Complementary foundations for mathematics: when do we choose?

Michael Shulman

University of San Diego

April 6, 2022
Joint Mathematics Meetings
AMS Special Session on "Competing Foundations for Mathematics: How Do We Choose?"

Outline

We need a new metaphor

The word foundation suggests a physical building:

- A building can only have one foundation.
- If the foundation has a flaw, the whole building can collapse.
- Can't change the foundation and leave the building alone.

But none of this is true for "mathematical foundations", at least insofar as they matter to a mathematician.

We need a new metaphor

The word foundation suggests a physical building:

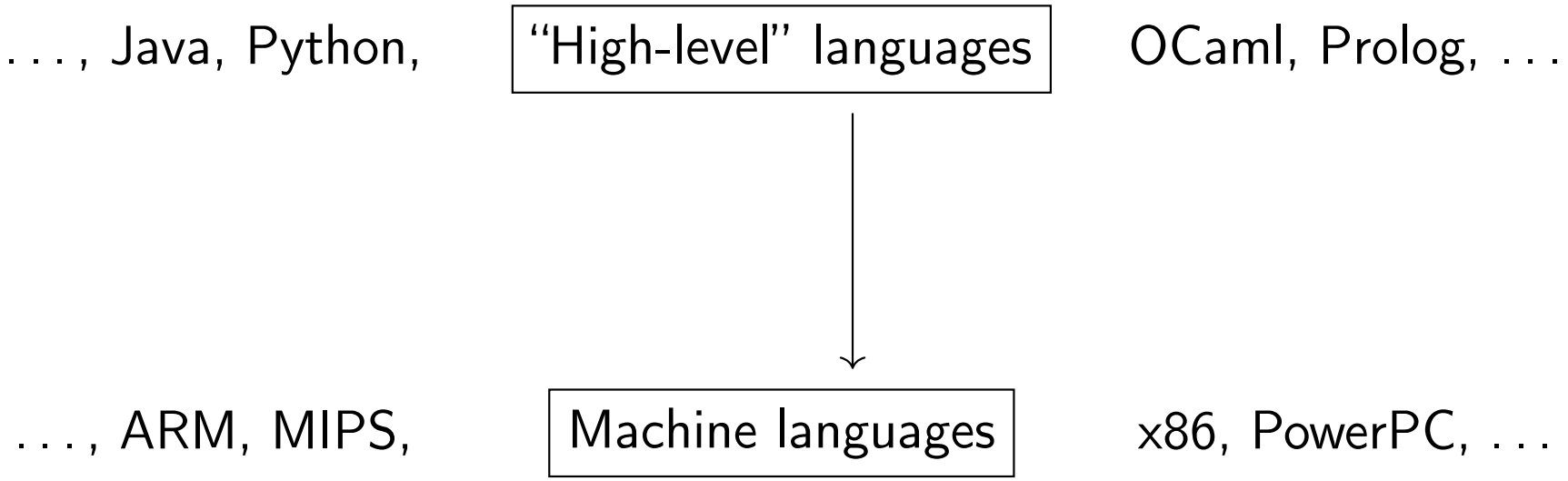
- A building can only have one foundation.
- If the foundation has a flaw, the whole building can collapse.
- Can't change the foundation and leave the building alone.

But none of this is true for "mathematical foundations", at least insofar as they matter to a mathematician.

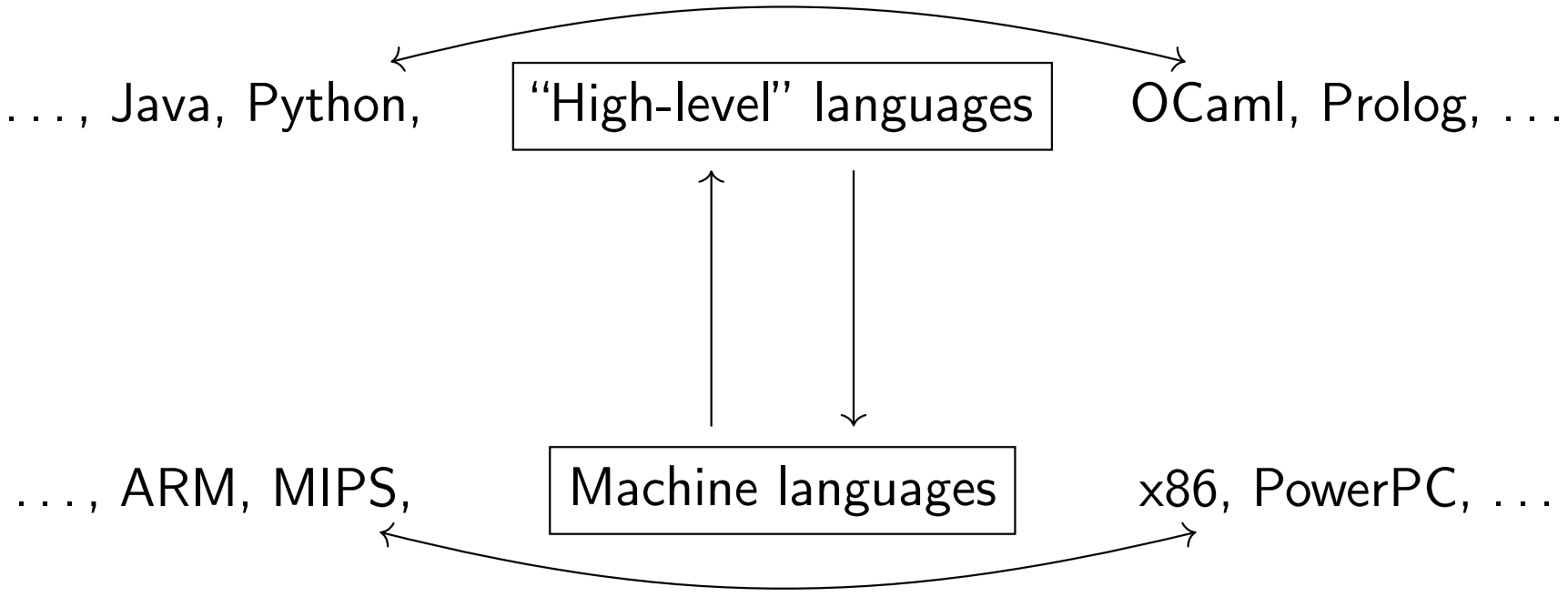
I prefer to think of a "mathematical foundation" as a programming language for mathematics.

