Bourbaki Isomorphism in Type Theory

David McAllester

TTIC

Progressive Levels of Automation

- Interactive Theorem Proving
- Atuomated Theorem Proving
- Auto-Formalization: Automated verification of published mathematics
- MathZero: Autonomous development of mathematics from the foundations

We are concerned here with the extreme forms of automation required for Auto-Formalization and MathZero.

Isomorphism

Mathematics is organized around concepts (groups, topological spaces, manifolds, . . .).

Each concept has an **abstract interface** where instances can be implemented (represented) in different concrete ways.

Each concept an has an associated notion of isomorphism.

Two objects are isomorphic if they are indistinguishable when accessed through their abstract interface.

Cryptomorphism

Two abstract concept are cryptomorphic if they "provide the same data" — we can define a bijection between their instances without violating the abstract interfaces involved.

Mathematical concepts are crypto-normalized — there is only one concept of "group" even though groups can be defined in extensionally different ways.

Recognizing crytomorphic equivalences seems essential for indexing and applying concepts.

Type Theory as Cognitive Science

The focus on set-theoretic type theory presented here is, in part, an attempt to understand the grammar of human mathematical language.

It is hoped that an understanding of the grammar of mathematical language will ultimately shed light on human language and cognition more generally.

Some History

Since its development in the 1970s, Martin-Löf Type Theory (MLTT) has made a distinction between judgemental equality and propositional equality.

In 1995 Haufmann and Streicher showed that propositional equality can be interpreted as isomorphism. But they did not give inference rules for deriving isomorphism from bijections — the did not "internalize" the semantics.

About 2009 Voevodsky introduced the **univalence axiom** which internalizes a semantics of isomorphism based on interpreting each type as a topological space and interpreting isomorphism as homotopy equivalence.

Bourbaki Isomorphism

However, the Bourbaki mathematicians gave a definition of structures and isomorphism in the 1930s.

As interpreted within type theory, Bourbaki defines a structure class, such as the class of groups, by a type expression.

Set-theoretic dependent type theory directly formalizes Bourbaki isomorphism without HoTT and without category theory.

Set-Theoretic Dependent Type Theory

Here we give a set of inference rules for deriving isomorphism from bijections (an internalization) based directly on the set-theoretic model of type theory and the 1930s Bourbaki notion of structure isomorphism.

There is no use of propositions-as-types (we have Bool rather than Prop) and axiom J (the MLTT axiom for propositional equality) is replaced by familiar Bourbaki-style inference rules for isomorphisnm.

Bourbaki Isomorphism

Bourbaki structure types have all set variables declared at the top level of the expression.

The top level set variables declare "carrier sets".

Bourbaki Isomorphism

Most familiar structure classes, such as groups, have only a single carrier set — the group elements.

In addition to the carrier sets, a structure type specifies functions and predicates over the carrier sets — the structure imposed on the carriers.

Two instances of a structure type are isomorphic if there exists a system of bijections between their carrier sets which carries the structure of the first to the structure of the second.

Bourbaki Structure Types

A Bourbaki structure type has the form

$$\sum_{s_1 : \mathbf{Set}} \cdots \sum_{s_n : \mathbf{Set}} \tau$$

where s_1, ..., s_n are carrier sets and τ is a set expression.

The Internalization of Isomorphism

$\sigma = \sum_{s_1 : \mathbf{Set}} \cdots \sum_{s_n : \mathbf{Set}} \ \tau \ \ \tau$ a set expression

$\Gamma \models \langle s_1, s_2, \ldots, s_n, x \rangle : \sigma$
$\Gamma \models \langle \tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n, \tilde{x} \rangle : \sigma$

$$\Gamma \models \begin{cases} \langle s_1, s_2, \ldots, s_n, x \rangle \ =_\sigma \langle \tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n, \tilde{x} \rangle \\ \\ \Leftrightarrow \exists \ f_1 : \mathrm{Bi}(s_1, \tilde{s}_1) \cdots f_n : \mathrm{Bi}(s_n, \tilde{s}_n) \quad \mathrm{EQ}(\tau, x, \tilde{x}) \end{cases}$$

