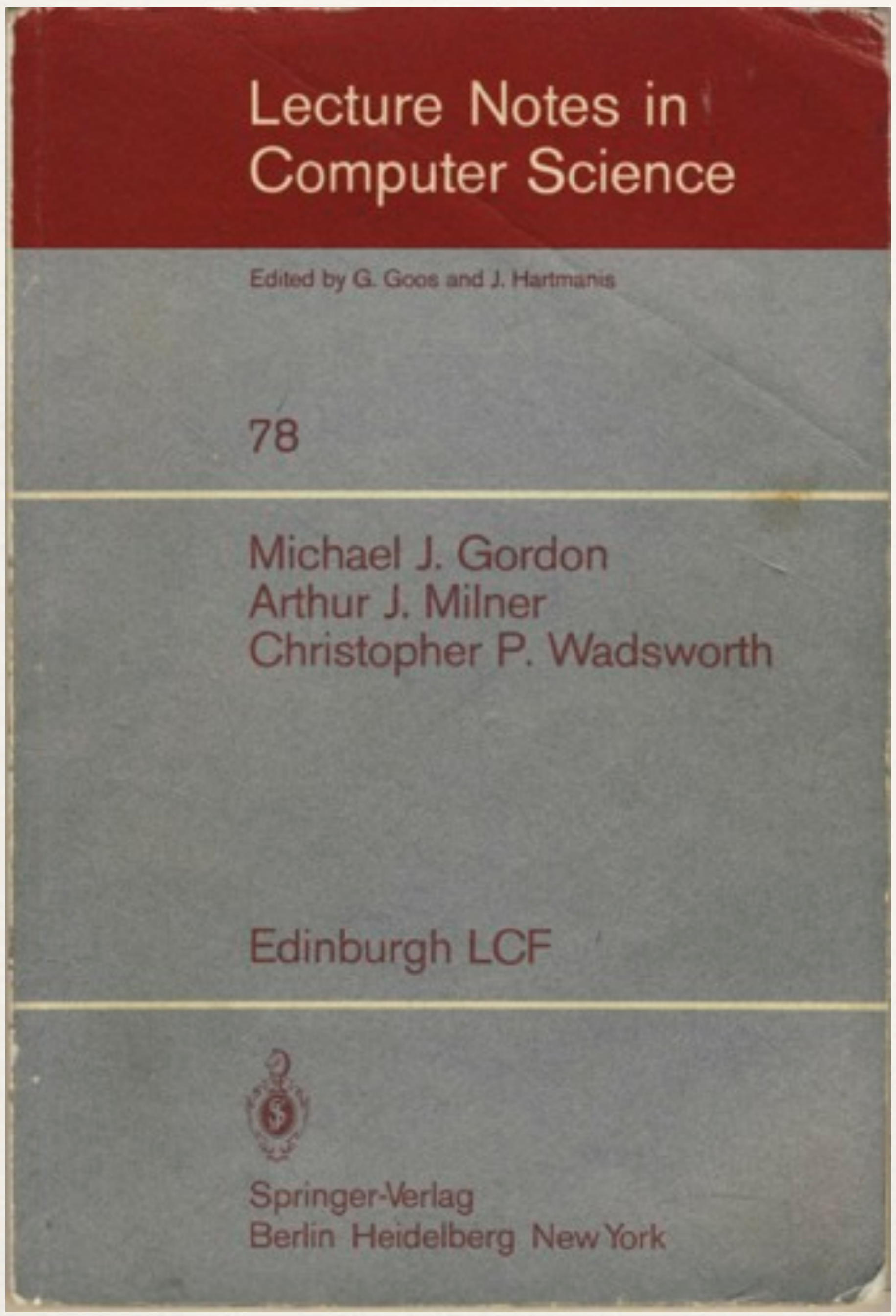
LCF + Logical Frameworks = Isabelle (25 Years Later)

Lawrence C. Paulson, Computer Laboratory, University of Cambridge

16 April 2012 — Milner Symposium, Edinburgh

1979

Edinburgh LCF: From the Preface



"… the ML type discipline is used… so that—whatever complex procedures are defined—all values of type <u>thm</u> must be theorems, as only inferences can compute such values…. This security releases us from the need to preserve whole proofs… — an important practical gain since large proofs tended to clog up the working space…" [page IV]

Robin Milner's LCF Architecture

- A programmable *metalanguage* (ML)
- An *abstract type* of *theorems*, to ensure soundness
- … and to eliminate *the need to store proofs*
- Plus the original objective: to support a novel and interesting formalism, Scott's Logic for Computable Functions.

