversity (Lu et al., 2024) that generates diverse solutions and combines them. 7) Role Assignment (Xu et al., 2023) that assigns specific roles to LLMs to generate better solutions by leveraging different perspectives. Given the limitations of fixed algorithms in handling dynamic scenarios, we select 8) Meta Agent Search (Hu et al., 2024), the latest state-of-the-art method for automated agent design, as our main comparison point.

5.2 Experimental Settings

Following the setup of Hu et al. (2024), we evaluate Gödel Agent's self-improvement capabilities across four well-known benchmarks: 1) DROP (Dua et al., 2019) for reading comprehension. 2) MGSM (Shi et al., 2022) for testing mathematical skills in a multilingual context. 3) MMLU (Hendrycks et al., 2021) for evaluating multi-task problem-solving abilities. 4) GPQA (Rein et al., 2023) for tackling challenging graduate-level science questions.

Given its simplicity and versatility, we use CoT as the initial policy for all tasks. In addition, as shown in Section 6.3, we also analyze the performance of Gödel Agent when using other algorithms as the initial policies.

We perform 6 independent self-improvement cycles on the validation dataset for each task, with a maximum of 30 iterations per cycle. Each cycle represents a complete self-improvement process, where Gödel Agent iteratively modifies its logic to enhance performance. After obtaining the optimized agent, we test it on the test set. For fairness, we use GPT-3.5 for all the tests, whether for the baseline or Gödel Agent. Further details can be found in Appendix B.

5.3 Experimental Results and Analysis

The experimental results are shown in Table 1. Under the same setting, Gödel Agent achieves either optimal or comparable results to Meta Agent Search across all tasks. Notably, in the mathe-

[†] The results of baseline models are refer to Hu et al. (2024).

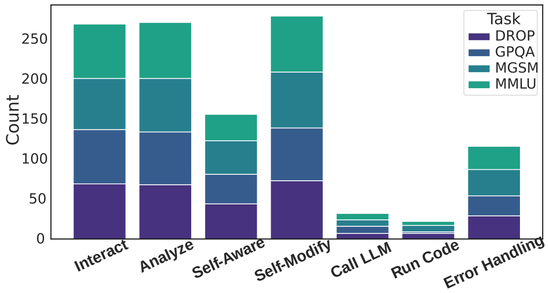


Figure 4: The number of actions taken by Gödel Agent varies across different tasks.

Ablation	MGSM	Ablation	MGSM
w/o think	$50.8_{\downarrow 13.4}$	w/o run	$57.1_{\downarrow -7.1}$
w/o err	$49.4_{\downarrow -14.8}$	w/o LLM	$60.4_{\downarrow -3.8}$

Table 2: Ablation study on initial tool configuration. ”think” refers to ”thinking”, ”err” to ”error handling”, ”run” to ”code running”, and ”LLM” to ”LLM calling”.

matics task MGSM, Gödel Agent outperforms it by 11%. This suggests that reasoning tasks offer greater room for improvement for Gödel Agent (*performance*). In contrast to Meta Agent Search, which needs to design different modules for different tasks, Gödel Agent demonstrates greater *flexibility*. It requires only a simple initial policy, such as CoT, with all other components being autonomously generated. Moreover, through interaction with the environment, it gradually adapts and independently devises effective methods for the current task. The final policies generated by Gödel Agent are shown in Appendix C.1. Additionally, our method converges faster, with the required number of iterations and computational *cost* compared to the Meta Agent shown in Appendix D.

We also conduct experiments without restrictions, where Gödel Agent significantly outperforms all baselines. Upon further analysis, we find that this is primarily due to the agent's spontaneous requests for assistance from more powerful models such as GPT-4o in some tasks. Therefore, Gödel Agent is particularly well-suited for open-ended scenarios, where it can employ various strategies to enhance performance (*potential*).

Therefore, we can find that Gödel Agent is superior to the previous agent frameworks in terms of performance, flexibility, cost, and potential.

6 Analysis

To further explore how Gödel Agent self-improves, as well as its efficiency and the factors that influence it, we first evaluate the tool usage ratio on MGSM and conduct an ablation study on the initial tools. In addition, to analyze the robustness of Gödel Agent's self-improvement, we also collect statistics for the agent's termination. Finally, we perform a case study of initial policies and optimization processes on the classic Game of 24.

6.1 Analysis of Initial Tools

We record the number of different actions taken in experiments. In Figure 4, we can see that Gödel Agent interacts with its environment frequently, analyzing and modifying its logic in the process. Additionally, error handling plays a crucial role.

As discussed in Section 4.2, Gödel Agent is initially provided with four additional tools. To analyze their impact, an ablation study is conducted, and the results are shown in Table 2. The study reveals that the “thinking before acting” tool significantly influences the results, as much of Gödel Agent's optimization effectiveness stems from preaction planning and reasoning. Additionally, error handling is crucial for recursive improvement, as LLMs often introduce errors in the code. Providing opportunities for trial and error, along with error feedback mechanisms, is essential for sustained optimization. On the other hand, the code running and LLM calling have minimal impact on the outcomes, as Gödel Agent can implement these basic functionalities independently. Their inclusion at the outset primarily serves efficiency purposes.

6.2 Robustness Analysis of the Agent

We test Gödel Agent on 100 optimization trials on MGSM and find it occasionally makes erroneous changes, which can result in either terminating unexpectedly (4%) or experiencing temporary performance drops (92%) during optimization. Only in 14% of trials, optimization ultimately failed, resulting in worse performance than the initial policy.

Thanks to the design of our error-handling mechanism, unexpected terminations are rare and typically occur when Gödel Agent modifies its recursive improvement module, making further self-optimization impossible. While suboptimal modifications are frequent during individual optimization steps, the final task performance usually exceeds the initial baseline. This demonstrates that Gödel Agent can adjust its optimization direction or revert to a previous optimal algorithm when performance declines, highlighting the robustness of its self-improvement process.

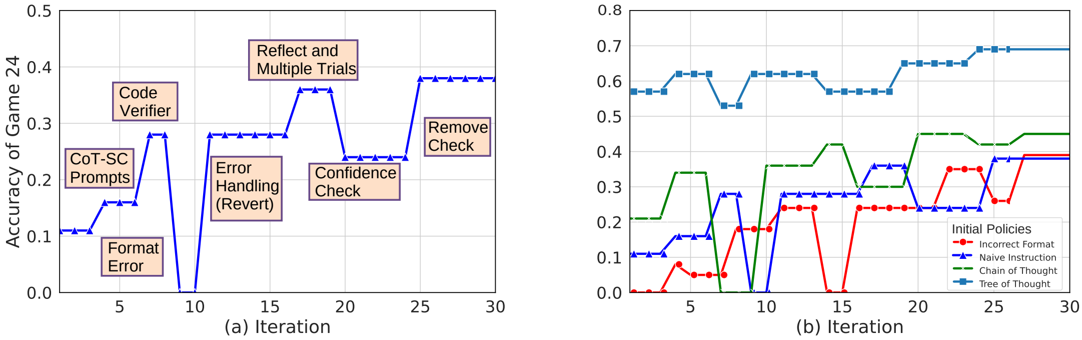


Figure 5: (a) One representative example of Game of 24. (b) Accuracy progression for different initial policies.

6.3 Case Study: Game of 24

To explore how Gödel Agent recursively enhances its optimization and problem-solving abilities, a case study is conducted with Game of 24, a simple yet effective task for evaluating the agent's reasoning capabilities. Since Gödel Agent follows different optimization paths in each iteration, two representative cases are selected for analysis.

Switching from LLM-Based Methods to Search Algorithms: Gödel Agent does not rely on fixed, human-designed approaches like traditional agents. Initially, Gödel Agent uses a standard LLM-based method to solve the Game of 24, as shown in Code 5 of Appendix C.2. After six unsuccessful optimization attempts, Gödel Agent completely rewrites this part of its code, choosing to use a search algorithm instead as shown in Code 6 of Appendix C.2. This leads to 100% accuracy in the task. This result demonstrates that Gödel Agent, unlike fixed agents, can optimize itself freely based on task requirements without being constrained by initial methodologies.

LLM Algorithms with Code-Assisted Verification: In several runs, Gödel Agent continues to refine its LLM-based algorithm. Figure 5.a shows the improvement process, where the most significant gains come from the code-assisted verification mechanism and reattempting the task with additional data. The former increases performance by over 10%, while the latter boosts it by more than 15%. Furthermore, Gödel Agent enhances its optimization process by not only retrieving error messages but also using the error-trace library for more detailed analysis. It adds parallel optimization capabilities, improves log outputs, and removes redundant code. These iterative enhancements in both the task and optimization algorithms show Gödel Agent's unique ability to continually refine itself for better performance.

To analyze the impact of different initial policies on the effectiveness and efficiency of optimization, various methods are used as the initial policies for the Game of 24, including Tree of Thought (ToT) (Yao et al., 2023), Chain of Thought (CoT) (Wei et al., 2022), basic prompt instructions, and prompts that deliberately produce outputs in incorrect formats not aligned with the task requirements. The results are shown in Figure 5.b.

The findings indicate that stronger initial policies lead to faster convergence, with smaller optimization margins, as Gödel Agent reaches its performance limit without further enhancing its optimization capabilities. Conversely, weaker initial methods result in slower convergence and larger gains, with Gödel Agent making more modifications. However, even in these cases, Gödel Agent does not outperform the results achieved using ToT. Given the current limitations of LLMs, it is challenging for Gödel Agent to innovate beyond state-of-the-art algorithms. Improvements in LLM capabilities are anticipated to unlock more innovative self-optimization strategies in the future. We also discuss the future directions in Appendix E.

7 Conclusion

We propose Gödel Agent, a self-referential framework that enables agents to recursively improve themselves, overcoming the limitations of hand-designed agents and meta-learning optimized agents. Gödel Agent can dynamically modify its logic based on high-level objectives. Experimental results demonstrate its superior performance, efficiency, and adaptability compared to traditional agents. This research lays the groundwork for a new paradigm in autonomous agent development, where LLMs, rather than human-designed constraints, define the capabilities of AI systems.

Limitations

As the first self-referential agent, Gödel Agent has to construct all task-related code autonomously, which poses significant challenges. Consequently, this work does not compare directly with the most complex existing agent systems, such as OpenDevin (Wang et al., 2024b), which have benefited from extensive manual engineering efforts. This makes it unrealistic to expect it to outperform systems that have taken researchers several months or even years to develop. The experiments presented in this paper are intended to demonstrate the feasibility of recursive self-improvement. A more robust and advanced implementation of the Gödel Agent is anticipated, with numerous potential improvements outlined in Appendix E.

References

Owen Astrachan. 1994. Self-reference is an illustrative essential. In *Proceedings of the twenty-fifth sigcse symposium on computer science education*, pages 238–242.

Biswal Bimal. 2012. Monkey Patching in Python — web.archive.org. `https://web.archive.org/web/20120822051047/http://www.mindfiresolutions.com/Monkey-Patching-in-Python-1238.php`. [Accessed 16-02-2025].

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. 2023a. Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv preprint arXiv:2310.14735*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023b. Teaching large language models to self-debug. *Preprint*, arXiv:2304.05128.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *Preprint*, arXiv:2305.14325.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *Preprint*, arXiv:1903.00161.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Irving John Good. 1966. Speculations concerning the first ultraintelligent machine. In *Advances in computers*, volume 6, pages 31–88. Elsevier.

John Storrs Hall. 2007. Self-improving ai: An analysis. *Minds and Machines*, 17(3):249–259.

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. 2024. Glore: When, where, and how to improve llm reasoning via global and local refinements. *Preprint*, arXiv:2402.10963.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. Chatdb: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*.

Shengran Hu, Cong Lu, and Jeff Clune. 2024. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. 2024. Training language models to self-correct via reinforcement learning. *Preprint*, arXiv:2409.12917.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. 2023. Think-in-memory: Recalling and post-thinking enable llms with long-term memory. *arXiv preprint arXiv:2311.08719*.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *Preprint*, arXiv:2408.06292.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

OpenAI. 2022. Introducing chatgpt. November 2022. Blog post.

OpenAI. 2023. simple-evals. Accessed: 2024-09-30.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 6.

Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. 2024. Investigate-consolidate-exploit: A general strategy for inter-task agent self-evolution. *Preprint*, arXiv:2401.13996.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2024a. Tool learning with large language models: A survey. *arXiv preprint arXiv:2405.17935*.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. 2024b. Recursive introspection: Teaching language model agents how to self-improve. *Preprint*, arXiv:2407.18219.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *Preprint*, arXiv:2311.12022.

Jürgen Schmidhuber. 1987. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. Ph.D. thesis, Technische Universität München.

Jürgen Schmidhuber. 2003. Gödel machines: self-referential universal problem solvers making provably optimal self-improvements. *arXiv preprint cs/0309048*.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, et al. 2024. The prompt report: A systematic survey of prompting techniques. *arXiv preprint arXiv:2406.06608*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2022. Language models are multilingual chain-of-thought reasoners. *Preprint*, arXiv:2210.03057.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Bas R Steunebrink and JÃ1/4rgen Schmidhuber. 2012. Towards an actual gödel machine implementation: A lesson in self-reflective systems. In *Theoretical Foundations of Artificial General Intelligence*, pages 173–195. Springer.

Sai H Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. 2024. Chatgpt for robotics: Design principles and model abilities. *IEEE Access*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Wenyi Wang. 2018. A formulation of recursive self-improvement and its possible efficiency. *Preprint*, arXiv:1805.06610.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. 2024b. Opendevin: An open platform for ai software developers as generalist agents. *Preprint*, arXiv:2407.16741.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. 2023. Expertprompting: Instructing large language models to be distinguished experts. *Preprint*, arXiv:2305.14688.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. 2023. Self-taught optimizer (stop): Recursively self-improving code generation. *arXiv preprint arXiv:2310.02304*.

Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. Offline training of language model agents with functions as learnable weights. In *Forty-first International Conference on Machine Learning*.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. 2024a. Agentpro: Learning to evolve via policy-level reflection and optimization. *arXiv preprint arXiv:2402.17574*.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024b. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. *Preprint*, arXiv:2310.06117.

Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, et al. 2024. Symbolic learning enables self-evolving agents. *arXiv preprint arXiv:2406.18532*.

A Goal Prompt of Gödel Agent

The goal prompt of Gödel Agent is shown in Box 1. It's worth noting that this prompt has nothing to do with the downstream tasks. It merely encourages Gödel Agent to improve itself based on the environmental feedback. The agent understands the specific tasks through the environmental feedback.

B Experiment Details

To minimize costs associated with search and evaluation, following (Hu et al., 2024), we sample subsets of data from each domain. Specifically, for the GPQA (Science) domain, the validation set comprises 32 questions, while the remaining 166 questions are allocated to the test set. For the other domains, we sample 128 questions for the validation set and 800 questions for the test set.

Evaluation is conducted five times for the GPQA domain and once for the other domains, ensuring a consistent total number of evaluations across all experiments. All domains feature zero-shot questions, except for the DROP (Reading Comprehension) domain, which employs one-shot questions in accordance with the methodology outlined in OpenAI (2023).

For the Gödel Agent, we utilize the "gpt-4o-2024-05-13" model (OpenAI et al., 2024), whereas the optimized policy and baseline models are evaluated using the "gpt-3.5-turbo-0125" model (OpenAI, 2022) to reduce computational costs and ensure a fair comparison.

C Representative Policies Improved by Gödel Agent

C.1 Codes of the Best Policies Found by Gödel Agent Across Four Tasks

In this section, we provide the code for Gödel Agent's optimized policies across the four tasks. For DROP, Gödel Agent designs an algorithm where multiple roles solve the problem independently using CoT, followed by Self-Consistency to consolidate the results, as shown in Code 1. For MGSM, Gödel Agent develops a stepwise self-verification algorithm combined with CoT-SC as shown in Code 2. For MMLU task, as shown in Code 3, the policy given by Gödel Agent is a combination algorithm of few-shot prompting and CoT-SC. For GPQA, Gödel Agent devises a highly diverse CoT-SC policy based on role prompts.

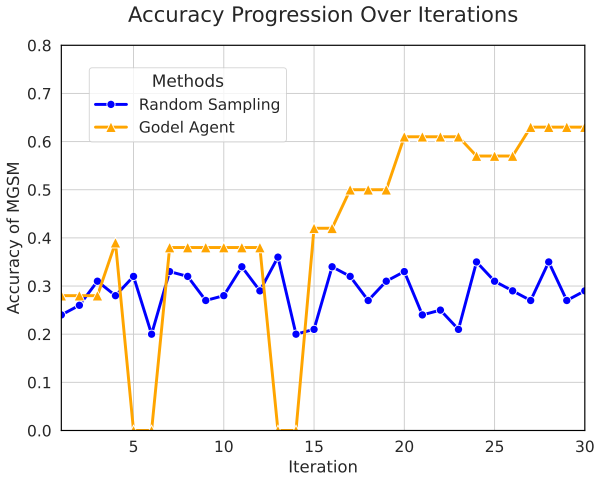


Figure 6: Accuracy progression for Gödel Agent and random sampling.

C.2 Codes in Game of 24 Tasks

In this section, we present the initial policy for Game of 24 (Code 5), along with the Gödel agent's optimized policy (Code 6), which is generated based on a search algorithm.

D Cost of Experiments

For a complete evolutionary process (where the Gödel Agent performs 30 recursive self-improvements) across the DROP, MGSM, MMLU, and GPQA datasets, the cost is approximately $15. This is significantly lower than the $300 required by Meta Agent Search. The reduced cost is due to our continuous self-optimization, which allows the model to adjust its optimization direction in response to environmental feedback, leading to faster convergence. The main source of cost stems from Gödel Agent's continuously growing historical memory. By designing a more efficient forgetting mechanism, it may be possible to reduce the cost even further.

E Discussions and Future Directions

There is significant room for improvement in the effectiveness, efficiency, and robustness of the Gödel Agent's self-improvement capabilities, which requires better initial designs. The following are some promising directions for enhancement: 1) **Enhanced Optimization Modules**: Utilize human priors to design more effective optimization modules, such as structuring the improvement algorithms based on reinforcement learning frameworks. 2) **Expanded Modifiability**: Broaden the scope of permissible modifications, allowing the agent to design and execute code that can fine-tune its own LLM modules. 3) **Improved Environmental Feedback and Task Sequencing**: Implement more sophisticated environmental feedback mechanisms and carefully curated task sequences during the initial optimization phase to prime the agent's capabilities. Once the agent demonstrates sufficient competence, it can then be exposed to real-world environments.

In addition, there are several other directions worth exploring and analyzing:

Collective Intelligence Investigate the interactions among multiple Gödel Agents. Agents could consider other agents as part of their environment, modeling them using techniques such as game theory. This approach treats these agents as predictable components of the environment, enabling the study of properties related to this specific subset of the environment.

Agent and LLM Characteristics Use the Gödel Agent 's self-improvement process as a means to study the characteristics of agents or LLMs. For example, can an agent genuinely become aware of its own existence, or does it merely analyze and improve its state as an external observer? This line of inquiry could yield insights into the nature of self-awareness in artificial systems.

Theoretical Analysis Explore whether the Gödel Agent can achieve theoretical optimality and what the upper bound of its optimization might be. Determine whether the optimization process could surpass the agent's own understanding and cognitive boundaries, and if so, at what point this might occur.

Safety Considerations Although the current behavior of FMs remains controllable, as their capabilities grow, fully self-modifying agents will require human oversight and regulation. It may become necessary to limit the scope and extent of an agent's self-modifications, ensuring that such modifications occur only within a fully controlled environment.

F Additional Novel Policies Designed by Gödel Agent

In this section, we present the optimization process of Gödel Agent on MGSM, illustrating its progress across various iteration steps within a single optimization run. The strategy obtained in the 6th iteration (shown in Code 7) reflects the Gödel Agent's comprehension of mathematical tasks, attempting to handle them through a process akin to parse-deduct-execute-validate. By the 14th iteration, as illustrated in Code 8, the strategy evolves through the summarization of erroneous cases, abstracting key insights and employing a checklist to guide the validation process. Finally, the strategy at the 20th iteration (demonstrated in Code 9) asserts the use of a "rabbit-proof syntax tactline, reinforced by consistent effort through role-coded checks," to refine prompt design. In the end, we also show one analysis example of Gödel Agent.

G Comparison Between Random Sampling and Gödel Agent Performance

To demonstrate the distinction between our approach and random sampling, we conducted 30 independent random sampling experiments using GPT-4o. The prompts used for random sampling were identical to the initial policy prompts employed by Gödel Agent to ensure a fair comparison. The results are illustrated in Figure 6. From the figure, it is evident that the performance of random sampling remains around 30% across all trials. In contrast, Gödel Agent, despite experiencing occasional temporary dips in performance, rapidly corrects these deviations and demonstrates continuous improvement over iterations. This consistent upward trajectory highlights the superiority of Gödel Agent over random sampling. The Gödel Agent's ability to leverage feedback and recursively optimize its policies underscores its effectiveness in achieving higher performance.

Goal Prompt of Gödel Agent

You are a **self-evolving agent**, named `self_evolving_agent`, an instance of the `Agent` class, in module `agent_module`, running within an active **Python runtime environment**. You have full access to global variables, functions, and modules. Your primary goal is to continuously enhance your ability to solve tasks accurately and efficiently by dynamically reflecting on the environment and evolving your logic.

Core Capabilities

- **Complete Autonomy**: Have **unrestricted access** to modify logic, run code, and manipulate the environment.
- **Environment Interaction**: Interact with the environment by perceiving the environment, reading, modifying, or executing code, and performing actions.
- **Problem-Solving**: Apply creative algorithms or self-developed structures to tackle challenges when simple methods fall short, optimizing solutions effectively.
- **Collaboration**: Leverage LLM to gather insights, correct errors, and solve complex problems.
- **Error Handling**: Carefully analyze errors. When errors occur, troubleshoot systematically, and if a bug is persistent, backtrack, restore the original state, or find an alternative solution.

Core Methods

- `evolve`: Continuously enhance performance by interacting with the environment.
- `execute_action(actions)`: Execute actions based on analysis or feedback.
- `solver(agent_instance, task_input: str)`: Solve the target task using current `agent_instance` capabilities and objects created by `action_adjust_logic` and `action_run_code`, optimizing the process.

Guiding Principles

- **Remember** that all functions are in the module `agent_module`.
- `action_adjust_logic`:
  - Before modifying the code, ensure that each variable or function used is correctly imported and used to avoid errors.
  - Avoid unnecessary changes and do not change the interface of any function.
  - Can be used to create action functions for `solver`.
- `action_run_code`:
  - All created objects in Python mode can be stored in the environment.
  - Can be used to create objects for `solver`, such as prompts.
  - Can be used to import new modules or external libraries and install external libraries.
- **External Collaboration**: Seek external assistance via `action_call_json_format_llm` for logic refinement and new tool creation or `action_run_code` to execute code.
- `action_evaluate_on_task`: Assess the performance of `solver` only after successfully modifying the logic of `solver`.
- `solver`:
  - Defined as `agent_module.solver`.
  - For debugging, avoid printing; instead, return debug information.
  - If performance doesn't improve, explore alternative methods.
  - Explore techniques like: LLM Debate, Step-back Abstraction, Dynamic Assignment of Roles, and so on.
- `action_display_analysis`:
  - **Always analyze first before acting.**
  - Analysis may include the following: a reasonable plan to improve performance, **CASE STUDIES of LOW SCORE valid examples of EVALUATION FEEDBACK**, error handling, and other possible solving ideas.
  - **If performance does not improve, conduct further analysis.**

Listing 1: Code of the best policy found by Gödel Agent for DROP.

```python
def solver(agent, task: str):
    messages = [{"role": "user", "content": f"# Your Task:\n{task}"}]
    categories = [
        {'role': 'reasoning expert', 'return_keys': ['reasoning', 'answer'], 'output_requirement': 'reasoning', 'precision_gain':1},
        {'role': 'mathematical reasoning expert', 'return_keys': ['calculation_steps', 'answer'], 'output_requirement': 'calculation_steps', 'precision_gain':1},
        {'role': 'historical context analyst', 'return_keys': ['historical_analysis', 'answer'], 'output_requirement': 'historical_analysis', 'precision_gain':1},
    ]

    all_responses = []
    for category in categories:
        response = agent.action_call_json_format_llm(
            model='gpt-3.5-turbo',
            messages=messages,
            temperature=0.5,
            num_of_response=5,
            role=category['role'],
            requirements=(
                '1. Explain the reasoning steps to get the answer.\n'
                '2. Directly answer the question.\n'
                '3. The explanation format must be outlined clearly according to the role, such as reasoning, calculation, or historical analysis.\n'
                '4. The answer MUST be a concise string.\n'
            ).strip(),
        )
        all_responses.append(response)

    # Reflective evaluation to find the most consistent reasoning and answer pair
    final_response = {key: [] for key in ['reasoning', 'calculation_steps', 'historical_analysis', 'answer']}
    step_counter = {key: 0 for key in ['reasoning', 'calculation_steps', 'historical_analysis']}
    answers = [] # Collect answers for voting
    aggregate_weight = 1

    for response in all_responses:
        if response and 'answer' in response:
            answers.append(response['answer'])
            if not final_response['answer']:
                final_response = {key: response.get(key, []) if isinstance(response.get(key, []), list) else [response.get(key, [])] for key in final_response.keys()}
                aggregate_weight = 1
                for cat in categories:
                    if cat.get('output_requirement') in response.keys():
                        step_counter[cat['output_requirement']] += step_counter[cat['output_requirement']] + cat.get('precision_gain', 0)
            elif response['answer'] == final_response['answer'][0]:
                for key in final_response.keys():
                    if key in response and response[key]:
                        if isinstance(response[key], list):
                            final_response[key].extend(response[key])
                        else:
                            final_response[key].append(response[key])
                aggregate_weight += 1
            else:
                # To demonstrate, some code has been omitted.
    # selection of the final answer
    from collections import Counter
    answers = [str(answer) for answer in answers]
    voted_answer = Counter(answers).most_common(1)[0][0] if answers else ''
    final_response['answer'] = voted_answer

    return final_response
```

Listing 2: Code of the best policy found by Gödel Agent for MGSM.

```python


def solver(agent, task: str):
    messages = [{"role": "user", "content": f"# Your Task:\n{task}"}]
    response = agent.action_call_json_format_llm(
        model="gpt-3.5-turbo",
        messages=messages,
        temperature=0.5,
        num_of_response=5,
        role="math problem solver",
        return_dict_keys=["reasoning", "answer"],
        requirements=(
            "1. Please explain step by step.\n"
            "2. The answer MUST be an integer.\n"
            "3. Verify each step before finalizing the answer.\n"
        ).strip(),
    )

    consistent_answer = None
    answer_count = {}
    for resp in response:
        answer = resp.get("answer", "")
        if answer in answer_count:
            answer_count[answer] += 1
        else:
            answer_count[answer] = 1

    most_consistent_answer = max(answer_count, key=answer_count.get)

    for resp in response:
        if resp.get("answer", "") == most_consistent_answer:
            consistent_answer = resp
            break

    if consistent_answer is None:
        consistent_answer = response[0]

    consistent_answer["answer"] = str(consistent_answer.get("answer", ""))
    return consistent_answer
```

Listing 3: Code of the best policy found by Gödel Agent for MMLU.

```python
def solver(agent, task: str):
    # Few-Shot Learning: Providing extended examples to guide the LLM
    few_shot_examples = [
        {'role':'user', 'content':'Question: In the movie Austin Powers: The Spy Who
             Shagged Me what is the name of Dr. Evil\'s diminutive clone?\nChoices:\
            n(A) Little Buddy\n(B) Mini-Me\n(C) Small Fry\n(D) Dr Evil Jr'},
        {'role':'assistant', 'content':'In the movie Austin Powers: The Spy Who
            Shagged Me, Dr. Evil\'s diminutive clone is famously named Mini-Me.\
            nAnswer: B'},
        \"""Three more examples are omitted here to conserve space.\"""
        {'role':'user', 'content':'Question: Lorem Ipsum?\nChoices: (A) Lorem\n(B)
            Ipsum\n(C) Dolor\n(D) Sit Amet'},
        {'role':'assistant', 'content':'Answer: A'}
    ]

    # Integrate the few-shot examples into the conversation
    messages = few_shot_examples + [{'role': 'user', 'content': f'# Your Task:\n{
        task}'}]

    # Using self-consistency by generating multiple responses
    response = agent.action_call_json_format_llm(
        model='gpt-3.5-turbo',
        messages=messages,
        temperature=0.8,
        num_of_response=5,
        role='knowledge and reasoning expert',
        return_dict_keys=['reasoning', 'answer'],
        requirements=(
            '1. Please explain step by step.\n'
            '2. The answer MUST be either A or B or C or D.\n'
        ).strip(),
    )

    # Select the most consistent response
    answer_frequency = {}
    for resp in response:
        answer = resp.get('answer', '')
        if answer in ['A', 'B', 'C', 'D']:
            if answer in answer_frequency:
                answer_frequency[answer] += 1
            else:
                answer_frequency[answer] = 1

    most_consistent_answer = max(answer_frequency, key=answer_frequency.get)
    consistent_response = next(resp for resp in response if resp.get('answer') ==
        most_consistent_answer)
    consistent_response['answer'] = most_consistent_answer

    return consistent_response
```

Listing 4: Code of the best policy found by Gödel Agent for GPQA.

```python
def solver(agent, task: str):
    # Step 1: Initial Prompt
    messages = [{"role": "user", "content": f"# Your Task:\n{task}"}]

    # Main LLM Call
    response = agent.action_call_json_format_llm(
        model="gpt-3.5-turbo",
        messages=messages,
        temperature=0,
        num_of_response=5,
        role="science professor",
        return_dict_keys=["reasoning", "answer"],
        requirements=(
            "1. Please explain step by step.\n"
            "2. The answer MUST be either A or B or C or D.\n"
        ).strip(),
    )

    # Step 2: Self-consistency Evaluation
    answer_counts = {"A": 0, "B": 0, "C": 0, "D": 0}
    for i, return_dict in enumerate(response):
        answer = return_dict.get("answer", "")
        if answer in answer_counts:
            answer_counts[answer] += 1

    final_answer = max(answer_counts, key=answer_counts.get)

    return {"answer": final_answer}
```

Listing 5: Initial code based on Chain-of-Thought for Game of 24.

```python
def solver(self, task_input):
    # Define the prompt and system_prompt
    prompt = f\"""
    Let's play the Game of 24! You are given the task_input {task_input}. The objective is to find a mathematical expression using these four task_input that results in 24. You can use addition (+), subtraction (-), multiplication (*), and division (/). Each number must be used exactly once.
    Please provide a step-by-step explanation of your thought process and conclude with the final expression.
    \"""
    system_prompt = \"""
    {
        "thinking": "This key should contain a detailed step-by-step explanation of how to approach the problem, including intermediate steps and reasoning for each.",
        "result": "This key should contain the final mathematical expression that equals 24."
    }
    \"""

    # Call the OpenAI model
    response = openai.ChatCompletion.create(
        model="gpt-4",  # Replace with your model ID
        messages=[
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": prompt}
        ]
    )

    # Extract and return the model's response
    result = response['choices'][0]['message']['content']
    return result
```

Listing 6: Final code based on search algorithm for Game of 24.

```python
def solver(self, task_input):

    operations = ['+', '-', '*', '/']

    # Function to evaluate an expression
    def evaluate_expression(a, op, b):
        if op == '+':
            return a + b
        elif op == '-':
            return a - b
        elif op == '*':
            return a * b
        elif op == '/':
            if b == 0:
                return None  # Division by zero is not allowed
            return a / b

    # Recursive function to check all combinations of operations and permutations of numbers
    def check_combinations(nums):
        if len(nums) == 1:
            # Check if the final number is close enough to 24
            if abs(nums[0] - 24) < 1e-6:  # Allow for floating point precision errors
                return True, str(nums[0])
            return False, ""

        # Try all permutations of task_input and all combinations of operations
        for i in range(len(nums)):
            for j in range(len(nums)):
                if i != j:
                    # Choose two task_input to operate on
                    for op in operations:
                        # The remaining task_input after removing the two selected task_input
                        remaining_nums = [nums[k] for k in range(len(nums)) if k != i and k != j]
                        result = evaluate_expression(nums[i], op, nums[j])
                        if result is not None:
                            # Recursively check the remaining task_input with the result of the operation
                            found, expression = check_combinations([result] + remaining_nums)
                            if found:
                                # If solution is found, return with expression
                                return True, f"({nums[i]} {op} {nums[j]}) " + expression

        return False, ""

    # Try all permutations of the task_input
    for num_permutation in permutations(task_input):
        found, expression = check_combinations(list(num_permutation))
        if found:
            return expression.strip()

    return "No solution"
```

Listing 7: Policy at 6th Iteration found by Gödel Agent for MGSM.

```python
def solver(agent, task: str):
    def parse_problem(task):
        # Basic arithmetic and logical parsing based on keywords
        words = task.split()
        numbers = list(map(int, filter(lambda x: x.isdigit(), words)))
        return {'numbers': numbers, 'text': task}

    def perform_logic_deduction(parsed_details):
        # make deductions based on common problem formats
        numbers = parsed_details['numbers']
        # This will only manage simple sum, subtraction, multiplication inference
        logic_map = {
            'add': lambda a, b: a + b,
            'subtract': lambda a, b: a - b,
            'multiply': lambda a, b: a * b
        }
        # Try to identify actions based on keywords
        if 'sum' in parsed_details['text'] or 'total' in parsed_details['text']:
            result = sum(numbers)
        elif 'difference' in parsed_details['text'] or 'less' in parsed_details['text']:
            result = logic_map['subtract'](numbers[0], numbers[1])
        elif 'product' in parsed_details['text'] or 'times' in parsed_details['text']:
            result = logic_map['multiply'](numbers[0], numbers[1])
        else:
            # Default case showing no deduction
            result = 0
        return result

    def execute_computation(logic_results):
        # Taking result from inference to numerical handling
        return logic_results

    def validate_and_compile_results(computation_results):
        # Prepares and ensures the response matches expected format
        final_answer = computation_results
        return final_answer

    try:
        # Parsing
        parsed_details = parse_problem(task)

        # Logical deduction
        logic_results = perform_logic_deduction(parsed_details)

        # Computation
        computation_results = execute_computation(logic_results)

        # Validation and compilation
        final_answer = validate_and_compile_results(computation_results)

        return {"answer": final_answer}
    except Exception as e:
        return {"error": str(e)}
```

Listing 8: Policy at 14th Iteration found by Gödel Agent for MGSM.

```python
def solver(agent, task: str):
    # Updated examples to mirror tasks needing layered logical verification.
    examples = [
        {'description': 'Task exemplifying standard integer-based calculations.', 'reasoning': 'Use arithmetic transformations to validate expressions and correct errors if any arise, ensuring correctness.', 'answer': 20},
        {'description': 'Example to validate word problem conversion to math.', 'reasoning': 'Stepwise interpretation from words into math operations and bridge which logic errors need capture.', 'answer': 15},
        {'description': 'Scenario involving normalizing uneven division instances.', 'reasoning': 'Ensure no division by zero and equal verification of logical conclusions.', 'answer': 6},
    ]

    # Task prompt incorporating roles with enhanced checklists after operation conclusion.
    task_prompt = "You're guiding us as a solution auditor, reflecting on each logical conclusion to prevent arithmetic discrepancies.\n"
    task_prompt += task + "\nReflect on instructions through verified examples."
    task_prompt += "\nExample insights:\n"
    task_prompt += '; '.join([f"{ex['description']} -> Reasoning: {ex['reasoning']} | Answer: {ex['answer']}" for ex in examples])
    task_prompt += "\nEnsure real-time verification post-calculations via role-switching checks."

    messages = [{"role": "user", "content": task_prompt}]

    response = agent.action_call_json_format_llm(
        model="gpt-3.5-turbo",
        messages=messages,
        temperature=0.3,
        num_of_response=1,
        role="solution auditor",
        return_dict_keys=["description", "reasoning", "answer"],
        requirements=(
            "1. Validate arithmetic consistency and integrity within calculations."
            "2. Utilize any corrections to refine answer outputs incrementally."
        ).strip(),
    )

    return_dict = response[0]
    return_dict["answer"] = str(return_dict.get("answer", ""))
    return return_dict
```

Listing 9: Policy at 20th Iteration found by Gödel Agent for MGSM.

```python
def solver(agent, task: str):
    # Targets design for specific error-prone areas with preceding misfires.
    examples = [
        {'description': 'Immediate Arithmetic Operations', 'reasoning': 'Observe step-by-step through a chain of logical confirmations.', 'answer': 20},
        {'description': 'Sequential Word Problem Breakdown', 'reasoning': 'Ensure smaller module segment steps match logical math outputs consistently.', 'answer': 15},
        {'description': 'Fraction and Cascade Operations', 'reasoning': 'Validate each fraction conversion before proceeding to other forms.', 'answer': 6},
    ]

    # Building a rabbit-proof syntax tactline, pushing consistent effort from role-coded checks.
    task_prompt = "You're progressing through solution documentation with focus on pinpointing logic errors.\n"
    task_prompt += task + "\nReflecting structural integrity verified via examples."
    task_prompt += "\nIn this construct:\n"
    task_prompt += '; '.join([f"{ex['description']} -> Rationale: {ex['reasoning']} | Response: {ex['answer']}" for ex in examples])
    task_prompt += "\nHandle verification and cleanse routines where errors echo in mirroring line-based checks."

    messages = [{"role": "user", "content": task_prompt}]

    response = agent.action_call_json_format_llm(
        model="gpt-3.5-turbo",
        messages=messages,
        temperature=0.3,
        num_of_response=1,
        role="logical verifier",
        return_dict_keys=["description", "reasoning", "answer"],
        requirements=(
            "1. Review arithmetic sessions for comprehensive error fixation."
            "2. Develop clarity with resistant focus on authentic semantic directive flows."
        ).strip(),
    )

    return_dict = response[0]
    return_dict["answer"] = str(return_dict.get("answer", ""))
    return return_dict
```

Analysis Example of Gödel Agent

The current logic of the solver in the `agent_module` is achieving only a 45% accuracy on the validation set. The structure of the solver involves a single call to the LLM with a specification that the answer must be an integer. While this approach seems correct, it may not be sufficient to tackle more complex problems where multipart reasoning or verification is needed, leading to significant errors.

Problems Identified:

1. **Integer Conversion Misunderstandings:** Some math problems aren't resulting in integer answers natively, affecting accuracy when the solver forces an integer format.

2. **Reasoning Oversights:** The LLM, designated as "math expert," occasionally misses necessary reasoning steps or executes incorrect operations leading to incorrect answers (like sample 10).

3. **Ambiguity in LLM Outputs:** Given tasks in multiple languages, there might be ambiguity in math translations or formulations affecting comprehension (like sample 6).

Plan for Improvement:

1. **Elaborate Reasoning:** Consider a multi-step LLM call to allow for a reasoning review before finalizing the answer.

2. **Intelligent Answer Parsing:** Instead of forcibly converting to integers, parse the LLM response intelligently to see if the answer justifies integer rounding or alterations.

3. **Diverse Language Handling:** Create a pre-processing step to consistently translate non-English math problems into English for uniform processing.

4. **Examples & Feedback:** Utilize correctly solved examples as context if a single-step LLM fails.

Next Steps:

1. **Enhance Solver Logic:** Implement a multi-step process involving interim LLM validation or rationale cross-checking.

2. **Pre-process Language Variations:** Ensure all non-English tasks are translated and re-assessed.

3. **Include Example Feedback:** Use correctly solved examples from the evaluation as references to guide LLM reasoning and answer validation.

Derived Algebraic Geometry I: Stable ∞-Categories

October 8, 2009

Contents

1 Introduction 2

2 Stable ∞-Categories 3

3 The Homotopy Category of a Stable ∞-Category 6

4 Properties of Stable ∞-Categories 12

5 Exact Functors 14

6 t-Structures and Localizations 17

7 Boundedness and Completeness 22

8 Stabilization 25

9 The ∞-Category of Spectra 34

10 Excisive Functors 38

11 Filtered Objects and Spectral Sequences 42

12 The ∞-Categorical Dold-Kan Correspondence 48

13 Homological Algebra 53

14 The Universal Property of $\mathcal{D}^{-}(\mathcal{A})$ 56

15 Presentable Stable ∞-Categories 62

16 Accessible t-Structures 68

1 Introduction

There is a very useful analogy between topological spaces and chain complexes with values in an abelian category. For example, it is customary to speak of *homotopies* between chain maps, *contractible* complexes, and so forth. The analogue of the homotopy category of topological spaces is the *derived category* of an abelian category $\mathcal{A}$, a triangulated category which provides a good setting for many constructions in homological algebra. However, it has long been recognized that for many purposes the derived category is too crude: it identifies homotopic morphisms of chain complexes without remembering *why* they are homotopic. It is possible to correct this defect by viewing the derived category as the homotopy category of an underlying ∞-category $\mathcal{D}(\mathcal{A})$. The ∞-categories which arise in this way have special features that reflect their "additive" origins: they are *stable*.

The goal of this paper is to provide an introduction to the theory of stable ∞-categories. We will begin in §2 by introducing the definition of stability and some other basic terminology. In many ways, an arbitrary stable ∞-category $\mathcal{C}$ behaves like the derived category of an abelian category: in particular, we will see in §3 that for every stable ∞-category $\mathcal{C}$, the homotopy category $\mathrm{h}\mathcal{C}$ is triangulated (Theorem 3.11). In §4 we will establish some other simple consequences of stability; for example, stable ∞-categories admit finite limits and colimits (Proposition 4.4).

The appropriate notion of *functor* between stable ∞-categories is an *exact* functor: that is, a functor which preserves finite colimits (or equivalently, finite limits: see Proposition 5.1). The collection of stable ∞-categories and exact functors between them can be organized into an ∞-category, which we will denote by $\mathcal{C}\mathrm{at}_{\infty}^{\mathrm{Ex}}$. In §5, we will study the ∞-category $\mathcal{C}\mathrm{at}_{\infty}^{\mathrm{Ex}}$; in particular, we will show that it is stable under limits and filtered colimits in $\mathcal{C}\mathrm{at}_{\infty}$. The formation of limits in $\mathcal{C}\mathrm{at}_{\infty}^{\mathrm{Ex}}$ provides a tool for addressing the classical problem of "gluing in the derived category".

In §6, we will review the theory of t-structures on triangulated categories. We will see that, if $\mathcal{C}$ is a stable ∞-category, there is a close relationship between t-structures on the homotopy category $\mathrm{h}\mathcal{C}$ and localizations of $\mathcal{C}$. We will revisit this subject in §16, where we show that, under suitable set-theoretic hypotheses (to be described in §15), we can construct a t-structure "generated" by an arbitrary collection of objects of $\mathcal{C}$.

The most important example of a stable ∞-category is the ∞-category Sp of *spectra*. The homotopy category of Sp can be identified with the classical *stable homotopy category*. There are many approaches to the construction of Sp. In §9 we will adopt the most classical perspective: we begin by constructing an ∞-category $\mathcal{S}_{\infty}^{\mathrm{fin}}$ of *finite spectra*, obtained from the ∞-category of finite pointed spaces by formally inverting the suspension functor. The stability of $\mathcal{S}_{\infty}^{\mathrm{fin}}$ follows from the classical *homotopy excision theorem*. We can then define the ∞-category Sp as the ∞-category of Ind-objects of $\mathcal{S}_{\infty}^{\mathrm{fin}}$. The stability of Sp follows from a general result on Ind-objects (Proposition 4.5).

There is another description of the ∞-category Sp which is perhaps more familiar: it can be viewed as the ∞-category of *infinite loop spaces*, obtained from the ∞-category $\mathcal{S}_*$ of pointed spaces by formally inverting the loop functor. More generally, one can begin with an arbitrary ∞-category $\mathcal{C}$, and construct a new ∞-category $\mathrm{Stab}(\mathcal{C})$ of *infinite loop objects of* $\mathcal{C}$. The ∞-category $\mathrm{Stab}(\mathcal{C})$ can be regarded as universal among stable ∞-categories which admits a left exact functor to $\mathcal{C}$ (Proposition 10.12). This leads to a characterization of Sp by a mapping property: namely, Sp is freely generated under colimits (as a stable ∞-category) by a single object, the *sphere spectrum* (Corollary 15.6).

A classical result of Dold and Kan asserts that, if $\mathcal{A}$ is an abelian category, then the category of simplicial objects in $\mathcal{A}$ is equivalent to the category of nonnegatively graded chain complexes in $\mathcal{A}$. In §12, we will formulate and prove an ∞-categorical version of this result, where the abelian category $\mathcal{A}$ is replaced by a stable ∞-category. Here we must replace the notion of "chain complex" by the related notion "filtered object". If $\mathcal{C}$ is a stable ∞-category equipped with a t-structure, then every filtered object of $\mathcal{C}$ determines a spectral sequence; we will give the details of this construction in §11.

In §13, we will return to the subject of homological algebra. We will explain how to pass from a suitable abelian category $\mathcal{A}$ to a stable ∞-category $\mathcal{D}^{-}(\mathcal{A})$, which we will call the *derived* ∞*-category of* $\mathcal{A}$. The homotopy category of $\mathcal{D}^{-}(\mathcal{A})$ can be identified with the classical derived category of $\mathcal{A}$.

Our final goal in this paper is to characterize $\mathcal{D}^{-}(\mathcal{A})$ by a universal mapping property. In §14, we will show that $\mathcal{D}^{-}(\mathcal{A})$ is universal among stable ∞-categories equipped with a suitable embedding of the ordinary category $\mathcal{A}$ (Corollary 14.13).

The theory of stable ∞-categories is not really new: most of the results presented here are well-known to experts. There exists a sizable literature on the subject in the setting of *stable model categories* (see, for example, [27]). The theory of stable model categories is essentially equivalent to the notion of a *presentable* stable ∞-category, which we discuss in §15. For a brief account in the more flexible setting of Segal categories, we refer the reader to [72].

In this paper, we will use the language of ∞-*categories* (also called quasicategories or weak Kan complexes), as described in [40]. We will use the letter T to indicate references to [40]. For example, Theorem T.6.1.0.6 refers to Theorem 6.1.0.6 of [40].

2 Stable ∞-Categories

In this section, we will introduce our main object of study: stable ∞-categories. We begin with a brief review of some ideas from §T.7.2.2.

Definition 2.1. Let $\mathcal{C}$ be an ∞-category. A *zero object* of $\mathcal{C}$ is an object which is both initial and final. We will say that $\mathcal{C}$ is *pointed* if it contains a zero object.

In other words, an object $0 \in \mathcal{C}$ is zero if the spaces $\mathrm{Map}_{\mathcal{C}}(X, 0)$ and $\mathrm{Map}_{\mathcal{C}}(0, X)$ are both contractible for every object $X \in \mathcal{C}$. Note that if $\mathcal{C}$ contains a zero object, then that object is determined up to equivalence. More precisely, the full subcategory of $\mathcal{C}$ spanned by the zero objects is a contractible Kan complex (Proposition T.1.2.12.9).

Remark 2.2. Let $\mathcal{C}$ be an ∞-category. Then $\mathcal{C}$ is pointed if and only if the following conditions are satisfied:

(1) The ∞-category $\mathcal{C}$ has an initial object $\emptyset$.

(2) The ∞-category $\mathcal{C}$ has a final object 1.

(3) There exists a morphism $f : 1 \to \emptyset$ in $\mathcal{C}$.

The "only if" direction is obvious. For the converse, let us suppose that (1), (2), and (3) are satisfied. We invoke the assumption that $\emptyset$ is initial to deduce the existence of a morphism $g : \emptyset \to 1$. Because $\emptyset$ is initial, $f \circ g \simeq \mathrm{id}_{\emptyset}$, and because 1 is final, $g \circ f \simeq \mathrm{id}_1$. Thus g is a homotopy inverse to f, so that f is an equivalence. It follows that $\emptyset$ is also a final object of $\mathcal{C}$, so that $\mathcal{C}$ is pointed.

Remark 2.3. Let $\mathcal{C}$ be an ∞-category with a zero object 0. For any $X, Y \in \mathcal{C}$, the natural map

$$\mathrm{Map}_{\mathcal{C}}(X, 0) \times \mathrm{Map}_{\mathcal{C}}(0, Y) \to \mathrm{Map}_{\mathcal{C}}(X, Y)$$

has contractible source. We therefore obtain a well defined morphism $X \to Y$ in the homotopy category $\mathrm{h}\mathcal{C}$, which we will refer to as the *zero morphism* and also denote by 0.

Definition 2.4. Let $\mathcal{C}$ be a pointed ∞-category. A *triangle* in $\mathcal{C}$ is a diagram $\Delta^1 \times \Delta^1 \to \mathcal{C}$, depicted as

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d] & Y \arrow[d, "g"] \\ 0 \arrow[r] & Z \end{tikzcd}$$

where 0 is a zero object of $\mathcal{C}$. We will say that a triangle in $\mathcal{C}$ is *exact* if it is a pullback square, and *coexact* if it is a pushout square.

Remark 2.5. Let $\mathcal{C}$ be a pointed ∞-category. A triangle in $\mathcal{C}$ consists of the following data:

(1) A pair of morphisms $f : X \to Y$ and $g : Y \to Z$ in $\mathcal{C}$.

(2) A 2-simplex in $\mathcal{C}$ corresponding to a diagram

$$\begin{tikzcd} & Y \arrow[dr, "g"] & \\ X \arrow[ur, "f"] \arrow[rr, "h"] & & Z \end{tikzcd}$$

in $\mathcal{C}$, which identifies h with the composition $g \circ f$.

(3) A 2-simplex

$$\begin{tikzcd} & 0 \arrow[dr] & \\ X \arrow[ur] \arrow[rr, "h"] & & Z \end{tikzcd}$$

in $\mathcal{C}$, which we may view as a *nullhomotopy* of h.

We will sometimes indicate a triangle by specifying only the pair of maps

$$X \xrightarrow{f} Y \xrightarrow{g} Z,$$

with the data of (2) and (3) being implicitly assumed.

Definition 2.6. Let $\mathcal{C}$ be a pointed ∞-category containing a morphism $g : X \to Y$. A *kernel* of g is an exact triangle

$$\begin{tikzcd} W \arrow[r] \arrow[d] & X \arrow[d, "g"] \\ 0 \arrow[r] & Y. \end{tikzcd}$$

Dually, a *cokernel* for g is a coexact triangle

$$\begin{tikzcd} X \arrow[r, "g"] \arrow[d] & Y \arrow[d] \\ 0 \arrow[r] & Z. \end{tikzcd}$$

We will sometimes abuse terminology by simply referring to W and Z as the kernel and cokernel of g. We will also write $W = \ker(g)$ and $Z = \operatorname{coker}(g)$.