- Many programming languages coexist without trouble.
- A bug in the compiler can be fixed, and most code is fine.
- Languages or libraries implementing the same interface can be used interchangeably.

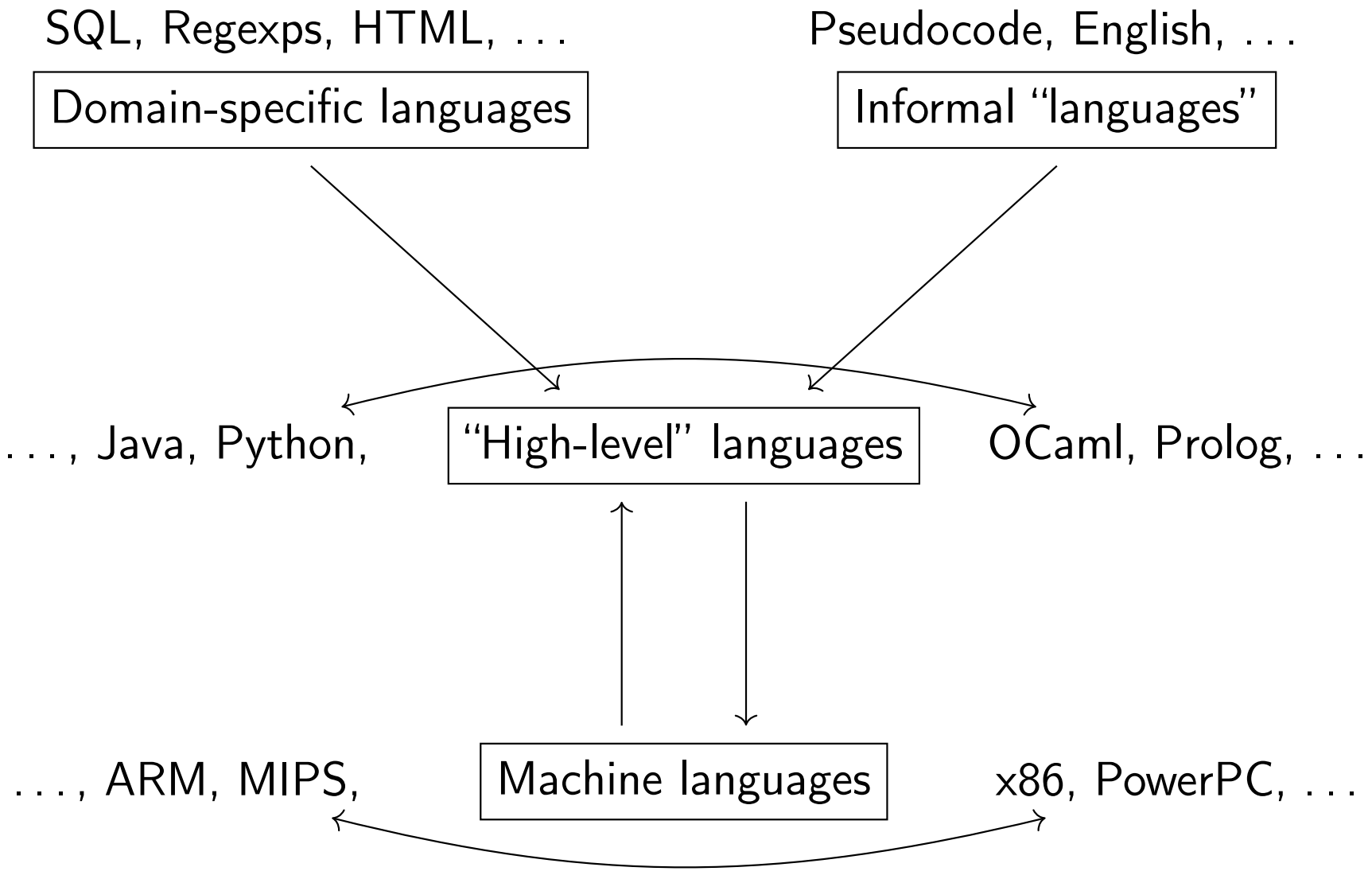
Expanding that metaphor



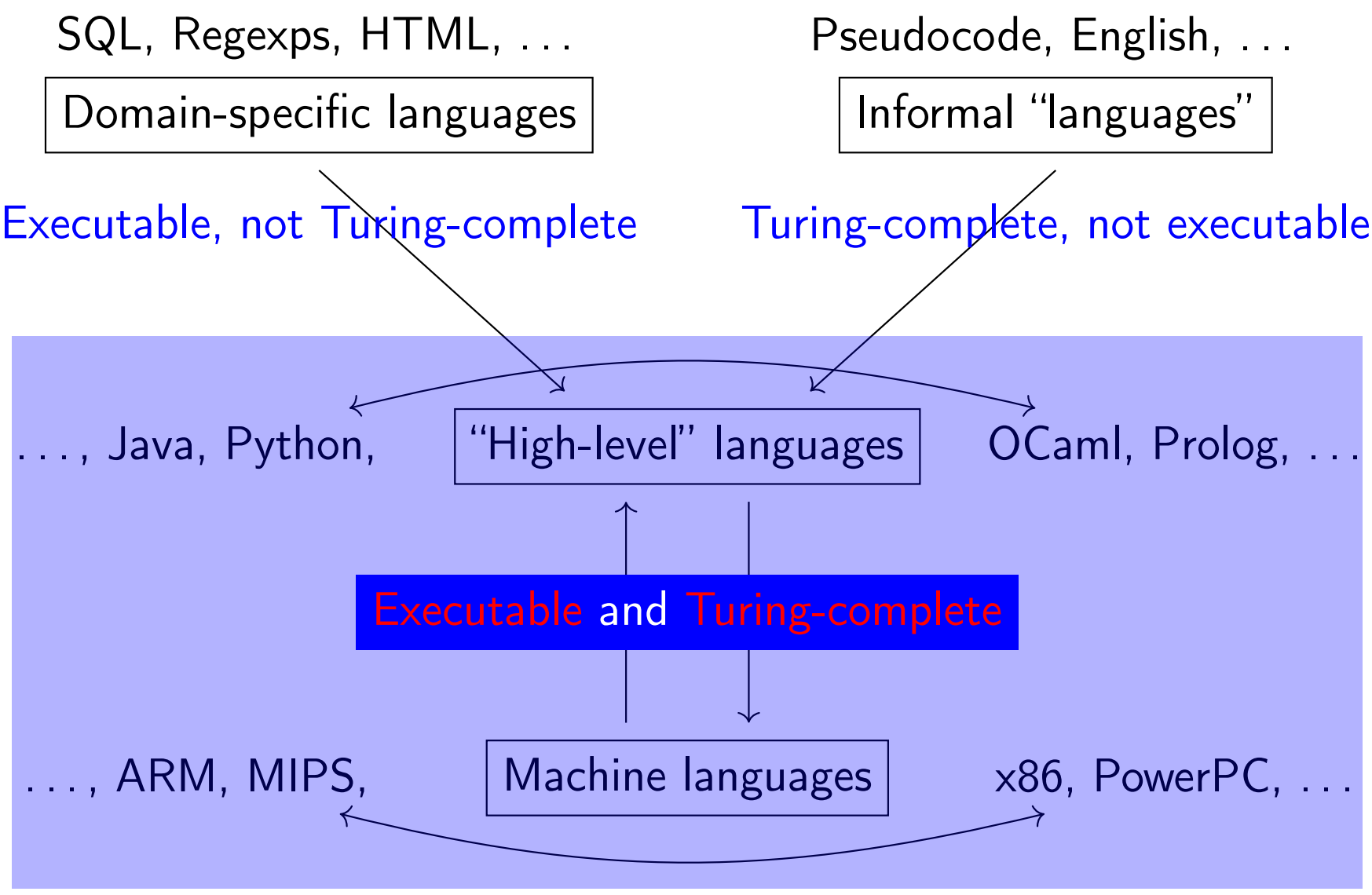
Expanding that metaphor



