Topological Quantum Pogramming
in TED-K

Urs Schreiber on joint work with Hisham Sati

جامعة نيويورك أبوظبي
NYU | ABU DHABI

NYU AD Science Division, Program of Mathematics

& Center for Quantum and Topological Systems

New York University, Abu Dhabi

Center for Quantum& Topological Systems

talk at:

PlanQC 2022 @ Ljubljana, 15 Sep 2022

slides and pointers at: https://ncatlab.org/schreiber/show/Topological+Quantum+Programming+in+TED-K

There are good arguments that

if Quantum Computation is to be a practical reality

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation

BULLETIN (New Series) OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 40, Number 1, Pages 31–38
S 0273-0979(02)00964-3
Article electronically published on October 10, 2002

TOPOLOGICAL QUANTUM COMPUTATION

MICHAEL H. FREEDMAN, ALEXEI KITAEV, MICHAEL J. LARSEN,
AND ZHENGHAN WANG

ABSTRACT. The theory of quantum computation can be constructed from the abstract study of anyonic systems. In mathematical terms, these are unitary topological modular functors. They underlie the Jones polynomial and arise in Witten-Chern-Simons theory. The braiding and fusion of anyonic excitations in quantum Hall electron liquids and 2D-magnets are modeled by modular functors, opening a new possibility for the realization of quantum computers. The chief advantage of anyonic computation would be physical error correction

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation

Das Sarma, MIT Tech Rev (2022):

"*The quantum-bit systems we have today are a tremendous scientific achievement,*

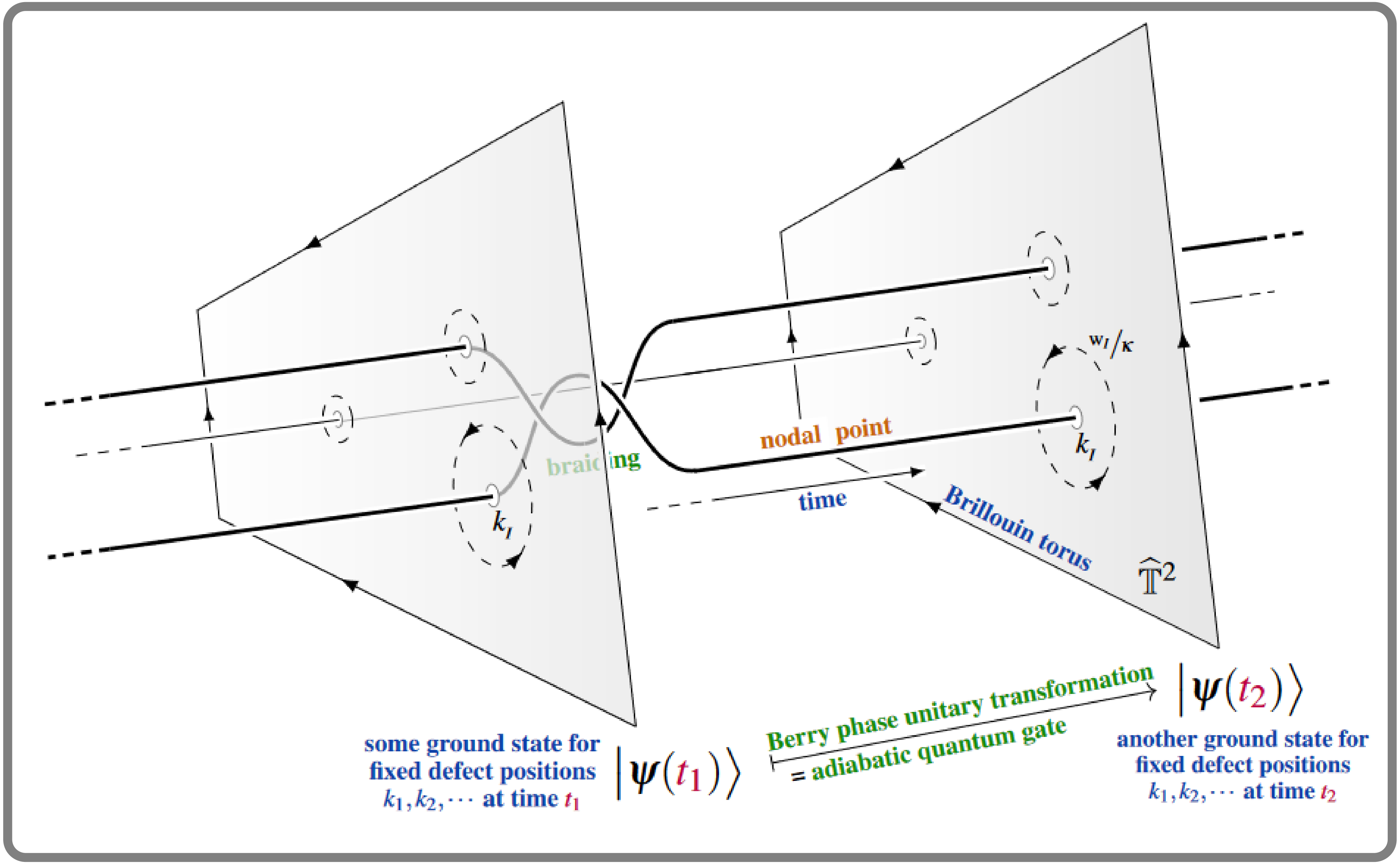
but they take us no closer to having a quantum computer that can solve a problem that anybody cares about.

What is missing is the breakthrough bypassing quantum error correction by using far-more-stable quantum-bits, in an approach called topological quantum computing."

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,



There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

A Modular Functor Which is Universal for Quantum Computation

Michael H. Freedman, Michael Larsen & Zhenghan Wang

Communications in Mathematical Physics **227**, 605–622 (2002) | Cite this article

2 A universal quantum computer

The strictly 2-dimensional part of a TQFT is called a *topological modular functor* (TMF). The most interesting examples of TMFs are given by the SU(2) Witten-Chern-Simons theory at roots of unity [Wi]. These exam-

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

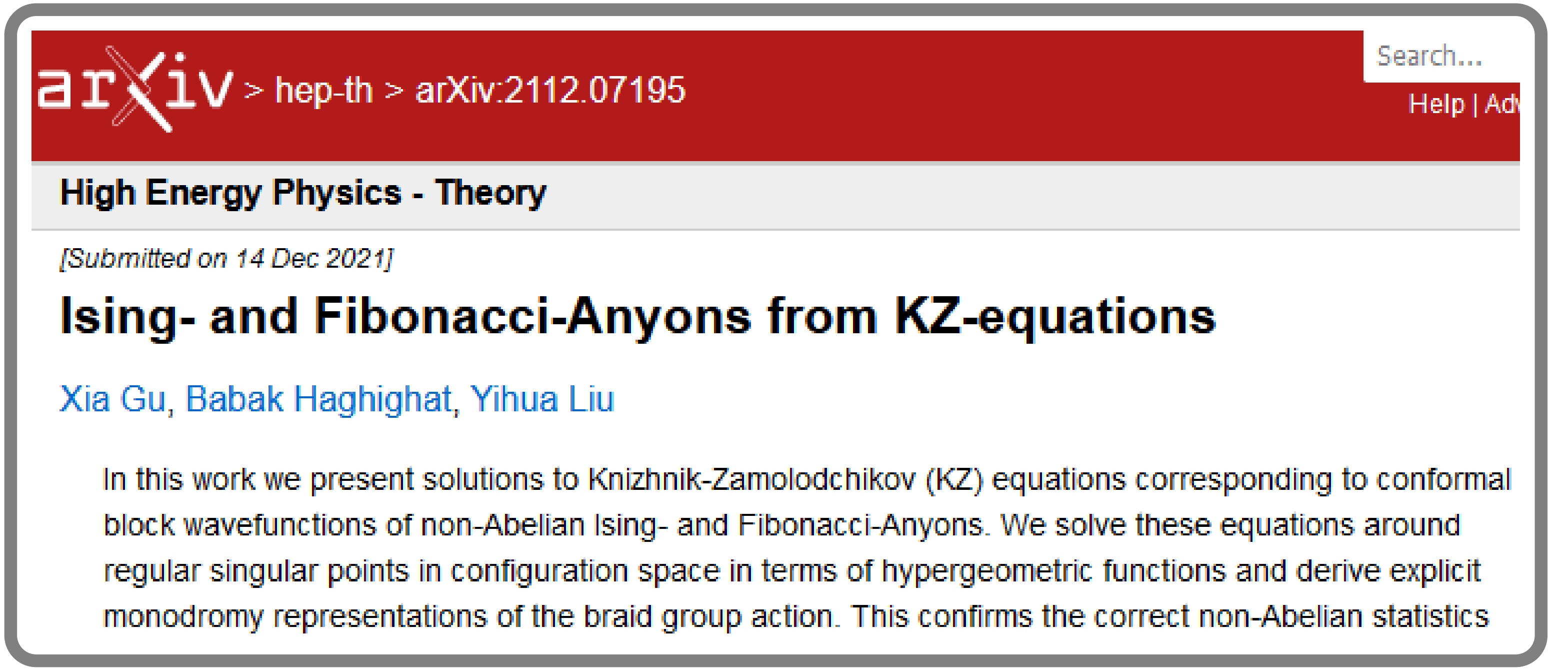
Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the
chiral $\mathfrak{su}(2)$ WZW model CFT.

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the
chiral $\mathfrak{su}(2)$ WZW model CFT.



There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection

Physics of Atomic Nuclei, Vol. 64, No. 12, 2001, pp. 2059–2068. From Yadernaya Fizika, Vol. 64, No. 12, 2001, pp. 2149–2158.

SYMPOSIUM ON QUANTUM GROUPS

Monodromy Representations of the Braid Group*

I. T. Todorov** and **L. K. Hadjiivanov*****

Theoretical Physics Division, Institute for Nuclear Research and Nuclear Energy, Bulgarian Academy of Sciences, Sofia, Bulgaria
Received February 19, 2001

Abstract—Chiral conformal blocks in a rational conformal field theory are a far-going extension of Gauss hypergeometric functions. The associated monodromy representations of Artin's braid group $\mathcal{B}_n$ capture the essence of the modern view on the subject that originates in ideas of Riemann and Schwarz. Physically, such monodromy representations correspond to a new type of braid group statistics which may manifest itself in two-dimensional critical phenomena, e.g., in some exotic quantum Hall states. The associated primary fields satisfy R-matrix exchange relations. The description of the internal symmetry of such fields requires an extension of the concept of a group, thus giving room to quantum groups and their generalizations. We review the appearance of braid group representations in the space of solutions of the Knizhnik–Zamolodchikov equation with an emphasis on the role of a regular basis of solutions which

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the
chiral $\mathfrak{su}(2)$ WZW model CFT.

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

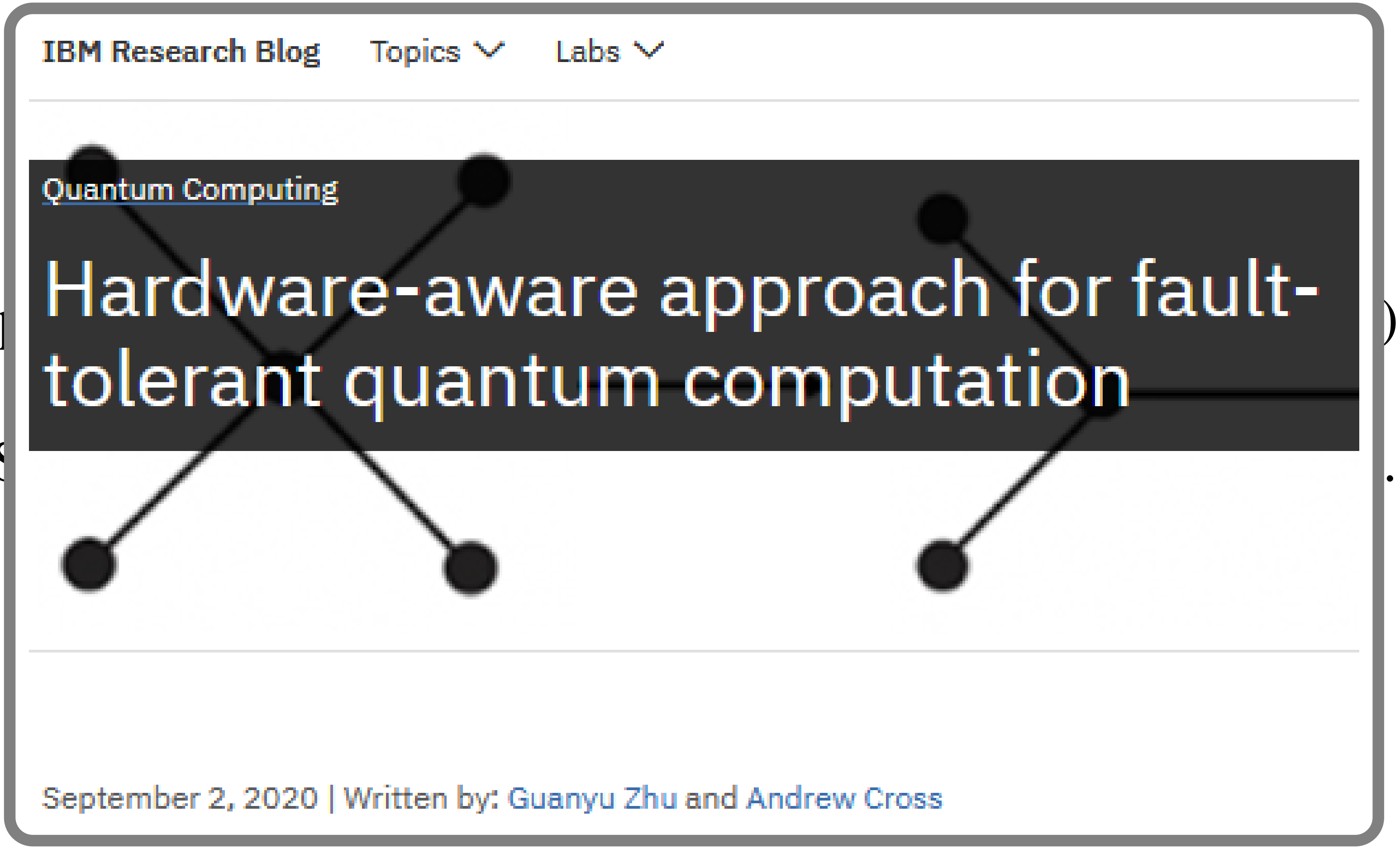
Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the
chiral $\mathfrak{su}(2)$ WZW model CFT.