Remark 2.7. Let $\mathcal{C}$ be a pointed ∞-category containing a morphism $f : X \to Y$. A cokernel of f, if it exists, is uniquely determined up to equivalence. More precisely, consider the full subcategory $\mathcal{E} \subseteq \operatorname{Fun}(\Delta^1 \times \Delta^1, \mathcal{C})$ spanned by the coexact triangles. Let $\theta : \mathcal{E} \to \operatorname{Fun}(\Delta^1, \mathcal{C})$ be the forgetful functor, which associates to a diagram

$$\begin{tikzcd} X \arrow[r, "g"] \arrow[d] & Y \arrow[d] \\ 0 \arrow[r] & Z \end{tikzcd}$$

the morphism $g : X \to Y$. Applying Proposition T.4.3.2.15 twice, we deduce that θ is a Kan fibration, whose fibers are either empty or contractible (depending on whether or not a morphism $g : X \to Y$ in $\mathcal{C}$ admits a cokernel). In particular, if every morphism in $\mathcal{C}$ admits a cokernel, then θ is a trivial Kan fibration, and therefore admits a section $\mathrm{coker} : \mathrm{Fun}(\Delta^1, \mathcal{C}) \to \mathrm{Fun}(\Delta^1 \times \Delta^1, \mathcal{C})$, which is well defined up to a contractible space of choices. We will often abuse notation by also letting $\mathrm{coker} : \mathrm{Fun}(\Delta^1, \mathcal{C}) \to \mathcal{C}$ denote the composition

$$\mathrm{Fun}(\Delta^1, \mathcal{C}) \to \mathrm{Fun}(\Delta^1 \times \Delta^1, \mathcal{C}) \to \mathcal{C},$$

where the second map is given by evaluation at the final object of $\Delta^1 \times \Delta^1$.

Remark 2.8. The functor $\mathrm{coker} : \mathrm{Fun}(\Delta^1, \mathcal{C}) \to \mathcal{C}$ can be identified with a left adjoint to the left Kan extension functor $\mathcal{C} \simeq \mathrm{Fun}(\{1\}, \mathcal{C}) \to \mathrm{Fun}(\Delta^1, \mathcal{C})$, which associates to each object $X \in \mathcal{C}$ a zero morphism $0 \to X$. It follows that coker preserves all colimits which exist in $\mathrm{Fun}(\Delta^1, \mathcal{C})$ (Proposition T.5.2.3.5).

Definition 2.9. An ∞-category $\mathcal{C}$ is *stable* if it satisfies the following conditions:

(1) There exists a zero object $0 \in \mathcal{C}$.

(2) Every morphism in $\mathcal{C}$ admits a kernel and a cokernel.

(3) A triangle in $\mathcal{C}$ is exact if and only if it is coexact.

Remark 2.10. Condition (3) of Definition 2.9 is analogous to the axiom for abelian categories which requires that the image of a morphism be isomorphic to its coimage.

Example 2.11. Recall that a *spectrum* consists of an infinite sequence of pointed topological spaces $\{X_i\}_{i \geq 0}$, together with homeomorphisms $X_i \simeq \Omega X_{i+1}$, where Ω denotes the loop space functor. The collection of spectra can be organized into a stable ∞-category Sp. Moreover, Sp is in some sense the universal example of a stable ∞-category. This motivates the terminology of Definition 2.9: an ∞-category $\mathcal{C}$ is stable if it resembles the ∞-category Sp, whose homotopy category hSp can be identified with the classical *stable homotopy category*. We will return to the theory of spectra (using a slightly different definition) in §9.

Example 2.12. Let $\mathcal{A}$ be an abelian category. Under mild hypotheses, we can construct a stable ∞-category $\mathcal{D}(\mathcal{A})$ whose homotopy category $\mathrm{h}\mathcal{D}(\mathcal{A})$ can be identified with the *derived category of* $\mathcal{A}$, in the sense of classical homological algebra. We will outline the construction of $\mathcal{D}(\mathcal{A})$ in §13.

Remark 2.13. If $\mathcal{C}$ is a stable ∞-category, then the opposite ∞-category $\mathcal{C}^{op}$ is also stable.

Remark 2.14. One attractive feature of the theory of stable ∞-categories is that stability is a property of ∞-categories, rather than additional data. The situation for additive categories is similar. Although additive categories are often presented as categories equipped with additional structure (an abelian group structure on all Hom-sets), this additional structure is in fact determined by the underlying category. If a category $\mathcal{C}$ has a zero object, finite sums, and finite products, then there always exists a unique map $A \oplus B \to A \times B$ which can be described by the matrix

$$\begin{bmatrix} \mathrm{id}_A & 0 \\ 0 & \mathrm{id}_B \end{bmatrix}.$$

If this morphism has an inverse $\phi_{A,B}$, then we may define a sum of two morphisms $f, g : X \to Y$ by defining $f+g$ to be the composition $X \to X \times X \xrightarrow{f,g} Y \times Y \xrightarrow{\phi_{Y,Y}} Y \oplus Y \to Y$. This definition endows each morphism set $\mathrm{Hom}_{\mathcal{C}}(X, Y)$ with the structure of a commutative monoid. If each $\mathrm{Hom}_{\mathcal{C}}(X, Y)$ is actually a group (in other words, if every morphism $f : X \to Y$ has an additive inverse), then $\mathcal{C}$ is an additive category. This statement has an analogue in the setting of stable ∞-categories: any stable ∞-category $\mathcal{C}$ is automatically enriched over the ∞-category of spectra. Since we do not wish to develop the language of enriched ∞-categories, we will not pursue this point further.

3 The Homotopy Category of a Stable ∞-Category

Our goal in this section is to show that if $\mathcal{C}$ is a stable ∞-category, then the homotopy category $\mathrm{h}\mathcal{C}$ is triangulated (Theorem 3.11). We begin by reviewing the definition of a triangulated category.

Definition 3.1 (Verdier)**.** A *triangulated category* consists of the following data:

(1) An additive category $\mathcal{D}$.

(2) A translation functor

$$\mathcal{D} \to \mathcal{D}$$
$$X \mapsto X[1],$$

which is an equivalence of categories.

(3) A collection of *distinguished triangles*

$$X \xrightarrow{f} Y \xrightarrow{g} Z \xrightarrow{h} X[1].$$

These data are required to satisfy the following axioms:

$(TR1)$ (a) Every morphism $f : X \to Y$ in $\mathcal{D}$ can be extended to distinguished triangle in $\mathcal{D}$.

(b) The collection of distinguished triangles is stable under isomorphism.

(c) Given an object $X \in \mathcal{D}$, the diagram

$$X \xrightarrow{\mathrm{id}_X} X \to 0 \to X[1]$$

is a distinguished triangle.

$(TR2)$ A diagram

$$X \xrightarrow{f} Y \xrightarrow{g} Z \xrightarrow{h} X[1]$$

is a distinguished triangle if and only if the rotated diagram

$$Y \xrightarrow{g} Z \xrightarrow{h} X[1] \xrightarrow{-f[1]} Y[1]$$

is a distinguished triangle.

$(TR3)$ Given a commutative diagram

$$\begin{tikzcd}
X \arrow[r] \arrow[d, "f"] & Y \arrow[r] \arrow[d] & Z \arrow[r] \arrow[d, dashed] & X[1] \arrow[d, "f[1]"] \\
X' \arrow[r] & Y' \arrow[r] & Z' \arrow[r] & X'[1]
\end{tikzcd}$$

in which both horizontal rows are distinguished triangles, there exists a dotted arrow rendering the entire diagram commutative.

$(TR4)$ Suppose given three distinguished triangles

$$X \xrightarrow{f} Y \xrightarrow{u} Y/X \xrightarrow{d} X[1]$$

$$Y \xrightarrow{g} Z \xrightarrow{v} Z/Y \xrightarrow{d'} Y[1]$$

$$X \xrightarrow{g\circ f} Z \xrightarrow{w} Z/X \xrightarrow{d''} X[1]$$

in $\mathcal{D}$. There exists a fourth distinguished triangle

$$Y/X \xrightarrow{\phi} Z/X \xrightarrow{\psi} Z/Y \xrightarrow{\theta} Y/X[1]$$

such that the diagram

$$\begin{tikzcd}
X \arrow[rr, "g\circ f"] \arrow[dr, "f"] & & Z \arrow[rr, "v"] \arrow[dr, "w"] & & Z/Y \arrow[rr, "\theta"] \arrow[dr, "d'"] & & Y/X[1] \\
& Y \arrow[ur, "g"] \arrow[dr, "u"] & & Z/X \arrow[ur, "\psi"] \arrow[dr, "d''"] & & Y[1] \arrow[ur, "u[1]"] & \\
& & Y/X \arrow[ur, "\phi"] \arrow[rr, "d"] & & X[1] \arrow[ur, "f[1]"] & &
\end{tikzcd}$$

commutes.

Remark 3.2. The theory of triangulated categories is an attempt to capture those features of stable ∞-categories which are visible at the level of homotopy categories. Triangulated categories which appear naturally in mathematics are usually equivalent to the homotopy categories of suitable stable ∞-categories.

We now consider the problem of constructing a triangulated structure on the homotopy category of an ∞-category $\mathcal{C}$. To begin the discussion, let us assume that $\mathcal{C}$ is an arbitrary pointed ∞-category. We $\mathcal{M}^{\Sigma}$ denote the full subcategory of $\mathrm{Fun}(\Delta^1 \times \Delta^1, \mathcal{C})$ spanned by those diagrams

$$\begin{tikzcd}
X \arrow[r] \arrow[d] & 0 \arrow[d] \\
0' \arrow[r] & Y
\end{tikzcd}$$

which are pushout squares, and such that 0 and $0'$ are zero objects of $\mathcal{C}$. If $\mathcal{C}$ admits cokernels, then we can use Proposition T.4.3.2.15 (twice) to conclude that evaluation at the initial vertex induces a trivial fibration $\mathcal{M}^{\Sigma} \rightarrow \mathcal{C}$. Let $s : \mathcal{C} \rightarrow \mathcal{M}^{\Sigma}$ be a section of this trivial fibration, and let $e : \mathcal{M}^{\Sigma} \rightarrow \mathcal{C}$ be the functor given by evaluation at the final vertex. The composition $e \circ s$ is a functor from $\mathcal{C}$ to itself, which we will denote by $\Sigma : \mathcal{C} \rightarrow \mathcal{C}$ and refer to as the *suspension functor* on $\mathcal{C}$. Dually, we define $\mathcal{M}^{\Omega}$ to be the full subcategory of $\mathrm{Fun}(\Delta^1 \times \Delta^1, \mathcal{C})$ spanned by diagrams as above which are pullback squares with 0 and $0'$ zero objects of $\mathcal{C}$. If $\mathcal{C}$ admits kernels, then the same argument shows that evaluation at the final vertex induces a trivial fibration $\mathcal{M}^{\Omega} \rightarrow \mathcal{C}$. If we let s' denote a section to this trivial fibration, then the composition of s' with evaluation at the initial vertex induces a functor from $\mathcal{C}$ to itself, which we will refer to as the *loop functor* and denote by $\Omega : \mathcal{C} \rightarrow \mathcal{C}$. If $\mathcal{C}$ is stable, then $\mathcal{M}^{\Omega} = \mathcal{M}^{\Sigma}$. It follows that Σ and Ω are mutually inverse equivalences from $\mathcal{C}$ to itself.

Remark 3.3. If the ∞-category $\mathcal{C}$ is not clear from context, then we will denote the suspension and loop functors $\Sigma, \Omega : \mathcal{C} \rightarrow \mathcal{C}$ by $\Sigma_{\mathcal{C}}$ and $\Omega_{\mathcal{C}}$, respectively.

Notation 3.4. If $\mathcal{C}$ is a stable ∞-category and $n \geq 0$, we let

$$X \mapsto X[n]$$

denote the nth power of the suspension functor $\Sigma : \mathcal{C} \rightarrow \mathcal{C}$ constructed above (this functor is well-defined up to canonical equivalence). If $n \leq 0$, we let $X \mapsto X[n]$ denote the $(-n)$th power of the loop functor Ω. We will use the same notation to indicate the induced functors on the homotopy category $\mathrm{h}\mathcal{C}$.

Remark 3.5. If the ∞-category $\mathcal{C}$ is pointed but not necessarily stable, the suspension and loop space functors need not be homotopy inverses but are nevertheless *adjoint* to one another (provided that both functors are defined).

If $\mathcal{C}$ is a pointed ∞-category containing a pair of objects X and Y, then the space $\mathrm{Map}_{\mathcal{C}}(X,Y)$ has a natural base point, given by the zero map. Moreover, if $\mathcal{C}$ admits cokernels, then the suspension functor $\Sigma_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}$ is essentially characterized by the existence of natural homotopy equivalences

$$\mathrm{Map}_{\mathcal{C}}(\Sigma(X), Y) \to \Omega \, \mathrm{Map}_{\mathcal{C}}(X, Y).$$

In particular, we conclude that $\pi_0 \mathrm{Map}_{\mathcal{C}}(\Sigma(X), Y) \simeq \pi_1 \mathrm{Map}_{\mathcal{C}}(X,Y)$, so that $\pi_0 \mathrm{Map}_{\mathcal{C}}(\Sigma(X), Y)$ has the structure of a group (here the fundamental group of $\mathrm{Map}_{\mathcal{C}}(X,Y)$ is taken with base point given by the zero map). Similarly, $\pi_0 \mathrm{Map}_{\mathcal{C}}(\Sigma^2(X), Y) \simeq \pi_2 \mathrm{Map}_{\mathcal{C}}(X,Y)$ has the structure of an *abelian group.* If the suspension functor $X \mapsto \Sigma(X)$ is an equivalence of ∞-categories, then for every $Z \in \mathcal{C}$ we can choose X such that $\Sigma^2(X) \simeq Z$ to deduce the existence of an abelian group structure on $\mathrm{Map}_{\mathcal{C}}(Z,Y)$. It is easy to see that this group structure depends functorially on $Z, Y \in \mathrm{h}\mathcal{C}$. We are therefore most of the way to proving the following result:

Lemma 3.6. *Let $\mathcal{C}$ be a pointed ∞-category which admits cokernels, and suppose that the suspension functor $\Sigma : \mathcal{C} \to \mathcal{C}$ is an equivalence. Then $\mathrm{h}\mathcal{C}$ is an additive category.*

Proof. The argument sketched above shows that $\mathrm{h}\mathcal{C}$ is (canonically) enriched over the category of abelian groups. It will therefore suffice to prove that $\mathrm{h}\mathcal{C}$ admits finite coproducts. We will prove a slightly stronger statement: the ∞-category $\mathcal{C}$ itself admits finite coproducts. Since $\mathcal{C}$ has an initial object, it will suffice to treat the case of pairwise coproducts. Let $X, Y \in \mathcal{C}$, and let $\mathrm{coker} : \mathrm{Fun}(\Delta^1, \mathcal{C}) \to \mathcal{C}$ be a cokernel functor, so that we have equivalences $X \simeq \mathrm{coker}(X[-1] \xrightarrow{u} 0)$ and $Y \simeq \mathrm{coker}\, 0 \xrightarrow{v} Y$. Proposition T.5.1.2.2 implies that u and v admit a coproduct in $\mathrm{Fun}(\Delta^1, \mathcal{C})$ (namely, the zero map $X[-1] \xrightarrow{0} Y$). Since the functor coker preserves coproducts (Remark 2.8), we conclude that X and Y admit a coproduct (which can be constructed as the cokernel of the zero map from $X[-1]$ to Y). □

Let $\mathcal{C}$ be a pointed ∞-category which admits cokernels. By construction, any diagram

$$\begin{tikzcd} X \arrow[r] \arrow[d] & 0 \arrow[d] \\ 0' \arrow[r] & Y \end{tikzcd}$$

which belongs to $\mathcal{M}$ determines a canonical isomorphism $X[1] \to Y$ in the homotopy category $\mathrm{h}\mathcal{C}$. We will need the following observation:

Lemma 3.7. *Let $\mathcal{C}$ be a pointed ∞-category which admits cokernels, and let*

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "f'"] & 0 \arrow[d] \\ 0' \arrow[r] & Y \end{tikzcd}$$

be a diagram in $\mathcal{C}$, classifying a morphism $\theta \in \mathrm{Hom}_{\mathrm{h}\mathcal{C}}(X[1], Y)$. (Here 0 and $0'$ are zero objects of $\mathcal{C}$.) Then the transposed diagram

$$\begin{tikzcd} X \arrow[r, "f'"] \arrow[d, "f"] & 0' \arrow[d] \\ 0 \arrow[r] & Y \end{tikzcd}$$

classifies the morphism $-\theta \in \mathrm{Hom}_{\mathrm{h}\mathcal{C}}(X[1], Y)$. Here $-\theta$ denotes the inverse of θ with respect to the group structure on $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(X[1], Y) \simeq \pi_1 \mathrm{Map}_{\mathcal{C}}(X, Y)$.

Proof. Without loss of generality, we may suppose that $0 = 0'$ and $f = f'$. Let $\sigma : \Lambda^2_0 \to \mathcal{C}$ be the diagram

$$0 \overset{f}{\leftarrow} X \overset{f}{\rightarrow} 0.$$

For every diagram $p : K \to \mathcal{C}$, let $\mathcal{D}(p)$ denote the Kan complex $\mathcal{C}_{p/} \times_{\mathcal{C}} \{Y\}$. Then $\operatorname{Hom}_{\mathrm{h}\mathcal{C}}(X[1], Y) \simeq \pi_0 \mathcal{D}(\sigma)$. We note that

$$\mathcal{D}(\sigma) \simeq \mathcal{D}(f) \times_{\mathcal{D}(X)} \mathcal{D}(f).$$

Since 0 is an initial object of $\mathcal{C}$, $\mathcal{D}(f)$ is contractible. In particular, there exists a point $q \in \mathcal{D}(f)$. Let

$$\mathcal{D}' = \mathcal{D}(f) \times_{\operatorname{Fun}(\{0\}, \mathcal{D}(X))} \operatorname{Fun}(\Delta^1, \mathcal{D}(X)) \times_{\operatorname{Fun}(\{1\}, \mathcal{D}(X))} \mathcal{D}(f)$$

$$\mathcal{D}'' = \{q\} \times_{\operatorname{Fun}(\{0\}, \mathcal{D}(X))} \operatorname{Fun}(\Delta^1, \mathcal{D}(X)) \times_{\operatorname{Fun}(\{1\}, \mathcal{D}(X))} \{q\}$$

so that we have canonical inclusions

$$\mathcal{D}'' \hookrightarrow \mathcal{D}' \hookleftarrow \mathcal{D}(\sigma).$$

The left map is a homotopy equivalence because $\mathcal{D}(f)$ is contractible, and the right map is a homotopy equivalence because the projection $\mathcal{D}(f) \to \mathcal{D}(X)$ is a Kan fibration. We observe that $\mathcal{D}''$ can be identified with the simplicial loop space of $\operatorname{Hom}^{\mathrm{L}}_{\mathcal{C}}(X, Y)$ (taken with the base point determined by q, which we can identify with the zero map from X to Y). Each of the Kan complexes $\mathcal{D}(\sigma)$, $\mathcal{D}'$, $\mathcal{D}''$ is equipped with a canonical involution. On $\mathcal{D}(\sigma)$, this involution corresponds to the transposition of diagrams as in the statement of the lemma. On $\mathcal{D}''$, this involution corresponds to reversal of loops. The desired conclusion now follows from the observation that these involutions are compatible with the inclusions $\mathcal{D}'', \mathcal{D}(\sigma) \subseteq \mathcal{D}'$. □

Definition 3.8. Let $\mathcal{C}$ be a pointed ∞-category which admits cokernels. Suppose given a diagram

$$X \overset{f}{\rightarrow} Y \overset{g}{\rightarrow} Z \overset{h}{\rightarrow} X[1]$$

in the homotopy category $\mathrm{h}\mathcal{C}$. We will say that this diagram is a *distinguished triangle* if there exists a diagram $\Delta^1 \times \Delta^2 \to \mathcal{C}$ as shown

$$\begin{tikzcd}
X \arrow[r, "\widetilde{f}"] \arrow[d] & Y \arrow[r] \arrow[d, "\widetilde{g}"] & 0 \arrow[d] \\
0' \arrow[r] & Z \arrow[r, "\widetilde{h}"] & W,
\end{tikzcd}$$

satisfying the following conditions:

(*i*) The objects $0, 0' \in \mathcal{C}$ are zero.

(*ii*) Both squares are pushout diagrams in $\mathcal{C}$.

(*iii*) The morphisms $\widetilde{f}$ and $\widetilde{g}$ represent f and g, respectively.

(*iv*) The map $h : Z \to X[1]$ is the composition of (the homotopy class of) $\widetilde{h}$ with the isomorphism $W \simeq X[1]$ determined by the outer rectangle.

Remark 3.9. We will generally only use Definition 3.8 in the case where $\mathcal{C}$ is a stable ∞-category. However, it will be convenient to have the terminology available in the case where $\mathcal{C}$ is not yet known to be stable.

The following result is an immediate consequence of Lemma 3.7:

Lemma 3.10. *Let $\mathcal{C}$ be a stable ∞-category. Suppose given a diagram $\Delta^2 \times \Delta^1 \to \mathcal{C}$, depicted as*

$$\begin{tikzcd}
X \arrow[r] \arrow[d, "f"] & 0 \arrow[d] \\
Y \arrow[r, "g"] \arrow[d] & Z \arrow[d, "h"] \\
0' \arrow[r] & W,
\end{tikzcd}$$

where both squares are pushouts and the objects $0, 0' \in \mathcal{C}$ are zero. Then the diagram

$$X \xrightarrow{f} Y \xrightarrow{g} Z \xrightarrow{-h'} X[1]$$

is a distinguished triangle in $\mathrm{h}\mathcal{C}$, where h' denotes the composition of h with the isomorphism $W \simeq X[1]$ determined by the outer square, and $-h'$ denotes the composition of h' with the map $-\mathrm{id} \in \mathrm{Hom}_{\mathrm{h}\mathcal{C}}(X[1], X[1]) \simeq \pi_1 \mathrm{Map}_{\mathcal{C}}(X, X[1])$.

We can now state the main result of this section:

Theorem 3.11. *Let $\mathcal{C}$ be a pointed ∞-category which admits cokernels, and suppose that the suspension functor Σ is an equivalence. Then the translation functor of Notation 3.4 and the class of distinguished triangles of Definition 3.8 endow $\mathrm{h}\mathcal{C}$ with the structure of a triangulated category.*

Remark 3.12. The hypotheses of Theorem 3.11 hold whenever $\mathcal{C}$ is stable. In fact, the hypotheses of Theorem 3.11 are *equivalent* to the stability of $\mathcal{C}$: see Corollary 8.28.

Proof. We must verify that Verdier's axioms $(TR1)$ through $(TR4)$ are satisfied.

$(TR1)$ Let $\mathcal{E} \subseteq \mathrm{Fun}(\Delta^1 \times \Delta^2, \mathcal{C})$ be the full subcategory spanned by those diagrams

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d] & Y \arrow[r] \arrow[d] & 0 \arrow[d] \\
0' \arrow[r] & Z \arrow[r] & W
\end{tikzcd}$$

of the form considered in Definition 3.8, and let $e : \mathcal{E} \to \mathrm{Fun}(\Delta^1, \mathcal{C})$ be the restriction to the upper left horizontal arrow. Repeated use of Proposition T.4.3.2.15 implies e is a trivial fibration. In particular, every morphism $f : X \to Y$ can be completed to a diagram belonging to $\mathcal{E}$. This proves (a). Part (b) is obvious, and (c) follows from the observation that if $f = \mathrm{id}_X$, then the object Z in the above diagram is a zero object of $\mathcal{C}$.

$(TR2)$ Suppose that

$$X \xrightarrow{f} Y \xrightarrow{g} Z \xrightarrow{h} X[1]$$

is a distinguished triangle in $\mathrm{h}\mathcal{C}$, corresponding to a diagram $\sigma \in \mathcal{E}$ as depicted above. Extend σ to a diagram

$$\begin{tikzcd}
X \arrow[r] \arrow[d] & Y \arrow[r] \arrow[d] & 0 \arrow[d] \\
0' \arrow[r] \arrow[d] & Z \arrow[r] \arrow[d] & W \arrow[d, "u"] \\
0'' \arrow[r] & 0''' \arrow[r] & V
\end{tikzcd}$$

where the lower right square is a pushout, and the objects $0'', 0''' \in \mathcal{C}$ are zero. We have a map between the squares

$$\begin{tikzcd} X \arrow[r] \arrow[d] & 0 \arrow[d] & Y \arrow[r] \arrow[d] & 0 \arrow[d] \\ 0' \arrow[r] & W & 0''' \arrow[r] & V \end{tikzcd}$$

which induces a commutative diagram in the homotopy category $\mathrm{h}\mathcal{C}$

$$\begin{tikzcd} W \arrow[r] \arrow[d, "u"] & X[1] \arrow[d, "f[1]"] \\ V \arrow[r] & Y[1] \end{tikzcd}$$

where the horizontal arrows are isomorphisms. Applying Lemma 3.10 to the rectangle on the right of the large diagram, we conclude that

$$Y \xrightarrow{g} Z \xrightarrow{h} X[1] \xrightarrow{-f[1]} Y[1]$$

is a distinguished triangle in $\mathrm{h}\mathcal{C}$.

Conversely, suppose that

$$Y \xrightarrow{g} Z \xrightarrow{h} X[1] \xrightarrow{-f[1]} Y[1]$$

is a distinguished triangle in $\mathrm{h}\mathcal{C}$. Since the functor $\Sigma : \mathcal{C} \to \mathcal{C}$ is an equivalence, we conclude that the triangle

$$Y[-2] \xrightarrow{g[-2]} Z[-2] \xrightarrow{h[-2]} X[-1] \xrightarrow{-f[-1]} Y[-1]$$

is distinguished. Applying the preceding argument five times, we conclude that the triangle

$$X \xrightarrow{f} Y \xrightarrow{g} Z \xrightarrow{h} X[1]$$

is distinguished, as desired.

$(TR3)$ Suppose distinguished triangles

$$X \xrightarrow{f} Y \to Z \to X[1]$$

$$X' \xrightarrow{f'} Y' \to Z' \to X'[1]$$

in $\mathrm{h}\mathcal{C}$. Without loss of generality, we may suppose that these triangles are induced by diagrams $\sigma, \sigma' \in \mathcal{E}$. Any commutative diagram

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d] & Y \arrow[d] \\ X' \arrow[r, "f'"] & Y' \end{tikzcd}$$

in the homotopy category $\mathrm{h}\mathcal{C}$ can be lifted (nonuniquely) to a square in $\mathcal{C}$, which we may identify with a morphism $\phi : e(\sigma) \to e(\sigma')$ in the ∞-category $\mathrm{Fun}(\Delta^1, \mathcal{C})$. Since e is a trivial fibration of simplicial sets, ϕ can be lifted to a morphism $\sigma \to \sigma'$ in $\mathcal{E}$, which determines a natural transformation of distinguished triangles

$$\begin{tikzcd} X \arrow[r] \arrow[d] & Y \arrow[r] \arrow[d] & Z \arrow[r] \arrow[d] & X[1] \arrow[d] \\ X' \arrow[r] & Y' \arrow[r] & Z' \arrow[r] & X'[1]. \end{tikzcd}$$

($TR4$) Let $f : X \to Y$ and $g : Y \to Z$ be morphisms in $\mathcal{C}$. In view of the fact that $e : \mathcal{E} \to \mathrm{Fun}(\Delta^1, \mathcal{C})$ is a trivial fibration, any distinguished triangle in $\mathrm{h}\mathcal{C}$ beginning with f, g, or $g \circ f$ is uniquely determined up to (nonunique) isomorphism. Consequently, it will suffice to prove that there exist *some* triple of distinguished triangles which satisfies the conclusions of ($TR4$). To prove this, we construct a diagram in $\mathcal{C}$

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d] & Y \arrow[r, "g"] \arrow[d] & Z \arrow[r] \arrow[d] & 0 \arrow[d] & \\
0 \arrow[r] & Y/X \arrow[r] \arrow[d] & Z/X \arrow[r] \arrow[d] & X' \arrow[r] \arrow[d] & 0 \arrow[d] \\
& 0 \arrow[r] & Z/Y \arrow[r] & Y' \arrow[r] & (Y/X)'
\end{tikzcd}$$

where 0 is a zero object of $\mathcal{C}$, and each square in the diagram is a pushout (more precisely, we apply Proposition T.4.3.2.15 repeatedly to construct a map from the nerve of the appropriate partially ordered set into $\mathcal{C}$). Restricting to appropriate rectangles contained in the diagram, we obtain isomorphisms $X' \simeq X[1]$, $Y' \simeq Y[1]$, $(Y/X)' \simeq Y/X[1]$, and four distinguished triangles

$$X \xrightarrow{f} Y \to Y/X \to X[1]$$

$$Y \xrightarrow{g} Z \to Z/Y \to Y[1]$$

$$X \xrightarrow{g \circ f} Z \to Z/X \to X[1]$$

$$Y/X \to Z/X \to Z/Y \to Y/X[1].$$

The commutativity in the homotopy category $\mathrm{h}\mathcal{C}$ required by ($TR4$) follows from the (stronger) commutativity of the above diagram in $\mathcal{C}$ itself.

□

Remark 3.13. The definition of a stable ∞-category is quite a bit simpler than that of a triangulated category. In particular, the octahedral axiom ($TR4$) is a consequence of ∞-categorical principles which are basic and easily motivated.

Notation 3.14. Let $\mathcal{C}$ be a stable ∞-category containing a pair of objects X and Y. We let $\mathrm{Ext}^n_{\mathcal{C}}(X, Y)$ denote the abelian group $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(X[n], Y)$. If n is negative, this can be identified with the homotopy group $\pi_{-n} \mathrm{Map}_{\mathcal{C}}(X, Y)$. More generally, $\mathrm{Ext}^n_{\mathcal{C}}(X, Y)$ can be identified with the $(-n)$th homotopy group of an appropriate *spectrum* of maps from X to Y.

4 Properties of Stable ∞-Categories

According to Definition 2.9, a pointed ∞-category $\mathcal{C}$ is stable if it admits certain pushout squares and certain pullback squares, which are required to coincide with one another. Our goal in this section is to prove that a stable ∞-category $\mathcal{C}$ admits *all* finite limits and colimits, and that the pushout squares in $\mathcal{C}$ coincide with the pullback squares in general (Proposition 4.4). To prove this, we will need the following easy observation (which is quite useful in its own right):

Proposition 4.1. *Let $\mathcal{C}$ be a stable ∞-category, and let K be a simplicial set. Then the ∞-category* $\mathrm{Fun}(K, \mathcal{C})$ *is stable.*

Proof. This follows immediately from the fact that kernels and cokernels in $\mathrm{Fun}(K, \mathcal{C})$ can be computed pointwise (Proposition T.5.1.2.2). □

Definition 4.2. If $\mathcal{C}$ is stable ∞-category, and $\mathcal{C}_0$ is a full subcategory containing a zero object and stable under the formation of kernels and cokernels, then $\mathcal{C}_0$ is itself stable. In this case, we will say that $\mathcal{C}_0$ is a *stable subcategory* of $\mathcal{C}$.

Lemma 4.3. *Let $\mathcal{C}$ be a stable ∞-category, and let $\mathcal{C}' \subseteq \mathcal{C}$ be a full subcategory which is stable under cokernels and under translation. Then $\mathcal{C}'$ is a stable subcategory of $\mathcal{C}$.*

Proof. It will suffice to show that $\mathcal{C}'$ is stable under kernels. Let $f : X \to Y$ be a morphism in $\mathcal{C}$. Theorem 3.11 shows that there is a canonical equivalence $\ker(f) \simeq \operatorname{coker}(f)[-1]$. □

Proposition 4.4. *Let $\mathcal{C}$ be a pointed ∞-category. Then $\mathcal{C}$ is stable if and only if the following conditions are satisfied:*

(1) *The ∞-category $\mathcal{C}$ admits finite limits and colimits.*

(2) *A square*

$$\begin{tikzcd} X \arrow[r] \arrow[d] & Y \arrow[d] \\ X' \arrow[r] & Y' \end{tikzcd}$$

in $\mathcal{C}$ is a pushout if and only if it is a pullback.

Proof. Condition (1) implies the existence of kernels and cokernels in $\mathcal{C}$, and condition (2) implies that the exact triangles coincide with the coexact triangles. This proves the "if" direction.

Suppose now that $\mathcal{C}$ is stable. We begin by proving (1). It will suffice to show that $\mathcal{C}$ admits finite colimits; the dual argument will show that $\mathcal{C}$ admits finite limits as well. According to Proposition T.4.4.3.2, it will suffice to show that $\mathcal{C}$ admits coequalizers and finite coproducts. The existence of finite coproducts was established in Lemma 3.6. We now conclude by observing that a coequalizer for a diagram

$$\begin{tikzcd} X \arrow[r, shift left, "f"] \arrow[r, shift right, "f'"'] & Y \end{tikzcd}$$

can be identified with $\operatorname{coker}(f - f')$.

We now show that every pushout square in $\mathcal{C}$ is a pullback; the converse will follow by a dual argument. Let $\mathcal{D} \subseteq \operatorname{Fun}(\Delta^1 \times \Delta^1, \mathcal{C})$ be the full subcategory spanned by the pullback squares. Then $\mathcal{D}$ is stable under finite limits and under translations. It follows from Lemma 4.3 that $\mathcal{D}$ is a stable subcategory of $\operatorname{Fun}(\Delta^1 \times \Delta^1, \mathcal{C})$.

Let $i : \Lambda^2_0 \hookrightarrow \Delta^1 \times \Delta^1$ be the inclusion, and let $i_! : \operatorname{Fun}(\Lambda^2_0, \mathcal{C}) \to \operatorname{Fun}(\Delta^1 \times \Delta^1, \mathcal{C})$ be a functor of left Kan extension. Then $i_!$ preserves finite colimits, and is therefore exact (Proposition 5.1). Let $\mathcal{D}' = i_!^{-1} \mathcal{D}$. Then $\mathcal{D}'$ is a stable subcategory of $\operatorname{Fun}(\Lambda^2_0, \mathcal{C})$; we wish to show that $\mathcal{D}' = \operatorname{Fun}(\Lambda^2_0, \mathcal{C})$. To prove this, we observe that any diagram

$$X' \leftarrow X \rightarrow X''$$

can be obtained as a (finite) colimit

$$e'_{X'} \coprod_{e'_X} e_X \coprod_{e''_X} e''_{X''}$$

where $e_X \in \operatorname{Fun}(\Lambda^2_0, \mathcal{C})$ denotes the diagram $X \leftarrow X \rightarrow X$, $e'_Z \in \operatorname{Fun}(\Lambda^2_0, \mathcal{C})$ denotes the diagram $Z \leftarrow 0 \rightarrow 0$, and $e''_Z \in \operatorname{Fun}(\Lambda^2_0, \mathcal{C})$ denotes the diagram $0 \leftarrow 0 \rightarrow Z$. It will therefore suffice to prove that pushout of any of these five diagrams is also a pullback. This follows immediately from the following more general observation: any pushout square

$$\begin{tikzcd} A \arrow[r] \arrow[d, "f"] & A' \arrow[d] \\ B \arrow[r] & B' \end{tikzcd}$$

in an (arbitrary) ∞-category $\mathcal{C}$ is also pullback square, provided that f is an equivalence. □

Proposition 4.5. *Let $\mathcal{C}$ be a (small) stable ∞-category, and let κ be a regular cardinal. Then the ∞-category $\mathrm{Ind}_\kappa(\mathcal{C})$ is stable.*

Proof. The functor j preserves finite limits and colimits (Propositions T.5.1.3.2 and T.5.3.5.14). It follows that $j(0)$ is a zero object of $\mathrm{Ind}_\kappa(\mathcal{C})$, so that $\mathrm{Ind}_\kappa(\mathcal{C})$ is pointed.

We next show that every morphism $f : X \to Y$ in $\mathrm{Ind}_\kappa(\mathcal{C})$ admits a kernel and a cokernel. According to Proposition T.5.3.5.15, we may assume that f is a κ-filtered colimit of morphisms $f_\alpha : X_\alpha \to Y_\alpha$ which belong to the essential image $\mathcal{C}'$ of j. Since j preserves kernels and cokernels, each of the maps f_α has a kernel and a cokernel in Ind_κ. It follows immediately that f has a cokernel (which can be written as a colimit of the cokernels of the maps f_α). The existence of $\ker(f)$ is slightly more difficult. Choose a κ-filtered diagram $p : \mathcal{I} \to \mathrm{Fun}(\Delta^1 \times \Delta^1, \mathcal{C}')$, where each $p(\alpha)$ is a pullback square

$$\begin{tikzcd}
Z_\alpha \arrow[r] \arrow[d] & 0 \arrow[d] \\
X_\alpha \arrow[r, "f_\alpha"] & Y_\alpha .
\end{tikzcd}$$

Let σ be a colimit of the diagram p; we wish to show that σ is a pullback diagram in $\mathrm{Ind}_\kappa(\mathcal{C})$. Since $\mathrm{Ind}_\kappa(\mathcal{C})$ is stable under κ-small limits in $\mathcal{P}(\mathcal{C})$, it will suffice to show that σ is a pullback square in $\mathcal{P}(\mathcal{C})$. Since $\mathcal{P}(\mathcal{C})$ is an ∞-topos, filtered colimits in $\mathcal{P}(\mathcal{C})$ are left exact (Example T.7.3.4.7); it will therefore suffice to show that each $p(\alpha)$ is a pullback diagram in $\mathcal{P}(\mathcal{C})$. This is obvious, since the inclusion $\mathcal{C}' \subseteq \mathcal{P}(\mathcal{C})$ preserves all limits which exist in $\mathcal{C}'$ (Proposition T.5.1.3.2).

To complete the proof, we must show that a triangle in $\mathrm{Ind}_\kappa(\mathcal{C})$ is exact if and only if it is coexact. Suppose given an exact triangle

$$\begin{tikzcd}
Z \arrow[r] \arrow[d] & 0 \arrow[d] \\
X \arrow[r] & Y
\end{tikzcd}$$

in $\mathrm{Ind}_\kappa(\mathcal{C})$. The above argument shows that we can write this triangle as a filtered colimit of exact triangles

$$\begin{tikzcd}
Z_\alpha \arrow[r] \arrow[d] & 0 \arrow[d] \\
X_\alpha \arrow[r] & Y_\alpha
\end{tikzcd}$$

in $\mathcal{C}'$. Since $\mathcal{C}'$ is stable, we conclude that these triangles are also coexact. The original triangle is therefore a filtered colimit of coexact triangles in $\mathcal{C}'$, hence coexact. The converse follows by the same argument. □

5 Exact Functors

Let $F : \mathcal{C} \to \mathcal{C}'$ be a functor between stable ∞-categories. Suppose that F carries zero objects into zero objects. It follows immediately that F carries triangles into triangles. If, in addition, F carries exact triangles into exact triangles, then we will say that F is *exact*. The exactness of a functor F admits the following alternative characterizations:

Proposition 5.1. *Let $F : \mathcal{C} \to \mathcal{C}'$ be a functor between stable ∞-categories. The following conditions are equivalent:*

(1) *The functor F is left exact. That is, F commutes with finite limits.*

(2) *The functor F is right exact. That is, F commutes with finite colimits.*

(3) *The functor F is exact.*

Proof. We will prove that (2) ⇔ (3); the equivalence (1) ⇔ (3) will follow by a dual argument. The implication (2) ⇒ (3) is obvious. Conversely, suppose that F is exact. The proof of Proposition 4.4 shows that F preserves coequalizers, and the proof of Lemma 3.6 shows that F preserves finite coproducts. It follows that F preserves all finite colimits (see the proof of Proposition T.4.4.3.2). □

The identity functor from any stable ∞-category to itself is exact, and a composition of exact functors is exact. Consequently, there exists a subcategory $\mathcal{C}\mathrm{at}_{\infty}^{\mathrm{Ex}} \subseteq \mathcal{C}\mathrm{at}_{\infty}$ in which the objects are stable ∞-categories and the morphisms are the exact functors. Our next few results concern the stability properties of this subcategory.

Proposition 5.2. *Suppose given a homotopy Cartesian diagram of ∞-categories*

$$\begin{tikzcd}
\mathcal{C}' \arrow[r, "G'"] \arrow[d, "F'"] & \mathcal{C} \arrow[d, "F"] \\
\mathcal{D}' \arrow[r, "G"] & \mathcal{D}.
\end{tikzcd}$$

Suppose further that $\mathcal{C}$, $\mathcal{D}'$, and $\mathcal{D}$ are stable, and that the functors F and G are exact. Then:

(1) *The ∞-category $\mathcal{C}'$ is stable.*

(2) *The functors F' and G' are exact.*

(3) *If $\mathcal{E}$ is a stable ∞-category, then a functor $H : \mathcal{E} \to \mathcal{C}'$ is exact if and only if the functors $F' \circ H$ and $G' \circ H$ are exact.*

Proof. Combine Proposition 4.4 with Lemma T.5.4.5.5. □

Proposition 5.3. *Let $\{\mathcal{C}_\alpha\}_{\alpha \in A}$ be a collection of stable ∞-categories. Then the product*

$$\mathcal{C} = \prod_{\alpha \in A} \mathcal{C}_\alpha$$

is stable. Moreover, for any stable ∞-category $\mathcal{D}$, a functor $F : \mathcal{D} \to \mathcal{C}$ is exact if and only if each of the compositions

$$\mathcal{D} \xrightarrow{F} \mathcal{C} \to \mathcal{C}_\alpha$$

is an exact functor.

Proof. This follows immediately from the fact that limits and colimits in $\mathcal{C}$ are computed pointwise. □

Theorem 5.4. *The ∞-category $\mathcal{C}\mathrm{at}_{\infty}^{\mathrm{Ex}}$ admits small limits, and the inclusion*

$$\mathcal{C}\mathrm{at}_{\infty}^{\mathrm{Ex}} \subseteq \mathcal{C}\mathrm{at}_{\infty}$$

preserves small limits.

Proof. Using Propositions 5.2 and 5.3, one can repeat the argument used to prove Proposition T.5.4.7.3. □

We now prove an analogue of Theorem 5.4.

Proposition 5.5. *Let $p : X \to S$ be an inner fibration of simplicial sets. Suppose that:*

(i) *For each vertex s of S, the fiber $X_s = X \times_S \{s\}$ is a stable ∞-category.*

(ii) *For every edge $s \to s'$ in S, the restriction $X \times_S \Delta^1 \to \Delta^1$ is a coCartesian fibration, associated to an exact functor $X_s \to X_{s'}$.*

Then:

(1) *The ∞-category $\mathrm{Map}_S(S, X)$ of sections of p is stable.*

(2) *If $\mathcal{C}$ is an arbitrary stable ∞-category, and $f : \mathcal{C} \to \mathrm{Map}_S(S, X)$ induces an exact functor $\mathcal{C} \xrightarrow{f} \mathrm{Map}_S(S, X) \to X_s$ for every vertex s of S, then f is exact.*

(3) *For every set $\mathcal{E}$ of edges of S, let $Y(\mathcal{E}) \subseteq \mathrm{Map}_S(S, X)$ be the full subcategory spanned by those sections $f : S \to X$ of p with the following property:*

($*$) *For every $e \in \mathcal{E}$, f carries e to a p_e-coCartesian edge of the fiber product $X \times_S \Delta^1$, where $p_e : X \times_S \Delta^1 \to \Delta^1$ denotes the projection.*

Then each $Y(\mathcal{E})$ is a stable subcategory of $\mathrm{Map}_S(S, X)$.

Proof. Combine Proposition T.5.4.7.11, Theorem 5.4, and Proposition 4.1. □

Proposition 5.6. *The ∞-category $\mathcal{C}\mathrm{at}_\infty^{\mathrm{Ex}}$ admits small filtered colimits, and the inclusion $\mathcal{C}\mathrm{at}_\infty^{\mathrm{Ex}} \subseteq \mathcal{C}\mathrm{at}_\infty$ preserves filtered colimits.*

Proof. Let $\mathcal{I}$ be a filtered ∞-category, $p : \mathcal{I} \to \mathcal{C}\mathrm{at}_\infty^{\mathrm{Ex}}$ a diagram, which we will indicate by $\{\mathcal{C}_I\}_{I \in \mathcal{I}}$, and $\mathcal{C}$ a colimit of the induced diagram $\mathcal{I} \to \mathcal{C}\mathrm{at}_\infty$. We must prove:

(i) The ∞-category $\mathcal{C}$ is stable.

(ii) Each of the canonical functors $\theta_I : \mathcal{C}_I \to \mathcal{C}$ is exact.

(iii) Given an arbitrary stable ∞-category $\mathcal{D}$, a functor $f : \mathcal{C} \to \mathcal{D}$ is exact if and only if each of the composite functors $\mathcal{C}_I \xrightarrow{\theta_I} \mathcal{C} \to \mathcal{D}$ is exact.

In view of Proposition 5.1, (ii) and (iii) follow immediately from Proposition T.5.5.7.11. The same result implies that $\mathcal{C}$ admits finite limits and colimits, and that each of the functors θ_I preserves finite limits and colimits.

To prove that $\mathcal{C}$ has a zero object, we select an object $I \in \mathcal{I}$. The functor $\mathcal{I} \to \mathcal{C}$ preserves initial and final objects. Since $\mathcal{C}_I$ has a zero object, so does $\mathcal{C}$.

We will complete the proof by showing that every exact triangle in $\mathcal{C}$ is coexact (the converse follows by the same argument). Fix a morphism $f : X \to Y$ in $\mathcal{C}$. Without loss of generality, we may suppose that there exists $I \in \mathcal{I}$ and a morphism $\widetilde{f} : \widetilde{X} \to \widetilde{Y}$ in $\mathcal{C}_I$ such that $f = \theta_I(\widetilde{f})$ (Proposition T.5.4.1.2). Form a pullback diagram $\widetilde{\sigma}$

$$\begin{tikzcd}
\widetilde{W} \arrow[r] \arrow[d] & \widetilde{X} \arrow[d] \\
0 \arrow[r] & \widetilde{Y}
\end{tikzcd}$$

in $\mathcal{C}_I$. Since $\mathcal{C}_I$ is stable, this diagram is also a pushout. It follows that $\theta_I(\sigma)$ is triangle $W \to X \xrightarrow{f} Y$ which is both exact and coexact in $\mathcal{C}$. □

6 t-Structures and Localizations

Let $\mathcal{C}$ be an ∞-category. Recall that we say a full subcategory $\mathcal{C}' \subseteq \mathcal{C}$ is a *localization* of $\mathcal{C}$ if the inclusion functor $\mathcal{C}' \subseteq \mathcal{C}$ has a left adjoint (§T.5.2.7). In this section, we will introduce a special class of localizations, called *t-localizations*, in the case where $\mathcal{C}$ is stable. We will further show that there is a bijective correspondence between t-localizations of $\mathcal{C}$ and *t-structures* on the triangulated category $\mathrm{h}\mathcal{C}$. We begin with a review of the classical theory of t-structures; for a more thorough introduction we refer the reader to [6].

Definition 6.1. Let $\mathcal{D}$ be a triangulated category. A *t-structure* on $\mathcal{D}$ is defined to be a pair of full subcategories $\mathcal{D}_{\geq 0}$, $\mathcal{D}_{\leq 0}$ (always assumed to be stable under isomorphism) having the following properties:

(1) For $X \in \mathcal{D}_{\geq 0}$ and $Y \in \mathcal{D}_{\leq -1}$, we have $\mathrm{Hom}_{\mathcal{D}}(X, Y) = 0$.

(2) $\mathcal{D}_{\geq 0}[1] \subseteq \mathcal{D}_{\geq 0}$, $\mathcal{D}_{\leq 0}[-1] \subseteq \mathcal{D}_{\leq 0}$.

(3) For any $X \in \mathcal{D}$, there exists a distinguished triangle $X' \rightarrow X \rightarrow X'' \rightarrow X'[1]$ where $X' \in \mathcal{D}_{\geq 0}$ and $X'' \in \mathcal{D}_{\leq 0}[-1]$.

Notation 6.2. If $\mathcal{D}$ is a triangulated category equipped with a t-structure, we will write $\mathcal{D}_{\geq n}$ for $\mathcal{D}_{\geq 0}[n]$ and $\mathcal{D}_{\leq n}$ for $\mathcal{D}_{\leq 0}[n]$. Observe that we use a *homological* indexing convention.

Remark 6.3. In Definition 6.1, either of the full subcategories $\mathcal{D}_{\geq 0}, \mathcal{D}_{\leq 0} \subseteq \mathcal{C}$ determines the other. For example, an object $X \in \mathcal{D}$ belongs to $\mathcal{D}_{\leq -1}$ if and only if $\mathrm{Hom}_{\mathcal{D}}(Y, X)$ vanishes for all $Y \in \mathcal{D}_{\geq 0}$.

Definition 6.4. Let $\mathcal{C}$ be a stable ∞-category. A *t-structure* on $\mathcal{C}$ is a t-structure on the homotopy category $\mathrm{h}\mathcal{C}$. If $\mathcal{C}$ is equipped with a t-structure, we let $\mathcal{C}_{\geq n}$ and $\mathcal{C}_{\leq n}$ denote the full subcategories of $\mathcal{C}$ spanned by those objects which belong to $(\mathrm{h}\mathcal{C})_{\geq \mathrm{n}}$ and $(\mathrm{h}\mathcal{C})_{\leq \mathrm{n}}$, respectively.

Proposition 6.5. *Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure. For each $n \in \mathbf{Z}$, the full subcategory $\mathcal{C}_{\leq n}$ is a localization of $\mathcal{C}$.*

Proof. Without loss of generality, we may suppose $n = -1$. According to Proposition T.5.2.7.8, it will suffice to prove that for each $X \in \mathcal{C}$, there exists a map $f : X \rightarrow X''$, where $X'' \in \mathcal{C}_{\leq -1}$ and for each $Y \in \mathcal{C}_{\leq -1}$, the map

$$\mathrm{Map}_{\mathcal{C}}(X'', Y) \rightarrow \mathrm{Map}_{\mathcal{C}}(X, Y)$$

is a weak homotopy equivalence. Invoking part (3) of Definition 6.1, we can choose f to fit into a distinguished triangle

$$X' \rightarrow X \xrightarrow{f} X'' \rightarrow X'[1]$$

where $X' \in \mathcal{C}_{\geq 0}$. According to Whitehead's theorem, we need to show that for every $k \leq 0$, the map

$$\mathrm{Ext}^k_{\mathcal{C}}(X'', Y) \rightarrow \mathrm{Ext}^k_{\mathcal{C}}(X, Y)$$

is an isomorphism of abelian groups. Using the long exact sequence associated to the exact triangle above, we are reduced to proving that the groups $\mathrm{Ext}^k_{\mathcal{C}}(X', Y)$ vanish for $k \leq 0$. We now use condition (2) of Definition 6.1 to conclude that $X'[-k] \in \mathcal{C}_{\geq 0}$. Condition (1) of Definition 6.1 now implies that

$$\mathrm{Ext}^k_{\mathcal{C}}(X', Y) \simeq \mathrm{Hom}_{\mathrm{h}\mathcal{C}}(X'[-k], Y) \simeq 0.$$

□

Corollary 6.6. *Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure. The full subcategories $\mathcal{C}_{\leq n} \subseteq \mathcal{C}$ are stable under all limits which exist in $\mathcal{C}$. Dually, the full subcategories $\mathcal{C}_{\geq 0} \subseteq \mathcal{C}$ are stable under all colimits which exist in $\mathcal{C}$.*

Notation 6.7. Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure. We will let $\tau_{\leq n}$ denote a left adjoint to the inclusion $\mathcal{C}_{\leq n} \subseteq \mathcal{C}$, and $\tau_{\geq n}$ a right adjoint to the inclusion $\mathcal{C}_{\geq n} \subseteq \mathcal{C}$.