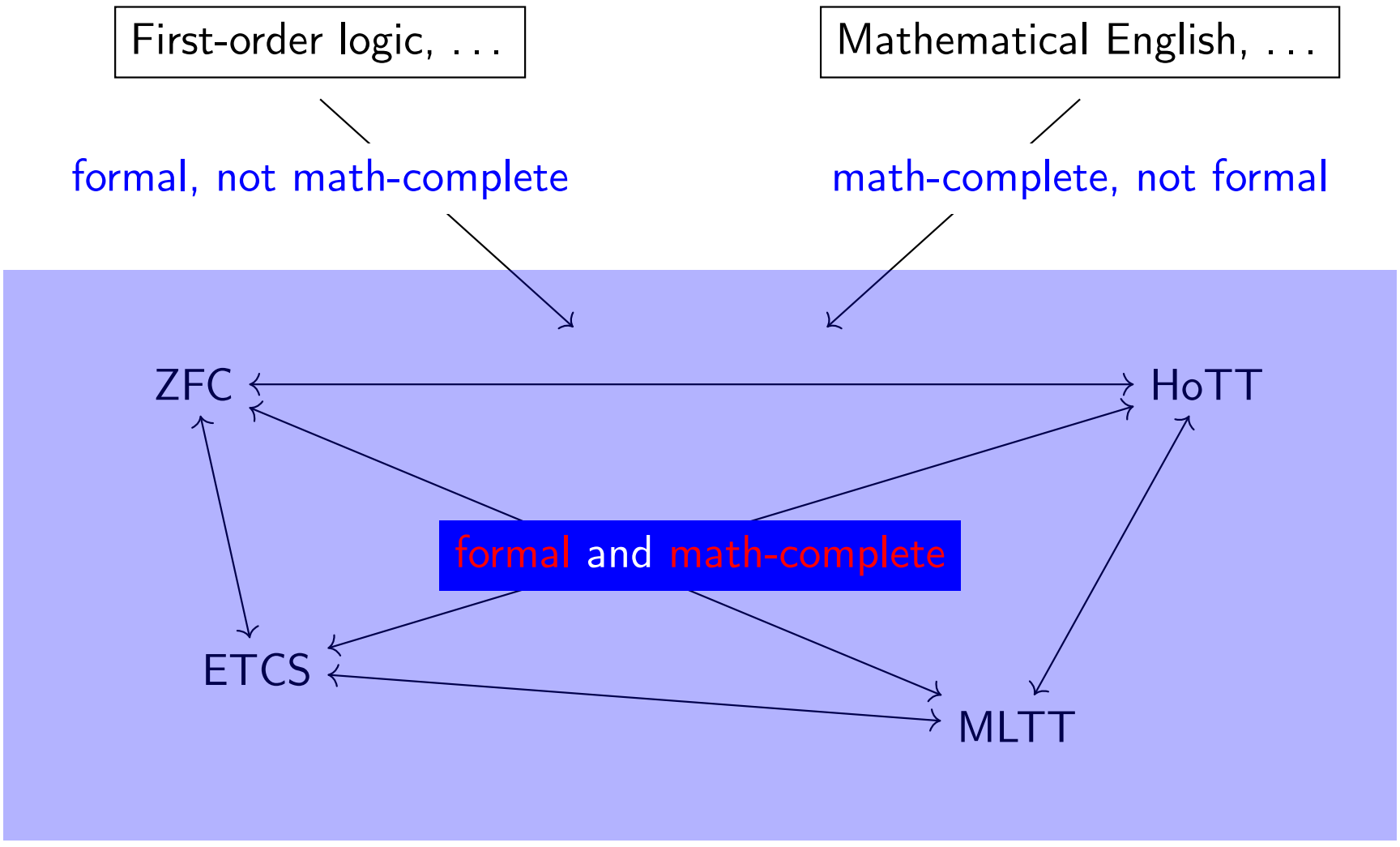
Expanding that metaphor



Expanding that metaphor

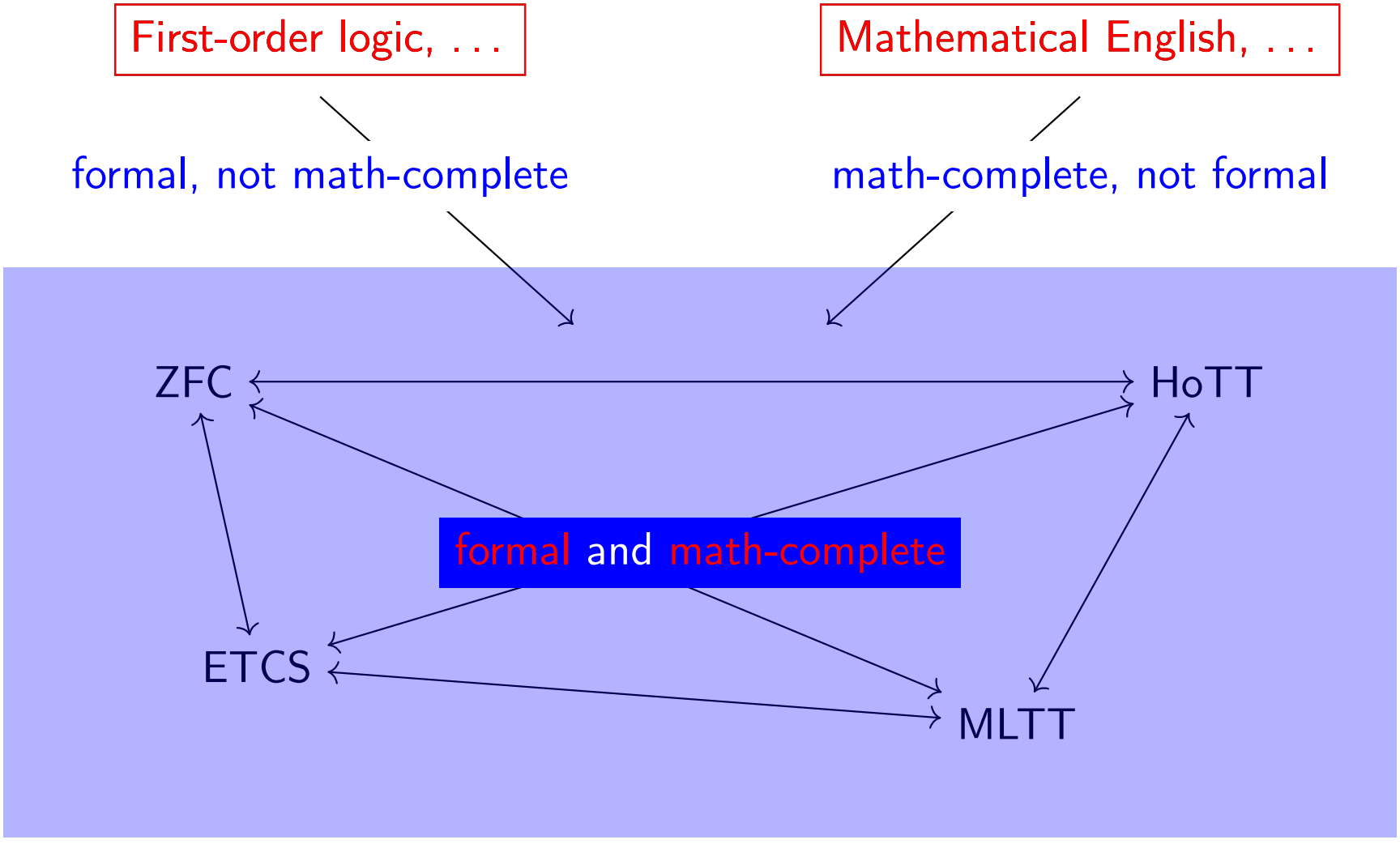


The landscape of mathematics