LCF Proof Style

"There are three important elements in our proposed 'natural' proof style. Most important is the adoption of natural deduction… here **inference rules** play the dominant role…

the second element is to use **goal-directed proof** procedures… one aim in designing ML was thus to make it easy to program **tactics and tacticals**.…

The third element of natural proof style is to emphasise **theory structure**" [*Edinburgh LCF*, page 2]

Goal-Directed Proof in LCF

$$\text{goal} \xrightarrow{\text{tactic}} \text{subgoal}_1, \ldots, \text{subgoal}_n$$

$$\text{theorem!} \xleftarrow{\text{validation}} \text{theorem}_1, \ldots, \text{theorem}_n$$

- *Inference rules*: coded as ML functions from premises to the conclusion, within the abstract type barrier
- *Tactics*: coded as ML functions from the goal to the subgoals, **outside** of the abstract type
- … but also returning a *validation* function coded using inference rules

Invalid Tactics

$$\text{goal} \xrightarrow{\text{bad tactic}} \text{subgoal}_1, \ldots, \text{subgoal}_n$$

$$??? \xleftarrow{\text{bad validation}} \text{theorem}_1, \ldots, \text{theorem}_n$$

- An *invalid* tactic is one that doesn't correctly invert an inference rule.
  - It doesn't violate soundness, but it wastes your time!
  - Proving the subgoals *doesn't* prove the original goal.
  - The function delivers the wrong theorem, or otherwise fails.

Proof without Programming?

- Most inference rules are symbolic. Can they be expressed *declaratively*?
- No need to code inference rules
- … and no need to code their inverses, to create tactics.
- No validation functions. No invalid tactics.
- Instead of calling functions, simply paste partial proofs together.

Some Declarative Inference Rules

$$\bigwedge AB \,.\, [\![A]\!] \Rightarrow ([\![B]\!] \Rightarrow [\![A \,\&\, B]\!])$$

Built-in concept of "for all"

Built-in concept of "implies"

User-defined logical symbols

Higher-order variables

$$\bigwedge F \,.\, (\bigwedge x \,.\, [\![F(x)]\!]) \Rightarrow [\![\forall x . F(x)]\!]$$

Declaring the Rules of Intuitionistic Propositional Logic

$$\bigwedge AB\,.\,[\![A]\!] \Rightarrow ([\![B]\!] \Rightarrow [\![A \,\&\, B]\!]) \qquad (\&\mathrm{I})$$

$$\bigwedge AB\,.\,[\![A \,\&\, B]\!] \Rightarrow [\![A]\!] \qquad \bigwedge AB\,.\,[\![A \,\&\, B]\!] \Rightarrow [\![B]\!] \qquad (\&\mathrm{E})$$

$$\bigwedge AB\,.\,[\![A]\!] \Rightarrow [\![A \vee B]\!] \qquad \bigwedge AB\,.\,[\![B]\!] \Rightarrow [\![A \vee B]\!] \qquad (\vee\mathrm{I})$$

$$\bigwedge ABC\,.\,[\![A \vee B]\!] \Rightarrow ([\![A]\!] \Rightarrow [\![C]\!]) \Rightarrow ([\![B]\!] \Rightarrow [\![C]\!]) \Rightarrow [\![C]\!] \qquad (\vee\mathrm{E})$$

$$\bigwedge AB\,.\,([\![A]\!] \Rightarrow [\![B]\!]) \Rightarrow [\![A \supset B]\!] \qquad (\supset \mathrm{I})$$

$$\bigwedge AB\,.\,[\![A \supset B]\!] \Rightarrow [\![A]\!] \Rightarrow [\![B]\!] \qquad (\supset \mathrm{E})$$