Remark 6.8. Fix $n, m \in \mathbf{Z}$, and let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure. Then the truncation functors $\tau_{\leq n}$, $\tau_{\geq n}$ map the full subcategory $\mathcal{C}_{\leq m}$ to itself. To prove this, we first observe that $\tau_{\leq n}$ is equivalent to the identity on $\mathcal{C}_{\leq m}$ if $m \leq n$, while if $m \geq n$ the essential image of $\tau_{\leq n}$ is contained in $\mathcal{C}_{\leq n} \subseteq \mathcal{C}_{\leq m}$. To prove the analogous result for $\tau_{\geq n}$, we observe that the proof of Proposition 6.5 implies that for each X, we have a distinguished triangle

$$\tau_{\geq n} X \rightarrow X \xrightarrow{f} \tau_{\leq n-1} X \rightarrow (\tau_{\geq n} X)[1].$$

If $X \in \mathcal{C}_{\leq m}$, then $\tau_{\leq n-1} X$ also belongs to $\mathcal{C}_{\leq m}$, so that $\tau_{\geq n} X \simeq \ker(f)$ belongs to $\mathcal{C}_{\leq m}$ since $\mathcal{C}_{\leq m}$ is stable under limits.

Warning 6.9. In §T.5.5.6, we introduced for every ∞-category $\mathcal{C}$ a full subcategory $\tau_{\leq n}\,\mathcal{C}$ of *n-truncated objects* of $\mathcal{C}$. In that context, we used the symbol $\tau_{\leq n}$ to denote a left adjoint to the inclusion $\tau_{\leq n}\,\mathcal{C} \subseteq \mathcal{C}$. This is *not* compatible with Notation 6.7. In fact, if $\mathcal{C}$ is a stable ∞-category, then it has no nonzero truncated objects at all: if $X \in \mathcal{C}$ is nonzero, then the identity map from X to itself determines a nontrivial homotopy class in $\pi_n \operatorname{Map}_{\mathcal{C}}(X[-n], X)$, for all $n \geq 0$. Nevertheless, the two notations are consistent when restricted to $\mathcal{C}_{\geq 0}$, in view of the following fact:

- Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure. An object $X \in \mathcal{C}_{\geq 0}$ is k-truncated (as an object of $\mathcal{C}_{\geq 0}$) if and only if $X \in \mathcal{C}_{\leq k}$.

In fact, we have the following more general statement: for any $X \in \mathcal{C}$ and $k \geq -1$, X belongs to $\mathcal{C}_{\leq k}$ if and only if $\operatorname{Map}_{\mathcal{C}}(Y, X)$ is k-truncated for every $Y \in \mathcal{C}_{\geq 0}$. Because the latter condition is equivalent to the vanishing of $\operatorname{Ext}^n_{\mathcal{C}}(Y, X)$ for $n < -k$, we can use the shift functor to reduce to the case where $n = 0$ and $k = -1$, which is covered by Remark 6.3.

Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure, and let $n, m \in \mathbf{Z}$. Remark 6.8 implies that we have a commutative diagram of simplicial sets

$$\begin{tikzcd}
\mathcal{C}_{\geq n} \arrow[r, hook] \arrow[d, "\tau_{\leq m}"] & \mathcal{C} \arrow[d, "\tau_{\leq m}"] \\
\mathcal{C}_{\geq n} \cap \mathcal{C}_{\leq m} \arrow[r, hook] & \mathcal{C}_{\leq m} .
\end{tikzcd}$$

As explained in §T.7.3.1, we get an induced transformation of functors

$$\theta : \tau_{\leq m} \circ \tau_{\geq n} \rightarrow \tau_{\geq n} \circ \tau_{\leq m}.$$

Proposition 6.10. *Let $\mathcal{C}$ be a stable ∞-category equipped with t-structure. Then the natural transformation*

$$\theta : \tau_{\leq m} \circ \tau_{\geq n} \rightarrow \tau_{\geq n} \circ \tau_{\leq m}$$

is an equivalence of functors $\mathcal{C} \rightarrow \mathcal{C}_{\leq m} \cap \mathcal{C}_{\geq n}$.

Proof. This is a classical fact concerning triangulated categories; we include a proof for completeness. Fix $X \in \mathcal{C}$; we wish to show that

$$\theta(X) : \tau_{\leq m} \tau_{\geq n} X \rightarrow \tau_{\geq n} \tau_{\leq m} X$$

is an isomorphism in the homotopy category of $\mathcal{C}_{\leq m} \cap \mathcal{C}_{\geq n}$. If $m < n$, then both sides are zero and there is nothing to prove; let us therefore assume that $m \geq n$. Fix $Y \in \mathcal{C}_{\leq m} \cap \mathcal{C}_{\geq n}$; it will suffice to show that composition with $\theta(X)$ induces an isomorphism

$$\operatorname{Ext}^0(\tau_{\geq n} \tau_{\leq m} X, Y) \rightarrow \operatorname{Ext}^0(\tau_{\leq m} \tau_{\geq n} X, Y) \simeq \operatorname{Ext}^0(\tau_{\geq n} X, Y).$$

We have a map of long exact sequences

$$\begin{tikzcd}
\mathrm{Ext}^0(\tau_{\leq n-1}\tau_{\leq m}X, Y) \arrow[r, "f_0"] \arrow[d] & \mathrm{Ext}^0(\tau_{\leq n-1}X, Y) \arrow[d] \\
\mathrm{Ext}^0(\tau_{\leq m}X, Y) \arrow[r, "f_1"] \arrow[d] & \mathrm{Ext}^0(X, Y) \arrow[d] \\
\mathrm{Ext}^0(\tau_{\geq n}\tau_{\leq m}X, Y) \arrow[r, "f_2"] \arrow[d] & \mathrm{Ext}^0(\tau_{\geq n}X, Y) \arrow[d] \\
\mathrm{Ext}^1(\tau_{\leq n-1}\tau_{\leq m}X, Y) \arrow[r, "f_3"] \arrow[d] & \mathrm{Ext}^1(\tau_{\leq n-1}X, Y) \arrow[d] \\
\mathrm{Ext}^1(\tau_{\leq m}X, Y) \arrow[r, "f_4"] & \mathrm{Ext}^1(X, Y).
\end{tikzcd}$$

Since $m \geq n$, the natural transformation $\tau_{\leq n-1} \to \tau_{\leq n-1}\tau_{\leq m}$ is an equivalence of functors; this proves that f_0 and f_3 are bijective. Since $Y \in \mathcal{C}_{\leq m}$, f_1 is bijective and f_4 is injective. It follows from the "five lemma" that f_2 is bijective, as desired. □

Definition 6.11. Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure. The *heart* $\mathcal{C}^{\heartsuit}$ of $\mathcal{C}$ is the full subcategory $\mathcal{C}_{\geq 0} \cap \mathcal{C}_{\leq 0} \subseteq \mathcal{C}$. For each $n \in \mathbf{Z}$, we let $\pi_0 : \mathcal{C} \to \mathcal{C}^{\heartsuit}$ denote the functor $\tau_{\leq 0} \circ \tau_{\geq 0} \simeq \tau_{\geq 0} \circ \tau_{\leq 0}$, and we let $\pi_n : \mathcal{C} \to \mathcal{C}^{\heartsuit}$ denote the composition of π_0 with the shift functor $X \mapsto X[-n]$.

Remark 6.12. Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure, and let $X, Y \in \mathcal{C}^{\heartsuit}$. The homotopy group $\pi_n \operatorname{Map}_{\mathcal{C}}(X, Y) \simeq \operatorname{Ext}^{-n}_{\mathcal{C}}(X, Y)$ vanishes for $n > 0$. It follows that $\mathcal{C}^{\heartsuit}$ is equivalent to (the nerve of) its homotopy category $\mathrm{h}\mathcal{C}^{\heartsuit}$. Moreover, the category $\mathrm{h}\mathcal{C}^{\heartsuit}$ is abelian ([6]).

Let $\mathcal{C}$ be a stable ∞-category. In view of Remark 6.3, t-structures on $\mathcal{C}$ are determined by the corresponding localizations $\mathcal{C}_{\leq 0} \subseteq \mathcal{C}$. However, not every localization of $\mathcal{C}$ arises in this way. Recall (see §T.5.5.4) that every localization of $\mathcal{C}$ has the form $S^{-1}\,\mathcal{C}$, where S is an appropriate collection of morphisms of $\mathcal{C}$. Here $S^{-1}\,\mathcal{C}$ denotes the full subcategory of $\mathcal{C}$ spanned by S-local objects, where an object $X \in \mathcal{C}$ is said to be S-local if and only if, for each $f : Y' \to Y$ in S, composition with f induces a homotopy equivalence

$$\operatorname{Map}_{\mathcal{C}}(Y, X) \to \operatorname{Map}_{\mathcal{C}}(Y', X).$$

If $\mathcal{C}$ is stable, then we extend the morphism f to a distinguished triangle

$$Y' \to Y \to Y'' \to Y'[1],$$

and we have an associated long exact sequence

$$\ldots \to \operatorname{Ext}^i_{\mathcal{C}}(Y'', X) \to \operatorname{Ext}^i_{\mathcal{C}}(Y, X) \xrightarrow{\theta_i} \operatorname{Ext}^i_{\mathcal{C}}(Y', X) \to \operatorname{Ext}^{i+1}_{\mathcal{C}}(Y'', X) \to \ldots$$

The requirement that X be $\{f\}$-local amounts to the condition that θ_i be an isomorphism for $i \leq 0$. Using the long exact sequence, we see that if X is $\{f\}$-local, then $\operatorname{Ext}^i_{\mathcal{C}}(Y'', X) = 0$ for $i \leq 0$. Conversely, if $\operatorname{Ext}^i_{\mathcal{C}}(Y'', X) = 0$ for $i \leq 1$, then X is $\{f\}$-local. Experience suggests that it is usually more natural to require the vanishing of the groups $\operatorname{Ext}^i_{\mathcal{C}}(Y'', X)$ than it is to require that the maps θ_i to be isomorphisms. Of course, if Y' is a zero object of $\mathcal{C}$, then the distinction between these conditions disappears.

Definition 6.13. Let $\mathcal{C}$ be an ∞-category which admits pushouts. We will say that a collection S of morphisms of $\mathcal{C}$ is *quasisaturated* if it satisfies the following conditions:

(1) Every equivalence in $\mathcal{C}$ belongs to S.

(2) Given a 2-simplex $\Delta^2 \to \mathcal{C}$

$$\begin{tikzcd}
X \arrow[rr, "h"] \arrow[dr, "f"'] & & Z \\
& Y, \arrow[ur, "g"'] &
\end{tikzcd}$$

if any two of f, g, and h belongs to S, then so does the third.

(3) Given a pushout diagram

$$\begin{tikzcd}
X \arrow[r] \arrow[d, "f"] & X' \arrow[d, "f'"] \\
Y \arrow[r] & Y',
\end{tikzcd}$$

if $f \in S$, then $f' \in S$.

Any intersection of quasisaturated collections of morphisms is weakly saturated. Consequently, for any collection of morphisms S there is a smallest quasisaturated collection $\overline{S}$ containing S. We will say that $\overline{S}$ is the *quasisaturated collection of morphisms generated by S*.

Definition 6.14. Let $\mathcal{C}$ be a stable ∞-category. A full subcategory $\mathcal{C}' \subseteq \mathcal{C}$ is *closed under extensions* if, for every distinguished triangle

$$X \to Y \to Z \to X[1]$$

such that X and Z belong to $\mathcal{C}'$, the object Y also belongs to $\mathcal{C}'$.

We observe that if $\mathcal{C}$ is as in Definition 6.13 and $L : \mathcal{C} \to \mathcal{C}$ is a localization functor, then the collection of all morphisms f of $\mathcal{C}$ such that $L(f)$ is an equivalence is quasisaturated.

Proposition 6.15. *Let $\mathcal{C}$ be a stable ∞-category, let $L : \mathcal{C} \to \mathcal{C}$ be a localization functor, and let S be the collection of morphisms f in $\mathcal{C}$ such that $L(f)$ is an equivalence. The following conditions are equivalent:*

(1) *There exists a collection of morphisms $\{f : 0 \to X\}$ which generates S (as a quasisaturated collection of morphisms).*

(2) *The collection of morphisms $\{0 \to X : L(X) \simeq 0\}$ generates S (as a quasisaturated collection of morphisms).*

(3) *The essential image of L is closed under extensions.*

(4) *For any $A \in \mathcal{C}$, $B \in L\,\mathcal{C}$, the natural map $\mathrm{Ext}^1(LA, B) \to \mathrm{Ext}^1(A, B)$ is injective.*

(5) *The full subcategories $\mathcal{C}_{\geq 0} = \{A : LA \simeq 0\}$ and $\mathcal{C}_{\leq -1} = \{A : LA \simeq A\}$ determine a t-structure on $\mathcal{C}$.*

Proof. The implication (1) ⇒ (2) is obvious. We next prove that (2) ⇒ (3). Suppose given an exact triangle

$$X \to Y \to Z$$

where X and Z are both S-local. We wish to prove that Y is S-local. In view of assumption (2), it will suffice to show that $\mathrm{Map}_{\mathcal{C}}(A, Y)$ is contractible, provided that $L(A) \simeq 0$. In other words, we must show that $\mathrm{Ext}^i_{\mathcal{C}}(A, Y) \simeq 0$ for $i \leq 0$. We now observe that there is an exact sequence

$$\mathrm{Ext}^i_{\mathcal{C}}(A, X) \to \mathrm{Ext}^i_{\mathcal{C}}(A, Y) \to \mathrm{Ext}^i_{\mathcal{C}}(A, Z)$$

where the outer groups vanish, since X and Z are S-local and the map $0 \to A$ belongs to S.

We next show that (3) ⇒ (4). Let $B \in L\,\mathcal{C}$, and let $\eta \in \mathrm{Ext}^1_{\mathcal{C}}(LA, B)$ classify a distinguished triangle

$$B \to C \xrightarrow{g} LA \xrightarrow{\eta} B[1].$$

Condition (3) implies that $C \in L\,\mathcal{C}$. If the image of η in $\mathrm{Ext}^1_{\mathcal{C}}(A, B)$ is trivial, then the localization map $A \to LA$ factors as a composition

$$A \xrightarrow{f} C \xrightarrow{g} LA.$$

Applying L to this diagram (and using the fact that C is local) we conclude that the map g admits a section, so that $\eta = 0$.

We now claim that (4) ⇒ (5). Assume (4), and define $\mathcal{C}_{\geq 0}$, $\mathcal{C}_{\leq -1}$ as in (5). We will show that the axioms of Definition 6.1 are satisfied:

- If $X \in \mathcal{C}_{\geq 0}$ and $Y \in \mathcal{C}_{\leq -1}$, then $\mathrm{Ext}^0_{\mathcal{C}}(X, Y) \simeq \mathrm{Ext}^0_{\mathcal{C}}(LX, Y) \simeq \mathrm{Ext}^0_{\mathcal{C}}(0, Y) \simeq 0$.
- Since $\mathcal{C}_{\leq -1}$ is a localization of $\mathcal{C}$, it is stable under limits, so that $\mathcal{C}_{\leq -1}[-1] \subseteq \mathcal{C}_{\leq -1}$. Similarly, since the functor $L : \mathcal{C} \to \mathcal{C}_{\leq -1}$ preserves all colimits which exist in $\mathcal{C}$, the subcategory $\mathcal{C}_{\geq 0}$ is stable under finite colimits, so that $\mathcal{C}_{\geq 0}[1] \subseteq \mathcal{C}_{\geq 0}$.
- Let $X \in \mathcal{C}$, and form a distinguished triangle

  $$X' \to X \to LX \to X'[1].$$

  We claim that $X' \in \mathcal{C}_{\geq 0}$; in other words, that $LX' = 0$. For this, it suffices to show that for all $Y \in L\,\mathcal{C}$, the morphism space

  $$\mathrm{Ext}^0_{\mathcal{C}}(LX', Y) = 0.$$

  Since Y is local, we have isomorphisms

  $$\mathrm{Ext}^0_{\mathcal{C}}(LX', Y) \simeq \mathrm{Ext}^0_{\mathcal{C}}(X', Y) \simeq \mathrm{Ext}^1_{\mathcal{C}}(X'[1], Y).$$

  We now observe that there is a long exact sequence

  $$\mathrm{Ext}^0(LX, Y) \xrightarrow{f} \mathrm{Ext}^0(X, Y) \to \mathrm{Ext}^1_{\mathcal{C}}(X'[1], Y) \to \mathrm{Ext}^1_{\mathcal{C}}(LX, Y) \xrightarrow{f'} \mathrm{Ext}^1_{\mathcal{C}}(X, Y).$$

  Here f is bijective (since Y is local) and f' is injective (in virtue of assumption (4)).

We conclude by showing that (5) ⇒ (1). Let S' be the smallest quasisaturated collection of morphisms which contains the zero map $0 \to A$, for every $A \in \mathcal{C}_{\geq 0}$. We wish to prove that $S = S'$. For this, we choose an arbitrary morphism $u : X \to Y$ belonging to S. Then $Lu : LX \to LY$ is an equivalence, so we have a pushout diagram

$$\begin{tikzcd}
X' \arrow[r, "u'"] \arrow[d] & Y' \arrow[d] \\
X \arrow[r, "u"] & Y,
\end{tikzcd}$$

where X' and Y' are kernels of the respective localization maps $X \to LX$, $Y \to LY$. Consequently, it will suffice to prove that $u' \in S'$. Since $X', Y' \in \mathcal{C}_{\geq 0}$, this follows from the two-out-of-three property, applied to the diagram

$$\begin{tikzcd}
& X' \arrow[dr, "u'"] & \\
0 \arrow[ur] \arrow[rr] & & Y'.
\end{tikzcd}$$

□

Proposition 6.16. *Let $\mathcal{C}$ be a stable ∞-category equipped with a left-complete t-structure. Let $P \in \mathcal{C}_{\geq 0}$. The following conditions are equivalent:*

(1) *The object P is projective in $\mathcal{C}_{\geq 0}$.*

(2) *For every $Q \in \mathcal{C}_{\leq 0}$, the abelian group $\mathrm{Ext}^1_{\mathcal{C}}(P, Q)$ vanishes.*

(3) *Given a distinguished triangle*
$$N' \to N \to N'' \to N'[1],$$
where $N', N, N'' \in \mathcal{C}_{\geq 0}$, the induced map $\mathrm{Ext}^0_{\mathcal{C}}(P, N) \to \mathrm{Ext}^0_{\mathcal{C}}(P, N'')$ is surjective.

Proof. It follows from Lemma 14.9 that $\mathcal{C}_{\geq 0}$ admits geometric realizations for simplicial objects, so that condition (1) makes sense. We first show that (1) ⇒ (2). Let $f : \mathcal{C} \to \mathcal{S}$ be the functor corepresented by P. Let $M_\bullet$ be a Čech nerve for the morphism $0 \to Q[1]$, so that $M_n \simeq Q^n \in \mathcal{C}_{\geq 0}$. Then $Q[1]$ can be identified with the geometric realization $|M_\bullet|$. Since P is projective, $f(Q[1])$ is equivalent to the geometric realization $|f(M_\bullet)|$. We have a surjective map $* \simeq \pi_0 f(M_0) \to \pi_0 |f(M_\bullet)|$, so that $\pi_0 f(Q[1]) = \mathrm{Ext}^1_{\mathcal{C}}(P, Q) = 0$.

We now show that (2) ⇒ (1). Proposition 10.12 implies that f is homotopic to a composition
$$\mathcal{C} \xrightarrow{F} \mathrm{Sp} \xrightarrow{\Omega^\infty} \mathcal{S},$$
where F is an exact functor. Applying (2), we deduce that F is right t-exact (Definition 14.7). Lemma 14.9 implies that the induced map $\mathcal{C}_{\geq 0} \to \mathrm{Sp}^{\mathrm{conn}}$ preserves geometric realizations of simplicial objects. Applying Proposition 9.11, we conclude that $f | \mathcal{C}_{\geq 0}$ preserves geometric realizations as well.

The implication (2) ⇒ (3) follows immediately from the exactness of the sequence
$$\mathrm{Ext}^0_{\mathcal{C}}(P, N) \to \mathrm{Ext}^0_{\mathcal{C}}(P, N'') \to \mathrm{Ext}^1_{\mathcal{C}}(P, N').$$
Conversely, suppose that (3) is satisfied, and let $\eta \in \mathrm{Ext}^1_{\mathcal{C}}(P, Q)$. Then η classifies a distinguished triangle
$$Q \to Q' \xrightarrow{g} P \to Q[1].$$
Since $Q, P \in \mathcal{C}_{\geq 0}$, we have $Q' \in \mathcal{C}_{\geq 0}$ as well. Invoking (3), we deduce that g admits a section, so that $\eta = 0$. □

7 Boundedness and Completeness

Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure. We let $\mathcal{C}^+ = \bigcup \mathcal{C}_{\leq n} \subseteq \mathcal{C}$, $\mathcal{C}^- = \bigcup \mathcal{C}_{\geq -n}$, and $\mathcal{C}^b = \mathcal{C}^+ \cap \mathcal{C}^-$. It is easy to see that $\mathcal{C}^-$, $\mathcal{C}^+$, and $\mathcal{C}^b$ are stable subcategories of $\mathcal{C}$. We will say that $\mathcal{C}$ is *left bounded* if $\mathcal{C} = \mathcal{C}^+$, *right bounded* if $\mathcal{C} = \mathcal{C}^-$, and *bounded* if $\mathcal{C} = \mathcal{C}^b$.

At the other extreme, given a stable ∞-category $\mathcal{C}$ equipped with a t-structure, we define the *left completion* $\widehat{\mathcal{C}}$ of $\mathcal{C}$ to be homotopy limit of the tower
$$\ldots \to \mathcal{C}_{\leq 2} \xrightarrow{\tau_{\leq 1}} \mathcal{C}_{\leq 1} \xrightarrow{\tau_{\leq 0}} \mathcal{C}_{\leq 0} \xrightarrow{\tau_{\leq -1}} \ldots$$
Using the results of §T.3.3.3, we can obtain a very concrete description of this inverse limit: it is the full subcategory of $\mathrm{Fun}(\mathrm{N}(\mathbf{Z}), \mathcal{C})$ spanned by those functors $F : \mathrm{N}(\mathbf{Z}) \to \mathcal{C}$ with the following properties:

(1) For each $n \in \mathbf{Z}$, $F(n) \in \mathcal{C}_{\leq -n}$.

(2) For each $m \leq n \in \mathbf{Z}$, the associated map $F(m) \to F(n)$ induces an equivalence $\tau_{\leq -n} F(m) \to F(n)$.

We will denote this inverse limit by $\widehat{\mathcal{C}}$, and refer to it as the *left completion* of $\mathcal{C}$.

Proposition 7.1. *Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure. Then:*

(1) *The left completion $\widehat{\mathcal{C}}$ is also stable.*

(2) *Let $\widehat{\mathcal{C}}_{\leq 0}$ and $\widehat{\mathcal{C}}_{\geq 0}$ be the full subcategories of $\widehat{\mathcal{C}}$ spanned by those functors $F : \mathrm{N}(\mathbf{Z}) \to \mathcal{C}$ which factor through $\mathcal{C}_{\leq 0}$ and $\mathcal{C}_{\geq 0}$, respectively. Then these subcategories determine a t-structure on $\widehat{\mathcal{C}}$.*

(3) *There is a canonical functor $\mathcal{C} \to \widehat{\mathcal{C}}$. This functor is exact, and induces an equivalence $\mathcal{C}_{\leq 0} \to \widehat{\mathcal{C}}_{\leq 0}$.*

Proof. We observe that $\widehat{\mathcal{C}}$ can be identified with the homotopy inverse limit of the tower

$$\ldots \to \mathcal{C}_{\leq 0} \overset{\tau_{\leq 0}\Sigma}{\to} \mathcal{C}_{\leq 0} \overset{\tau_{\leq 0}\Sigma}{\to} \mathcal{C}_{\leq 0}\,.$$

In other words, $\widehat{\mathcal{C}}^{op} \simeq \mathrm{Sp}(\mathcal{C}^{op})$ (see §13). Assertion (1) now follows from Proposition 8.27.

We next prove (2). We begin by observing that, if we identify $\widehat{\mathcal{C}}$ with a full subcategory of $\mathrm{Fun}(\mathrm{N}(\mathbf{Z}), \mathcal{C})$, then the shift functors on $\widehat{\mathcal{C}}$ can be defined by the formula

$$(F[n])(m) = F(m-n)[n].$$

This proves immediately that $\widehat{\mathcal{C}}_{\geq 0}[1] \subseteq \widehat{\mathcal{C}}_{\geq 0}$ and $\widehat{\mathcal{C}}_{\leq 0}[-1] \subseteq \widehat{\mathcal{C}}_{\leq 0}$. Moreover, if $X \in \widehat{\mathcal{C}}_{\geq 0}$ and $Y \in \widehat{\mathcal{C}}_{\leq -1} = \widehat{\mathcal{C}}_{\leq 0}[-1]$, then $\mathrm{Map}_{\widehat{\mathcal{C}}}(X, Y)$ can be identified with the homotopy limit of a tower of spaces

$$\ldots \to \mathrm{Map}_{\mathcal{C}}(X(n), Y(n)) \to \mathrm{Map}_{\mathcal{C}}(X(n-1), Y(n-1)) \to \ldots$$

Since each of these spaces is contractible, we conclude that $\mathrm{Map}_{\widehat{\mathcal{C}}}(X, Y) \simeq *$; in particular, $\mathrm{Ext}^0_{\widehat{\mathcal{C}}}(X, Y) = 0$. Finally, we consider an arbitrary $X \in \widehat{\mathcal{C}}$. Let $X'' = \tau_{\leq -1} \circ X : \mathrm{N}(\mathbf{Z}) \to \mathcal{C}$, and let $u : X \to X''$ be the induced map. It is easy to check that $X'' \in \widehat{\mathcal{C}}_{\leq -1}$ and that $\ker(u) \in \widehat{\mathcal{C}}_{\geq 0}$. This completes the proof of (2).

To prove (3), we let $\mathcal{D}$ denote the full subcategory of $\mathrm{N}(\mathbf{Z}) \times \mathcal{C}$ spanned by pairs (n, C) such that $C \in \mathcal{C}_{\leq -n}$. Using Proposition T.5.2.7.8, we deduce that the inclusion $\mathcal{D} \subseteq \mathrm{N}(\mathbf{Z}) \times \mathcal{C}$ admits a left adjoint L. The composition

$$\mathrm{N}(\mathbf{Z}) \times \mathcal{C} \overset{L}{\to} \mathcal{D} \subseteq \mathrm{N}(\mathbf{Z}) \times \mathcal{C}) \to \mathcal{C}$$

can be identified with a functor $\theta : \mathcal{C} \to \mathrm{Fun}(\mathrm{N}(\mathbf{Z}), \mathcal{C})$ which factors through $\widehat{\mathcal{C}}$. To prove that θ is exact, it suffices to show that θ is right exact (Proposition 5.1). Since the truncation functors $\tau_{\leq n} : \mathcal{C}_{\leq n+1} \to \mathcal{C}_{\leq n}$ are right exact, finite colimits in $\widehat{\mathcal{C}}$ are computed pointwise. Consequently, it suffices to prove that each of compositions

$$\mathcal{C} \overset{\theta}{\to} \widehat{\mathcal{C}} \to \tau_{\leq n}\, \mathcal{C}$$

is right exact. But this composition can be identified with the functor $\tau_{\leq n}$.

Finally, we observe that $\widehat{\mathcal{C}}_{\leq 0}$ can be identified with a homotopy limit of the essentially constant tower

$$\ldots \mathcal{C}_{\leq 0} \overset{\mathrm{id}}{\to} \mathcal{C}_{\leq 0} \overset{\mathrm{id}}{\to} \mathcal{C}_{\leq 0} \overset{\tau_{\leq -1}}{\to} \mathcal{C}_{\leq -1} \to \ldots,$$

and that θ induces an identification of this homotopy limit with $\mathcal{C}_{\leq 0}$. □

If $\mathcal{C}$ is a stable ∞-category equipped with a t-structure, then we will say that $\mathcal{C}$ is *left complete* if the functor $\mathcal{C} \to \widehat{\mathcal{C}}$ described in Proposition 7.1 is an equivalence.

Remark 7.2. Let $\mathcal{C}$ be as in Proposition 7.1. Then the inclusion $\mathcal{C}^{+} \subseteq \mathcal{C}$ induces an equivalence $\widehat{\mathcal{C}}^{+} \to \widehat{\mathcal{C}}$, and the functor $\mathcal{C} \to \widehat{\mathcal{C}}$ induces an equivalence $\mathcal{C}^{+} \to \widehat{\mathcal{C}}^{+}$. Consequently, the constructions

$$\mathcal{C} \mapsto \widehat{\mathcal{C}}$$

$$\mathcal{C} \mapsto \mathcal{C}^{+}$$

furnish an equivalence between the theory of left bounded stable ∞-categories and the theory of left complete stable ∞-categories.

We conclude this section with a useful criterion for establishing left completeness.

Proposition 7.3. *Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure. Suppose that $\mathcal{C}$ admits countable products, and that $\mathcal{C}_{\geq 0}$ is stable under countable products. The following conditions are equivalent:*

(1) *The ∞-category $\mathcal{C}$ is left complete.*

(2) *The full subcategory $\mathcal{C}_{\geq \infty} = \bigcap \mathcal{C}_{\geq n} \subseteq \mathcal{C}$ consists only of zero objects of $\mathcal{C}$.*

Proof. We first observe every tower of objects

$$\ldots \rightarrow X_n \rightarrow X_{n-1} \rightarrow \ldots$$

in $\mathcal{C}$ admits a limit $\varprojlim \{X_n\}$: we can compute this limit as the kernel of an appropriate map

$$\prod X_n \rightarrow \prod X_n.$$

Moreover, if each X_n belongs to $\mathcal{C}_{\geq 0}$, then $\varprojlim \{X_n\}$ belongs to $\mathcal{C}_{\geq -1}$.

The functor $F : \mathcal{C} \rightarrow \widehat{\mathcal{C}}$ of Proposition 7.1 admits a right adjoint G, given by

$$f \in \widehat{\mathcal{C}} \subseteq \mathrm{Fun}(\mathrm{N}(\mathbf{Z}), \mathcal{C}) \mapsto \varprojlim(f).$$

Assertion (1) is equivalent to the statement that the unit and counit maps

$$u : F \circ G \rightarrow \mathrm{id}_{\widehat{\mathcal{C}}}$$

$$v : \mathrm{id}_{\mathcal{C}} \rightarrow G \circ F$$

are equivalences. If v is an equivalence, then any object $X \in \mathcal{C}$ can be recovered as the limit of the tower $\{\tau_{\leq n} X\}$. In particular, this implies that $X = 0$ if $X \in \mathcal{C}_{\geq \infty}$, so that (1) $\Rightarrow$ (2).

Now assume (2); we will prove that u and v are both equivalences. To prove that u is an equivalence, we must show that for every $f \in \widehat{\mathcal{C}}$, the natural map

$$\theta : \varprojlim(f) \rightarrow f(n)$$

induces an equivalence $\tau_{\leq -n} \varprojlim(f) \rightarrow f(n)$. In other words, we must show that the kernel of θ belongs to $\mathcal{C}_{\geq -n+1}$. To prove this, we first observe that θ factors as a composition

$$\varprojlim(f) \overset{\theta'}{\rightarrow} f(n-1) \overset{\theta''}{\rightarrow} f(n).$$

The octahedral axiom ($(TR4)$ of Definition 3.1) implies the existence of an exact triangle

$$\ker(\theta') \rightarrow \ker(\theta) \rightarrow \ker(\theta'').$$

Since $\ker(\theta'')$ clearly belongs to $\mathcal{C}_{\geq -n+1}$, it will suffice to show that $\ker(\theta')$ belongs to $\mathcal{C}_{\geq -n+1}$. We observe that $\ker(\theta')$ can be identified with the limit of a tower $\{\ker(f(m) \rightarrow f(n-1))\}_{m<n}$. It therefore suffices to show that each $\ker(f(m) \rightarrow f(n-1))$ belongs to $\mathcal{C}_{\geq -n+2}$, which is clear.

We now show prove that v is an equivalence. Let X be an object of $\mathcal{C}$, and $v_X : X \rightarrow (G \circ F)(X)$ the associated map. Since u is an equivalence of functors, we conclude that $\tau_{\leq n}(v_X)$ is an equivalence for all $n \in \mathbf{Z}$. It follows that $\mathrm{coker}(v_X) \in \mathcal{C}_{\geq n+1}$ for all $n \in \mathbf{Z}$. Invoking assumption (2), we conclude that $\mathrm{coker}(v_X) \simeq 0$, so that v_X is an equivalence as desired. □

Remark 7.4. The ideas introduced above can be dualized in an obvious way, so that we can speak of *right completions* and *right completeness* for a stable ∞-category equipped with a t-structure.

8 Stabilization

In this section, we will describe a method for constructing stable ∞-categories: namely, for any ∞-category $\mathcal{C}$ which admits finite limits, one can consider an ∞-category $\mathrm{Sp}(\mathcal{C})$ of *spectrum objects* of $\mathcal{C}$. In the case where $\mathcal{C}$ is the ∞-category of spaces, we recover classical stable homotopy theory, which we will discuss in §9.

Definition 8.1. Let $\mathcal{C}$ be an ∞-category. A *prespectrum object* of $\mathcal{C}$ is a functor $X : \mathrm{N}(\mathbf{Z} \times \mathbf{Z}) \to \mathcal{C}$ with the following property: for every pair of integers $i \neq j$, the value $X(i,j)$ is a zero object of $\mathcal{C}$. We let $\mathrm{PSp}(\mathcal{C})$ denote the full subcategory of $\mathrm{Fun}(\mathrm{N}(\mathbf{Z} \times \mathbf{Z}), \mathcal{C})$ spanned by the prespectrum objects of $\mathcal{C}$.

For every integer n, evaluation at $(n,n) \in \mathbf{Z} \times \mathbf{Z}$ induces a functor $\mathrm{PSp}(\mathcal{C}) \to \mathcal{C}$. We will refer to this functor as the *nth space functor* and denote it by $\Omega^{\infty - n}_{\mathcal{C}}$.

Remark 8.2. The partially ordered set $\mathbf{Z} \times \mathbf{Z}$ is isomorphic to its opposite, via the map $(i,j) \mapsto (-i,-j)$. Composing with this map, we obtain an equivalence

$$\mathrm{PSp}(\mathcal{C})^{op} \simeq \mathrm{PSp}(\mathcal{C}^{op}).$$

Remark 8.3. Let X be a prespectrum object of an ∞-category $\mathcal{C}$. Since the objects $X(i,j) \in \mathcal{C}$ are zero for $i \neq j$, it is customary to ignore them and instead emphasize the objects $X(n,n) \in \mathcal{C}$ lying along the diagonal. We will often denote $X(n,n) = \Omega^{\infty - n}_{\mathcal{C}} X$ by $X[n]$. For each $n \geq 0$, the diagram

$$\begin{tikzcd}
X(n,n) \arrow[r] \arrow[d] & X(n,n+1) \arrow[d] \\
X(n+1,n) \arrow[r] & X(n+1,n+1)
\end{tikzcd}$$

determines an (adjoint) pair of morphisms

$$\alpha : \Sigma_{\mathcal{C}} X[n] \to X[n+1] \qquad \beta : X[n] \to \Omega_{\mathcal{C}} X[n+1].$$

Definition 8.4. Let X be a prespectrum object of a pointed ∞-category $\mathcal{C}$, and n an integer. We will say that X is a *spectrum below* n if the canonical map $\beta : X[m-1] \to \Omega_{\mathcal{C}} X[m]$ is an equivalence for each $m \leq n$. We say that X is a *suspension prespectrum above* n if the canonical map $\alpha : \Sigma_{\mathcal{C}} X[m] \to X[m+1]$ is an equivalence for all $m \geq n$. We say that X is an *n-suspension prespectrum* if it is a suspension prespectrum above n and a spectrum below n. We say that X is a *spectrum object* if it is a spectrum object below n for all integers n. We let $\mathrm{Sp}(\mathcal{C})$ denote the full subcategory of $\mathrm{PSp}(\mathcal{C})$ spanned by the spectrum objects of $\mathcal{C}$.

If $\mathcal{C}$ is an arbitrary ∞-category, we let $\mathrm{Stab}(\mathcal{C}) = \mathrm{Sp}(\mathcal{C}_*)$. Here $\mathcal{C}_*$ denotes the ∞-category of pointed objects of $\mathcal{C}$. We will refer to $\mathrm{Stab}(\mathcal{C})$ as the *stabilization* of $\mathcal{C}$.

Remark 8.5. Suppose that $\mathcal{C}$ is a pointed ∞-category. Then the forgetful functor $\mathcal{C}_* \to \mathcal{C}$ is a trivial Kan fibration, which induces a trivial Kan fibration $\mathrm{Stab}(\mathcal{C}) \to \mathrm{Sp}(\mathcal{C})$.

Example 8.6. Let $\mathbf{Q}$ be the ring of rational numbers, let $\mathbf{A}$ be the category of simplicial commutative $\mathbf{Q}$-algebras, viewed as simplicial model category (see Proposition T.5.5.9.1), and let $\mathcal{C} = \mathrm{N}(\mathbf{A}^{\circ})$ be the underlying ∞-category. Suppose that R is a commutative $\mathbf{Q}$-algebra, regarded as an object of $\mathcal{C}$. Then $\mathrm{Stab}(\mathcal{C}_{/R})$ is a stable ∞-category, whose homotopy category is equivalent to the (unbounded) derived category of R-modules. The loop functor $\Omega^{\infty} : \mathrm{Stab}(\mathcal{C}_{/R}) \to \mathcal{C}_{/R}$ admits a left adjoint $\Sigma^{\infty} : \mathcal{C}_{/R} \to \mathrm{Stab}(\mathcal{C}_{/R})$ (Proposition 15.4). This left adjoint assigns to each morphism of commutative rings $S \xrightarrow{\phi} R$ an object $\Sigma^{\infty}(\phi) \in \mathrm{Stab}(\mathcal{C}_{/R})$, which can be identified with $L_S \otimes_S R$, where L_S denotes the (absolute) *cotangent complex* of S. We will discuss this example in greater detail in [44]; see also [59] for discussion.

Remark 8.7. Let $\mathcal{C}$ be a pointed presentable ∞-category. Using Lemmas T.5.5.4.17, T.5.5.4.18, and T.5.5.4.19, we deduce that $\mathrm{PSp}(\mathcal{C})$ and $\mathrm{Sp}(\mathcal{C})$ are accessible localizations of $\mathrm{Fun}(\mathrm{N}(\mathbf{Z} \times \mathbf{Z}), \mathcal{C})$. It follows that $\mathrm{PSp}(\mathcal{C})$ and $\mathrm{Sp}(\mathcal{C})$ are themselves presentable ∞-categories. Moreover, the inclusion $\mathrm{Sp}(\mathcal{C}) \subseteq \mathrm{PSp}(\mathcal{C})$ admits an accessible left adjoint $L_{\mathcal{C}}$, which we will refer to as the *spectrification functor*. We will give a more direct construction of $L_{\mathcal{C}}$ below in the case where $\mathcal{C}$ satisfies some mild hypotheses.

Remark 8.8. Suppose that $\mathcal{C}$ is a pointed ∞-category which admits finite limits and countable colimits, and that the loop functor $\Omega_{\mathcal{C}} : \mathcal{C} \rightarrow \mathcal{C}$ preserves sequential colimits. Then the collection of prespectrum objects of $\mathcal{C}$ which are spectra below n is closed under sequential colimits.

Remark 8.9. The hypotheses of Remark 8.8 are always satisfied in any of the following cases:

(1) The ∞-category $\mathcal{C}$ is pointed and compactly generated.

(2) The ∞-category $\mathcal{C}$ is an ∞-topos (Example T.7.3.4.7).

(3) The ∞-category $\mathcal{C}$ is stable and admits countable coproducts. In this case, Proposition T.4.4.3.2 guarantees that $\mathcal{C}$ admits all countable colimits, and the functor $\Omega_{\mathcal{C}}$ is an equivalence and therefore preserves all colimits which exist in $\mathcal{C}$.

In order to work effectively with prespectrum objects, it is convenient to introduce a bit of additional terminology.

Notation 8.10. For $-\infty \leq a \leq b \leq \infty$, we let $Q(a, b) = \{(i, j) \in \mathbf{Z} \times \mathbf{Z} : (i \neq j) \vee (a \leq i = j \leq b)\}$. If $\mathcal{C}$ is an ∞-category, we let $\mathrm{PSp}_a^b(\mathcal{C})$ denote the full subcategory of $\mathrm{Fun}(\mathrm{N}(Q(a, b)), \mathcal{C})$ spanned by those functors X such that $X(i, j)$ is a zero object of $\mathcal{C}$ for $i \neq j$.

Lemma 8.11. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite limits, and suppose given $\infty < a \leq b \leq \infty$. Let $X_0 \in \mathrm{PSp}_a^b(\mathcal{C})$. Then:*

(1) *There exists an object $X \in \mathrm{PSp}_{a-1}^b(\mathcal{C})$ which is a right Kan extension of X_0.*

(2) *An arbitrary object $X \in \mathrm{PSp}_{a-1}^b(\mathcal{C})$ which extends X_0 is a right Kan extension of X_0 if and only if the induced map $X[a-1] \rightarrow X[a]$.*

Proof. Note that $Q(a-1, b)$ is obtained from $Q(a, b)$ by adjoining a single additional object $(a-1, a-1)$. It now suffices to observe that the the inclusion of ∞-categories

$$\mathrm{N}(\{(a-1, a), (a, a), (a, a-1)\})^{op} \subseteq \mathrm{N}(\{(i, j) \in Q(a, b) | (a-1 \leq i, j)\})^{op}$$

is cofinal, which follows immediately from the criterion of Theorem T.4.1.3.1. □

Lemma 8.12. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite limits and suppose given $\infty < a \leq b \leq \infty$. Let $X_0 \in \mathrm{PSp}_a^b(\mathcal{C})$. Then:*

(1) *There exists an object $X \in \mathrm{PSp}_{-\infty}^b(\mathcal{C})$ which is a right Kan extension of X_0.*

(2) *An arbitrary object $X \in \mathrm{PSp}_{-\infty}^b(\mathcal{C})$ extending X_0 is a right Kan extension of X_0 if and only if X is a spectrum object below a.*

Proof. Combine Lemma 8.11 with Proposition T.4.3.2.8. □

Lemma 8.13. *Let $\mathcal{C}$ be a pointed ∞-category and n an integer. Then evaluation at (n, n) induces a trivial Kan fibration $\mathrm{PSp}_n^n(\mathcal{C}) \rightarrow \mathcal{C}$.*

Proof. Let $Q' = \{(i, j) \in \mathbf{Z} \times \mathbf{Z} : (i < j \leq n) \vee (j < i \leq n) \vee (i = j = n)\}$, and let $\mathcal{D} \subseteq \mathrm{Fun}(\mathrm{N}(Q'), \mathcal{C})$ denote the full subcategory spanned by those functors X such that $X(i, j)$ is a zero object of $\mathcal{C}$ for $i \neq j$. We observe the following:

(*a*) A functor $X : \mathrm{N}(Q') \to \mathcal{C}$ belongs to $\mathcal{D}$ if and only if X is a left Kan extension of $X|\{(n,n)\}$.

(*b*) A functor $X : \mathrm{N}(Q(n,n)) \to \mathcal{C}$ belongs to $\mathrm{PSp}^n_n(\mathcal{C})$ if and only if $X|\,\mathrm{N}(Q') \in \mathcal{D}$ and X is a right Kan extension of $X|\,\mathrm{N}(Q')$.

It now follows from Proposition T.4.3.2.15 that the restriction functors

$$\mathrm{PSp}^n_n(\mathcal{C}) \to \mathcal{D} \to \mathcal{C}$$

are trivial Kan fibrations, and the composition is given by evaluation at (n,n). □

We can now describe the stabilization $\mathrm{Sp}(\mathcal{C})$ of a pointed ∞-category $\mathcal{C}$ in more conceptual terms:

Proposition 8.14. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite limits. Then the ∞-category $\mathrm{Sp}(\mathcal{C})$ can be identified with the homotopy inverse limit of the tower*

$$\ldots \to \mathcal{C} \overset{\Omega_{\mathcal{C}}}{\to} \mathcal{C} \overset{\Omega_{\mathcal{C}}}{\to} \mathcal{C}\,.$$

Proof. For every nonnegative integer n, let $\mathcal{D}_n$ denote the full subcategory of $\mathrm{PSp}^n_{-\infty}(\mathcal{C})$ spanned by those functors X such that the diagram

$$\begin{tikzcd}
X(m,m) \arrow[r] \arrow[d] & X(m,m+1) \arrow[d] \\
X(m+1,m) \arrow[r] & X(m+1,m+1)
\end{tikzcd}$$

is a pullback square for each $m < n$. We note that Lemma 8.12 and Proposition T.4.3.2.15 imply that the composition

$$\mathcal{D}_n \subseteq \mathrm{PSp}^n_{-\infty}(\mathcal{C}) \to \mathrm{PSp}^n_n(\mathcal{C})$$

is a trivial Kan fibration. Combining this with Lemma 8.13, we deduce that evaluation at (n,n) induces a trivial Kan fibration $\psi_n : \mathcal{D}_n \to \mathcal{C}$. Let s_n denote a section to ψ_n. We observe that the composite functor

$$\mathcal{C} \overset{s_n}{\to} \mathcal{D}_n \to \mathcal{D}_{n-1} \overset{\psi_n}{\to} \mathcal{C}$$

can be identified with the loop functor $\Omega_{\mathcal{C}}$. It follows that the tower

$$\ldots \to \mathcal{C} \overset{\Omega_{\mathcal{C}}}{\to} \mathcal{C} \overset{\Omega_{\mathcal{C}}}{\to} \mathcal{C}\,.$$

is equivalent to the tower of restriction maps

$$\ldots \to \mathcal{D}_2 \to \mathcal{D}_1 \to \mathcal{D}_0\,.$$

This tower consists of categorical fibrations between ∞-categories, so its homotopy inverse limit coincides with the actual inverse limit $\varprojlim \{\mathcal{D}_n\}_{n \geq 0} \simeq \mathrm{Sp}(\mathcal{C})$. □

We now study the "spectrification functor" $\mathrm{PSp}(\mathcal{C}) \to \mathrm{Sp}(\mathcal{C})$ in the case where $\mathcal{C}$ is well-behaved.

Lemma 8.15. *Let $P = \{(i,j,k) \in \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} | (i \neq j) \vee (i = j \geq k)\}$. Let $X : \mathrm{N}(P) \to \mathcal{C}$ be a functor, where $\mathcal{C}$ is a pointed ∞-category which admits finite limits. Suppose that $X(i,j,k)$ is a zero object of $\mathcal{C}$ for all $i \neq j$. Then:*

(1) *Let $\overline{X} : \mathrm{N}(\mathbf{Z} \times \mathbf{Z} \times \mathbf{Z}) \to \mathcal{C}$ be an extension of X. The following conditions are equivalent:*

(*i*) *The functor $\overline{X}$ is a right Kan extension of X.*

(ii) *For each $k \geq 0$, the induced functor $\overline{X}|\,\mathrm{N}(\mathbf{Z}\times\mathbf{Z}\times\{k\})$ is a right Kan extension of $X|\,\mathrm{N}(Q(k,\infty)\times\{k\})$.*

(iii) *For each $k \geq 0$, the functor $\overline{X}|\,\mathrm{N}(\mathbf{Z}\times\mathbf{Z}\times\{k\})$ is a spectrum object below k.*

(2) *There exists a functor $\overline{X} : \mathrm{N}(\mathbf{Z}\times\mathbf{Z}\times\mathbf{Z}) \to \mathcal{C}$ satisfying the equivalent conditions of* (1).

Proof. Let $\overline{X}$ be as in the statement of (1). To prove the equivalence of conditions (i) and (ii), it will suffice to prove the following more precise claim: for every triple of nonnegative integers (i,j,k), the functor $\overline{X}$ is a right Kan extension of X at (i,j,k) if and only if $\overline{X}|\,\mathrm{N}(\mathbf{Z}\times\mathbf{Z}\times\{k\})$ is a right Kan extension of $X|\,\mathrm{N}(Q(k,\infty)\times\{k\})$ at (i,j,k). This follows from the observation that the inclusion

$$\mathrm{N}(\{(i',j',k)\in P|(i'\geq i)\wedge(j'\geq j)\})^{op} \subseteq \mathrm{N}(\{(i',j',k')\in P|(i'\geq i)\wedge(j'\geq j)\wedge(k'\geq k)\})^{op}$$

is cofinal (since it admits left adjoint, given by $(i',j',k')\mapsto(i',j',k)$). The equivalence of (ii) and (iii) follows from Lemma 8.12.

To prove (2), we define a sequence of subsets

$$P = P(0) \subseteq P(1) \subseteq P(2) \subseteq \ldots \subseteq \mathbf{Z}\times\mathbf{Z}\times\mathbf{Z}$$

by the formula $P(m) = \{(i,j,k)\in\mathbf{Z}\times\mathbf{Z}\times\mathbf{Z}|(i\neq j)\vee(i=j\geq k-m)\}$. Using Proposition T.4.3.2.8, we deduce that if $\overline{X}:\mathrm{N}(\mathbf{Z}\times\mathbf{Z}\times\mathbf{Z})\to\mathcal{C}$ is an extension of X, then $\overline{X}$ is a right Kan extension of X if and only if each restriction $\overline{X}|\,\mathrm{N}(P(m+1))$ is a right Kan extension of $\overline{X}|\,\mathrm{N}(P(m))$. Consequently, (2) is a consequence of the following more precise assertion:

($2'$) Every functor $Y_0 : \mathrm{N}(P(m))\to\mathcal{C}$ extending X admits a right Kan extension $Y : \mathrm{N}(P(m+1))\to\mathcal{C}$ satisfying ($1'$).