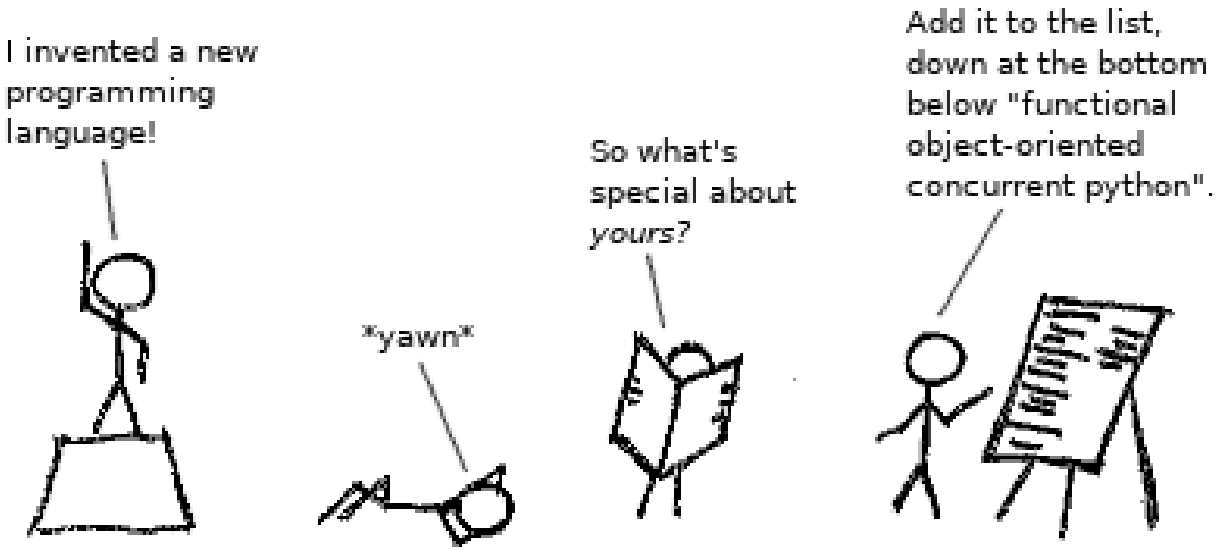


The landscape of mathematics

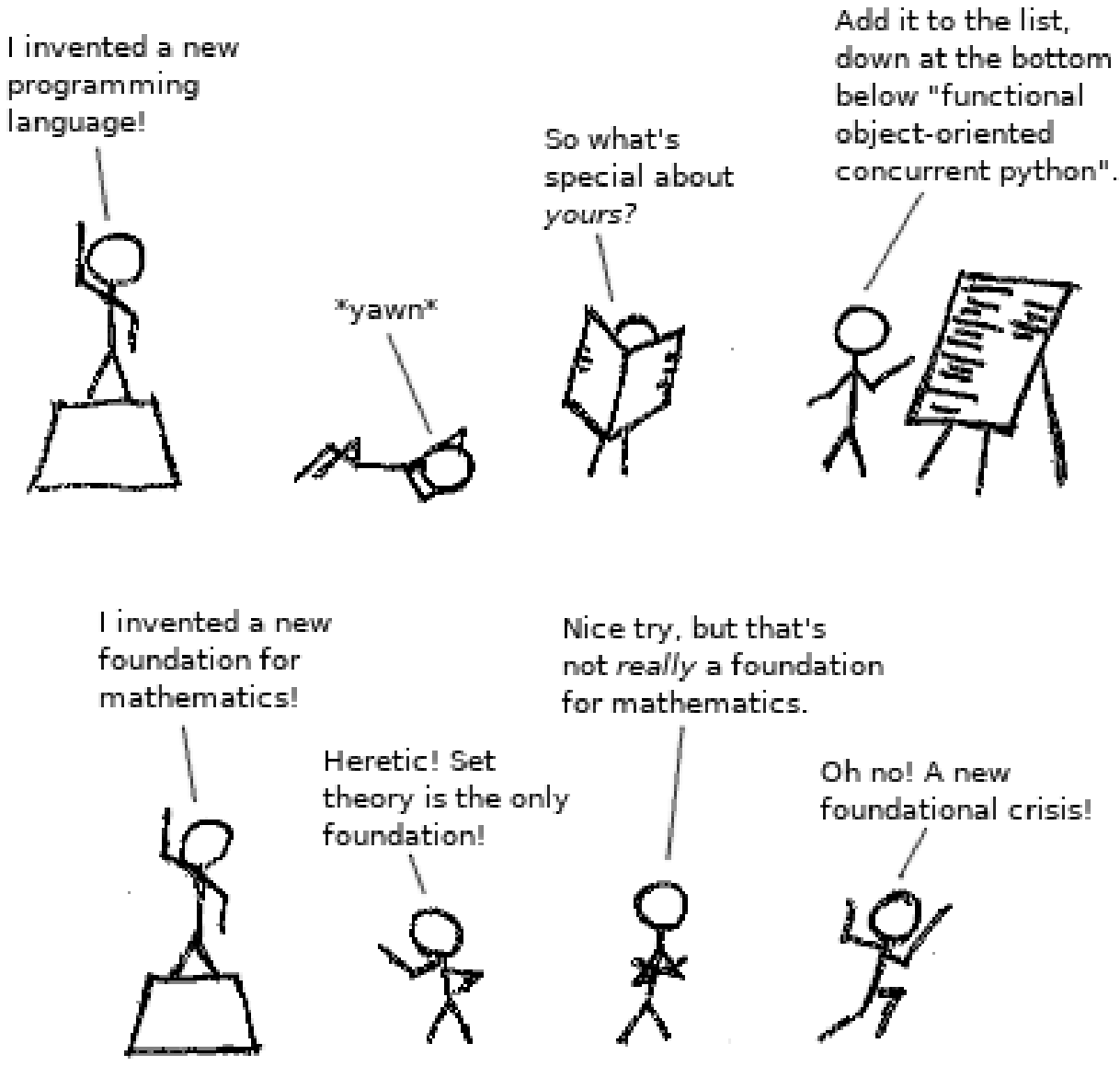
Most mathematicians work up here, and don't care about what's below.