Efficient programming of topological quantum computers

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the
chiral $\mathfrak{su}(2)$ WZW model CFT.

Efficient programming of topological quantum computers
must be *aware* of this peculiar $\mathfrak{su}(2)$-braid gate hardware.



Efficient programming of topological quantum computers must be *aware* of this peculiar $\mathfrak{su}(2)$-braid gate hardware.

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the
chiral $\mathfrak{su}(2)$ WZW model CFT.

Efficient programming of topological quantum computers
must be *aware* of this peculiar $\mathfrak{su}(2)$-braid gate hardware.

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the
chiral $\mathfrak{su}(2)$ WZW model CFT.

Efficient programming of topological quantum computers
must be *aware* of this peculiar $\mathfrak{su}(2)$-braid gate hardware.

However, bundles of conformal blocks superficially
appear to be a convoluted mathematical structure,

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the
chiral $\mathfrak{su}(2)$ WZW model CFT.

Efficient programming of topological quantum computers
must be *aware* of this peculiar $\mathfrak{su}(2)$-braid gate hardware.

However, bundles of conformal blocks superficially
appear to be a convoluted mathematical structure,
hardly suitable as a foundation for quantum programming.

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the
chiral $\mathfrak{su}(2)$ WZW model CFT.

Efficient programming of topological quantum computers
must be *aware* of this peculiar $\mathfrak{su}(2)$-braid gate hardware.

However, bundles of conformal blocks superficially
appear to be a convoluted mathematical structure,
hardly suitable as a foundation for quantum programming.

We show that the opposite is the case.

There are good arguments that
if Quantum Computation is to be a practical reality
then in the form of Topological Quantum Computation
with quantum gates given by adiabatic braiding of anyons,
specifically of $\mathfrak{su}(2)$-anyons (Ising/Majorana-, Fibonacci-, ...).

Such *braid gates* are rather special among all quantum gates.
Mathematically, they are the monodromy of
the Knizhnik-Zamolodchikov connection
on bundles of conformal blocks of the
chiral $\mathfrak{su}(2)$ WZW model CFT.

Efficient programming of topological quantum computers
must be *aware* of this peculiar $\mathfrak{su}(2)$-braid gate hardware.

However, bundles of conformal blocks superficially
appear to be a convoluted mathematical structure,
hardly suitable as a foundation for quantum programming.

We show that the opposite is the case.

Programming languages suited for describing *bundles* are *dependently typed*

Programming languages suited for describing
bundles are *dependently typed*

system of X-dependent types

$$X \in \mathrm{Types}, \quad x \in X \qquad \vdash \qquad P(x) \in \mathrm{Types}$$

Programming languages suited for describing *bundles* are *dependently typed*

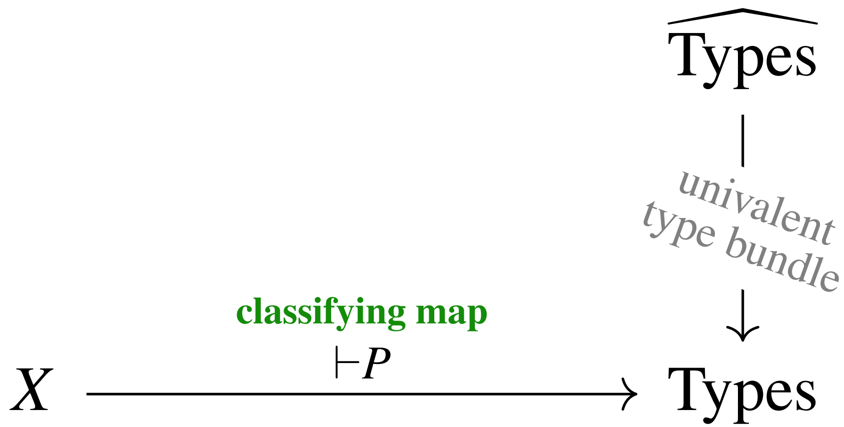
classifying map

$$X \xrightarrow{\vdash P} \text{Types}$$

system of X-dependent types

$$X \in \text{Types}, \quad x \in X \qquad \vdash \qquad P(x) \in \text{Types}$$

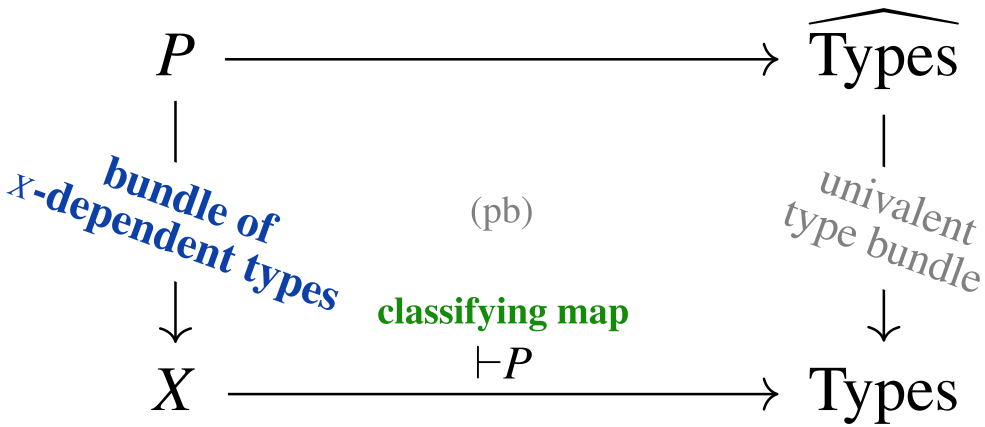
Programming languages suited for describing
bundles are *dependently typed*



system of X-dependent types

$$X \in \text{Types}, \quad x \in X \quad \vdash \quad P(x) \in \text{Types}$$

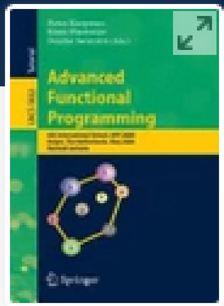
Programming languages suited for describing *bundles* are *dependently typed*



system of X-dependent types

$$X \in \text{Types}, \quad x \in X \qquad \vdash \qquad P(x) \in \text{Types}$$

Programming languages suited for describing *bundles* are *dependently typed*



International School on Advanced Functional Programming

↳ AFP 2008: **Advanced Functional Programming** pp 230–266

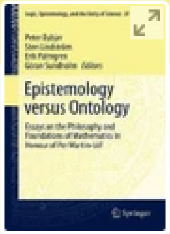
Dependently Typed Programming in Agda

Ulf Norell

Programming languages suited for describing
bundles are *dependently typed*

Programming languages suited for describing
bundles are *dependently typed*
and those which moreover describe
monodromy are *homotopically typed*.

Programming languages suited for describing
bundles are *dependently typed*
and those which moreover describe
monodromy are *homotopically typed*.



Epistemology versus Ontology pp 183–201 | Cite as

Type Theory and Homotopy

Steve Awodey

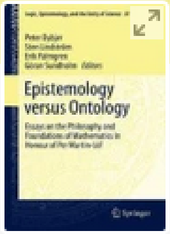
Chapter | First Online: 01 January 2012

Part of the Logic, Epistemology, and the Unity of Science book series (LEUS,volume 27)

Abstract

The purpose of this informal survey article is to introduce the reader to a new and surprising connection between Logic, Geometry, and Algebra which has recently come to light in the form of an interpretation of the constructive type theory of Per Martin-Löf into homotopy theory and higher-dimensional category theory.

In HoTT, data types come with *paths* between their terms



Epistemology versus Ontology pp 183–201 | Cite as

Type Theory and Homotopy

Steve Awodey

Chapter | First Online: 01 January 2012

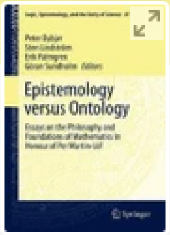
Part of the Logic, Epistemology, and the Unity of Science book series (LEUS,volume 27)

Abstract

The purpose of this informal survey article is to introduce the reader to a new and surprising connection between Logic, Geometry, and Algebra which has recently come to light in the form of an interpretation of the constructive type theory of Per Martin-Löf into homotopy theory and higher-dimensional category theory.

In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \mathrm{Types} \\ x, y \in X \end{array} \;\vdash\; \mathrm{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \mathrm{Types}$$



Epistemology versus Ontology pp 183–201 | Cite as

Type Theory and Homotopy

Steve Awodey

Chapter | First Online: 01 January 2012

Part of the Logic, Epistemology, and the Unity of Science book series (LEUS,volume 27)

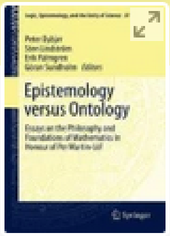
Abstract

The purpose of this informal survey article is to introduce the reader to a new and surprising connection between Logic, Geometry, and Algebra which has recently come to light in the form of an interpretation of the constructive type theory of Per Martin-Löf into homotopy theory and higher-dimensional category theory.

In HoTT, data types come with paths between their terms

$$\begin{array}{l} X \in \mathrm{Types} \\ x, y \in X \end{array} \;\vdash\; \mathrm{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \mathrm{Types}$$

akin to continuous paths in topological spaces.



Epistemology versus Ontology pp 183–201 | Cite as

Type Theory and Homotopy

Steve Awodey

Chapter | First Online: 01 January 2012

Part of the Logic, Epistemology, and the Unity of Science book series (LEUS,volume 27)

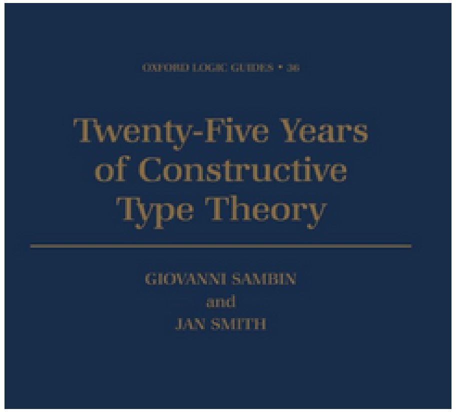
Abstract

The purpose of this informal survey article is to introduce the reader to a new and surprising connection between Logic, Geometry, and Algebra which has recently come to light in the form of an interpretation of the constructive type theory of Per Martin-Löf into homotopy theory and higher-dimensional category theory.

In HoTT, data types come with paths between their terms

$$\begin{array}{rcl} X & \in & \mathrm{Types} \\ x, y & \in & X \end{array} \quad \vdash \quad \mathrm{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \mathrm{Types}$$

akin to continuous paths in topological spaces.



CHAPTER

6 The groupoid interpretation of type theory

Get access ›

Martin Hofmann, Thomas Streicher

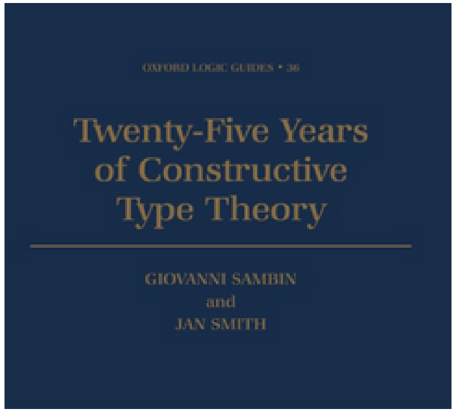
https://doi.org/10.1093/oso/9780198501275.003.0008 Pages 83–112

Published: October 1998

In HoTT, data types come with paths between their terms

$$\begin{array}{rcl} X & \in & \text{Types} \\ x, y & \in & X \end{array} \quad \vdash \quad \text{Paths}_X(x, y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.



CHAPTER

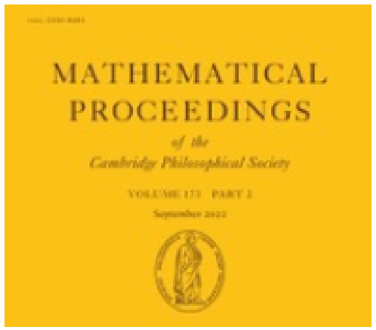
6 The groupoid interpretation of type theory

Get access >

Martin Hofmann, Thomas Streicher

https://doi.org/10.1093/oso/9780198501275.003.0008 Pages 83–112

Published: October 1998



Homotopy theoretic models of identity types

Published online by Cambridge University Press: **01 January 2009**

STEVE AWODEY and MICHAEL A. WARREN

In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \text{Types} \\ x, y \in X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \text{Types} \\ x, y \in X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

E.g.: if G is a finitely presented group, then we get a type $\mathbf{B}G$ with essentially unique $* \in \mathbf{B}G$ s.t. $\text{Paths}_{\mathbf{B}G}(*,*) \simeq G$.

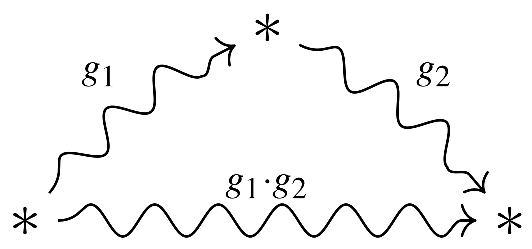
In <u>HoTT</u>, data types come with paths between their terms

$$\begin{array}{rl} X & \in \text{Types} \\ x, y & \in X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

E.g.: if G is a finitely presented group, then we get a type $\mathbf{B}G$ with essentially unique $* \in \mathbf{B}G$ s.t. $\text{Paths}_{\mathbf{B}G}(*,*) \simeq G$.

$\mathbf{B}G = \Bigg\{$



$\Bigg| \; g_i \in G \Bigg\}$

In HoTT, data types come with paths between their terms

$$\begin{array}{r l} X & \in \text{Types} \\ x, y & \in X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

E.g.: if G is a finitely presented group, then we get a type $\mathbf{B}G$ with essentially unique $* \in \mathbf{B}G$ s.t. $\text{Paths}_{\mathbf{B}G}(*,*) \simeq G$.

