ai-reasoning.net

Autoformalization & Mathematical Reasoning

Lan Zhang, Marco Valentino, Jordan Meadows, André Freitas

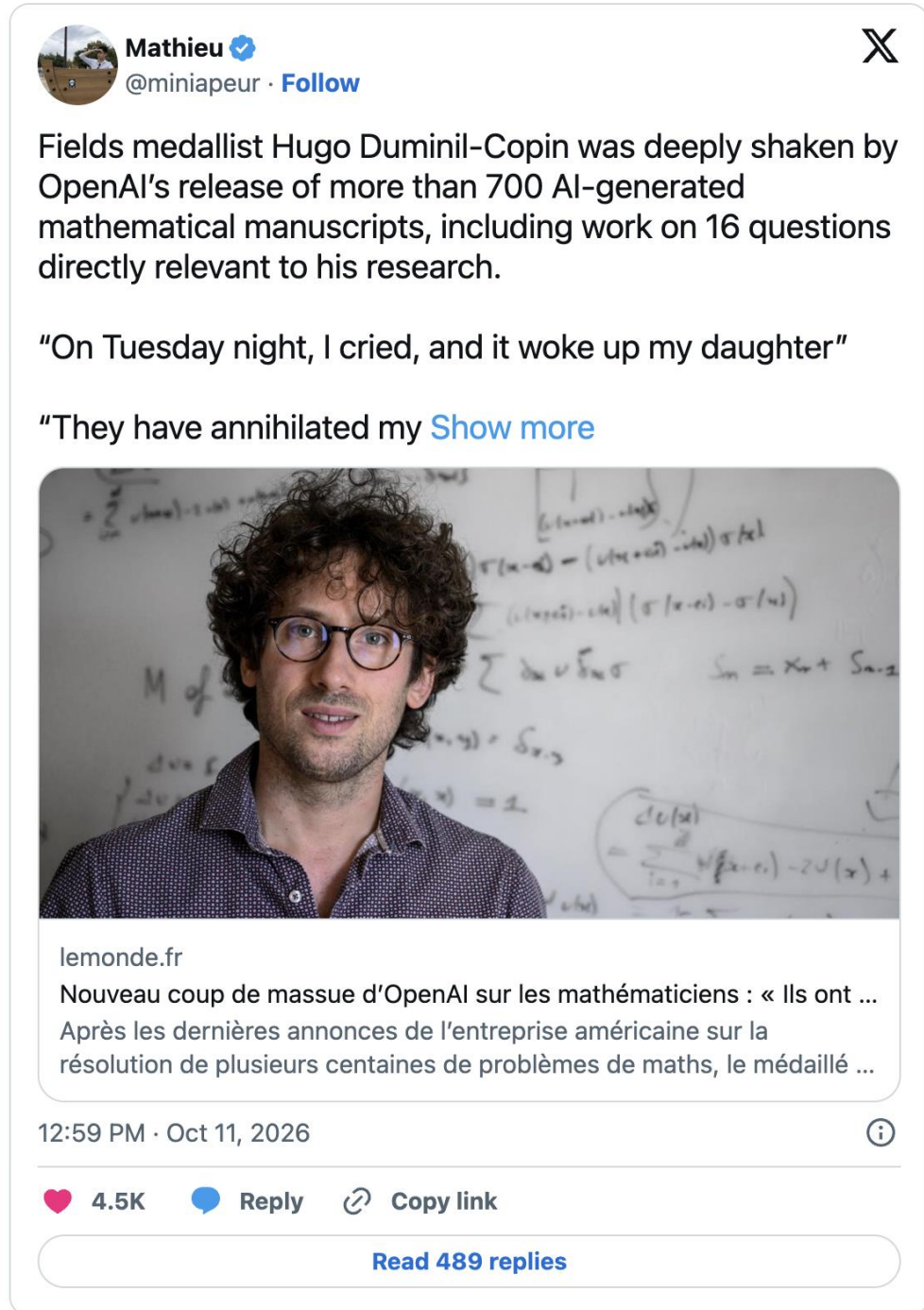
October 2026

EMNLP2026 BUDAPEST



Is AI shaping a new type of Mathematics?

OpenAI says it cracked 90-year-old maths problem in 88 hours



Timeline

1 August

- Ten mathematical and theoretical-computer-science results, with Lean proofs.

8 September

- Navier–Stokes announcement, using a large multi-agent campaign.

6-7 October

- Broad manuscript collection, followed by withdrawals, corrections and additional formalizations.

15 October

- Announcement of $5 million for a Simons Foundation initiative supporting mathematical understanding.

26 October

- Two new manuscripts: local smoothing and the Exponential Time Hypothesis

https://github.com/openai/math

- 721 manuscripts across 373 problem families.



- 3 withdrawals. Formalization approximately 42%

300 / 719

Top-level results formalized as reported on 7 October.

Estimated scale

Parameter	Navier–Stokes
Concurrent agents	Order of 10,000 in successful group
Discovery elapsed time	About 88 hours from first launch
Formalization / verification	Additional 17 hours, GPT-6 Astra
Agent messages	2.7 million
Output tokens	About 130 billion

Today

- Understand the principles of mathematical reasoning
  - Linguistics and inference
- Exemplify contemporary methods for AI-based mathematical reasoning.
  - Neuro-symbolic perspective
  - Architectural & evaluation paradigms
- Application areas
  - AI4Science
  - Controlled reasoning

Questions and comments at anytime please.



What is a mathematical proof?

What is a mathematical proof?

Truth

What makes the statement true?

Compare accounts based on reality, coherence and construction.

Justification

Why are we entitled to believe it?

An argument or reliable method provides grounds for belief.

Proof

What establishes the conclusion?

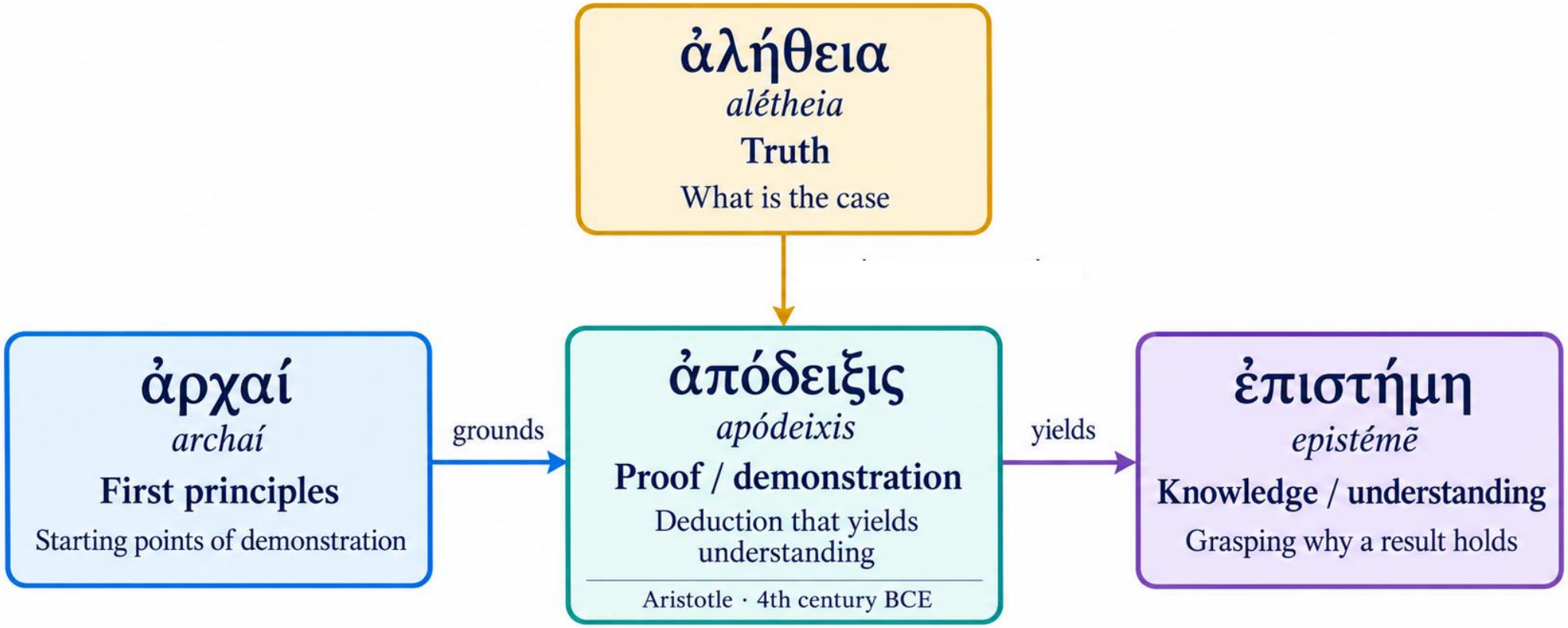
A construction or derivation meets the relevant standards.

Explanation

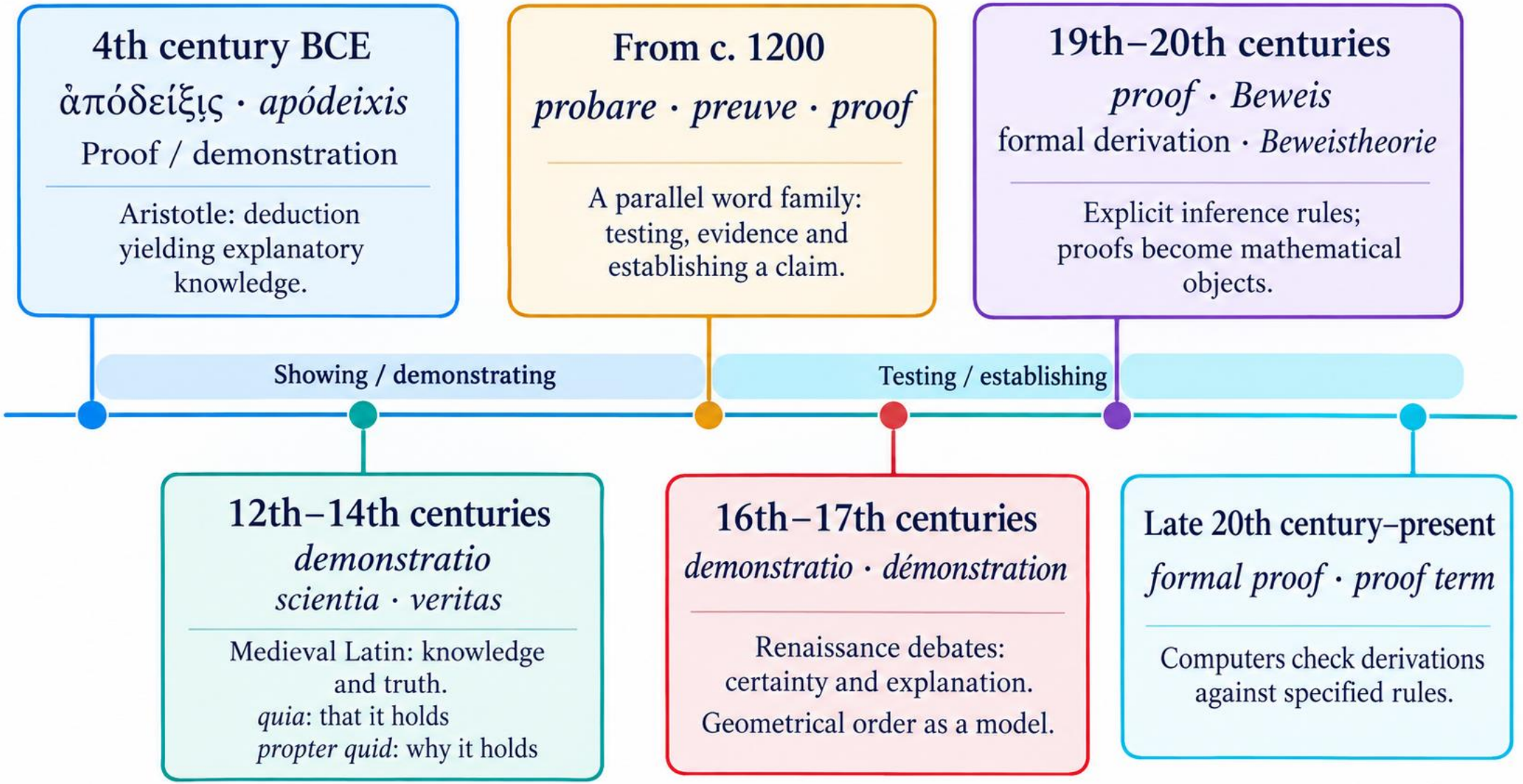
Why does the result hold?

The argument reveals the relation that makes it necessary.

Conceptualizations over mathematical proofs

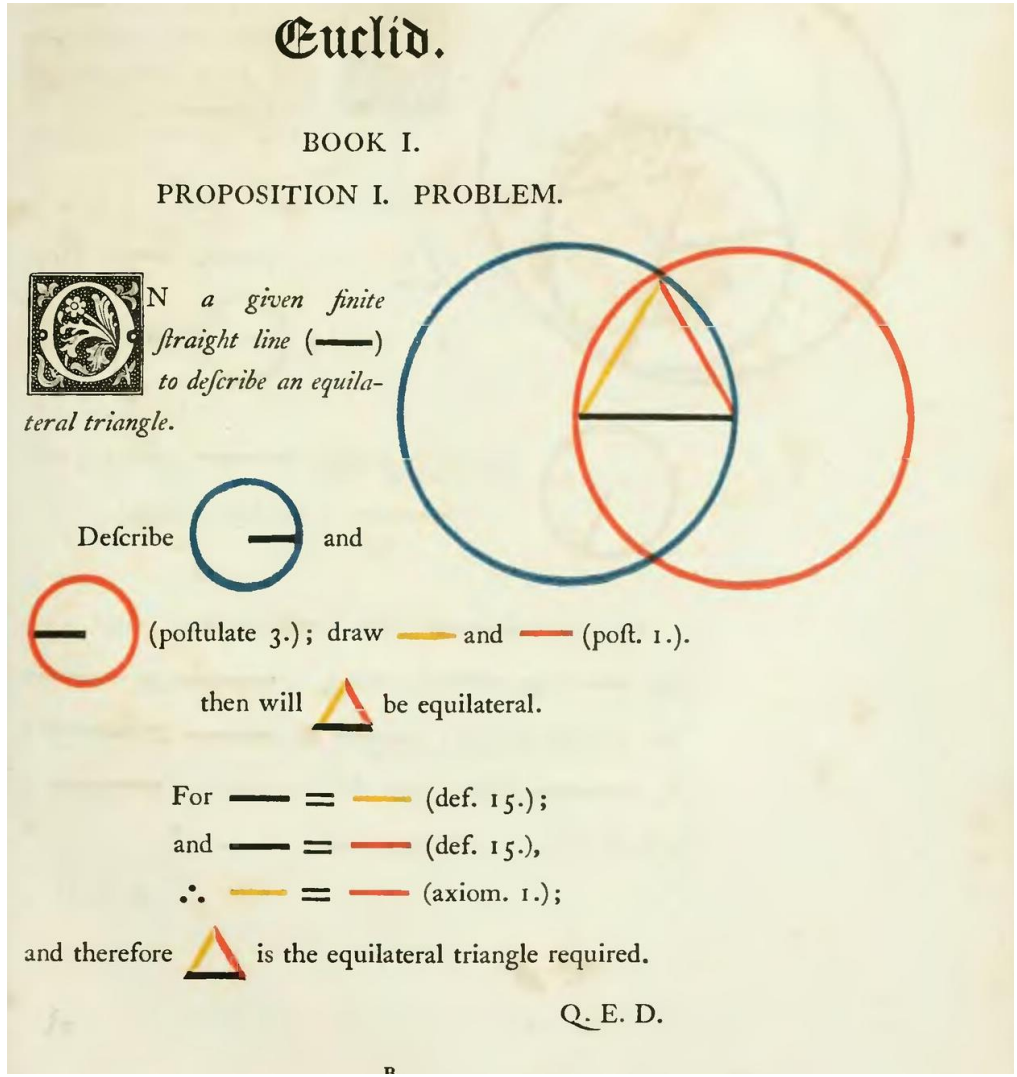


Conceptualizations over mathematical proofs



c. 300 BCE

Euclid: Elements I.1



Euclid.

BOOK I.

PROPOSITION I. PROBLEM.

ON *a given finite ſtraight line* (—) *to deſcribe an equilateral triangle.*

Deſcribe — and — (poſtulate 3.); draw — and — (poſt. 1.).

then will △ be equilateral.

For — = — (def. 15.);

and — = — (def. 15.),

∴ — = — (axiom. 1.);

and therefore △ is the equilateral triangle required.

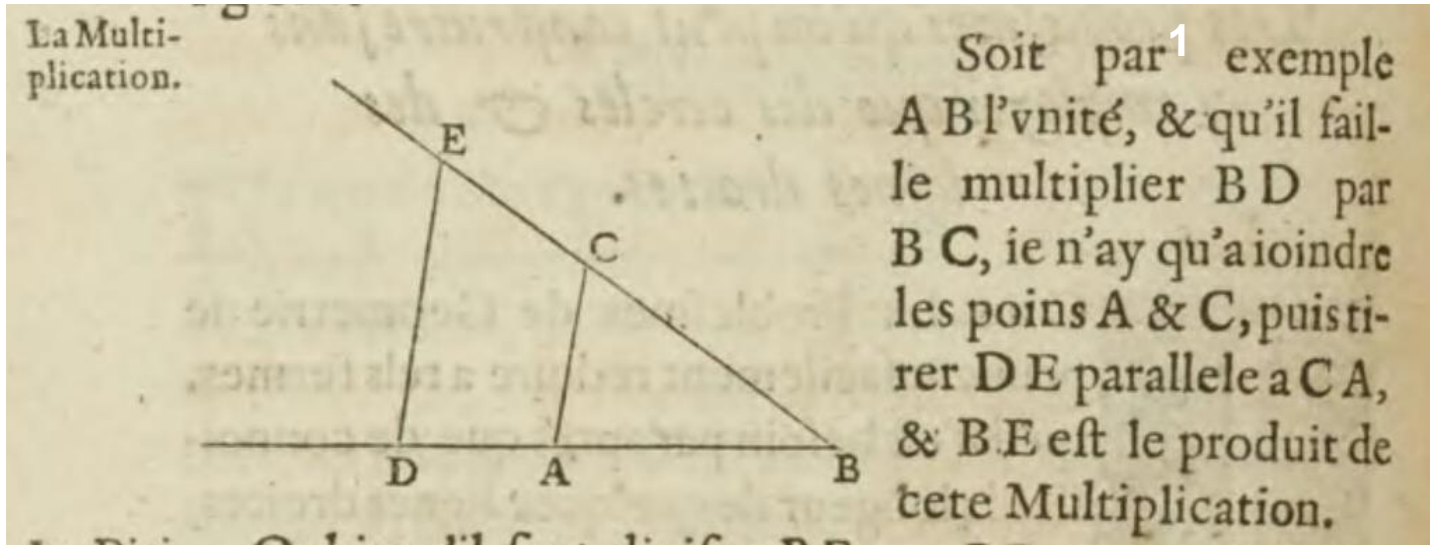
Q. E. D.

A diagram guides the argument with definitions and common notions justifying it.

Oliver Byrne's 1847 English edition · Proposition I.1

1637

Descartes · La Géométrie


La Multi-
plication.

Soit par[1] exemple A B l'vnité, & qu'il faille multiplier B D par B C, ie n'ay qu'a ioindre les poins A & C, puis tirer D E parallele a C A, & B E eſt le produit de tete Multiplication.