Bourbaki Isomorphism at Simple Types (Bourbaki 1930s)

$\sigma = \sum_{s_1:\mathbf{Set}} \cdots \sum_{s_n:\mathbf{Set}} \ \tau \ \ \tau$ a set expression
$\Gamma \models \langle s_1, s_2, \ldots, s_n, x\rangle : \sigma$
$\Gamma \models \langle \tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n, \tilde{x}\rangle : \sigma$

$$\Gamma \models \begin{cases} \langle s_1, s_2, \ldots, s_n, x\rangle \ =_\sigma \langle \tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n, \tilde{x}\rangle \\ \\ \quad \Leftrightarrow \exists \ \ f_1 : \mathrm{Bi}(s_1, \tilde{s}_1) \cdots f_n : \mathrm{Bi}(s_n, \tilde{s}_n) \ \ \mathrm{EQ}(\tau, x, \tilde{x}) \end{cases}$$

$$\mathrm{EQ}(\tau, x, \tilde{x}) \ \equiv \ x = \tilde{x} \text{ for } \tau \text{ not containing any } s_i$$

$$\mathrm{EQ}(s_i, x, \tilde{x}) \ \equiv \ f_i(x) = \tilde{x}$$

$$\mathrm{EQ}\left((S_{z:\tau}\Phi), x, \tilde{x}\right) \ \equiv \ \mathrm{EQ}(\tau, x, \tilde{x})$$

$$\mathrm{EQ}((\tau_1 \times \tau_2), x, \tilde{x}) \ \equiv \ \mathrm{EQ}(\tau_1, \pi_1(x), \pi_1(\tilde{x})) \wedge \mathrm{EQ}(\tau_2, \pi_2(x), \pi_2(\tilde{x}))$$

$$\mathrm{EQ}(\tau_1 \to \tau_2, \ f, \tilde{f}) \ \equiv \ \forall \ \ x:\tau_1, \ \tilde{x}:\tilde{\tau}_1 \ \ \ \mathrm{EQ}(\tau_1, x, \tilde{x}) \ \Rightarrow \ \mathrm{EQ}(\tau_2, f(x), \tilde{f}(\tilde{x}))$$

Isomorphism Congruence

Isomorphism congruence is needed to generalize Bourbaki isomorphism to types other than structure types.

$$\frac{\Gamma; x:\sigma \models e[x]:\tau \quad x \text{ not free in } \tau \qquad \Gamma \models u =_\sigma v}{\Gamma \models e[u] =_\tau e[v]}$$

But the validity of isomorphism congruence relies on delicate well-formedness constraints.

Isomorphism Congruence at General Dependent Types

A type not of the Bourbaki form is "Group with action":

$$\sum_{G:\mathbf{Group}} \sum_{s:\mathbf{Set}} S_{f:(\pi_1(G)\to(s\to s))}\, \Phi[G, s, f]$$

To prove isomorphism congruence we must define isomorphism at all class expressions, not just the Bourbaki classes.

The Set/Class Distinction

The type **Group** is a class — it is too large to be a set.

However, for a set variable s, the function space $s \to s$ is a set.

An expression e is set-level if the constant **Set** does not appear in e outside of Boolean expressions.

An expression is class-level if it is not set-level.

Well Formedness of Equality

For a set-theoretic equality $u = v$ to be well-formed we require that there exists a **set expression** σ such that

$$\Gamma \models u : \sigma \quad \text{and} \quad \Gamma \models v : \sigma.$$

Without this condition we have the following counter-example to isomorphism congruence.

$$s : \mathbf{Set};\ x : s;\ P : \left(\sum_{w : \mathbf{Set}} w \right) \models (\pi_2(P) = x) : \mathbf{Bool}.$$

Here we can have isomorphic pointed sets P and P' giving different truth values for the formula.