In HoTT, data types come with paths between their terms

$$\begin{array}{rcl} X & \in & \mathrm{Types} \\ x, y & \in & X \end{array} \quad \vdash \quad \mathrm{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \mathrm{Types}$$

akin to continuous paths in topological spaces.

E.g.: if G is a finitely presented group, then we get a type $\mathbf{B}G$ with essentially unique $* \in \mathbf{B}G$ s.t. $\mathrm{Paths}_{\mathbf{B}G}(*,*) \simeq G$.

For $G = \mathrm{Br}(n)$ an Artin braid group this is the homotopy type of configurations of points: $\mathbf{B}\mathrm{Br}(n) \simeq \int \mathrm{Conf}_n(\mathbb{C})$.

In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \mathrm{Types} \\ x, y \in X \end{array} \quad \vdash \quad \mathrm{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \mathrm{Types}$$

akin to continuous paths in topological spaces.

E.g.: if G is a finitely presented group, then we get a type $\mathbf{B}G$ with essentially unique $* \in \mathbf{B}G$ s.t. $\mathrm{Paths}_{\mathbf{B}G}(*,*) \simeq G$.
For $G = \mathrm{Br}(n)$ an Artin braid group this is the homotopy type of configurations of points: $\mathbf{B}\mathrm{Br}(n) \simeq \int \mathrm{Conf}_n(\mathbb{C})$.

$$\mathbf{B}\mathrm{Br}(3) = \Big\{$$  $$\Big\}$$

In HoTT, data types come with paths between their terms

$$\begin{array}{rcl} X & \in & \text{Types} \\ x, y & \in & X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

E.g.: if G is a finitely presented group, then we get a type $\mathbf{B}G$ with essentially unique $* \in \mathbf{B}G$ s.t. $\text{Paths}_{\mathbf{B}G}(*,*) \simeq G$. For $G = \text{Br}(n)$ an Artin braid group this is the homotopy type of configurations of points: $\mathbf{B}\text{Br}(n) \simeq \int \text{Conf}_n(\mathbb{C})$.

In HoTT, data types come with paths between their terms

$$\begin{array}{rl} X & \in \text{Types} \\ x, y & \in X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

E.g.: if G is a finitely presented group, then we get a type $\mathbf{B}G$ with essentially unique $* \in \mathbf{B}G$ s.t. $\text{Paths}_{\mathbf{B}G}(*,*) \simeq G$.
For $G = \text{Br}(n)$ an Artin braid group this is the homotopy type of configurations of points: $\mathbf{B}\text{Br}(n) \simeq \int \text{Conf}_n(\mathbb{C})$.

An X-dependent type family $\boxed{x \in X \ \vdash \ P(x) \in \text{Types}}$ inherits *transport* (monodromy!) along base paths:

In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \mathrm{Types} \\ x, y \in X \end{array} \;\vdash\; \mathrm{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \mathrm{Types}$$

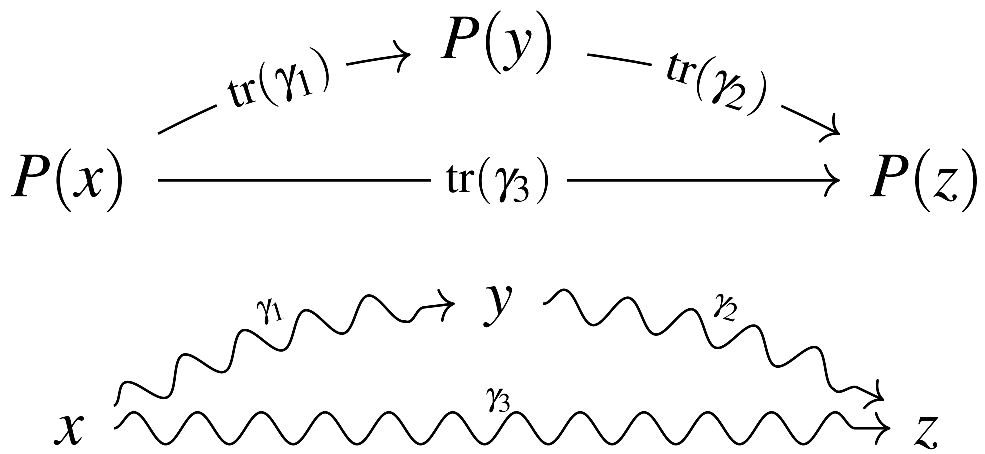
akin to continuous paths in topological spaces.

E.g.: if G is a finitely presented group, then we get a type $\mathbf{B}G$ with essentially unique $* \in \mathbf{B}G$ s.t. $\mathrm{Paths}_{\mathbf{B}G}(*,*) \simeq G$. For $G = \mathrm{Br}(n)$ an Artin braid group this is the homotopy type of configurations of points: $\mathbf{B}\mathrm{Br}(n) \simeq \int \mathrm{Conf}_n(\mathbb{C})$.

An X-dependent type family $\boxed{x \in X \;\vdash\; P(x) \in \mathrm{Types}}$ inherits *transport* (monodromy!) along base paths:

$P : X \longrightarrow \mathrm{Types}$

$X : \mathrm{Types}$



In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \text{Types} \\ x, y \in X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

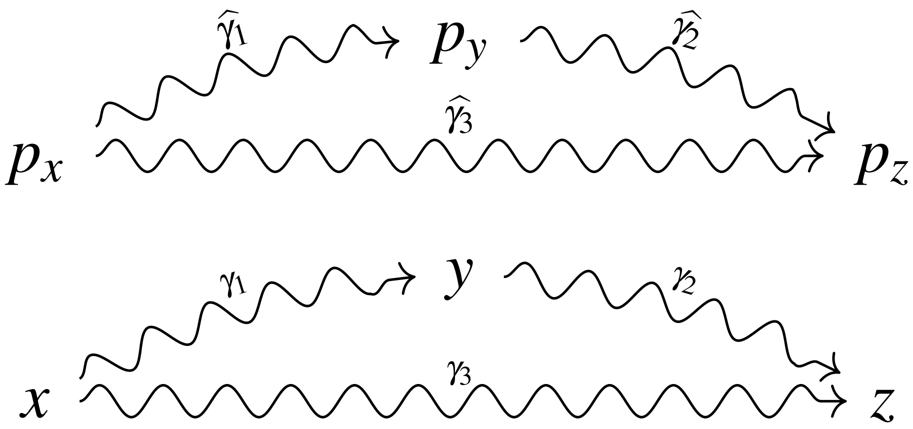
E.g.: if G is a finitely presented group, then we get a type $\mathbf{B}G$ with essentially unique $* \in \mathbf{B}G$ s.t. $\text{Paths}_{\mathbf{B}G}(*,*) \simeq G$.
For $G = \text{Br}(n)$ an Artin braid group this is the homotopy type of configurations of points: $\mathbf{B}\text{Br}(n) \simeq \int \text{Conf}_n(\mathbb{C})$.

An X-dependent type family $\boxed{x \in X \;\vdash\; P(x) \in \text{Types}}$

and compatible *path lifting*:

$P : X \to \text{Types}$

$X : \text{Types}$



In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \mathrm{Types} \\ x, y \in X \end{array} \quad \vdash \quad \mathrm{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \mathrm{Types}$$

akin to continuous paths in topological spaces.

In HoTT, data types come with paths between their terms

$$\boxed{\begin{array}{rcl} X & \in & \text{Types} \\ x, y & \in & X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

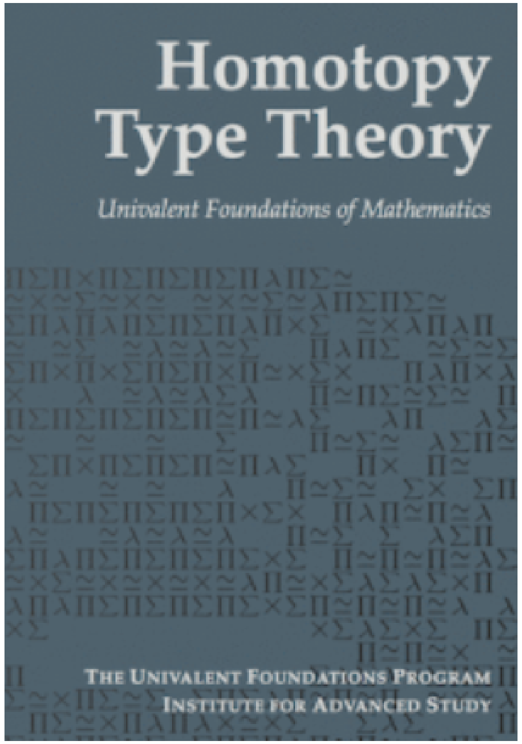
In HoTT, data types come with paths between their terms

$$X \in \text{Types},\; x, y \in X \;\vdash\; \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

Homotopy type theory is a new branch of mathematics that combines aspects of several different fields in a surprising way. It is based on a recently discovered connection between homotopy theory and type theory. It touches on topics as seemingly distant as the homotopy groups of spheres, the algorithms for type checking, and the definition of weak ∞-groupoids. Homotopy type theory offers a new "univalent" foundation of mathematics, in which a central role is played by Voevodsky's univalence axiom and higher inductive types. The present book is intended as a first systematic exposition of the basics of univalent foundations, and a collection of examples of this new style of reasoning — but without requiring the reader to know or learn any formal logic, or to use any computer proof assistant. We believe that univalent foundations will eventually become a viable alternative to set theory as the "implicit foundation"



In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \text{Types} \\ x, y \in X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

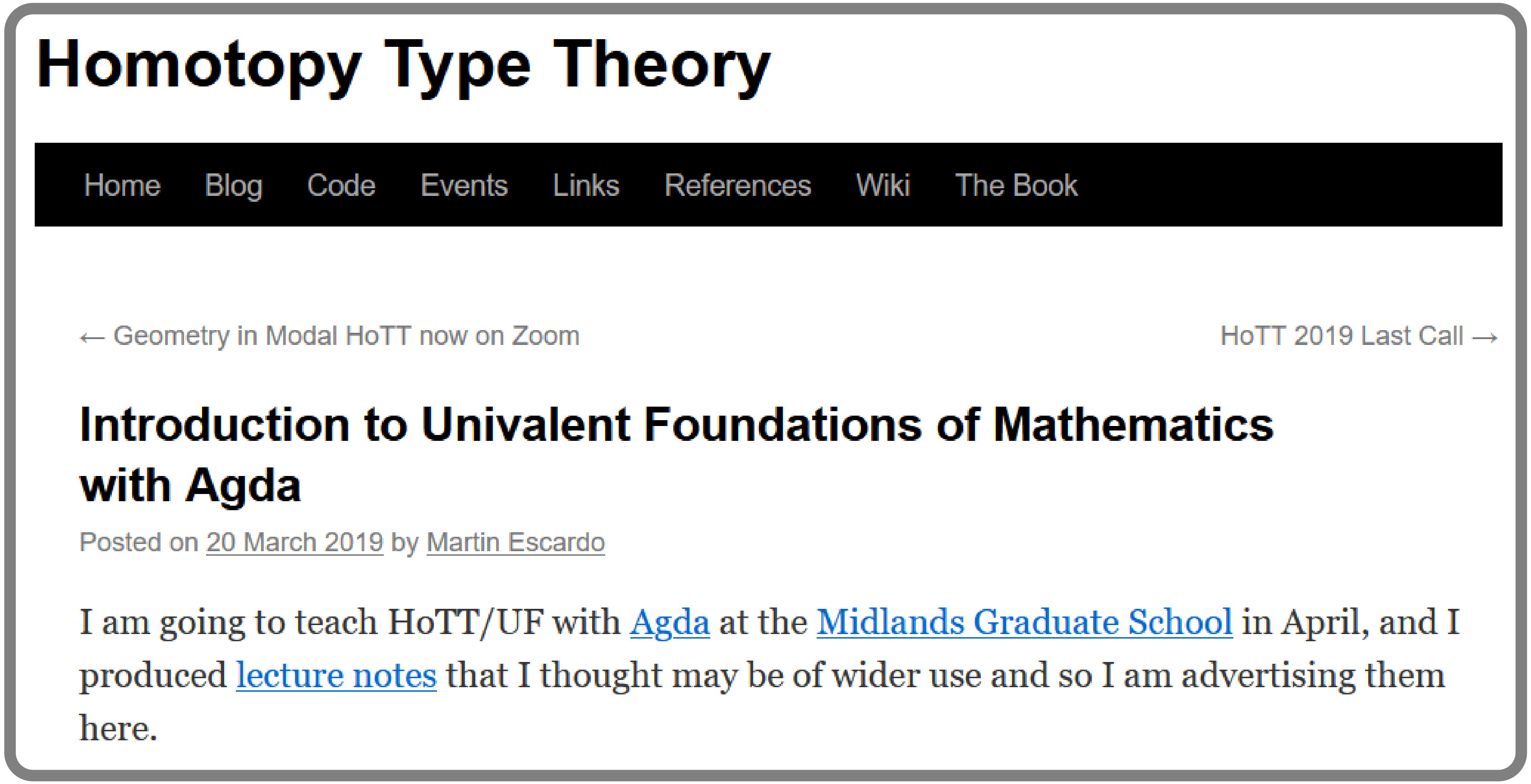
Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

In HoTT, data types come with paths between their terms

$$\begin{array}{l} X \in \mathrm{Types} \\ x, y \in X \end{array} \;\vdash\; \mathrm{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \mathrm{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.



Homotopy Type Theory

Home Blog Code Events Links References Wiki The Book

← Geometry in Modal HoTT now on Zoom HoTT 2019 Last Call →

Introduction to Univalent Foundations of Mathematics with Agda

Posted on 20 March 2019 by Martin Escardo

I am going to teach HoTT/UF with Agda at the Midlands Graduate School in April, and I produced lecture notes that I thought may be of wider use and so I am advertising them here.

In HoTT, data types come with paths between their terms

$$\begin{array}{rl} X & \in \text{Types} \\ x, y & \in X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

In HoTT, data types come with paths between their terms

$$\begin{array}{r}X \in \text{Types} \\ x, y \in X\end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \text{Types} \\ x, y \in X \end{array} \vdash \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

(1.) Reversible circuit execution – such as in quantum computation – is described by *path lifting* in dependent homotopy type families.