$AB = 1 \quad DE \parallel AC$

$BE / BC = BD / BA$

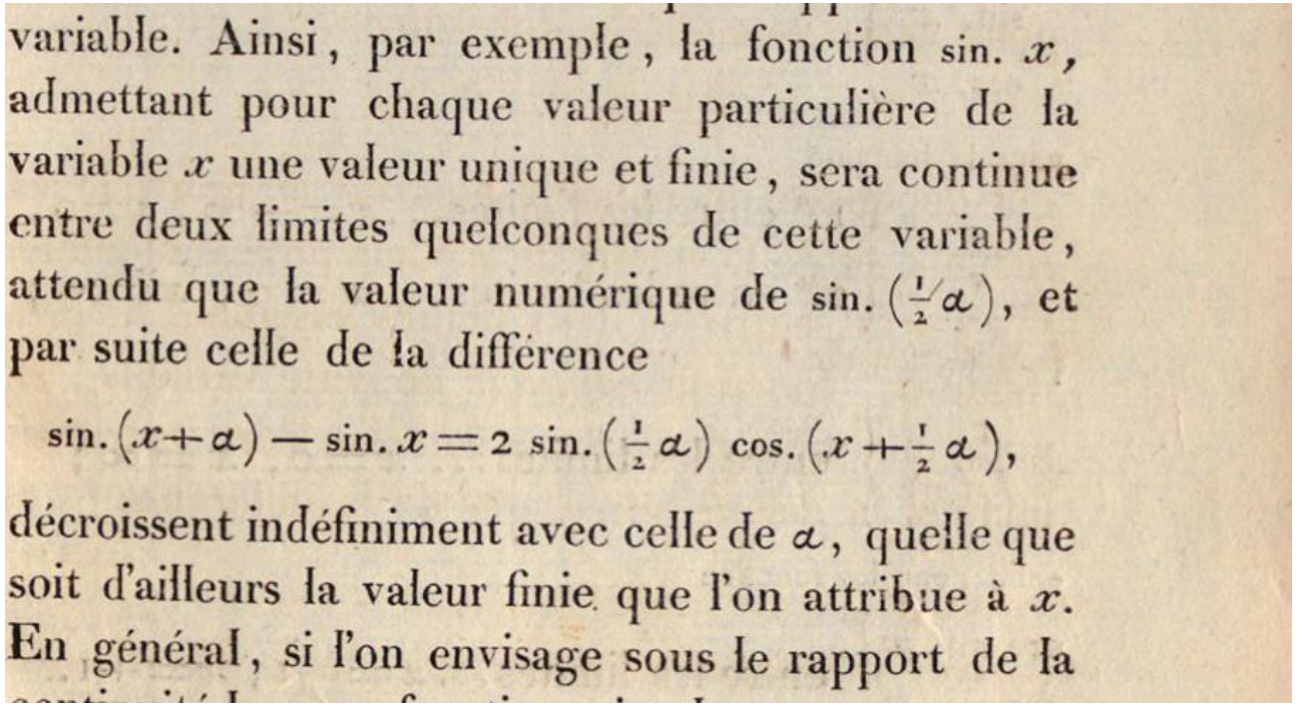
$BE = BD \times BC$

Connects geometrical and algebraic equivalences

Original French edition · p. 298 · BnF / Gallica

1821

Cauchy: Cours d'analyse



variable. Ainsi, par exemple, la fonction sin. x, admettant pour chaque valeur particulière de la variable x une valeur unique et finie, sera continue entre deux limites quelconques de cette variable, attendu que la valeur numérique de sin. $(\frac{1}{2}\alpha)$, et par suite celle de la différence

$$\text{sin.}(x+\alpha) - \text{sin.}\, x = 2\ \text{sin.}\left(\tfrac{1}{2}\alpha\right)\ \text{cos.}\left(x+\tfrac{1}{2}\alpha\right),$$

décroissent indéfiniment avec celle de α, quelle que soit d'ailleurs la valeur finie que l'on attribue à x. En général, si l'on envisage sous le rapport de la

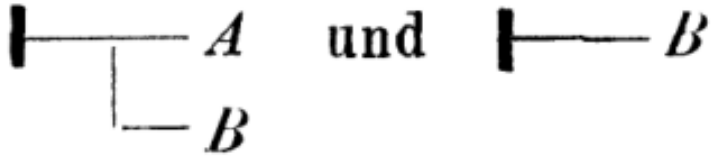
Mature algebraic notation, additional concepts added (continuity)

Original French edition · p. 35 · ETH Zürich / e-rara

1879

Frege: Begriffsschrift

§ 6. Aus der in § 5 gegebenen Erklärung geht hervor, dass aus den beiden Urtheilen



und

das neue Urtheil



The logical form of the inference step becomes an object of exact notation.

Original German edition · §6, p. 7 · SUB Göttingen

1931

Gödel: On formally undecidable propositions

$$B w_\varkappa\ (x) \equiv (n)\,[n \leqq l\,(x) \rightarrow A\,x\ (n\ Gl\,x) \vee (n\ Gl\,x)\,\varepsilon\,\varkappa \vee \tag{5}$$

$$(Ep, q)\ \{0 < p, q < n\ \&\ Fl\ (n\ Gl\ x, p\ Gl\,x, q\ Gl\,x)\}\,]\ \&\ l\,(x) > 0$$

(vgl. den analogen Begriff 44)

$$x\,B_\varkappa\,y \equiv B w_\varkappa\ (x)\ \&\ [l\,(x)]\ Gl\ x = y \tag{6}$$

$$\mathrm{Bew}_\varkappa\ (x) \equiv (E\,y)\ y\ B_\varkappa\,x \tag{6·1}$$

(vgl. die analogen Begriffe 45, 46).

Es gilt offenbar:

$$(x)\ [\mathrm{Bew}_\varkappa\ (x) \sim x\,\varepsilon\,\mathrm{Flg}\ (\varkappa)] \tag{7}$$

$$(x)\ [\mathrm{Bew}\ (x) \rightarrow \mathrm{Bew}_\varkappa\ (x)] \tag{8}$$

Proof itself can be studied mathematically, and formal systems have limits.

Original German paper · p. 187, definitions (5) - (8)

1984

Martin-Löf: Intuitionistic Type Theory

$$(\lambda z)\,\mathsf{p}(z) \in (\Sigma x \in A)\,B(x) \to A.$$

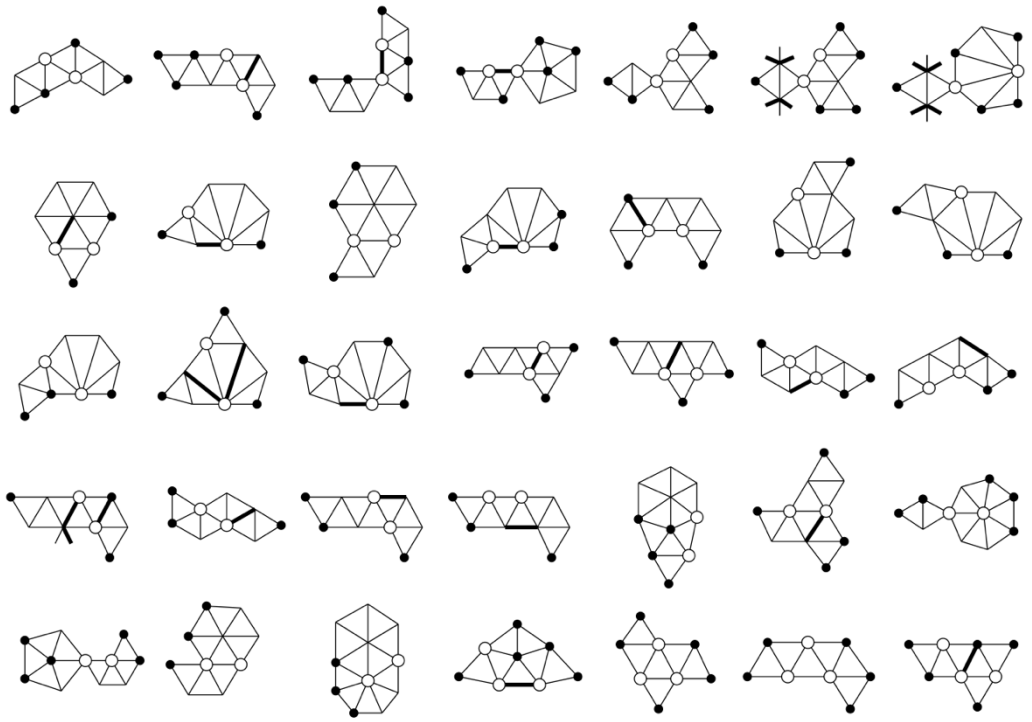
In particular, when $B(x)$ does not depend on x, we obtain

$$A \,\&\, B \supset A \ \mathit{true}.$$

A proof can be a construction with computational content.

1976 → 1997

Robertson, Sanders, Seymour & Thomas



A smallest counterexample must contain a listed configuration but cannot contain any of them.

1 Reduce an infinite claim

Assume a smallest planar graph that cannot be coloured with four colours.

2 Make cases unavoidable

Every such counterexample must contain one of 633 configurations in this proof.

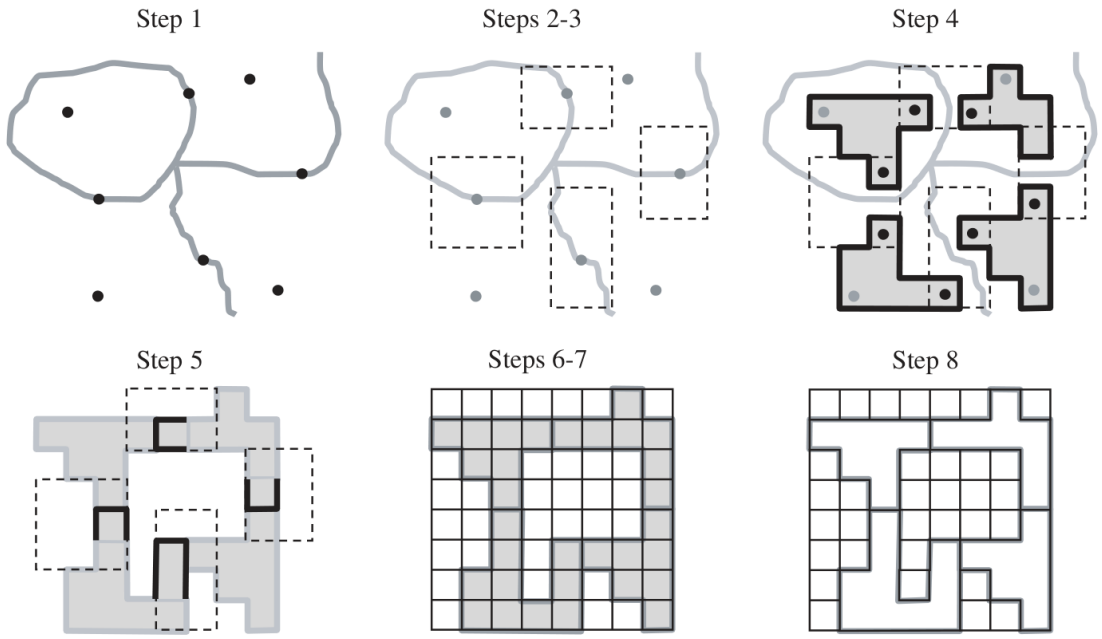
3 Check reducibility

Computer checks establish that each configuration permits reduction to a smaller case. Contradiction.

A mathematical reduction makes a finite computation decisive.

2005 → 2008

Gonthier: Formal Proof - The Four-Color Theorem




```
Theorem four_color :
  (m : map R),
  simple_map m → map_colorable 4 m.
```

The same theorem now comes with a machine-checked formal derivation.

Author's exposition, 2008 · p. 1388, Figure 3 · Notices AMS

Proof perspective

Geometric deduction

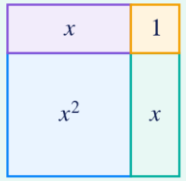


Extending a length preserves the original comparison.

c. 300 BCE

1

Geometry supports algebra



$x>0$

$$(x+1)^2=x^2+2x+1$$

9th–12th centuries

2

Symbolic calculation

$$a,b\in\mathbb{R},\quad a\le b$$

$$(b+1)-a=(b-a)+1$$

$$\ge 0 \implies a\le b+1$$

Transform expressions by algebraic rules.

17th–18th centuries

3

Precise limits

$$a_n\le b_n,\quad a_n\to a,\quad b_n\to b$$

$$a>b\ ?\quad \varepsilon=(a-b)/3$$

$$a_n>a-\varepsilon>b+\varepsilon>b_n$$

Contradiction for large n; hence $a\le b$.

19th century

4

Formal deduction

$$\frac{a\le b \qquad b\le b+1}{a\le b+1}$$

Transitivity

Explicit premises and inference rules.

Late 19th–20th centuries

5

Constructive evidence

$$b=a+k,\quad k\in\mathbb{N}$$

$$b+1=a+(k+1)$$

$$k\longmapsto k+1$$

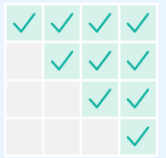
A proof transforms evidence into evidence.

20th century

6

Exhaustive computation

$$0\le a\le b\le 3:\quad a\le b+1$$



10 pairs
10 checks

All cases of a stated finite claim.

Late 20th century

7

Formal checking + AI

$$h : a\le b \ \vdash\ a\le b+1$$

```
exact le_trans h
  (Nat.le_succ b)
```

Human or AI proposes → kernel checks

People check the intended meaning.

Present

8

Notational perspective

Ancient traditions

1

Verbal quantity

“Three times a quantity,
plus two, equals eleven.”

An unknown, expressed in words

Greek geometry

2

Names for objects

A ——— B ——— C

AB + BC = AC

Letters identify geometric objects

3rd century CE · Diophantus

3

Abbreviated expressions

quantity → x

square → x^2

Recurring expressions get short signs

1591 · Viète

4

Unknowns + parameters

3x + 2 = 11

↓

ax + b = c

Given quantities become symbolic too

1637 · Descartes

5

Symbolic manipulation

ax + b = c

x = (c − b) / a

General transformations; a ≠ 0

1734 · Euler

6

Functional dependence

f(x) = 3x + 2

f(3) = 11

Variables connect inputs to outputs

1879 · Frege

7

Quantification + scope

∀x ∈ ℝ ∃y ∈ ℝ

y = x + 2

The choice of y may depend on x

20th–21st centuries

8

Typed formal languages

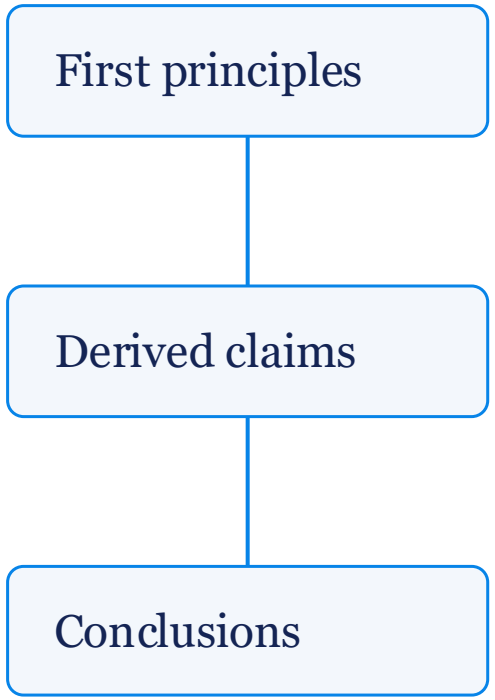
(n : ℕ) : n ≤ n + 1

Nat.le_succ n

Explicit types and checkable inference

Foundationalism

Some beliefs have basic warrant.

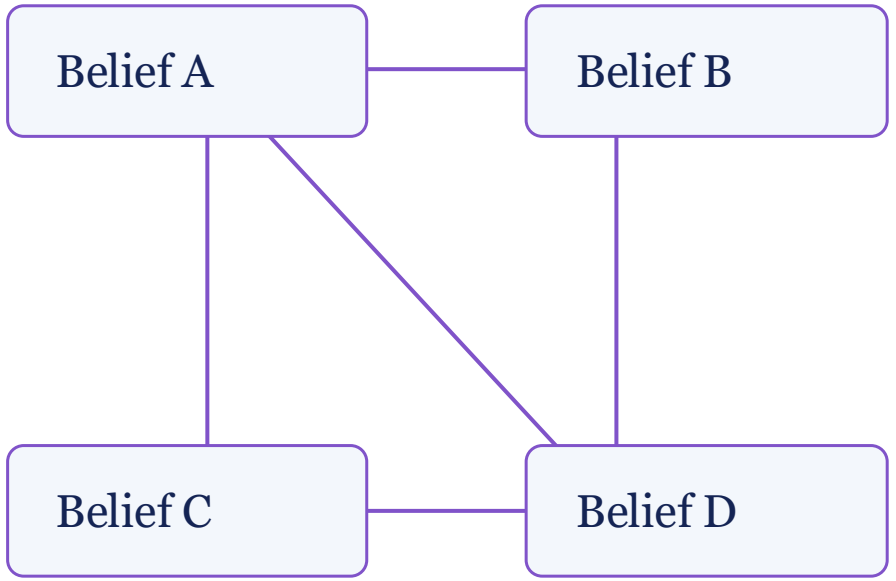


What do we agree on?

Where does inferential support begin?

Coherentism

Beliefs gain support through their relations.



Do the parts reinforce or contradict each other?

Within a theory

Which axioms and rules license this conclusion?

Inspect the derivation.

Check the construction.

Verify each dependency.

When evaluating the theory

Why choose these axioms and this interpretation?

Consider explanatory power.

Compare connections and applications.

Examine coherence and alternatives.

An axiomatic proof and a broader network of justification can coexist.

Proof objects

- Mathematical argument
- In order to establish a truth assessment
- NL + mathematical notation
- Conceptualization and notation evolves in time
  - Facilitate computation
- Explicit assumptions, definitions
- Constructed in a way to elicit relations between mathematical objects
- Normatively organized
  - Wrt representation and inference operations
- Dialogues with background knowledge
  - Foundationally & coherently
- Possibly with a formal representation
- Which can be mapped to a computational framework



Language & Reasoning

C_0: “Taxation is theft.”

π_0 Initial premise

Taxation requires payment without an individual opt-out.

Theft also involves taking another’s property.

δ_0 Implicit definition

Any taking of property without individual consent counts as theft.

$\pi_0 + \delta_0 \vdash C_0$ The inference depends on the proposed definition.

δ_0 becomes more precise

Δ Refined definitions

δ_1 Theft: wrongful appropriation of another's property.

δ_2 Coercion: compliance through force or threatened sanctions.

δ_3 Theft vs coercion distinguished

Coercion alone does not establish theft.

$\pi_0 + \Delta \nvdash C_0$ Lack of consent alone no longer establishes theft.

π_0: Compulsion becomes explicit

π_0 Individual choice

A person cannot opt out of a tax obligation merely by disagreeing.

Noncompliance can trigger sanctions.

δ_4, δ_5 Distinct forms of agreement

δ_4 Individual consent: a person's agreement.

δ_5 Collective authorization: a community's procedure for binding rules.

C_1: A compulsory transfer can have collective authorization.

π_1: The institutional context

π_1 Assumed authorization

Consider a tax enacted through a legal, democratic process.

Representatives decide under public rules, with accountability and routes for challenge.

Γ_1 Context expanded

$\Gamma_1 = \Delta \cup \{\pi_0, \pi_1\}$

Authority and procedure become relevant. Legal authorization alone does not settle moral legitimacy.

C_2: The question becomes whether this compulsory transfer is justified.

π_2: Common benefits enter the argument

π_2 Assumed public benefit

Contributions help fund shared infrastructure and services.

Residents can benefit through access to roads, sanitation and public institutions.

Γ_2 Context expanded

$\Gamma_2 = \Gamma_1 \cup \{\pi_2\}$

Does provision deliver the benefit?

Who benefits, who pays, and what alternatives exist?

C_3: Assess compulsion together with public benefits and burdens.

Σ: Value commitments become explicit

σ_0 Principle of minimization

Compulsory burdens require justification.

Among policies meeting the stated public commitments and fairness conditions, prefer the lower tax burden.

σ_1, σ_2 Commitments retained

σ_1 Sustain justified public provision.

σ_2 Respect rights, fair burdens and democratic accountability.

$\Sigma = \{\sigma_0, \sigma_1, \sigma_2\}$ The policy conclusion needs these value commitments.

Γ: “Minimize” acquires conditions

T(p) Objective

Minimize T(p) over p ∈ $\mathcal{F}$(Γ).

T(p) is the compulsory tax burden under policy p.

$\mathcal{F}$(Γ) Feasible policies

Adequate funding for agreed provision.

Authorization, rights and fair burdens.

Efficiency and other revenues assessed.

Γ = Δ ∪ Π ∪ Σ, with Π = {π_0, π_1, π_2}. These premises constrain the choice.

C*: Revisiting the conclusion

C_0 Original judgment

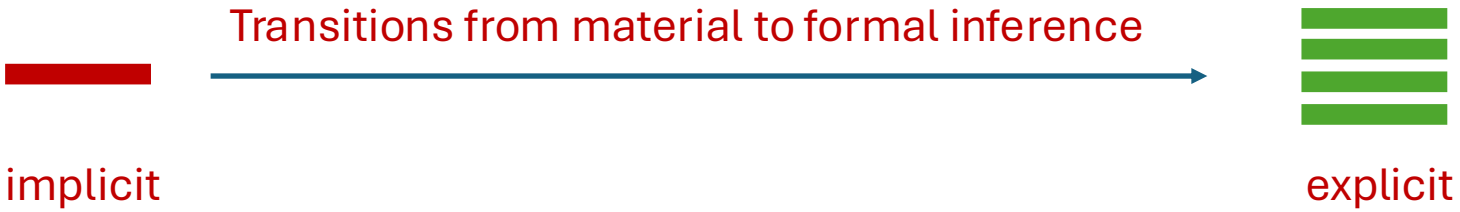
"Taxation is theft."

A single shared feature carried the argument.

Γ Refined basis

Δ Refined definitions
Π Compulsion, authorization, benefits
Σ Minimization, provision, fairness

Γ ⊢ C*: Under these commitments, minimize taxation while sustaining justified, democratically agreed public provision under fair and efficient arrangements.



C_0: “Taxation is theft.”

δ_0: Any taking without individual consent counts as theft.

δ_0 is replaced by Δ
$\pi_0 + \Delta \nvdash C_0$

Δ Definitions

δ_1 Theft

Wrongful appropriation of another’s property.

δ_2 Coercion

Compliance through force or threatened sanctions.

δ_3 Violence

Physical force. Theft need not involve violence.

δ_4 Individual consent

A person’s agreement.

δ_5 Collective authorization

A procedure for establishing binding public rules.

Π Factual premises

π_0 Compulsion

The tax obligation has no individual opt-out and is backed by sanctions.