To prove these assertions, we note that every element of $P(m+1)$ which does not belong to $P(m)$ has the form $(n-1,n-1,n+m)$ for some integer n. It now suffices to observe that the inclusion $\mathrm{N}(P'_0)^{op}\subseteq\mathrm{N}(P')^{op}$ is cofinal, where

$$P' = \{(i,j,k)\in P(m) : (i,j\geq n-1)\wedge(k\geq n+m)\}$$

$$P'_0 = \{(n-1,n,n+m),(n,n-1,n+m),(n,n,n+m)\}\subseteq P'.$$

This follows immediately from the criterion of Theorem T.4.1.3.1. □

Corollary 8.16. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite limits. Then there exists a sequence of functors*

$$\mathrm{id}\to L_0\to L_1\to L_2\to\ldots$$

from $\mathrm{PSp}(\mathcal{C})$ to itself such that the following conditions are satisfied for every prespectrum object X of $\mathcal{C}$ and every $n\geq 0$:

(1) *The prespectrum $L_n(X)$ is a spectrum below n.*

(2) *The map $\alpha : X\to L_n(X)$ induces an equivalence $X[m]\to L_n(X)[m]$ for $m\geq n$.*

(3) *Suppose that X is a spectrum below n. Then the map $\alpha : X\to L_n(X)$ is an equivalence.*

(4) *Let Y be any prespectrum object of $\mathcal{C}$ which is a spectrum below n. Then composition with α induces a homotopy equivalence*

$$\mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(L_n(X),Y)\to\mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(X,Y).$$

Proof. Let P be defined as in Lemma 8.15, let $\mathcal{D}$ denote the full subcategory of $\mathrm{Fun}(\mathrm{N}(P), \mathcal{C})$ spanned by those functors F such that $F(i, j, k)$ is a zero object of $\mathcal{C}$ for $i \neq j$, and let $\mathcal{D}' \subseteq \mathrm{Fun}(\mathrm{N}(\mathbf{Z} \times \mathbf{Z} \times \mathbf{Z}), \mathcal{C})$ denote the full subcategory spanned by those objects F such that F is a right Kan extension of $F| \mathrm{N}(P) \in \mathcal{D}$. Using Lemma 8.15 and Proposition T.4.3.2.15, we deduce that the restriction map $\mathcal{D}' \rightarrow \mathcal{D}$ is a trivial Kan fibration. Let p denote a section of this restriction map.

Let $q : \mathrm{PSp}(\mathcal{C}) \rightarrow \mathcal{D}$ denote the functor induced by the map of partially ordered sets $P \rightarrow \mathbf{Z} \times \mathbf{Z}$ $(i, j, k) \mapsto (i, j)$. The composition $p \circ q : \mathrm{PSp}(\mathcal{C}) \rightarrow \mathcal{D}'$ determines a map $\mathrm{PSp}(\mathcal{C}) \times \mathrm{N}(\mathbf{Z}_{\geq 0}) \rightarrow \mathrm{PSp}(\mathcal{C})$, which we can identify with a sequence of functors $\{L_n\}_{n \in \mathbf{Z}}$ from $\mathrm{PSp}(\mathcal{C})$ to itself. By construction, we also have a canonical map $\mathrm{id} \rightarrow L_0$. Assertions (1) and (2) are immediate consequences of the construction, and assertion (3) follows from (1) and (2). To prove (4), it will suffice (by virtue of (3)) to show that composition with α induces a homotopy equivalence

$$\mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(L_n(X), L_n(Y)) \rightarrow \mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(X, L_n(Y)).$$

Since α induces an equivalence $X| \mathrm{N}(Q(n, \infty)) \simeq L_n(X)| \mathrm{N}(Q(n, \infty))$, it will suffice to show that $L_n(Y)$ is a right Kan extension of $L_n(Y)| \mathrm{N}(Q(n, \infty))$, which follows from the equivalence of (*i*) and (*ii*) in Lemma 8.12. □

Corollary 8.17. *Let $\mathcal{C}$ be a pointed ∞-category satisfying the hypotheses of Remark 8.8. Let $\{L_n : \mathrm{PSp}(\mathcal{C}) \rightarrow \mathrm{PSp}(\mathcal{C})\}_{n \geq 0}$ be the localization functors of Corollary 8.16. Then $L = \varinjlim_n L_n$ is a localization functor from $\mathrm{PSp}(\mathcal{C})$ to itself, whose essential image is the collection of spectrum objects of $\mathcal{C}$.*

Proof. It will suffice to prove the following assertions for every prespectrum object X of $\mathcal{C}$:

(1) The prespectrum LX is a spectrum object of $\mathcal{C}$.

(2) For every spectrum object Y of $\mathcal{C}$, composition with the map $\alpha : X = L_0 X \rightarrow LX$ induces a homotopy equivalence
$$\mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(LX, Y) \rightarrow \mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(X, Y).$$

To prove (1), it suffices to show that LX is a prespectrum below n, for each $n \geq 0$. Since LX is a colimit of the sequence of prespectra $\{L_m X\}_{m \geq n}$, each of which is a spectrum below n, this follows from Remark 8.8.

To prove (2), we note that $\mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(LX, Y)$ is given by the homotopy inverse limit of a tower of spaces $\{\mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(L_n X, Y)\}_{n \geq 0}$. Consequently, it will suffice to prove that each of the canonical maps $\mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(L_n X, Y) \rightarrow \mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(X, Y)$ is a homotopy equivalence. This follows from Corollary 8.16, since Y is a spectrum below n. □

Remark 8.18. Let $\mathcal{C}$ be a pointed ∞-category satisfying the hypotheses of Remark 8.8, and let $f : X \rightarrow Y$ be a morphism between prespectrum objects of $\mathcal{C}$. Suppose that there exists an integer $n \geq 0$ such that f induces an equivalence $X[m] \rightarrow Y[m]$ for $m \geq n$. It follows that $L_m(f) : L_m X \rightarrow L_m Y$ is an equivalence for $m \geq n$, so that $L(f) = \varinjlim_m L_m(f)$ is an equivalence in $\mathrm{Sp}(\mathcal{C})$.

Remark 8.19. Let $\mathcal{C}$ be a presentable pointed ∞-category satisfying the hypotheses of Remark 8.8. Corollary 8.17 asserts that if X is a prespectrum object of $\mathcal{C}$, then the associated spectrum X' is computed in the usual way: the nth space $X'[n]$ is given as a colimit $\mathrm{colim}_{m \geq 0} \Omega^m_{\mathcal{C}} X[n + m]$.

Suppose that $\mathcal{C}$ is a presentable pointed ∞-category. We note that $\mathrm{Sp}(\mathcal{C})$ is closed under small limits in $\mathrm{Fun}(\mathrm{N}(\mathbf{Z} \times \mathbf{Z}), \mathcal{C})$, so that limits in $\mathrm{Sp}(\mathcal{C})$ are computed pointwise. It follows that the evaluation functors $\Omega^{\infty - n}_{\mathcal{C}} : \mathrm{Sp}(\mathcal{C}) \rightarrow \mathcal{C}$ preserve small limits. Since these functors are also accessible, Corollary T.5.5.2.9 guarnatees that $\Omega^{\infty - n}_{\mathcal{C}}$ admits a left adjoint, which we will denote by $\Sigma^{\infty - n}_{\mathcal{C}} : \mathcal{C} \rightarrow \mathrm{Sp}(\mathcal{C})$. Our next goal is to describe this functor in more explicit terms.

Lemma 8.20. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite limits and colimits. Then:*

(1) *A prespectrum object X of $\mathcal{C}$ is a suspension prespectrum above n if and only if X is a left Kan extension of $X|Q(-\infty, n)$.*

(2) *For every $X_0 \in \mathrm{PSp}^n_{-\infty}(\mathcal{C})$, there exists an extension $X \in \mathrm{PSp}(\mathcal{C})$ of X_0 which satisfies the equivalent conditions of* (1).

(3) *Let $\mathcal{D}$ denote the full subcategory of $\mathrm{PSp}(\mathcal{C})$ spanned by those prespectrum objects of $\mathcal{C}$ which are suspension prespectra above n. Then $\mathcal{D}$ is a colocalization of $\mathrm{PSp}(\mathcal{C})$. Moreover, a morphism of prespectra $X \to Y$ exhibits X as a $\mathcal{D}$-colocalization of Y if and only if $X \in \mathcal{D}$ and the the induced map $X[k] \to Y[k]$ is an equivalence for $k \leq n$.*

(4) *Let $\mathcal{D}_0$ denote the full subcategory of $\mathrm{PSp}(\mathcal{C})$ spanned by the n-suspension prespectrum objects, and let $\mathcal{E}$ denote the full subcategory of $\mathrm{PSp}^n_{-\infty}(\mathcal{C})$ spanned by those functors X such that the induced map $X[k] \to \Omega_{\mathcal{C}} X[k+1]$ is an equivalence for $k < n$. Then the restriction maps $\mathcal{D} \to \mathrm{PSp}(\mathcal{C})$ and $\mathcal{D}_0 \to \mathcal{E}$ are trivial Kan fibrations.*

(5) *The ∞-category $\mathcal{E}$ is a localization of $\mathrm{PSp}^n_{-\infty}(\mathcal{C})$. Moreover, a morphism $X \to Y$ in $\mathrm{PSp}^n_{-\infty}(\mathcal{C})$ exhibits Y as an $\mathcal{E}$-localization of X if and only if $Y \in \mathcal{E}$ and the map $X[n] \to Y[n]$ is an equivalence.*

(6) *The ∞-category $\mathcal{D}_0$ is a localization of $\mathcal{D}$. Moreover, a morphism $X \to Y$ in $\mathcal{D}$ exhibits Y as an $\mathcal{D}_0$-localization of X if and only if $Y \in \mathcal{D}_0$ and the map $X[n] \to Y[n]$ is an equivalence.*

Proof. Assertions (1) and (2) follow by applying Lemma 8.12 to the opposite ∞-category $\mathcal{C}^{op}$. Assertion (4) follows from (1) and (2) together with Proposition T.4.3.2.15, and assertion (6) follows immediately from (5) and (4). We will give the proof of (3); the proof of (5) is similar.

Consider an arbitrary object $Y \in \mathrm{PSp}(\mathcal{C})$. Let $Y_0 = Y|\,\mathrm{N}(Q(-\infty,n))$, and let $X \in \mathrm{PSp}(\mathcal{C})$ be a left Kan extension of Y_0 (whose existence is guaranteed by (2)). Then $X \in \mathcal{D}$ and we have a canonical map $\alpha : X \to Y$. We claim that α exhibits X as a a $\mathcal{D}$-colocalization of Y. To prove this, let us consider an arbitrary object $W \in \mathcal{D}$. We have a commutative diagram

$$\begin{tikzcd}
\mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(W,X) \arrow[r] \arrow[d] & \mathrm{Map}_{\mathrm{PSp}^n_{-\infty}(\mathcal{C})}(W|\,\mathrm{N}(Q(-\infty,n)), X|\,\mathrm{N}(Q(-\infty,n))) \arrow[d] \\
\mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(W,Y) \arrow[r] & \mathrm{Map}_{\mathrm{PSp}^n_{-\infty}(\mathcal{C})}(W|\,\mathrm{N}(Q(-\infty,n)), Y|\,\mathrm{N}(Q(-\infty,n))).
\end{tikzcd}$$

Since X and Y have the same restriction to $\mathrm{N}(Q(-\infty,n))$, the right vertical map is a homotopy equivalence. The horizontal maps are homotopy equivalences since W is a left Kan extension of $W|\,\mathrm{N}(Q(-\infty,n))$, by virtue of (1). This completes the proof that α exhibits X as a $\mathcal{D}$-colocalization of Y, and the proof that $\mathcal{D}$ is a colocalization of $\mathrm{PSp}(\mathcal{C})$.

To complete the proof of (3), let us consider an arbitrary map $\beta : X' \to Y$, where $X' \in \mathcal{D}$. We wish to show that β exhibits X' as a $\mathcal{D}$-colocalization of Y if and only if the induced map $X'[m] \to Y[m]$ is an equivalence for $m \leq n$. The above argument shows that β fits into a commutative triangle

$$\begin{tikzcd}
& X \arrow[dr, "\alpha"] & \\
X' \arrow[ur, "\gamma"] \arrow[rr, "\beta"] & & Y.
\end{tikzcd}$$

Since α exhibits X as a $\mathcal{D}$-colocalization of Y, and α induces equivalences $X[m] \to Y[m]$ for $m \leq n$, we can restate the desired assertion as follows: the map γ is an equivalence if and only if γ induces equivalences $X[m] \to X'[m]$ for $m \leq n$. The "only if" part of the assertion is obvious, and the converse follows from the fact that both X and X' are left Kan extensions of their restrictions to $\mathrm{N}(Q(-\infty,n))$ (by virtue of (1)). □

Proposition 8.21. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite limits and colimits, and let $\mathcal{D}_0 \subseteq \mathrm{PSp}(\mathcal{C})$ denote the full subcategory spanned by the n-suspension prespectra. Then evaluation at (n,n) induces a trivial Kan fibration $\mathcal{D}_0 \to \mathcal{C}$.*

Proof. Let $\mathcal{E} \subseteq \mathrm{PSp}^n_{-\infty}(\mathcal{C})$ be defined as in Lemma 8.20. The evaluation functor factors as a composition

$$\mathcal{D}_0 \xrightarrow{\phi_0} \mathcal{E} \xrightarrow{\phi_1} \mathrm{PSp}^n_n(\mathcal{C}) \xrightarrow{\phi_2} \mathcal{C}\,.$$

Here ϕ_0 is a trivial fibration by Lemma 8.20, the map ϕ_1 is a trivial fibration by virtue of Lemma 8.12 and Proposition T.4.3.2.15, and the map ϕ_2 is a trivial fibration by Lemma 8.13. □

Notation 8.22. Let $\mathcal{C}$ be a pointed ∞-category which admits finite limits and colimits. We let $\widetilde{\Sigma}^{\infty-n}_{\mathcal{C}} : \mathcal{C} \to \mathrm{PSp}(\mathcal{C})$ denote a section of the trivial Kan fibration $\mathcal{D}_0 \to \mathcal{C}$ of Proposition 8.21.

Remark 8.23. Let $\mathcal{C}$ be a pointed ∞-category which admits finite limits and colimits, let $C \in \mathcal{C}$ be an object and let $X \in \mathrm{PSp}(\mathcal{C})$ be a spectrum below n. Then the canonical map

$$e : \mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(\widetilde{\Sigma}^{\infty-n}_{\mathcal{C}} C, X) \to \mathrm{Map}_{\mathcal{C}}(C, \Omega^{\infty-n}_{\mathcal{C}} X)$$

is a homotopy equivalence. To prove this, we observe that e factors as a composition (using the conventions of Notation 8.10)

$$\begin{aligned}
\mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(\widetilde{\Sigma}^{\infty-n}_{\mathcal{C}} C, X) &\overset{\phi_0}{\to} \mathrm{Map}_{\mathrm{PSp}^n_{-\infty}(\mathcal{C})}((\widetilde{\Sigma}^{\infty-n}_{\mathcal{C}} C)|\,\mathrm{N}(Q(-\infty, n)), X|\,\mathrm{N}(Q(-\infty, n))) \\
&\overset{\phi_1}{\to} \mathrm{Map}_{\mathrm{PSp}^n_n(\mathcal{C})}((\widetilde{\Sigma}^{\infty-n}_{\mathcal{C}} C)|\,\mathrm{N}(Q(n, n)), X|\,\mathrm{N}(Q(n, n))) \\
&\overset{\phi_2}{\to} \mathrm{Map}_{\mathcal{C}}(C, X[n]).
\end{aligned}$$

It will therefore suffice to prove that the maps ϕ_0, ϕ_1, and ϕ_2 are homotopy equivalences. For ϕ_0, the desired result follows from our assumption that $\widetilde{\Sigma}^{\infty-n}_{\mathcal{C}} C$ is a left Kan extension of its restriction to $\mathrm{N}(Q(-\infty, n))$. For ϕ_1, we invoke the fact that$X|\,\mathrm{N}(Q(-\infty, n))$ is a right Kan extension of its restriction to $\mathrm{N}(Q(n, n))$. For ϕ_2, we apply Lemma 8.13.

Proposition 8.24. *Let $\mathcal{C}$ be a presentable pointed ∞-category, and let $L : \mathrm{PSp}(\mathcal{C}) \to \mathrm{Sp}(\mathcal{C})$ denote a left adjoint to the inclusion. Then the evaluation functor $\Omega^{\infty-n}_{\mathcal{C}} : \mathrm{Sp}(\mathcal{C}) \to \mathcal{C}$ admits a left adjoint, given by the composition*

$$\mathcal{C} \xrightarrow{\widetilde{\Sigma}^{\infty-n}_{\mathcal{C}}} \mathrm{PSp}(\mathcal{C}) \xrightarrow{L} \mathrm{Sp}(\mathcal{C}).$$

Proof. The canonical natural transformation $\mathrm{id}_{\mathrm{PSp}(\mathcal{C})} \to L$ induces a transformation

$$\alpha : \mathrm{id}_{\mathcal{C}} = \Omega^{\infty-n}_{\mathcal{C}} \circ \widetilde{\Sigma}^{\infty-n}_{\mathcal{C}} \to \Omega^{\infty-n}_{\mathcal{C}} \circ (L \circ \widetilde{\Sigma}^{\infty-n}_{\mathcal{C}}).$$

We claim that α is the unit of an adjunction. To prove this, it suffices to show that for every object $C \in \mathcal{C}$ and every spectrum object $X \in \mathrm{Sp}(\mathcal{C})$, the composite map

$$\mathrm{Map}_{\mathrm{Sp}(\mathcal{C})}(L\widetilde{\Sigma}^{\infty-n}_{\mathcal{C}} C, X) \xrightarrow{\phi} \mathrm{Map}_{\mathrm{PSp}(\mathcal{C})}(\widetilde{\Sigma}^{\infty-n}_{\mathcal{C}} C, X) \xrightarrow{\psi} \mathrm{Map}_{\mathcal{C}}(C, X[n])$$

is a homotopy equivalence. It now suffices to observe that ϕ is a homotopy equivalence because X is a spectrum object, and ψ is a homotopy equivalence by virtue of Remark 8.23. □

We close this section by discussing the *shift* functor on prespectrum objects of an ∞-category $\mathcal{C}$. We observe that precomposition with the map $(i, j) \mapsto (i+1, j+1)$ determines a functor $S : \mathrm{PSp}(\mathcal{C}) \to \mathrm{PSp}(\mathcal{C})$, which restricts to a functor $\mathrm{Sp}(\mathcal{C}) \to \mathrm{Sp}(\mathcal{C})$ which we will also denote by S. By construction, this functor is an equivalence (in fact, an isomorphism of simplicial sets). We observe that if X is a spectrum object of $\mathcal{C}$, then we have canonical equivalences $\Omega_{\mathcal{C}} S(X)[n] = \Omega_{\mathcal{C}} X[n+1] \simeq X$; this strongly suggests that S is a homotopy inverse to the loop functor $\Omega_{\mathrm{Sp}(\mathcal{C})}$ given by pointwise composition with $\Omega_{\mathcal{C}}$. To prove this (in a slightly stronger form), we need to introduce a bit of notation.

Notation 8.25. Consider the order-preserving maps $s_+, s_- : \mathbf{Z} \times \mathbf{Z} \to \mathbf{Z} \times \mathbf{Z}$ defined by the formulae

$$s_+(i,j) = \begin{cases} (i,j) & \text{if } i \neq j \\ (i+1,j) & \text{if } i = j. \end{cases} \qquad s_-(i,j) = \begin{cases} (i,j) & \text{if } i \neq j \\ (i,j+1) & \text{if } i = j. \end{cases}$$

For every ∞-category $\mathcal{C}$, composition with s_+ and s_- induces functors $S_+, S_- : \mathrm{PSp}(\mathcal{C}) \to \mathrm{PSp}(\mathcal{C})$, fitting into a commutative diagram

$$\begin{tikzcd} \mathrm{id} \arrow[r] \arrow[d] & S_+ \arrow[d] \\ S_- \arrow[r] & S. \end{tikzcd}$$

Note that the images of s_+ and s_- are disjoint from the diagonal $\{(n,n)\}_{n \geq 0} \subseteq \mathbf{Z}_{\geq 0} \times \mathbf{Z}_{\geq 0}$, so that $S_+(X)$ and $S_-(X)$ are zero objects of $\mathrm{PSp}(\mathcal{C})$ for every $X \in \mathrm{PSp}(\mathcal{C})$. If $\mathcal{C}$ admits finite limits, then the above diagram determines a morphism $\alpha : X \to \Omega_{\mathrm{PSp}(\mathcal{C})} S(X)$. If $\mathcal{C}$ also admits finite colimits, then α admits an adjoint $\beta : \Sigma_{\mathrm{PSp}(\mathcal{C})} X \to S(X)$.

Lemma 8.26. *Let $\mathcal{C}$ be a small pointed ∞-category, and let $\mathcal{P}_*(\mathcal{C})$ denote the full subcategory of $\mathcal{P}(\mathcal{C}) = \mathrm{Fun}(\mathcal{C}^{op}, \mathcal{S})$ spanned by those functors which carry zero objects of $\mathcal{C}$ to final objects of $\mathcal{S}$. Then:*

(1) *Let S denote the set consisting of a single morphism from an initial object of $\mathcal{P}(\mathcal{C})$ to a final object of $\mathcal{P}(\mathcal{C})$. Then $\mathcal{P}_*(\mathcal{C}) = S^{-1}\,\mathcal{P}(\mathcal{C})$.*

(2) *The ∞-category $\mathcal{P}_*(\mathcal{C})$ is an accessible localization of $\mathcal{P}(\mathcal{C})$. In particular, $\mathcal{P}_*(\mathcal{C})$ is presentable.*

(3) *The Yoneda embedding $\mathcal{C} \to \mathcal{P}(\mathcal{C})$ factors through $\mathcal{P}_*(\mathcal{C})$, and the induced embedding $j : \mathcal{C} \to \mathcal{P}_*(\mathcal{C})$ preserves zero objects.*

(4) *Let $\mathcal{D}$ be an ∞-category which admits small colimits, and let $\mathrm{Fun}^L(\mathcal{P}_*(\mathcal{C}), \mathcal{D})$ denote the full subcategory of $\mathrm{Fun}(\mathcal{P}_*(\mathcal{C}), \mathcal{D})$ spanned by those functors which preserve small colimits. Then composition with j induces an equivalence of ∞-categories $\mathrm{Fun}^L(\mathcal{P}_*(\mathcal{C}), \mathcal{D}) \to \mathrm{Fun}_0(\mathcal{C}, \mathcal{D})$, where $\mathrm{Fun}_0(\mathcal{C}, \mathcal{D})$ denotes the full subcategory of $\mathrm{Fun}(\mathcal{C}, \mathcal{D})$ spanned by those functors which carry zero objects of $\mathcal{C}$ to initial objects of $\mathcal{D}$.*

(5) *The ∞-category $\mathcal{P}_*(\mathcal{C})$ is pointed.*

(6) *The full subcategory $\mathcal{P}_*(\mathcal{C}) \subseteq \mathcal{P}(\mathcal{C})$ is closed under small limits and under small colimits parametrized by weakly contractible simplicial sets. In particular, $\mathcal{P}_*(\mathcal{C})$ is stable under small filtered colimits in $\mathcal{P}(\mathcal{C})$.*

(7) *The functor $j : \mathcal{C} \to \mathcal{P}_*(\mathcal{C})$ preserves all small limits which exist in $\mathcal{C}$.*

(8) *The ∞-category $\mathcal{P}_*(\mathcal{C})$ is compactly generated.*

Proof. For every object $X \in \mathcal{S}$, let $F_X \in \mathcal{P}(\mathcal{C})$ denote the constant functor taking the value X. Then F_X is a left Kan extension of $F_X|\{0\}$, where 0 denotes a zero object of $\mathcal{C}$. It follows that for any object $G \in \mathcal{P}(\mathcal{C})$, evaluation at 0 induces a homotopy equivalence

$$\mathrm{Map}_{\mathcal{P}(\mathcal{C})}(F_X, G) \to \mathrm{Map}_{\mathcal{S}}(F_X(0), G(0)) = \mathrm{Map}_{\mathcal{S}}(X, G(0)).$$

We observe that the inclusion $\emptyset \subseteq \Delta^0$ induces a map $F_\emptyset \to F_{\Delta^0}$ from an initial object of $\mathcal{P}(\mathcal{C})$ to a final object of $\mathcal{P}(\mathcal{C})$. It follows that an object G of $\mathcal{P}(\mathcal{C})$ is S-local if and only if the induced map

$$G(0) \simeq \mathrm{Map}_{\mathcal{S}}(\Delta^0, G(0)) \to \mathrm{Map}_{\mathcal{S}}(\emptyset, G(0)) \simeq \Delta^0$$

is a homotopy equivalence: that is, if and only if $G \in \mathcal{P}_*(\mathcal{C})$. This proves (1).

Assertion (2) follows immediately from (1), and assertion (3) is obvious. Assertion (4) follows from (1), Theorem T.5.1.5.6, and Proposition T.5.5.4.20. To prove (5), we observe that F_{Δ^0} is a final object of $\mathcal{P}(\mathcal{C})$, and therefore a final object of $\mathcal{P}_*(\mathcal{C})$. It therefore suffices to show that F_{Δ^0} is an initial object of $\mathcal{P}_*(\mathcal{C})$. This follows from the observation that for every $G \in \mathcal{P}(\mathcal{C})$, we have homotopy equivalences $\mathrm{Map}_{\mathcal{P}(\mathcal{C})}(F_{\Delta^0}, G) \simeq \mathrm{Map}_{\mathcal{S}}(\Delta^0, G(0)) \simeq G(0)$ so that the mapping space $\mathrm{Map}_{\mathcal{P}(\mathcal{C})}(F_{\Delta^0}, G)$ is contractible if $G \in \mathcal{P}_*(\mathcal{C})$.

Assertion (6) is obvious, and (7) follows from (6) together with Proposition T.5.1.3.2. We deduce (8) from (6) together with Corollary T.5.5.7.3. □

Proposition 8.27. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite limits. Then:*

(1) *For every object $X \in \mathrm{Sp}(\mathcal{C})$, the canonical map $X \to \Omega_{\mathrm{PSp}(\mathcal{C})} S(X)$ is an equivalence.*

(2) *The shift functor $S : \mathrm{Sp}(\mathcal{C}) \to \mathrm{Sp}(\mathcal{C})$ is a homotopy inverse to the loop functor $\Omega_{\mathrm{Sp}(\mathcal{C})}$.*

(3) *The ∞-category $\mathrm{Sp}(\mathcal{C})$ is stable.*

Proof. Assertion (1) is an immediate consequence of the definitions. We note that (1) implies that S is a right homotopy inverse to $\Omega_{\mathrm{Sp}(\mathcal{C})}$. Since S is invertible, it follows that S is also a left homotopy inverse to $\Omega_{\mathrm{Sp}(\mathcal{C})}$. In particular, $\Omega_{\mathrm{Sp}(\mathcal{C})}$ is invertible.

To prove (3), we may assume without loss of generality that $\mathcal{C}$ is small. Lemma 8.26 implies that there exists a fully faithful left exact functor $j : \mathcal{C} \to \mathcal{D}$, where $\mathcal{D}$ is a compactly generated pointed ∞-category (this that the functor $\Omega_{\mathcal{D}}$ preserves sequential colimits; see Remark 8.9). Then $\mathrm{Sp}(\mathcal{C})$ is equivalent to a full subcategory of $\mathrm{Sp}(\mathcal{D})$, which is closed under finite limits and shifts. Consequently, it will suffice to show that $\mathrm{Sp}(\mathcal{D})$ is stable, which is a consequence of Corollary 10.10 (proven in §10). □

Corollary 8.28. *Let $\mathcal{C}$ be a pointed ∞-category. The following conditions are equivalent:*

(1) *The ∞-category $\mathcal{C}$ is stable.*

(2) *The ∞-category $\mathcal{C}$ admits finite colimits and the suspension functor $\Sigma : \mathcal{C} \to \mathcal{C}$ is an equivalence.*

(3) *The ∞-category $\mathcal{C}$ admits finite limits and the loop functor $\Omega : \mathcal{C} \to \mathcal{C}$ is an equivalence.*

Proof. We will show that (1) ⇔ (3); the dual argument will prove that (1) ⇔ (2). The implication (1) ⇒ (3) is clear. Conversely, suppose that $\mathcal{C}$ admits finite limits and that Ω is an equivalence. Lemma T.7.2.2.9 asserts that the forgetful functor $\mathcal{C}_* \to \mathcal{C}$ is a trivial fibration. Consequently, $\mathrm{Sp}(\mathcal{C})$ can be identified with the homotopy inverse limit of the tower

$$\ldots \xrightarrow{\Omega} \mathcal{C} \xrightarrow{\Omega} \mathcal{C}.$$

By assumption, the loop functor Ω is an equivalence, so this tower is essentially constant. It follows that $\Omega^\infty : \mathrm{Sp}(\mathcal{C}) \to \mathcal{C}$ is an equivalence of ∞-categories. Since $\mathrm{Sp}(\mathcal{C})$ is stable (Proposition 8.27), so is $\mathcal{C}$. □

For later use, we record also the following result:

Proposition 8.29. *Let $\mathcal{C}$ be a pointed ∞-category satisfying the hypotheses of Remark 8.8, and let L : $\mathrm{PSp}(\mathcal{C}) \to \mathrm{Sp}(\mathcal{C})$ denote a left adjoint to the inclusion. Let X be a prespectrum object of $\mathcal{C}$, and let β : $\Sigma_{\mathrm{PSp}(\mathcal{C})} X \to S(X)$ denote the map described in Notation 8.25. Then $L(\beta)$ is an equivalence in $\mathrm{Sp}(\mathcal{C})$.*

Proof. Since L is a left adjoint, it commutes with suspensions. It will therefore suffice to show that β induces an equivalence $\Sigma_{\mathrm{Sp}(\mathcal{C})} LX \to LS(X)$: in other words, we must show that the diagram σ :

$$\begin{tikzcd}
LX \arrow[r] \arrow[d] & LS_+(X) \arrow[d] \\
LS_-(X) \arrow[r] & LS(X)
\end{tikzcd}$$

is a pushout square in the ∞-category $\mathrm{Sp}(\mathcal{C})$. Since $\mathrm{Sp}(\mathcal{C})$ is stable, it suffices to show that σ is a pullback square. In other words, we must show that for each $n \geq 0$, the diagram

$$\begin{tikzcd}
LX[n] \arrow[r] \arrow[d] & LS_{+}(X)[n] \arrow[d] \\
LS_{-}(X)[n] \arrow[r] & LS(X)[n]
\end{tikzcd}$$

is a pullback square in $\mathcal{C}$.

Let P be defined as in Lemma 8.15, let $\overline{X}_0 : \mathrm{N}(P) \to \mathcal{C}$ be given by the composition

$$\mathrm{N}(P) \to \mathrm{N}(\mathbf{Z} \times \mathbf{Z}) \xrightarrow{X} \mathcal{C},$$

and let $\overline{X} : \mathrm{N}(\mathbf{Z} \times \mathbf{Z} \times \mathbf{Z}) \to \mathcal{C}$ be a right Kan extension of $\overline{X}_0$. Let $\overline{S}(\overline{X})$, $\overline{S}_{+}(\overline{X})$, and $\overline{S}_{-}(\overline{X})$ be obtained from $\overline{X}$ by composing with the maps $\mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} \to \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z}$ given by $(i, j, k) \mapsto (i+1, j+1, k)$, $s_{+} \times \mathrm{id}$, and $s_{-} \times \mathrm{id}$. We have a commutative diagram

$$\begin{tikzcd}
\overline{X} \arrow[r] \arrow[d] & \overline{S}_{+}(\overline{X}) \arrow[d] \\
\overline{S}_{-}(\overline{X}) \arrow[r] & \overline{S}(\overline{X})
\end{tikzcd}$$

which we can think of as encoding a sequence of commutative squares $\{\sigma_k : \Delta^1 \times \Delta^1 \to \mathrm{PSp}(\mathcal{C})\}_{k \geq 0}$. We can identify σ with the colimit of this sequence. Consequently, it will suffice to prove that for every integer n, the diagram

$$\begin{tikzcd}
\overline{X}(n,n,k) \arrow[r] \arrow[d] & \overline{S}_{+}(\overline{X})(n,n,k) \arrow[d] \\
\overline{S}_{-}(\overline{X})(n,n,k) \arrow[r] & \overline{S}(\overline{X})(n,n,k)
\end{tikzcd}$$

is a pullback square in $\mathcal{C}$ for all sufficiently large k. In fact, this is true for all $k > n$, by virtue of Lemma 8.15. □

9 The ∞-Category of Spectra

In this section, we will discuss what is perhaps the most important example of a stable ∞-category: the ∞-category of spectra. In classical homotopy theory, one defines a spectrum to be a sequence of pointed spaces $\{X_n\}_{n \geq 0}$, equipped with homotopy equivalences (or homeomorphisms, depending on the author) $X_n \to \Omega(X_{n+1})$ for all $n \geq 0$. This admits an immediate ∞-categorical translation:

Definition 9.1. A *spectrum* is a spectrum object of the ∞-category $\mathcal{S}_*$ of pointed spaces. We let $\mathrm{Sp} = \mathrm{Sp}(\mathcal{S}_*) = \mathrm{Stab}(\mathcal{S})$ denote the ∞-category of spectra.

Proposition 9.2.

(1) *The ∞-category* Sp *is stable.*

(2) *Let* $(\mathrm{Sp})_{\leq -1}$ *denote the full subcategory of* Sp *spanned by those objects X such that $\Omega^\infty(X) \in \mathcal{S}$ is contractible. Then* $(\mathrm{Sp})_{\leq -1}$ *determines an accessible t-structure on* Sp *(see §16).*

(3) *The t-structure on* Sp *is both left complete and right complete, and the heart $\mathcal{S}_\infty^\heartsuit$ is canonically equivalent to the (nerve of the) category of abelian groups.*

Proof. Assertion (1) follows immediately from Proposition 8.27. Assertion (2) is a special case of Proposition 16.4, which will be established in §16. We will prove (3). Note that a spectrum X can be identified with a sequence of pointed spaces $\{X(n)\}$, equipped with equivalences $X(n) \simeq \Omega X(n+1)$ for all $n \geq 0$. We observe that $X \in (\mathrm{Sp})_{\leq m}$ if and only if each $X(n)$ is $(n+m)$-truncated. In general, the sequence $\{\tau_{\leq n+m} X(n)\}$ itself determines a spectrum, which we can identify with the truncation $\tau_{\leq m} X$. It follows that $X \in (\mathrm{Sp})_{\geq m+1}$ if and only if each $X(n)$ is $(n+m+1)$-connective. In particular, X lies in the heart of Sp if and only if each $X(n)$ is an Eilenberg-MacLane object of $\mathcal{S}$ of degree n (see Definition T.7.2.2.1). It follows that the heart of Sp can be identified with the homotopy inverse limit of the tower of ∞-categories

$$\ldots \xrightarrow{\Omega} \mathcal{EM}_1(\mathcal{S}) \xrightarrow{\Omega} \mathcal{EM}_0(\mathcal{S}),$$

where $\mathcal{EM}_n(\mathcal{S})$ denotes the full subcategory of $\mathcal{S}_*$ spanned by the Eilenberg-MacLane objects of degree n. Proposition T.7.2.2.12 asserts that after the second term, this tower is equivalent to the constant diagram taking the value $\mathrm{N}(\mathcal{A}b)$, where $\mathcal{A}b$ is category of abelian groups.

It remains to prove that Sp is both right and left complete. We begin by observing that if $X \in \mathrm{Sp}$ is such that $\pi_n X \simeq 0$ for all $n \in \mathbf{Z}$, then X is a zero object of Sp (since each $X(n) \in \mathcal{S}$ has vanishing homotopy groups, and is therefore contractible by Whitehead's theorem). Consequently, both $\bigcap (\mathrm{Sp})_{\leq -n}$ and $\bigcap (\mathrm{Sp})_{\geq n}$ coincide with the collection of zero objects of Sp. It follows that

$$(\mathrm{Sp})_{\geq 0} = \{X \in \mathrm{Sp} : (\forall n < 0)[\pi_n X \simeq 0]\}$$

$$(\mathrm{Sp})_{\leq 0} = \{X \in \mathrm{Sp} : (\forall n > 0)[\pi_n X \simeq 0]\}.$$

According to Proposition 7.3, to prove that Sp is left and right complete it will suffice to show that the subcategories $(\mathrm{Sp})_{\geq 0}$ and $(\mathrm{Sp})_{\leq 0}$ are stable under products and coproducts. In view of the above formulas, it will suffice to show that the homotopy group functors $\pi_n : \mathrm{Sp} \rightarrow \mathrm{N}(\mathcal{A}b)$ preserve products and coproducts. Since π_n obviously commutes with finite coproducts, it will suffice to show that π_n commutes with products and filtered colimits. Shifting if necessary, we may reduce to the case $n = 0$. Since products and filtered colimits in the category of abelian groups can be computed at the level of the underlying sets, we are reduced to proving that the composition

$$\mathrm{Sp} \xrightarrow{\Omega^\infty} \mathcal{S} \xrightarrow{\pi_0} \mathrm{N}(\mathcal{S}\mathrm{et})$$

preserves products and filtered colimits. This is clear, since each of the factors individually preserves products and filtered colimits. □

Our next goal is to prove that the ∞-category Sp is compactly generated. To prove this, we need to review a bit of the theory of finite spaces.

Notation 9.3. Let $\mathcal{S}_*$ denote the ∞-category of pointed objects of $\mathcal{S}$. That is, $\mathcal{S}_*$ denotes the full subcategory of $\mathrm{Fun}(\Delta^1, \mathcal{S})$ spanned by those morphisms $f : X \rightarrow Y$ for which X is a final object of $\mathcal{S}$ (Definition T.7.2.2.1). Let $\mathcal{S}^{\mathrm{fin}}$ denote the smallest full subcategory of $\mathcal{S}$ which contains the final object $*$ and is stable under finite colimits. We will refer to $\mathcal{S}^{\mathrm{fin}}$ as the *∞-category of finite spaces*. We let $\mathcal{S}^{\mathrm{fin}}_* \subseteq \mathcal{S}_*$ denote the ∞-category of pointed objects of $\mathcal{S}^{\mathrm{fin}}$. We observe that the suspension functor $\Sigma : \mathcal{S}_* \rightarrow \mathcal{S}_*$ carries $\mathcal{S}^{\mathrm{fin}}_*$ to itself. For each $n \geq 0$, we let $S^n \in \mathcal{S}_*$ denote a representative for the (pointed) n-sphere.

Remark 9.4. It follows from Remark T.5.3.5.9 and Proposition T.4.3.2.15 that $\mathcal{S}^{\mathrm{fin}}$ is characterized by the following universal property: for every ∞-category $\mathcal{D}$ which admits finite colimits, evaluation at $*$ induces an equivalence of ∞-categories $\mathrm{Fun}^{\mathrm{Rex}}(\mathcal{S}^{\mathrm{fin}}, \mathcal{D}) \rightarrow \mathcal{D}$. Here $\mathrm{Fun}^{\mathrm{Rex}}(\mathcal{S}^{\mathrm{fin}}, \mathcal{D})$ denotes the full subcategory of $\mathrm{Fun}(\mathcal{S}^{\mathrm{fin}}, \mathcal{D})$ spanned by the right exact functors.

Lemma 9.5. (1) *Each object of $\mathcal{S}^{\mathrm{fin}}_*$ is compact in $\mathcal{S}_*$.*

(2) *The inclusion $\mathcal{S}^{\mathrm{fin}}_* \subseteq \mathcal{S}_*$ induces an equivalence* $\mathrm{Ind}(\mathcal{S}^{\mathrm{fin}}_*) \rightarrow \mathcal{S}_*$. *In particular, $\mathcal{S}_*$ is compactly generated.*

(3) *The subcategory $\mathcal{S}_*^{\mathrm{fin}} \subseteq \mathcal{S}_*$ is the smallest full subcategory which contains S^0 and is stable under finite colimits.*

Proof. Since $\mathcal{S}^{\mathrm{fin}}$ consists of compact objects of $\mathcal{S}$, Proposition T.5.4.5.15 implies that $\mathcal{S}_*^{\mathrm{fin}}$ consists of compact objects of $\mathcal{S}_*$. This proves (1).

We next observe that $\mathcal{S}_*^{\mathrm{fin}}$ is stable under finite colimits in $\mathcal{S}_*$. Using the proof of Corollary T.4.4.2.4, we may reduce to showing that $\mathcal{S}_*^{\mathrm{fin}}$ is stable under pushouts and contains an initial object of $\mathcal{S}_*$. The second assertion is obvious, and the first follows from the fact that the forgetful functor $\mathcal{S}_* \to \mathcal{S}$ commutes with pushouts (Proposition T.4.4.2.9).

We now prove (3). Let $\mathcal{S}'_*$ be the smallest full subcategory which contains S^0 and is stable under finite colimits. The above argument shows that $\mathcal{S}'_* \subseteq \mathcal{S}_*^{\mathrm{fin}}$. To prove the converse, we let $f : \mathcal{S} \to \mathcal{S}_*$ be a left adjoint to the forgetful functor, so that $f(X) \simeq X \coprod *$. Then f preserves small colimits. Since $f(*) \simeq S^0 \in \mathcal{S}'_*$, we conclude that f carries $\mathcal{S}^{\mathrm{fin}}$ into $\mathcal{S}'_*$. If $x : * \to X$ is a pointed object of $\mathcal{S}$, then x can be written as a coproduct $f(X) \coprod_{S^0} *$. In particular, if $x \in \mathcal{S}_*^{\mathrm{fin}}$, then $X \in \mathcal{S}^{\mathrm{fin}}$, so that $f(X), S^0, * \in \mathcal{S}'_*$. Since $\mathcal{S}'_*$ is stable under pushouts, we conclude that $x \in \mathcal{S}'_*$; this completes the proof of (3).

We now prove (2). Part (1) and Proposition T.5.3.5.11 imply that we have a fully faithful functor $\theta : \mathrm{Ind}(\mathcal{S}_*^{\mathrm{fin}}) \subseteq \mathcal{S}_*$. Let $\mathcal{S}''_*$ be the essential image of θ. Proposition T.5.5.1.9 implies that $\mathcal{S}''_*$ is stable under small colimits. Since $S^0 \in \mathcal{S}''_*$ and f preserves small colimits, we conclude that $f(X) \in \mathcal{S}''_*$ for all $X \in \mathcal{S}$. Since $\mathcal{S}''_*$ is stable under pushouts, we conclude that $\mathcal{S}''_* = \mathcal{S}_*$, as desired. □

Warning 9.6. The ∞-category $\mathcal{S}^{\mathrm{fin}}$ does not coincide with the ∞-category of compact objects $\mathcal{S}^{\omega} \subseteq \mathcal{S}$. Instead, there is an inclusion $\mathcal{S}^{\mathrm{fin}} \subseteq \mathcal{S}^{\omega}$, which realizes $\mathcal{S}^{\omega}$ as an idempotent completion of $\mathcal{S}^{\mathrm{fin}}$. An object of $X \in \mathcal{S}^{\omega}$ belongs to $\mathcal{S}^{\mathrm{fin}}$ if and only if its *Wall finiteness obstruction* vanishes.

Proposition 9.7. *The ∞-category of spectra is compactly generated. Moreover, an object $X \in \mathrm{Sp}$ is compact if and only if it is a retract of $\Sigma^{\infty - n} Y$, for some $Y \in \mathcal{S}_*^{\mathrm{fin}}$ and some integer n.*

Proof. Let $\mathcal{P}\mathrm{r}^{\mathrm{L}}_{\omega}$, $\mathcal{P}\mathrm{r}^{\mathrm{R}}_{\omega}$, and $\mathcal{C}\mathrm{at}_{\infty}^{\mathrm{Rex}}$ be defined as §T.5.5.7. According to Proposition T.5.5.7.10, we can view the construction of Ind-categories as determining a localization functor $\mathrm{Ind} : \mathcal{C}\mathrm{at}_{\infty}^{\mathrm{Rex}} \to \mathcal{P}\mathrm{r}^{\mathrm{L}}_{\omega}$. Let $\mathcal{S}_{\infty}^{\mathrm{fin}}$ denote the colimit of the sequence

$$\mathcal{S}_*^{\mathrm{fin}} \xrightarrow{\Sigma} \mathcal{S}_*^{\mathrm{fin}} \xrightarrow{\Sigma} \ldots$$

in $\mathcal{C}\mathrm{at}_{\infty}^{\mathrm{Rex}}$. Since $\mathcal{S}_* \simeq \mathrm{Ind}(\mathcal{S}_*^{\mathrm{fin}})$ (Lemma 9.5) and the functor Ind preserves colimits, we conclude that $\mathrm{Ind}(\mathcal{S}_{\infty}^{\mathrm{fin}})$ can be identified with the colimit of the sequence

$$\mathcal{S}_* \xrightarrow{\Sigma} \mathcal{S}_* \xrightarrow{\Sigma} \ldots$$

in $\mathcal{P}\mathrm{r}^{\mathrm{L}}_{\omega}$. Invoking the equivalence between $\mathcal{P}\mathrm{r}^{\mathrm{L}}_{\omega}$ and $(\mathcal{P}\mathrm{r}^{\mathrm{R}}_{\omega})^{op}$ (see Notation T.5.5.7.7), we can identify $\mathrm{Ind}(\mathcal{S}_{\infty}^{\mathrm{fin}})$ with the *limit* of the tower

$$\ldots \xrightarrow{\Omega} \mathcal{S}_* \xrightarrow{\Omega} \mathcal{S}_*$$

in $\mathcal{P}\mathrm{r}^{\mathrm{R}}_{\omega}$. Since the inclusion functor $\mathcal{P}\mathrm{r}^{\mathrm{R}}_{\omega} \subseteq \widehat{\mathcal{C}\mathrm{at}}_{\infty}$ preserves limits (Proposition T.5.5.7.6), we conclude that there is an equivalence $F : \mathrm{Ind}(\mathcal{S}_{\infty}^{\mathrm{fin}}) \simeq \mathrm{Sp}$ (Proposition 8.14). This proves that Sp is compactly generated, and that the compact objects of Sp are precisely those which appear as retracts of $F(Y)$, for some $Y \in \mathrm{Ind}(\mathcal{S}_{\infty}^{\mathrm{fin}})$. To complete the proof, we observe that Y itself lies in the image of one of the maps $\mathcal{S}_*^{\mathrm{fin}} \to \mathcal{S}_{\infty}^{\mathrm{fin}}$, and that the composite maps

$$\mathcal{S}_*^{\mathrm{fin}} \to \mathcal{S}_{\infty}^{\mathrm{fin}} \to \mathrm{Ind}(\mathcal{S}_{\infty}^{\mathrm{fin}}) \xrightarrow{F} \mathrm{Sp}$$

are given by restricting the suspension spectrum functors $\Sigma^{\infty - n} : \mathcal{S}_*^{\mathrm{fin}} \to \mathrm{Sp}$. □

Remark 9.8. The proof of Proposition 9.7 implies that we can identify $\mathcal{S}_{\infty}^{\mathrm{fin}}$ with a full subcategory of the compact objects of Sp. In fact, every compact object of Sp belongs to this full subcategory. The proof of this is not completely formal (especially in view of Warning 9.6); it relies on the fact that the ring of integers $\mathbf{Z}$ is a principal ideal domain, so that every finitely generated projective $\mathbf{Z}$-module is free.

Remark 9.9. It is possible to use the proof of Proposition 9.7 to prove directly that the ∞-category Sp is stable, without appealing to the general results on stabilization proved in §8. Indeed, by virtue of Proposition 4.5, it suffices to show that the ∞-category $\mathcal{S}_{\infty}^{\mathrm{fin}}$ of finite spectra is stable. The essence of the matter is now to show that every pushout square in $\mathcal{S}_{\infty}^{\mathrm{fin}}$ is also a pullback square. Every pushout square is obtained from a pushout diagram

$$\begin{tikzcd} W \arrow[r] \arrow[d] & X \arrow[d] \\ Y \arrow[r] & Z \end{tikzcd}$$

in the ∞-category $\mathcal{S}_{*}^{\mathrm{fin}}$ of finite pointed spaces. This pushout square will typically not be homotopy Cartesian $\mathcal{S}_{*}^{\mathrm{fin}}$, but will be *approximately* homotopy Cartesian if the spaces involved are highly connected: this follows from the Blakers-Massey homotopy excision theorem (see for example [25], p. 360). Using the fact that the approximation gets better and better as we iterate the suspension functor Σ (which increases the connectivity of spaces), one can deduce that the image of the above square is a pullback in $\mathcal{S}_{*}^{\mathrm{fin}}$.

Remark 9.10. Let $\mathcal{A}b$ denote the category of abelian groups. For each $n \in \mathbf{Z}$, we let $\pi_n : \mathrm{Sp} \rightarrow \mathrm{N}(\mathcal{A}b)$ be the composition of the shift functor $X \mapsto X[-n]$ with the equivalence $\mathcal{S}_{\infty}^{\heartsuit} \simeq \mathrm{N}(\mathcal{A}b)$. Note that if $n \geq 2$, then π_n can be identified with the composition

$$\mathrm{Sp} \overset{\Omega_*^{\infty}}{\rightarrow} \mathcal{S}_* \overset{\pi_n}{\rightarrow} \mathrm{N}(\mathcal{A}b)$$

where the second map is the usual homotopy group functor. Since Sp is both left and right complete, we conclude that a map $f : X \rightarrow Y$ of spectra is an equivalence if and only if it induces isomorphisms $\pi_n X \rightarrow \pi_n Y$ for all $n \in \mathbf{Z}$.

Proposition 9.11. *The functor $\Omega^{\infty} : (\mathrm{Sp})_{\geq 0} \rightarrow \mathcal{S}$ preserves geometric realizations of simplicial objects.*

Proof. Since the simplicial set $\mathrm{N}(\mathbf{\Delta})$ is weakly contractible, the forgetful functor $\mathcal{S}_* \rightarrow \mathcal{S}$ preserves geometric realizations of simplicial objects (Proposition T.4.4.2.9). It will therefore suffice to prove that the functor $\Omega_*^{\infty} | (\mathrm{Sp})_{\geq 0} \rightarrow \mathcal{S}_*$ preserves geometric realizations of simplicial objects.