What's different?



What's different?



It doesn't have to be that way

We should treat "mathematical foundations" more like programming languages.

- It doesn't make sense to argue about which is "correct".
- As Colin said, all foundations can do the same things.
- Ask not just how they do it, but how well?
  Then choose an appropriate one for each task.

Let's see what that looks like with some examples.

Outline

ZFC and ETCS

Zermelo–Fraenkel set theory (ZFC)

- Formulated in single-sorted logic with axioms
- Everything is a set, and can be an element of other sets
- Well-founded global membership predicate $\in$
- Strict global equality

Elementary Theory of the Category of Sets (ETCS)

(The closest thing to "category theory" as a foundation.)

- Best formulated in two-sorted logic: sets and functions
- Axioms assert category-theoretic universal properties
- "Elements" of X are functions $1 \to X$, "subsets" are injections
- Sets are not elements of other sets
- Equality only for functions with same domain/codomain

ZFC $\leftrightarrow$ ETCS

The sets and functions in ZFC satisfy the axioms of ETCS+R (ETCS augmented with a replacement/collection axiom).

In ETCS+R, the extensional well-founded accessible pointed graphs satisfy the axioms of ZFC.

- A *graph* is a set with a binary relation $\prec$.
- A *pointed* graph is equipped with a *root* element $\star$.
- A pointed graph is *accessible* if every element admits a path to the root $x \prec x_1 \prec \cdots \prec x_n \prec \star$.
- A *well-founded* graph has no infinite path $\cdots \prec y_2 \prec y_1 \prec y_0$.
- An *extensional* graph satisfies $(\forall z.(z \prec x \leftrightarrow z \prec y)) \rightarrow x = y$.
- For graphs X, Y, say $X \in Y$ if $\exists y \prec \star$ in Y such that X is isomorphic to the graph of elements admitting some path to y.

ZFC vs ETCS: Some bogus arguments

Against ETCS:

- "It's ugly to define elements and subsets as functions."
  - Can add separate sorts for elements and subsets to ETCS (c.f. e.g. SEAR). But collapsing them reduces redundancy.
  - It's no uglier than defining $(a, b) = \{\{a\}, \{a, b\}\}$ in ZFC.
- "The elements of a subset should *be* elements of its superset."
  - Can also modify ETCS to make this true.
  - This sort of thing is everywhere (e.g. $2 \in \mathbb{N}$ vs $\frac{2}{1} \in \mathbb{Q}$).
- "Can't use sets as data structures": a group in ZFC is a pair (G, m), but in ETCS we can't pair up a set with a function.
  - This is just part of the encoding: statements about groups become statements about sets with a function.

Against ZFC:

- "Defining 3 as a set like $\{0, 1, 2\}$ gives truth values to meaningless statements like $1 \in 3$ and $\pi \in 3$."
  - Math is full of such "irrelevant implementation details", like whether $\mathbb{R}$ consists of Cauchy sequences or Dedekind cuts.

Those problems belong to the elaborator

These sorts of problems should be handled by non-core aspects of an implementation:

- Elaboration and syntactic sugar: the compiler can use clever algorithms to determine what the user probably meant.
- Abstraction barriers, like namespacing and access control. After we use $\{\{a\}, \{a, b\}\}$ to implement the interface of (a, b), or functions $1 \to X$ to implement the interface of "elements", or Cauchy sequences to implement the interface of $\mathbb{R}$, external code is only allowed to use the interface.
- Implicit coercions: If we declare $i : \mathbb{N} \to \mathbb{Q}$ as such, the compiler can insert it wherever needed to make things compile.

In programming, such features are considered part of a language. In mathematics, they aren't considered part of the "foundations", but they are needed in any formalized implementation, and are performed mentally by human mathematicians.

ZFC vs ETCS: More bogus arguments

Against ETCS:

- "It requires category theory, which is sophisticated and not foundational".
  - The axioms of ETCS are simple first-order statements.
  - Incorporating ideas of category theory doesn't make them harder or non-foundational, just as ZFC is not invalidated by incorporating ideas of well-foundedness.

Against ZFC:

- "The complexity of ETCS$\rightarrow$ZFC means ETCS is simpler."
  - Or, it means ETCS ignores structure that's naturally present.
  - It's a *tribute* to ZFC that it automatically produces such a rich structure from simple axioms.

A real advantage of ETCS: internal languages

Observation

Many categories arising in nature ("toposes") satisfy all the axioms of ETCS+R except

- Well-pointed: a set X is determined by its elements $1 \to X$.
- The axiom of choice ($\Rightarrow$ the law of excluded middle).