$$\bigwedge A\,.\,[\![\bot]\!] \Rightarrow [\![A]\!] \qquad (\bot\mathrm{E})$$

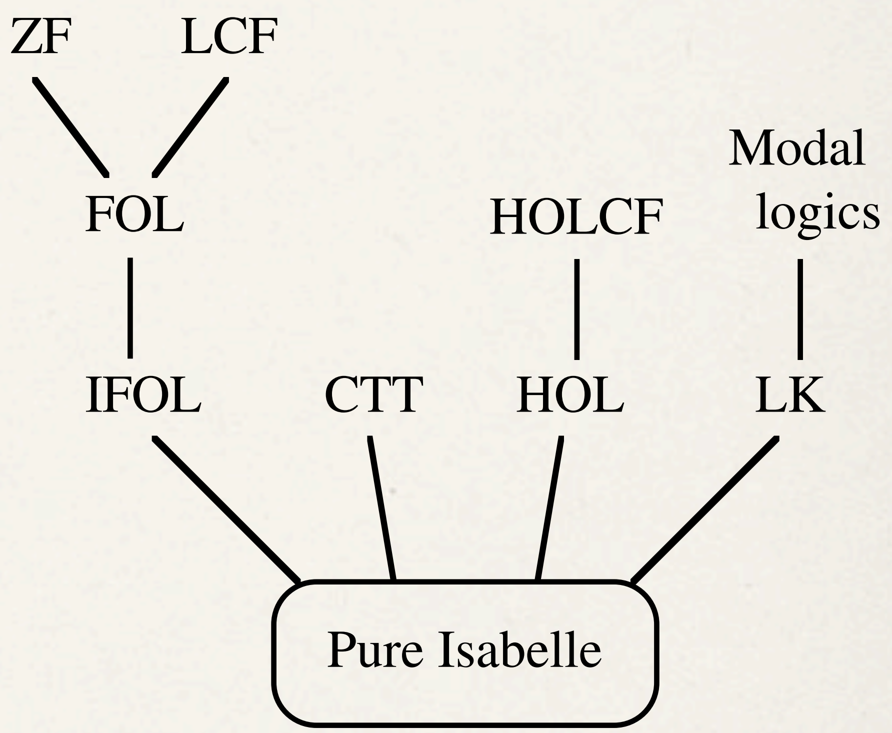
But the LCF Architecture…???

- It's still there! Only now, the abstract type of theorems encodes a *logical framework*.
- Which logical framework? *Intuitionistic higher-order logic*. No proof objects!
- Because Robin Milner said we don't need to store proofs.
- [And proofs still take up too much "working space", even though we have 10,000 times as much memory as in 1975!]

Combining LCF with a logical framework yields *Isabelle*.

One system, many logics! And…

- Support for *new logics*, including *embedded* logics, sharing infrastructure.
- *Logical variables* in subgoals. [With Huet's *higher-order unification* to join proofs.]
- … so *proof search* (Prolog-style) is easy to implement. And tactics have been generalised to return a *lazy list* of possible outcomes.



1989–2011

Supporting Higher-Order Logic

- Identifying HOL types with those of the logical framework
- Order-sorted polymorphism (Nipkow)
- Axiomatic type classes (Wenzel)
- Isabelle/HOL is the most popular Isabelle instance and receives most development…

It is even the basis for a formalisation of LCF!

Automatic Proof and Disproof

- The *classical reasoner*: generalised backtracking proof search both forward and backward chaining, available to all classical logics
- *Sledgehammer*: one-click delivery of the Isabelle proof state to a collection of automatic theorem provers
- Automatic *counterexample* finding: (1) Quickcheck and (2) Nitpick.

  1. for problems that are executable in a very general sense
  2. a separate, SAT-based tool for non-executable situations

A Few Applications

- seL4: the first machine proo

Archive of Formal Proofs

2012

2012-03-1: Abortable Linearizable Modules
Author: Rachid Guerraoui, Viktor Kuncak and Giuliano Losa

2012-02-29: Executable Transitive Closures
Author: René Thiemann

2012-02-06: A Probabilistic Proof of the Girth-Chromatic Number Theorem
Author: Lars Noschinski

2012-01-30: Dijkstra's Shortest Path Algorithm
Author: Benedikt Nordhoff and Peter Lammich