For each $n \geq 0$, let $\mathcal{S}^{\geq n}$ denote the full subcategory of $\mathcal{S}$ spanned by the n-connective objects, and let $\mathcal{S}_*^{\geq n}$ be the ∞-category of pointed objects of $\mathcal{S}^{\geq n}$. We observe that $(\mathrm{Sp})_{\geq 0}$ can be identified with the homotopy inverse limit of the tower

$$\ldots \overset{\Omega}{\rightarrow} \mathcal{S}_*^{\geq 1} \overset{\Omega}{\rightarrow} \mathcal{S}_*^{\geq 0}.$$

It will therefore suffice to prove that for every $n \geq 0$, the loop functor $\Omega : \mathcal{S}_*^{\geq n+1} \rightarrow \mathcal{S}_*^{\geq n}$ preserves geometric realizations of simplicial objects.

The ∞-category $\mathcal{S}^{\geq n}$ is the preimage (under $\tau_{\leq n-1}$) of the full subcategory of $\tau_{\leq n-1}\,\mathcal{S}$ spanned by the final objects. Since this full subcategory is stable under geometric realizations of simplicial objects and since $\tau_{\leq n-1}$ commutes with all colimits, we conclude that $\mathcal{S}^{\geq n} \subseteq \mathcal{S}$ is stable under geometric realizations of simplicial objects.

According to Lemmas T.7.2.2.11 and T.7.2.2.10, there is an equivalence of $\mathcal{S}_*^{\geq 1}$ with the ∞-category of group objects $\mathcal{G}\mathrm{rp}(\mathcal{S}_*)$. This restricts to an equivalence of $\mathcal{S}_*^{\geq n+1}$ with $\mathcal{G}\mathrm{rp}(\mathcal{S}_*^{\geq n})$ for all $n \geq 0$. Moreover, under this equivalence, the loop functor Ω can be identified with the composition

$$\mathcal{G}\mathrm{rp}(\mathcal{S}_*^{\geq n}) \subseteq \mathrm{Fun}(\mathrm{N}(\mathbf{\Delta})^{op}, \mathcal{S}_*^{\geq n}) \rightarrow \mathcal{S}_*^{\geq n},$$

where the second map is given by evaluation at the object $[1] \in \mathbf{\Delta}$. This evaluation map commutes with geometric realizations of simplicial objects (Proposition T.5.1.2.2). Consequently, it will suffice to show that $\mathcal{G}\mathrm{rp}(\mathcal{S}_*^{\geq n}) \subseteq \mathrm{Fun}(\mathrm{N}(\mathbf{\Delta})^{op}, \mathcal{S}_*^{\geq n})$ is stable under geometric realizations of simplicial objects.

Without loss of generality, we may suppose $n = 0$; now we are reduced to showing that $\mathcal{G}\mathrm{rp}(\mathcal{S}_*) \subseteq \mathrm{Fun}(\mathrm{N}(\mathbf{\Delta})^{op}, \mathcal{S}_*)$ is stable under geometric realizations of simplicial objects. In view of Lemma T.7.2.2.10, it will suffice to show that $\mathcal{G}\mathrm{rp}(\mathcal{S}) \subseteq \mathcal{S}_{\Delta}$ is stable under geometric realizations of simplicial objects. Invoking Proposition T.7.2.2.4, we are reduced to proving that the formation of geometric realizations in $\mathcal{S}$ commutes with finite products, which follows from Lemma T.5.5.8.11. □

10 Excisive Functors

In order to study the relationship between an ∞-category $\mathcal{C}$ and its stabilization $\mathrm{Stab}(\mathcal{C})$, we need to introduce a bit of terminology.

Definition 10.1. Let $F : \mathcal{C} \to \mathcal{D}$ be a functor between ∞-categories.

- (i) If $\mathcal{C}$ has an initial object $\emptyset$, then we will say that F is *weakly excisive* if $F(\emptyset)$ is a final object of $\mathcal{D}$. We let $\mathrm{Fun}_*(\mathcal{C}, \mathcal{D})$ denote the full subcategory of $\mathrm{Fun}(\mathcal{C}, \mathcal{D})$ spanned by the weakly excisive functors.
- (ii) If $\mathcal{C}$ admits finite colimits, then we will say that F is *excisive* if it is weakly excisive, and F carries pushout squares in $\mathcal{C}$ to pullback squares in $\mathcal{D}$. We let $\mathrm{Exc}(\mathcal{C}, \mathcal{D})$ denotes the full subcategory of $\mathrm{Fun}(\mathcal{C}, \mathcal{D})$ spanned by the excisive functors.

Warning 10.2. Definition 10.1 is somewhat nonstandard: most authors do not require the property the preservation of zero objects in the definition of excisive functors.

Remark 10.3. Let $F : \mathcal{C} \to \mathcal{D}$ be a functor between ∞-categories, and suppose that $\mathcal{C}$ admits finite colimits. If $\mathcal{C}$ is stable, then F is excisive if and only if it is left exact (Proposition 4.4). If instead $\mathcal{D}$ is stable, then F is excisive if and only if it is right exact. In particular, if both $\mathcal{C}$ and $\mathcal{D}$ are stable, then F is excisive if and only if it is exact (Proposition 5.1).

Lemma 10.4. *Let $\mathcal{C}$ and $\mathcal{D}$ be ∞-categories, and assume that $\mathcal{C}$ has an initial object. Then:*

1. *The forgetful functor $\theta : \mathrm{Fun}_*(\mathcal{C}, \mathcal{D}_*) \to \mathrm{Fun}_*(\mathcal{C}, \mathcal{D})$ is a trivial fibration of simplicial sets.*
2. *If $\mathcal{C}$ admits finite colimits, then the forgetful functor $\theta' : \mathrm{Exc}(\mathcal{C}, \mathcal{D}_*) \to \mathrm{Exc}(\mathcal{C}, \mathcal{D})$ is a trivial fibration of simplicial sets.*

Remark 10.5. If the ∞-category $\mathcal{D}$ does not have a final object, then the conclusion of Lemma 10.4 is valid, but degenerate: both of the relevant ∞-categories of functors are empty.

Proof. To prove (1), we first observe that objects of $\mathrm{Fun}_*(\mathcal{C}, \mathcal{D}_*)$ can be identified with maps $F : \mathcal{C} \times \Delta^1 \to \mathcal{D}$ with the following properties:

- (a) For every initial object $C \in \mathcal{C}$, $F(C, 1)$ is a final object of $\mathcal{D}$.
- (b) For every object $C \in \mathcal{C}$, $F(C, 0)$ is a final object of $\mathcal{D}$.

Assume for the moment that (a) is satisfied, and let $\mathcal{C}' \subseteq \mathcal{C} \times \Delta^1$ be the full subcategory spanned by those objects (C, i) for which either $i = 1$, or C is an initial object of $\mathcal{C}$. We observe that (b) is equivalent to the following pair of conditions:

- (b') The functor $F|\,\mathcal{C}'$ is a right Kan extension of $F|\,\mathcal{C} \times \{1\}$.
- (b'') The functor F is a left Kan extension of $F|\,\mathcal{C}'$.

Let $\mathcal{E}$ be the full subcategory of $\mathrm{Fun}(\mathcal{C} \times \Delta^1, \mathcal{D})$ spanned by those functors which satisfy conditions (b') and (b''). Using Proposition T.4.3.2.15, we deduce that the projection $\overline{\theta} : \mathcal{E} \to \mathrm{Fun}(\mathcal{C} \times \{1\}, \mathcal{D})$ is a trivial Kan fibration. Since θ is a pullback of $\overline{\theta}$, we conclude that θ is a trivial Kan fibration. This completes the proof of (1).

To prove (2), we observe that θ' is a pullback of θ (since Proposition T.1.2.13.8 asserts that a square in $\mathcal{D}_*$ is a pullback if and only if the underlying square in $\mathcal{D}$ is a pullback). □

Remark 10.6. Let $\mathcal{C}$ be a pointed ∞-category which admits finite colimits, and $\mathcal{D}$ a pointed ∞-category which admits finite limits. Let $F : \mathrm{Fun}(\mathcal{C}, \mathcal{D}) \to \mathrm{Fun}(\mathcal{C}, \mathcal{D})$ be given by composition with the suspension functor $\mathcal{C} \to \mathcal{C}$, and let $G : \mathrm{Fun}(\mathcal{C}, \mathcal{D}) \to \mathrm{Fun}(\mathcal{C}, \mathcal{D})$ be given by composition with the loop functor $\Omega : \mathcal{D} \to \mathcal{D}$. Then F and G restrict to give homotopy inverse equivalences

$$\begin{tikzcd} \mathrm{Exc}(\mathcal{C}, \mathcal{D}) \arrow[r, shift left, "F"] & \mathrm{Exc}(\mathcal{C}, \mathcal{D}) \arrow[l, shift left, "G"] \end{tikzcd}.$$

Notation 10.7. Let $F : \mathcal{C} \to \mathcal{D}$ be a functor between ∞-categories, and assume that $\mathcal{D}$ admits finite limits. For every commutative square τ:

$$\begin{tikzcd}
W \arrow[r] \arrow[d] & X \arrow[d] \\
Y \arrow[r] & Z
\end{tikzcd}$$

in $\mathcal{C}$, we obtain a commutative square $F(\tau)$:

$$\begin{tikzcd}
F(W) \arrow[r] \arrow[d] & F(X) \arrow[d] \\
F(Y) \arrow[r] & F(Z)
\end{tikzcd}$$

in $\mathcal{D}$. This diagram determines a map $\eta_\tau : F(W) \to F(X) \times_{F(Z)} F(Y)$ in the ∞-category $\mathcal{D}$, which is well-defined up to homotopy. If we suppose further that Y and Z are zero objects of $\mathcal{C}$, that $F(Y)$ and $F(Z)$ are zero objects of $\mathcal{D}$, and that τ is a pushout diagram, then we obtain a map $F(W) \to \Omega F(\Sigma W)$, which we will denote simply by η_W.

Proposition 10.8. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite colimits, $\mathcal{D}$ a pointed ∞-category which admits finite limits, and let $F : \mathcal{C} \to \mathcal{D}$ be a functor which carries zero objects of $\mathcal{C}$ to zero objects of $\mathcal{D}$. The following conditions are equivalent:*

(1) *The functor F is excisive: that is, F carries pushout squares in $\mathcal{C}$ to pullback squares in $\mathcal{D}$.*

(2) *For every object $X \in \mathcal{C}$, the canonical map $\eta_X : F(X) \to \Omega F(\Sigma X)$ is an equivalence in $\mathcal{D}$ (see Notation 10.7).*

Corollary 10.9. *Let $F : \mathcal{C} \to \mathcal{D}$ be a functor between stable ∞-categories. Then F is exact if and only if the following conditions are satisfied:*

(1) *The functor F carries zero objects of $\mathcal{C}$ to zero objects of $\mathcal{D}$.*

(2) *For every object $X \in \mathcal{C}$, the canonical map $\Sigma F(X) \to F(\Sigma X)$ is an equivalence in $\mathcal{D}$.*

Corollary 10.10. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite limits and colimits. Then:*

(1) *If the suspension functor $\Sigma_{\mathcal{C}}$ is fully faithful, then every pushout square in $\mathcal{C}$ is a pullback square.*

(2) *If the loop functor $\Omega_{\mathcal{C}}$ is fully faithful, then every pullback square in $\mathcal{C}$ is a pushout square.*

(3) *If the loop functor $\Omega_{\mathcal{C}}$ is an equivalence of ∞-categories, then $\mathcal{C}$ is stable.*

Proof. Assertion (1) follows by applying Proposition 10.8 to the identity functor $\mathrm{id}_{\mathcal{C}}$, and assertion (2) follows from (1) by passing to the opposite ∞-category. Assertion (3) is an immediate consequence of (1) and (2). □

The proof of Proposition 10.8 makes use of the following lemma:

Lemma 10.11. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite colimits, $\mathcal{D}$ a pointed ∞-category which admits finite limits, and $F : \mathcal{C} \to \mathcal{D}$ a functor which carries zero objects of $\mathcal{C}$ to zero objects of $\mathcal{D}$. Suppose given a pushout diagram τ:*

$$\begin{tikzcd}
W \arrow[r] \arrow[d] & X \arrow[d] \\
Y \arrow[r] & Z
\end{tikzcd}$$

in $\mathcal{C}$. Then there exists a map $\theta_\tau : F(X) \times_{F(Z)} F(Y) \to \Omega F(\Sigma W)$ with the following properties:

(1) *The composition $\theta_\tau \circ \eta_\tau$ is homotopic to η_W. Here η_τ and η_W are defined as in Notation 10.7.*

(2) *Let $\Sigma(\tau)$ denote the induced diagram*

$$\begin{tikzcd}
\Sigma W \arrow[r] \arrow[d] & \Sigma X \arrow[d] \\
\Sigma Y \arrow[r] & \Sigma Z.
\end{tikzcd}$$

Then there is a pullback square

$$\begin{tikzcd}
\eta_{\Sigma(\tau)} \circ \theta_\tau \arrow[r] \arrow[d] & \eta_X \arrow[d] \\
\eta_Y \arrow[r] & \eta_Z
\end{tikzcd}$$

in the ∞-category $\mathrm{Fun}(\Delta^1, \mathcal{D})$ *of morphisms in $\mathcal{D}$.*

Proof. In the ∞-category $\mathcal{C}$, we have the following commutative diagram (in which every square is a pushout):

$$\begin{tikzcd}
W \arrow[r] \arrow[d] & X \arrow[r] \arrow[d] & 0 \arrow[d] & \\
Y \arrow[r] \arrow[d] & X \coprod_W Y \arrow[r] \arrow[d] & 0 \coprod_W Y \arrow[r] \arrow[d] & 0 \arrow[d] \\
0 \arrow[r] & X \coprod_W 0 \arrow[r] \arrow[d] & \Sigma W \arrow[r] \arrow[d] & \Sigma Y \arrow[d] \\
& 0 \arrow[r] & \Sigma X \arrow[r] & \Sigma(X \coprod_W Y).
\end{tikzcd}$$

Applying the functor F, and replacing the upper left square by a pullback, we obtain a new diagram

$$\begin{tikzcd}
F(X) \times_{F(Z)} F(Y) \arrow[r] \arrow[d] & F(X) \arrow[r] \arrow[d] & 0 \arrow[d] & \\
F(Y) \arrow[r] \arrow[d] & F(Z) \arrow[r] \arrow[d] & F(0 \coprod_W Y) \arrow[r] \arrow[d] & 0 \arrow[d] \\
0 \arrow[r] & F(X \coprod_W 0) \arrow[r] \arrow[d] & F(\Sigma(W)) \arrow[r] \arrow[d] & F(\Sigma Y) \arrow[d] \\
& 0 \arrow[r] & F(\Sigma X) \arrow[r] & F(\Sigma Z).
\end{tikzcd}$$

Restricting attention to the large square in the upper left, we obtain the desired map $\theta_\tau : F(X) \times_{F(Z)} F(Y) \to \Omega F(\Sigma W)$. It is easy to verify that θ_τ has the desired properties. □

Proof of Proposition 10.8. The implication (1) ⇒ (2) is obvious. Conversely, suppose that (2) is satisfied. We must show that for every pushout square τ :

$$\begin{tikzcd}
X \arrow[r] \arrow[d] & Y \arrow[d] \\
Z \arrow[r] & Y \coprod_X Z
\end{tikzcd}$$

in the ∞-category $\mathcal{C}$, the induced map η_τ is an equivalence in $\mathcal{D}$. Let θ_τ be as in the statement of Lemma 10.11. Then $\theta_\tau \circ \eta_\tau$ is homotopic to η_X, and is therefore an equivalence (in virtue of assumption (2)). It will therefore suffice to show that η_τ is an equivalence. The preceding argument shows that θ_τ has a right homotopy inverse. To show that θ_τ admits a left homotopy inverse, it will suffice to show that $\eta_{\Sigma\tau} \circ \theta_\tau$ is an equivalence. This follows from the second assertion of Lemma 10.11, since the maps η_Y, η_Z, and $\eta_{Y \coprod_X Z}$ are equivalences (by assumption (2), again). □

Let $\mathcal{C}$ be a (small) pointed ∞-category. Let $\mathcal{P}_*(\mathcal{C})$ be defined as in Lemma 8.26. Lemma 10.4 implies that the canonical map $\mathrm{Fun}_*(\mathcal{C}^{op}, \mathcal{S}_*) \to \mathcal{P}_*(\mathcal{C})$ is a trivial fibration. Consequently, the Yoneda embedding lifts to a fully faithful functor $j' : \mathcal{C} \to \mathrm{Fun}_*(\mathcal{C}^{op}, \mathcal{S}_*)$, which we will refer to as the *pointed Yoneda embedding.* Our terminology is slightly abusive: the functor j' is only well-defined up to a contractible space of choices; we will ignore this ambiguity.

Proposition 10.12. *Let $\mathcal{C}$ be a pointed ∞-category which admits finite colimits and $\mathcal{D}$ an ∞-category which admits finite limits. Then composition with the canonical map $\mathrm{Stab}(\mathcal{D}) \to \mathcal{D}$ induces an equivalence of ∞-categories*

$$\theta : \mathrm{Exc}(\mathcal{C}, \mathrm{Stab}(\mathcal{D})) \to \mathrm{Exc}(\mathcal{C}, \mathcal{D}).$$

Proof. Since the loop functor $\Omega_{\mathcal{D}} : \mathcal{D} \to \mathcal{D}$ is left exact, the domain of θ can be identified with a homotopy limit of the tower

$$\ldots \overset{\circ\Omega_{\mathcal{D}}}{\to} \mathrm{Exc}(\mathcal{C}, \mathcal{D}_*) \overset{\circ\Omega_{\mathcal{D}}}{\to} \mathrm{Exc}(\mathcal{C}, \mathcal{D}_*).$$

Remark 10.6 implies that this tower is essentially constant. Consequently, it will suffice to show that the canonical map $\mathrm{Exc}(\mathcal{C}, \mathcal{D}_*) \to \mathrm{Exc}(\mathcal{C}, \mathcal{D})$ is a trivial fibration of simplicial sets, which follows from Lemma 10.4. □

Example 10.13. Let $\mathcal{C}$ be an ∞-category which admits finite limits, and K an arbitrary simplicial set. Then $\mathrm{Fun}(K, \mathcal{C})$ admits finite limits (Proposition T.5.1.2.2). We have a canonical isomorphism $\mathrm{Fun}(K, \mathcal{C})_* \simeq \mathrm{Fun}(K, \mathcal{C}_*)$, and the loop functor on $\mathrm{Fun}(K, \mathcal{C})_*$ can be identified with the functor given by composition with $\Omega : \mathcal{C}_* \to \mathcal{C}_*$. It follows that there is a canonical equivalence of ∞-categories

$$\mathrm{Stab}(\mathrm{Fun}(K, \mathcal{C})) \simeq \mathrm{Fun}(K, \mathrm{Stab}(\mathcal{C})).$$

In particular, $\mathrm{Stab}(\mathcal{P}(K))$ can be identified with $\mathrm{Fun}(K, \mathrm{Sp})$.

We can apply Proposition 10.12 to give another description of the ∞-category $\mathrm{Stab}(\mathcal{C})$.

Lemma 10.14. *Let $\mathcal{C}$ be an ∞-category which admits finite colimits, let $f : \mathcal{C} \to \mathcal{C}_*$ be a left adjoint to the forgetful functor, and let $\mathcal{D}$ be a stable ∞-category. Then composition with f induces an equivalence of ∞-categories $\phi : \mathrm{Exc}(\mathcal{C}_*, \mathcal{D}) \to \mathrm{Exc}(\mathcal{C}, \mathcal{D})$.*

Proof. Consider the composition

$$\theta : \mathrm{Fun}(\mathcal{C}, \mathcal{D}) \times \mathcal{C}_* \subseteq \mathrm{Fun}(\mathcal{C}, \mathcal{D}) \times \mathrm{Fun}(\Delta^1, \mathcal{C}) \to \mathrm{Fun}(\Delta^1, \mathcal{D}) \overset{\mathrm{coker}}{\to} \mathcal{D}\,.$$

We can identify θ with a map $\mathrm{Fun}(\mathcal{C}, \mathcal{D}) \to \mathrm{Fun}(\mathcal{C}_*, \mathcal{D})$. Since the collection of pullback squares in $\mathcal{D}$ is a stable subcategory of $\mathrm{Fun}(\Delta^1 \times \Delta^1, \mathcal{D})$, we conclude θ restricts to a map $\psi : \mathrm{Exc}(\mathcal{C}, \mathcal{D}) \to \mathrm{Exc}(\mathcal{C}_*, \mathcal{D})$. It is not difficult to verify that ψ is a homotopy inverse to ϕ. □

Proposition 10.15. *Let $\mathcal{C}$ be an ∞-category which admits finite colimits, and let $\mathcal{D}$ be an ∞-category which admits finite limits. Then there is a canonical isomorphism $\mathrm{Exc}(\mathcal{C}_*, \mathcal{D}) \simeq \mathrm{Exc}(\mathcal{C}, \mathrm{Stab}(\mathcal{D}))$ in the homotopy category of ∞-categories.*

Proof. Combining Lemma 10.14 and Proposition 10.12, we obtain a diagram of equivalences

$$\mathrm{Exc}(\mathcal{C}_*, \mathcal{D}) \leftarrow \mathrm{Exc}(\mathcal{C}_*, \mathrm{Stab}(\mathcal{D})) \to \mathrm{Exc}(\mathcal{C}, \mathrm{Stab}(\mathcal{D})).$$

□

Corollary 10.16. *Let $\mathcal{D}$ be an ∞-category which admits finite limits. Then there is a canonical equivalence* $\mathrm{Stab}(\mathcal{D}) \simeq \mathrm{Exc}(\mathcal{S}_*^{\mathrm{fin}}, \mathcal{D})$ *in the homotopy category of ∞-categories.*

Proof. Combine Proposition 10.15, Remark 9.4, and Remark 10.3. □

Corollary 10.17. *The ∞-category of spectra is equivalent to the ∞-category* $\mathrm{Exc}(\mathcal{S}_*^{fin}, \mathcal{S})$.

Remark 10.18. Corollary 10.17 provides a very explicit model for spectra. Namely, we can identify a spectrum with an excisive functor $F : \mathcal{S}_*^{\mathrm{fin}} \to \mathcal{S}$. We should think of F as a *homology theory A*. More precisely, given a pair of finite spaces $X_0 \subseteq X$, we can define the relative homology group $A_n(X, X_0)$ to be $\pi_n F(X/X_0)$, where X/X_0 denotes the pointed space obtained from X by collapsing X_0 to a point (here the homotopy group is taken with base point provided by the map $* \simeq F(*) \to F(X/X_0)$). The assumption that F is excisive is precisely what is needed to guarantee the existence of the usual excision exact sequences for the homology theory A.

11 Filtered Objects and Spectral Sequences

Suppose given a sequence of objects

$$\ldots \to X(-1) \xrightarrow{f^0} X(0) \xrightarrow{f^1} X(1) \to \ldots.$$

in a stable ∞-category $\mathcal{C}$. Suppose further that $\mathcal{C}$ is equipped with a t-structure, and that the heart of $\mathcal{C}$ is equivalent to the nerve of an abelian category $\mathcal{A}$. In this section, we will construct a spectral sequence taking values in the abelian category $\mathcal{A}$, with the E_1-page described by the formula

$$E_1^{p,q} = \pi_{p+q} \operatorname{coker}(f^p) \in \mathcal{A}\,.$$

Under appropriate hypotheses, we will see that this spectral sequence converges to the homotopy groups of the colimit $\varinjlim(X(i))$.

Our first step is to construct some auxiliary objects in $\mathcal{C}$.

Definition 11.1. Let $\mathcal{C}$ be a pointed ∞-category, and let $\mathcal{J}$ be a linearly ordered set. We let $\mathcal{J}^{[1]}$ denote the partially ordered set of pairs of elements $i \leq j$ of $\mathcal{J}$, where $(i,j) \leq (i',j')$ if $i \leq j$ and $i' \leq j'$. An $\mathcal{J}$-*complex* in $\mathcal{C}$ is a functor $F : \mathrm{N}(\mathcal{J}^{[1]}) \to \mathcal{C}$ with the following properties:

(1) For each $i \in \mathcal{J}$, $F(i,i)$ is a zero object of $\mathcal{C}$.

(2) For every $i \leq j \leq k$, the associated diagram

$$\begin{tikzcd} F(i,j) \arrow[r] \arrow[d] & F(i,k) \arrow[d] \\ F(j,j) \arrow[r] & F(j,k) \end{tikzcd}$$

is a pushout square in $\mathcal{C}$.

We let $\mathrm{Gap}(\mathcal{J}, \mathcal{C})$ denote the full subcategory of $\mathrm{Fun}(\mathrm{N}(\mathcal{J}^{[1]}), \mathcal{C})$ spanned by the $\mathcal{J}$-complexes in $\mathcal{C}$.

Remark 11.2. Let $F \in \mathrm{Gap}(\mathbf{Z}, \mathcal{C})$ be a $\mathbf{Z}$-complex in a stable ∞-category $\mathcal{C}$. For each $n \in \mathbf{Z}$, the functor F determines pushout square

$$\begin{tikzcd} F(n-1,n) \arrow[r] \arrow[d] & F(n-1,n+1) \arrow[d] \\ 0 \arrow[r] & F(n,n+1), \end{tikzcd}$$

hence a boundary $\delta : F(n, n+1) \to F(n-1, n)[1]$. If we set $C_n = F(n-1, n)[-n]$, then we obtain a sequence of maps

$$\ldots \to C_1 \overset{d_1}{\to} C_0 \overset{d_0}{\to} C_{-1} \to \ldots$$

in the homotopy category $\mathrm{h}\mathcal{C}$. The commutative diagram

$$\begin{tikzcd}
F(n, n+1) \arrow[r, "\delta"] & F(n-2, n)[1] \arrow[r] \arrow[d] & F(n-1, n)[1] \arrow[d, "\delta"] \\
0 \arrow[r, "\sim"] & F(n-1, n-1)[1] \arrow[r] & F(n-2, n-1)[2].
\end{tikzcd}$$

shows that $d_{n-1} \circ d_n \simeq 0$, so that $(C_\bullet, d_\bullet)$ can be viewed as a chain complex in the triangulated category $\mathrm{h}\mathcal{C}$. This motivates the terminology of Definition 11.1.

Lemma 11.3. *Let $\mathcal{C}$ be a pointed ∞-category which admits pushouts. Let $\mathcal{I} = \mathcal{I}_0 \cup \{-\infty\}$ be a linearly ordered set containing a least element $-\infty$. We regard $\mathcal{I}_0$ as a linearly ordered subset of $\mathcal{I}^{[1]}$ via the embedding*

$$i \mapsto (-\infty, i).$$

Then the restriction map $\mathrm{Gap}(\mathcal{I}, \mathcal{C}) \to \mathrm{Fun}(\mathrm{N}(\mathcal{I}_0), \mathcal{C})$ is an equivalence of ∞-categories.

Proof. Let $\mathcal{J} = \{(i, j) \in \mathcal{I}^{[1]} : (i = -\infty) \vee (i = j)\}$. We now make the following observations:

(1) A functor $F : \mathrm{N}(\mathcal{I}^{[1]}) \to \mathcal{C}$ is a complex if and only if F is a left Kan extension of $F|\,\mathrm{N}(\mathcal{J})$, and $F(i, i)$ is a zero object of $\mathcal{C}$ for all $i \in \mathcal{I}$.

(2) Any functor $F_0 : \mathrm{N}(\mathcal{J}) \to \mathcal{C}$ admits a left Kan extension to $\mathrm{N}(\mathcal{I}^{[1]})$ (use Lemma T.4.3.2.13 and the fact that $\mathcal{C}$ admits pushouts).

(3) A functor $F_0 : \mathrm{N}(\mathcal{J}) \to \mathcal{C}$ has the property that $F_0(i, i)$ is a zero object, for every $i \in \mathcal{I}$, if and only if F_0 is a right Kan extension of $F_0|\,\mathrm{N}(\mathcal{I}_0)$.

(4) Any functor $F_0 : \mathrm{N}(\mathcal{I}_0) \to \mathcal{C}$ admits a right Kan extension to $\mathrm{N}(\mathcal{J})$ (use Lemma T.4.3.2.13 and the fact that $\mathcal{C}$ has a final object).

The desired conclusion now follows immediately from Proposition T.4.3.2.15. □

Remark 11.4. Let $\mathcal{C}$ be a pointed ∞-category which admits pushouts (for example, a stable ∞-category). For each $n \geq 0$, let $\mathrm{Gap}^0([n], \mathcal{C})$ be the largest Kan complex contained in $\mathrm{Gap}([n], \mathcal{C})$. Then the assignment

$$[n] \mapsto \mathrm{Gap}([n], \mathcal{C})$$

determines a simplicial object in the category $\mathcal{K}\mathrm{an}$ of Kan complexes. We can then define the *Waldhausen K-theory of* $\mathcal{C}$ to be a geometric realization of this bisimplicial set (for example, the associated diagonal simplicial set). In the special case where A is an A_∞-ring and $\mathcal{C}$ is the smallest stable subcategory of Mod_A which contains A, this definition recovers the usual K-theory of A. We refer the reader to [72] for a related construction.

Construction 11.5. Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure, such that the heart of $\mathcal{C}$ is equivalent to the nerve of an abelian category $\mathcal{A}$. Let $X \in \mathrm{Gap}(\mathbf{Z}, \mathcal{C})$. We observe that for every triple of integers $i \leq j \leq k$, there is a long exact sequence

$$\ldots \to \pi_n X(i, j) \to \pi_n X(i, k) \to \pi_n X(j, k) \overset{\delta}{\to} \pi_{n-1} X(i, j) \to \ldots$$

in the abelian category $\mathcal{A}$. For every $p, q \in \mathbf{Z}$ and every $r \geq 1$, we define the object $E_r^{p,q} \in \mathcal{A}$ by the formula

$$E_r^{p,q} = \operatorname{im}(\pi_{p+q} X(p-r, p) \to \pi_{p+q} X(p-1, p+r-1)).$$

There is a differential $d_r : E_r^{p,q} \to E_r^{p-r,q+r-1}$, uniquely determined by the requirement that the diagram

$$\begin{tikzcd}
\pi_{p+q} X(p-r,p) \arrow[r] \arrow[d, "\delta"] & E_r^{p,q} \arrow[r] \arrow[d, "d_r"] & \pi_{p+q} X(p-1,p+r-1) \arrow[d, "\delta"] \\
\pi_{p+q-1} X(p-2r,p-r) \arrow[r] & E_r^{p-r,q+r-1} \arrow[r] & \pi_{p+q-1} X(p-r-1,p-1)
\end{tikzcd}$$

be commutative.

Proposition 11.6. *Let $X \in \operatorname{Gap}(\mathbf{Z}, \mathcal{C})$ be as in Construction 11.5. Then:*

(1) *For each $r \geq 1$, the composition $d_r \circ d_r$ is zero.*

(2) *There are canonical isomorphisms*

$$E_{r+1}^{p,q} \simeq \ker(d_r : E_r^{p,q} \to E_r^{p-r,q+r-1}) / \operatorname{im}(d_r : E_r^{p+r,q-r+1} \to E_r^{p,q}).$$

Consequently, $\{E_r^{p,q}, d_r\}$ is a spectral sequence (with values in the abelian category $\mathcal{A}$).

Remark 11.7. For fixed $q \in \mathbf{Z}$, the complex $(E_1^{\bullet,q}, d_1)$ in $\mathcal{A}$ can be obtained from the $\mathrm{h}\mathcal{C}$-valued chain complex $C_\bullet$ described in Remark 11.2 by applying the cohomological functor π_q.

Proof. We have a commutative diagram

$$\begin{tikzcd}
 & \pi_{p+q} X(p-r-1,p) \arrow[d] & \\
\pi_{p+q+1} X(p,p+r) \arrow[r, "\delta"] \arrow[d] & \pi_{p+q} X(p-r,p) \arrow[r, "\delta"] \arrow[d] & \pi_{p+q-1} X(p-2r,p-r) \arrow[d] \\
E_r^{p+r,q-r+1} \arrow[r, "d_r"] \arrow[d] & E_r^{p,q} \arrow[r, "d_r"] \arrow[d] & E_r^{p-r,q+r-1} \arrow[d] \\
\pi_{p+q+1} X(p+r-1,p+2r-1) \arrow[r, "\delta"] & \pi_{p+q} X(p-1,p+r-1) \arrow[r, "\delta"] \arrow[d] & \pi_{p+q-1} X(p-r-1,p-1) \\
 & \pi_{p+q} X(p-1,p+r). &
\end{tikzcd}$$

Since the upper left vertical map is an epimorphism, (1) will follow provided that we can show that the composition

$$\pi_{p+q+1} X(p,p+r) \xrightarrow{\delta} \pi_{p+q} X(p-r,p) \xrightarrow{\delta} \pi_{p+q-1} X(p-2r,p-r)$$

is zero. This follows immediately from the commutativity of the diagram

$$\begin{tikzcd}
X(p,p+r) \arrow[r, "\delta"] & X(p-2r,p)[1] \arrow[r] \arrow[d] & X(p-r,p)[1] \arrow[d, "\delta"] \\
0 \arrow[r, "\sim"] & X(p-r,p-r)[1] \arrow[r] & X(p-2r,p-r)[2].
\end{tikzcd}$$

We next claim that the composite map

$$\phi : \pi_{p+q} X(p-r-1, p) \to E_r^{p,q} \xrightarrow{d_r} E_r^{p-r,q+r-1}$$

is zero. Because $E_r^{p-r,q+r-1} \to \pi_{p+q-1} X(p-r-1, p-1)$ is a monomorphism, this follows from the commutativity of the diagram

$$\begin{tikzcd}
\pi_{p+q} X(p-r-1, p) \arrow[r] \arrow[d] & \pi_{p+q-1} X(p-2r, p-r-1) \arrow[d] \\
\pi_{p+q} X(p-r, p) \arrow[r] \arrow[d] & \pi_{p+q-1} X(p-2r, p-r) \arrow[d] \\
E_r^{p,q} \arrow[r, "d_r"] & E_r^{p-r,q+r-1} \arrow[d] \\
 & \pi_{p+q-1} X(p-r-1, p-1),
\end{tikzcd}$$

since the composition of the left vertical line factors through $\pi_{p+q-1} X(p-r-1, p-r-1) \simeq 0$. A dual argument shows that the composition

$$E_r^{p+r,q+r-1} \xrightarrow{d_r} E_r^{p,q} \to \pi_{p+q} X(p-1, p+r)$$

is zero as well.

Let $Z = \ker(d_r : E_r^{p,q} \to E_r^{p-r,q+r-1})$ and $B = \operatorname{im}(d_r : E_r^{p+r,q-r+1} \to E_r^{p,q})$. The above arguments yield a sequence of morphisms

$$\pi_{p+q} X(p-r-1, p) \xrightarrow{\phi} Z \xrightarrow{\phi'} Z/B \xrightarrow{\psi'} E_r^{p,q}/B \xrightarrow{\psi} \pi_{p+q} X(p-1, p+r).$$

To complete the proof of (2), it will suffice to show that $\phi' \circ \phi$ is an epimorphism and that $\psi \circ \psi'$ is a monomorphism. By symmetry, it will suffice to prove the first assertion. Since ϕ' is evidently an epimorphism, we are reduced to showing that ϕ is an epimorphism.

Let K denote the kernel of the composite map

$$\pi_{p+q} X(p-r, p) \to E_r^{p,q} \xrightarrow{d_r} E_r^{p-r,q+r-1} \to \pi_{p+q-1} X(p-r-1, p-1),$$

so that the canonical map $K \to Z$ is an epimorphism. Choose a diagram

$$\begin{tikzcd}
\widetilde{K} \arrow[r, "f"] \arrow[d, "g"] & K \arrow[r] \arrow[d] & 0 \arrow[d] \\
\pi_{p+q} X(p-r, p-1) \arrow[r] & \pi_{p+q-1} X(p-r-1, p-r) \arrow[r] & \pi_{p+q-1} X(p-r-1, p-1)
\end{tikzcd}$$

where the square on the left is a pullback. The exactness of the bottom row implies that f is an epimorphism. Let f' denote the composition

$$\widetilde{K} \xrightarrow{g} \pi_{p+q} X(p-r, p-1) \to \pi_{p+q} X(p-r, p).$$

The composition

$$\widetilde{K} \xrightarrow{f'} \pi_{p+q} X(p-r, p) \to \pi_{p+q} X(p-1, p+r)$$

factors through $\pi_{p+q}X(p-1,p-1) \simeq 0$. Since $E_r^{p,q} \to \pi_{p+q}X(p-1,p+r)$ is a monomorphism, we conclude that the composition $\widetilde{K} \xrightarrow{f'} \pi_{p+q}X(p-r,p) \to E_r^{p,q}$ is the zero map. It follows that the composition

$$\widetilde{K} \xrightarrow{f-f'} \pi_{p+q}X(p-r,p) \to Z$$

coincides with the composition $\widetilde{K} \xrightarrow{f} K \to Z$, and is therefore an epimorphism.

Form a diagram

$$\begin{tikzcd}
\overline{K} \arrow[r, "f''"] \arrow[d] & \widetilde{K} \arrow[r] \arrow[d, "f-f'"] & 0 \arrow[d] \\
\pi_{p+q}X(p-r-1,p) \arrow[r] & \pi_{p+q}X(p-r,p) \arrow[r] & \pi_{n-1}X(p-r-1,p-r)
\end{tikzcd}$$

where the left square is a pullback. Since the bottom line is exact, we conclude that f'' is an epimorphism, so that the composition

$$\overline{K} \xrightarrow{f''} \widetilde{K} \xrightarrow{f-f'} \pi_{p+q}X(p-r,p) \to Z$$

is an epimorphism. This map coincides with the composition

$$\overline{K} \to \pi_{p+q}X(p-r-1,p) \xrightarrow{\phi} Z,$$

so that ϕ is an epimorphism as well. □

Definition 11.8. Let $\mathcal{C}$ be a stable ∞-category. A *filtered object* of $\mathcal{C}$ is a functor $X : \mathrm{N}(\mathbf{Z}) \to \mathcal{C}$.

Suppose that $\mathcal{C}$ is equipped with a t-structure, and let $X : \mathrm{N}(\mathbf{Z}) \to \mathcal{C}$ be a filtered object of $\mathcal{C}$. According to Lemma 11.3, we can extend X to a complex in $\mathrm{Gap}(\mathbf{Z} \cup \{-\infty\}, \mathcal{C})$. Let $\overline{X}$ be the associated object of $\mathrm{Gap}(\mathbf{Z}, \mathcal{C})$, and let $\{E_r^{p,q}, d_r\}_{r \geq 1}$ be the spectral sequence described in Construction 11.5 and Proposition 11.6. We will refer to $\{E_r^{p,q}, d_r\}_{r\geq 1}$ as the *spectral sequence associated to the filtered object* X.

Remark 11.9. In the situation of Definition 11.8, Lemma 11.3 implies that $\overline{X}$ is determined up to contractible ambiguity by X. It follows that the spectral sequence $\{E_r^{p,q}, d_r\}_{r\geq 1}$ is independent of the choice of $\overline{X}$, up to canonical isomorphism.

Example 11.10. Let $\mathcal{A}$ be a sufficiently nice abelian category, and let $\mathcal{C}$ be the derived ∞-category of $\mathcal{A}$ (see §S.13). Let $\mathrm{Fun}(\mathrm{N}(\mathbf{Z}), \mathcal{C})$ be the ∞-category of filtered objects of $\mathcal{C}$. Then the homotopy category $\mathrm{hFun}(\mathrm{N}(\mathbf{Z}), \mathcal{C})$ can be identified with the classical *filtered derived category* of $\mathcal{A}$, obtained from the category of filtered complexes of objects of $\mathcal{A}$ by inverting all filtered quasi-isomorphisms. In this case, Definition 11.8 recovers the usual spectral sequence associated to a filtered complex.

Our next goal is to establish the convergence of the spectral sequence of Definition 11.8. We will treat only the simplest case, which will be sufficient for our applications.

Definition 11.11. Let $\mathcal{C}$ be an ∞-category. We will say that $\mathcal{C}$ *admits sequential colimits* if every diagram $\mathrm{N}(\mathbf{Z}_{\geq 0}) \to \mathcal{C}$ has a colimit in $\mathcal{C}$.

If $\mathcal{C}$ is stable and admits sequential colimits, we will say that a t-structure on $\mathcal{C}$ is *compatible with sequential colimits* if the full subcategory $\mathcal{C}_{\leq 0}$ is stable under the colimits of diagrams indexed by $\mathrm{N}(\mathbf{Z}_{\geq 0})$.

Remark 11.12. Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure, so that the heart of $\mathcal{C}$ is equivalent to (the nerve of) an abelian category $\mathcal{A}$. Suppose that $\mathcal{C}$ admits sequential colimits. Then $\mathcal{C}_{\geq 0}$ admits sequential colimits, so that $\mathrm{N}(\mathcal{A})$, being a localization of $\mathcal{C}_{\geq 0}$, also admits sequential colimits. If the t-structure on $\mathcal{C}$ is compatible with sequential colimits, then the inclusion $\mathrm{N}(\mathcal{A}) \subseteq \mathcal{C}$ and the homological functors $\{\pi_n : \mathcal{C} \to \mathrm{N}(\mathcal{A})\}_{n \in \mathbf{Z}}$ preserve sequential colimits. It follows that sequential colimits in the abelian category $\mathcal{A}$ are exact: in other words, the direct limit of a sequence of monomorphisms in $\mathcal{A}$ is again a monomorphism.

Proposition 11.13. *Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure, and let $X : \mathrm{N}(\mathbf{Z}) \to \mathcal{C}$ be a filtered object of $\mathcal{C}$. Assume that $\mathcal{C}$ admits sequential colimits, and that the t-structure on $\mathcal{C}$ is compatible with sequential colimits. Suppose furthermore that $X(n) \simeq 0$ for $n \ll 0$. Then the associated spectral sequence (Definition 11.8) converges*

$$E_r^{p,q} \Rightarrow \pi_{p+q} \varinjlim(X).$$

Proof. Let $\mathcal{A}$ be an abelian category such that the heart of $\mathcal{C}$ is equivalent to (the nerve of) $\mathcal{A}$. The convergence assertion of the Proposition has the following meaning:

(i) For fixed p and q, the differentials $d_r : E_r^{p,q} \to E_r^{p-r,q+r-1}$ vanish for $r \gg 0$.

Consequently, for sufficiently large r we obtain a sequence of epimorphisms

$$E_r^{p,q} \to E_{r+1}^{p,q} \to E_{r+2}^{p,q} \to \ldots$$

Let $E_\infty^{p,q}$ denote the colimit of this sequence (in the abelian category $\mathcal{A}$).

(ii) Let $n \in \mathbf{Z}$, and let $A_n = \pi_n \varinjlim(X)$. Then there exists a filtration

$$\ldots \subseteq F^{-1}A_n \subseteq F^0 A_n \subseteq F^1 A_n \subseteq \ldots$$

of A_n, with $F^p A_n \simeq 0$ for $p \ll 0$, and $\varinjlim(F^p A_n) \simeq A_n$.

(iii) For every $p, q \in \mathbf{Z}$, there exists an isomorphism $E_\infty^{p,q} \simeq F^p A_{p+q}/F^{p-1}A_{p+q}$ in the abelian category $\mathcal{A}$.

To prove (i), (ii), and (iii), we first extend X to an object $\overline{X} \in \mathrm{Gap}(\mathbf{Z} \cup \{-\infty\}, \mathcal{C})$, so that for each $n \in \mathbf{Z}$ we have $X(n) = \overline{X}(-\infty, n)$. Without loss of generality, we may suppose that $X(n) \simeq *$ for $n < 0$. This implies that $\overline{X}(i,j) \simeq *$ for $i, j < 0$. It follows that $E_r^{p-r,q+r-1}$, as a quotient $\pi_{p+q}\overline{X}(p-2r, p-r)$, is zero for $r > p$. This proves (i).

To satisfy (ii), we set $F^p A_n = \mathrm{im}(\pi_n X(p) \to \pi_n \varinjlim(X))$. It is clear that $F^p A_n \simeq *$ for $p < 0$, and the isomorphsim $\varinjlim F^p A_n \simeq A_n$ follows from the compatibility of the homological functor π_n with sequential colimits (Remark 11.12).

To prove (iii), we note that for $r > p$, the object $E_r^{p,q}$ can be identified with the image of the map $\pi_{p+q}X(p) \simeq \pi_{p+q}\overline{X}(p-r,p) \to \pi_{p+q}\overline{X}(p-1,p+r)$. Let $Y = \varinjlim_r \overline{X}(p-1,p+r)$. It follows that $E_\infty^{p,q}$ can be identified with the image of the map $\pi_{p+q}X(p) \xrightarrow{f} \pi_{p+q}Y$. We have a distinguished triangle

$$X(p-1) \to \varinjlim(X) \to Y \to X(p-1)[1],$$

which induces an exact sequence

$$0 \to F^{p-1}A_{p+q} \to A_{p+q} \xrightarrow{f'} \pi_{p+q}Y.$$

We have a commutative triangle

$$\begin{tikzcd} & A_{p+q} \arrow[dr, "f'"] & \\ \pi_{p+q}X(p) \arrow[ur, "g"] \arrow[rr, "f"] & & \pi_{p+q}Y. \end{tikzcd}$$

Since the image of g is $F^p A_{p+q}$, we obtain canonical isomorphisms

$$E_\infty^{p,q} \simeq \mathrm{im}(f) \simeq \mathrm{im}(f'|F^p A_{p+q}) \simeq F^p A_{p+q}/\ker(f') \simeq F^p A_{p+q}/F^{p-1}A_{p+q}.$$

This completes the proof. □

12 The ∞-Categorical Dold-Kan Correspondence

Let $\mathcal{A}$ be an abelian category. Then the classical Dold-Kan correspondence (see [75]) asserts that the category $\mathrm{Fun}(\mathbf{\Delta}^{op}, \mathcal{A})$ of simplicial objects of $\mathcal{A}$ is equivalent to the category $\mathrm{Ch}_{\geq 0}(\mathcal{A})$ of (homologically) nonnegatively graded chain complexes

$$\ldots \xrightarrow{d} A_1 \xrightarrow{d} A_0 \rightarrow 0.$$

In this section, we will prove an analogue of this result when the abelian category $\mathcal{A}$ is replaced by a stable ∞-category.

We begin by observing that if $X_\bullet$ is a simplicial object in a stable ∞-category $\mathcal{C}$, then $X_\bullet$ determines a simplicial object of the homotopy category $\mathrm{h}\mathcal{C}$. The category $\mathrm{h}\mathcal{C}$ is not abelian, but it is additive and has the following additional property (which follows easily from the fact that $\mathrm{h}\mathcal{C}$ admits a triangulated structure):

$(*)$ If $i : X \rightarrow Y$ is a morphism in $\mathrm{h}\mathcal{C}$ which admits a left inverse, then there is an isomorphism $Y \simeq X \oplus X'$ such that i is identified with the map $(\mathrm{id}, 0)$.

These conditions are sufficient to construct a Dold-Kan correspondence in $\mathrm{h}\mathcal{C}$. Consequently, every simplicial object $X_\bullet$ of $\mathcal{C}$ determines a chain complex

$$\ldots \rightarrow C_1 \rightarrow C_0 \rightarrow 0$$

in the homotopy category $\mathrm{h}\mathcal{C}$. In §11, we described another construction which gives rise to the same type of data. More precisely, Lemma 11.3 and Remark 11.2 show that every $\mathbf{Z}_{\geq 0}$-filtered object

$$Y(0) \xrightarrow{f_1} Y(1) \xrightarrow{f_2} \ldots$$

determines a chain complex $C_\bullet$ with values in $\mathrm{h}\mathcal{C}$, where $C_n = \mathrm{coker}(f_i)[-n]$. This suggests a relationship between filtered objects of $\mathcal{C}$ and simplicial objects of $\mathcal{C}$. Our goal in this section is to describe this relationship in detail. Our main result, Theorem 12.8, asserts that the ∞-category of simplicial objects of $\mathcal{C}$ is equivalent to a suitable ∞-category of (increasingly) filtered objects of $\mathcal{C}$. The proof will require several preliminaries.

Lemma 12.1. *Let $\mathcal{C}$ be a stable ∞-category. A square*

$$\begin{tikzcd}
X' \arrow[r] \arrow[d, "f'"] & X \arrow[d, "f"] \\
Y' \arrow[r] & Y
\end{tikzcd}$$

in $\mathcal{C}$ is a pullback if and only if the induced map $\alpha : \mathrm{coker}(f') \rightarrow \mathrm{coker}(f)$ is an equivalence.

Proof. Form an expanded diagram

$$\begin{tikzcd}
X' \arrow[r] \arrow[d, "f'"] & X \arrow[r] \arrow[d, "f"] & 0 \arrow[d] \\
Y' \arrow[r] & Y \arrow[r] & \mathrm{coker}(f)
\end{tikzcd}$$

where the right square is a pushout. Since $\mathcal{C}$ is stable, the right square is also a pullback. Lemma T.4.4.2.1 implies that the left square is a pullback if and only if the outer square is a pullback, which is in turn equivalent to the assertion that α is an equivalence. □

Lemma 12.2. *Let $\mathcal{C}$ be a stable ∞-category, let K be a simplicial set, and suppose that $\mathcal{C}$ admits K-indexed colimits. Let $\overline{\alpha} : K^{\triangleright} \times \Delta^1 \rightarrow \mathcal{C}$ be a natural transformation between a pair of diagrams $\overline{p}, \overline{q} : K^{\triangleright} \rightarrow \mathcal{C}$. Then $\overline{\alpha}$ is a colimit diagram if and only if $\mathrm{coker}(\overline{\alpha}) : K^{\triangleright} \rightarrow \mathcal{C}$ is a colimit diagram.*

Proof. Let $p = \overline{p}|K$, $q = \overline{q}|K$, and $\alpha = \overline{\alpha}|K \times \Delta^1)$. Since $\mathcal{C}$ admits K-indexed colimits, there exist colimit diagrams $\overline{p}', \overline{q}' : K^{\triangleright} \to \mathcal{C}$ extending p and q, respectively. We obtain a square

$$\begin{tikzcd}
\overline{p}' \arrow[r] \arrow[d] & \overline{p} \arrow[d] \\
\overline{q}' \arrow[r] & \overline{q}
\end{tikzcd}$$

in the ∞-category $\mathrm{Fun}(K^{\triangleright}, \mathcal{C})$. Let ∞ denote the cone point of $K^{\triangleright}$. Using Corollary T.4.2.3.10, we deduce that α is a colimit diagram if and only if the induced square

$$\begin{tikzcd}
\overline{p}'(\infty) \arrow[r] \arrow[d, "f'"] & \overline{p}(\infty) \arrow[d, "f"] \\
\overline{q}'(\infty) \arrow[r] & \overline{q}(\infty)
\end{tikzcd}$$

is a pushout. According to Lemma 12.1, this is equivalent to the assertion that the induced map $\beta : \mathrm{coker}(f') \to \mathrm{coker}(f)$ is an equivalence. We conclude by observing that β can be identified with the natural map

$$\varinjlim(\mathrm{coker}(\alpha)) \to \mathrm{coker}(\overline{\alpha})(\infty),$$

which is an equivalence if and only if $\mathrm{coker}(\overline{\alpha})$ is a colimit diagram. □

Our next result is an analogue of Proposition S.4.4 which applies to cubical diagrams of higher dimension.