We can force well-pointedness by interpreting "elements" as generalized elements $U \to X$ for variable objects U, yielding Kripke–Joyal semantics.

In this way, any topos interprets constructive ETCS.

Free models

Toposes are an algebraic structure: their operations are those of geometric logic ($\wedge, \bigvee, \exists$). Thus, any geometric theory $\mathbb{T}$ generates a free topos (a.k.a. classifying topos).

So if we want to show that $\mathbb{T}$ is consistent with ETCS+R, we can just:

1. Give an explicit construction of its classifying topos,
2. Check that this topos is nontrivial, and
3. Check that it still satisfies the axiom of choice.

E.g. with $\mathbb{T} =$ the theory of an uncountable proper subset of $\mathbb{R}$, we get consistency of $\neg CH$.

Forcing

To do this in ZFC, we can either:

1. Construct a classifying topos, interpret ETCS+R in it, then build a model of ZFC in that.
2. Write down the Kripke-Joyal semantics of the model of ZFC in the classifying topos without mentioning either explicitly.
3. Find a small enough model of ZFC such that the classifying topos "already exists" outside it, like defining the polynomial ring $\mathbb{Z}[x]$ as the set $\{ a + b\pi \mid a, b \in \mathbb{Z} \} \subseteq \mathbb{R}$.

All of these work, but the algebraic perspective is clean and general.

A real advantage of ZFC: inner models

In a model $(V, \in)$ of ZF(C), we can look for $M \subseteq V$ that also models ZF(C) with the same $\in$.

Given the von Neumann hierarchy $V = \bigcup_\alpha V_\alpha$ indexed by ordinals, we can construct $M = \bigcup_\alpha M_\alpha$ by transfinite recursion.

Example

If $L_{\alpha+1}$ consists of all the sets definable from L_α, we have Gödel's constructible universe L, which satisfies AC and GCH.

To do this in ETCS, we seemingly have to construct ZF(C) first.

Outline

The need for types

Many bogus objections to ZFC/ETCS involve their paucity of types.

In ZFC:

- If we had a separate type of ordered pairs, we wouldn't have to encode (a, b) as $\{\{a\}, \{a, b\}\}$.
- If we had a separate type of natural numbers, then $\pi \in 3$ would be ill-typed and meaningless.

In ETCS:

- If we had separate types of elements and subsets, we wouldn't have to encode them as $1 \to X$ or $A \rightarrowtail X$.
- If we had a type of sets, we could pair a set with a multiplication to make a type of groups.

Dependent type theory

- Replace the homogeneous collection of sets with a heterogeneous collection of types.
- Type-forming operations: $A \times B$, $A \sqcup B$, B^A, ...
- Each type has different elements: $(a, b) : A \times B$, while $\mathrm{inl}(a) : A \sqcup B$ and $(x \mapsto x^2) : \mathbb{N}^{\mathbb{N}}$, etc.
- Types are elements of a universe type, $A : \mathscr{U}$.
  To avoid paradoxes, $\mathscr{U} : \mathscr{U}_1$, while $\mathscr{U}_1 : \mathscr{U}_2$,...
- Families of types are functions $B : \mathscr{U}^A$.
  Then have $\prod_{a:A} B(a)$ and $\coprod_{a:A} B(a)$.
- Propositions are particular types, proofs are their elements.
- Equality only meaningful for elements of the same type.

For now, consider classical extensional type theory (CETT), where $a : A$ and $A = B : \mathscr{U}$ imply $a : B$, and the axiom of choice holds.

Modulo details with universes, this interprets mutually with ETCS (hence also ZFC).

Some bogus arguments against DTT

"It's ugly/wrong/unnatural to define propositions as types."

- This is just a redundancy-reducing trick, like elements-as-functions in ETCS and pairs-as-sets in ZFC.

"Too complicated, with all those type formers."

- Apples and oranges: DTT includes a lot of the "non-core" features we need anyway in using ETCS/ZFC.
- The logic of ETCS/ZFC has lots of connectives ($\wedge, \vee, \Rightarrow, \exists, \forall$); in DTT these are incorporated via propositions-as-types.

Types also solve other problems

- Insert an implicit coercion $i : \mathbb{N} \to \mathbb{R}$ whenever we expect something of type $\mathbb{R}$ but are given something of type $\mathbb{N}$.
- To hide the definition of $\mathbb{R}$, export "$\mathbb{R}$" as an abstract type.
- Can use data structures as data structures.

$$\text{Groups} := \coprod_{G:\mathscr{U}} \coprod_{m:G\times G\to G}$$

Outline

Synthetic mathematics

ZFC, ETCS, and CETT all agree that the basic objects of mathematics are set-like, with discrete distinguishable "elements" unconnected to each other.

But there are other kinds of objects that are rich enough to build mathematics out of, e.g.

1. Topological spaces
2. Smooth spaces
3. Computational data structures
4. Quantum objects
5. Sets with infinitesimals
6. Higher groupoids
7. Higher categories

Synthetic topology (SynTop)

- Built on constructive ETT: no choice or excluded middle.
- Now view types as spaces, with elements "cohering" together and all functions being "continuous".
- Choice fails because choices can't be made continuously.
- Add "nonclassical" axioms and structure describing cohesion.
- The naïve definition of "group" now automatically represents a "topological group", etc.
- Basic examples turn out discrete as long as that's sensible (e.g. $\mathbb{Z}$, finite groups), while synthetic "cohesion" arises automatically when needed (e.g. $\mathbb{R}$, profinite groups), instead of having to be added by hand.