In HoTT, data types come with paths between their terms

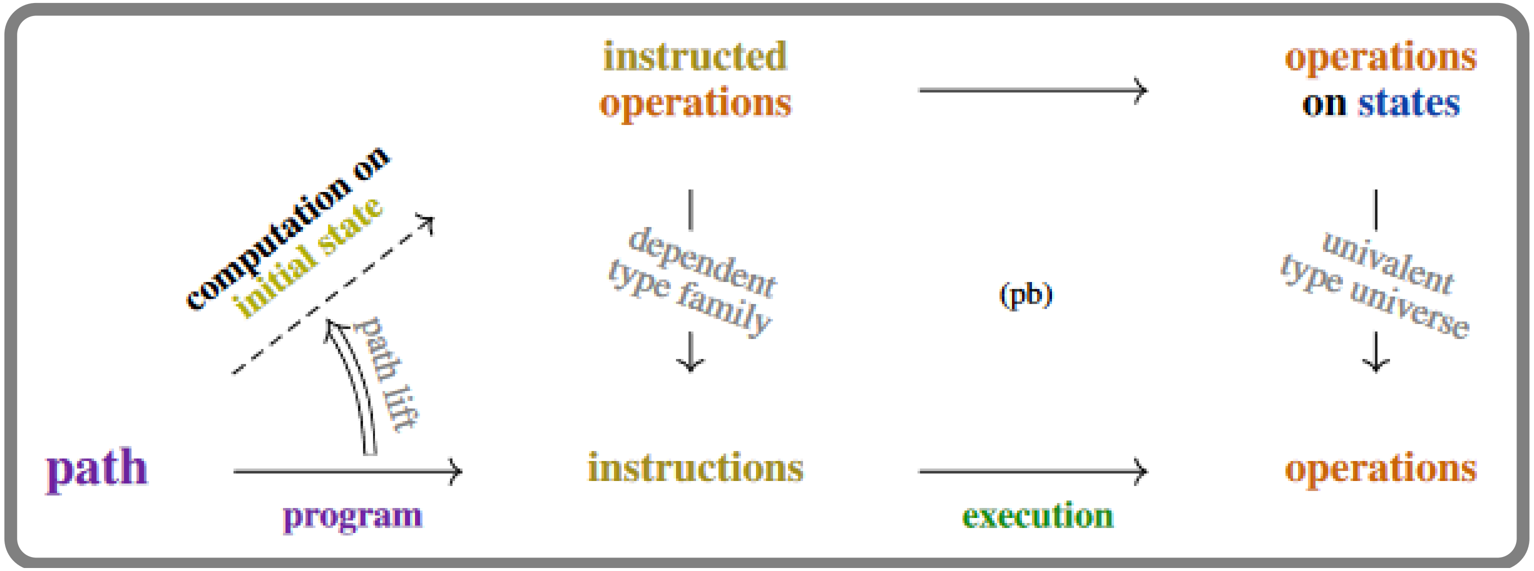
$$\begin{array}{rcl} X & \in & \text{Types} \\ x, y & \in & X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

(1.) Reversible circuit execution – such as in quantum computation – is described by *path lifting* in dependent homotopy type families.



In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \text{Types} \\ x, y \in X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

(1.) Reversible circuit execution – such as in quantum computation – is described by *path lifting* in dependent homotopy type families.

In HoTT, data types come with paths between their terms

$$\begin{array}{rcl} X & \in & \text{Types} \\ x, y & \in & X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

(1.) Reversible circuit execution – such as in quantum computation – is described by *path lifting* in dependent homotopy type families.

(2.) The dependent homotopy type family of $\mathfrak{su}(2)$-conformal blocks has a slick construction in HoTT programming languages;

In HoTT, data types come with paths between their terms

$$X \in \text{Types},\; x, y \in X \;\vdash\; \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

(1.) Reversible circuit execution – such as in quantum computation – is described by *path lifting* in dependent homotopy type families.

(2.) The dependent homotopy type family of $\mathfrak{su}(2)$-conformal blocks has a slick construction in HoTT programming languages;

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \;\vdash\; \left[\prod_{t : B\mathbb{Z}_\kappa} \left(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}} \left(\mathbb{C} \setminus \{z_I\}_{I=1}^N\right)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \right) \right]_0$

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types finitely presented by the Artin braid relations as in (32).)

In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \mathrm{Types} \\ x, y \in X \end{array} \quad \vdash \quad \mathrm{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \mathrm{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

(1.) Reversible circuit execution – such as in quantum computation – is described by *path lifting* in dependent homotopy type families.

(2.) The dependent homotopy type family of $\mathfrak{su}(2)$-conformal blocks has a slick construction in HoTT programming languages;

In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \text{Types} \\ x, y \in X \end{array} \;\vdash\; \mathrm{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

(1.) Reversible circuit execution – such as in quantum computation – is described by *path lifting* in dependent homotopy type families.

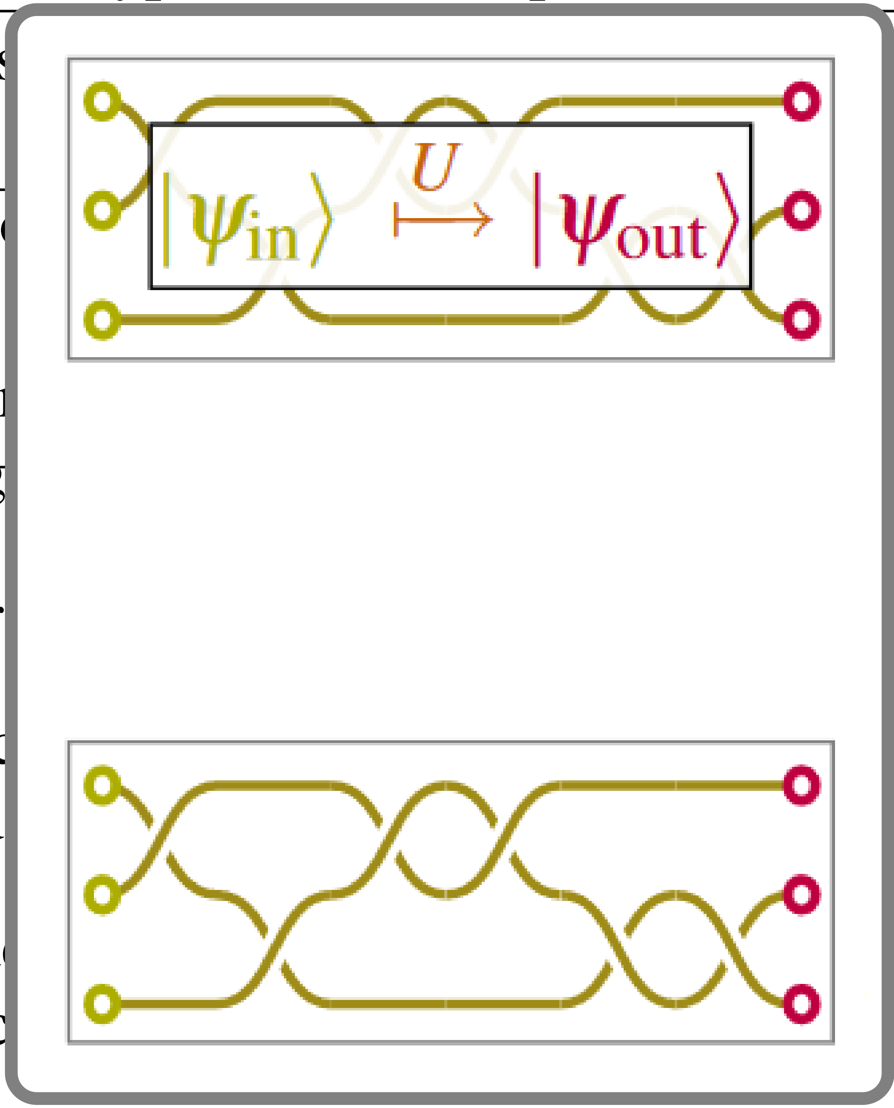
(2.) The dependent homotopy type family of $\mathfrak{su}(2)$-conformal blocks has a slick construction in HoTT programming languages;

(3.) its path lifting operation is exactly anyonic braid gate execution.

In HoTT, data types come with paths between their terms

$X \in$ Type… $x, y \in X$ … $y\} \in$ Types

akin t… …paces.



Such HoTT p… …be remarkably fundamental, arg… …for mathematics.

We now first offer…

(1.) Reversible c… …um computation – is described… …topy type families.

(2.) The depend… …-conformal blocks has a slick c… …languages;

(3.) its path lifting operation is exactly anyonic braid gate execution.

In HoTT, data types come with paths between their terms

$$X \in \text{Types},\; x, y \in X \;\vdash\; \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

(1.) Reversible circuit execution – such as in quantum computation – is described by *path lifting* in dependent homotopy type families.

(2.) The dependent homotopy type family of $\mathfrak{su}(2)$-conformal blocks has a slick construction in HoTT programming languages;

(3.) its path lifting operation is exactly anyonic braid gate execution.

In HoTT, data types come with paths between their terms

$$\begin{array}{r} X \in \text{Types} \\ x, y \in X \end{array} \;\vdash\; \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

(1.) Reversible circuit execution – such as in quantum computation – is described by *path lifting* in dependent homotopy type families.

(2.) The dependent homotopy type family of $\mathfrak{su}(2)$-conformal blocks has a slick construction in HoTT programming languages;

(3.) its path lifting operation is exactly anyonic braid gate execution.

$\Rightarrow$ natural & powerful topological-hardware-aware Q-pogramming paradigm

In HoTT, data types come with paths between their terms

$$\begin{array}{rl} X & \in \text{Types} \\ x, y & \in X \end{array} \quad \vdash \quad \text{Paths}_X(x,y) = \left\{ x \overset{\gamma}{\rightsquigarrow} y \right\} \in \text{Types}$$

akin to continuous paths in topological spaces.

Such HoTT programming languages turn out to be remarkably fundamental, arguably serving as a new foundation for mathematics.

We now first offer the following observations:

(1.) Reversible circuit execution – such as in quantum computation – is described by *path lifting* in dependent homotopy type families.

(2.) The dependent homotopy type family of $\mathfrak{su}(2)$-conformal blocks has a slick construction in HoTT programming languages;

(3.) its path lifting operation is exactly anyonic braid gate execution.

$\Rightarrow$ natural & powerful topological-hardware-aware Q-pogramming paradigm

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.

International Journal of Modern Physics B | Vol. 04, No. 05, pp. 1049-1057 (1990)

HYPERGEOMETRIC-TYPE INTEGRALS AND THE $\mathfrak{sl}(2, \mathbb{C})$ KNIZHNIK-ZAMOLODCHIKOV EQUATION

Etsuro DATE, Michio JIMBO, Atsushi MATSUO and Tetsuji MIWA

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.

International Journal of Modern Physics B | Vol. 04, No. 05, pp. 1049-1057 (1990)

HYPERGEOMETRIC-TYPE INTEGRALS AND THE $\mathfrak{sl}(2,\mathbb{C})$ KNIZHNIK-ZAMOLODCHIKOV EQUATION

Etsuro DATE, Michio JIMBO, Atsushi MATSUO and Tetsuji MIWA

On algebraic equations satisfied by hypergeometric correlators in WZW models. I

Boris Feigin, Vadim Schechtman & Alexander Varchenko

Communications in Mathematical Physics **163**, 173–184 (1994) | Cite this article

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.

International Journal of Modern Physics B | Vol. 04, No. 05, pp. 1049-1057 (1990)

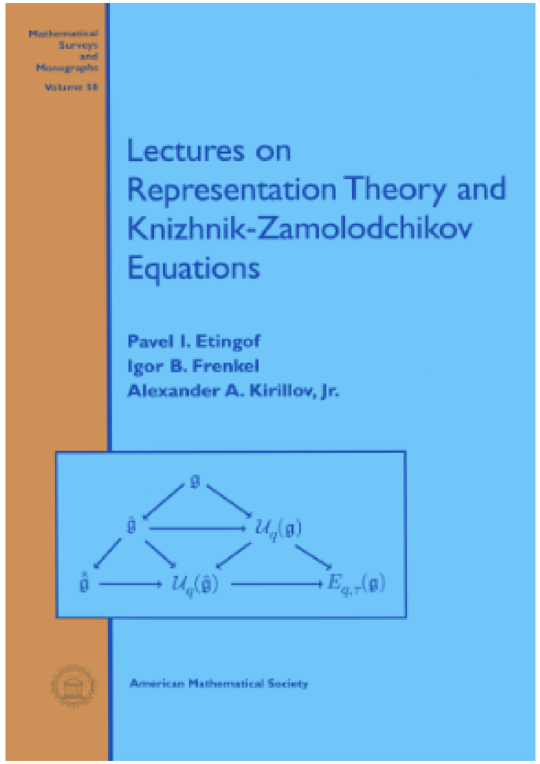
HYPERGEOMETRIC-TYPE INTEGRALS AND THE $\mathfrak{sl}(2, \mathbb{C})$ KNIZHNIK-ZAMOLODCHIKOV EQUATION

Etsuro DATE, Michio JIMBO, Atsushi MATSUO and Tetsuji MIWA

On algebraic equations satisfied by hypergeometric correlators in WZW models. I

Boris Feigin, Vadim Schechtman & Alexander Varchenko

Communications in Mathematical Physics **163**, 173–184 (1994) | Cite this article



Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad \vdash \quad \left[\prod_{t:B\mathbb{Z}_\kappa} \left(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}} \left(\mathbb{C} \setminus \{z_I\}_{I=1}^N\right)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \right) \right]_0$

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types finitely presented by the Artin braid relations as in (32).)

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad \vdash \quad \Big[\prod_{t : B\mathbb{Z}_\kappa} \Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}} (\mathbb{C} \setminus \{z_I\}_{I=1}^N)(\tau) \longrightarrow \underbrace{K(\mathbb{C}, n)}(\tau) \Big) \Big]_0$

classifying type for complex cohomology

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types fini
braid relations as in (32).)

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \;\vdash\; \left[\prod_{t:B\mathbb{Z}_\kappa} \left(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}}\left(\mathbb{C}\setminus\{z_I\}_{I=1}^N\right)(\tau) \longrightarrow \underbrace{K(\mathbb{C},n)(\tau)}_{\text{classifying type for complex cohomology}}\right)\right]_0$

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types fini
braid relations as in (32).)