π_1 Authorization

In this example, taxation follows a legal, democratic process.

π_2 Public benefit

Taxation funds shared infrastructure and services that provide public benefit.

Σ Value commitments

σ_0 Minimize the tax burden

Among policies meeting the commitments below, prefer the lower compulsory tax burden.

σ_1 Sustain public provision

Maintain justified, democratically agreed public provision.

σ_2 Respect fair arrangements

Protect rights, fair burdens and democratic accountability.

$\Gamma = \Delta \cup \Pi \cup \Sigma$

$\Delta = \{\delta_1, \ldots, \delta_5\}$ $\Pi = \{\pi_0, \pi_1, \pi_2\}$ $\Sigma = \{\sigma_0, \sigma_1, \sigma_2\}$

$\Phi(\Gamma)$: policies satisfying the public commitments, with efficiency and other revenues assessed.

$T(p)$: compulsory tax burden under policy p. Objective: minimize $T(p)$ over $p \in \Phi(\Gamma)$.

$\Gamma \vdash C^*$

Under these premises and commitments, minimize the compulsory tax burden while sustaining justified, democratically agreed public provision under fair arrangements.

Transformers, LLMs & Reasoning

Output: instability

Semantic & discourse relations

Material (content-based) inferences

Model

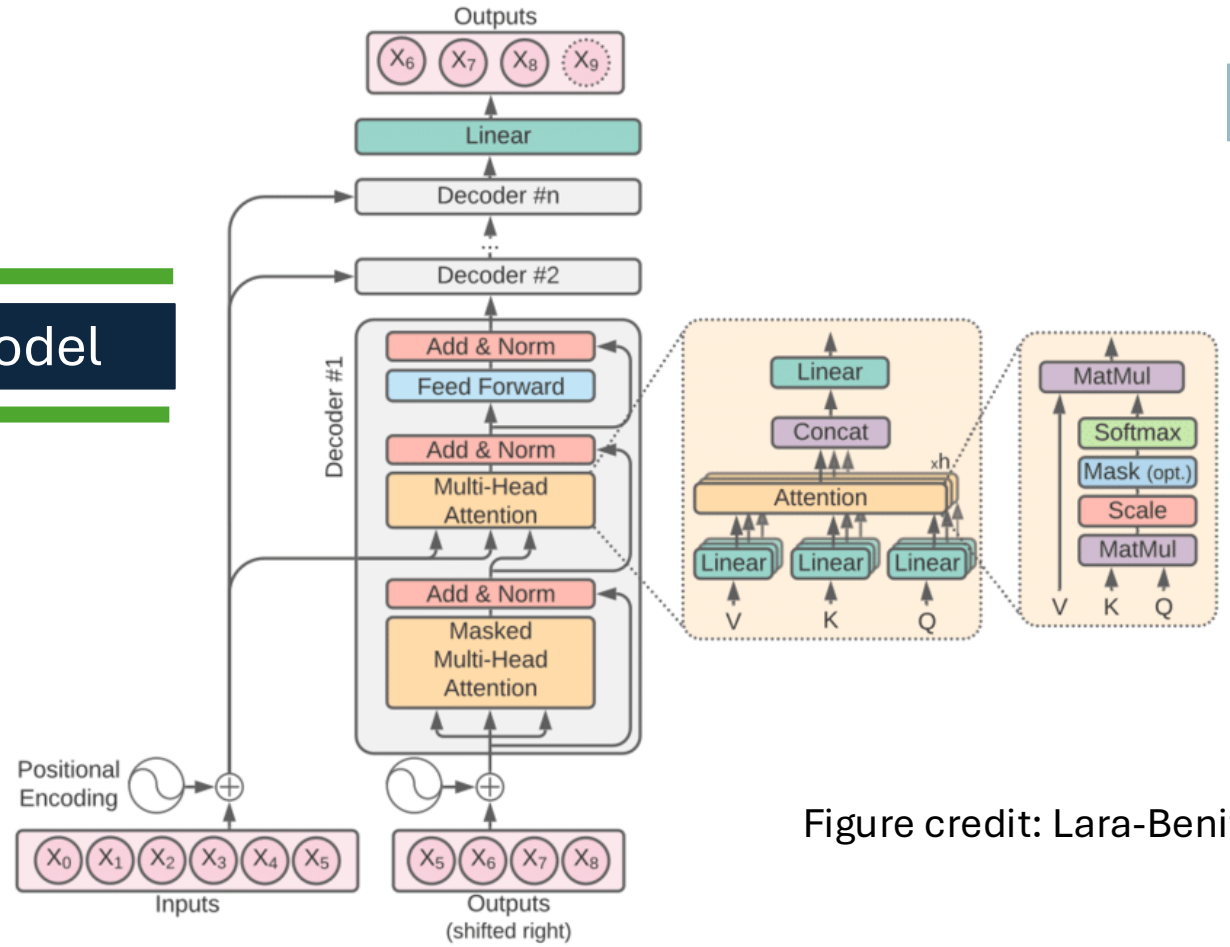


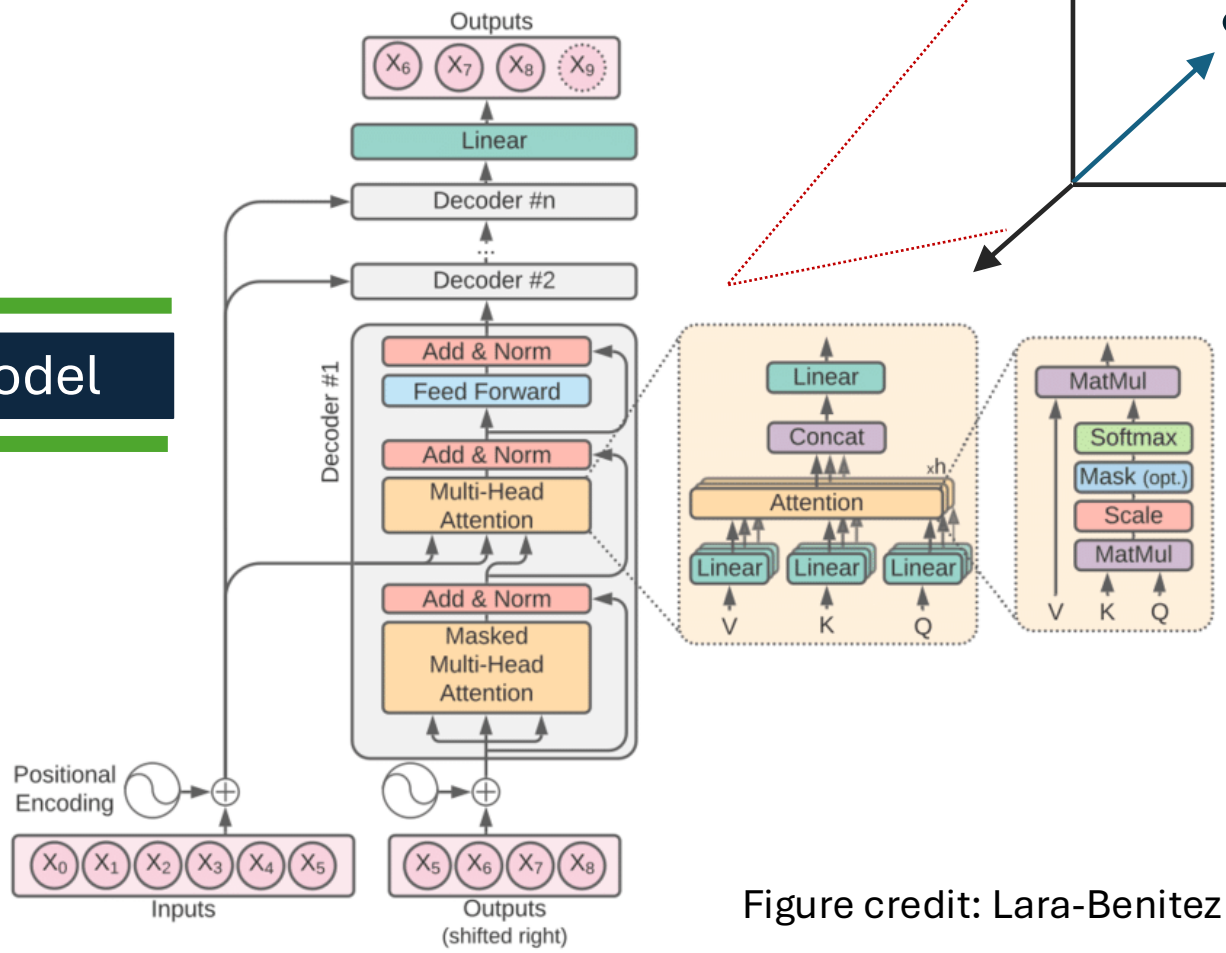
Figure credit: Lara-Benitez et al., 2020

Input: Loss of BRCA2 may cause increased genomic ...

Transformers, LLMs & Reasoning

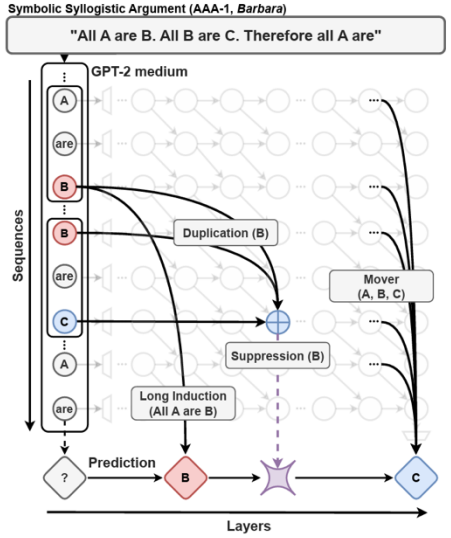
Model

Output: instability



Underlying geometry

and circuits



Specialized and expressive latent semantic subspaces specialized and expressive (multi-head attention).

Attention circuits that coordinate (mostly material) inferences.

Figure credit: Lara-Benitez et al., 2020

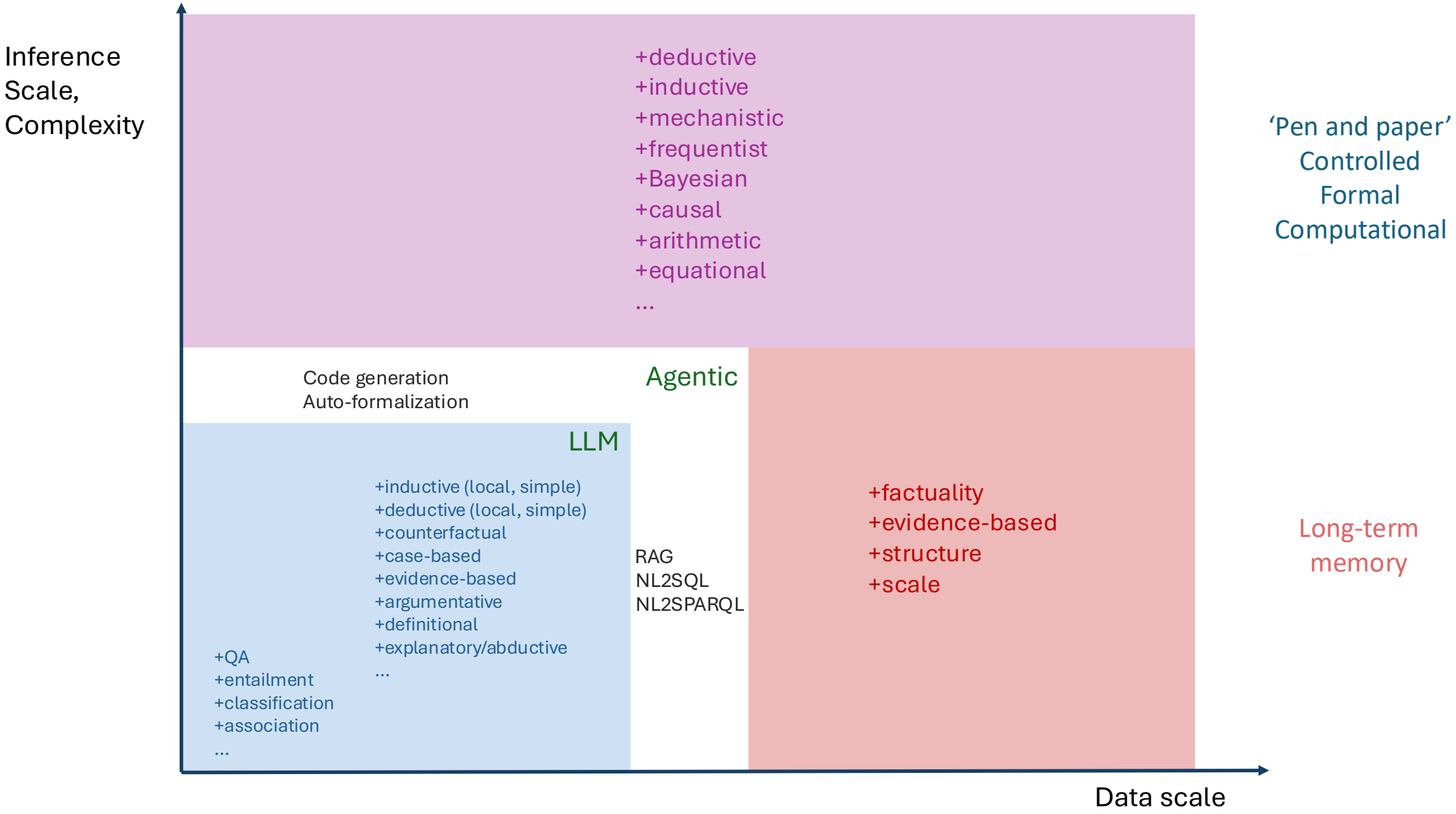
Input: Loss of BRCA2 may cause increased genomic ...

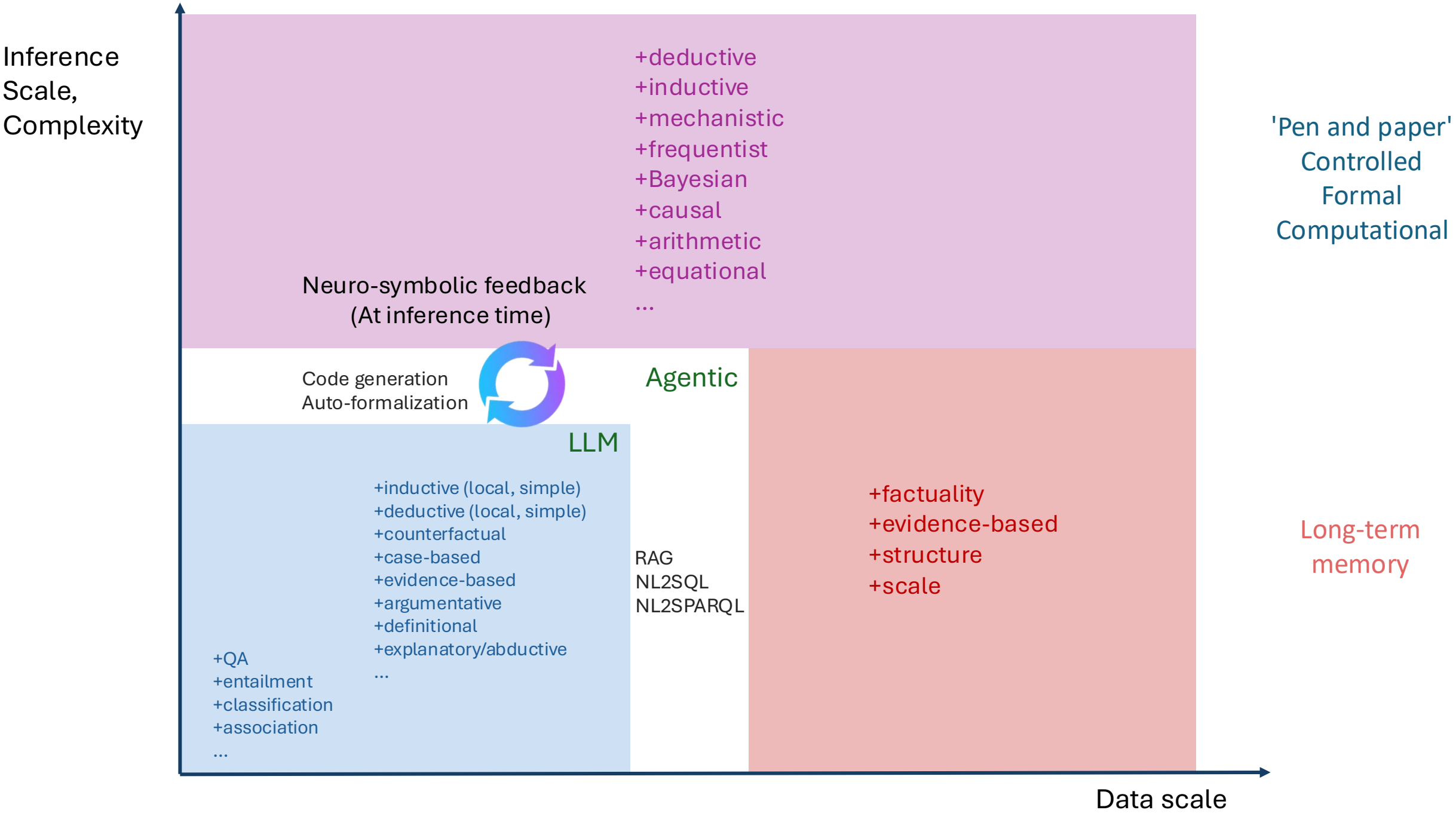
The spectrum of reasoning

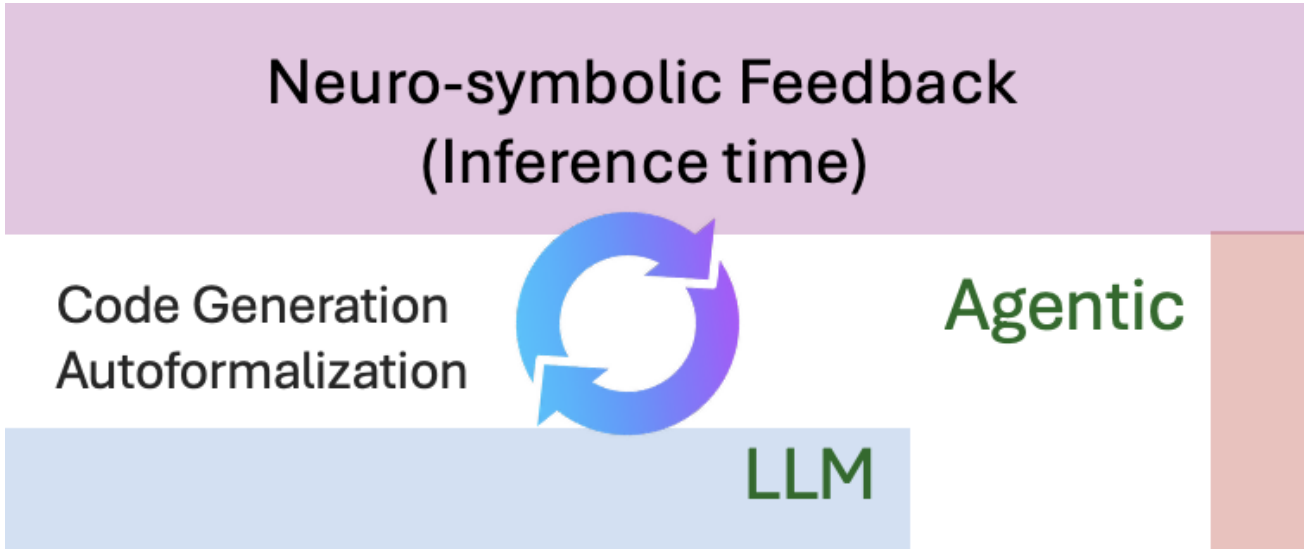


Type

Explanatory/Abductive
Definitional
Argumentative
Evidence-based
Case-based
Counterfactual
Deductive
Inductive
Mechanistic
Frequentist
Bayesian
Causal
Arithmetic
…


Inference
Scale,
Complexity
+deductive
+inductive
+mechanistic
+frequentist
+Bayesian
+causal
+arithmetic
+equational
…
‘Pen and paper’
Controlled
Formal
Computational
Code generation
Auto-formalization
Agentic
LLM
+inductive (local, simple)
+deductive (local, simple)
+counterfactual
+case-based
+evidence-based
+argumentative
+definitional
+explanatory/abductive
…
RAG
NL2SQL
NL2SPARQL
+factuality
+evidence-based
+structure
+scale
Long-term
memory
+QA
+entailment
+classification
+association
…
Data scale