2012-01-30: Refinement for Monadic Programs
Author: Peter Lammich

2012-01-03: Markov Models
Author: Johannes Hölzl and Tobias Nipkow

2011

2011-11-19: A Definitional Encoding of TLA* in Isabelle/
Author: Gudmund Grov and Stephan Merz

2011-11-09: Efficient Mergesort
Author: Christian Sternagel

2011-09-22: Pseudo Hoops
Author: George Georgescu, Laurentiu Leustean and Vior

2011-09-22: Algebra of Monotonic Boolean Transformer
Author: Viorel Preoteasa

2011-09-22: Lattice Properties
Author: Viorel Preoteasa

2011-08-26: The Myhill-Nerode Theorem Based on Regular Expressions
Author: Chunhan Wu, Xingyuan Zhang and Christian Urban

2011-08-19: Gauss-Jordan Elimination for Matrices Represented as Functions
Author: Tobias Nipkow

2011-07-21: Maximum Cardinality Matching
Author: Christine Rizkallah

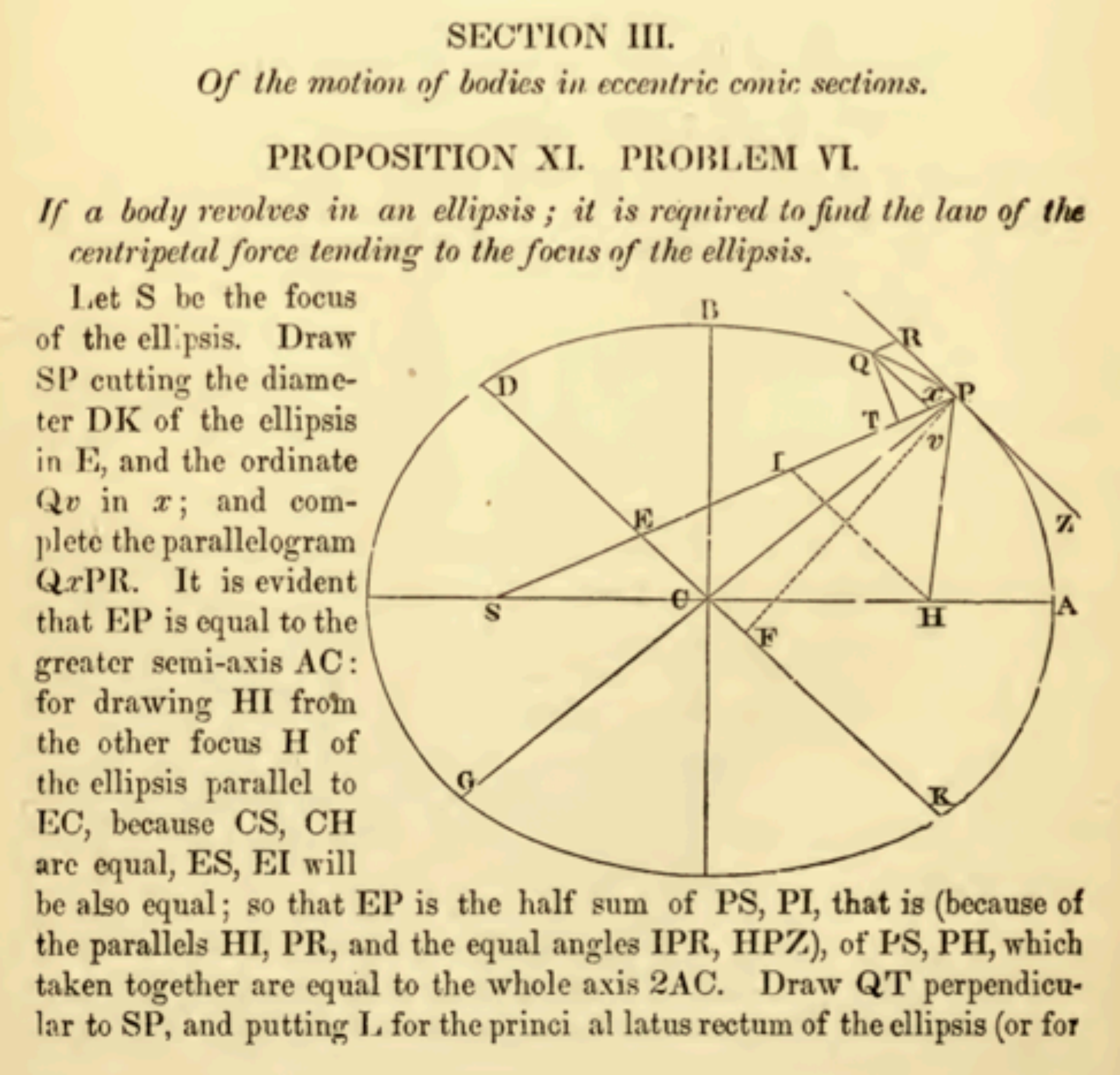
Formalising Mathematics

"To explore both the expressive and deductive power of a particular logic and the pragmatic problems which arise in conducting proofs in it"
— [Edinburgh LCF, page 1]

- Many people have formalised many, many mathematical results.
- Sometimes, these formalisations yield special insights…
  - Newton's *Principia* (formalised by Fleuriot)
  - Axiomatic set theory (K Grąbczewski, LCP)

Newton’s Non-Standard Geometry

- Newton’s treatise on the orbits of planets did *not* use calculus.
- His proofs used geometric arguments and infinitesimals.
- Here, he proves the inverse-square law for gravity.
- Can such proofs be formalised as they were written, within infinitesimal geometry?



SECTION III.

Of the motion of bodies in eccentric conic sections.

PROPOSITION XI. PROBLEM VI.

If a body revolves in an ellipsis; it is required to find the law of the centripetal force tending to the focus of the ellipsis.