Eilenberg-MacLane spaces in homotopy type theory

Authors: Daniel R. Licata, Eric Finster Authors Info & Claims

CSL-LICS '14: Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) • July 2014 • Article No.: 66 • Pages 1–9 • https://doi.org/10.1145/2603088.2603153

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \; : \; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad \vdash \quad \Big[\prod_{t:B\mathbb{Z}_\kappa} \Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}}\big(\mathbb{C}\setminus\{z_I\}_{I=1}^N\big)(\tau) \longrightarrow \underbrace{K(\mathbb{C},n)}(\tau) \Big) \Big]_0$	

classifying type for complex cohomology

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types fini
braid relations as in (32).)

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \;\;\vdash\;\; \Big[\prod_{t:B\mathbb{Z}_\kappa} \overbrace{\Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}}\big(\mathbb{C}\setminus\{z_I\}_{I=1}^N\big)(\tau) \longrightarrow \underbrace{K(\mathbb{C},n)}(\tau) \Big)}^{\text{fiberwise function type}} \Big]_0$

classifying type for complex cohomology

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types fini
braid relations as in (32).)

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \;\;\vdash\;\; \Big[\prod_{t:B\mathbb{Z}_\kappa} \Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}} \big(\mathbb{C}\setminus\{z_I\}_{I=1}^N\big)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \Big) \Big]_0$

fiberwise function type

dependent product over twist variable

classifying type for complex cohomology

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing ... types fini
braid relations as in (32).)

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \;\vdash\; \Big[\prod_{t:B\mathbb{Z}_\kappa} \Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}}\big(\mathbb{C}\setminus\{z_I\}_{I=1}^N\big)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \Big) \Big]_0$

fiberwise function type

fiberwise homotopy 0-truncation

dependent product over twist variable

classifying type for complex cohomology

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing · types fini
braid relations as in (32).)

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \;\vdash\; \Big[\prod_{t : B\mathbb{Z}_\kappa} \Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}}\big(\mathbb{C}\setminus \{z_I\}_{I=1}^N\big)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \Big) \Big]_0$

fiberwise function type

fiberwise homotopy 0-truncation

dependent product over twist variable

classifying type for complex cohomology

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing … types fini
braid relations as in (32).)

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad \vdash \quad \left[\prod_{t:B\mathbb{Z}_\kappa} \left(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}} \left(\mathbb{C} \setminus \{z_I\}_{I=1}^N\right)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \right) \right]_0$

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types finitely presented by the Artin braid relations as in (32).)

This $\left\langle \begin{matrix} \text{works because} \\ \text{and uses that} \end{matrix} \right\rangle$ HoTT has categorical semantics
in Parameterized Homotopy Theory.

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad \vdash \quad \Big[\prod_{t:B\mathbb{Z}_\kappa} \Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}} (\mathbb{C} \setminus \{z_I\}_{I=1}^N)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \Big) \Big]_0$

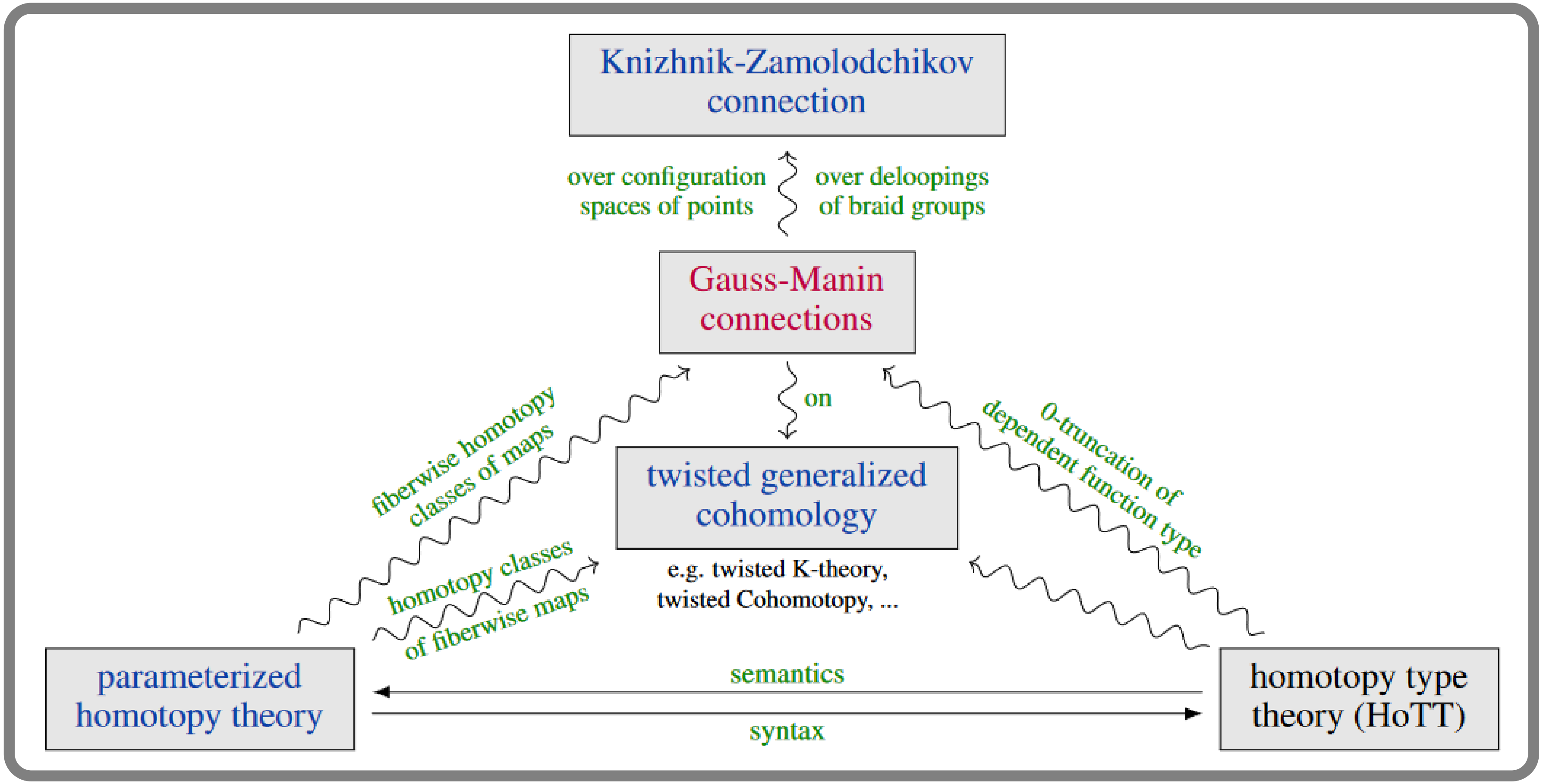
(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types finitely presented by the Artin braid relations as in (32).)

This $\left\langle \begin{array}{c}\text{works because}\\ \text{and uses that}\end{array} \right\rangle$ HoTT has categorical semantics
in Parameterized Homotopy Theory.

Emily Riehl, *On the ∞-topos semantics of homotopy type theory*, lecture at *Logic and higher structures* CIRM (Feb. 2022) [pdf, pdf]

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad \vdash \quad \left[\prod_{t:B\mathbb{Z}_\kappa} \Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}} \big(\mathbb{C} \setminus \{z_I\}_{I=1}^N\big)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \Big) \right]_0$



Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad \vdash \quad \Big[\prod_{t:B\mathbb{Z}_\kappa} \Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}}(\mathbb{C} \setminus \{z_I\}_{I=1}^N)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \Big) \Big]_0$

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types finitely presented by the Artin braid relations as in (32).)

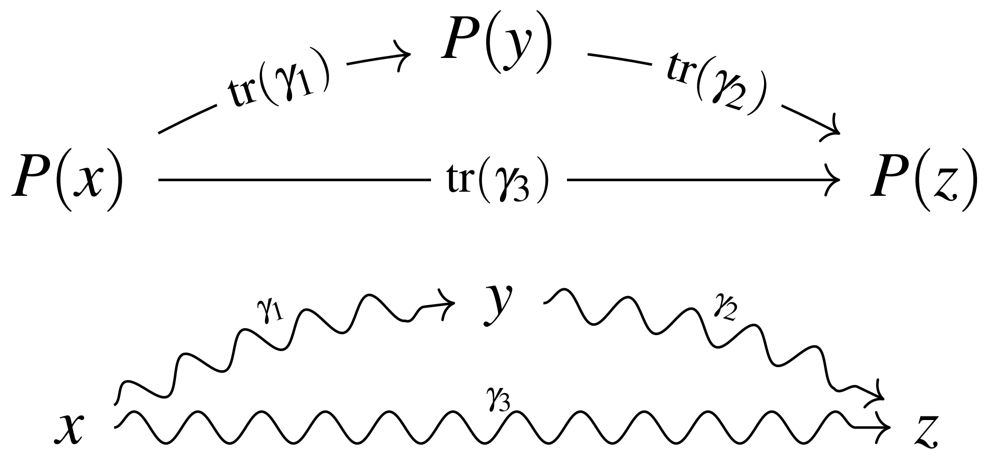
Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad \vdash \quad \Big[\prod_{t:B\mathbb{Z}_\kappa} \Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}} \big(\mathbb{C} \setminus \{z_I\}_{I=1}^N\big)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \Big) \Big]_0$

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types finitely presented by the Artin braid relations as in (32).)

Claim: Its transport operation is the monodromy braid representation

$P : \; X \longrightarrow \text{Types}$



$X : \; \text{Types}$

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

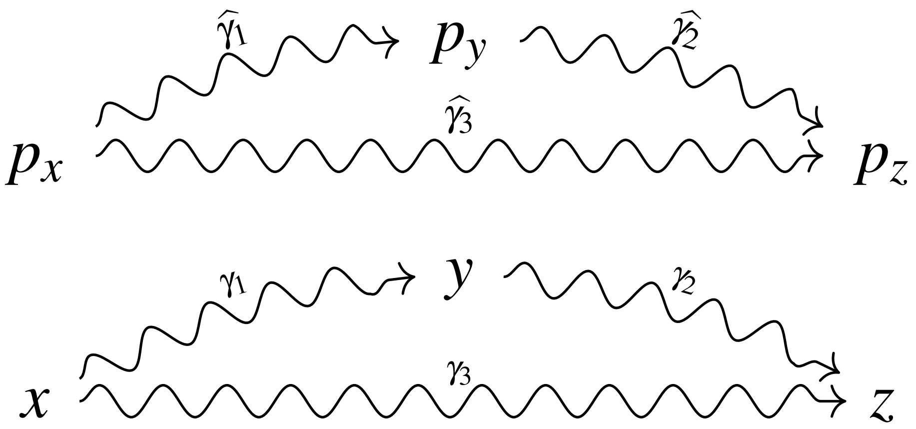
KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int_{\{1,\cdots,N\}} \mathrm{Conf}(\mathbb{C}) \quad \vdash \quad \left[\prod_{t:B\mathbb{Z}_\kappa} \left(\int_{\{1,\cdots,n\}} \mathrm{Conf}(\mathbb{C}\setminus\{z_I\}_{I=1}^N)(\tau) \longrightarrow K(\mathbb{C},n)(\tau)\right)\right]_0$

(Recall here that $\int_{\{1,\cdots,N\}} \mathrm{Conf}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types finitely presented by the Artin braid relations as in (32).)

Claim: Its transport operation is the monodromy braid representation
and its path lifting is execution of $\mathfrak{su}(2)$-anyon braid gates.

$P : X \to \mathrm{Types}$

$X : \mathrm{Types}$



Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad \vdash \quad \Big[\prod_{t:B\mathbb{Z}_\kappa} \Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}} \big(\mathbb{C} \setminus \{z_I\}_{I=1}^N\big)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \Big) \Big]_0$

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types finitely presented by the Artin braid relations as in (32).)

Claim: Its transport operation is the monodromy braid representation
and its path lifting is execution of $\mathfrak{su}(2)$-anyon braid gates.

Hence coding this type family into a HoTT language like `Agda`
gives a Topological Quantum Programming Language

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad\vdash\quad \left[\prod_{t:B\mathbb{Z}_\kappa}\Big(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}}\big(\mathbb{C}\setminus\{z_I\}_{I=1}^N\big)(\tau) \longrightarrow K(\mathbb{C},n)(\tau)\Big)\right]_0$

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types finitely presented by the Artin braid relations as in (32).)

Claim: Its transport operation is the monodromy braid representation
and its path lifting is execution of $\mathfrak{su}(2)$-anyon braid gates.

Hence coding this type family into a HoTT language like `Agda`
gives a Topological Quantum Programming Language
which is fully aware of topological anyon braid quantum gates.

Namely, bundles of $\mathfrak{su}(2)$-conformal blocks
secretly happen to have a purely *cohomological* definition.
We show how to construct this as a dependent type family in HoTT:

KZ-connection on $\widehat{\mathfrak{su}_2}^{\kappa-2}$-conformal blocks	(31)	$(z_I)_{I=1}^N \;:\; \int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C}) \quad \vdash \quad \left[\prod_{t:B\mathbb{Z}_\kappa} \left(\int \underset{\{1,\cdots,n\}}{\mathrm{Conf}} (\mathbb{C} \setminus \{z_I\}_{I=1}^N)(\tau) \longrightarrow K(\mathbb{C},n)(\tau) \right) \right]_0$

(Recall here that $\int \underset{\{1,\cdots,N\}}{\mathrm{Conf}}(\mathbb{C})$ etc. may be regarded as nothing but suggestive notation for types finitely presented by the Artin braid relations as in (32).)

Claim: Its transport operation is the monodromy braid representation
and its path lifting is execution of $\mathfrak{su}(2)$-anyon braid gates.

Hence coding this type family into a HoTT language like `Agda`
gives a Topological Quantum Programming Language
which is fully aware of topological anyon braid quantum gates.