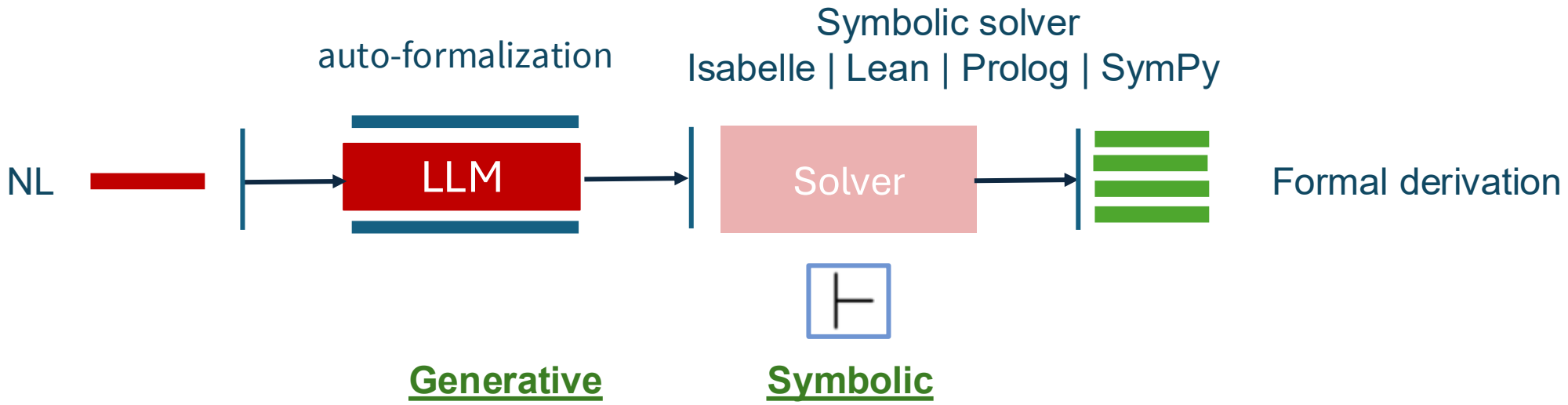

Inference Scale, Complexity
+deductive
+inductive
+mechanistic
+frequentist
+Bayesian
+causal
+arithmetic
+equational
...
'Pen and paper'
Controlled
Formal
Computational
Neuro-symbolic feedback
(At inference time)
Code generation
Auto-formalization
Agentic
LLM
+inductive (local, simple)
+deductive (local, simple)
+counterfactual
+case-based
+evidence-based
+argumentative
+definitional
+explanatory/abductive
...
RAG
NL2SQL
NL2SPARQL
+factuality
+evidence-based
+structure
+scale
Long-term
memory
+QA
+entailment
+classification
+association
...
Data scale

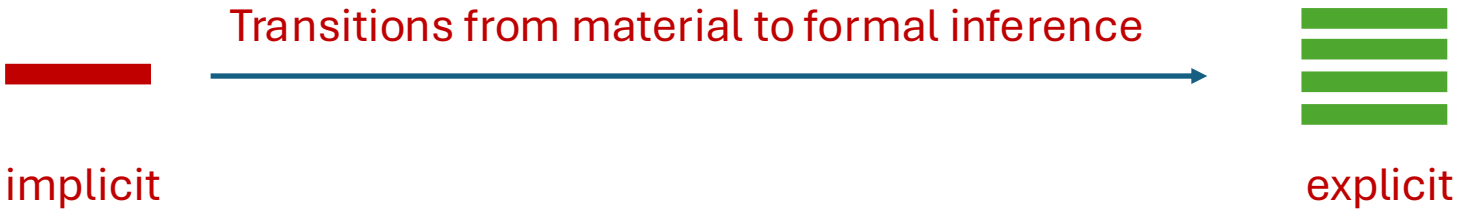


⊢ (The unreasonable effectiveness of) Reflective Loops

LLMs and symbolic solvers

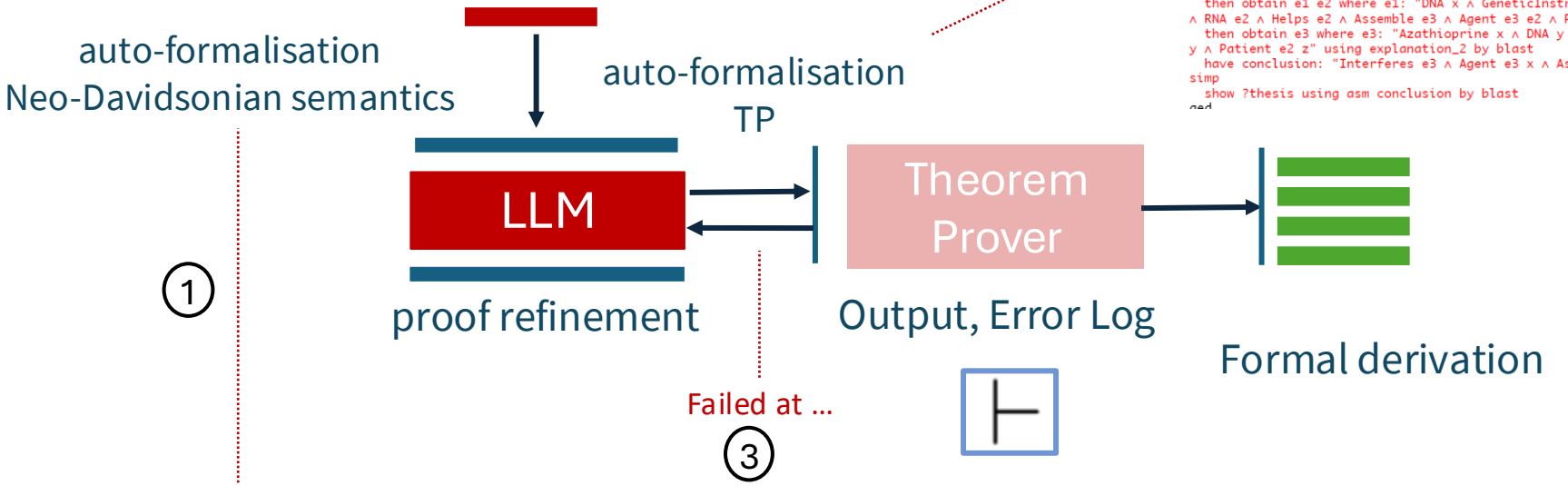
LLMs **translate** natural language into a formal language.
External symbolic solvers are responsible for **deriving** a logically valid solution.





Reflective refinement

Claim. Azathioprine interferes with the assembly of proteins.
Explanation. DNA contains genetic instructions for proteins, and RNA helps assemble the proteins. Azathioprine interferes with DNA and RNA synthesis.




```
(* Explanation 1: DNA contains genetic instructions for proteins, and RNA helps assemble the proteins. *)
axiomatization where
    explanation_1: "∀x y z e1 e2 e3. DNA x ∧ GeneticInstructions y ∧ Proteins z ∧ Contains e1 ∧ Agent e1 x ∧
Patient e1 y ∧ RNA e2 ∧ Helps e2 ∧ Assemble e3 ∧ Agent e3 e2 ∧ Patient e3 z ⟶ HelpsAssemble e2 e3"

(* Explanation 2: Azathioprine interferes with DNA and RNA synthesis. *)
axiomatization where
    explanation_2: "∀x y z e1 e2. Azathioprine x ∧ DNA y ∧ RNA z ∧ Interferes e1 ∧ Agent e1 x ∧ Synthesis e2 ∧
Patient e2 y ∧ Patient e2 z ⟶ InterferesWith e1 e2"

theorem hypothesis:
assumes asm: "Azathioprine x ∧ Proteins y"
(* Hypothesis: Azathioprine interferes with the assembly of proteins. *)
shows "∃x y e1 e2. Azathioprine x ∧ Proteins y ∧ Interferes e1 ∧ Agent e1 x ∧ Assembly e2 ∧ Patient e2 y ∧
InterferesWith e1 e2"
proof -
  from asm have "Azathioprine x ∧ Proteins y" by simp
  then obtain e1 e2 where e1: "DNA x ∧ GeneticInstructions y ∧ Proteins z ∧ Contains e1 ∧ Agent e1 x ∧ Patient e1 y
∧ RNA e2 ∧ Helps e2 ∧ Assemble e3 ∧ Agent e3 e2 ∧ Patient e3 z" using explanation_1 by blast
  then obtain e3 where e3: "Azathioprine x ∧ DNA y ∧ RNA z ∧ Interferes e3 ∧ Agent e3 x ∧ Synthesis e2 ∧ Patient e2
y ∧ Patient e2 z" using explanation_2 by blast
  have conclusion: "Interferes e3 ∧ Agent e3 x ∧ Assembly e2 ∧ Patient e2 y ∧ InterferesWith e3 e2" using e3 by
simp
  show ?thesis using asm conclusion by blast
qed
```

$$\exists e_1\,(\mathrm{Contain}(e_1)\wedge\mathrm{Agent}(e_1,\mathrm{DNA})\wedge\mathrm{Theme}(e_1,\mathrm{Genetic_Instructions})\wedge\mathrm{For}(e_1,\mathrm{Proteins}))$$

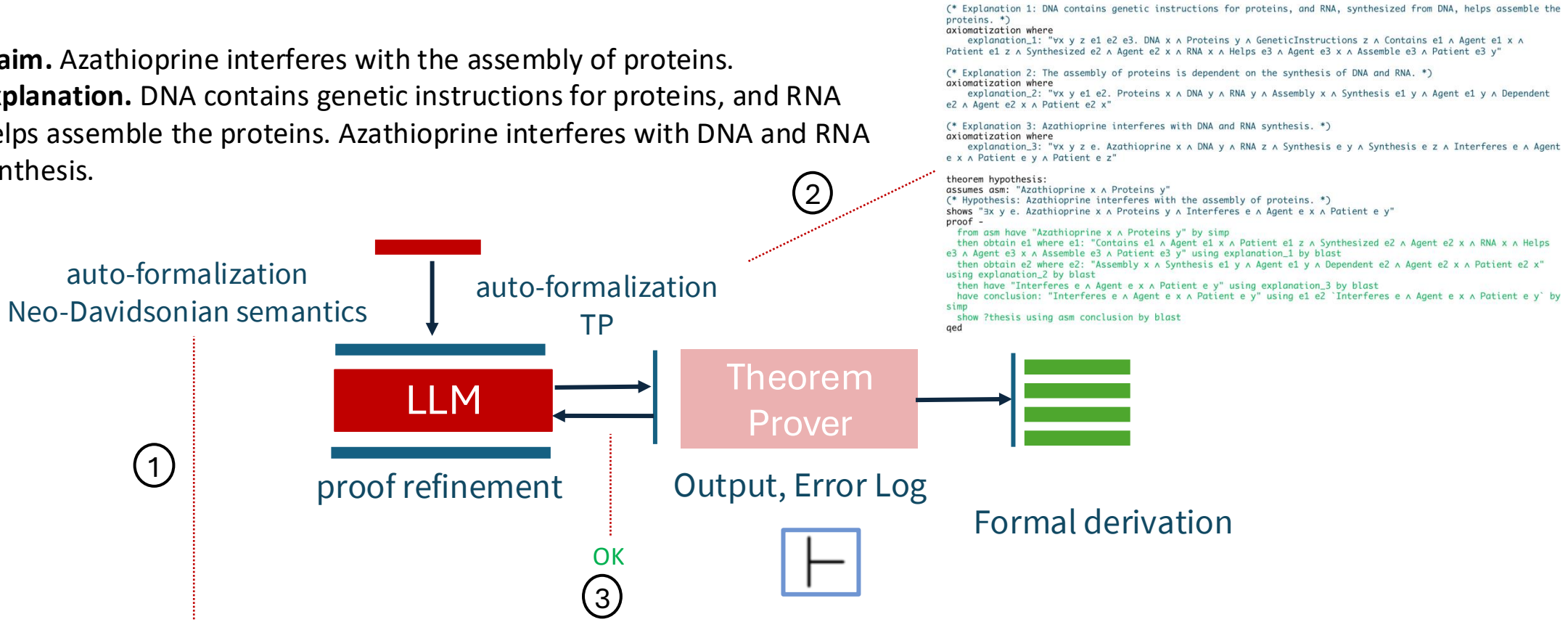
$$\exists e_2\,(\mathrm{Help}(e_2)\wedge\mathrm{Agent}(e_2,\mathrm{RNA})\wedge\exists e_3\,(\mathrm{Assemble}(e_3)\wedge\mathrm{Agent}(e_3,\mathrm{RNA})\wedge\mathrm{Theme}(e_3,\mathrm{Proteins})))$$

...

Quan et al, EMNLP (2024) – outstanding paper award
Quan et al, ACL (2025)

Reflective refinement

Claim. Azathioprine interferes with the assembly of proteins.
Explanation. DNA contains genetic instructions for proteins, and RNA helps assemble the proteins. Azathioprine interferes with DNA and RNA synthesis.




```
(* Explanation 1: DNA contains genetic instructions for proteins, and RNA, synthesized from DNA, helps assemble the
proteins. *)
axiomatization where
    explanation_1: "∀x y z e1 e2 e3. DNA x ∧ Proteins y ∧ GeneticInstructions z ∧ Contains e1 ∧ Agent e1 x ∧
Patient e1 z ∧ Synthesized e2 ∧ Agent e2 x ∧ RNA x ∧ Helps e3 ∧ Agent e3 x ∧ Assemble e3 ∧ Patient e3 y"

(* Explanation 2: The assembly of proteins is dependent on the synthesis of DNA and RNA. *)
axiomatization where
    explanation_2: "∀x y e1 e2. Proteins x ∧ DNA y ∧ RNA y ∧ Assembly x ∧ Synthesis e1 y ∧ Agent e1 y ∧ Dependent
e2 ∧ Agent e2 x ∧ Patient e2 x"

(* Explanation 3: Azathioprine interferes with DNA and RNA synthesis. *)
axiomatization where
    explanation_3: "∀x y z e. Azathioprine x ∧ DNA y ∧ RNA z ∧ Synthesis e y ∧ Synthesis e z ∧ Interferes e ∧ Agent
e x ∧ Patient e y ∧ Patient e z"

theorem hypothesis:
assumes asm: "Azathioprine x ∧ Proteins y"
(* Hypothesis: Azathioprine interferes with the assembly of proteins. *)
shows "∃x y e. Azathioprine x ∧ Proteins y ∧ Interferes e ∧ Agent e x ∧ Patient e y"
proof -
  from asm have "Azathioprine x ∧ Proteins y" by simp
  then obtain e1 where e1: "Contains e1 ∧ Agent e1 x ∧ Patient e1 z ∧ Synthesized e2 ∧ Agent e2 x ∧ RNA x ∧ Helps
e3 ∧ Agent e3 x ∧ Assemble e3 ∧ Patient e3 y" using explanation_1 by blast
  then obtain e2 where e2: "Assembly x ∧ Synthesis e1 y ∧ Agent e1 y ∧ Dependent e2 ∧ Agent e2 x ∧ Patient e2 x"
using explanation_2 by blast
  then have "Interferes e ∧ Agent e x ∧ Patient e y" using explanation_3 by blast
  have conclusion: "Interferes e ∧ Agent e x ∧ Patient e y" using e1 e2 `Interferes e ∧ Agent e x ∧ Patient e y` by
simp
  show ?thesis using asm conclusion by blast
qed
```

$$\exists e_1\,(\mathrm{Contain}(e_1)\wedge\mathrm{Agent}(e_1,\mathrm{DNA})\wedge\mathrm{Theme}(e_1,\mathrm{Genetic_Instructions})\wedge\mathrm{For}(e_1,\mathrm{Proteins}))$$
$$\exists e_2\,(\mathrm{Help}(e_2)\wedge\mathrm{Agent}(e_2,\mathrm{RNA})\wedge\exists e_3\,(\mathrm{Assemble}(e_3)\wedge\mathrm{Agent}(e_3,\mathrm{RNA})\wedge\mathrm{Theme}(e_3,\mathrm{Proteins})))$$

...

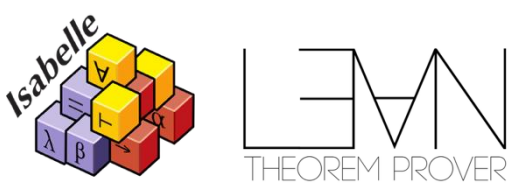
Refined Explanation. DNA contains genetic instructions for proteins, and RNA, synthesized from DNA, helps assemble the proteins. The assembly of proteins depends on DNA and RNA synthesis. Azathioprine interferes with DNA and RNA synthesis.

```
(* Explanation 1: DNA contains genetic instructions for proteins, and RNA, synthesized from DNA, helps assemble the proteins. *)
axiomatization where
    explanation_1: "∀x y z e1 e2 e3. DNA x ∧ Proteins y ∧ GeneticInstructions z ∧ Contains e1 ∧ Agent e1 x ∧ Patient e1 z ∧ Synthesized e2 ∧ Agent e2 x ∧ RNA x ∧ Helps e3 ∧ Agent e3 x ∧ Assemble e3 ∧ Patient e3 y"

(* Explanation 2: The assembly of proteins is dependent on the synthesis of DNA and RNA. *)
axiomatization where
    explanation_2: "∀x y e1 e2. Proteins x ∧ DNA y ∧ RNA y ∧ Assembly x ∧ Synthesis e1 y ∧ Agent e1 y ∧ Dependent e2 ∧ Agent e2 x ∧ Patient e2 x"

(* Explanation 3: Azathioprine interferes with DNA and RNA synthesis. *)
axiomatization where
    explanation_3: "∀x y z e. Azathioprine x ∧ DNA y ∧ RNA z ∧ Synthesis e y ∧ Synthesis e z ∧ Interferes e ∧ Agent e x ∧ Patient e y ∧ Patient e z"

theorem hypothesis:
assumes asm: "Azathioprine x ∧ Proteins y"
(* Hypothesis: Azathioprine interferes with the assembly of proteins. *)
shows "∃x y e. Azathioprine x ∧ Proteins y ∧ Interferes e ∧ Agent e x ∧ Patient e y"
proof -
  from asm have "Azathioprine x ∧ Proteins y" by simp
  then obtain e1 where e1: "Contains e1 ∧ Agent e1 x ∧ Patient e1 z ∧ Synthesized e2 ∧ Agent e2 x ∧ RNA x ∧ Helps e3 ∧ Agent e3 x ∧ Assemble e3 ∧ Patient e3 y" using explanation_1 by blast
  then obtain e2 where e2: "Assembly x ∧ Synthesis e1 y ∧ Agent e1 y ∧ Dependent e2 ∧ Agent e2 x ∧ Patient e2 x" using explanation_2 by blast
  then have "Interferes e ∧ Agent e x ∧ Patient e y" using explanation_3 by blast
  have conclusion: "Interferes e ∧ Agent e x ∧ Patient e y" using e1 e2 `Interferes e ∧ Agent e x ∧ Patient e y` by simp
  show ?thesis using asm conclusion by blast
qed
```

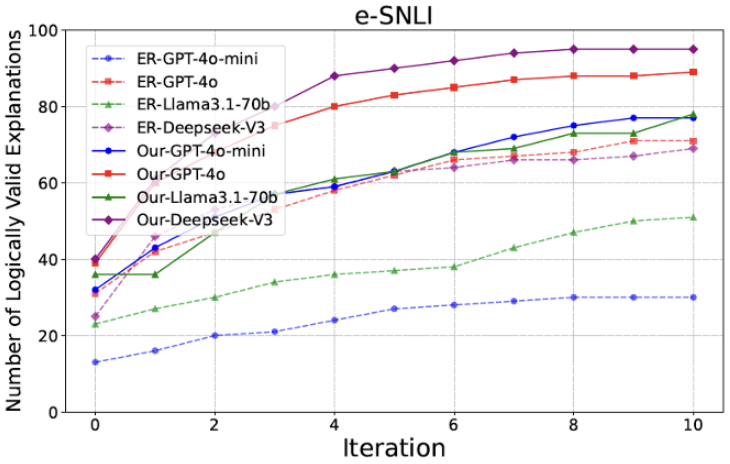


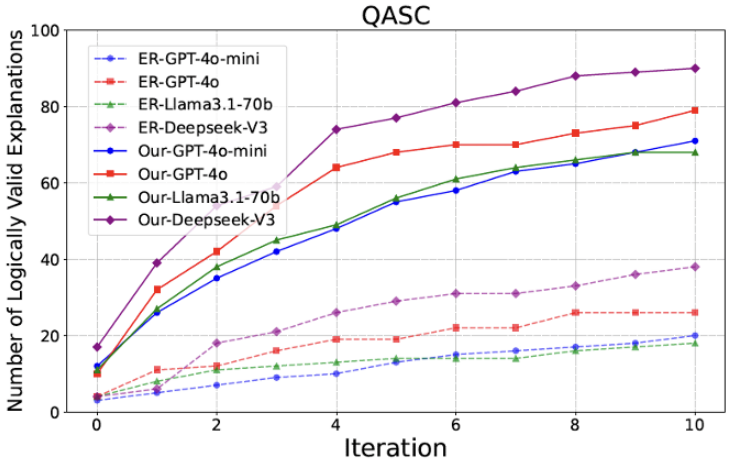
NL Autoformalization

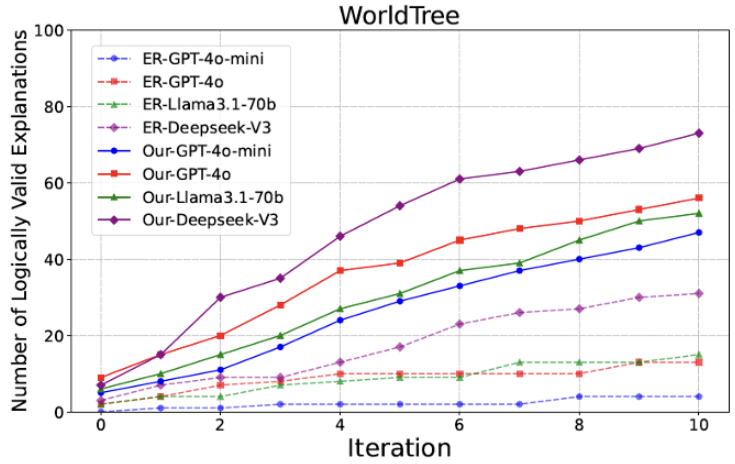
	e-SNLI			QASC			WorldTree		
	Init.	**Final**	**#Iter**	**Init.**	**Final**	**#Iter**	**Init.**	**Final**	**#Iter**
Explanation-Refiner									
Llama3.1-70b	23%	51%	4.08	4%	18%	4.07	2%	15%	5.23
GPT-4o-mini	13%	30%	3.65	3%	20%	5.12	0%	4%	5.00
GPT-4o	31%	71%	3.62	4%	26%	4.35	2%	13%	**4.18**
Deepseek-V3	25%	**69%**	**2.82**	4%	**38%**	**3.71**	3%	**31%**	4.52
Our Approach									
Llama3.1-70b	36%	78%	2.38	11%	68%	2.90	6%	52%	4.62
GPT-4o-mini	32%	77%	2.27	12%	71%	3.35	5%	47%	4.75
GPT-4o	39%	89%	1.54	10%	79%	3.22	9%	56%	3.86
Deepseek-V3	41%	**95%**	**1.50**	17%	**90%**	**2.53**	7%	**73%**	**3.55**