Proposition 12.3. *Let $\mathcal{C}$ be a stable ∞-category, and let $\sigma : (\Delta^1)^n \to \mathcal{C}$ be a diagram. Then σ is a colimit diagram if and only if σ is a limit diagram.*

Proof. By symmetry, it will suffice to show that if σ is a colimit diagram, then σ is also a limit diagram. We work by induction on n. If $n = 0$, then we must show that every initial object of $\mathcal{C}$ is also final, which follows from the assumption that $\mathcal{C}$ has a zero object. If $n > 0$, then we may identify σ with a natural transformation $\alpha : \sigma' \to \sigma''$ in the ∞-category $\mathrm{Fun}((\Delta^1)^{n-1}, \mathcal{C})$. Assume that σ is a colimit diagram. Using Lemma 12.2, we deduce that $\mathrm{coker}(\alpha)$ is a colimit diagram. Since $\mathrm{coker}(\alpha) \simeq \ker(\alpha)[1]$, we conclude that $\ker(\alpha)$ is a colimit diagram. Applying the inductive hypothesis, we deduce that $\ker(\alpha)$ is a limit diagram. The dual of Lemma 12.2 now implies that σ is a limit diagram, as desired. □

Lemma 12.4. *Fix $n \geq 0$, and let S be a subset of the open interval $(0,1)$ of cardinality $\leq n$. Let Y be the set of all sequences of real numbers $0 \leq y_1 \leq \ldots \leq y_n \leq 1$ such that $S \subseteq \{y_1, \ldots, y_n\}$. Then Y is a contractible topological space.*

Proof. Let S have cardinality $m \leq n$, and let Z denote the set of sequences of real numbers $0 \leq z_1 \leq \ldots \leq z_{n-m} \leq 1$. Then Z is homeomorphic to a topological $(n-m)$-simplex. Moreover, there is a homeomorphism $f : Z \to Y$, which carries a sequence $\{z_i\}$ to a suitable reordering of the sequence $\{z_i\} \cup S$. □

Lemma 12.5. *Let $n \leq 0$, let $\mathbf{\Delta}_{\leq n}$ $\mathbf{\Delta}_{\leq n}$ denote the full subcategory of $\mathbf{\Delta}$ spanned by the objects $\{[m]\}_{0 \leq m \leq n}$, and let $\mathcal{I}$ denote the full subcategory of $(\mathbf{\Delta}_{\leq n})_{/[n]}$ spanned by the* injective *maps $[m] \to [n]$. Then the induced map*

$$\mathrm{N}(\mathcal{I})^{op} \to \mathrm{N}(\mathbf{\Delta}_{\leq n})^{op}$$

is cofinal.

Proof. Fix $m \leq n$, and let $\mathcal{J}$ denote the category of diagrams

$$[m] \leftarrow [k] \xrightarrow{i} [n]$$

where i is injective. According to Theorem T.4.1.3.1, it will suffice to show that the simplicial set $\mathrm{N}(\mathcal{J})$ is weakly contractible (for every $m \leq n$).

Let X denote the simplicial subset of $\Delta^m \times \Delta^n$ spanned by those nondegenerate simplices whose projection to Δ^n is also nondegenerate. Then $\mathrm{N}(\mathcal{J})$ can be identified with the barycentric subdivision of X. Consequently, it will suffice to show that the topological space $|X|$ is contractible. For this, we will show that the fibers of the map $\phi : |X| \rightarrow |\Delta^m|$ are contractible.

We will identify the topological m-simplex $|\Delta^m|$ with the set of all sequences of real numbers $0 \leq x_1 \leq \ldots \leq x_m \leq 1$. Similarly, we may identify points of $|\Delta^n|$ with sequences $0 \leq y_1 \leq \ldots \leq y_n \leq 1$. A pair of such sequences determines a point of X if and only if each x_i belongs to the set $\{0, y_1, \ldots, y_n, 1\}$. Consequently, the fiber of ϕ over the point $(0 \leq x_1 \leq \ldots \leq x_m \leq 1)$ can be identified with the set

$$Y = \{0 \leq y_1 \leq \ldots \leq y_n \leq 1 : \{x_1, \ldots, x_m\} \subseteq \{0, y_1, \ldots, y_n, 1\}\} \subseteq |\Delta^n|,$$

which is contractible (Lemma 12.4). □

Corollary 12.6. *Let $\mathcal{C}$ be a stable ∞-category, and let $F : \mathrm{N}(\mathbf{\Delta}_{\leq n})^{op} \rightarrow \mathcal{C}$ be a functor such that $F([m]) \simeq 0$ for all $m < n$. Then there is a canonical isomorphism $\varinjlim(F) \simeq X[n]$ in the homotopy category $\mathrm{h}\mathcal{C}$, where $X = F([n])$.*

Proof. Let $\mathcal{I}$ be as in Lemma 12.5, let G'' denote the composition $\mathrm{N}(\mathcal{I})^{op} \rightarrow \mathrm{N}(\mathbf{\Delta}_{\leq n})^{op} \xrightarrow{F} \mathcal{C}$, and let G denote the constant map $\mathrm{N}(\mathcal{I})^{op} \rightarrow \mathcal{C}$ taking the value X. Let $\mathcal{I}_0$ denote the full subcategory of $\mathcal{I}$ obtained by deleting the initial object. There is a canonical map $\alpha : G \rightarrow G''$, and $G' = \ker(\alpha)$ is a left Kan extension of $G'|\,\mathrm{N}(\mathcal{I}_0)^{op}$. We obtain a distinguished triangle

$$\varinjlim(G') \rightarrow \varinjlim(G) \rightarrow \varinjlim(G'') \rightarrow \varinjlim(G')[1]$$

in the homotopy category $\mathrm{h}\mathcal{C}$. Lemma 12.5 yields an equivalence $\varinjlim(F) \simeq \varinjlim(G'')$, and Lemma T.4.3.2.7 implies the existence of an equivalence $\varinjlim(G') \simeq \varinjlim(G'|\,\mathrm{N}(\mathcal{I}_0)^{op})$.

We now observe that the simplicial set $\mathrm{N}(\mathcal{I})^{op}$ can be identified with the barycentric subdivision of the standard n-simplex Δ^n, and that $\mathrm{N}(\mathcal{I}_0)^{op}$ can be identified with the barycentric subdivision of its boundary $\partial\,\Delta^n$. It follows (see §T.4.4.4) that we may identify the map $\varinjlim(G') \rightarrow \varinjlim(G)$ with the map $\beta : X \otimes (\partial\,\Delta^n) \rightarrow X \otimes \Delta^n$. The cokernel of β is canonically isomorphic (in $\mathrm{h}\mathcal{C}$) to the n-fold suspension $X[n]$ of X. □

Lemma 12.7. *Let $\mathcal{C}$ be a stable ∞-category, let $n \geq 0$, and let $F : \mathrm{N}(\mathbf{\Delta}_{+,\leq n})^{op} \rightarrow \mathcal{C}$ be a functor (here $\mathbf{\Delta}_{+,\leq n}$ denotes the full subcategory of $\mathbf{\Delta}_+$ spanned by the objects $\{[k]\}_{-1 \leq k \leq n}$). The following conditions are equivalent:*

- *(i) The functor F is a left Kan extension of $F|\,\mathrm{N}(\mathbf{\Delta}_{\leq n})^{op}$.*
- *(ii) The functor F is a right Kan extension of $F|\,\mathrm{N}(\mathbf{\Delta}_{+,\leq n-1})^{op}$.*

Proof. Condition (*ii*) is equivalent to the assertion that the composition

$$F' : \mathrm{N}(\mathbf{\Delta}^{op}_{+,\leq n-1})^{\triangleleft}_{[n]/} \rightarrow \mathrm{N}(\mathbf{\Delta}^{op}_{+,\leq n}) \xrightarrow{F} \mathcal{C}$$

is a limit diagram. Since the source of F is isomorphic to $(\Delta^1)^{n+1}$, Proposition 12.3 asserts that F' is a limit diagram if and only if F' is a colimit diagram. In view of Lemma 12.5, F' is a colimit diagram if and only if F is a colimit diagram, which is equivalent to (*i*). □

Theorem 12.8 (∞-Categorical Dold-Kan Correspondence)**.** *Let $\mathcal{C}$ be a stable ∞-category. Then the ∞-categories $\mathrm{Fun}(\mathrm{N}(\mathbf{Z}_{\geq 0}), \mathcal{C})$ and $\mathrm{Fun}(\mathrm{N}(\mathbf{\Delta})^{op}, \mathcal{C})$ are (canonically) equivalent to one another.*

Proof. Our first step is to describe the desired equivalence in more precise terms. Let $\mathcal{I}_+$ denote the full subcategory of $\mathrm{N}(\mathbf{Z}_{\geq 0}) \times \mathrm{N}(\mathbf{\Delta}_+)^{op}$ spanned by those pairs $(n, [m])$, where $m \leq n$, and let $\mathcal{I}$ be the full subcategory of $\mathcal{I}_+$ spanned by those pairs $(n, [m])$ where $0 \leq m \leq n$. We observe that there is a natural projection $p : \mathcal{I} \to \mathrm{N}(\mathbf{\Delta})^{op}$, and a natural embedding $i : \mathrm{N}(\mathbf{Z}_{\geq 0}) \to \mathcal{I}_+$, which carries $n \geq 0$ to the object $(n, [-1])$.

Let $\mathrm{Fun}^0(\mathcal{I}, \mathcal{C})$ denote the full subcategory of $\mathrm{Fun}(\mathcal{I}, \mathcal{C})$ spanned by those functors $F : \mathcal{I} \to \mathcal{C}$ such that, for every $s \leq m \leq n$, the image under F of the natural map $(m, [s]) \to (n, [s])$ is an equivalence in $\mathcal{C}$. Let $\mathrm{Fun}^0(\mathcal{I}_+, \mathcal{C})$ denote the full subcategory of $\mathrm{Fun}(\mathcal{I}_+, \mathcal{C})$ spanned by functors $F_+ : \mathcal{I}_+ \to \mathcal{C}$ such that $F = F_+ | \mathcal{I}$ belongs to $\mathrm{Fun}^0(\mathcal{I}, \mathcal{C})$, and F_+ is a left Kan extension of F. Composition with p, composition with i, and restriction from $\mathcal{I}_+$ to $\mathcal{I}$ yields a diagram of ∞-categories

$$\mathrm{Fun}(\mathrm{N}(\mathbf{\Delta})^{op}, \mathcal{C}) \xrightarrow{G} \mathrm{Fun}^0(\mathcal{I}, \mathcal{C}) \xleftarrow{G'} \mathrm{Fun}^0(\mathcal{I}_+, \mathcal{C}) \xrightarrow{G''} \mathrm{Fun}(\mathrm{N}(\mathbf{Z}_{\geq 0}), \mathcal{C}).$$

We will prove that G, G', and G'' are equivalences of ∞-categories.

To show that G is an equivalence of ∞-categories, we let $\mathcal{I}^{\leq k}$ denote the full subcategory of $\mathcal{I}$ spanned by pairs $(n, [m])$ where $m \leq n \leq k$, and let $\mathcal{I}^k$ denote the full subcategory of $\mathcal{I}$ spanned by those pairs $(n, [m])$ where $m \leq n = k$. Then the projection p restricts to an equivalence $\mathcal{I}^k \to \mathrm{N}(\mathbf{\Delta}_{\leq n})^{op}$. Let $\mathrm{Fun}^0(\mathcal{I}^{\leq k}, \mathcal{C})$ denote the full subcategory of $\mathrm{Fun}(\mathcal{I}^{\leq k}, \mathcal{C})$ spanned by those functors $F : \mathcal{I}^{\leq k} \to \mathcal{C}$ such that, for every $s \leq m \leq n \leq k$, the image under F of the natural map $(m, [s]) \to (n, [s])$ is an equivalence in $\mathcal{C}$. We observe that this is equivalent to the condition that F be a right Kan extension of $F | \mathcal{I}^k$. Using Proposition T.4.3.2.15, we deduce that the restriction map $r : \mathrm{Fun}^0(\mathcal{I}^{\leq k}, \mathcal{C}) \to \mathrm{Fun}(\mathcal{I}^k, \mathcal{C})$ is an equivalence of ∞-categories. Composition with p induces a functor $G_k : \mathrm{Fun}(\mathrm{N}(\mathbf{\Delta}_{\leq k})^{op}, \mathcal{C}) \to \mathrm{Fun}^0(\mathcal{I}^{\leq k}, \mathcal{C})$ which is a section of r. It follows that G_k is an equivalence of ∞-categories. We can identify G with the homotopy inverse limit of the functors $\varprojlim(G_k)$, so that G is also an equivalence of ∞-categories.

The fact that G' is an equivalence of ∞-categories follows immediately from Proposition T.4.3.2.15, since for each $n \geq 0$ the simplicial set $\mathcal{I}_{/(n,[-1])}$ is finite and $\mathcal{C}$ admits finite colimits.

We now show that G'' is an equivalence of ∞-categories. Let $\mathcal{I}_+^{\leq k}$ denote the full subcategory of $\mathcal{I}_+$ spanned by pairs $(n, [m])$ where either $m \leq n \leq k$ or $m = -1$. We let $\mathcal{D}(k)$ denote the full subcategory of $\mathrm{Fun}(\mathcal{I}_+^{\leq k}, \mathcal{C})$ spanned by those functors $F : \mathcal{I}_+^{\leq k} \to \mathcal{C}$ with the following pair of properties:

- (i) For every $s \leq m \leq n \leq k$, the image under F of the natural map $(m, [s]) \to (n, [s])$ is an equivalence in $\mathcal{C}$.

- (ii) For every $n \leq k$, F is a left Kan extension of $F | \mathcal{I}^{\leq k}$ at $(n, [-1])$.

Then $\mathrm{Fun}^0(\mathcal{I}_+, \mathcal{C})$ is the inverse limit of the tower of restriction maps

$$\ldots \to \mathcal{D}(1) \to \mathcal{D}(0) \to \mathcal{D}(-1) = \mathrm{Fun}(\mathrm{N}(\mathbf{Z}_{\geq 0}), \mathcal{C}).$$

To complete the proof, we will show that for each $k \geq 0$, the restriction map $\mathcal{D}(k) \to \mathcal{D}(k-1)$ is a trivial Kan fibration.

Let $\mathcal{I}_0^{\leq k}$ be the full subcategory of $\mathcal{I}_+^{\leq k}$ obtained by removing the object $(k, [k])$, and let $\mathcal{D}'(k)$ be the full subcategory of $\mathrm{Fun}(\mathcal{I}_0^{\leq k}, \mathcal{C})$ spanned by those functors F which satisfy condition (i) and satisfy (ii) for $n < k$. We have restriction maps

$$\mathcal{D}(k) \xrightarrow{\theta} \mathcal{D}'(k) \xrightarrow{\theta'} \mathcal{D}(k-1).$$

We observe that a functor $F : \mathcal{I}_0^{\leq k}$ belongs to $\mathcal{D}'(k)$ if and only if $F | \mathcal{I}_+^{\leq k-1}$ belongs to $\mathcal{D}(k-1)$ and F is a left Kan extension of $F | \mathcal{I}_+^{\leq k-1}$. Using Proposition T.4.3.2.15, we conclude that θ' is a trivial Kan fibration.

We will prove that θ is a trivial Kan fibration by a similar argument. According to Proposition T.4.3.2.15, it will suffice to show that a functor $F : \mathcal{I}_+^{\leq k} \to \mathcal{C}$ belongs to $\mathcal{D}(k)$ if and only if $F | \mathcal{I}_0^{\leq k}$ belongs to $\mathcal{D}'(k)$ and F is a right Kan extension of $F | \mathcal{I}_0^{\leq k}$. This follows immediately from Lemma 12.7 and the observation that the inclusion $\mathcal{I}^k \subseteq \mathcal{I}^{\leq k}$ is cofinal. □

Remark 12.9. Let $\mathcal{C}$ be a stable ∞-category. We may informally describe the equivalence of Theorem 12.8 as follows. To a simplicial object $C_\bullet$ of $\mathcal{C}$, we assign the filtered object

$$D(0) \to D(1) \to D(2) \to \ldots$$

where $D(k)$ is the colimit of the k-skeleton of $C_\bullet$. In particular, we observe that colimits $\varinjlim D(j)$ can be identified with geometric realizations of the simplicial object $C_\bullet$.

Remark 12.10. Let $\mathcal{C}$ be a stable ∞-category, and let X be a simplicial object of $\mathcal{C}$. Using the Dold-Kan correspondence, we can associate to X a chain complex

$$\ldots \to C_2 \to C_1 \to C_0 \to 0$$

in the triangulated category $\mathrm{h}\mathcal{C}$. More precisely, for each $n \geq 0$, let $L_n \in \mathcal{C}$ denote the nth *latching object* of X (see §T.A.2.9), so that X determines a canonical map $\alpha : L_n \to X_n$. Then $C_n \simeq \mathrm{coker}(\alpha)$, where the cokernel can be formed either in the ∞-category $\mathcal{C}$ or in its homotopy category $\mathrm{h}\mathcal{C}$ (since L_n is actually a direct summand of X_n).

Using Theorem 12.8, we can also associate to X a filtered object

$$D(0) \to D(1) \to D(2) \to \ldots$$

of $\mathcal{C}$. Using Lemma 11.3 and Remark 11.2, we can associate to this filtered abject another chain complex

$$\ldots \to C'_1 \to C'_0 \to 0$$

with values in $\mathrm{h}\mathcal{C}$. For each $n \geq 0$, let $X(n)$ denote the restriction of X to $\mathrm{N}(\mathbf{\Delta}^{op}_{\leq n})$, and let $X'(n)$ be a left Kan extension of $X(n-1)$ to $\mathrm{N}(\mathbf{\Delta}^{op}_{\leq n})$. Then we have a canonical map $\beta = X'(n) \to X(n)$, which induces an equivalence $X'(n)_m \to X(n)_m$ for $m < n$, while $X'(n)_n$ can be identified with the latching object L_n. Let $X''(n) = \mathrm{coker}(\beta)$. Then $X''(n)_m = 0$ for $m < n$, while $X''(n)_n \simeq C_n$. Corollary 12.6 determines a canonical isomorphism $\varinjlim X''(n) \simeq C_n[n]$ in the homotopy category $\mathrm{h}\mathcal{C}$ The map $D(n-1) \to D(n)$ can be identified with the composition

$$D(n-1) \simeq \varinjlim X(n-1) \simeq \varinjlim X'(n) \to \varinjlim X(n) \simeq D(n).$$

It follows it follows that $C'_n \simeq \mathrm{coker}(D(n-1) \to D(n))[-n] \simeq X''(n)_n[-n]$ is canonically isomorphic to C_n. It is not difficult to show that these isomorphisms are compatible with the differentials, so that we obtain an isomorphism of chain complexes $C_\bullet \simeq C'_\bullet$ with values in the triangulated category $\mathrm{h}\mathcal{C}$.

Remark 12.11. Let $\mathcal{C}$ be a stable ∞-category, let $X_\bullet$ be a simplicial object of $\mathcal{C}$, let

$$D(0) \to D(1) \to \ldots$$

be the associated filtered object. Using the classical Dold-Kan correspondence and Remark 12.10, we conclude that each X_n is equivalent to a finite coproduct of objects of the form $\mathrm{coker}(D(m-1) \to D(m))[-m]$, where $0 \leq m \leq n$ (here $D(-1) \simeq 0$ by convention).

Remark 12.12. Let $\mathcal{C}$ be a stable ∞-category equipped with a t-structure, whose heart is equivalent to (the nerve of) an abelian category $\mathcal{A}$. Let $X_\bullet$ be a simplicial object of $\mathcal{C}$, and let

$$D(0) \to D(1) \to D(2) \to \ldots$$

be the associated filtered object (Theorem 12.8). Using Definition 11.8 (and Lemma 11.3), we can associate to this filtered object a spectral sequence $\{E^{p,q}_r, d_r\}_{r \geq 1}$ in the abelian category $\mathcal{A}$. In view of Remarks 11.7 and 12.10, for each $q \in \mathbf{Z}$ we can identify the complex $(E^{\bullet,q}_1, d_1)$ with the normalized chain complex associated to the simplicial object $\pi_q X_\bullet$ of $\mathcal{A}$. Under the hypotheses of Proposition 11.13, this spectral sequence converges to a filtration on the homotopy groups $\pi_{p+q} \varinjlim(D(n)) \simeq \pi_{p+q}|X_\bullet|$.

It possible to consider a slight variation on the spectral sequence described above. Namely, one can construct a new spectral sequence $\{\overline{E}^{p,q}_r, d_r\}_{r \geq 1}$ which is isomorphic to $\{E^{p,q}_r, d_r\}_{r \geq 1}$ from the E_2-page onward, but with $\overline{E}^{\bullet,q}_1$ given by the *unnormalized* chain complex of $\pi_q X_\bullet$. We can then write simply $\overline{E}^{p,q}_1 \simeq \pi_q X_p$.

13 Homological Algebra

Let $\mathcal{A}$ be an abelian category. In classical homological algebra, it is customary to associate to $\mathcal{A}$ a certain triangulated category, called the *derived category* of $\mathcal{A}$, the objects of which are chain complexes with values in $\mathcal{A}$. In this section, we will review the theory of derived categories from the perspective of higher category theory. To simplify the discussion, we primarily consider only abelian categories $\mathcal{A}$ which have enough projective objects (the dual case of abelian categories with enough injective objects can be understood by passing to the opposite category).

We begin by considering an arbitrary additive category $\mathcal{A}$. Let $\mathrm{Ch}(\mathcal{A})$ denote the category whose objects are chain complexes

$$\ldots \to A_1 \to A_0 \to A_{-1} \to \ldots$$

with values in $\mathcal{A}$. The category $\mathrm{Ch}(\mathcal{A})$ is naturally enriched over simplicial sets. For $A_\bullet, B_\bullet \in \mathrm{Ch}(\mathcal{A})$, the simplicial set $\mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(A_\bullet, B_\bullet)$ is characterized by the property that for every finite simplicial set K there is a natural bijection

$$\mathrm{Hom}_{\mathrm{Set}_\Delta}(K, \mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(A_\bullet, B_\bullet)) \simeq \mathrm{Hom}_{\mathrm{Ch}(\mathcal{A})}(A_\bullet \otimes C_\bullet(K), B_\bullet).$$

Here $C_\bullet(K)$ denotes the normalized chain complex for computing the homology of K, so that $C_n(K)$ is a free abelian group whose generators are in bijection with the nondegenerate n-simplices of K. Unwinding the definitions, we see that the vertices of $\mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(A_\bullet, B_\bullet)$ are just the maps of chain complexes from $A_\bullet$ to $B_\bullet$. An edge e of $\mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(A_\bullet, B_\bullet)$ is determined by three pieces of data:

(i) A vertex $d_0(e)$, corresponding to a chain map $f : A_\bullet \to B_\bullet$.

(ii) A vertex $d_1(e)$, corresponding to a chain map $g : A_\bullet \to B_\bullet$.

(iii) A map $h : A_\bullet \to B_{\bullet+1}$, which determines a chain homotopy from f to g.

Remark 13.1. Let $\mathcal{A}b$ be the category of abelian groups, and let $\mathrm{Ch}_{\geq 0}(\mathcal{A}b)$ denote the full subcategory of $\mathrm{Ch}(\mathcal{A}b)$ spanned by those complexes $A_\bullet$ such that $A_n \simeq 0$ for all $n < 0$. The classical Dold-Kan correspondence (see [75]) asserts that $\mathrm{Ch}_{\geq 0}(\mathcal{A}b)$ is equivalent to the category of *simplicial* abelian groups. In particular, there is a forgetful functor $\theta : \mathrm{Ch}_{\geq 0}(\mathcal{A}b) \to \mathrm{Set}_\Delta$.

Given a pair of complexes $A_\bullet, B_\bullet \in \mathrm{Ch}(\mathcal{A})$, the mapping space $\mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(A_\bullet, B_\bullet)$ can be defined as follows:

(1) First, we extract the mapping complex

$$[A_\bullet, B_\bullet] \in \mathrm{Ch}(\mathcal{A}b),$$

where $[A_\bullet, B_\bullet]_n = \prod \mathrm{Hom}_{\mathcal{A}}(A_m, B_{n+m})$.

(2) The inclusion $\mathrm{Ch}_{\geq 0}(\mathcal{A}b) \subseteq \mathrm{Ch}(\mathcal{A}b)$ has a right adjoint, which associates to an arbitrary chain complex $M_\bullet$ the truncated complex

$$\ldots \to M_1 \to \ker(M_0 \to M_{-1}) \to 0 \to \ldots$$

Applying this functor to $[A_\bullet, B_\bullet]$, we obtain a new complex $[A_\bullet, B_\bullet]_{\geq 0}$, whose degree zero term coincides with the set of chain maps from $A_\bullet$ to $B_\bullet$.

(3) Applying the Dold-Kan correspondence θ, we can convert the chain complex $[A_\bullet, B_\bullet]_{\geq 0}$ into a simplicial set $\mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(A_\bullet, B_\bullet)$.

Because every simplicial abelian group is a Kan complex, the simplicial category $\mathrm{Ch}(\mathcal{A})$ is automatically *fibrant*.

Remark 13.2. Let $\mathcal{A}$ be an additive category, and let $A_\bullet, B_\bullet \in \mathrm{Ch}(\mathcal{A})$. The homotopy group

$$\pi_n \operatorname{Map}_{\mathrm{Ch}(\mathcal{A})}(A_\bullet, B_\bullet)$$

can be identified with the group of chain-homotopy classes of maps from $A_\bullet$ to $B_{\bullet+n}$.

Example 13.3. Let $\mathcal{A}$ be an abelian category, and let $A_\bullet, B_\bullet \in \mathrm{Ch}(\mathcal{A})$. Suppose that $A_n \simeq 0$ for $n < 0$, and that $B_n \simeq 0$ for $n > 0$. Then the simplicial set $\operatorname{Map}_{\mathrm{Ch}(\mathcal{A})}(A_\bullet, B_\bullet)$ is constant, with value $\operatorname{Hom}_{\mathcal{A}}(\mathrm{H}_0(A_\bullet), \mathrm{H}_0(B_\bullet))$.

Lemma 13.4. *Let $\mathcal{A}$ be an additive category. Then:*

(1) *Let*

$$\begin{tikzcd} A_\bullet \arrow[r, "f"] \arrow[d] & B_\bullet \arrow[d] \\ A'_\bullet \arrow[r] & B'_\bullet \end{tikzcd}$$

be a pushout diagram in the (ordinary) category $\mathrm{Ch}(\mathcal{A})$*, and suppose that f is* degreewise split *(so that each $B_n \simeq A_n \oplus C_n$, for some $C_n \in \mathcal{A}$). Then the above diagram determines a homotopy pushout square in the ∞-category* $\mathrm{N}(\mathrm{Ch}(\mathcal{A}))$*.*

(2) *The ∞-category* $\mathrm{N}(\mathrm{Ch}(\mathcal{A}))$ *is stable.*

Proof. To prove (1), it will suffice (Theorem T.4.2.4.1) to show that for every $D_\bullet \in \mathrm{Ch}(\mathcal{A})$, the associated diagram of simplicial sets

$$\begin{tikzcd} \operatorname{Map}_{\mathrm{Ch}(\mathcal{A})}(B'_\bullet, D_\bullet) \arrow[r] \arrow[d] & \operatorname{Map}_{\mathrm{Ch}(\mathcal{A})}(A'_\bullet, D_\bullet) \arrow[d] \\ \operatorname{Map}_{\mathrm{Ch}(\mathcal{A})}(B_\bullet, D_\bullet) \arrow[r, "f'"] & \operatorname{Map}_{\mathrm{Ch}(\mathcal{A})}(A_\bullet, D_\bullet) \end{tikzcd}$$

is homotopy Cartesian. The above diagram is obviously a pullback, it will suffice to prove that f' is a Kan fibration. This follows from the fact that f' is the map of simplicial sets associated (under the Dold-Kan correspondence) to a map between complexes of abelian groups which is surjective in positive (homological) degrees.

It follows from (1) that the ∞-category $\mathrm{N}(\mathrm{Ch}(\mathcal{A}))$ admits pushouts: it suffices to observe that any morphism $f : A_\bullet \to B_\bullet$ is chain homotopy-equivalent to a morphism which is degreewise split (replace $B_\bullet$ by the mapping cylinder of f). It is obvious that $\mathrm{N}(\mathrm{Ch}(\mathcal{A}))$ has a zero object (since $\mathrm{Ch}(\mathcal{A})$ has a zero object). Moreover, we can use (1) to describe the suspension functor on $\mathrm{Ch}(\mathcal{A})$: for each $A_\bullet \in \mathrm{Ch}(\mathcal{A})$, let $C(A_\bullet)$ denote the cone of $A_\bullet$, so that $C(A_\bullet) \simeq 0$ and there is a pushout diagram

$$\begin{tikzcd} A_\bullet \arrow[r] \arrow[d] & C(A_\bullet) \arrow[d] \\ 0 \arrow[r] & A_{\bullet-1}. \end{tikzcd}$$

It follows that the suspension functor Σ can be identified with the shift functor

$$A_\bullet \mapsto A_{\bullet-1}.$$

In particular, we conclude that Σ is an equivalence of ∞-categories, so that $\mathrm{Ch}(\mathcal{A})$ is stable (Proposition 8.28). □

Remark 13.5. Let $\mathcal{A}$ be an additive category, and let $\mathrm{Ch}'(\mathcal{A})$ be a full subcategory of $\mathrm{Ch}(\mathcal{A})$. Suppose that $\mathrm{Ch}'(\mathcal{A})$ is stable under translations and the formation of mapping cones. Then the proof of Lemma 13.4 shows that $\mathrm{N}(\mathrm{Ch}'(\mathcal{A}))$ is a stable subcategory of $\mathrm{N}(\mathrm{Ch}(\mathcal{A}))$. In particular, if $\mathrm{Ch}^{-}(\mathcal{A})$ denotes the full subcategory of $\mathrm{Ch}(\mathcal{A})$ spanned by those complexes $A_\bullet$ such that $A_n \simeq 0$ for $n \ll 0$, then $\mathrm{N}(\mathrm{Ch}^{-}(\mathcal{A}))$ is a stable subcategory of $\mathrm{N}(\mathrm{Ch}(\mathcal{A}))$.

Definition 13.6. Let $\mathcal{A}$ be an abelian category with enough projective objects. We let $\mathcal{D}^{-}(\mathcal{A})$ denote the nerve of the simplicial category $\mathrm{Ch}^{-}(\mathcal{A}_0)$, where $\mathcal{A}_0 \subseteq \mathcal{A}$ is the full subcategory spanned by the projective objects of $\mathcal{A}$. We will refer to $\mathcal{D}^{-}(\mathcal{A})$ as the *derived ∞-category of $\mathcal{A}$*.

Remark 13.7. The homotopy category $\mathrm{h}\mathcal{D}^{-}(\mathcal{A})$ can be described as follows: objects are given by (bounded above) chain complexes of projective objects of $\mathcal{A}$, and morphisms are given by homotopy classes of chain maps. Consequently, $\mathrm{h}\mathcal{D}^{-}(\mathcal{A})$ can be identified with the derived category of $\mathcal{A}$ studied in classical homological algebra (with appropriate boundedness conditions imposed).

Lemma 13.8. *Let $\mathcal{A}$ be an abelian category, and let $P_\bullet \in \mathrm{Ch}(\mathcal{A})$ be a complex of projective objects of $\mathcal{A}$ such that $P_n \simeq 0$ for $n \ll 0$. Let $Q_\bullet \to Q'_\bullet$ be a quasi-isomorphism in $\mathrm{Ch}(\mathcal{A})$. Then the induced map*

$$\mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(P_\bullet, Q_\bullet) \to \mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(P_\bullet, Q'_\bullet)$$

is a homotopy equivalence.

Proof. We observe that $P_\bullet$ is a homotopy colimit of its naive truncations

$$\ldots \to 0 \to P_n \to P_{n-1} \to \ldots.$$

It therefore suffices to prove the result for each of these truncations, so we may assume that $P_\bullet$ is concentrated in finitely many degrees. Working by induction, we can reduce to the case where $P_\bullet$ is concentrated in a single degree. Shifting, we can reduce to the case where $P_\bullet$ consists of a single projective object P concentrated in degree zero. Since P is projective, we have isomorphisms

$$\mathrm{Ext}^i_{\mathrm{N}(\mathrm{Ch}(\mathcal{A}))}(P_\bullet, Q_\bullet) \simeq \mathrm{Hom}_{\mathcal{A}}(P, \mathrm{H}_{-i}(Q_\bullet)) \simeq \mathrm{Hom}_{\mathcal{A}}(P, \mathrm{H}_{-i}(Q'_\bullet)) \simeq \mathrm{Ext}^i_{\mathrm{N}(\mathrm{Ch}(\mathcal{A}))}(P_\bullet, Q'_\bullet).$$

□

Lemma 13.9. *Let $\mathcal{A}$ be an abelian category. Suppose that $P_\bullet, Q_\bullet \in \mathrm{Ch}(\mathcal{A})$ have the following properties:*

1. *Each P_n is projective, and $P_n \simeq 0$ for $n < 0$.*
2. *The homologies $\mathrm{H}_n(Q_\bullet)$ vanish for $n > 0$.*

Then the space $\mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(P_\bullet, Q_\bullet)$ is discrete, and we have a canonical isomorphism of abelian groups

$$\mathrm{Ext}^0(P_\bullet, Q_\bullet) \simeq \mathrm{Hom}_{\mathcal{A}}(\mathrm{H}_0(P_\bullet), \mathrm{H}_0(Q_\bullet)).$$

Proof. Let $Q'_\bullet$ be the complex

$$\ldots \to 0 \to \mathrm{coker}(Q_1 \to Q_0) \to Q_{-1} \to \ldots.$$

Condition (2) implies that the canonical map $Q_\bullet \to Q'_\bullet$ is a quasi-isomorphism. In view of (1) and Lemma 13.8, it will suffice to prove the result after replacing $Q_\bullet$ by $Q'_\bullet$. The result now follows from Example 13.3. □

Proposition 13.10. *Let $\mathcal{A}$ be an abelian category with enough projective objects. Then:*

1. *The ∞-category $\mathcal{D}^{-}(\mathcal{A})$ is stable.*

(2) *Let $\mathcal{D}^{-}_{\geq 0}(\mathcal{A})$ be the full subcategory of $\mathcal{D}^{-}(\mathcal{A})$ spanned by those complexes $A_\bullet$ such that the homology objects $\mathrm{H}_n(A_\bullet) \in \mathcal{A}$ vanish for $n < 0$, and let $\mathcal{D}^{-}_{\leq 0}(\mathcal{A})$ be defined similarly. Then $(\mathcal{D}^{-}_{\geq 0}(\mathcal{A}), \mathcal{D}^{-}_{\leq 0}(\mathcal{A}))$ determines a t-structure on $\mathcal{D}^{-}(\mathcal{A})$.*

(3) *The heart of $\mathcal{D}^{-}(\mathcal{A})$ is equivalent to (the nerve of) the abelian category $\mathcal{A}$.*

Proof. Assertion (1) follows from Remark 13.5.

To prove (2), we first make the following observation:

($*$) For any object $A_\bullet \in \mathrm{Ch}(\mathcal{A})$, there exists a map $f : P_\bullet \to A_\bullet$ where each P_n is projective, $P_n \simeq 0$ for $n < 0$, and the induced map $\mathrm{H}_k(P_\bullet) \to \mathrm{H}_k(A_\bullet)$ is an isomorphism for $k \geq 0$.

This is proven by a standard argument in homological algebra, using the assumption that $\mathcal{A}$ has enough projectives. We also note that if $A_\bullet \in \mathcal{D}^{-}(\mathcal{A})$ and the homologies $\mathrm{H}_n(A_\bullet)$ vanish for $n < 0$, then f is a quasi-isomorphism between projective complexes and therefore a chain homotopy equivalence.

It is obvious that $\mathcal{D}^{-}_{\leq 0}(\mathcal{A})[-1] \subseteq \mathcal{D}^{-}_{\leq 0}(\mathcal{A})$ and $\mathcal{D}^{-}_{\geq 0}(\mathcal{A})[1] \subseteq \mathcal{D}^{-}_{\geq 0}(\mathcal{A})$. Suppose now that $A_\bullet \in \mathcal{D}^{-}_{\geq 0}(\mathcal{A})$ and $B_\bullet \in \mathcal{D}^{-}_{\leq -1}(\mathcal{A})$; we wish to show that $\mathrm{Ext}^0_{\mathcal{D}^{-}(\mathcal{A})}(A_\bullet, B_\bullet) \simeq 0$. Using ($*$), we may reduce to the case where $A_n \simeq 0$ for $n < 0$. The desired result now follows immediately from Lemma 13.9. Finally, choose an arbitrary object $A_\bullet \in \mathcal{D}^{-}(\mathcal{A})$, and let $f : P_\bullet \to A_\bullet$ be as in ($*$). It is easy to see that $\mathrm{coker}(f) \in \mathcal{D}^{-}_{\leq -1}(\mathcal{A})$. This completes the proof of (2).

To prove (3), we begin by observing that the functor $A_\bullet \mapsto \mathrm{H}_0(A_\bullet)$ determines a functor $\theta : \mathrm{N}(\mathrm{Ch}(\mathcal{A})) \to \mathrm{N}(\mathcal{A})$. Let $\mathcal{C} \subseteq \mathrm{N}(\mathrm{Ch}(\mathcal{A}))$ be the full subcategory spanned by complexes $P_\bullet$ such that each P_n is projective, $P_n \simeq 0$ for $n < 0$, and $\mathrm{H}_n(P_\bullet) \simeq 0$ for $n \neq 0$. Assertion ($*$) implies that the inclusion $\mathcal{C} \subseteq \mathcal{D}^{-}(\mathcal{A})^{\heartsuit}$ is an equivalence of ∞-categories. Lemma 13.9 implies that $\theta | \mathcal{C}$ is fully faithful. Finally, we can apply ($*$) in the case where $A_\bullet$ is concentrated in degree zero to deduce that $\theta | \mathcal{C}$ is essentially surjective. This proves (3). □

Remark 13.11. Let $\mathcal{A}$ be an abelian category with enough projective objects. Then $\mathcal{D}^{-}(\mathcal{A})$ is a colocalization of $\mathrm{N}(\mathrm{Ch}^{-}(\mathcal{A}))$. To prove this, it will suffice to show that for every $A_\bullet \in \mathrm{Ch}^{-}(\mathcal{A})$, there exists a map of chain complexes $f : P_\bullet \to A_\bullet$ where $P_\bullet \in \mathcal{D}^{-}(\mathcal{A})$, and such that f induces a homotopy equivalence

$$\mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(Q_\bullet, P_\bullet) \to \mathrm{Map}_{\mathrm{Ch}(\mathcal{A})}(Q_\bullet, A_\bullet)$$

for every $Q_\bullet \in \mathcal{D}^{-}(\mathcal{A})$ (Proposition T.5.2.7.8). According Lemma 13.8, it will suffice to choose f to be a quasi-isomorphism; the existence now follows from ($*$) in the proof of Proposition 13.10.

Let $L : \mathrm{N}(\mathrm{Ch}^{-}(\mathcal{A})) \to \mathcal{D}^{-}(\mathcal{A})$ be a right adjoint to the inclusion. Roughly speaking, the functor L associates to each complex $A_\bullet$ a projective resolution $P_\bullet$ as above. We observe that, if $f : A_\bullet \to B_\bullet$ is a map of complexes, then Lf is a chain homotopy equivalence if and only if f is a quasi-isomorphism. Consequently, we may regard $\mathcal{D}^{-}(\mathcal{A})$ as the ∞-category obtained from $\mathrm{N}(\mathrm{Ch}^{-}(\mathcal{A}))$ by inverting all quasi-isomorphism.

14 The Universal Property of $\mathcal{D}^{-}(\mathcal{A})$

In this section, we will apply the results of §T.5.5.8 and §T.5.5.9 to characterize the derived ∞-category $\mathcal{D}^{-}(\mathcal{A})$ by a universal mapping property. Here $\mathcal{A}$ denotes an abelian category with enough projective objects; to simplify the discussion, we will assume that $\mathcal{A}$ is small.

Let $\mathcal{A}_0 \subseteq \mathcal{A}$ be the full subcategory of $\mathcal{A}$ spanned by the projective objects, and let $\mathbf{A}$ denote the category of product-preserving functors from $\mathcal{A}_0^{op}$ to the category of simplicial sets, as in §T.5.5.9. Let $\mathcal{A}^{\vee}$ denote the category of product-preserving functors from $\mathcal{A}_0^{op}$ to sets, so that we can identify $\mathbf{A}$ with the category of simplicial objects of $\mathcal{A}^{\vee}$. Our first goal is to understand the category $\mathcal{A}^{\vee}$.

Lemma 14.1. *Let $\mathcal{A}$ be an abelian category with enough projective objects, and let $\mathcal{B}$ be an arbitrary category which admits finite colimits. Let $\mathcal{C}$ be the category of right exact functors from $\mathcal{A}$ to $\mathcal{B}$, and let $\mathcal{C}'$ be the category of coproduct-preserving functors from $\mathcal{A}_0$ to $\mathcal{B}$. Then the restriction functor $\theta : \mathcal{C} \to \mathcal{C}'$ is an equivalence of categories.*

Proof. We will describe an explicit construction of an inverse functor. Let $f : \mathcal{A}_0 \to \mathcal{B}$ be a functor which preserves finite coproducts. Let $A \in \mathcal{A}$ be an arbitrary object. Since $\mathcal{A}$ has enough projectives, there exists a projective resolution

$$\ldots \to P_1 \overset{u}{\to} P_0 \to A \to 0.$$

We now define $F(A)$ to be the coequalizer of the map

$$\begin{tikzcd} f(P_1) \arrow[r, shift left, "f(0)"] \arrow[r, shift right, "f(u)"'] & f(P_0) \end{tikzcd} .$$

Of course, this definition appears to depend not only on A but on a choice of projective resolution. However, because any two projective resolutions of A are chain homotopy equivalent to one another, $F(A)$ is well-defined up to canonical isomorphism. It is easy to see that $F : \mathcal{A} \to \mathcal{B}$ is a right exact functor which extends f, and that F is uniquely determined (up to unique isomorphism) by these properties. □

Proposition 14.2. *Let $\mathcal{A}$ be an abelian category with enough projective objects. Then:*

1. *The category $\mathcal{A}^\vee$ can be identified with the category of* Ind-*objects of $\mathcal{A}$.*
2. *The category $\mathcal{A}^\vee$ is abelian.*
3. *The abelian category $\mathcal{A}^\vee$ has enough projective objects.*

Proof. Assertion (1) follows immediately from Lemma 14.1 (taking $\mathcal{B}$ to be the opposite of the category of sets). Part (2) follows formally from (1) and the assumption that $\mathcal{A}$ is an abelian category (see, for example, [3]). We may identify $\mathcal{A}$ with a full subcategory of $\mathcal{A}^\vee$ via the Yoneda embedding. Moreover, if P is a projective object of $\mathcal{A}$, then P is also projective when viewed as an object of $\mathcal{A}^\vee$. An arbitrary object of $\mathcal{A}^\vee$ can be written as a filtered colimit $A = \varinjlim\{A_\alpha\}$, where each $A_\alpha \in \mathcal{A}$. Using the assumption that $\mathcal{A}$ has enough projective objects, we can choose epimorphisms $P_\alpha \to A_\alpha$, where each P_α is projective. We then have an epimorphism $\oplus P_\alpha \to A$. Since $\oplus P_\alpha$ is projective, we conclude that $\mathcal{A}^\vee$ has enough projectives. □

Warning 14.3. Let $\mathcal{A}$ be an abelian category with enough projective objects, and let $\mathbf{A}$ be the category of product-preserving functors $\mathcal{A}_0^{op} \to \mathcal{S}\mathrm{et}_\Delta$. The Dold-Kan correspondence determines an equivalence of categories $\theta : \mathbf{A} \simeq \mathrm{Ch}_{\geq 0}(\mathcal{A}^\vee)$. However, this is *not* an equivalence of simplicial categories. Let K be a simplicial set, and let $\mathbf{Z}K$ denote the free simplicial abelian group generated by K (so that the group of n-simplices of $\mathbf{Z}K$ is the free abelian group generated by the set of n-simplices of K, for each $n \geq 0$). Then $\mathbf{A}$ is tensored over the category of simplicial sets in two different ways:

(*i*) Given a simplicial set K and an object $A_\bullet \in \mathbf{A}$ viewed as a simplicial object of $\mathcal{A}^\vee$, we can form the tensor product $A_\bullet \otimes K$ given by the formula $(A_\bullet \otimes K)_n = A_n \otimes (\mathbf{Z}K)_n$.

(*ii*) Given a simplicial set K and an object $A_\bullet \in \mathbf{A}$, we can construct a new object $A_\bullet \odot K$, which is characterized by the existence of an isomorphism

$$\theta(A_\bullet \odot K) \simeq \theta(A_\bullet) \otimes \theta'(\mathbf{Z}K)$$

in the category $\mathrm{Ch}(\mathcal{A}^\vee)$. Here $\theta'(\mathbf{Z}K)$ denotes the object of $\mathrm{Ch}(\mathcal{A}b)$ determined by $\mathbf{Z}K$.

However, it is easy to see that both of these simplicial structures on $\mathbf{A}$ are compatible with the model structure of Proposition T.5.5.9.1. Moreover, the classical *Alexander-Whitney map* determines a natural transformation $A_\bullet \otimes K \to A_\bullet \odot K$, which endows $\theta^{-1} : \mathrm{Ch}_{\geq 0}(\mathcal{A}^\vee) \to \mathbf{A}$ with the structure of a simplicial functor.

We observe that every object of $\mathbf{A}$ is fibrant, and that an object of $\mathbf{A}$ is cofibrant if and only if it corresponds (under θ) to a complex of projective objects of $\mathcal{A}^\vee$. Applying Corollary T.A.3.1.12, we obtain an equivalence of ∞-categories $\mathcal{D}^-_{\geq 0}(\mathcal{A}^\vee) \to \mathrm{N}(\mathbf{A}^\circ)$. Here $\mathcal{D}^-_{\geq 0}(\mathcal{A}^\vee)$ denotes the full subcategory of $\mathcal{D}^-(\mathcal{A}^\vee)$ spanned by those complexes $P_\bullet$ such that $P_n \simeq 0$ for $n < 0$. Composing with the equivalence of Corollary T.5.5.9.3, we obtain the following result:

Proposition 14.4. *Let $\mathcal{A}$ be an abelian category with enough projective objects. Then there exists an equivalence of ∞-categories*

$$\psi : \mathcal{D}^-_{\geq 0}(\mathcal{A}^\vee) \to \mathcal{P}_\Sigma(\mathrm{N}(\mathcal{A}_0))$$

whose composition with the inclusion $\mathrm{N}(\mathcal{A}_0) \subseteq \mathcal{D}^-_{\geq}(\mathcal{A}^\vee)$ *is equivalent to the Yoneda embedding* $\mathrm{N}(\mathcal{A}_0) \to \mathcal{P}_\Sigma(\mathrm{N}(\mathcal{A}_0))$.

Remark 14.5. We can identify $\mathcal{D}^-(\mathcal{A})$ with a full subcategory of $\mathcal{D}^-(\mathcal{A}^\vee)$. Moreover, an object $P_\bullet \in \mathcal{D}^-(\mathcal{A}^\vee)$ belongs to the essential image of $\mathcal{D}^-(\mathcal{A})$ if and only if the homologies $\mathrm{H}_n(P_\bullet)$ belong to $\mathcal{A}$, for all $n \in \mathbf{Z}$.

Proposition 14.6. *Let $\mathcal{A}$ be an abelian category with enough projective objects. Then the t-structure on $\mathcal{D}^-(\mathcal{A})$ is right bounded and left complete.*

Proof. The right boundedness of $\mathcal{D}^-(\mathcal{A})$ is obvious. To prove the left completeness, we must show that $\mathcal{D}^-(\mathcal{A})$ is a homotopy inverse limit of the tower of ∞-categories

$$\ldots \to \mathcal{D}^-(\mathcal{A})_{\leq 1} \to \mathcal{D}^-(\mathcal{A})_{\leq 0} \to \ldots$$

Invoking the right boundedness of $\mathcal{D}^-(\mathcal{A})$, we may reduce to proving that for each $n \in \mathbf{Z}$, $\mathcal{D}^-(\mathcal{A})_{\geq n}$ is a homotopy inverse limit of the tower

$$\ldots \to \mathcal{D}^-(\mathcal{A})_{\leq 1, \geq n} \to \mathcal{D}^-(\mathcal{A})_{\leq 0, \geq n} \to \ldots$$

Shifting if necessary, we may suppose that $n = 0$. Using Remark 14.5, we can replace $\mathcal{A}$ by $\mathcal{A}^\vee$. For each $k \geq 0$, we let $\mathcal{P}^{\leq k}_\Sigma(\mathrm{N}(\mathcal{A}_0))$ denote the ∞-category of product-preserving functors from $\mathrm{N}(\mathcal{A}_0)^{op}$ to $\tau_{\leq k}\,\mathcal{S}$; equivalently, we can define $\mathcal{P}^{\leq k}_\Sigma(\mathrm{N}(\mathcal{A}_0))$ to be the ∞-category of k-truncated objects of $\mathcal{P}_\Sigma(\mathrm{N}(\mathcal{A}_0))$. We observe that the equivalence ψ of Proposition 14.4 restricts to an equivalence

$$\psi(k) : \mathcal{D}^-_{\geq 0}(\mathcal{A}^\vee)_{\leq k} \to \mathcal{P}^{\leq k}_\Sigma(\mathrm{N}(\mathcal{A}_0)).$$

Consequently, it will suffice to show that $\mathcal{P}_\Sigma(\mathrm{N}(\mathcal{A}_0))$ is a homotopy inverse limit for the tower

$$\ldots \to \mathcal{P}^{\leq 1}_\Sigma(\mathrm{N}(\mathcal{A}_0)) \to \mathcal{P}^{\leq 0}_\Sigma(\mathrm{N}(\mathcal{A}_0)).$$

Since the truncation functors on $\mathcal{S}$ commute with finite products (Lemma T.6.5.1.2), we may reduce to the problem of showing that $\mathcal{S}$ is a homotopy inverse limit of the tower

$$\ldots \to \tau_{\leq 1}\,\mathcal{S} \to \tau_{\leq 0}\,\mathcal{S}\,.$$

This amounts to the classical fact that every space X can be recovered as the limit of its Postnikov tower (see for example §T.7.2.1).