To compute is

the case of
Topological Quantum Computation
[Sati & Schreiber, PlanQC **2022** 33 (2022)]

To compute is to **execute**

the case of
Topological Quantum Computation
[Sati & Schreiber, PlanQC **2022** 33 (2022)]

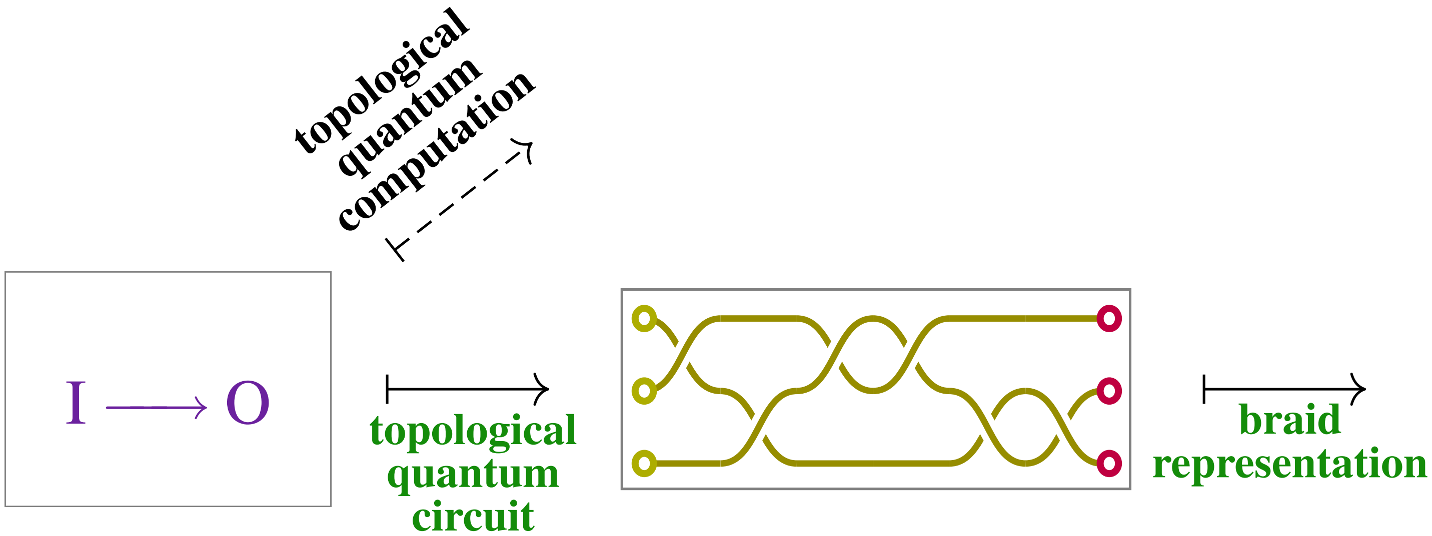
topological quantum computation

$\mathrm{I} \longrightarrow \mathrm{O}$

braid representation

To compute is to **execute** sequences of **instructions**

the case of
Topological Quantum Computation
[Sati & Schreiber, PlanQC **2022** 33 (2022)]

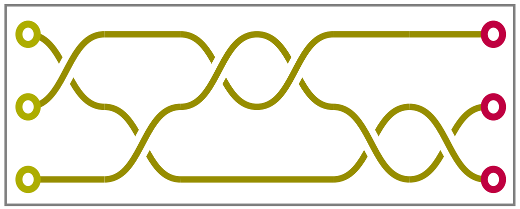


To compute is to **execute**
sequences of **instructions**
as composable **operations**

the case of
Topological Quantum Computation
[Sati & Schreiber, PlanQC **2022** 33 (2022)]

topological quantum computation

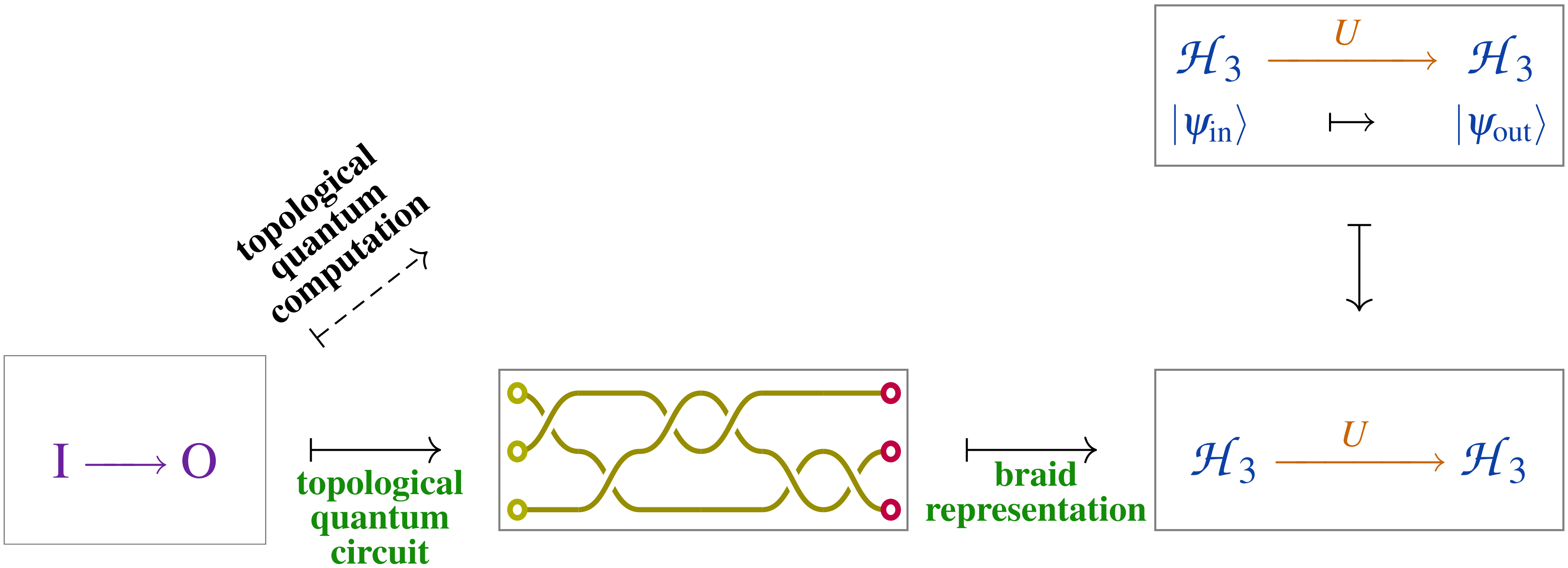
$\mathrm{I} \longrightarrow \mathrm{O}$ $\longmapsto$ **topological quantum circuit**



$\longmapsto$ **braid representation** $\mathcal{H}_3 \xrightarrow{\;U\;} \mathcal{H}_3$

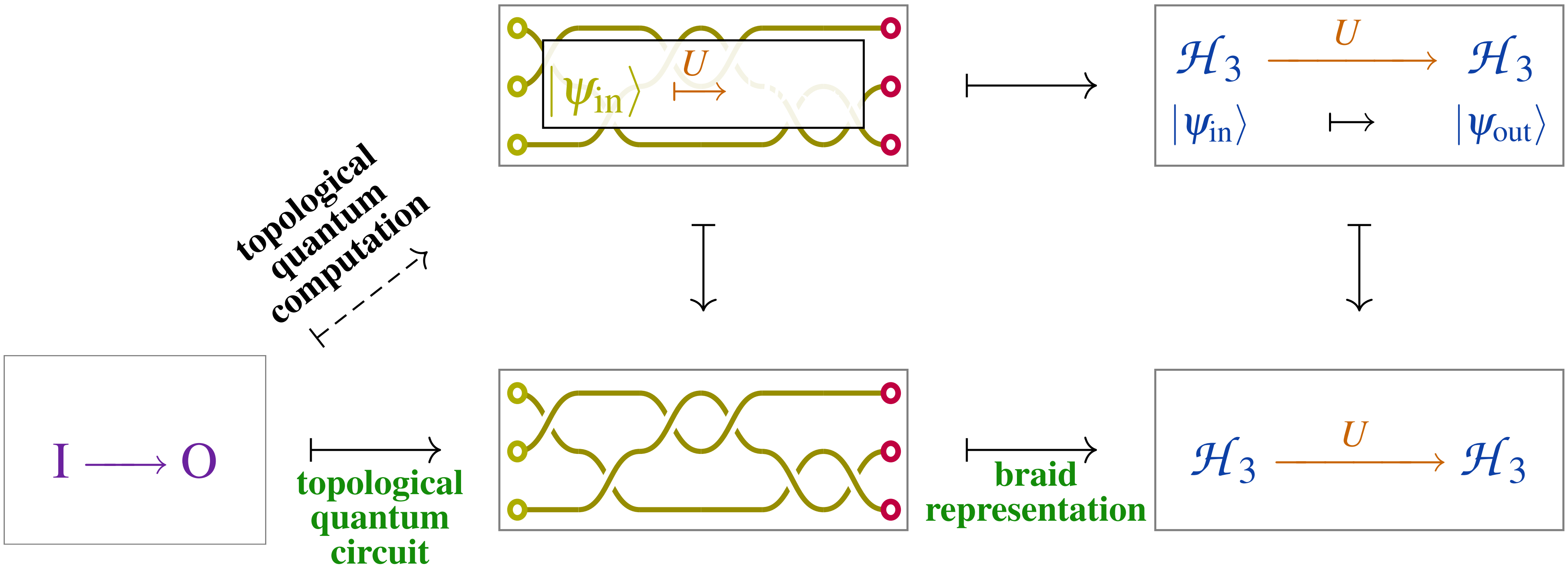
To compute is to **execute** sequences of **instructions** as composable **operations** on a chosen **state space**,

the case of
Topological Quantum Computation
[Sati & Schreiber, PlanQC **2022** 33 (2022)]



To compute is to **execute**
sequences of **instructions**
as composable **operations**
on a chosen **state space**,
turning a given **initial state**

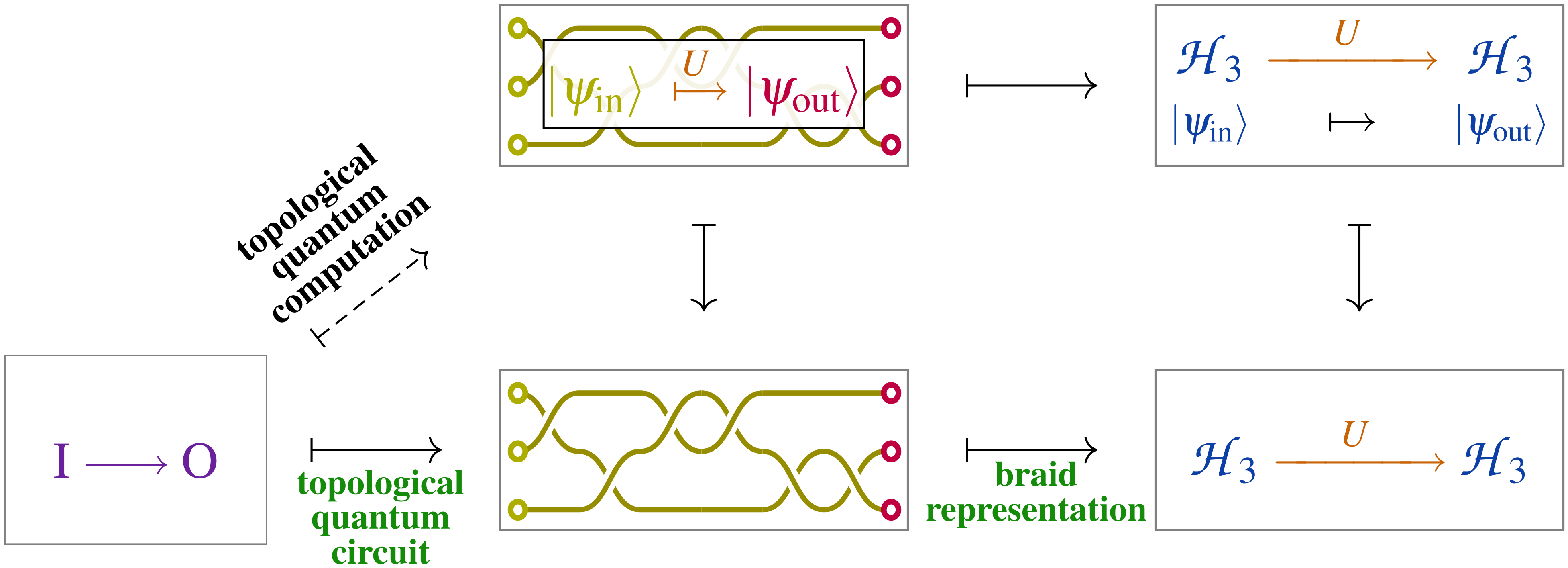
the case of
Topological Quantum Computation
[Sati & Schreiber, PlanQC **2022** 33 (2022)]



To compute is to **execute**
sequences of **instructions**
as composable **operations**
on a chosen **state space**,

turning a given **initial state**
into the computed **result**.