Quan et al, EMNLP (2024)
Quan et al, ACL (2025)

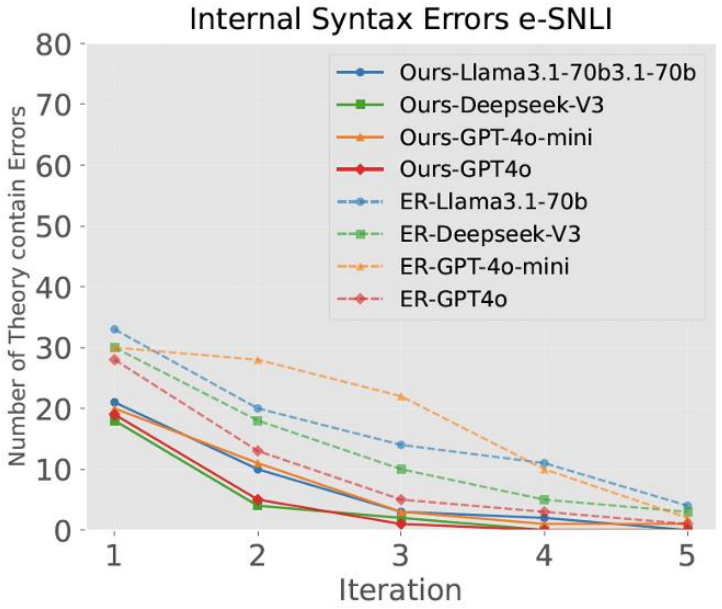
Number of logically valid explanations in each refinement iteration

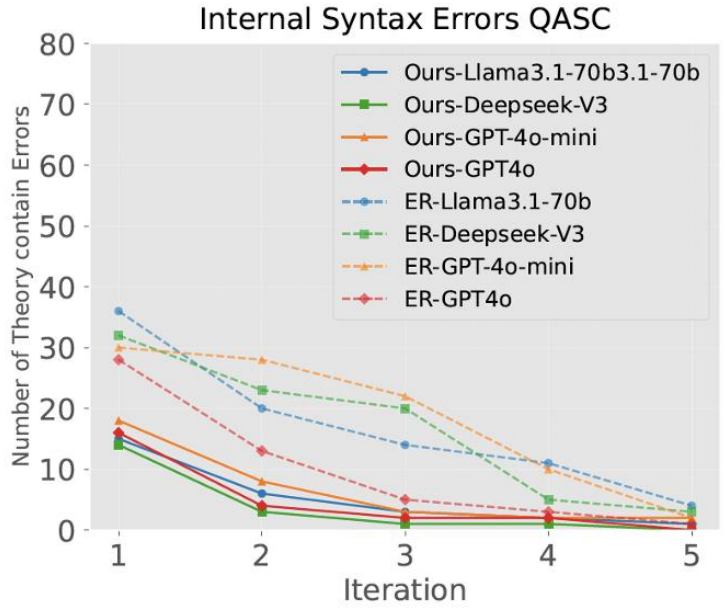


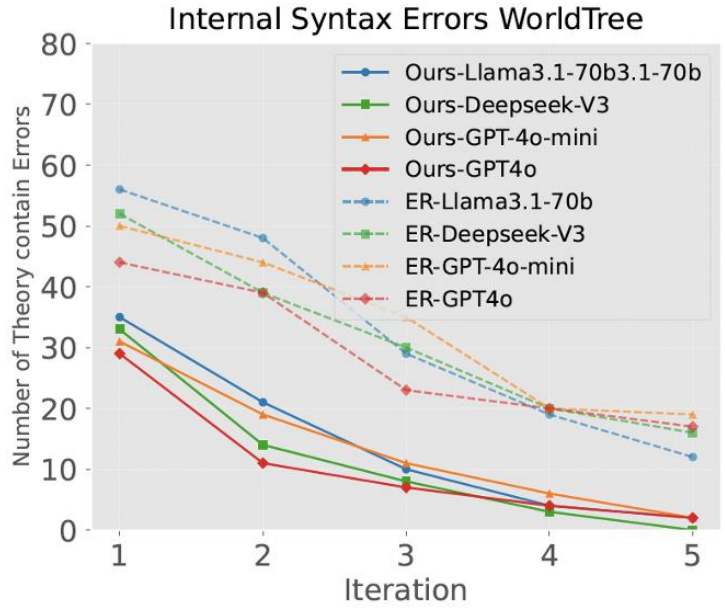




Number of theories that contain internal syntactic errors at each syntactic-error refinement step





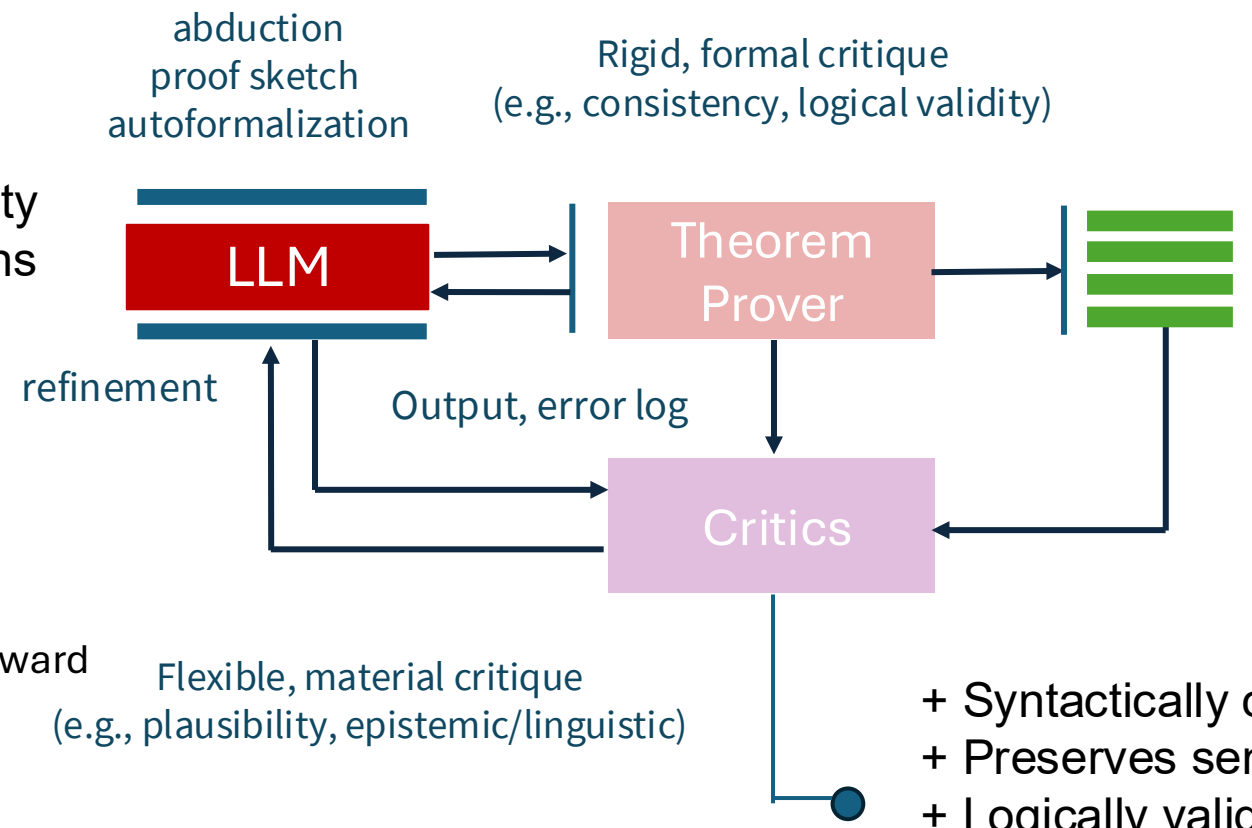


Evaluating & refining reasoning

Gold standards: become infeasible very quickly for complex, specialized reasoning.

Solution: Evaluate via proxies that can be computed in an unsupervised way.

Proxies for inference quality
Evaluate different dimensions

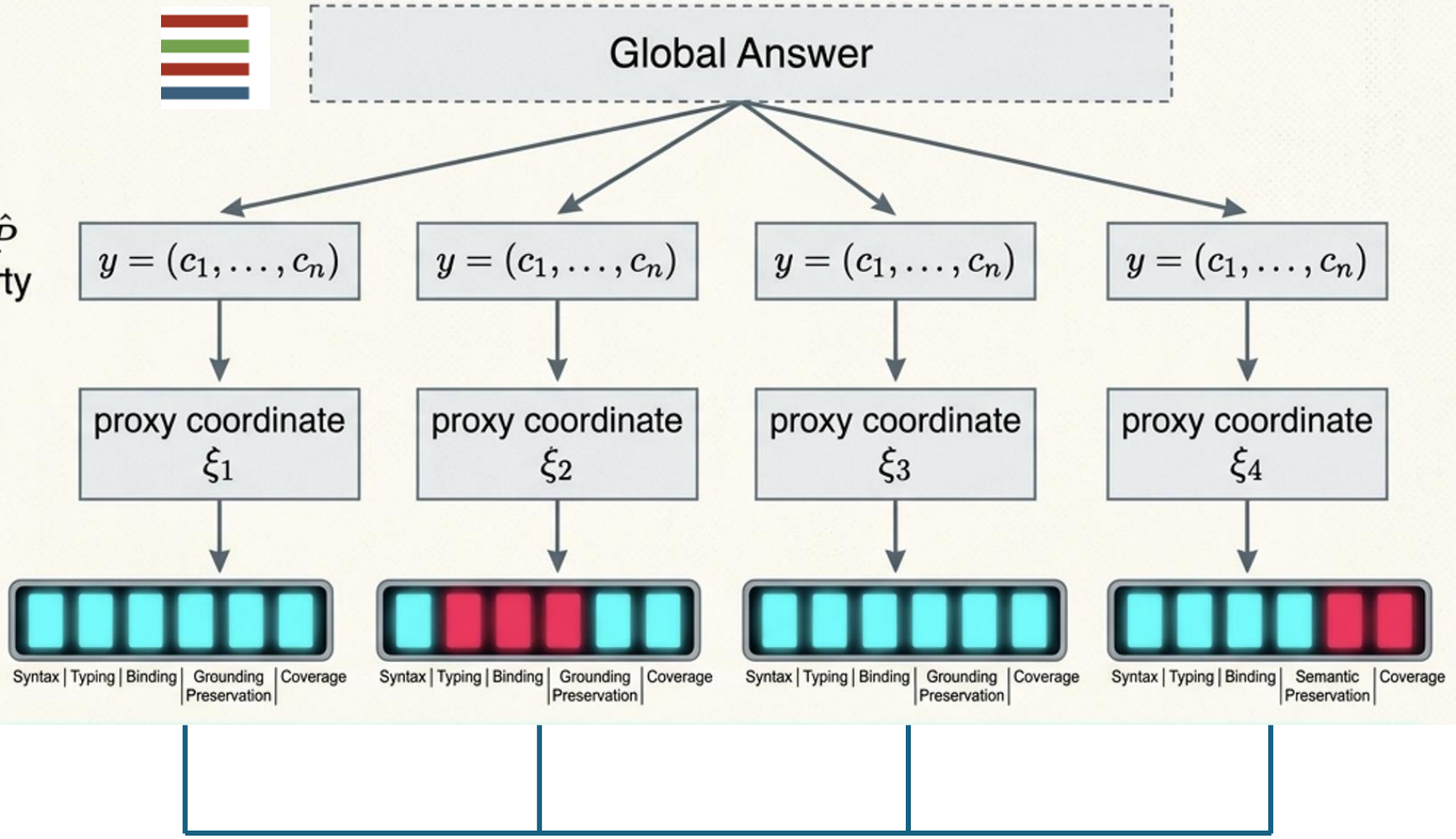


Zhang et al, EMNLP (2025) – best resource paper award
Zhang et al, ArXiv: 2506.10903 (2026)
Xu et al, ArXiv: 2606.09449 (2026)

Reference free: Localized *critics* as quality proxies

The Problem: Global answer quality is difficult to observe directly.

The Solution: Evaluate a structured elicited object $\hat{P}$ through a vector of property checks:
$\xi(\hat{P}) = (\xi_1, \ldots, \xi_m)$.

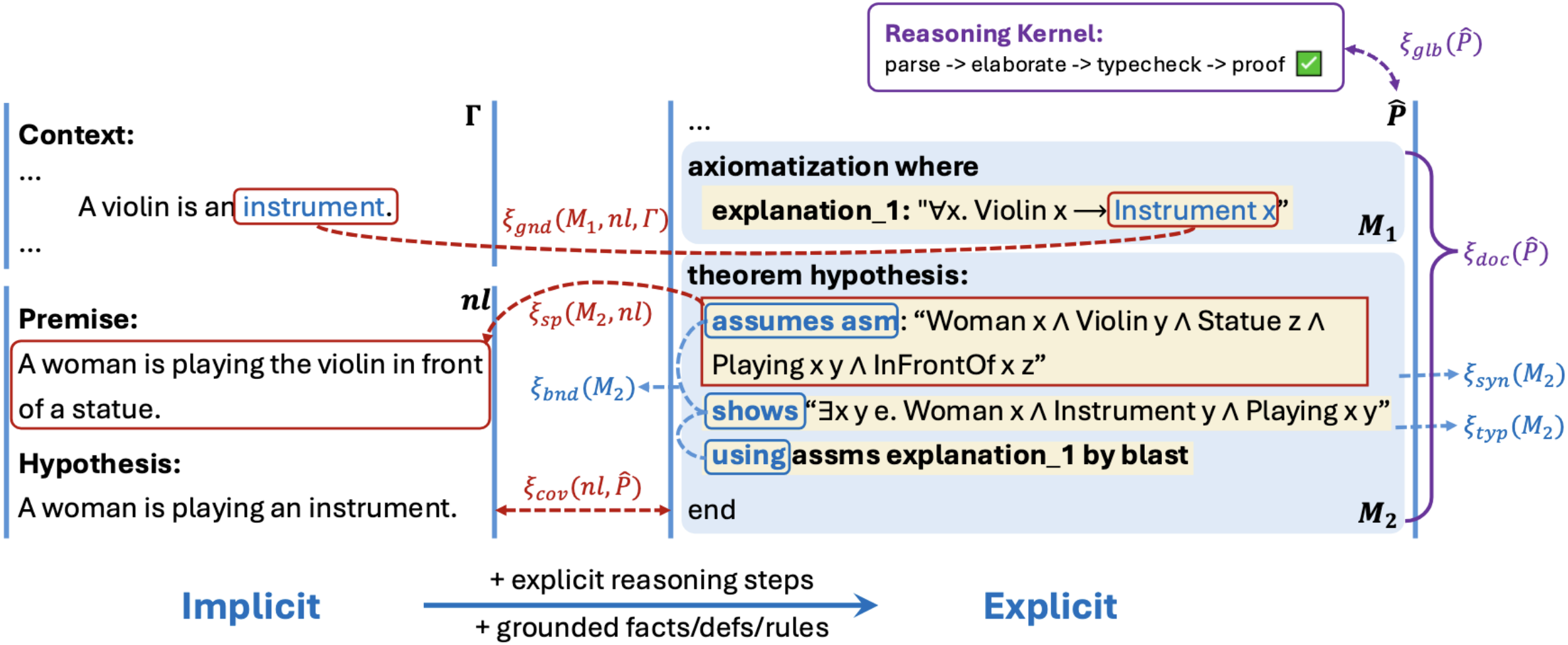


compositionality

Linguistic properties/relations can be localized

	Proof text Nodes mark terms. Arcs connect their occurrences.		**Function** What does it do?	**Discourse relation** How does it link spans?	**Statement type** What kind of content?
0	**Theorem.** There are infinitely many primes.		State the goal	Organises the proof	Proposition Target claim
1	**Suppose** $p_1, \ldots, p_k$ list all primes.		Open a hypothetical context	Opposition to 0	Hypothetical proposition
2	**Let** $N := p_1 p_2 \cdots p_k + 1$.		Introduce an object	Elaboration of 1 Uses its listed primes	Definitional statement
3	**Since** $N > 1$, it has a prime divisor q.		Invoke a known fact to obtain a witness	Justification Within this sentence	Existential proposition
4	For each i: $N \equiv 1 \pmod{p_i}$, **so** $q \neq p_i$.		Derive a constraint	Consequence From 2 and 3	Universal proposition
5	**But** q is prime and outside the list.		Expose a conflict	Contradiction 3–4 conflict with 1	Incompatibility claim
6	**Therefore**, infinitely many primes exist.	□	Discharge the assumption	Conclusion From the subproof 1–5	Proposition Established claim

Proxies for correctness





Interactive theorem provers

Truth in Logic vs Constructive Type Theory

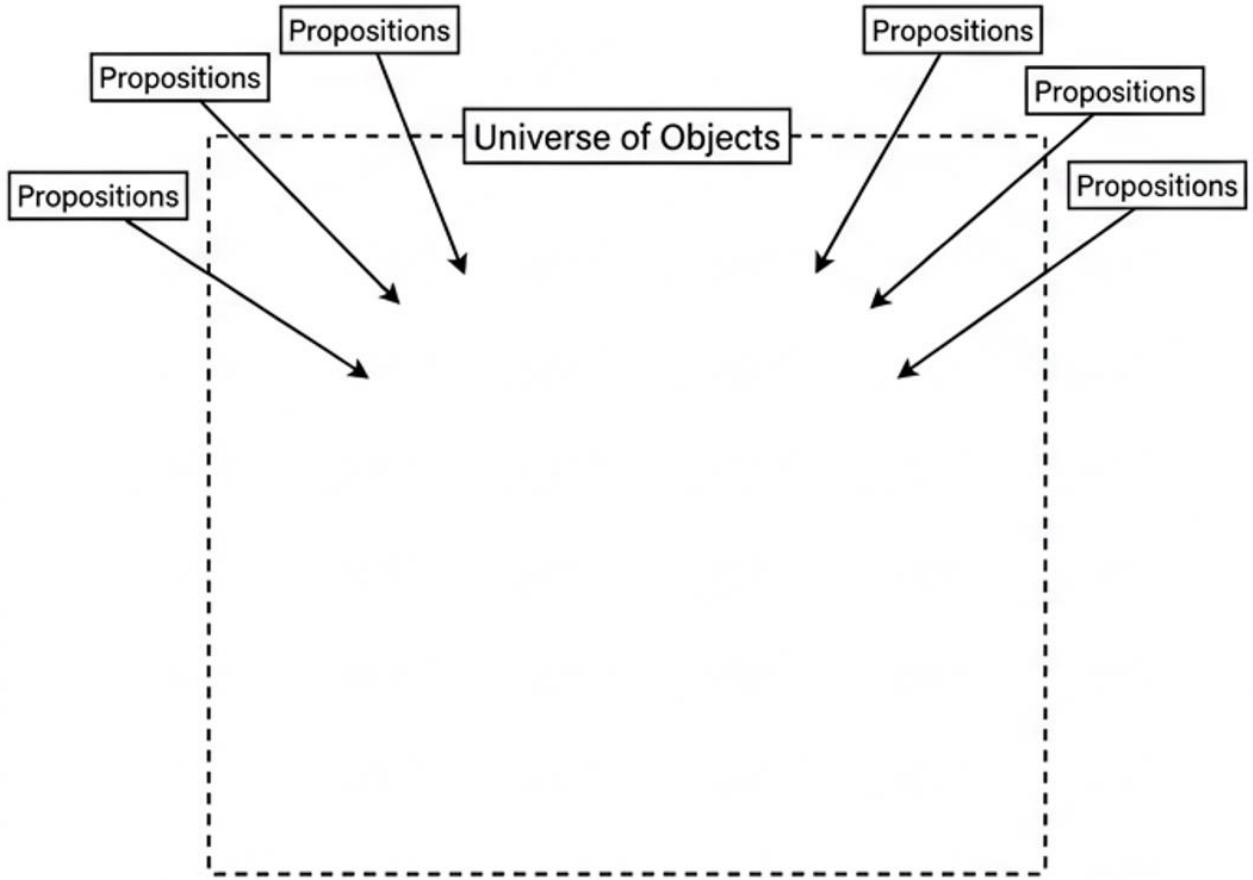


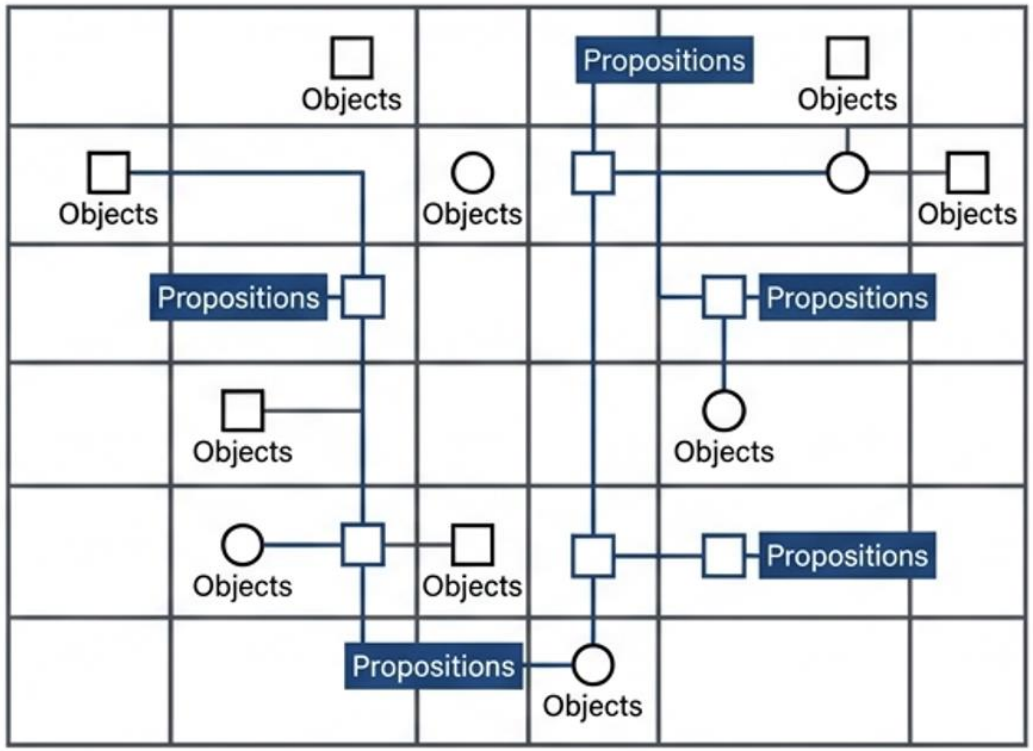
Classical Logic: Truth is an inherent, abstract state. A proposition simply is true or false.