Let S be the focus of the ellipsis. Draw SP cutting the diameter DK of the ellipsis in E, and the ordinate Qv in x; and complete the parallelogram QxPR. It is evident that EP is equal to the greater semi-axis AC: for drawing HI from the other focus H of the ellipsis parallel to EC, because CS, CH are equal, ES, EI will be also equal; so that EP is the half sum of PS, PI, that is (because of the parallels HI, PR, and the equal angles IPR, HPZ), of PS, PH, which taken together are equal to the whole axis 2AC. Draw QT perpendicular to SP, and putting L for the princi al latus rectum of the ellipsis (or for

The “Kepler Problem”

Formalised Infinitesimal Geometry (Fleuriot's PhD Work)

- defining non-standard analysis: the hyperreals, limits, continuity,...
- defining geometric concepts using the signed-area and full-angle methods
- formalising Newton's infinitesimal arguments *directly*
- Fleuriot found an error in Newton's proof of Proposition XI, but found an alternative route to the result.

Despite lacking a rigorous theory of infinitesimals,
Newton usually reasoned soundly with them.

Axiomatic Set Theory

$$a \in \{x \in A \,.\, \psi[x]\} \leftrightarrow a \in A \wedge \psi[a]$$

- It is "well known" that ZF set theory is not suitable for machine implementation because it requires infinitely many axioms.
- This belief doesn't reckon with the use of a logical framework with higher-order variables! (And yes, ψ remains a *first-order* formula.)
- But can we work effectively in this formalism, supposedly the foundation of mathematics? **Yes!**

We can address some of the most fundamental issues in logic.

Set Theory: Equivalents of AC

- K. Grąbczewski for
  Rubin's *Equivalents*
    - the equivalence
    - and 20 formulati
    - Lots of highly te

Case 1. $(\forall\beta)[\beta < \alpha$ and $f(\beta) \neq \emptyset \rightarrow (\exists\gamma)(\exists\delta)[\gamma, \delta < \alpha,$ $\mathscr{D}(u_{\beta\gamma\delta}) \neq \emptyset$ and $\mathscr{D}(u_{\beta\gamma\delta}) \prec m]]$.