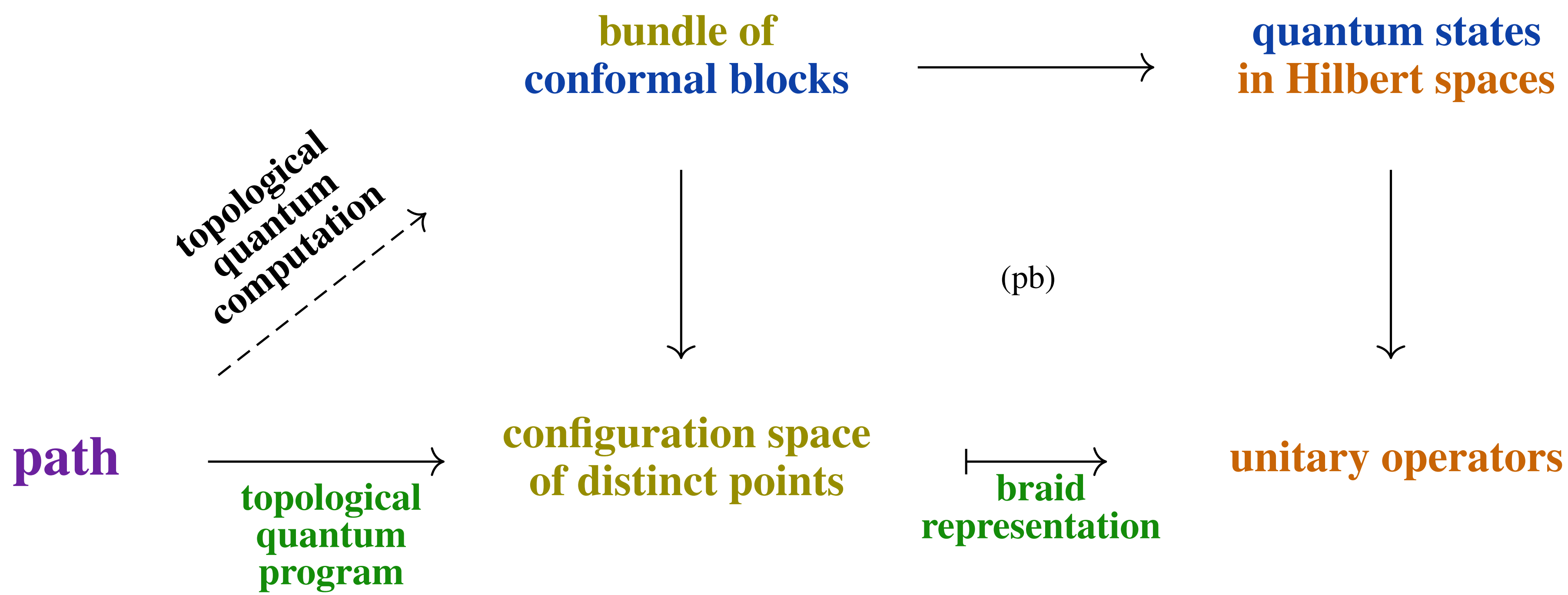
the case of
Topological Quantum Computation
[Sati & Schreiber, PlanQC **2022** 33 (2022)]



To compute is to **execute**
sequences of **instructions**
as composable **operations**
on a chosen **state space**,

turning a given **initial state**
into the computed **result**.

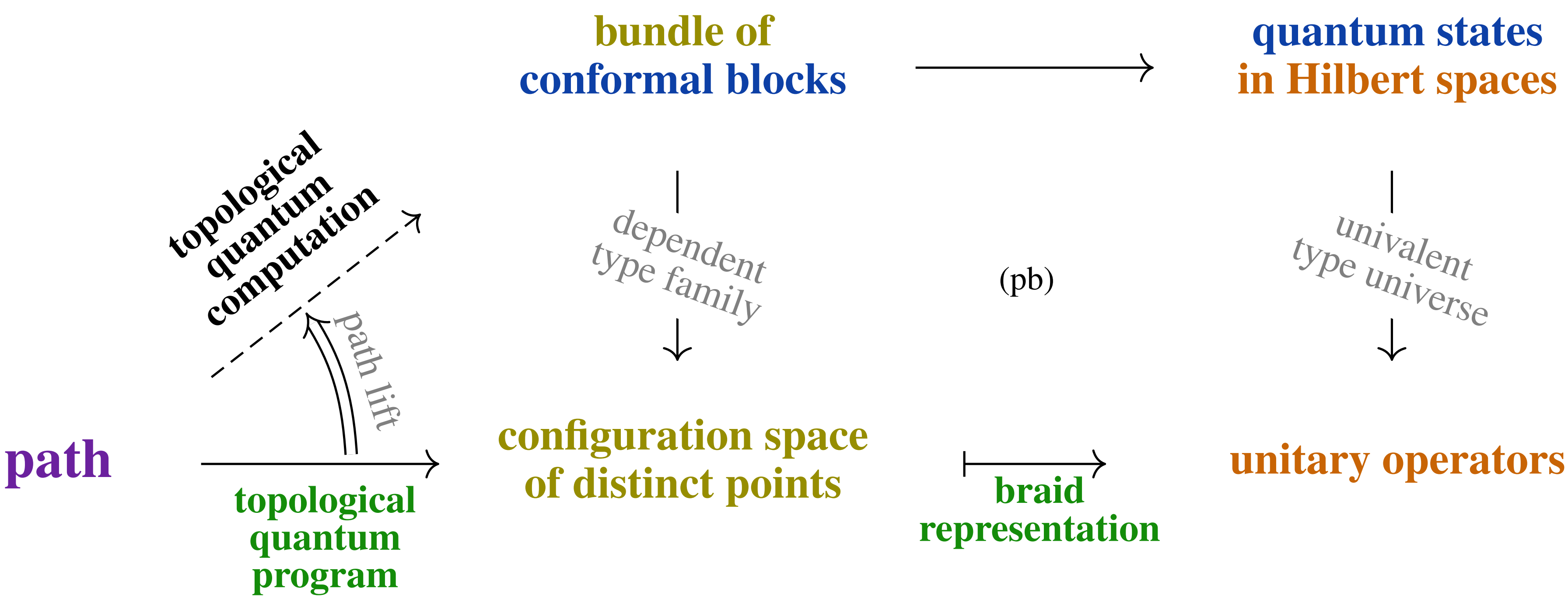
the case of
Topological Quantum Computation
[Sati & Schreiber, PlanQC **2022** 33 (2022)]



To compute is to **execute**
sequences of **instructions**
as composable **operations**
on a chosen **state space**,
turning a given **initial state**
into the computed **result**.

the case of
Topological Quantum Computation
[Sati & Schreiber, PlanQC **2022** 33 (2022)]

Claim: This has natural construction in HoTT languages:



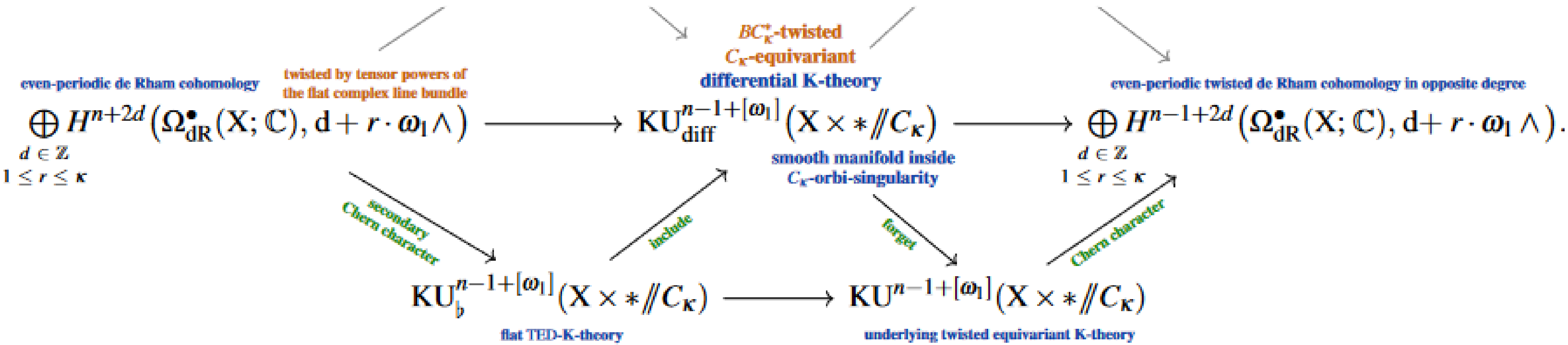
In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology;

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology;



arXiv > hep-th > arXiv:2203.11838

High Energy Physics - Theory

[Submitted on 22 Mar 2022]

Anyonic Defect Branes and Conformal Blocks in Twisted Equivariant Differential (TED) K-theory

Hisham Sati, Urs Schreiber

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology;

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

arXiv > cond-mat > arXiv:0901.2686

Search...

Help | A

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter

Condensed Matter > Mesoscale and Nanoscale Physics

[Submitted on 18 Jan 2009 (v1), last revised 20 Jan 2009 (this version, v2)]

Periodic table for topological insulators and superconductors

Alexei Kitaev

Gapped phases of noninteracting fermions, with and without charge conservation and time-reversal symmetry, are classified using Bott periodicity. The symmetry and spatial dimension determines a general universality class, which corresponds to one of the 2 types of complex and 8 types of real Clifford algebras. The phases within a given class are further characterized by a topological invariant, an element of some Abelian group that can be 0, Z, or Z_2. The interface between two infinite phases with different topological numbers must carry some gapless mode. Topological properties of finite systems are described in terms of K-homology. This classification is robust with respect to disorder, provided electron states near the Fermi energy are absent or localized. In some cases (e.g., integer quantum Hall systems) the K-theoretic classification is stable to interactions, but a counterexample

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

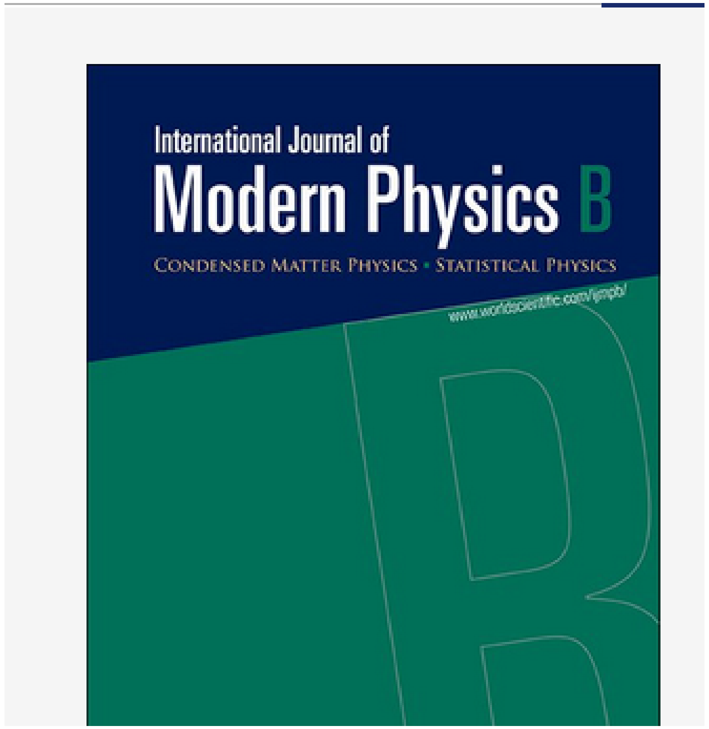
These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.

International Journal of Modern Physics B
| Vol. 05, No. 10, pp. 1641-1648 (1991)
| IV. CHERN-SIMONS FIELD ...

TOPOLOGICAL ORDERS AND CHERN-SIMONS THEORY IN STRONGLY CORRELATED QUANTUM LIQUID

XIAO-GANG WEN

https://doi.org/10.1142/S0217979291001541 |



In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.


arXiv > hep-th > arXiv:2206.13563

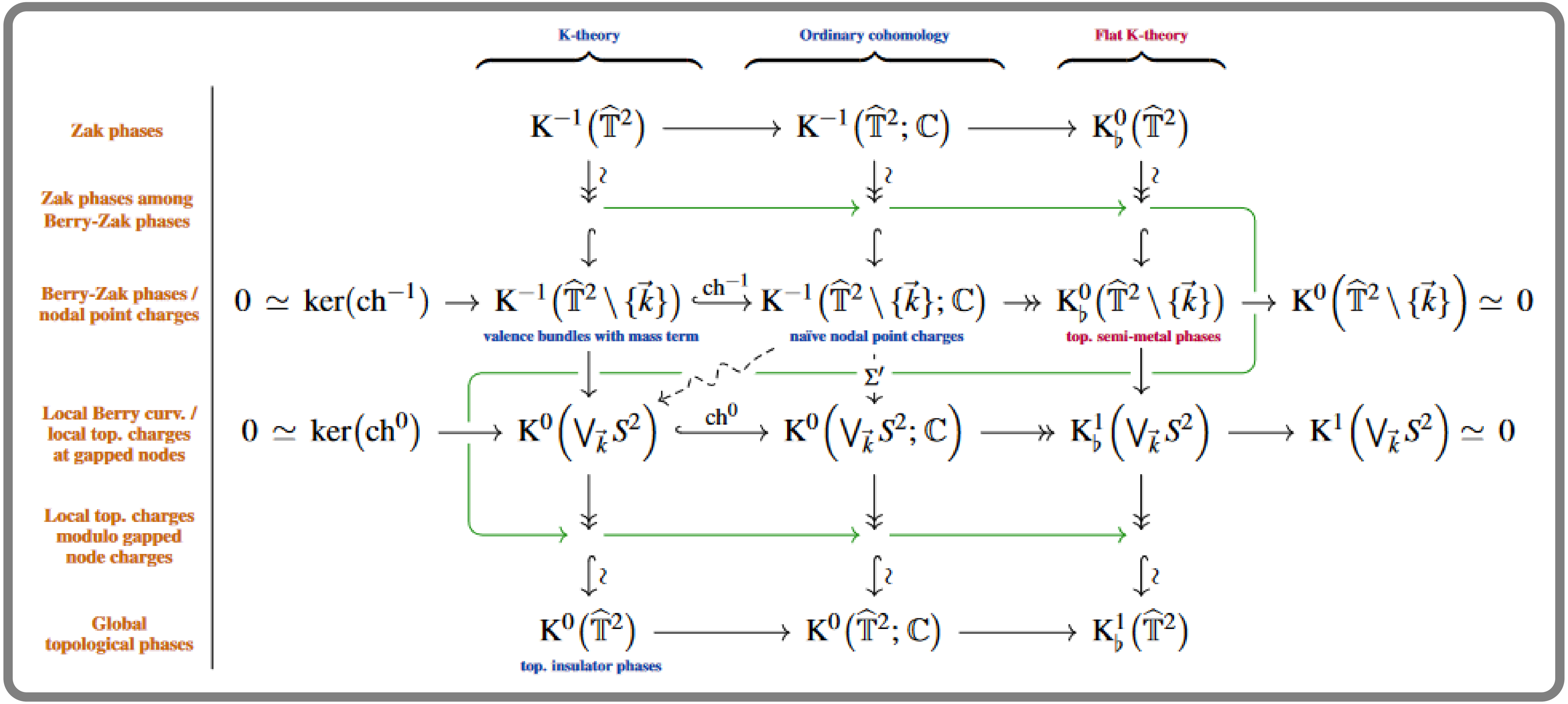
High Energy Physics - Theory

[Submitted on 27 Jun 2022]

Anyonic Topological Order in Twisted Equivariant Differential (TED) K-Theory

Hisham Sati, Urs Schreiber

expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.