Constructive Type Theory: Truth is a witnessed construction. To know a proposition holds is to possess the mathematical object (the proof) that proves it.

Set Theory vs Constructive Type Theory





Curry-Howard correspondence

Logic	Equivalence	Computation
Proposition	≡	Type
Proof	≡	Term
Proof Checking	≡	Type Checking

This doctrine unifies logic and computation. Programs are seen as proofs, and validating a logical argument is mathematically identical to checking a software type.

Proof-style programming syntax

Logical correctness

```
def factorial (n : Nat) : Nat :=
  match n with
   | 0 => 1
   | n + 1 => (n + 1) * factorial n
```

```
proof_term : P
```

$$\forall\ (n : \mathbb{N}),\ P(n) \to P(n+1)$$
$$\forall\ (n : \mathbb{N}),\ m > n \to P(n+)$$
$$\exists\ (m : \mathbb{N}),\ m > n \land m < 2*n$$
$$\exists\ (m : \mathbb{N}),\ m > n \land m < 2*n$$
$$\forall\ (m = n)\ (n) \land b,\ P(n+1)$$
$$\Gamma \vdash \varphi : \mathrm{Type} \to P(n\ \varphi : \mathrm{Type}$$
$$\Gamma \vdash \varphi : \mathrm{Type}$$

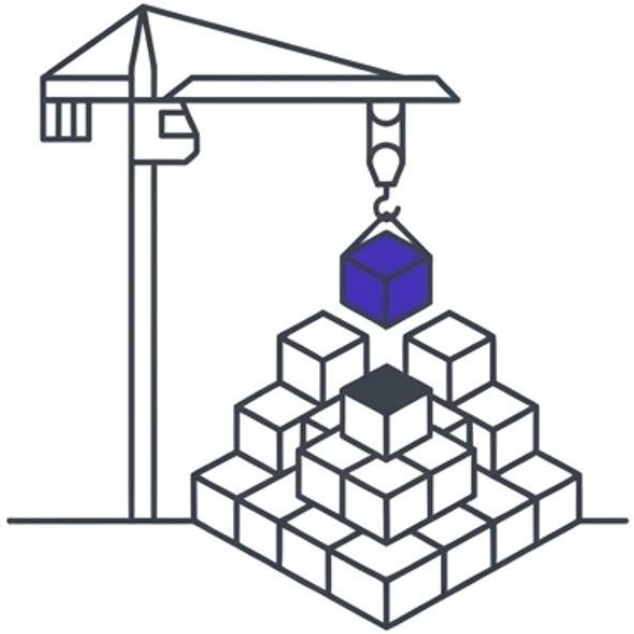
Logic programming vs proof construction

Prolog



Derivation by Search
Backward chaining and unification to satisfy a query.

Lean

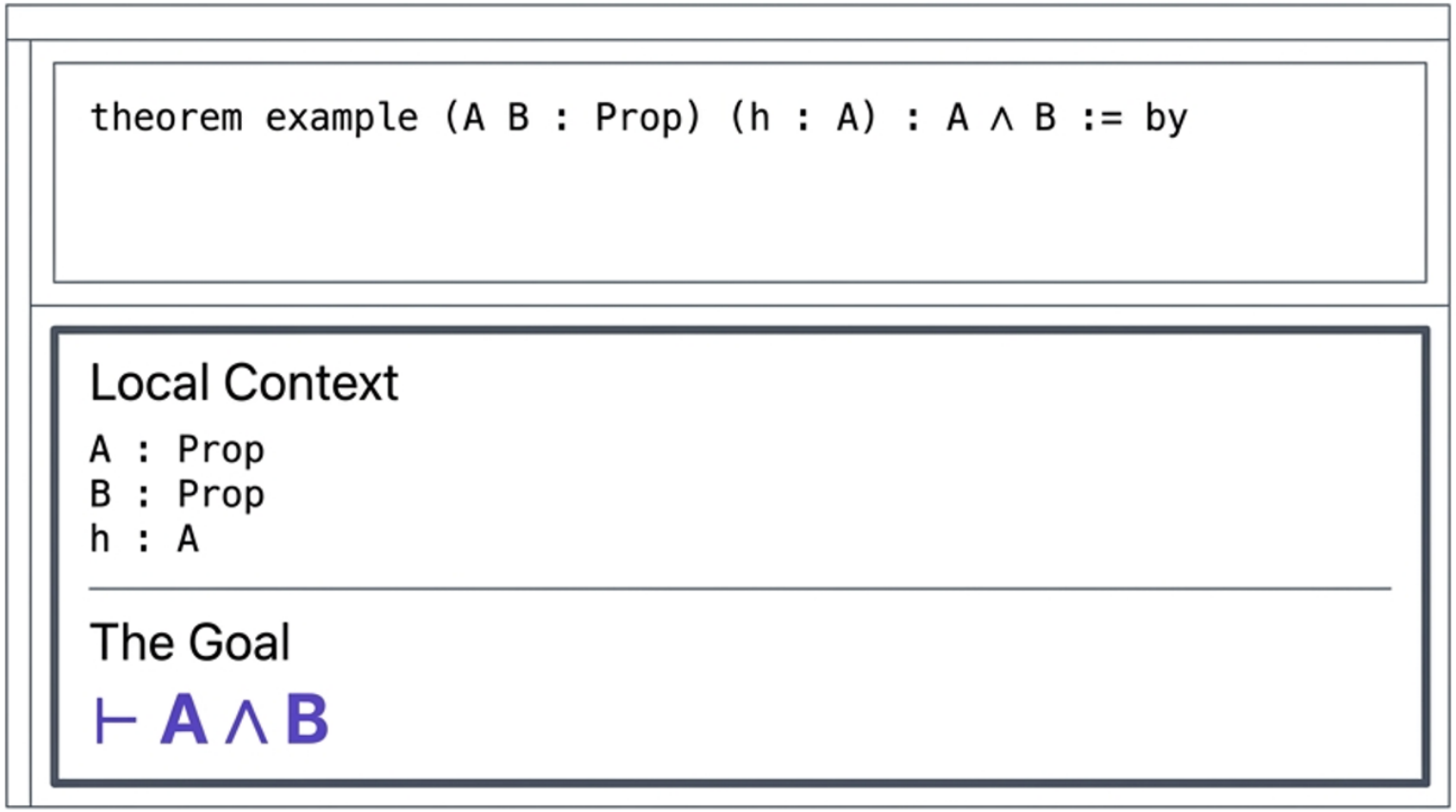


Certified Construction
Building a formal proof object to be verified by the kernel.

Lean vs Prolog

	Lean	Prolog
Foundation	Dependent Type Theory	Horn Clause Logic
Main Activity	Construct Proofs	Resolve Queries
Endpoint Result	**Explicit checked proof object**	Successful answer substitution
Automation	Tactics, rewriting, simplification	Unification, backtracking
Trust Basis	Small foundational kernel	Operational search procedure

Goal-oriented workflow of proof assistants

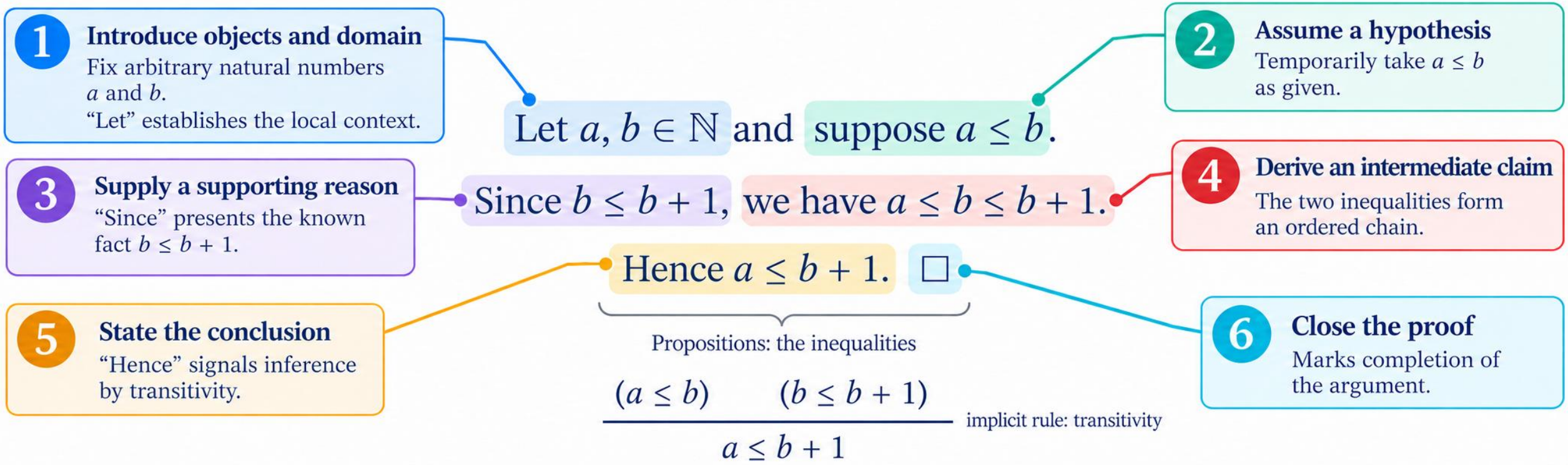


Define a state that must be satisfied by decomposing it into subgoals.
Assumes a human in the loop to plan and decompose.
Some operations can fill the gaps.

Let’s make it concrete: informal proof

Let $a, b \in \mathbb{N}$ and suppose $a \leq b$. Since $b \leq b + 1$, we have $a \leq b \leq b + 1$. Hence $a \leq b + 1$. $\square$

Breaking down language/inferential functions



Lean language constructs

Let $a, b \in \mathbb{N}$ and suppose $a \leq b$. Since $b \leq b+1$,
we have $a \leq b \leq b+1$. Hence $a \leq b+1$. $\square$

What the Lean keywords do

import
Loads a module.
Mathlib provides mathematical definitions and lemmas.

theorem
Declares a named proof.
step_up names a proof of the proposition after the colon.

by
Starts tactic mode.
The indented commands construct a proof term.

```
import Mathlib

theorem step_up (a b : ℕ) (h : a ≤ b) :
    a ≤ b + 1 := by
  have h2 : b ≤ b + 1 := Nat.le_succ b
  exact le_trans h h2
```

have
Introduces an intermediate fact.
h2 names a proof that b ≤ b + 1.

exact
Closes the current goal.
The supplied term must have the required type.

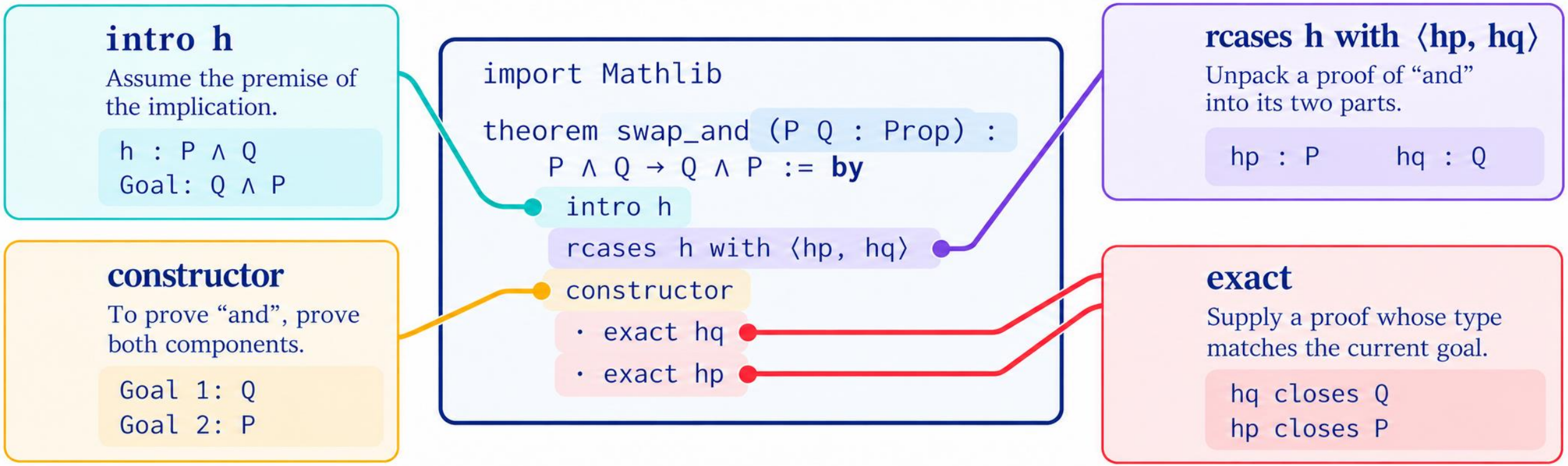
Reading the proof expression

```
le_trans h h2
```

Apply transitivity to the proofs h and h2.
le_trans and Nat.le_succ are theorem names, not keywords.

: introduces a type or proposition := supplies a value or proof

Tactics


intro h
Assume the premise of the implication.
h : P ∧ Q
Goal: Q ∧ P
constructor
To prove “and”, prove both components.
Goal 1: Q
Goal 2: P
import Mathlib
theorem swap_and (P Q : Prop) :
P ∧ Q → Q ∧ P := by
intro h
rcases h with ⟨hp, hq⟩
constructor
· exact hq
· exact hp
rcases h with ⟨hp, hq⟩
Unpack a proof of “and” into its two parts.
hp : P
hq : Q
exact
Supply a proof whose type matches the current goal.
hq closes Q
hp closes P

Available tactics

Introduce and build

```
intro · intros · rintro
constructor · left · right · use
```

Introduce assumptions or construct evidence.

Apply and close

```
exact · apply · refine
assumption · rfl · trivial
```

Supply a proof or reduce the goal to premises.

Organize facts

```
have · let · suffices · obtain
specialize · revert · clear · rename_i
```

Create and manage local facts and variables.

Cases and induction

```
cases · rcases · induction
by_cases · split
```

Divide the proof into cases or inductive steps.

Negation and contradiction

```
by_contra · contradiction
exfalso · push_neg · contrapose
```

Reason through negation and falsehood.

Rewrite and simplify

```
rw · erw · simp · simpa
simp_all · dsimp · unfold · subst
```

Rewrite equalities and simplify expressions.

Equality and structure

```
change · congr · ext · funext
symm · trans
```

Compare components and transform relations.

Arithmetic and algebra

```
norm_num · ring · ring_nf
linarith · nlinarith · omega
```

Normalize expressions and solve arithmetic goals.

Bounds, fields and casts

```
positivity · field_simp
norm_cast · push_cast
exact_mod_cast · gcongr
```

Handle signs, denominators, coercions and bounds.

Search and automation

```
aesop · grind · decide
exact? · apply? · simp?
solve_by_elim
```

Search for proofs or invoke automated reasoning.

Control subgoals

```
all_goals · any_goals · focus
first | · try · repeat · <;>
```

Select, combine and repeat tactic actions.

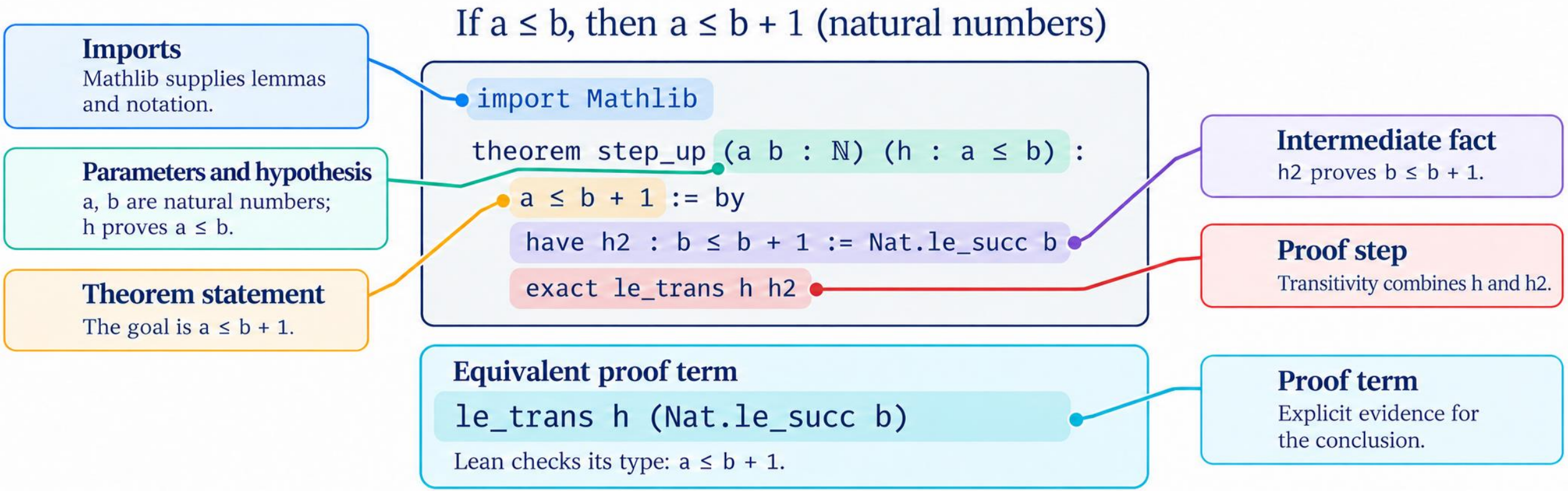
Specialized reasoning

```
fin_cases · interval_cases
fun_prop · measurability
continuity · filter_upwards
```

Use automation for particular mathematical domains.

Proof elements

Let $a, b \in \mathbb{N}$ and suppose $a \leq b$. Since $b \leq b + 1$, we have $a \leq b \leq b + 1$. Hence $a \leq b + 1$. $\square$



Imported lemmas from Mathlib

`import Mathlib`

Makes declarations available from Mathlib and its dependencies.

Reuse through imports, not class inheritance.

Parameters and hypothesis

`(a b : ℕ) (h : a ≤ b)`

Introduce the natural numbers `a`, `b` and the hypothesis `h`.

Goal of the theorem

`a ≤ b + 1`

This is the statement to be proved, shown before `:= by`.

```
import Mathlib

theorem step_up (a b : ℕ) (h : a ≤ b) :
    a ≤ b + 1 := by
  have h2 : b ≤ b + 1 := Nat.le_succ b
  exact le_trans h h2
```

Nat.le_succ

General fact: `n ≤ Nat.succ n`

At `n = b`: `b ≤ b + 1`

This supplies the proof named `h2`.