For each $\beta < \alpha$ with $f(\beta) \neq \emptyset$, let λ_β and μ_β be the lexicographically $<$- first ordinal numbers γ and δ such that $\mathscr{D}(u_{\beta\gamma\delta}) \neq \emptyset$ and $\mathscr{D}(u_{\beta\gamma\delta}) \prec m$. (That is, first find ordinal numbers γ and δ which satisfy the conditions. Then let λ_β be the $<$- smallest such γ which satisfies the conditions. Then given λ_β, let μ_β be the $<$- smallest δ which satisfies the conditions.) Now define:

$$v_\beta = \begin{cases} \mathscr{D}(u_{\beta,\lambda_\beta,\mu_\beta}) & \text{if } f(\beta) \neq \emptyset \\ \emptyset & \text{if } f(\beta) = \emptyset \end{cases},$$

and $w_\beta = f(\beta) \sim v_\beta$. Next we define a function g as follows:

$\mathscr{D}(g) = \alpha + \alpha$,

if $\beta < \alpha$ then $g(\beta) = v_\beta$,

if $\alpha \leq \beta$, and $\beta \sim \alpha \cong \gamma < \alpha$ then $g(\beta) = w_\gamma$.

Set Theory: Reflection Theorem

$$\mathbf{M} = \bigcup_{\alpha \in \mathbf{ON}} M_\alpha$$

- relating truth of some ψ in the class **M** to its truth in certain sets M_α
  - *impossible to formalise* as a single statement in ZF set theory (because the proof depends upon the structure of ψ)
  - *meta-level reasoning is necessary*, but can be reduced to an induction over the structure of formulas
- This yields a repetitive tactic for proving any instances of the reflection theorem.

Set Theory: Gödel's Proof of the Relative Consistency of AC

- A technically difficult milestone in 20th century logic, addressing Hilbert's First Problem and introducing the "inner model method"
    - definition of the class **L** of "constructible sets" (these are the sets that can be defined by formulas and therefore must be present)
    - proof that the concept of "constructible set" is *absolute* across models of ZF set theory
    - proof that **L** is a model of set theory, including the axiom of choice
- Any contradiction in set theory + AC can be *effectively transformed* into a contradiction in set theory alone.

Absoluteness; Skolem's Paradox

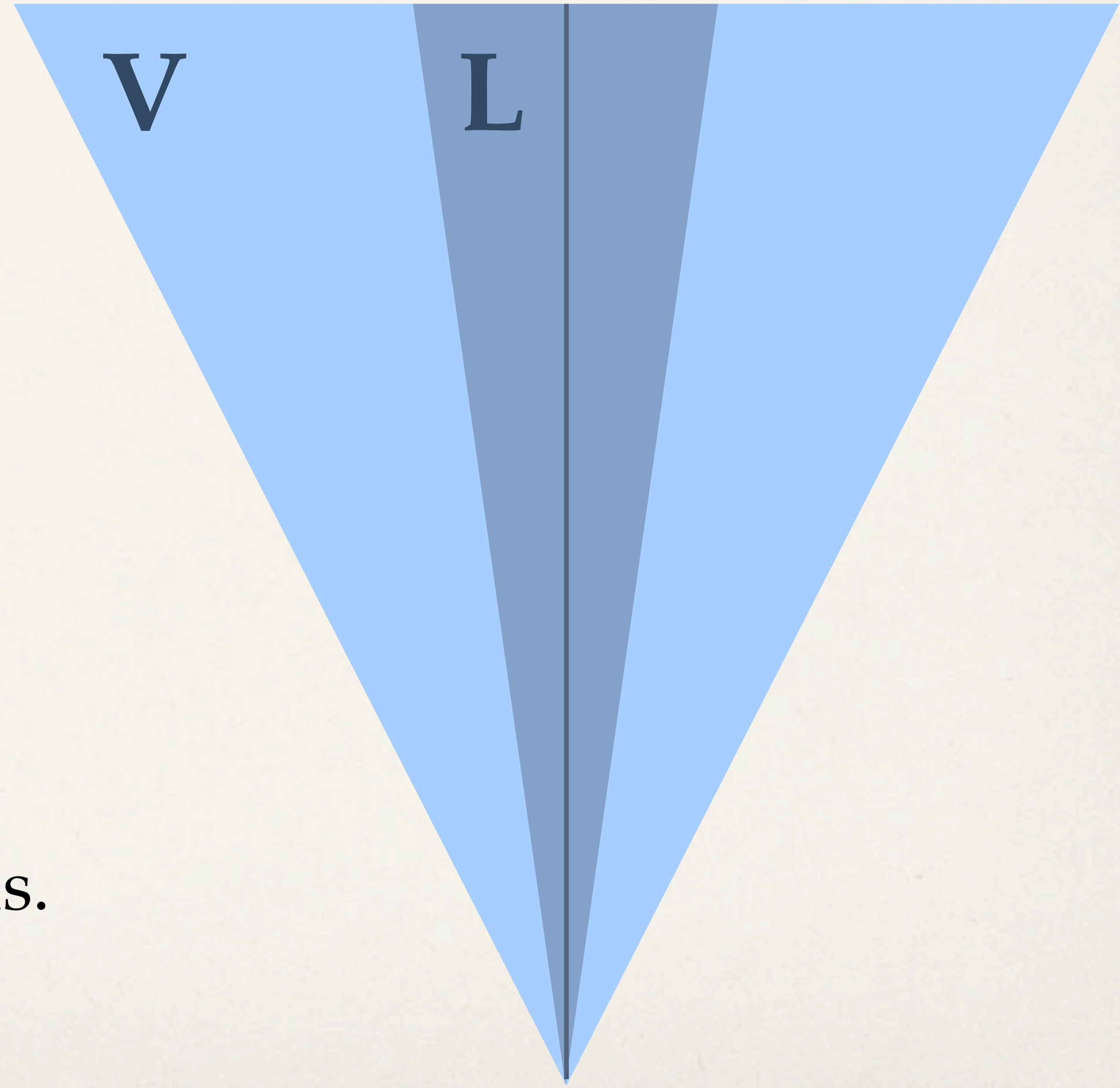
- If set theory is consistent, then (of course) it has models.
- It even has a *countable* model, *M*, by the Löwenheim-Skolem theorem!
- In *M*, all sets are countable, apparently violating Cantor's theorem. *How can this be??*
- Countability is *not absolute*: no function enumerating the elements of *M* is itself in *M*.
- Crucial to Gödel's consistency proof is that ...