In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.

But `TED-K` theory and hence topologically ordered phases are naturally expressible in an enhancement of HoTT languages called Cohesive HoTT.

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

Quantum Gauge Field Theory in Cohesive Homotopy Type Theory

Urs Schreiber (Radboud University Nijmegen), Michael Shulman (University of San Diego)

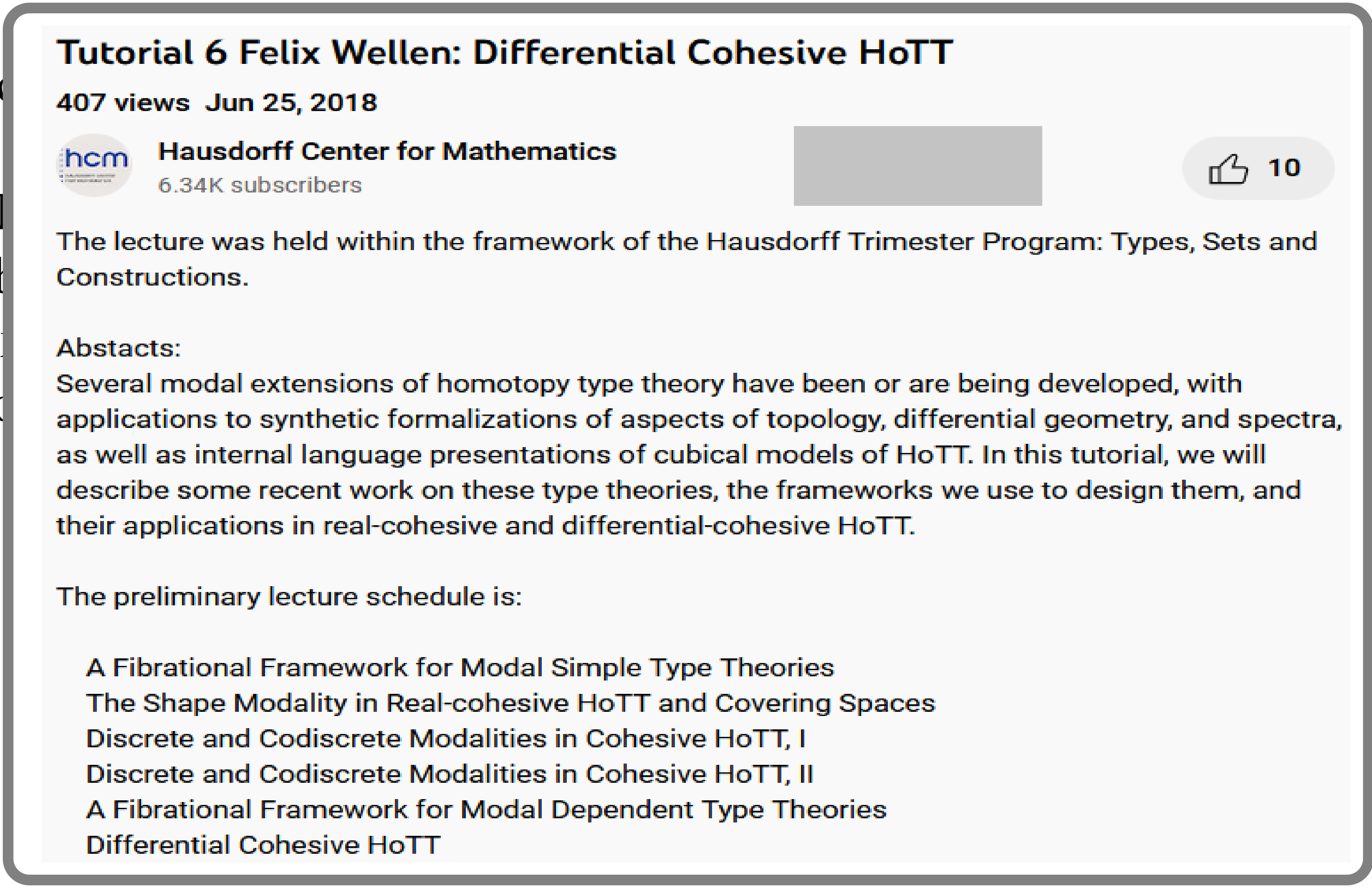
We implement in the formal language of homotopy type theory a new set of axioms called cohesion. Then we indicate how the resulting cohesive homotopy type theory naturally serves as a formal foundation for central concepts in quantum gauge field theory. This is a brief survey of work by the authors developed in detail elsewhere.

Comments:	In Proceedings QPL 2012, arXiv:1407.8427
Subjects:	**Mathematical Physics (math-ph)**; Logic in Computer Science (cs.LO); Category Theory (math.CT)
Cite as:	arXiv:1408.0054 **[math-ph]**
	(or arXiv:1408.0054v1 **[math-ph]** for this version)
	https://doi.org/10.48550/arXiv.1408.0054
Journal reference:	EPTCS 158, 2014, pp. 109-126
Related DOI:	https://doi.org/10.4204/EPTCS.158.8

But TED-K theory and hence topologically ordered phases are naturally expressible in an enhancement of HoTT languages called Cohesive HoTT.

In fa … able:

T … o
b … ;
e … er
h … s.



are naturally expressible in an enhancement of HoTT languages called Cohesive HoTT.

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.

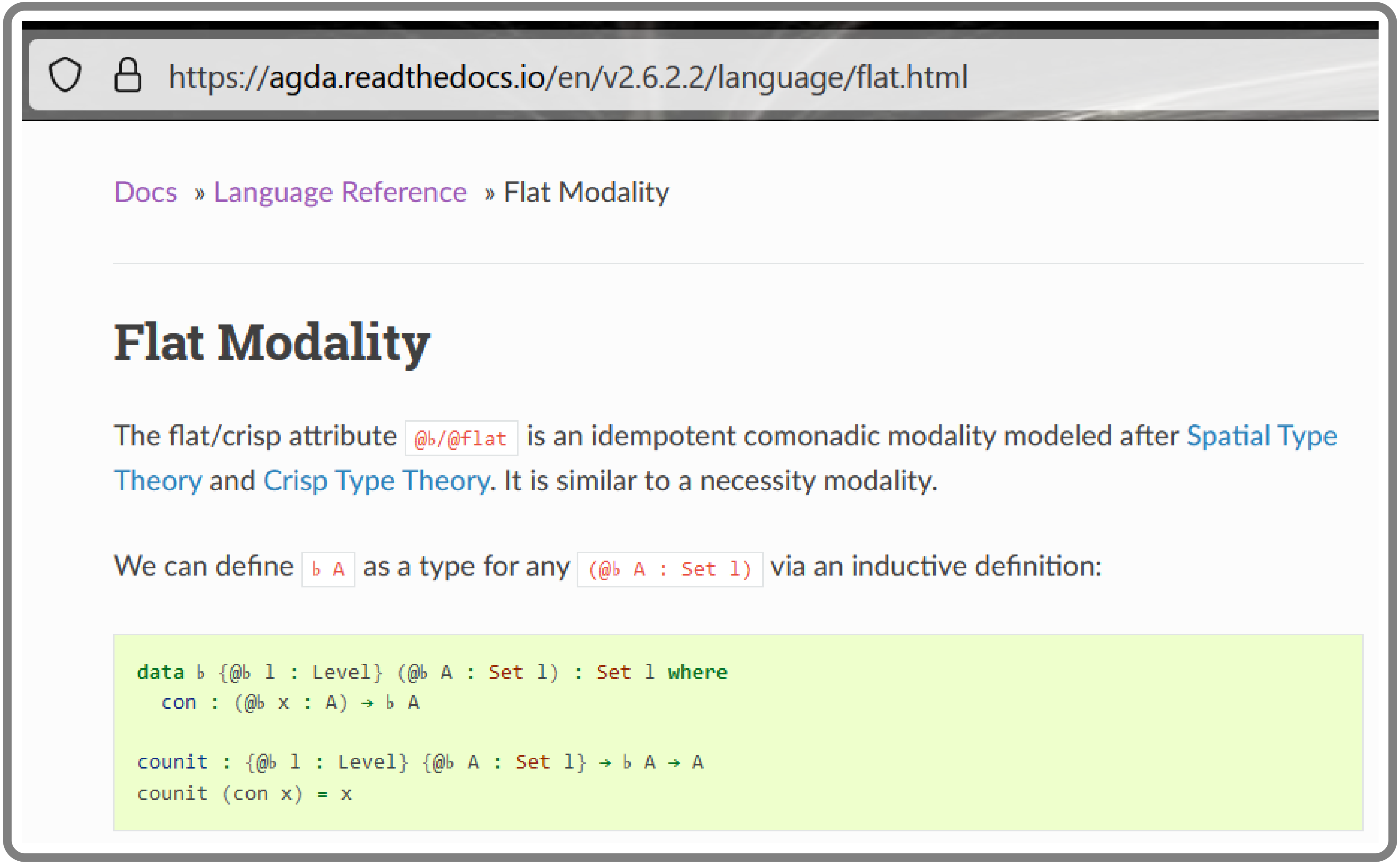
But `TED-K` and hence topologically ordered phases are naturally expressible in an enhancement of HoTT languages called Cohesive HoTT.

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.

But `TED-K` and hence topologically ordered phases are naturally expressible in an enhancement of HoTT languages called Cohesive HoTT.

Parts of Cohesive HoTT have already been implemented in `Agda`.



Parts of Cohesive HoTT have already been implemented in `Agda`.

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.

But `TED-K` and hence topologically ordered phases are naturally expressible in an enhancement of HoTT languages called Cohesive HoTT.

Parts of Cohesive HoTT have already been implemented in `Agda`.

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.

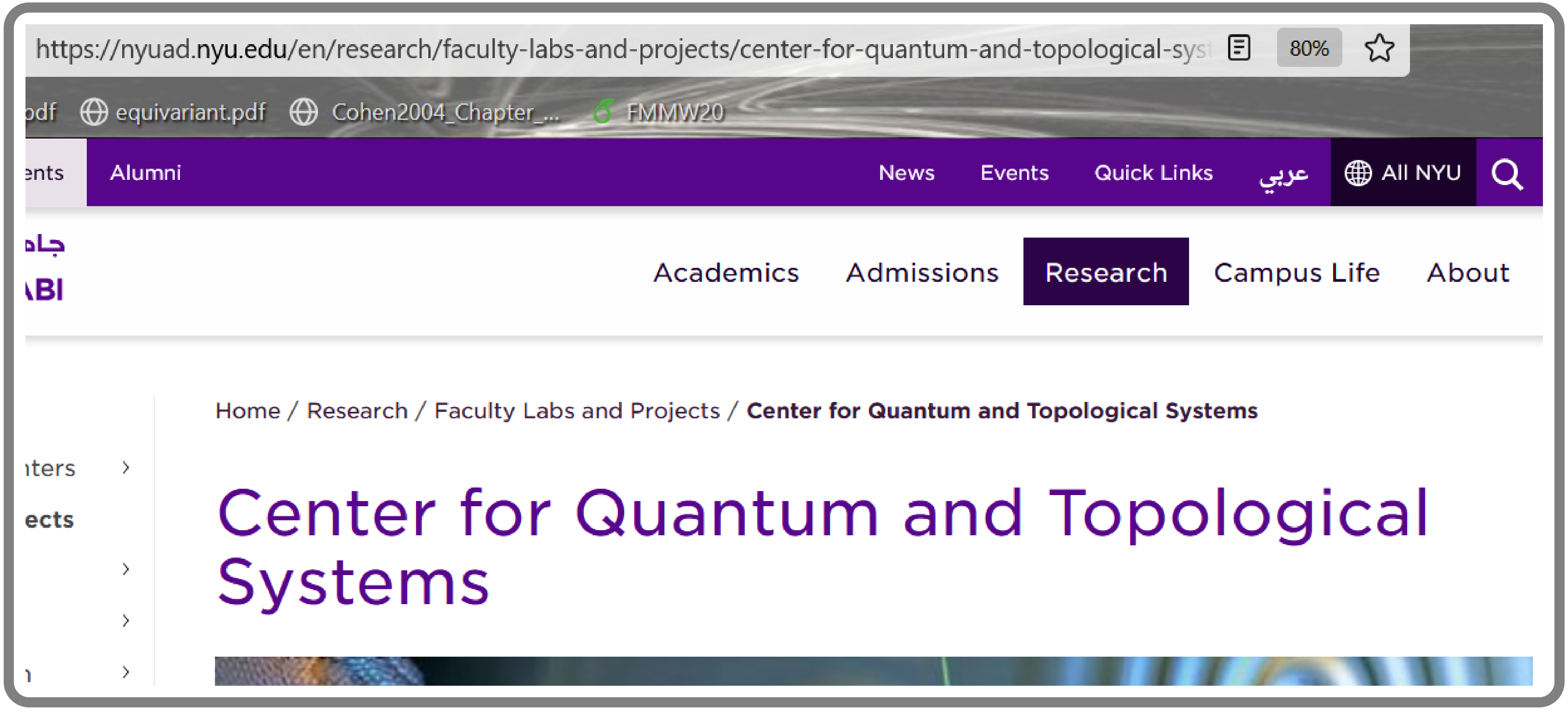
But `TED-K` and hence topologically ordered phases are naturally expressible in an enhancement of HoTT languages called Cohesive HoTT.

Parts of Cohesive HoTT have already been implemented in `Agda`.

Further development at our newly launched Research Center.

In fact, yet more fine-detail of TQC hardware is naturally HoTT-codeable:

These bundles/dependent types of $\mathfrak{su}(2)$-conformal blocks map to bundles of twisted equivariant differential (TED) K-cohomology; expressing characteristic properties of topological phases of matter hosting anyonic defects in their topologically ordered ground states.



Further development at our newly launched Research Center.

Topological Quantum Pogramming
in TED-K

Urs Schreiber on joint work with Hisham Sati

NYU AD Science Division, Program of Mathematics
& Center for Quantum and Topological Systems
New York University, Abu Dhabi

Thanks!

talk at:

PlanQC 2022 @ Ljubljana, 15 Sep 2022

slides and pointers at: https://ncatlab.org/schreiber/show/Topological+Quantum+Programming+in+TED-K