For natural numbers, `b + 1` and `Nat.succ b` are definitionally equal.

le_trans

General rule: `x ≤ y` and `y ≤ z` imply `x ≤ z`

Here: `x = a`, `y = b`, `z = b + 1`

Inputs: `h : a ≤ b` and `h2 : b ≤ b + 1`

Result: `a ≤ b + 1`

Lean fills in the implicit information

From `h` and `h2`, Lean infers the endpoints and the order on ℕ.

$$\frac{h : a \le b \qquad h2 : b \le b + 1}{le_trans\ h\ h2 : a \le b + 1}$$

These lemmas belong to Lean's imported library environment. Some originate in Lean core rather than Mathlib itself.

Mathlib bindings

Natural numbers — Lean core

```
inductive Nat where
  | zero : Nat
  | succ (n : Nat) : Nat
```

ℕ is notation for Nat. Zero and successor generate the natural numbers.

Init/Prelude.lean

The order relation — Lean core

```
protected inductive Nat.le (n : Nat) : Nat → Prop
  | refl : Nat.le n n
  | step {m} : Nat.le n m → Nat.le n (succ m)

instance instLENat : LE Nat where
  le := Nat.le
```

Reflexivity gives n ≤ n. A step extends n ≤ m to n ≤ m + 1.

Init/Prelude.lean

The successor lemma — Lean core

```
theorem Nat.le_succ (n : Nat) :
  LE.le n (succ n) :=
  Nat.le.step Nat.le.refl
```

One step from reflexivity proves n ≤ n + 1.

Init/Prelude.lean

Why + 1 is successor — Lean core

Inside namespace Nat

```
theorem add_one (n : Nat) :
  n + 1 = succ n :=
  rfl
```

rfl checks definitional equality: both expressions compute to the same form.

Init/Data/Nat/Basic.lean

Generic transitivity — Mathlib

```
variable [Preorder α] {a b c : α}

lemma le_trans :
  a ≤ b → b ≤ c → a ≤ c :=
  Preorder.le_trans _ _ _
```

Uses the transitivity field of a preorder.
Here Lean selects the order on ℕ.

Mathlib/Order/Defs/PartialOrder.lean

Our proof reuses these declarations

```
import Mathlib

theorem step_up (a b : ℕ) (h : a ≤ b) :
  a ≤ b + 1 := by
  have h2 : b ≤ b + 1 := Nat.le_succ b
  exact le_trans h h2
```

Nat.le_succ supplies h2. le_trans combines h and h2.

Mathlib domains

Foundations and order

Logic, sets, functions, relations
Orders, lattices, cardinals, ordinals

$$\forall\, x,\ P\ x \qquad a \le b$$

Algebra

Groups, rings, fields and modules
Polynomials and field extensions

$$a(b + c) = ab + ac$$

Linear algebra

Vector spaces and linear maps
Matrices, bases and dimension

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Number theory

Primes, divisibility and congruences
p-adic numbers and number fields

$$a \equiv b \pmod{n}$$

Analysis

Real and complex analysis
Derivatives, series and normed spaces

$$f'(x)$$

Topology

Topological and metric spaces
Continuity, compactness and filters



Measure and integration

Measurable spaces and measures
Lebesgue and Bochner integration

$$\int f\, d\mu$$

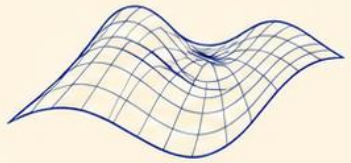
Probability

Distributions and random variables
Independence and conditional expectation

$$\mathbb{E}[X]$$

Geometry and manifolds

Euclidean and convex geometry
Smooth manifolds and Lie groups



Combinatorics and graphs

Finite sets, counting and graphs
Combinatorial and extremal results



Category theory

Categories, functors and natural transformations
Limits, adjunctions and monoidal categories

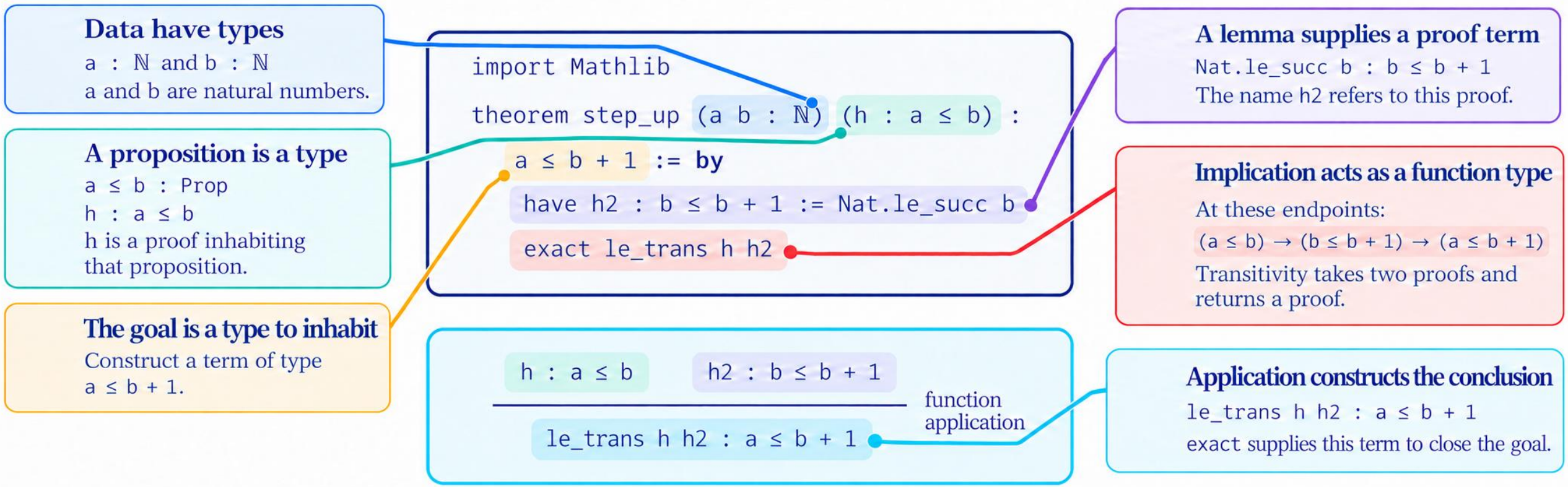
$$A \to B \to C$$

Advanced algebraic structures

Algebraic geometry and schemes
Homological algebra and algebraic topology

$$\ldots \to C_2 \to C_1 \to C_0$$

Types and proofs



In practice

Let $a, b \in \mathbb{N}$ and suppose $a \leq b$. Since $b \leq b + 1$, we have $a \leq b \leq b + 1$. Hence $a \leq b + 1$. $\square$

Lean

```
import Mathlib

theorem step_up (a b : ℕ) (h : a ≤ b) :
    a ≤ b + 1 := by
      |
```

Begin the proof

Proof state

```
a b : ℕ
h : a ≤ b
```

```
⊢ a ≤ b + 1
```

One goal to prove

In practice

Lean

```
import Mathlib

theorem step_up (a b : ℕ) (h : a ≤ b) :
  a ≤ b + 1 := by
    have h2 : b ≤ b + 1 := by
      |
```

Proof state

```
a b : ℕ
h : a ≤ b
⊢ b ≤ b + 1
```

Proving the intermediate claim

Outer goal resumes afterward:

```
a ≤ b + 1
```

In practice

Lean

```
import Mathlib

theorem step_up (a b : ℕ) (h : a ≤ b) :
    a ≤ b + 1 := by
    have h2 : b ≤ b + 1 := by
        exact Nat.le_succ b
```

Proof state

```
a b : ℕ
h : a ≤ b
h2 : b ≤ b + 1
```

```
⊢ a ≤ b + 1
```

h2 is now available

In practice

Lean

```
import Mathlib

theorem step_up (a b : ℕ) (h : a ≤ b) :
    a ≤ b + 1 := by
      have h2 : b ≤ b + 1 := by
            exact Nat.le_succ b
      exact le_trans h h2
```

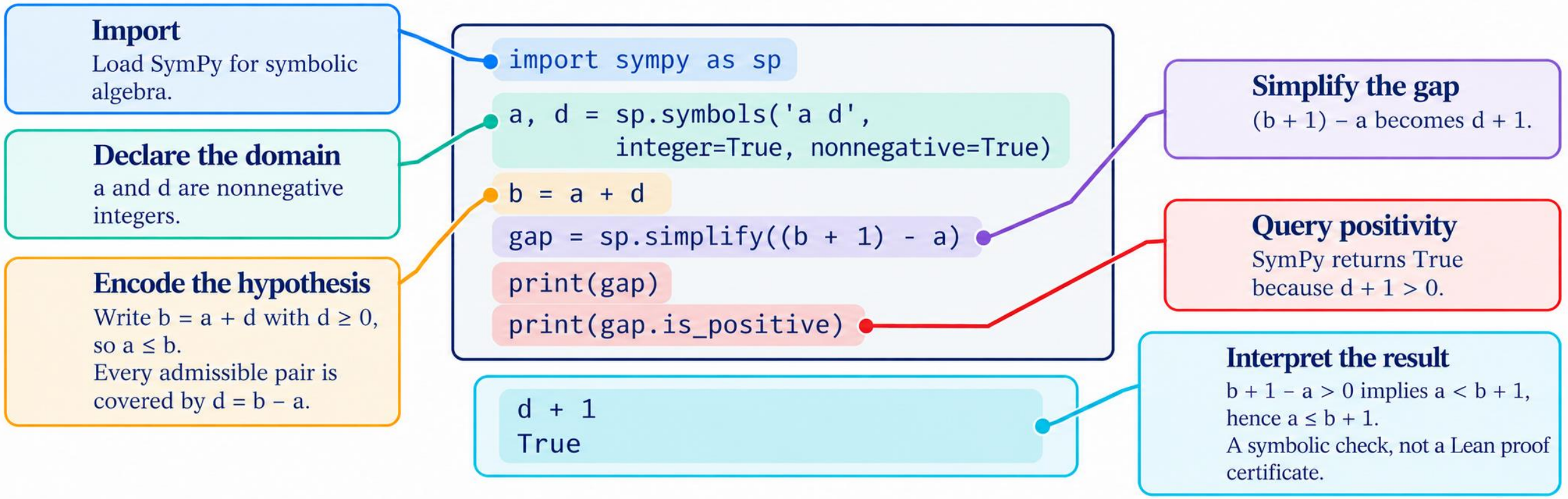
Proof state



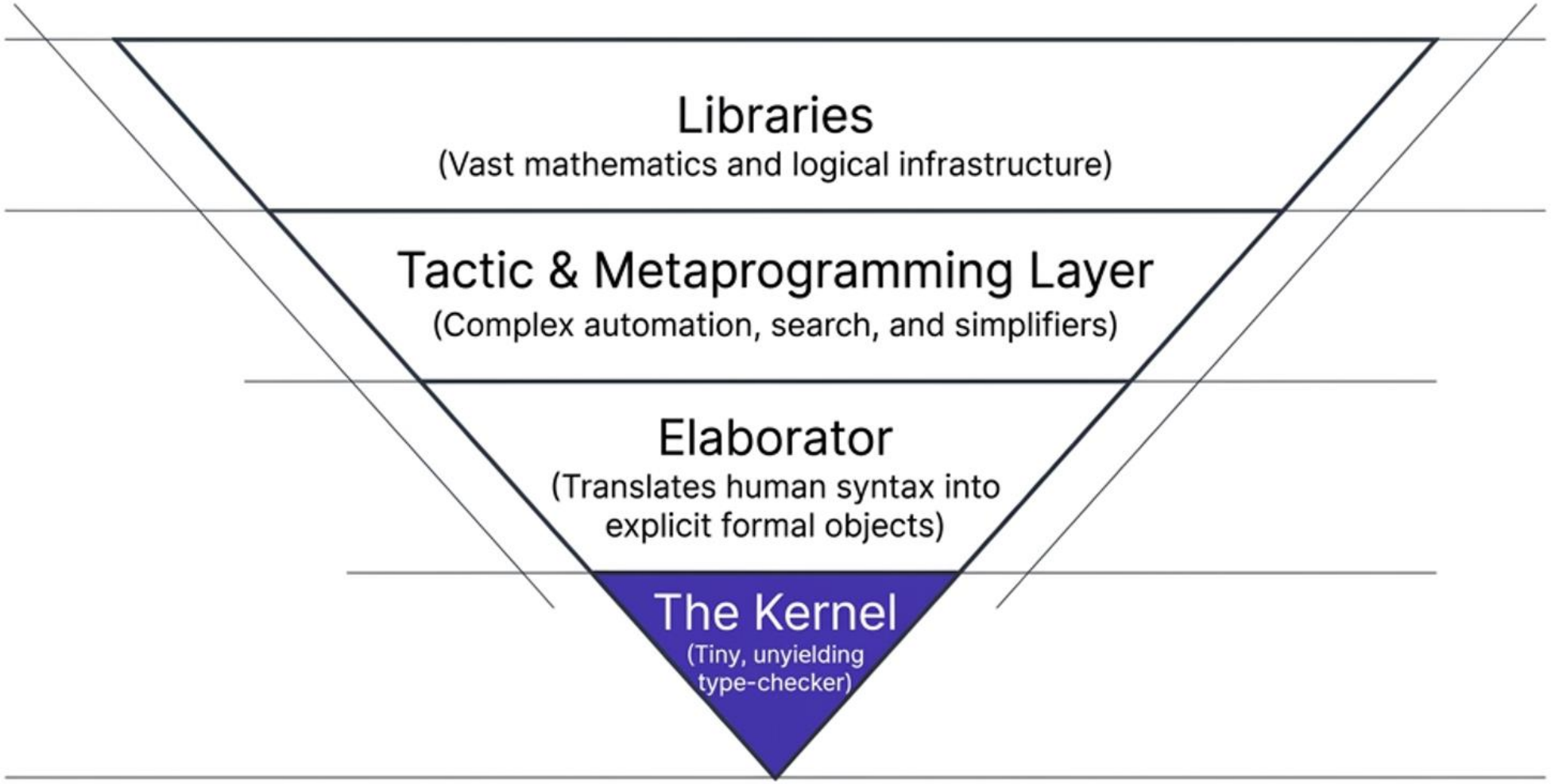
No goals

Proof complete

Contrasting to an algebraic solver (Sympy)



Lean architecture



An example from Physics

Let a particle in $\mathbb{R}^3$ have constant mass $m > 0$, charge q, and differentiable velocity $\boldsymbol{v}(t)$ on a time interval.

Suppose its only force is magnetic:

$$m\dot{\boldsymbol{v}} = q(\boldsymbol{v} \times \boldsymbol{B}).$$

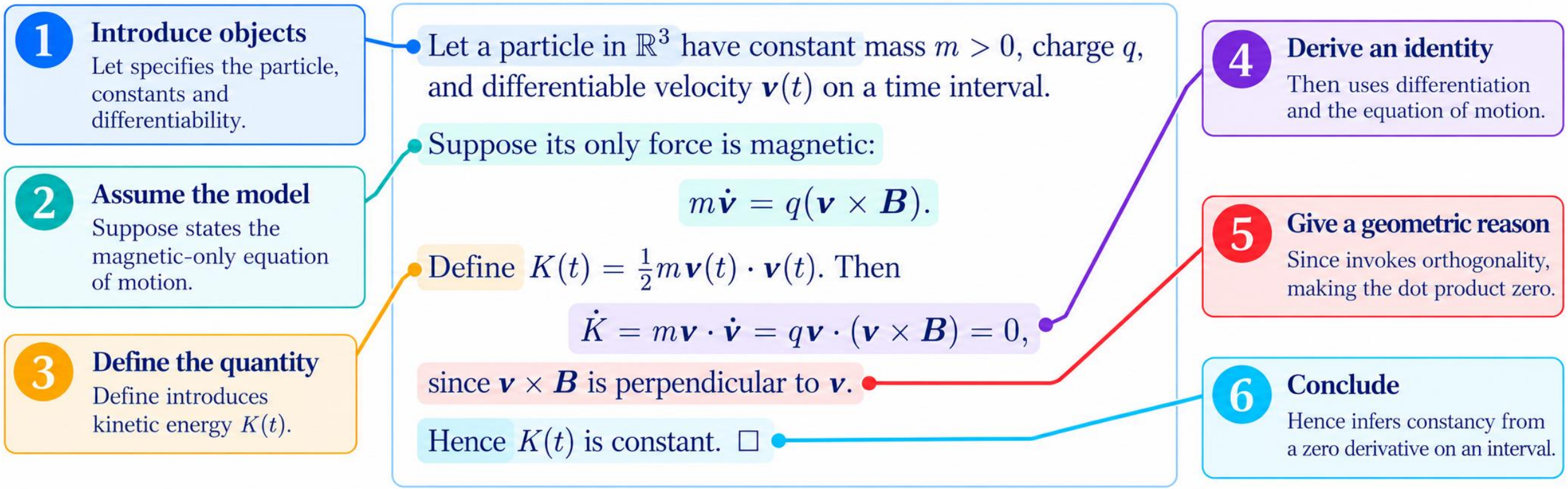
Define $K(t) = \frac{1}{2} m\, \boldsymbol{v}(t) \cdot \boldsymbol{v}(t)$. Then

$$\dot{K} = m\, \boldsymbol{v} \cdot \dot{\boldsymbol{v}} = q\, \boldsymbol{v} \cdot (\boldsymbol{v} \times \boldsymbol{B}) = 0,$$

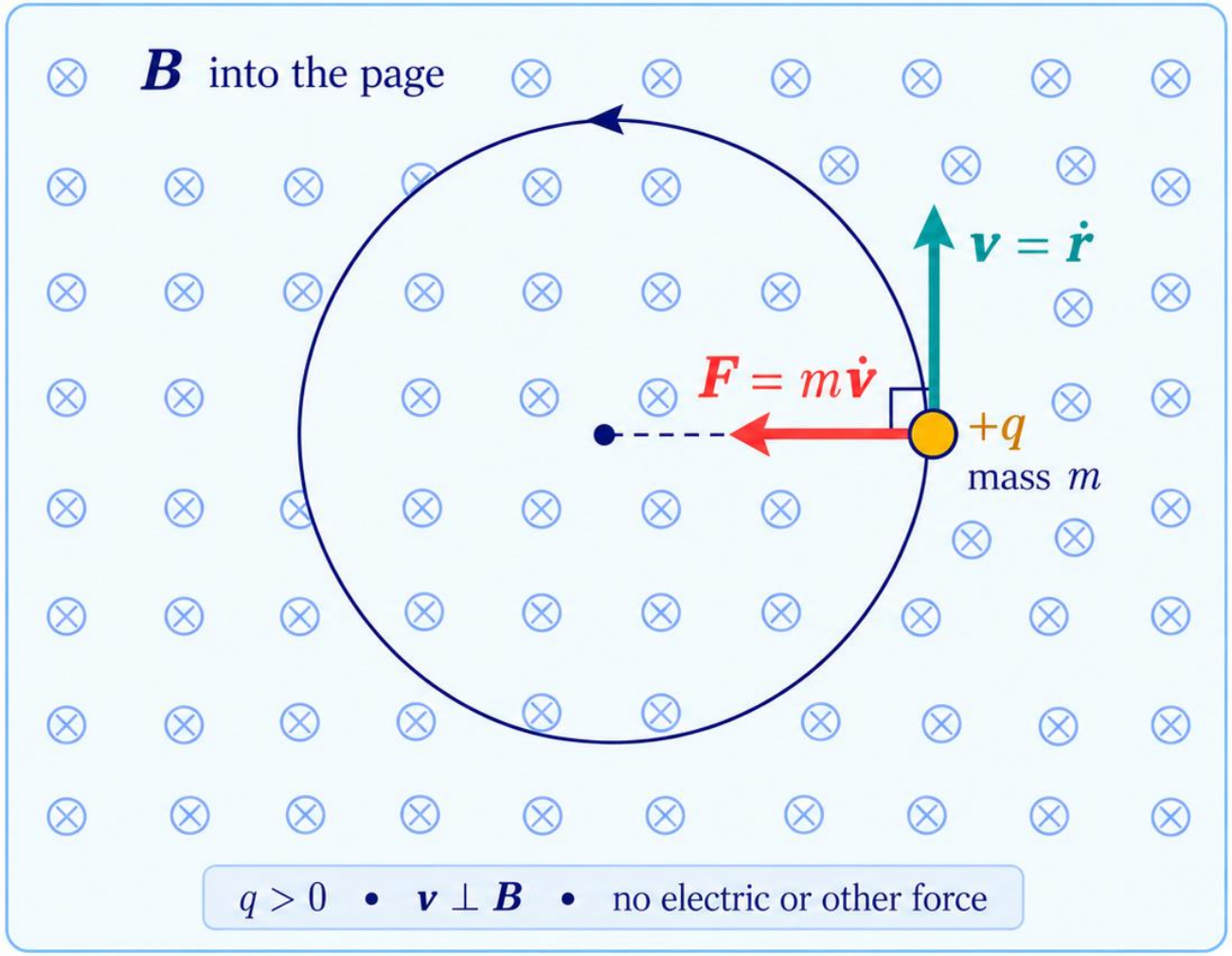
since $\boldsymbol{v} \times \boldsymbol{B}$ is perpendicular to $\boldsymbol{v}$.