CONSTRUCTIBILITY IS ABSOLUTE

- The proof requires a detailed analysis of the definition of constructibility.

Gödel's inner model method

- **V**, the class of all sets (the universe)
- **L**, the *constructible* sets
  - From within **L**, all sets are constructible
  - and the axiom of choice holds.



Gödel's Proof: Special Motivations

No formal theorem statement, just a series of suggestive results!

> "This clearly is a momentous achievement. Nevertheless, viewed 65 years later, the proof has very little flavor of a mathematical character. Rather, it is an achievement of definitions and of a point of view." — Paul Cohen

Can we formalise such a thing??

Gödel's Proof in Isabelle

- formalising sections of Kunen's textbook *Set Theory*
- a detailed formal definition of the concept of constructibility
- absoluteness proofs for constructibility, using no meta-theoretical reasoning; a proof that the axiom of choice holds in the class **L**
- a specific, finite list of axiom instances used in these proofs

eliciting some interest from philosophical logicians
(albeit none from computer scientists…)

2012

A Break with LCF: Oracles

- Theorems can be created by trusted external components (such as model checkers; also code generated for computational reflection)
- … but never in “normal” proofs (not even using sledgehammer)
- … and all such dependencies are tracked internally
- … and let’s not mention `mk_thm`

A Break with LCF: hiding ML

```
lemma inj_not_surj_succ:
  assumes fi: "f ∈ inj(A, succ(m))" and fns: "f ∉ surj(A, succ(m))"
  shows "∃f. f ∈ inj(A,m)"
proof -
  from fi [THEN inj_is_fun] fns
  obtain y where y: "y ∈ succ(m)" "⋀x. x∈A ⟹ f ` x ≠ y"
    by (auto simp add: surj_def)
  show ?thesis
    proof
      show "(λz∈A. if f`z = m then y else f`z) ∈ inj(A, m)" using y fi
        by (simp add: inj_def)
           (auto intro!: if_type [THEN lam_type] intro: Pi_type dest: apply_funtype)
      qed
qed
```

Structured proofs are much clearer!

Users can extend this language using ML.

Conclusion: Milner's LCF Architecture Still Stands

- an abstract type of theorems
- no proof objects (most of the time)
- a simple hierarchical theory structure
- a higher-order programming language

and of course: investigating unusual formalisms
is still good science