SynTop $\leftrightarrow$ set-like theories

The discrete types in SynTop satisfy ETCS/CETT.

Some collections of spaces in ZFC/ETCS/CETT satisfy SynTop.

- Identify a good small category $\mathscr{T}$ of *test spaces* (e.g. $\mathbb{R}^n$'s)
- Equip $\mathscr{T}$ with a *Grothendieck topology* specifying which families of continuous maps are *covering*.
- A *sheaf* is a functor $X : \mathscr{T}^{\mathrm{op}} \to$ Set taking covers to limits.
- The collection of sheaves on $\mathscr{T}$ satisfies SynTop.

Some bogus arguments against SynTop

"Spaces are defined using sets, hence not foundational."

- In a set-based theory, spaces are defined using sets. But in SynTop, spaces are a primitive undefined thing, and sets are defined in terms of those (the discrete spaces).

"Spaces are a sophisticated concept; foundations should be simple."

- We think spaces are sophisticated because 20th century mathematics trained us to build them out of sets. But thinkers back to Aristotle argued instead for a non-punctiform continuum.
- In SynTop, spaces are just as simple as "sets" are in ZFC/ETCS.

Internal Set Theory (IST)

There are innumerable possible "synthetic" theories, to which most of the same arguments apply. I will highlight just two more.

In Internal Set Theory (Nelson):

- We augment ZFC by a new primitive predicate standard.
- We add axioms, including transfer and an idealization principle ensuring that there are nonstandard objects.
- In particular, we get infinitesimals behaving like those of Robinson's non-standard analysis.

If we forget "standard", then IST includes ZFC. Conversely, an "ultrapower" construction like Robinson's models IST from ZFC.

Homotopy type theory (HoTT)

- Built on intensional dependent type theory.
- Now view types as ∞-groupoids.
- The equality/identity type $x = y$ can contain multiple distinct identifications, acting like isomorphisms in a groupoid.
- All the structure of an ∞-groupoid follows automatically.
- We can do synthetic homotopy theory, e.g. homotopy groups.
- Basic examples (e.g. $\mathbb{N}$, $\mathbb{R}$, algebra, topology, most of math) turn out groupoidally discrete, while synthetic groupoid structure arises automatically when needed (e.g. universes and types built from them, like the type of groups).

HoTT $\leftrightarrow$ set-like theories

The discrete types in HoTT satisfy ETCS/ETT.

- A is discrete if for $x, y : A$, any two $p, q : x = y$ are equal.

The ∞-groupoids in ZFC/ETCS/CETT satisfy HoTT.

- The *simplex category* $\mathbb{\Delta}$ consists of nonempty finite ordered sets.
- A *simplicial set* is a functor $X : \mathbb{\Delta}^{\mathrm{op}} \to \mathrm{Set}$. It has sets X_n of *abstract n-simplices* with faces and degeneracies.
- A *Kan complex* is a simplicial set in which every *horn* (potential n-simplex missing interior and one face) has a filler.
- The collection of Kan complexes satisfies HoTT (Voevodsky).

Some bogus arguments against HoTT

"∞-groupoids are defined using sets, hence not foundational."

- In a set-based theory, groupoids are defined using sets. But in HoTT, ∞-groupoids are a primitive undefined thing, and sets are defined in terms of those.

"∞-groupoids are sophisticated; foundations should be simple."

- We think ∞-groupoids are sophisticated because 20th century mathematics trained us to build them out of sets.
- In HoTT, ∞-groupoids are just as simple as "sets" in ZFC/ETCS.

The advantages of synthetic mathematics

A synthetic theory makes it easier to talk about its basic objects.

- In SynTop, topological structure is carried through all constructions automatically.
- In IST, all functions also act on infinitesimals, so we can do analysis, $f'(x) \approx \frac{f(x+\delta)-f(x)}{\delta}$ for $\delta \approx 0$.
- In HoTT, homotopy-theoretic structure is carried through, avoiding infinite-dimensional combinatorics. With new primitives like univalence, we obtain an entirely new style of proof for homotopical calculations: synthetic homotopy theory.

The first synthetic theory: ZFC?

ZFC has powerful tools for working with well-foundedness (e.g. Mostowski collapse), and the complexity of building ZFC from ETCS is not unlike that of modeling synthetic theories.

Is ZFC really a synthetic theory of well-founded structures?

- Imagine a mathematician raised on ETCS/CETT encountering ZFC for the first time: "These hereditary membership structures are sophisticated things built out of sets, hence shouldn't be foundational!"
- But ZFC is foundational; and so are SynTop, IST, HoTT, ...

Mathematical worlds

We have learnt to look for connections between branches of mathematics, and now we must also learn to look for connections that span worlds of mathematics. . . . Mathematicians have to be educated so that they develop multiple mathematical intuitions that help them feel how the worlds of mathematics behave.

. . . any technical definition of a mathematical world can never be exhaustive. A world of mathematics may be a forcing extension of set theory, or a topos, or a pretopos, or a model of type theory, or any other structure within which it is possible to interpret the basic language of mathematics.

–Andrej Bauer

Returning to that metaphor

Coders have individual preferences about programming languages. So do mathematicians for their foundations.

But still, some programming languages are objectively better at some things. Likewise for mathematical foundations.

Rather than arguing for a "correct" or "best" foundation, we should study them all, and select the best for any particular task.

> *"A language that doesn't affect the way you think about programming is not worth knowing."* – *Alan J. Perlis*