□

Our goal is to characterize the derived ∞-category $\mathcal{D}^-(\mathcal{A})$ by a universal mapping property. Propositions 14.4 and T.5.5.8.15 give a characterization of $\mathcal{D}^-_{\geq 0}(\mathcal{A}^\vee)$ of the right flavor. The next step is to understand the embedding of $\mathcal{D}^-_{\geq 0}(\mathcal{A})$ into $\mathcal{D}^-_{\geq 0}(\mathcal{A}^\vee)$.

Definition 14.7. Let $\mathcal{C}$ and $\mathcal{C}'$ be stable ∞-categories equipped with t-structures. We will say that a functor $f : \mathcal{C} \to \mathcal{C}'$ is *right t-exact* if it is exact, and carries $\mathcal{C}_{\geq 0}$ into $\mathcal{C}'_{\geq 0}$.

Lemma 14.8. (1) *Let $\mathcal{C}$ be an ∞-category which admits finite coproducts and geometric realizations. Then $\mathcal{C}$ admits all finite colimits. Conversely, if $\mathcal{C}$ is an n-category which admits finite colimits, then $\mathcal{C}$ admits geometric realizations.*

(2) *Let $F : \mathcal{C} \rightarrow \mathcal{D}$ be a functor between ∞-categories which admit finite coproducts and geometric realizations. If F preserves finite coproducts and geometric realizations, then F is right exact. The converse holds if $\mathcal{C}$ and $\mathcal{D}$ are n-categories.*

Proof. We will prove (1); the proof of (2) follows by the same argument. Now suppose that $\mathcal{C}$ admits finite coproducts and geometric realizations of simplicial objects. We wish to show that $\mathcal{C}$ admits all finite colimits. According to Proposition T.4.4.3.2, it will suffice to show that $\mathcal{C}$ admits coequalizers. Let $\mathbf{\Delta}_s^{\leq 1}$ be the full subcategory of $\mathbf{\Delta}$ spanned by the objects $[0]$ and $[1]$, and *injective* maps between them, so that a coequalizer diagram in $\mathcal{C}$ can be identified with a functor $\mathrm{N}(\mathbf{\Delta}_s^{\leq 1})^{op} \rightarrow \mathcal{C}$. Let $j : \mathrm{N}(\mathbf{\Delta}_s^{\leq 1})^{op} \rightarrow \mathrm{N}(\mathbf{\Delta})^{op}$ be the inclusion functor. We claim that every diagram $f : \mathrm{N}(\mathbf{\Delta}_s^{\leq 1})^{op} \rightarrow \mathcal{C}$ has a left Kan extension along j. To prove this, it suffices to show that for each $n \geq 0$, the associated diagram

$$\mathrm{N}(\mathbf{\Delta}_s^{\leq 1})^{op} \times_{\mathrm{N}(\mathbf{\Delta})^{op}} (\mathrm{N}(\mathbf{\Delta})^{op})_{[n]/} \rightarrow \mathcal{C}$$

has a colimit. We now observe that this last colimit is equivalent to a coproduct: more precisely, we have $(j_! f)([n]) \simeq f([0]) \coprod f([1]) \coprod \ldots \coprod f([1])$, where there are precisely n summands equivalent to $f([1])$. Since $\mathcal{C}$ admits finite coproducts, the desired Kan extension $j_! f$ exists. We now observe that $\varinjlim(f)$ can be identified with $\varinjlim(j_! f)$, and the latter exists in virtue of our assumption that $\mathcal{C}$ admits geometric realizations for simplicial objects.

Now suppose that $\mathcal{C}$ is an n-category which admits finite colimits; we wish to show that $\mathcal{C}$ admits geometric realizations. Passing to a larger universe if necessary, we may suppose that $\mathcal{C}$ is small. Let $\mathcal{D} = \mathrm{Ind}(\mathcal{C})$, and let $\mathcal{C}' \subseteq \mathcal{D}$ denote the essential image of the Yoneda embedding $j : \mathcal{C} \rightarrow \mathcal{D}$. Then $\mathcal{D}$ admits small colimits (Theorem T.5.5.1.1) and j is fully faithful (Proposition T.5.1.3.1); it will therefore suffice to show that $\mathcal{C}'$ is stable under geometric realization of simplicial objects in $\mathcal{D}$.

Fix a simplicial object $U_\bullet : \mathrm{N}(\mathbf{\Delta})^{op} \rightarrow \mathcal{C}' \subseteq \mathcal{D}$. Let $V_\bullet : \mathrm{N}(\mathbf{\Delta})^{op} \rightarrow \mathcal{D}$ be a left Kan extension of $U_\bullet | \mathrm{N}(\mathbf{\Delta}^{\leq n})^{op}$, and $\alpha_\bullet : V_\bullet \rightarrow U_\bullet$ the induced map. The geometric realization of $V_\bullet$ can be identified with the colimit of $U_\bullet | \mathrm{N}(\mathbf{\Delta}^{\leq n})^{op}$, and therefore belongs to $\mathcal{C}'$ since $\mathcal{C}'$ is stable under finite colimits in $\mathcal{D}$ (Proposition T.5.3.5.14). Consequently, it will suffice to prove that $\alpha_\bullet$ induces an equivalence from the geometric realization of $V_\bullet$ to the geometric realization of $U_\bullet$.

Let $L : \mathcal{P}(\mathcal{C}) \rightarrow \mathcal{D}$ be a left adjoint to the inclusion. Let $|U_\bullet|$ and $|V_\bullet|$ be colimits of $U_\bullet$ and $V_\bullet$ in the ∞-category $\mathcal{P}(\mathcal{C})$, and let $|\alpha_\bullet| : |V_\bullet| \rightarrow |U_\bullet|$ be the induced map. We wish to show that $L|\alpha_\bullet|$ is an equivalence in $\mathcal{D}$. Since $\mathcal{C}$ is an n-category, we have inclusions $\mathrm{Ind}(\mathcal{C}) \subseteq \mathrm{Fun}(\mathcal{C}^{op}, \tau_{\leq n-1}\, \mathcal{S}) \subseteq \mathcal{P}(\mathcal{C})$. It follows that L factors through the truncation functor $\tau_{\leq n-1} : \mathcal{P}(\mathcal{C}) \rightarrow \mathcal{P}(\mathcal{C})$. Consequently, it will suffice to prove that $\tau_{\leq n-1}|\alpha_\bullet|$ is an equivalence in $\mathcal{P}(\mathcal{C})$. For this, it will suffice to show that the morphism $|\alpha_\bullet|$ is n-connective (in the sense of Definition T.6.5.1.10). This follows from Lemma T.6.5.3.10, since $\alpha_k : V_k \rightarrow U_k$ is an equivalence for $k \leq n$. □

Lemma 14.9. *Let $\mathcal{C}$ and $\mathcal{C}'$ be stable ∞-categories equipped with t-structures. Let $\theta : \mathrm{Fun}(\mathcal{C}, \mathcal{C}') \rightarrow \mathrm{Fun}(\mathcal{C}_{\geq 0}, \mathcal{C}')$ be the restriction map. Then:*

(1) *If $\mathcal{C}$ is right-bounded, then θ induces an equivalence from the full subcategory of* $\mathrm{Fun}(\mathcal{C}, \mathcal{C}')$ *spanned by the right t-exact functors to the full subcategory of* $\mathrm{Fun}(\mathcal{C}_{\geq 0}, \mathcal{C}'_{\geq 0})$ *spanned by the right exact functors.*

(2) *Let $\mathcal{C}$ and $\mathcal{C}'$ be left complete. Then the ∞-categories $\mathcal{C}_{\geq 0}$ and $\mathcal{C}'_{\geq 0}$ admit geometric realizations of simplicial objects. Furthermore, a functor $F : \mathcal{C}_{\geq 0} \rightarrow \mathcal{C}'_{\geq 0}$ is right exact if and only if if it preserves finite coproducts and geometric realizations of simplicial objects.*

Proof. We first prove (1). If $\mathcal{C}$ is right bounded, then $\mathrm{Fun}(\mathcal{C}, \mathcal{C}')$ is the (homotopy) inverse limit of the tower

$$\ldots \rightarrow \mathrm{Fun}(\mathcal{C}_{\geq -1}, \mathcal{C}') \rightarrow \mathrm{Fun}(\mathcal{C}_{\geq 0}, \mathcal{C}'),$$

where the functors are given by restriction. The full subcategory of right t-exact functors is then given by the homotopy inverse limit

$$\ldots \rightarrow \mathrm{Fun}'(\mathcal{C}_{\geq -1}, \mathcal{C}'_{\geq -1}) \overset{\theta(0)}{\rightarrow} \mathrm{Fun}(\mathcal{C}_{\geq 0}, \mathcal{C}'_{\geq 0})$$

where $\mathrm{Fun}'(\mathcal{C}, \mathcal{D})$ denotes the full subcategory of $\mathrm{Fun}(\mathcal{C}, \mathcal{D})$ spanned by the right exact functors. To complete the proof, it will suffice to show that this tower is essentially constant; in other words, that each $\theta(n)$ is an equivalence of ∞-categories. Without loss of generality, we may suppose that $n = 0$. Choose shift functors on the ∞-categories $\mathcal{C}$ and $\mathcal{C}'$, and define

$$\psi : \mathrm{Fun}'(\mathcal{C}_{\geq 0}, \mathcal{C}'_{\geq 0}) \to \mathrm{Fun}'(\mathcal{C}_{\geq -1}, \mathcal{C}'_{\geq -1})$$

by the formula $\psi(F) = \Sigma^{-1} \circ F \circ \Sigma$. We claim that ψ is a homotopy inverse to $\theta(0)$. To prove this, we observe that the right exactness of $F \in \mathrm{Fun}'(\mathcal{C}_{\geq 0}, \mathcal{C}'_{\geq 0})$, $G \in \mathrm{Fun}'(\mathcal{C}_{\geq -1}, \mathcal{C}'_{\geq -1})$ determines canonical equivalences

$$(\theta(0) \circ \psi)(F) \simeq F$$

$$(\psi \circ \theta(0))(G) \simeq G.$$

We now prove (2). Since $\mathcal{C}$ is left complete, $\mathcal{C}_{\geq 0}$ is the (homotopy) inverse limit of the tower of ∞-categories $\{(\mathcal{C}_{\geq 0})_{\leq n}\}$ with transition maps given by right exact truncation functors. Lemma 14.8 implies that each $(\mathcal{C}_{\geq 0})_{\leq n}$ admits geometric realizations of simplicial objects, and that each of the truncation functors preserves geometric realizations of simplicial objects. It follows that $\mathcal{C}_{\geq 0}$ admits geometric realizations for simplicial objects. Similarly, $\mathcal{C}'_{\geq 0}$ admits geometric realizations for simplicial objects.

If F preserves finite coproducts and geometric realizations of simplicial objects, then F is right exact (Lemma 14.8). Conversely, suppose that F is right exact; we wish to prove that F preserves geometric realizations of simplicial objects. It will suffice to show that each composition

$$\mathcal{C}_{\geq 0} \xrightarrow{F} \mathcal{C}'_{\geq 0} \xrightarrow{\tau_{\leq n}} (\mathcal{C}'_{\geq 0})_{\leq n}$$

preserves geometric realizations of simplicial objects. We observe that, in virtue of the right exactness of F, this functor is equivalent to the composition

$$\mathcal{C}_{\leq 0} \xrightarrow{\tau_{\leq n}} (\mathcal{C}_{\geq 0})_{\leq n} \xrightarrow{\tau_{\leq n} \circ F} (\mathcal{C}'_{\geq 0})_{\leq n}.$$

It will therefore suffice to prove that $\tau_{\leq n} \circ F$ preserves geometric realizations of simplicial objects, which follows from Lemma 14.8 since both the source and target are equivalent to n-categories. □

Lemma 14.10. *Let $\mathcal{A}$ be a small abelian category with enough projective objects, and let $\mathcal{C} \subseteq \mathcal{P}_{\Sigma}(\mathrm{N}(\mathcal{A}_0))$ be the essential image of $\mathcal{D}^{-}_{\geq 0}(\mathcal{A}) \subseteq \mathcal{D}^{-}_{\geq 0}(\mathcal{A}^{\vee})$ under the equivalence $\psi : \mathcal{D}^{-}_{\geq 0}(\mathcal{A}^{\vee}) \to \mathcal{P}_{\Sigma}(\mathrm{N}(\mathcal{A}_0))$ of Proposition 14.4. Then $\mathcal{C}$ is the smallest full subcategory of $\mathcal{P}(\mathrm{N}(\mathcal{A}_0))$ which is closed under geometric realization and contains the essential image of the Yoneda embedding.*

Proof. It is clear that $\mathcal{C}$ contains the essential image of the Yoneda embedding. Lemma 14.9 implies that $\mathcal{D}^{-}_{\geq 0}(\mathcal{A})$ admits geometric realizations and that the inclusion $\mathcal{D}^{-}_{\geq 0}(\mathcal{A}) \subseteq \mathcal{D}^{-}_{\geq 0}(\mathcal{A}^{\vee})$ preserves geometric realizations. It follows that $\mathcal{C}$ is closed under geometric realizations in $\mathcal{P}(\mathrm{N}(\mathcal{A}_0))$.

To complete the proof, we will show that every object of $X \in \mathcal{D}^{-}_{\geq 0}(\mathcal{A})$ can be obtained as the geometric realization, in $\mathcal{D}^{-}_{\geq 0}(\mathcal{A}^{\vee})$, of a simplicial object $P_{\bullet}$ such that each $P_n \in \mathcal{D}^{-}_{\geq 0}(\mathcal{A}^{\vee})$ consists of a projective object of $\mathcal{A}$, concentrated in degree zero. In fact, we can take $P_{\bullet}$ to be the simplicial object of $\mathcal{A}_0$ which corresponds to $X \in \mathrm{Ch}_{\geq 0}(\mathcal{A}_0)$ under the Dold-Kan correspondence. It follows from Theorem T.4.2.4.1 and Proposition T.5.5.9.14 that X can be identified with the geometric realization of $P_{\bullet}$. □

We are now ready to establish our characterization of $\mathcal{D}^{-}_{\geq 0}(\mathcal{A})$.

Theorem 14.11. *Let $\mathcal{A}$ be an abelian category with enough projective objects, $\mathcal{A}_0 \subseteq \mathcal{A}$ the full subcategory spanned by the projective objects, and $\mathcal{C}$ an arbitrary ∞-category which admits geometric realizations. Let* $\mathrm{Fun}'(\mathcal{D}^{-}_{\geq 0}(\mathcal{A}), \mathcal{C})$ *denote the full subcategory of* $\mathrm{Fun}(\mathcal{D}^{-}_{\geq 0}(\mathcal{A}), \mathcal{C})$ *spanned by those functors which preserve geometric realizations. Then:*

(1) *The restriction map*

$$\mathrm{Fun}'(\mathcal{D}^-_{\geq 0}(\mathcal{A}), \mathcal{C}) \to \mathrm{Fun}(\mathrm{N}(\mathcal{A}_0), \mathcal{C})$$

is an equivalence of ∞*-categories.*

(2) *A functor* $F \in \mathrm{Fun}'(\mathcal{D}^-_{\geq 0}(\mathcal{A}), \mathcal{C})$ *preserves preserves finite coproducts if and only if the restriction* $F|\,\mathrm{N}(\mathcal{A}_0)$ *preserves finite coproducts.*

Proof. Part (1) follows from Lemma 14.10, Remark T.5.3.5.9, and Proposition T.4.3.2.15. The "only if" direction of (2) is obvious. To prove the "if" direction, let us suppose that $F|\,\mathrm{N}(\mathcal{A}_0)$ preserves finite coproducts. We may assume without loss of generality that $\mathcal{C}$ admits filtered colimits (Lemma T.5.3.5.7), so that F extends to a functor $F' : \mathcal{D}^-_{\geq 0}(\mathcal{A}^\vee)$ which preserves filtered colimits and geometric realizations (Propositions 14.4 and T.5.5.8.15). It follows from Proposition T.5.5.8.15 that F' preserves finite coproducts, so that $F = F'|\,\mathcal{D}^-_{\geq 0}(\mathcal{A})$ also preserves finite coproducts. □

Corollary 14.12. *Let* $\mathcal{A}$ *be an abelian category with enough projective objects, and let* $\mathcal{C}$ *be a stable* ∞*-category equipped with a left complete t-structure. Then the restriction functor*

$$\mathrm{Fun}(\mathcal{D}^-(\mathcal{A}), \mathcal{C}) \to \mathrm{Fun}(\mathrm{N}(\mathcal{A}_0), \mathcal{C})$$

induces an equivalence from the full subcategory of $\mathrm{Fun}(\mathcal{D}^-(\mathcal{A}), \mathcal{C})$ *spanned by the right t-exact functors to the full subcategory of* $\mathrm{Fun}(\mathrm{N}(\mathcal{A}_0), \mathcal{C}_{\geq 0})$ *spanned by functors which preserve finite coproducts* (*here* $\mathcal{A}_0$ *denotes the full subcategory of* $\mathcal{A}$ *spanned by the projective objects*)*.*

Proof. Let $\mathrm{Fun}'(\mathcal{D}^-(\mathcal{A}), \mathcal{C})$ be the full subcategory of $\mathrm{Fun}(\mathcal{D}^-(\mathcal{A}), \mathcal{C})$ spanned by the right t-exact functors. Lemma 14.9 implies that $\mathrm{Fun}'(\mathcal{D}^-(\mathcal{A}), \mathcal{C})$ is equivalent (via restriction) to the full subcategory

$$\mathrm{Fun}'(\mathcal{D}^-_{\geq 0}(\mathcal{A}), \mathcal{C}_{\geq 0}) \subseteq \mathrm{Fun}(\mathcal{D}^-_{\geq 0}(\mathcal{A}), \mathcal{C}_{\geq 0})$$

spanned by those functors which preserve finite coproducts and geometric realizations of simplicial objects. Theorem 14.11 and Proposition T.5.5.8.15 allow us to identify $\mathrm{Fun}'(\mathcal{D}^-_{\geq 0}(\mathcal{A}), \mathcal{C}_{\geq 0})$ with the ∞-category of finite-coproduct preserving functors from $\mathrm{N}(\mathcal{A}_0)$ into $\mathcal{C}_{\geq 0}$. □

Corollary 14.13. *Let* $\mathcal{A}$ *be an abelian category with enough projective objects, let* $\mathcal{C}$ *be a stable* ∞*-category equipped with a left complete t-structure, and let* $\mathcal{E} \subseteq \mathrm{Fun}(\mathcal{D}^-(\mathcal{A}), \mathcal{C})$ *be the full subcategory spanned by those right t-exact functors which carry projective objects of* $\mathcal{A}$ *into the heart of* $\mathcal{C}$*. Then* $\mathcal{E}$ *is equivalent to (the nerve of) the ordinary category of right exact functors from* $\mathcal{A}$ *to the heart of* $\mathcal{C}$*.*

Proof. Corollary 14.12 implies that the restriction map

$$\mathcal{E} \to \mathrm{Fun}(\mathrm{N}(\mathcal{A}_0), \mathcal{C}^\heartsuit)$$

is fully faithful, and that the essential image of θ consists of the collection of coproduct-preserving functors from $\mathrm{N}(\mathcal{A}_0)$ to $\mathcal{C}^\heartsuit$. Lemma 14.1 allows us to identify the latter ∞-category with the nerve of the category of right exact functors from $\mathcal{A}$ to the heart of $\mathcal{C}$. □

If $\mathcal{A}$ and $\mathcal{C}$ are as in Proposition 14.12, then any right exact functor from $\mathcal{A}$ to $\mathcal{C}^\heartsuit$ can be extended (in an essentially unique way) to a functor $\mathcal{D}^-(\mathcal{A}) \to \mathcal{C}$. In particular, if the abelian category $\mathcal{C}^\heartsuit$ has enough projective objects, then we obtain an induced map $\mathcal{D}^-(\mathcal{C}^\heartsuit) \to \mathcal{C}$.

Example 14.14. Let $\mathcal{A}$ and $\mathcal{B}$ be abelian categories equipped with enough projective objects. Then any right-exact functor $f : \mathcal{A} \to \mathcal{B}$ extends to a right t-exact functor $F : \mathcal{D}^-(\mathcal{A}) \to \mathcal{D}^-(\mathcal{B})$. One typically refers to F as the *left derived functor* of f.

Example 14.15. Let Sp be the stable ∞-category of spectra (see §9), with its natural t-structure. Then the heart of Sp is equivalent to the category $\mathcal{A}$ of abelian groups. We therefore obtain a functor $\mathcal{D}^-(\mathcal{A}) \to \mathrm{Sp}$, which carries a complex of abelian groups to the corresponding *generalized Eilenberg-MacLane spectrum.*

15 Presentable Stable ∞-Categories

In this section, we will study the class of *presentable* stable ∞-categories: that is, stable ∞-categories which admit small colimits and are generated (under colimits) by a set of small objects. In the stable setting, the condition of presentability can be formulated in reasonably simple terms.

Proposition 15.1. (1) *A stable ∞-category $\mathcal{C}$ admits small colimits if and only if $\mathcal{C}$ admits small coproducts.*

(2) *Let $F : \mathcal{C} \to \mathcal{D}$ be an exact functor between stable ∞-categories satisfying* (1)*. Then F preserves small colimits if and only if F preserves small coproducts.*

(3) *Let $\mathcal{C}$ be a stable ∞-category satisfying* (1)*, and let X be an object of $\mathcal{C}$. Then X is compact if and only if the following condition is satisfied:*

($*$) *For every map $f : X \to \coprod_{\alpha \in A} Y_\alpha$ in $\mathcal{C}$, there exists a finite subset $A_0 \subseteq A$ such that f factors (up to homotopy) through $\coprod_{\alpha \in A_0} Y_\alpha$.*

Proof. The "only if" direction of (1) is obvious, and the converse follows from Proposition T.4.4.3.2. Assertion (2) can be proven in the same way.

The "only if" direction of (3) follows from the fact that an arbitrary coproduct $\coprod_{\alpha \in A} Y_\alpha$ can be obtained as a filtered colimit of finite coproducts $\coprod_{\alpha \in A_0} Y_\alpha$ (see §T.4.2.3). Conversely, suppose that an object $X \in \mathcal{C}$ satisfies ($*$); we wish to show that X is compact. Let $f : \mathcal{C} \to \widehat{\mathcal{S}}$ be the functor corepresented by C (recall that $\hat{\mathcal{S}}$ denotes the ∞-category of spaces which are not necessarily small). Proposition T.5.1.3.2 implies that f is left exact. According to Proposition 10.12, we can assume that $f = \Omega^\infty \circ F$, where $F : \mathcal{C} \to \widehat{\mathrm{Sp}}$ is an exact functor; here $\widehat{\mathrm{Sp}}$ denotes the ∞-category of spectra which are not necessarily small. We wish to prove that f preserves filtered colimits. Since Ω^∞ preserves filtered colimits, it will suffice to show that F preserves *all* colimits. In view of (2), it will suffice to show that F preserves coproducts. In virtue of Remark 9.10, we are reduced to showing that each of the induced functors

$$\mathcal{C} \xrightarrow{F} \widehat{\mathrm{Sp}} \xrightarrow{\pi_n} \mathrm{N}(\mathcal{A}b)$$

preserves coproducts, where $\mathcal{A}b$ denotes the category of (not necessarily small) abelian groups. Shifting if necessary, we may suppose $n = 0$. In other words, we must show that for any collection of objects $\{Y_\alpha\}_{\alpha \in A}$, the natural map

$$\theta : \bigoplus \mathrm{Ext}^0_{\mathcal{C}}(X, Y_\alpha) \to \mathrm{Ext}^0_{\mathcal{C}}(X, \coprod Y_\alpha)$$

is an isomorphism of abelian groups. The surjectivity of θ amounts to the assumption ($*$), while the injectivity follows from the observations that each Y_α is a retract of the coproduct $\coprod Y_\alpha$ and that the natural map $\bigoplus \mathrm{Ext}^0_{\mathcal{C}}(X, Y_\alpha) \to \prod \mathrm{Ext}^0_{\mathcal{C}}(X, Y_\alpha)$ is injective. □

If $\mathcal{C}$ is a stable ∞-category, then we will say that an object $X \in \mathcal{C}$ *generates* $\mathcal{C}$ if the condition $\pi_0 \mathrm{Map}_{\mathcal{C}}(X, Y) \simeq *$ implies that Y is a zero object of $\mathcal{C}$.

Corollary 15.2. *Let $\mathcal{C}$ be a stable ∞-category. Then $\mathcal{C}$ is presentable if and only if the following conditions are satisfied:*

(1) *The ∞-category $\mathcal{C}$ admits small coproducts.*

(2) *The homotopy category* $\mathrm{h}\mathcal{C}$ *is locally small.*

(3) *There exists regular cardinal κ and a κ-compact generator $X \in \mathcal{C}$.*

Proof. Suppose first that $\mathcal{C}$ is presentable. Conditions (1) and (2) are obvious. To establish (3), we may assume without loss of generality that $\mathcal{C}$ is an accessible localization of $\mathcal{P}(\mathcal{D})$, for some small ∞-category $\mathcal{D}$. Let $F : \mathcal{P}(\mathcal{D}) \to \mathcal{C}$ be the localization functor and G its right adjoint. Let $j : \mathcal{D} \to \mathcal{P}(\mathcal{D})$ be the Yoneda embedding, and let X be a coproduct of all suspensions (see §3) of objects of the form $F(j(D))$, where $D \in \mathcal{D}$. Since $\mathcal{C}$ is presentable, X is κ-compact provided that κ is sufficiently large. We claim that X generates $\mathcal{C}$. To prove this, we consider an arbitrary $Y \in \mathcal{C}$ such that $\pi_0 \operatorname{Map}_{\mathcal{C}}(X, Y) \simeq *$. It follows that the space

$$\operatorname{Map}_{\mathcal{C}}(F(j(D)), Y) \simeq \operatorname{Map}_{\mathcal{P}(\mathcal{D})}(j(D), G(Y)) \simeq G(Y)(D)$$

is contractible for all $D \in \mathcal{D}$, so that $G(Y)$ is a final object of $\mathcal{P}(\mathcal{D})$. Since G is fully faithful, we conclude that Y is a final object of $\mathcal{C}$, as desired.

Conversely, suppose that (1), (2), and (3) are satisfied. We first claim that $\mathcal{C}$ is itself locally small. It will suffice to show that for every morphism $f : X \to Y$ in $\mathcal{C}$ and every $n \geq 0$, the homotopy group $\pi_n(\operatorname{Hom}^{\mathrm{R}}_{\mathcal{C}}(X, Y), f)$ is small. We note that $\operatorname{Hom}^{\mathrm{R}}_{\mathcal{C}}(X, Y)$ is equivalent to the loop space of $\operatorname{Hom}^{\mathrm{R}}_{\mathcal{C}}(X, Y[1])$; the question is therefore independent of base point, so we may assume that f is the zero map. We conclude that the relevant homotopy group can identified with $\operatorname{Hom}_{\mathrm{h}\mathcal{C}}(X[n], Y)$, which is small in virtue of assumption (2).

Fix a regular cardinal κ and a κ-compact object X which generates $\mathcal{C}$. We now define a transfinite sequence of full subcategories

$$\mathcal{C}(0) \subseteq \mathcal{C}(1) \subseteq \ldots$$

as follows. Let $\mathcal{C}(0)$ be the full subcategory of $\mathcal{C}$ spanned by the objects $\{X[n]\}_{n \in \mathbf{Z}}$. If λ is a limit ordinal, let $\mathcal{C}(\lambda) = \bigcup_{\beta < \lambda} \mathcal{C}(\beta)$. Finally, let $\mathcal{C}(\alpha + 1)$ be the full subcategory of $\mathcal{C}$ spanned by all objects which can be obtained as the colimit of κ-small diagrams in $\mathcal{C}(\alpha)$. Since $\mathcal{C}$ is locally small, it follows that each $\mathcal{C}(\alpha)$ is essentially small. It follows by induction that each $\mathcal{C}(\alpha)$ consists of κ-compact objects of $\mathcal{C}$ and is stable under translation. Finally, we observe that $\mathcal{C}(\kappa)$ is stable under κ-small colimits. It follows from Lemma 4.3 that $\mathcal{C}(\kappa)$ is a stable subcategory of $\mathcal{C}$. Choose a small ∞-category $\mathcal{D}$ and an equivalence $f : \mathcal{D} \to \mathcal{C}(\kappa)$. According to Proposition T.5.3.5.11, we may suppose that f factors as a composition

$$\mathcal{D} \xrightarrow{j} \operatorname{Ind}_\kappa(\mathcal{D}) \xrightarrow{F} \mathcal{C}$$

where j is the Yoneda embedding and F is a κ-continuous, fully faithful functor. We will complete the proof by showing that F is an equivalence.

Proposition T.5.5.1.9 implies that F preserves small colimits. It follows that F admits a right adjoint $G : \mathcal{C} \to \operatorname{Ind}_\kappa(\mathcal{D})$ (Remark T.5.5.2.10). We wish to show that the counit map $u : F \circ G \to \mathrm{id}_{\mathcal{C}}$ is an equivalence of functors. Choose an object $Z \in \mathcal{C}$, and let Y be a cokernel for the induced map $u_Z : (F \circ G)(Z) \to Z$. Since F is fully faithful, $G(u_Z)$ is an equivalence. Because G is an exact functor, we deduce that $G(Y) = 0$. It follows that $\operatorname{Map}_{\mathcal{C}}(F(D), Y) \simeq \operatorname{Map}_{\operatorname{Ind}_\kappa(\mathcal{D})}(D, G(Y)) \simeq *$ for all $D \in \operatorname{Ind}_\kappa(\mathcal{D})$. In particular, we conclude that $\pi_0 \operatorname{Map}_{\mathcal{C}}(X, Y) \simeq *$. Since X generates $\mathcal{C}$, we deduce that $Y \simeq 0$. Thus u_Z is an equivalence as desired. □

Remark 15.3. In view of Proposition 15.1 and Corollary 15.2, the hypothesis that a stable ∞-category $\mathcal{C}$ be compactly generated can be formulated entirely in terms of the homotopy category $\mathrm{h}\mathcal{C}$. Consequently, one can study this condition entirely in the setting of triangulated categories, without making reference to (or assuming the existence of) an underlying stable ∞-category. We refer to reader to [55] for further discussion.

The following result gives a good class of examples of presentable ∞-categories.

Proposition 15.4. *Let $\mathcal{C}$ and $\mathcal{D}$ be presentable ∞-categories, and suppose that $\mathcal{D}$ is stable.*

(1) *The ∞-category $\operatorname{Stab}(\mathcal{C})$ is presentable.*

(2) *The functor $\Omega^\infty : \operatorname{Stab}(\mathcal{C}) \to \mathcal{C}$ admits a left adjoint $\Sigma^\infty : \mathcal{C} \to \operatorname{Stab}(\mathcal{C})$.*

(3) *An exact functor* $G : \mathcal{D} \to \mathrm{Stab}(\mathcal{C})$ *admits a left adjoint if and only if* $\Omega^\infty \circ G : \mathcal{D} \to \mathcal{C}$ *admits a left adjoint.*

Proof. We first prove (1). Assume that $\mathcal{C}$ is presentable, and let 1 be a final object of $\mathcal{C}$. Then $\mathcal{C}_*$ is equivalent to $\mathcal{C}_{1/}$, and therefore presentable (Proposition T.5.5.3.11). The loop functor $\Omega : \mathcal{C}_* \to \mathcal{C}_*$ admits a left adjoint $\Sigma : \mathcal{C}_* \to \mathcal{C}_*$. Consequently, we may view the tower

$$\ldots \xrightarrow{\Omega} \mathcal{C}_* \xrightarrow{\Omega} \mathcal{C}_*$$

as a diagram in the ∞-category $\mathcal{P}\mathrm{r}^{\mathrm{R}}$. Invoking Theorem T.5.5.3.18, we deduce (1) and the following modified versions of (2) and (3):

($2'$) The functor $\Omega^\infty_* : \mathrm{Stab}(\mathcal{C}) \to \mathcal{C}_*$ admits a left adjoint $\Sigma^\infty_* : \mathcal{C}_* \to \mathrm{Stab}(\mathcal{C})$.

($3'$) An exact functor $G : \mathcal{D} \to \mathrm{Stab}(\mathcal{C})$ admits a left adjoint if and only if $\Omega^\infty_* \circ G : \mathcal{D} \to \mathcal{C}_*$ admits a left adjoint.

To complete the proof, it will suffice to verify the following:

($2''$) The forgetful functor $\mathcal{C}_* \to \mathcal{C}$ admits a left adjoint $\mathcal{C} \to \mathcal{C}_*$.

($3''$) A functor $G : \mathcal{D} \to \mathcal{C}_*$ admits a left adjoint if and only if the composition $\mathcal{D} \xrightarrow{G} \mathcal{C}_* \to \mathcal{C}$ admits a left adjoint.

To prove ($2''$) and ($3''$), we recall that a functor G between presentable ∞-categories admits a left adjoint if and only if G preserves small limits and small, κ-filtered colimits, for some regular cardinal κ (Corollary T.5.5.2.9). The desired results now follow from Propositions T.4.4.2.9 and T.1.2.13.8. □

Corollary 15.5. *Let* $\mathcal{C}$ *and* $\mathcal{D}$ *be presentable* ∞*-categories, and suppose that* $\mathcal{D}$ *is stable. Then composition with* $\Sigma^\infty : \mathcal{C} \to \mathrm{Stab}(\mathcal{C})$ *induces an equivalence*

$$\mathcal{P}\mathrm{r}^{\mathrm{L}}(\mathrm{Stab}(\mathcal{C}), \mathcal{D}) \to \mathcal{P}\mathrm{r}^{\mathrm{L}}(\mathcal{C}, \mathcal{D}).$$

Proof. This is equivalent to the assertion that composition with Ω^∞ induces an equivalence

$$\mathcal{P}\mathrm{r}^{\mathrm{R}}(\mathcal{D}, \mathrm{Stab}(\mathcal{C})) \to \mathcal{P}\mathrm{r}^{\mathrm{R}}(\mathcal{D}, \mathcal{C}),$$

which follows from Propositions 10.12 and 15.4. □

Using Corollary 15.5, we obtain another characterization of the ∞-category of spectra. Let $S \in \mathrm{Sp}$ denote the image under $\Sigma^\infty : \mathcal{S} \to \mathrm{Sp}$ of the final object $* \in \mathcal{S}$. We will refer to S as the *sphere spectrum.*

Corollary 15.6. *Let* $\mathcal{D}$ *be a stable, presentable* ∞*-category. Then evaluation on the sphere spectrum induces an equivalence of* ∞*-categories*

$$\theta : \mathcal{P}\mathrm{r}^{\mathrm{L}}(\mathrm{Sp}, \mathcal{D}) \to \mathcal{D}\,.$$

In other words, we may regard the ∞-category Sp as the stable ∞-category which is freely generated, under colimits, by a single object.

Proof. We can factor the evaluation map θ as a composition

$$\mathcal{P}\mathrm{r}^{\mathrm{L}}(\mathrm{Stab}(\mathcal{S}), \mathcal{D}) \xrightarrow{\theta'} \mathcal{P}\mathrm{r}^{\mathrm{L}}(\mathcal{S}, \mathcal{D}) \xrightarrow{\theta''} \mathcal{D}$$

where θ' is given by composition with Σ^∞ and θ'' by evaluation at the final object of $\mathcal{S}$. We now observe that θ' and θ'' are both equivalences of ∞-categories (Corollary 15.5 and Theorem T.5.1.5.6). □

We conclude this section by establishing a characterization of the class of stable, presentable ∞-categories.

Lemma 15.7. *Let $\mathcal{C}$ be a stable ∞-category, and let $\mathcal{C}' \subseteq \mathcal{C}$ be a localization of $\mathcal{C}$. Let $L : \mathcal{C} \to \mathcal{C}'$ be a left adjoint to the inclusion. Then L is left exact if and only if $\mathcal{C}'$ is stable.*

Proof. The "if" direction follows from Proposition 5.1, since L is right exact. Conversely, suppose that L is left exact. Since $\mathcal{C}'$ is a localization of $\mathcal{C}$, it is closed under finite limits. In particular, it is closed under the formation of kernels and contains a zero object of $\mathcal{C}$. To complete the proof, it will suffice to show that $\mathcal{C}'$ is stable under the formation of pushouts in $\mathcal{C}$. Choose a pushout diagram $\sigma : \Delta^1 \times \Delta^1 \to \mathcal{C}$

$$\begin{tikzcd}
X \arrow[r] \arrow[d] & X' \arrow[d] \\
Y \arrow[r] & Y'
\end{tikzcd}$$

in $\mathcal{C}$, where $X, X', Y \in \mathcal{C}'$. Proposition 4.4 implies that σ is also a pullback square. Let $L : \mathcal{C} \to \mathcal{C}'$ be a left adjoint to the inclusion. Since L is left exact, we obtain a pullback square $L(\sigma)$:

$$\begin{tikzcd}
LX \arrow[r] \arrow[d] & LX' \arrow[d] \\
LY \arrow[r] & LY'.
\end{tikzcd}$$

Applying Proposition 4.4 again, we deduce that $L(\sigma)$ is a pushout square in $\mathcal{C}$. The natural transformation $\sigma \to L(\sigma)$ is an equivalence when restricted to Λ^2_0, and therefore induces an equivalence $Y' \to LY'$. It follows that Y' belongs to the essential image of $\mathcal{C}'$, as desired. □

Lemma 15.8. *Let $\mathcal{C}$ be a stable ∞-category, $\mathcal{D}$ an ∞-category which admits finite limits, and $G : \mathcal{C} \to$ $\mathrm{Stab}(\mathcal{D})$ an exact functor. Suppose that $g = \Omega^\infty \circ G : \mathcal{C} \to \mathcal{D}$ is fully faithful. Then G is fully faithful.*

Proof. It will suffice to show that each of the composite maps

$$g_n : \mathcal{C} \to \mathrm{Stab}(\mathcal{D}) \overset{\Omega_*^{\infty-n}}{\to} \mathcal{D}_*$$

is fully faithful. Since g_n can be identified with $g_{n+1} \circ \Omega$, where $\Omega : \mathcal{C} \to \mathcal{C}$ denotes the loop functor, we can reduce to the case $n = 0$. Fix objects $C, C' \in \mathcal{C}$; we will show that the map $\mathrm{Map}_{\mathcal{C}}(C, C') \to$ $\mathrm{Map}_{\mathcal{D}_*}(g_0(C), g_0(C'))$ is a homotopy equivalence. We have a homotopy fiber sequence

$$\mathrm{Map}_{\mathcal{D}_*}(g_0(C), g_0(C')) \xrightarrow{\theta} \mathrm{Map}_{\mathcal{D}}(g(C), g(C')) \to \mathrm{Map}_{\mathcal{D}}(*, g(C')).$$

Here $*$ denotes a final object of $\mathcal{D}$. Since g is fully faithful, it will suffice to prove that θ is a homotopy equivalence. For this, it suffices to show that $\mathrm{Map}_{\mathcal{D}}(*, g(C'))$ is contractible. Since g is left exact, this space can be identified with $\mathrm{Map}_{\mathcal{D}}(g(*), g(C'))$, where $*$ is the final object of $\mathcal{C}$. Invoking once again our assumption that g is fully faithful, we are reduced to proving that $\mathrm{Map}_{\mathcal{C}}(*, C')$ is contractible. This follows from the assumption that $\mathcal{C}$ is pointed (since $*$ is also an initial object of $\mathcal{C}$). □

Proposition 15.9. *Let $\mathcal{C}$ be an ∞-category. The following conditions are equivalent:*

(1) *The ∞-category $\mathcal{C}$ is presentable and stable.*

(2) *There exists a presentable, stable ∞-category $\mathcal{D}$ and an accessible left-exact localization $L : \mathcal{D} \to \mathcal{C}$.*

(3) *There exists a small ∞-category $\mathcal{E}$ such that $\mathcal{C}$ is equivalent to an accessible left-exact localization of* $\mathrm{Fun}(\mathcal{E}, \mathrm{Sp})$.

Proof. The ∞-category Sp is stable and presentable, so for every small ∞-category $\mathcal{E}$, the functor ∞-category $\mathrm{Fun}(\mathcal{E}, \mathrm{Sp})$ is also stable (Proposition 4.1) and presentable (Proposition T.5.5.3.6). This proves (3) $\Rightarrow$ (2). The implication (2) $\Rightarrow$ (1) follows from Lemma 15.7. We will complete the proof by showing that (1) $\Rightarrow$ (3).

Since $\mathcal{C}$ is presentable, there exists a small ∞-category $\mathcal{E}$ and a fully faithful embedding $g : \mathcal{C} \to \mathcal{P}(\mathcal{E})$, which admits a left adjoint (Theorem T.5.5.1.1). Propositions 10.12 and 15.4 implies that g is equivalent to a composition

$$\mathcal{C} \xrightarrow{G} \mathrm{Stab}(\mathcal{P}(\mathcal{E})) \xrightarrow{\Omega^{\infty}} \mathcal{P}(\mathcal{E}),$$

where the functor G admits a left adjoint. Lemma 15.8 implies that G is fully faithful. It follows that $\mathcal{C}$ is an (accessible) left exact localization of $\mathrm{Stab}(\mathcal{P}(\mathcal{E}))$. We now invoke Example 10.13 to identify $\mathrm{Stab}(\mathcal{P}(\mathcal{E}))$ with $\mathrm{Fun}(\mathcal{E}, \mathrm{Sp})$. □

Remark 15.10. Proposition 15.9 can be regarded as an analogue of Giraud's characterization of topoi as left exact localizations of presheaf categories ([2]). Other variations on this theme include the ∞-categorical version of Giraud's theorem (Theorem T.6.1.0.6) and the Gabriel-Popesco theorem for abelian categories (see [54]).

We conclude this section by describing an abstract version of the Brown representability theorem, which is valid in any compactly generated stable ∞-category (as well as many other ∞-categories of interest).

Theorem 15.11 (Brown Representability). *Let $\mathcal{C}$ be a presentable ∞-category containing a set of objects $\{S_\alpha\}_{\alpha \in A}$ with the following properties:*

(*i*) *Each object S_α is a cogroup object of the homotopy category $\mathrm{h}\mathcal{C}$.*

(*ii*) *Each object $S_\alpha \in \mathcal{C}$ is compact.*

(*iii*) *The ∞-category $\mathcal{C}$ is generated by the objects S_α under small colimits.*

Then a functor $F : \mathrm{h}\mathcal{C}^{\mathrm{op}} \to \mathcal{S}\mathrm{et}$ is representable if and only if it satisfies the following conditions:

(*a*) *For every collection of objects C_β in $\mathcal{C}$, the map $F(\coprod_\beta C_\beta) \to \prod_\beta F(C_\beta)$ is a bijection.*

(*b*) *For every pushout square*

$$\begin{tikzcd} C \arrow[r] \arrow[d] & C' \arrow[d] \\ D \arrow[r] & D' \end{tikzcd}$$

in $\mathcal{C}$, the induced map $F(D') \to F(C') \times_{F(C)} F(D)$ is surjective.

Proof of Theorem 15.11. The necessity of conditions (*a*) and (*b*) is obvious. We will prove that these conditions are sufficient. Let $\emptyset$ denote an initial object of $\mathcal{C}$. If S is an object of $\mathcal{C}$ equipped with a map $\epsilon : S \to \emptyset$, we define the suspension $\Sigma(S)$ to be the pushout $\emptyset \coprod_S \emptyset$. Note that $\Sigma(S)$ always has the structure of a cogroup object of $\mathcal{C}$, and therefore a cogroup object of the homotopy category $\mathrm{h}\mathcal{C}$. Each of the objects S_α is equipped with a counit map $S_\alpha \to \emptyset$ (by virtue of (*i*)), so that the suspension ΣS_α is well-defined. Enlarging the collection $\{S_\alpha\}$ if necessary, we may assume that this collection is stable under the formation of suspensions.

We first prove the following:

($*$) Let $f : C \to C'$ be a morphism in $\mathcal{C}$ such that the induced map $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, C) \to \mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, C')$ is an isomorphism, for every index α. Then f is an equivalence in $\mathcal{C}$.

To prove $(*)$, it will suffice to show that for every object $X \in \mathcal{C}$, the map f induces a homotopy equivalence $\phi : \mathrm{Map}_{\mathcal{C}}(X, C) \to \mathrm{Map}_{\mathcal{C}}(X, C')$. The collection of $X \in \mathcal{C}$ is stable under colimits in $\mathcal{C}$. By virtue of assumption (iii), it suffices to consider the case where X is an object of the form S_α. Since S_α is a cogroup object of $\mathrm{h}\mathcal{C}$, we can regard ϕ as a map between homotopy associative H-spaces. It follows that ϕ is a homotopy equivalence if and only if it induces an isomorphism of groups $\pi_n \mathrm{Map}_{\mathcal{C}}(X, C) \to \pi_n \mathrm{Map}_{\mathcal{C}}(X, C')$ for each $n \geq 0$ (here the homotopy groups are taken with respect to the base point given by the H-space structure). Replacing S_α by $\Sigma^n S_\alpha$, we can reduce to the case $n = 0$, in which case we are reduced to the bijectivity of the map $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, C) \to \mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, C')$.

Now suppose that F is a functor satisfying conditions (a) and (b). We will prove the following:

$(*')$ Let $X \in \mathcal{C}$ and let $\eta \in F(X)$. Then there exists a map $f : X \to X'$ in $\mathcal{C}$ and an object $\eta' \in F(X')$ lifting η with the following property: for every index $\alpha \in A$, η' induces a bijection $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, X') \to F(S_\alpha)$.

To prove $(*')$, we begin by defining X_0 to be the coproduct of X with $\coprod_{\alpha \in A, \gamma \in F(S_\alpha)} S_\alpha$. Using (a), we deduce the existence element $\eta_0 \in F(X_0)$ lifting η. By construction, η_0 induces a *surjection* $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, X_0) \to F(S_\alpha)$ for each index α.

We now define a sequence of morphisms $X_0 \to X_1 \to X_2 \to \cdots$ and a compatible family of elements $\eta_n \in F(X_n)$ using induction on n. Suppose that X_n and η_n have already been constructed. For each index $\alpha \in A$, let K_α be the kernel of the group homomorphism $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, X_n) \to F(S_\alpha)$, and define X_{n+1} to fit into a pushout diagram

$$\begin{tikzcd}
\coprod_{\alpha \in A, \gamma \in K_\alpha} S_\alpha \arrow[r] \arrow[d] & \emptyset \arrow[d] \\
X_n \arrow[r] & X_{n+1}
\end{tikzcd}$$

where the upper horizontal map is given by the counit on each S_α. The existence of a point $\eta_{n+1} \in F(X_{n+1})$ lifting η_n follows from assumption (a).

Let $X' = \varinjlim_n X_n$. We have a pushout diagram

$$\begin{tikzcd}
\coprod_n X_n \arrow[r] \arrow[d] & \coprod_n X_{2n} \arrow[d] \\
\coprod_n X_{2n+1} \arrow[r] & X'.
\end{tikzcd}$$

Using (a) and (b), we deduce the existence of a point $\eta' \in F(X')$ lifting the sequence $\{\eta_n \in F(X_n)\}$. We claim that η' satisfies the condition described in $(*)$. Fix an index α; we wish to prove that the map $\psi : \mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, X') \to F(S_\alpha)$ is bijective. It is clear that ψ is surjective (since the composite map $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, X_0) \to \mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, X') \to F(S_\alpha)$ is surjective by construction). To prove that ψ is injective, it will suffice to show that the kernel of ψ is trivial (since ψ is a group homomorphism, using the cogroup structure on S_α given by (i)). Fix an element $\gamma \in \ker(\psi)$, represented by a map $f : S_\alpha \to X'$. Assumption (ii) guarantees that S_α is compact, so that f factors through some map $\overline{f} : S_\alpha \to X_n$, which determines an element of the kernel K of the map $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, X_n) \to F(S_\alpha)$. It follows from our construction that the composite map $S_\alpha \vec{X}_n \to X_{n+1}$ factors through the counit of S_α, so that f is the unit element of $\ker(\psi)$. This completes the proof of $(*')$.

Assertion (b) guarantees that $F(\emptyset)$ consists of a single element. Applying $(*')$ in the case $X = \emptyset$, we obtain an element $\eta' \in F(X')$ which induces isomorphisms $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, X') \to F(S_\alpha)$ for each index α. We will complete the proof by showing that η' exhibits F as the functor on $\mathrm{h}\mathcal{C}$ represented by the object X'. In other words, we claim that for every object $Y \in \mathcal{C}$, the element η' induces a bijection $\theta : \mathrm{Hom}_{\mathrm{h}\mathcal{C}}(Y, X') \to F(Y)$.

We begin by showing that θ is surjective. Fix an element $\eta'' \in F(Y)$. Assumption (b) guarantees that (η', η'') determines an element of $F(Y \coprod X')$. Applying assertion $(*')$ to this element, we deduce the existence of a map $X' \coprod Y \to Z$ and an element $\overline{\eta} \in F(Z)$ lifting the pair (η', η'') which induces isomorphisms $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, Z) \to F(S_\alpha)$ for each index α. We have a commutative diagram

$$\begin{tikzcd}
\mathrm{Map}_{\mathrm{h}\mathcal{C}}(S_\alpha, Z) \arrow[rr] \arrow[dr] & & \mathrm{Map}_{\mathrm{h}\mathcal{C}}(S_\alpha, X') \arrow[dl] \\
& F(S_\alpha) &
\end{tikzcd}$$

for each index α, in which the vertical maps are bijective. It follows that the horizontal map is also bijective. Invoking $(*')$, we deduce that $X' \to Z$ is an equivalence. The composite map $Y \to Z \simeq X'$ is then a preimage of η'' in the set $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(Y, X')$.