Functor Types

A naive semantics for functor types violates isomorphism congruence. Consider:

$$P : \mathbf{Group} \to \mathbf{Bool};\ G : \mathbf{Group} \models P(G) : \mathbf{Bool}$$

A naive semantics of all functions allows P to distinguish isomorphic groups.

This can be fixed by interpreting functor types as sets of lambda terms (a term model).

Set-level function spaces are still interpreted as all functions.

Cryptomorphism

People immediately recognize when two different structure types are "the same" or "provide the same data" even when the structure (the signature) is different.

For example, $\Sigma_{s:\mathbf{Set}}\ \sigma[s] \times \tau[s]$ provides the same data as $\Sigma_{s:\mathbf{Set}}\ \tau[s] \times \sigma[s]$.

A group can be defined as a tuple of a set, a group operation, an inverse operation, and an identity element or as a pair of a set and a group operation such that an identity element and inverses exist.

Following Birkhoff and Rota we call this cryptomorphism.

Cryptomorphism

Here we formally define two classes σ and τ to be cryptomorphic if there exist functors $F : \sigma \to \tau$ and $G : \tau \to \sigma$ such that

$$x : \sigma \models G(F(x)) = x$$

and

$$y : \tau \models F(G(y)) = y$$

Completeness of Isomorphism Inference Rules

By isomorphism congruence we have that if F is a cryptomorphism from σ to τ then $u =_\sigma v$ iff $F(u) =_\tau F(v)$.

The paper shows that every class expression is crytomorphic to a Bourbaki structure class.

These together imply that every isomorphism relation can be expressed as a formula in the base language.

History Revisited

Martin-Löf Type Theory (MLTT) has always made a distinction between definitional equality and propositional equality.

Haufmann and Streicher, 1995, showed that propositional equality can be interpreted as isomorphism as distinct from numerical equality.

The model can be interpreted as replacing $s : \mathbf{Set}$ with $s : \mathbf{GRPD}$ where $\mathbf{GRPD}$ is the class of all groupoids (categories in which every morphism is a isomorphism).

They give a semantics for the constructs of the language such that every type denotes a groupoid.

For example $s : \mathbf{GRPD} \models s \to s : \mathbf{GRPD}$.

They define $u =_\sigma v$ to mean that u and v are isomorphic in the groupoid σ (actually, under proposition as types, $u =_\sigma v$ is the type containing the isomorphisms in σ from u to v).

The Groupoid model

The formulation assumes propositions as types, is based on axiom J, and does not provide set-theoretic propositional equality.

The Bourbaki isomorphism rule is not mentioned.

Homotopy Type Theory (HoTT)

HoTT can be interpreted as replacing $x : \mathbf{Set}$ with $x : \mathbf{Top}$ where **Top** is the class of topological spaces definable by simplicial sets.

In HoTT each type denotes a topological space.

For example $s : \mathbf{Top} \models s \to s : \mathbf{Top}$ where $s \to s$ is interpreted as the space of continuous maps.

$\sum_{x:\sigma} \tau[x]$ is interpreted as a fibration.

Homotopy Type Theory (HoTT)

HoTT interprets $u =_\sigma v$ as saying that the space u contains a homotopy path from u to v (again, under proposition as a types, we have that the type $u =_\sigma v$ is a set representing all the homotopy paths from u to v).

Sophisticated algebraic topology seems required to understand what the formulas mean and what holds true in the calculus.

Summary

- Bourbaki isomorphism can be directly and fully internalized in naive set-theoretic type theory. There is no need for homotopy theory.

- Syntactic well-formedness conditions guarantee isomorphism congruence.

- Type theory can be interpreted as cognitive science.

- Isomorphism and cryptomorphism are fundamental to achieving Auto-Formalization and MathZero.

END