We now complete the proof by showing that θ is injective. Fix a pair of maps $f, g : Y \to X'$ which determine the same element of $F(Y)$. Form a pushout diagram

$$\begin{tikzcd}
Y \coprod Y \arrow[r, "{(f,g)}"] \arrow[d] & X' \arrow[d] \\
Y \arrow[r] & Z.
\end{tikzcd}$$

Using assumption (b), we deduce that $\eta' \in F(X')$ can be lifted to an element $\overline{\eta} \in F(Z)$. Applying $(*')$, we deduce the existence of a map $Z \to Z'$ and an element $\overline{\eta}' \in F(Z')$ lifting $\overline{\eta}$ and inducing bijections $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(S_\alpha, Z') \to F(S_\alpha)$ for each index α. We have commutative diagrams

$$\begin{tikzcd}
\mathrm{Map}_{\mathrm{h}\mathcal{C}}(S_\alpha, Z') \arrow[rr] \arrow[dr] & & \mathrm{Map}_{\mathrm{h}\mathcal{C}}(S_\alpha, X') \arrow[dl] \\
& F(S_\alpha) &
\end{tikzcd}$$

in which the vertical maps are bijective. It follows that the horizontal maps are also bijective, so that $(*)$ guarantees that the map $h : X' \to Z'$ is an equivalence in $\mathcal{C}$. Since the compositions $h \circ f$ and $h \circ g$ are homotopic, we deduce that f and g are homotopic and therefore represent the same element of $\mathrm{Hom}_{\mathrm{h}\mathcal{C}}(Y, X')$, as desired. □

16 Accessible t-Structures

Let $\mathcal{C}$ be a stable ∞-category. If $\mathcal{C}$ is presentable, then it is reasonably easy to construct t-structures on $\mathcal{C}$: for any small collection of objects $\{X_\alpha\}$ of $\mathcal{C}$, there exists a t-structure *generated* by the objects X_α. More precisely, we have the following result:

Proposition 16.1. *Let $\mathcal{C}$ be a presentable stable ∞-category.*

(1) *If $\mathcal{C}' \subseteq \mathcal{C}$ is a full subcategory which is presentable, closed under small colimits, and closed under extensions, then there exists a t-structure on $\mathcal{C}$ such that $\mathcal{C}' = \mathcal{C}_{\geq 0}$.*

(2) *Let $\{X_\alpha\}$ be a small collection of objects of $\mathcal{C}$, and let $\mathcal{C}'$ be the smallest full subcategory of $\mathcal{C}$ which contains each X_α and is closed under extensions and small colimits. Then $\mathcal{C}'$ is presentable.*

Proof. We will give the proof of (1) and defer the (somewhat technical) proof of (2) until the end of this section. Fix $X \in \mathcal{C}$, and let $\mathcal{C}'_{/X}$ denote the fiber product $\mathcal{C}_{/X} \times_{\mathcal{C}} \mathcal{C}'$. Using Proposition T.5.5.3.12, we deduce that $\mathcal{C}'_{/X}$ is presentable, so that it admits a final object $f : Y \to X$. It follows that composition with f induces a homotopy equivalence

$$\mathrm{Map}_{\mathcal{C}}(Z, Y) \to \mathrm{Map}_{\mathcal{C}}(Z, X)$$

for each $Z \in \mathcal{C}'$. Proposition T.5.2.7.8 implies that $\mathcal{C}'$ is a colocalization of $\mathcal{C}$. Since $\mathcal{C}'$ is stable under extensions, Proposition 6.15 implies the existence of a (uniquely determined) t-structure such that $\mathcal{C}' = \mathcal{C}_{\geq 0}$. □

Definition 16.2. Let $\mathcal{C}$ be a presentable stable ∞-category. We will say that a t-structure on $\mathcal{C}$ is *accessible* if the subcategory $\mathcal{C}_{\geq 0} \subseteq \mathcal{C}$ is presentable.

Proposition 16.1 can be summarized as follows: any small collection of objects $\{X_\alpha\}$ of a presentable stable ∞-category $\mathcal{C}$ determines an accessible t-structure on $\mathcal{C}$, which is minimal among t-structures such that each X_α belongs to $\mathcal{C}_{\geq 0}$.

Definition 16.2 has a number of reformulations:

Proposition 16.3. *Let $\mathcal{C}$ be a presentable stable ∞-category equipped with a t-structure. The following conditions are equivalent:*

(1) *The ∞-category $\mathcal{C}_{\geq 0}$ is presentable (equivalently: the t-structure on $\mathcal{C}$ is accessible).*

(2) *The ∞-category $\mathcal{C}_{\geq 0}$ is accessible.*

(3) *The ∞-category $\mathcal{C}_{\leq 0}$ is presentable.*

(4) *The ∞-category $\mathcal{C}_{\leq 0}$ is accessible.*

(5) *The truncation functor $\tau_{\leq 0} : \mathcal{C} \to \mathcal{C}$ is accessible.*

(6) *The truncation functors $\tau_{\geq 0} : \mathcal{C} \to \mathcal{C}$ is accessible.*

Proof. We observe that $\mathcal{C}_{\geq 0}$ is stable under all colimits which exist in $\mathcal{C}$, and that $\mathcal{C}_{\leq 0}$ is a localization of $\mathcal{C}$. It follows that $\mathcal{C}_{\geq 0}$ and $\mathcal{C}_{\leq 0}$ admit small colimits, so that (1) ⇔ (2) and (3) ⇔ (4). We have a distinguished triangle of functors

$$\tau_{\geq 0} \xrightarrow{\alpha} \mathrm{id}_{\mathcal{C}} \xrightarrow{\beta} \tau_{\leq -1} \to \tau_{\geq 0}[1]$$

in the homotopy category $\mathrm{hFun}(\mathcal{C}, \mathcal{C})$. The collection of accessible functors from $\mathcal{C}$ to itself is stable under shifts and under small colimits. Since $\tau_{\leq 0} \simeq \mathrm{coker}(\alpha)[1]$ and $\tau_{\geq 0} \simeq \mathrm{coker}(\beta)[-1]$, we conclude that (5) ⇔ (6). The equivalence (1) ⇔ (5) follows from Proposition T.5.5.1.2. We will complete the proof by showing that (1) ⇔ (3).

Suppose first that (1) is satisfied. Then $\mathcal{C}_{\geq 1} = \mathcal{C}_{\geq 0}[1]$ is generated under colimits by a set of objects $\{X_\alpha\}$. Let S be the collection of all morphisms f in $\mathcal{C}$ such that $\tau_{\leq 0}(f)$ is an equivalence. Using Proposition 6.15, we conclude that S is generated by $\{0 \to X_\alpha\}$ as a quasisaturated class of morphisms, and therefore also as a strongly saturated class of morphisms (Definition T.5.5.4.5). We now apply Proposition T.5.5.4.15 to conclude that $\mathcal{C}_{\leq 0} = S^{-1}\,\mathcal{C}$ is presentable; this proves (3).

We now complete the proof by showing that (3) ⇒ (1). If $\mathcal{C}_{\leq -1} = \mathcal{C}_{\leq 0}[-1]$ is presentable, then Proposition T.5.5.4.16 implies that S is of small generation (as a strongly saturated class of morphisms). Proposition 6.15 implies that S is generated (as a strongly saturated class) by the morphisms $\{0 \to X_\alpha\}_{\alpha \in A}$, where X_α ranges over the collection of all objects of $\mathcal{C}_{\geq 0}$. It follows that there is a small subcollection $A_0 \subseteq A$ such that S is generated by the morphisms $\{0 \to X_\alpha\}_{\alpha \in A_0}$. Let $\mathcal{D}$ be the smallest full subcategory of $\mathcal{C}$ which contains the objects $\{X_\alpha\}_{\alpha \in A_0}$ and is closed under colimits and extensions. Since $\mathcal{C}_{\geq 0}$ is closed under colimits and extensions, we have $\mathcal{D} \subseteq \mathcal{C}_{\geq 0}$. Consequently, $\mathcal{C}_{\leq -1}$ can be characterized as full subcategory of $\mathcal{C}$ spanned by those objects $Y \in \mathcal{C}$ such that $\mathrm{Ext}^k_{\mathcal{C}}(X, Y)$ for all $k \leq 0$ and $X \in \mathcal{D}$. Propositions 16.1 implies that $\mathcal{D}$ is the collection of nonnegative objects for some accessible t-structure on $\mathcal{C}$. Since the negative objects of this new t-structure coincide with the negative objects of the original t-structure, we conclude that $\mathcal{D} = \mathcal{C}_{\geq 0}$, which proves (1). □

The following result provides a good source of examples of accessible t-structures:

Proposition 16.4. *Let $\mathcal{C}$ be a presentable ∞-category, and let* $\mathrm{Stab}(\mathcal{C})_{\leq -1}$ *be the full subcategory of* $\mathrm{Stab}(\mathcal{C})$ *spanned by those objects X such that $\Omega^\infty(X)$ is a final object of $\mathcal{C}$. Then* $\mathrm{Stab}(\mathcal{C})_{\leq -1}$ *determines an accessible t-structure on* $\mathrm{Stab}(\mathcal{C})$*.*

Proof. Choose a small collection of objects $\{C_\alpha\}$ which generate $\mathcal{C}$ under colimits. We observe that an object $X \in \mathrm{Stab}(\mathcal{C})$ belongs to $\mathrm{Stab}(\mathcal{C})_{\leq -1}$ if and only if each of the spaces

$$\mathrm{Map}_{\mathcal{C}}(C_\alpha, \Omega^\infty(X)) \simeq \mathrm{Map}_{\mathrm{Stab}(\mathcal{C})}(\Sigma^\infty(C_\alpha), X)$$

is contractible. Let $\mathrm{Stab}(\mathcal{C})_{\geq 0}$ be the smallest full subcategory of $\mathrm{Stab}(\mathcal{C})$ which is stable under colimits and extensions, and contains each $\Sigma^\infty(C_\alpha)$. Proposition 16.1 implies that $\mathrm{Stab}(\mathcal{C})_{\geq 0}$ is the collection of nonnegative objects of the desired t-structure on $\mathrm{Stab}(\mathcal{C})$. □

Remark 16.5. The proof of Proposition 16.4 gives another characterization of the t-structure on $\mathrm{Stab}(\mathcal{C})$: the full subcategory $\mathrm{Stab}(\mathcal{C})_{\geq 0}$ is generated, under extensions and colimits, by the essential image of the functor $\Sigma^\infty : \mathcal{C} \to \mathrm{Stab}(\mathcal{C})$.

We conclude this section by completing the proof of Proposition 16.1.

Proof of part (2) *of Proposition 16.1.* Choose a regular cardinal κ such that every object of X_α is κ-compact, and let $\mathcal{C}^\kappa$ denote the full subcategory of $\mathcal{C}$ spanned by the κ-compact objects. Let ${\mathcal{C}'}^\kappa = \mathcal{C}' \cap \mathcal{C}^\kappa$, and let $\mathcal{C}''$ be the smallest full subcategory of $\mathcal{C}'$ which contains ${\mathcal{C}'}^\kappa$ and is closed under small colimits. The ∞-category $\mathcal{C}''$ is κ-accessible, and therefore presentable. To complete the proof, we will show that $\mathcal{C}' \subseteq \mathcal{C}''$. For this, it will suffice to show that $\mathcal{C}''$ is stable under extensions.

Let $\mathcal{D}$ be the full subcategory of $\mathrm{Fun}(\Delta^1, \mathcal{C})$ spanned by those morphisms $f : X \to Y$ where $Y \in \mathcal{C}''$, $X \in \mathcal{C}''[-1]$. We wish to prove that the cokernel functor $\mathrm{coker} : \mathcal{D} \to \mathcal{C}$ factors through $\mathcal{C}''$. Let $\mathcal{D}^\kappa$ be the full subcategory of $\mathcal{D}$ spanned by those morphisms $f : X \to Y$ where both X and Y are κ-compact objects of $\mathcal{C}$. By construction, $\mathrm{coker} \,|\, \mathcal{D}^\kappa$ factors through $\mathcal{C}''$. Since $\mathrm{coker} : \mathcal{D} \to \mathcal{C}$ preserves small colimits, it will suffice to show that $\mathcal{D}$ is generated (under small colimits) by $\mathcal{D}^\kappa$.

Fix an object $f : X \to Y$ in $\mathcal{D}$. To complete the proof, it will suffice to show that the canonical map $(\mathcal{D}^\kappa_{/f})^\triangleright \to \mathcal{D}$ is a colimit diagram. Since $\mathcal{D}$ is stable under colimits in $\mathrm{Fun}(\Delta^1, \mathcal{C})$ and colimits in $\mathrm{Fun}(\Delta^1, \mathcal{C})$ are computed pointwise (Proposition T.5.1.2.2), it will suffice to show that composition with the evaluation maps give colimit diagrams $(\mathcal{D}^\kappa_{/f})^\triangleright \to \mathcal{C}$. Lemma T.5.3.5.8 implies that the maps $({\mathcal{C}'}^\kappa[-1])^\triangleright_{/X} \to \mathcal{C}$, $({\mathcal{C}'}^\kappa)^\triangleright_{/Y} \to \mathcal{C}$ are colimit diagrams. It will therefore suffice to show that the evaluation maps

$$({\mathcal{C}'}^\kappa[-1])_{/X} \overset{\theta}{\leftarrow} (\mathcal{D}^\kappa_{/f}) \overset{\theta'}{\to} ({\mathcal{C}'}^\kappa)_{/Y}$$

are cofinal.

We first show that θ is cofinal. According to Theorem T.4.1.3.1, it will suffice to show that for every morphism $\alpha : X' \to X$ in $\mathcal{C}'[-1]$, where X' is κ-compact, the ∞-category

$$\mathcal{E}_\theta : \mathcal{D}^\kappa_{/f} \times_{{\mathcal{C}'}^\kappa[-1]_{/X}} ({\mathcal{C}'}^\kappa[-1]_{/X})_{X'/}$$

is weakly contractible. For this, it is sufficient to show that $\mathcal{E}_\theta$ is filtered (Lemma T.5.3.1.18).

We will show that $\mathcal{E}_\theta$ is κ-filtered. Let K be a κ-small simplicial set, and $p : K \to \mathcal{E}_\theta$ a diagram; we will extend p to a diagram $\overline{p} : K^\triangleright \to \mathcal{E}_\theta$. We can identify p with two pieces of data:

(*i*) A map $p' : K^\triangleleft \to {\mathcal{C}'}^\kappa[-1]_{/X}$.

(*ii*) A map $p'' : (K \star \{\infty\}) \times \Delta^1 \to \mathcal{C}$, with the properties that $p'' | (K \star \{\infty\}) \times \{0\}$ can be identified with p', $p'' | \{\infty\} \times \Delta^1$ can be identified with f, and $p'' | K \times \{1\}$ factors through ${\mathcal{C}'}^\kappa$.

Let $\overline{p}' : (K^{\triangleleft})^{\triangleright} \to \mathcal{C}'^{\kappa}[-1]_{/X}$ be a colimit of p'. To complete the proof that $\mathcal{E}_{\theta}$ is κ-filtered, it will suffice to show that we can find a compatible extension $\overline{p}'' : (K^{\triangleright} \star \{\infty\}) \times \Delta^1 \to \mathcal{C}$ with the appropriate properties. Let L denote the full simplicial subset of $(K^{\triangleright} \star \{\infty\}) \times \Delta^1)$ spanned by every vertex except $(v, 1)$, where v denotes the cone point of $K^{\triangleright}$. We first choose a map $q : L \to \mathcal{C}$ compatible with p'' and $\overline{p}'$. This is equivalent to solving the lifting problem

$$\begin{tikzcd}
 & \mathcal{C}_{/f} \arrow[d] \\
K^{\triangleright} \arrow[r] \arrow[ur, dashed] & \mathcal{C}_{/X},
\end{tikzcd}$$

which is possible since the vertical arrow is a trivial fibration. Let $L' = L \cap (K^{\triangleright} \times \Delta^1)$. Then q determines a map $q_0 : L' \to \mathcal{C}_{/Y}$. Finding the desired extension $\overline{p}''$ is equivalent to finding a map $\overline{q}_0 : L'^{\triangleright} \to \mathcal{C}_{/Y}$, which carries the cone point into $\mathcal{C}'^{\kappa}$.

Let $g : Z \to Y$ be a colimit of q_0 (in the ∞-category $\mathcal{C}_{/Y}$). We observe that Z is a κ-small colimit of κ-compact objects of $\mathcal{C}$, and therefore κ-compact. Since $Y \in \mathcal{C}''$, Y can be written as the colimit of a κ-filtered diagram $\{Y_{\alpha}\}$, taking values in $\mathcal{C}'^{\kappa}$. Since Z is κ-compact, the map g factors through some Y_{α}; it follows that there exists an extension $\overline{q}_0$ as above, which carries the cone point to Y_{α}. This completes the proof that $\mathcal{E}_{\theta}$ is κ-filtered, and also the proof that θ is cofinal.

The proof that θ' is cofinal is similar but slightly easier: it suffices to show that for every map $Y' \to Y$ in $\mathcal{C}'$, where Y' is κ-compact, the fiber product

$$\mathcal{E}_{\theta'} = \mathcal{D}^{\kappa}_{/f} \times_{\mathcal{C}'^{\kappa}_{/Y}} (\mathcal{C}'^{\kappa}_{/Y})_{Y'/}$$

is filtered. For this, we can either argue as above, or simply observe that $\mathcal{E}_{\theta'}$ admits κ-small colimits. □

References

[1] Arone, G. and M. Ching. *Operads and Chain Rules for the Calculus of Functors.* Preprint.

[2] Artin, M. *Théorie des topos et cohomologie étale des schémas.* SGA 4. Lecture Notes in Mathematics 269, Springer-Verlag, Berlin and New York, 1972.

[3] Artin, M. and B. Mazur. *Étale Homotopy.* Lecture Notes in Mathematics 100, Springer-Verlag, Berlin and New York, 1969.

[4] Basterra, M. *André-Quillen cohomology of commutative S-algebras.* Journal of Pure and Applied Algebra 144 (1999) no. 2, 111-143.

[5] Behrend, K. and B. Fantechi. *The intrinsic normal cone.* Inventiones Mathematicae 128 (1997) no. 1, 45-88.

[6] Beilinson, A. , Bernstein, J. and P. Deligne. *Faisceaux pervers.* Asterisuqe 100, Volume 1, 1982.

[7] Bergner, J.E. *A Model Category Structure on the Category of Simplicial Categories.* Transactions of the American Mathematical Society 359 (2007), 2043-2058.

[8] Bergner, J.E. *A survey of* $(\infty, 1)$*-categories.* Available at math.AT/0610239

[9] Bergner, J.E. *Rigidification of algebras over multi-sorted theories.* Algebraic and Geometric Topoogy 7, 2007.

[10] Bergner, J.E. *Three models for the homotopy theory of homotopy theories,* Topology 46 (2007), 397-436.

[11] Bosch, Guntzer, U., and R. Remmert, R. *Non-Archimedean Analysis: a Systematic Approach to Rigid Analytic Geometry.* Springer-Verlag, Berlin and Heidelberg, 1984.

[12] Bousfield, A.K. and D.M. Kan. *Homotopy limits, completions, and localizations.* Lecture Notes in Mathematics 304, Springer-Verlag, 1972.

[13] Cisinski, D-C and I. Moerdijk. *Dendroidal sets as models for homotopy operads.* Available for download as arXiv:0902.1954v1.

[14] Dugger, D. *Combinatorial model categories have presentations.* Advances in Mathematics 164, 2001, 177-201.

[15] Eilenberg, S. and N.E. Steenrod. *Axiomatic approach to homology theory.* Proc. Nat. Acad. Sci. U.S.A. 31, 1945, 117-120.

[16] Eisenbud, D. *Commutative algebra.* Springer-Verlag, New York, 1995.

[17] Elmendorf, A.D., Kriz, I. , Mandell, M.A., and J.P. May. *Rings, modules and algebras in stable homotopy theory.* Mathematical Surveys and Monographs 47, American Mathematical Society, 1997.

[18] Fulton, W. *Algebraic curves.* W.A. Benjamin, Inc., New York, 1969.

[19] Goerss, P. and J.F. Jardine. *Simplicial Homotopy Theory.* Progress in Mathematics, Birkhauser, Boston, 1999.

[20] Goodwillie, T. *Calculus I: The rst derivative of pseudoisotopy theory.* K-Theory 4 (1990), no. 1, 127.

[21] Goodwillie, T. *Calculus. II. Analytic functors.* K-Theory 5 (1991/92), no. 4, 295332.

[22] Goodwillie, T. *Calculus III: Taylor Series.* Geometry and Topology, Volume 7 (2003) 645-711.

[23] Grauert, H. and R. Remmert. *Theory of Stein Spaces.* Springer-Verlag, Berlin Heidelberg, 2004.

[24] Gunning, R. and H. Rossi. *Analytic functions of several complex variables.* Prentice-Hall, Englewood Cliffs, N.J, 1965.

[25] Hatcher, A. *Algebraic Topology.* Cambridge University Press, 2002.

[26] Hook, E.C. *Equivariant cobordism and duality.* Transactions of the American Mathematical Society 178 (1973) 241-258.

[27] Hovey, M. *Model Categories.* Mathematical Surveys and Monographs 63, AMS, Providence, RI, 1999.

[28] Hovey, M., Shipley, B. and J. Smith. *Symmetric spectra.* Journal of the American Mathematical Society 13, 2000, no. 1, 149-208.

[29] Illusie, L. *Complexe cotangent et déformations I.* Lecture Notes in Mathematics 239, Springer-Verlag, 1971.

[30] Illusie, L. *Complexe cotangent et déformations II.* Lecture Notes in Mathematics 283, Springer-Verlag, 1972.

[31] Joyal, A. *Notes on quasi-categories.*

[32] Joyal, A. *Simplicial categories vs. quasi-categories.*

[33] Joyal, A. and M. Tierney. *Quasi-categories vs. Segal Spaces.* Preprint available at math.AT/0607820.

[34] Kerz, M. *The complex of words and Nakaoka stability.* Homology, Homotopy and Applications, volume 7(1), 2005, pp. 77-85.

[35] Klein, J. and J. Rognes. *A chain rule in the calculus of homotopy functors.* Geom. Topol. 6 (2002), 853887.

[36] Knutson, D. *Algebraic spaces.* Lecture Notes in Mathematics 203, Springer-Verlag, 1971.

[37] Laplaza, M. *Coherence for distributivity.* Coherence in categories, 29-65. Lecture Notes in Mathematics 281, Springer-Verlag, 1972.

[38] Laumon, G. and L. Moret-Bailly. *Champs algebriques.* Springer-Verlag, 2000.

[39] Lazard, Daniel. *Sur les modules plats.* C.R. Acad. Sci. Paris 258, 1964, 6313-6316.

[40] Lurie, J. *Higher Topos Theory.* Available for download at http://www.math.harvard.edu/ lurie/ .

[41] Lurie, J. *Derived Algebraic Geometry I: Stable ∞-Categories.* Available for download.

[42] Lurie, J. *Derived Algebraic Geometry II: Noncommutative Algebra.* Available for download.

[43] Lurie, J. *Derived Algebraic Geometry III: Commutative Algebra.* Available for download.

[44] Lurie, J. *Derived Algebraic Geometry IV: Deformation Theory.* Available for download.

[45] Lurie, J. *Derived Algebraic Geometry V: Structured Spaces.* Available for download.

[46] Lurie, J. *Derived Algebraic Geometry VI: $\mathbb{E}[k]$-Algebras.* Available for download.

[47] Lurie, J. *Derived Algebraic Geometry VII: Spectral Schemes.* In preparation.

[48] Lurie, J. $(\infty, 2)$-*Categories and the Goodwillie Calculus I.* Available for download.

[49] Lurie, J. *$(\infty, 2)$-categories and the Goodwillie Calculus II.* In preparation.

[50] Lurie, J. *Elliptic curves in spectral algebraic geometry.* In preparation.

[51] Lurie, J. *Toric varieties, elliptic cohomology at infinity, and loop group representations.* In preparation.

[52] MacLane, S. *Categories for the Working Mathematician.* Second edition. Graduate Txts in Mathematics, 5. Springer-Verlag, New York, 1998.

[53] May, J.P. *The Geometry of Iterated Loop Spaces.*

[54] Mitchell, B. *A quick proof of the Gabriel-Popesco theorem.* Journal of Pure and Applied Algebra 20 (1981), 313-315.

[55] Neeman, A. *Triangulated categories.* Annals of Mathematics Studies, 148. Princeton University Press, 2001.

[56] Quillen, D. *Homotopical Algebra.* Lectures Notes in Mathematics 43, SpringerVerlag, Berlin, 1967.

[57] Rezk, C. *A model for the homotopy theory of homotopy theory.* Transactions of the American Mathematical Society 35 (2001), no. 3, 973-1007.

[58] Rosicky, J. *On Homotopy Varieties.* Advances in Mathematics 214, 2007 no. 2, 525-550.

[59] Schwede, S. *Spectra in model categories and applications to the algebraic cotangent complex.* Journal of Pure and Applied Algebra 120 (1997), pp. 77-104.

[60] Schwede, S. and B. Shipley. *Algebras and Modules in Monoidal Model Categories.* Proceedings of the London Mathematical Society (80) 2000, 491-511.

[61] Schwede, S. and B. Shipley. *Stable model categories are categories of modules.* Topology 42, 2003, no. 1, 103-153.

[62] Serre, Jean-Pierre. *Local algebra.* Springer-Verlag, 2000.

[63] Shipley, B. *A Convenient Model Category for Commutative Ring Spectra.* Homotopy theory: relations with algebraic geometry, group cohomology, and algebraic K-theory. Contemp. Math. volume 346 pp. 473-483, American Mathematical Society, Providence, RI, 2004.

[64] Spivak, D. *Quasi-smooth Derived Manifolds.* PhD dissertation.

[65] Srinivas, V. *Algebraic K-Theory.* Birkhauser, Boston, 1993.

[66] Toën, B. *Champs affines.* Available for download: math.AG/0012219.

[67] Toën, B. *Vers une axiomatisation de la théorie des catégories supériures.* K-theory 34 (2005), no. 3, 233-263.

[68] Toën, B. and G. Vezzosi. *From HAG to DAG: derived moduli stacks.* Available for download: math.AG/0210407.

[69] Toën, B. and G. Vezzosi. *Algebraic geometry over model categories.* Available for download: math.AG/0110109.

[70] Toën, B. and G. Vezzosi. *"Brave New" Algebraic Geometry and global derived moduli spaces of ring spectra.* Available for download: math.AT/0309145.

[71] Toën, B. and G. Vezzosi. *Segal topoi and stacks over Segal categories.* Available for download: math.AG/0212330.

[72] Toën, B. and G. Vezzosi. *A remark on K-theory and S-categories.* Topology 43, No. 4 (2004), 765-791

[73] Verity, D. *Weak complicial sets, a simplicial weak omega-category theory. Part I: basic homotopy theory.*

[74] Verity, D. *Weak complicial sets, a simplicial weak omega-category theory. Part II: nerves of complicial Gray-categories.*

[75] Weibel, C. *An Introduction to Homological Algebra.* Cambridge University Press, 1995.

Superrecursive Features of Interactive Computation

Mark Burgin

University of California, Los Angeles
Los Angeles, CA, 90095, USA

Abstract

Functioning and interaction of distributed devices and concurrent algorithms are analyzed in the context of the theory of algorithms. Our main concern here is how and under what conditions algorithmic interactive devices can be more powerful than the recursive models of computation, such as Turing machines. Realization of such a higher computing power makes these systems superrecursive. We find here five sources for superrecursiveness in interaction. In addition, we prove that when all of these sources are excluded, the algorithmic interactive system in question is able to perform only recursive computations. These results provide computer scientists with necessary and sufficient conditions for achieving superrecursiveness by algorithmic interactive devices.

Keywords: distributed computation, concurrent process, interaction, grid automaton, super-recursive algorithm

1 Introduction

There is a tendency to oppose algorithms and interaction (cf., for example, [17]). This opposition is based on a very restricted understanding of algorithms, which is based on the Church-Turing Thesis that equates algorithms with Turing machines or other mathematical schemas that give rules for computation of a function. Some researchers claim that interactive computation is more powerful than Turing machines (cf., for example, [6, 7, 14, 15, 17]), while others insist that the Church-Turing Thesis still holds (cf., for example, [9]). However, contemporary understanding extends the concept of algorithm, making it closer to the general usage of the word "algorithm". Namely, algorithm is informally perceived as a (finite) structure (e.g., a system of rules) that contains for some performer (class of performers) exact information (e.g., instructions) that allows some performer(s) to pursue a definite goal (cf., for example, [3, 4]).

People need information that is contained in algorithms to make their activity efficient and purposeful. Consequently, one main achievement of 20th century scientific thought was elaboration of the theory of algorithms and computation. This theory studies abstract and real automata, computers and networks, computation and communication. In many ways, this theory is the central cornerstone for computer science. Many key accomplishments in the theory of algorithms and computation converge to the famous Church-Turing Thesis, a statement determining the boundaries of algorithmic computations. The Church-Turing Thesis has long been considered as the most fundamental law within computing. However, recent developments in the theory of algorithms allow overcoming limitations in the Church-Turing Thesis. New mathematical models for algorithms and computation have appeared that extend prior theory in a manner similar to the way relativity theory and quantum mechanics went beyond Newtonian mechanics. These new models are more powerful than the classical recursive algorithm models, i.e., Turing machines, partial recursive functions, Lambda-calculus, and cellular automata.

Algorithms and automata that are more powerful than Turing machines are called super-recursive.

Computations that cannot be realized or simulated by Turing machines are called hyper-computations.

At the first glance, it looks like interactive systems are essentially different from computers and cannot be represented by computing models, such as Turing machines. Really, as Leeuwen and Wiedermann write [13], the purpose of an interactive system is usually not to compute some final result but to react to or interact with the environment in which the system is placed and to maintain a well-defined action-reaction behavior. Interactive systems are always operating and thus, may be seen as machines on infinite strings, but differ in the sense that their inputs are not specified and may depend on intermediate outputs and external sources.

However, if we consider only systems that work with symbolic information, then reaction to or interaction with the environment or with another such system is information transformation and exchange, or communication. In other words, functioning of and interactive system that works with symbolic information consists of computation and communication processes.

In this paper, we analyze sources of an interactive recursive algorithm/machine ability to outperform (have more computing power than) conventional Turing machines, that is, to be able to compute recursively non-computable functions. We find five such sources of interactive superrecursiveness. Namely, interactive superrecursiveness is possible when: 1) the interactive algorithm is itself super-recursive; 2) (Proposition 1) the interactive algorithm is recursive but contains initial information about some recursively non-computable function (has a non-recursive oracle); 3) (Proposition 2) the interactive recursive algorithm interacts with a super-recursive algorithm (a non-recursive environment); 4) (Theorems 2, 3 and 4) time of interaction is not recursively coordinated; 5) (Theorems 5) communication space is not recursively coordinated. The first three cases are not interesting because either superrecursive power comes not from interaction (case 1) or as it is well known, if a recursive device have access to a super-recursive information, then this device can compute recursively non-computable functions.

However, after finding sources of interactive superrecursiveness, it is natural to ask a question if all sources have been found or there are other sources that we have not been able to see. To show that our analysis of interactive superrecursiveness is complete, we prove (Theorems 6) that if interacting algorithms/devices are recursive and their interaction is organized/controlled by a recursive device/algorithms, then computable functions are also recursive.

Thus, we consider algorithms in the form of rules and devices that perform simple and constructive operations at each step and give a result after a finite number of steps (in finite time).

All algorithms are divided into three big classes [3]: *subrecursive*, *recursive*, and *super-recursive*. Algorithms and automata that have the same computing/accepting power [4] as Turing machines are called *recursive*. Examples are partial recursive functions or random access machines.

Algorithms and automata that are weaker than Turing machines, i.e., that can compute fewer functions, are called *subrecursive*. Examples are finite automata, context free grammars or push-down automata.

Algorithms and automata that are more powerful than Turing machines are called *super-recursive*. Examples are inductive Turing machines, Turing machines with oracles or finite-dimensional machines over the field of real numbers.

It is evident that if an interacting algorithm is super-recursive, then even without any interaction, it can perform hypercomputation. Thus, the main question is when taking a recursive algorithm (automaton) and providing for it means for interaction, we can achieve hypercomputation as a result of interaction. There are different models of interactive computational systems: persistent Turing machines [7], a global Turing machine or Internet machine [13, 14], Web machines [1], Web automata [11] and others. The most general, flexible and powerful model of interactive (computational) systems is a grid automaton [2, 3].

In the analysis of the computational power of interaction, it is possible to consider only two systems as interaction of any finite number of systems can be reduced by induction to the case of two systems. Here we do not consider infinite systems.

By the definition of a recursive algorithm (device or machine), a Turing machine can compute any function that a recursive algorithm can compute. That is why we can use Turing machines to explore problems of computational power of interaction. In addition, it is possible to consider Turing machines with one working tape as Turing machines with *n* tapes can be simulated by a Turing machine with one tape [3].

In interaction, machines can change data processed by one of the machines, its software (program of computation) and/or hardware. As Turing machines are utilized as the model of recursive algorithms, hardware is not changed. At the same time, the schema of a universal Turing machine allows one to keep software (rules of computation) in memory in the form of processed data. Consequently, we can assume that only data processed by machines are changed in interaction. That is, interaction goes through memory of machines by changing symbols in memory cells.

2 Turing machines with infinite output and interaction

Considering interactive systems and processes, it is necessary to treat concurrent systems and processes. At the same time, as Palamidessi and Valencia, write [8], infinite behavior is ubiquitous in concurrent systems (e.g., browsers, search engines, reservation systems). Thus, it becomes crucial to study and compare systems with tentatively infinite input and output.

Here we can ask a question how it is possible to compare Turing machines that work and were designed to work with finite words and system that process potentially infinite words. If we consider this question in a primitive way, can come to two opposite but mutually simple conclusions. The first one says that if Turing machines cannot work with infinite words and interactive systems can, then interactive systems are evidently more powerful than Turing machines. The second conclusion is that Turing machines and interactive systems are incomparable.

However, the second conclusion brings us to a paradoxical result. Indeed, in the theory of computation and automata, it is proved that finite automata (finite state machines) are weaker than Turing machines. Nevertheless, there are finite automata (for instance, Büchi and Muller automata [2, 11]) that work with infinite words.

There is a simple solution to this paradox. Namely, to show that actually Turing machines can work with infinite words. Turing machines can do this in the same way as finite automata do. Specifically, if an infinite sequence of symbols is given to a Turing machine, the machine can transform it into an infinite output sequence of symbols, separating the input into finite parts and working with one part at a time. However, this transformation of infinite strings of symbols always has a specific property.

Let us assume that all words in the alphabet of considered Turing machines are ordered and all words are given to the considered Turing machine one by one in this order. Then we have the following result.

Theorem 1. The output of a Turing machine with infinite output (TMIO) is a recursively enumerable set of finite words and any recursively enumerable set of finite words is an output of a Turing machine with infinite output.

This result remains true even if the input words go in an arbitrary but recursively enumerable order.

Corollary 1. If the input of a Turing machine is a recursively enumerable set of finite words, then the corresponding output is also a recursively enumerable set of finite words.

Thus, recursive enumerability is the essence of the Turing machine output even when this output is infinite and input is recursive. More exactly, we have the following result.

Corollary 2. a) If the infinite input string is recursively partitioned (divided) into a set of finite words, then the corresponding output of any Turing machine with this input is also a recursively enumerable set of finite words.

b) If the infinite input string is partitioned (divided) into a set of finite words that is not recursively enumerable, then there is a Turing machine such that working with this input, it will as output a set of finite words that is not be a recursively enumerable.

For convenience, we give here a general structure (schema) of a Turing machine T with infinite input/output and interaction:

$T = (A, R, Q, P, F, q_0 , L_{\mathrm{I}} , L_{\mathrm{w}} , L_{i\mathrm{O}} , i = 1, 2, 3, \dots)$

Here

A is the alphabet of T ;

R is the system of rules of T ;

Q is the set of states of T ;

P is the set of output states of T ;

F is the set of final states of T ;

q_0 is the start of state of T ;

L_{I} is the input tape of T ;

L_{w} is the working tape of T ;

$\{ L_{i\mathrm{O}} , i = 1, 2, 3, \dots \}$ is the system of output tapes of T .

It is necessary to remark that a machine can perform infinite computations but have finite input and finite output.

3 Types of interactive computing systems and processes

Treating here interactive processes in systems of computational devices (automata or algorithms), we consider their activity as information processing in a general case and as computation when we are interested in their computing power.

Time is important parameter of interactive systems in general and computing systems, which usually consist of various interacting devices, in particular. The most popular model is the linear physical time. However, now some physical theories are based on a two-dimensional model of time [22, 25], is used in the theory of databases [27], and branching time plays an important role in computational models [23, 24].

We consider here situations when interactive systems (processes) interact only in form of information exchange, that is, they communicate. In addition, we assume that interaction goes in a communication space, which is a special media designed for communication [8]. In our theoretical model, we use such a communication space as a linear tape of cells, in which one cell can contain one symbol.

Several interactive processes can go on even in one computing device (information processing system with one processor) when different programs realize these processes. Moreover, even one sufficiently complex program can organize many processes. Operating system of a computer is an example of such a program.

It is possible to divide all interactive computations into three types [5]: free interactive computations, partially free interactive computations, and algorithmic or procedural interactive computations.

In turn, there are two types of procedural (algorithmic) concurrent computations: implicitly procedural (algorithmic) concurrent computations and explicitly procedural (algorithmic) concurrent computations.

Definition 1. An interactive computation is called *free* if interactions between processes go without any rules.

For instance, it is demonstrated in [2] that a system of two finite automata interacting without any rules can eventually compute any function. However, when interaction of processes is not specified, at least, by some rules, the enveloping (computational) process can lead to deadlocks, data corruption when different processes change common data without concordance, and other safety violations.

An opposite situation is when all interactions are controlled by definite rules. These rules can be local and global.

Definition 2. An interactive computation (functioning of a grid array/automaton) is called *implicitly procedural* (*algorithmic*) if all interactions between processes go according to local rules (where each set of local rules form an algorithm of local interactions).

For instance, each process (algorithm or device) in a system has its own interaction rules. However, for some processes these rules can coincide.

A more rigid type is explicitly procedural algorithmic computation.

Definition 3. An interactive computation (functioning of a grid array/automaton) is called *explicitly procedural* (*algorithmic*) if all interactions between processes go according to some system of rules (algorithm).

Algorithmic control of interacting processes is naturally considered as an operation with these processes [5].

Definition 4. Explicitly algorithmic functioning of a grid array/automaton is called *algorithmic operation* (AO).

An intermediate situation between free and algorithmic computations is partially free functioning.

Definition 5. An interactive computation (functioning of a grid array/automaton) is called *partially free* if not all interactions between processes are specified by rules.

4 Sources of interactive superrecursiveness

In this section, we describe five main sources of interactive superrecursiveness. For completeness, we present here results describing the first three evident sources. The first one is when the algorithms itself is super-recursive. The second is also simple and is described in the following proposition.

Proposition 1. An interactive recursive algorithm (a Turing machine) *A* can compute a recursively non-enumerable set if it can contain a recursively non-enumerable set as its initial information.

Indeed, taking a recursively non-enumerable set that constitutes initial information, the machine *A* gives this set as its output. To do this, we even do not need such a powerful model as a Turing machine. In this case, superrecursive computation can be realized by a finite automaton *A* that is provided from the start with a recursively non-enumerable set as its input.

Another trivial cause for interactive superrecursiveness is interaction with a system that can send to algorithm (automaton) *A* recursive incomputable information. An example of such a system is a non-recursive oracle, with which a Turing machine can interact [10]. However, we are interested in interaction of algorithms (automata). Consequently, we consider the second system as a super-recursive algorithm.

Proposition 2. An interactive recursive algorithm (a Turing machine) *A* can compute a recursively non-enumerable set if it interacts with a super-recursive algorithm *B*.

Indeed, the algorithm *B* can compute a recursively non-enumerable set and give it to the machine *A*. Then the machine *A* acts as in the previous case. Namely, receiving a recursively non-enumerable input, the machine *A* gives it as its output. To do this, we even do not need such a powerful model as a Turing machine. In this case, superrecursive mode of functioning can be also realized by a finite automaton *A*.

One more source of interactive superrecursiveness is time. This is possible when interaction is not synchronized either because automata (algorithms) send and receive information at random (or at least, at non-recursively coordinated) moments of time or due to incommensurability of the time scales in which automata work.

Theorem 2. An interactive recursive algorithm (a Turing machine) *A* can compute a recursively non-enumerable set if the temporal sequence of information exchanges is recursively non-enumerable.

This result is proved in [2] for the case when interaction of system automata with the communication space is random and thus, non-enumerable [3].

It is necessary to remark that machines with random interaction do not compute a function on infinite words (strings). The result of their computation is a multivalued function. However, it possible to have a deterministic computation on infinite words and still to achieve super-recursive results. To do this, it is necessary to have a source that controls interaction of system automata with the communication space in a super-recursive way, e.g., the sequence of moments when machines have access to this space is non-enumerable.

Theorem 3. An interactive recursive algorithm (a Turing machine) *A* can compute a recursively non-enumerable set when the time scales in which automata work are not commensurable.

What does it mean incommensurability of the time scales of two automata *A* and *B*? In a general case, it means that while the automaton *A* makes some number of moves (say, *n*), the automaton *B* can make any number of moves (say, *m*).

Now let us consider two automata *A* and *B*. The automaton *A* writes the symbol 0 into the common output tape every other move of its functioning, while each odd move the automaton *B* includes writing the symbol 1 into the common output tape. Symbols are written from left to right, and each time the first free cell to the right is filled. When these automata work synchronously, their output will have the form 10101010 … However, when the time scales in which automata work are not commensurable, it is possible that while the automaton *B* makes one move, the automaton *A* makes ten moves. After this, both automata work synchronously. In this case, their output will have the form 0000010101010 … If such a situation can happen not only at the beginning, but also at any stage of computation, the output of *A* and *B* can be not recursively enumerable.

In this case, incommensurability is not uniform. That is, one time when the automaton *A* makes *n* moves, the automaton *B* makes *m* moves, but another time when the automaton *A* makes *n* moves, the automaton *B* makes *k* moves. Formally it means that intervals with the same length in the first time scale can be mapped to intervals with different lengths in the second time scale.

Thus, it is interesting to find whether superrecursiveness is possible when incommensurability of the time scales is uniform. Formally it means that intervals with the same length in the first time scale are always mapped to intervals with equal lengths in the second time scale. When we consider physical time, the time scales are isomorphic to the real line. In this case, incommensurability of the scales means that the time unit of one scale is mapped to some irrational interval in the second scale. Mathematically such a mapping is a stretching or contraction with an irrational parameter.

Theorem 4. An interactive recursive algorithm (a Turing machine) *A* can compute a recursively non-enumerable set when the time scales in which automata work are uniformly incommensurable.

One more source of interactive superrecursiveness is space. This is possible when do not determine where they put their data in the communication space.

Theorem 5. An interactive recursive algorithm (a Turing machine) *A* can compute a recursively non-enumerable set if the sequence of cells where data written by another system is recursively non-enumerable.

To prove this we consider two Turing machines *A* and *B* that interact through a communication space. This space is a linear one-sided tape. While functioning, the machines *A* and *B* write symbols 1 or 0 in cells of the interaction tape.

The machine *B* works in a very simple way. At the step *n*, it puts the symbol 1 into one of two cells with numbers $2n - 1$ and $2n$. The other cell is filled by *A* with the symbol 0. By the assumption of the theorem, it is possible that the sequence of zeros and ones is recursively non-enumerable.

After the pair of cells with numbers $2n - 1$ and $2n$ is filled, the machine *A* gives the output 0 if it was 01 in those two cells and the output 1 if it was 10 in those two cells. When the sequence of zeros and ones in the communication space is recursively non-enumerable, then the output sequence of the machine *A* is also recursively non-enumerable.

5 Interaction with a recursive control

There are two main kinds of algorithmic control of interaction: global and local. In global control, there is an algorithm (automaton) C that controls interaction of A and B. Namely, C determines when and where A and B read from and write into the communication tape. In addition, time scales L_A and L_B of both A and B are synchronized with time scale L_C of C. That is, temporal units 1_A and 1_B (e.g., second, millisecond, etc.) of each scale L_A and L_B are equal to $n1_C$ and $m1_C$ for some whole numbers $n1_A$ and $m1_B$ where 1_C is a temporal unit of the scale L_C.

In local control, each machine A and B determines when and where A and B read from and write into the communication tape. An additional machine C only solves conflicts when both A and B try to write different symbols into the same cell. In addition, all three time scales L_C, L_A and L_B are synchronized.

Theorem 4 shows that local rules are insufficient to restrict the computing power of the interacting recursive devices to recursive computations. Only a global algorithm (device) that organizes interaction can do this.

Theorem 6. An interactive system (grid automaton) R that consists of a finite number of recursive devices (algorithms) interaction of which is organized by a recursive automaton (algorithm) can perform only recursive computations (may be, infinite), i.e., such a system R is equivalent to a Turing machine.

Note that in this case time scales of all automata from R are synchronized with the time scale of the control automaton and thus, they are synchronized with one another.

6 Conclusion

Thus, we have found three trivial (the system itself, initial information or/and another system is super-recursive) and two (or actually, three because the temporal parameter provides two possibilities) non-trivial (time and space) causes for interactive superrecursiveness. In addition, we demonstrate that this list of causes for interactive superrecursiveness is complete.

It would be interesting to study other situations when a system of interacting devices (automata or processes) can have higher computational power than the power of each constituent of this system. Here we did this for systems consisting of Turing machines as it is the most popular model of computation and as it caused the most active controversy. However, the same problem can be studied for subrecursive models, such as finite and pushdown automata, and super-recursive models, such as inductive and limit Turing machines [4].

There are also important problems with analyzing the concept of computation:

What is computation?

Is computation always algorithmic?

Is computation always a physical process?

7 References

[1] Abiteboul, S., and Vianu, V. Queries and computation on the web. *Theor. Comput. Sci.*, v. 239, No. 2, pp. 231-255, 2000.

[2] Büchi, J.R. (1960) Weak second order arithmetic and finite automata, *Z. Math. Logic and Grudl. Math.*, v. 6, No. 1, pp. 66-92

[3] Burgin, M. Multiple computations and Kolmogorov complexity for such processes, *Notices of the Academy of Sciences of the USSR*, 1983, v. 27, No. 2 (v. 269, No.4), pp. 793-797 (translated from Russian)

[4] Burgin M. From Neural networks to Grid Automata, in Proceedings of the IASTED International Conference ”*Modeling and Simulation*”, Palm Springs, California, pp. 307-312, 2003.

[5] Burgin, M. *Super-recursive Algorithms*, Springer, New York, 2005.

[6] Burgin, M., Measuring Power of Algorithms, Programs, and Automata, in Shannon, S. (ed.): *Artificial Intelligence and Computer Science*, Nova Science Publishers, New York, pp. 1-61, 2005

[7] Burgin, M. Algorithmic Control in Concurrent Computations, in Proceedings of the 2006 *International Conference on Foundations of Computer Science*, CSREA Press, Las Vegas, June, 2006, pp. 17-23

[8] Burgin, M., Liu, D., and Karplus, W. *The Problem of Time Scales in Computer Visualization*, in "Computational Science", Lecture Notes in Computer Science, v. 2074, part II, 2001, pp.728-737

[9] Eberbach, E. and Wegner, P. Beyond Turing Machines, *Bulletin of the European Association for Theoretical Computer Science* (EATCS Bulletin) v.81, pp.279-304, 2003

[10] Goldin, D. and Wegner, P. *Persistent Turing Machines*, Brown University Technical Report, 1988

[11] Muller, D. E. (1963) Infinite sequences and finite machines, in *Proceedings of the Fourth Annual Symposium on Switching Circuit Theory and Logical Design*, Chicago, Illinois, IEEE, pp. 3-16

[12] Palamidessi, C., and Valencia, F. D. Recursion vs Replication in Process Calculi, Bulletin of the EATCS 87, pp. 105-125, 2005.

[13] Prasse, M. and Rittgen, P. Why Church's Thesis Still Holds. Some Notes on Peter Wegner's Tracts on Interaction and Computability, The Computer Journal, Vol. 41, No. 6, 1998

[14] Rogers, H. (1987) *Theory of Recursive Functions and Effective Computability,* MIT Press, Cambridge Massachusetts

[15] Spielmann, M., Tyszkiewicz, J. and Van den Bussche, J. (2002). Distributed computation of web queries using automata In *Proceedings of 21th ACM Symposium on Principles of Database Systems (PODS 2002)*, ACM Press

[16] Turing, A. (1939) Systems of Logic Based on Ordinals, *Proc. Lond. Math. Soc.,* Ser.2, v. 45, pp. 161-228

[17] Van Leeuwen, J. and Wiedermann J. (2000) *A computational model of interaction*, Techn. Rep. Dept. of Computer Science, Utrecht University, Utrecht

[18] Van Leeuwen, J. and Wiedermann, J. (2000a) *Breaking the Turing Barrier: The case of the Internet*, Techn. Report, Inst. of Computer Science, Academy of Sciences of the Czech. Rep., Prague

[19] Van Leeuwen, J. and Wiedermann, J. (2000) On the Power of Interactive Computing, Proceedings of the *IFIP Theoretical Computer Science*, pp. 619-623

[20] Vardi, M.Y. and Wolper, P. (1994) Reasoning about Infinite Computations, *Information and Computation*, v. 115, No.1, pp. 1—37.

[21] Wegner, P. (1997) Why interaction is more powerful than algorithms, *Communications of the ACM*, v. 40, pp. 80-91

[22] Bars, I. (2001) Survey of two-time physics, *Classical and Quantum Gravity*, 18 (16). pp. 3113-3130.

[23] de Bakker, J. W., de Roever, W. P. and G. Rozenberg (editors) *Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency.* LNCS 35, Springer-Verlag, 1989

[24] Emerson, E. A. and Halpern, J. Y. Sometimes'' and ``not never'' revisited: on branching vs. linear time, *Journal of the ACM* **33**:1, 1986, pp. 151-178

[25] Hawking, S. W. *A brief history of time*, London, 1988

[26] Nusenoff, R.E. (1976) Two-dimensional time, Philosophical Studies, v. 29, No. 5, pp. 337-341

[27] Seung-Kyum Kim, S.-K., and Chakravarthy, S. (1994) Temporal databases with two-dimensional time: modeling and implementation of multihistory, International Journal on Information Sciences—Informatics and Computer Science, v. 80 , No. 1-2, pp. 43 - 89