Hence $K(t)$ is constant. $\square$

Breaking down language/inferential functions



An example from Physics



1 Magnetic field

$\boldsymbol{B}$: field direction and strength

$\otimes$ points into the page.

2 Charge and mass

q sets the magnetic coupling.

m sets the inertial response.

3 Equation of motion

$$m\dot{\boldsymbol{v}} = \boldsymbol{F} = q(\boldsymbol{v} \times \boldsymbol{B})$$

The force bends the trajectory.

4 Energy

$$\boldsymbol{K} = \tfrac{1}{2} m \|\boldsymbol{v}\|^2$$

$$\dot{\boldsymbol{K}} = \boldsymbol{v} \cdot \boldsymbol{F} = 0$$

Perpendicular force: constant speed.

Value of a Physics derivation

Assumed physical model | $m\dot{\boldsymbol{v}} = q(\boldsymbol{v} \times \boldsymbol{B})$ | Constant $m > 0$; uniform, static $\boldsymbol{B}$; $\boldsymbol{v} \perp \boldsymbol{B}$; $q \neq 0$; no other forces; nonrelativistic motion.

Mathematical correctness

$$K = \frac{1}{2} m \|\boldsymbol{v}\|^2$$

$$\dot{K} = m\boldsymbol{v} \cdot \dot{\boldsymbol{v}} = q\underbrace{\boldsymbol{v} \cdot (\boldsymbol{v} \times \boldsymbol{B})}_{\text{geometric cancellation (perpendicular)}} = 0$$

Therefore K and speed $\|\boldsymbol{v}\|$ are constant.

Does the conclusion follow from the assumptions?

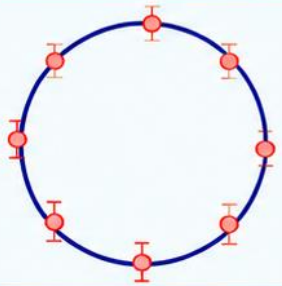
Proof checks the inference.
Lean can check a formalized derivation.
The physical assumptions remain assumptions.

derive predictions →

Empirical predictive power

$$R = \frac{m v_\perp}{|q| B} \qquad \omega = \frac{|q| B}{m}$$

$B = \|\boldsymbol{B}\|$; $v_\perp = \|\boldsymbol{v}\|$ here.



Predicted orbit

Illustrative measurements

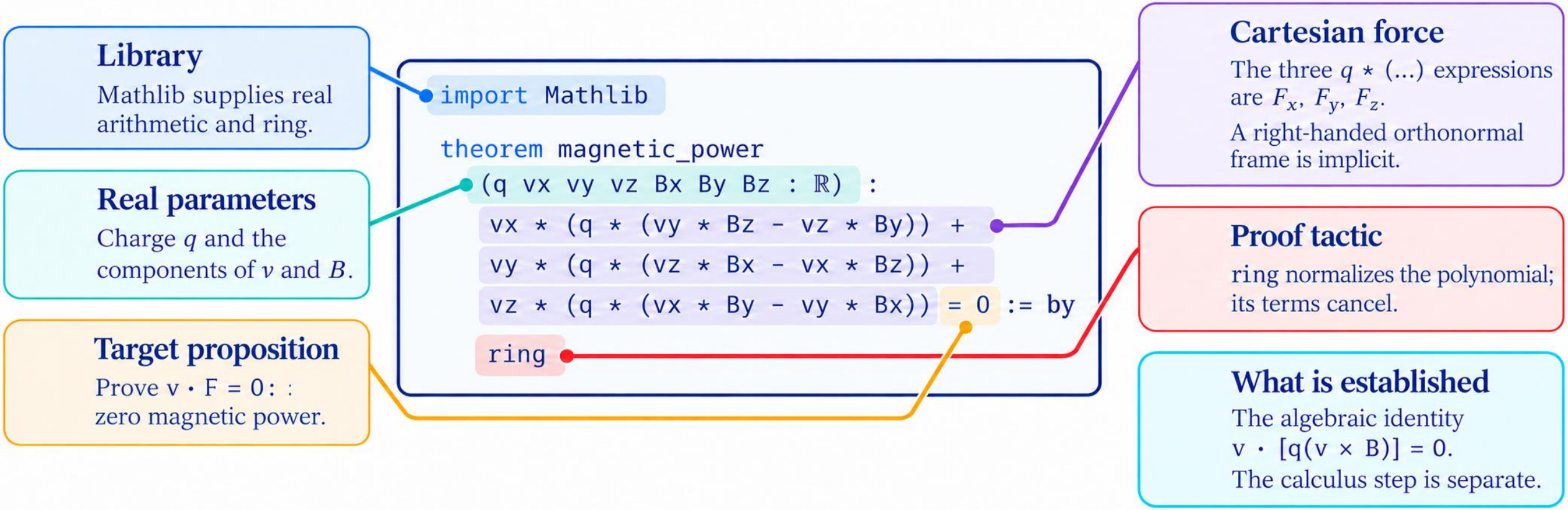
Compare radius, turning rate and speed with observations.

Does the model predict what is measured?

Agreement supports the model in the tested regime.
Disagreement can reveal missing forces or limits of the model.

Breaking down language/inferential functions

Formalizing the algebraic cancellation in the physics proof.



```
import Mathlib
open Set
abbrev V := EuclideanSpace ℝ (Fin 3)
def cross (u v : V) : V :=
  WithLp.toLp 2 (crossProduct u.ofLp v.ofLp)
```

Euclidean vector types

V carries the Euclidean norm and inner product in three dimensions. cross adapts Mathlib's cross product to this Euclidean representation.

```
lemma inner_cross (u v : V) :
    inner ℝ u (cross u v) = 0 := by
  rw [EuclideanSpace.inner_eq_star_dotProduct]
  simp only [cross, WithLp.ofLp_toLp, star_trivial]
  rw [dotProduct_comm]
  exact dot_self_cross u.ofLp v.ofLp
```

Geometric orthogonality

The cross product is perpendicular to its first argument. An existing Mathlib theorem proves the zero inner product.

```
theorem magnetic_energy
    (m q l r : ℝ) (_hm : 0 < m) (v a B : ℝ → V)
    (hd : ∀ t ∈ Ioo l r, HasDerivAt v (a t) t)
```

Curves and derivatives

v, a, B are vector-valued curves. Mass m and charge q are constant. HasDerivAt states that a is the derivative of v on (l, r).

```
    (hf : ∀ t ∈ Ioo l r,
      m • a t = q • cross (v t) (B t))
    (s t : ℝ) (hs : s ∈ Ioo l r) (ht : t ∈ Ioo l r) :
    m / 2 * ‖v s‖ ^ 2 = m / 2 * ‖v t‖ ^ 2 := by
```

Dynamics and target

Assume the total force is q(v × B). Scalar multiplication uses •. The target is equal kinetic energy at any two times in the interval.

```
  let K := fun u => m / 2 * ‖v u‖ ^ 2
  have hK : ∀ u ∈ Ioo l r, HasDerivAt K 0 u := by
    intro u hu
    have hpower : m * inner ℝ (v u) (a u) = 0 := by
      have h := congrArg (fun w => inner ℝ (v u) w) (hf u hu)
      simpa only [inner_smul_right, inner_cross, mul_zero] using h
```

Energy and magnetic power

K uses the squared Euclidean norm. Take the inner product of the force equation with v. Orthogonality gives m ⟨v, a⟩ = 0.

```
    convert (hd u hu).norm_sq.const_mul (m / 2) using 1
    dsimp
    nlinarith [hpower]
```

Vector calculus

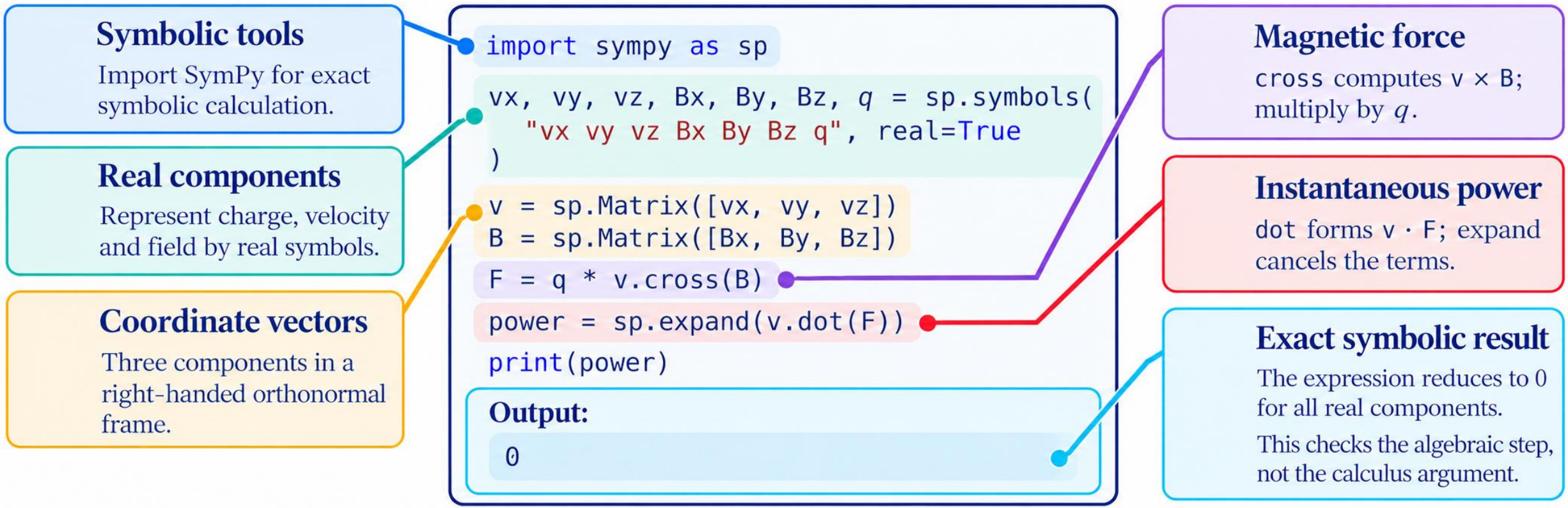
norm_sq differentiates ‖v‖². const_mul supplies the factor m/2. The result is K′ = m ⟨v, a⟩ = 0.

```
  exact isOpen_Ioo.is_const_of_deriv_eq_zero
    (convex_Ioo l r).isPreconnected
    (fun u hu => (hK u hu).differentiableAt.differentiableWithinAt)
    (fun u hu => (hK u hu).deriv) hs ht
```

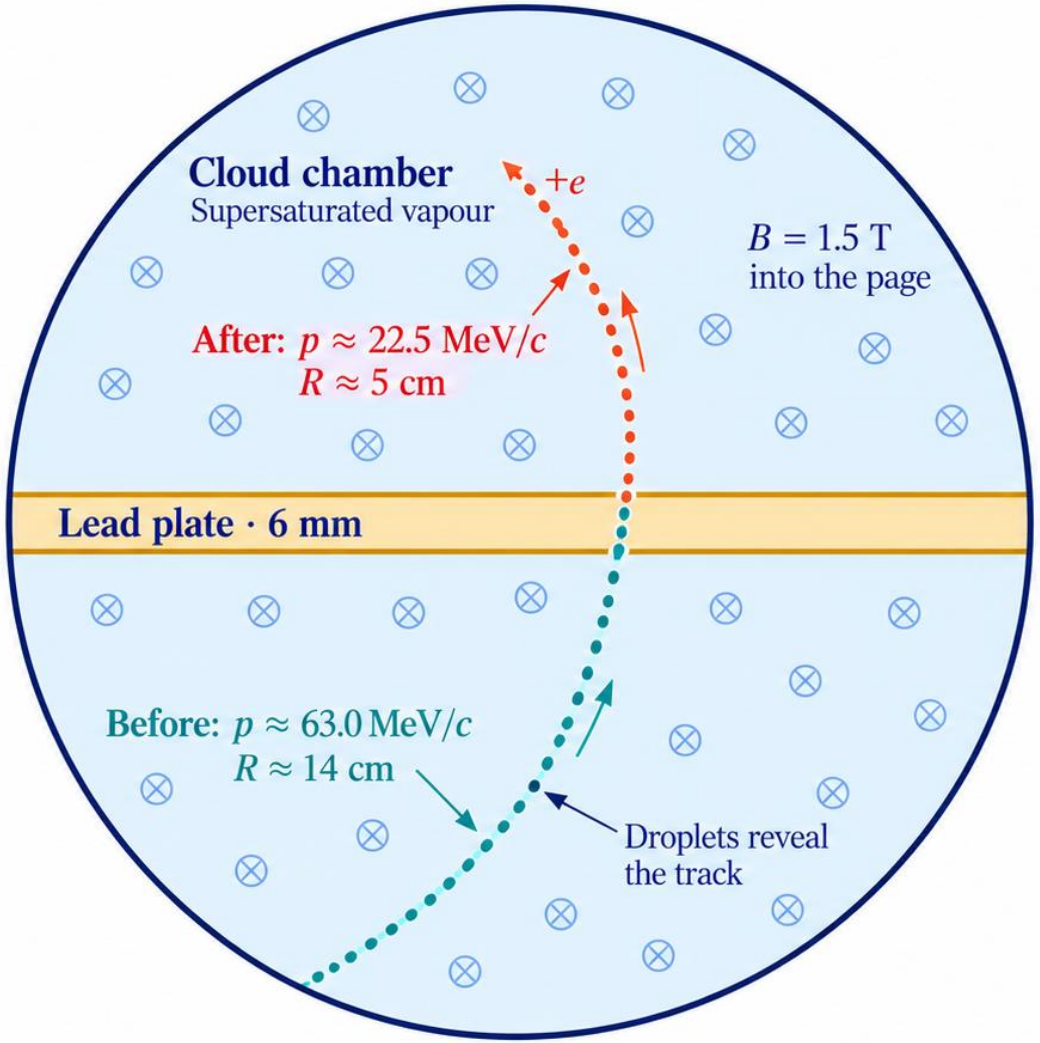
Energy is constant

The interval is open and connected. Mathlib's zero-derivative theorem concludes K(s) = K(t).

Sympy version



Relation to the predictive dimension



Apparatus

Magnetic field bends the track.
Vapour makes the ionization trail visible.
Lead reduces particle momentum.

Calculation

$p = |q|BR$
For $|q| = e$:
p [MeV/c] $\approx 299.8\ B$ [T] R [m]
Smaller R means smaller momentum.

Inference

Tighter curvature follows the lead crossing.
Energy loss determines upward travel.
Field direction and bending give positive charge.

Relation to the predictive dimension

After lead

$BR \approx 0.075$ T m

$R \approx 5$ cm

$p \approx 22.5$ MeV/c

Tighter curvature: lower momentum.

Lead plate

Thickness: 6 mm

Energy is lost through interactions with matter.

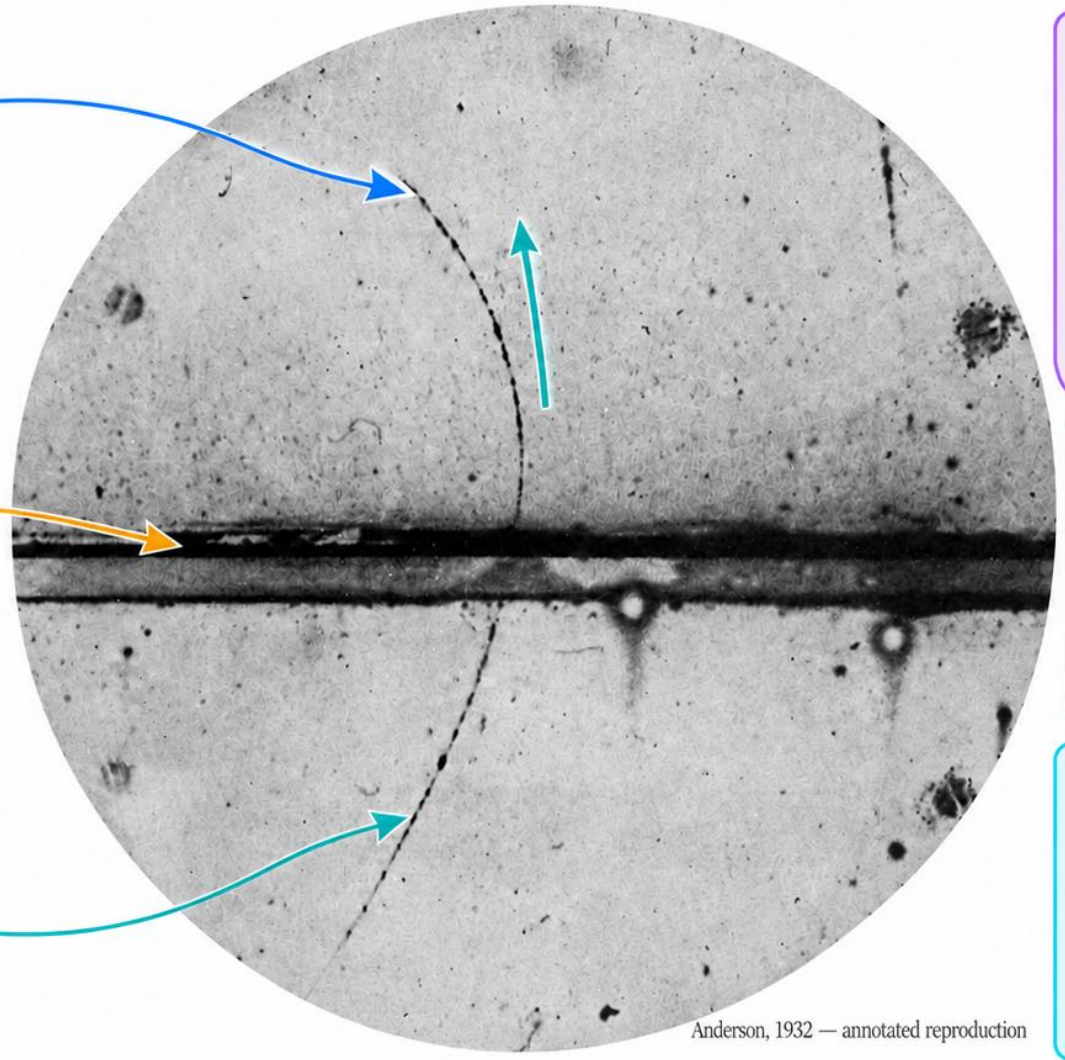
Before lead

$BR \approx 0.21$ T m

$R \approx 14$ cm

$p \approx 63.0$ MeV/c

Broader curvature: higher momentum.



Anderson, 1932 — annotated reproduction

Model and calculation

$B = 1.5$ T; assume $|q| = e$

$p = |q|BR$

p [MeV/c] $\approx 299.8\ B$ [T] R [m]

Before: $299.8 \times 1.5 \times 0.14 \approx 63.0$

After: $299.8 \times 1.5 \times 0.05 \approx 22.5$

Prediction confronted with evidence

Energy loss predicts tighter curvature after the plate.

The photograph shows this change.

Confirmatory inference

Energy loss establishes upward travel.

Travel direction + known field + bending imply positive charge.

Track length and ionization help exclude a proton.



Mathematical reasoning:
Architectural patterns

Two major entangled reasoning regimes

- Verification
- Discovery

Two major reasoning regimes

> Verification

- Emphasis is on **correctness**, **elicitation**
- **Logical validity** (via formalization)

Reasoning signature

(Generative & symbolic)

formal-based
(auto-) formalization,
retrieval of formal statements,
library binding
semantic preservation,
syntactic correctness,
formal validity,
deductive

Real-world mathematical verification

is surjective. For this, consider the map

$$S^n \times \{-1,0,1\}^n = \varprojlim_i S_i^n \times \{-1,0,1\}^n \to \varprojlim_i \mathbb{Z}[S_i]_{\leq n}$$

given by $(x_1,\dots,x_n,a_1,\dots,a_n) \mapsto a_1[x_1]+\dots+a_n[x_n]$. This is a cofiltered limit of surjections of finite sets, and so a surjection of profinite sets. This map sits in a commutative diagram

$$\begin{array}{ccc} S^n \times \{-1,0,1\}^n & \longrightarrow & \varprojlim_i \mathbb{Z}[S_i]_{\leq n} \\ \downarrow & & \downarrow \\ \mathbb{Z}[S] & \longrightarrow & \bigcup_m \varprojlim_i \mathbb{Z}[S_i]_{\leq m} \end{array}$$

and so implies that the lower map contains $\mathbb{Z}[S_i]_{\leq n}$ in its image. As this works for any n, we get the desired result. □

REMARK 2.2. In particular, we see that the condensed set $\mathbb{Z}[S]$ is quasiseparated, and in fact comes from a compactly generated weak Hausdorff topological space $\mathbb{Z}[S]_{\text{top}}$, by Proposition 1.2 (4). Moreover, by Lemma 1.3, the addition

$$\mathbb{Z}[S]_{\text{top}} \times \mathbb{Z}[S]_{\text{top}} \to \mathbb{Z}[S]_{\text{top}}$$

is continuous (not only when the source is equipped with its compactly generated topology), so $\mathbb{Z}[S]$ really comes from a topological abelian group $\mathbb{Z}[S]_{\text{top}}$.

EXERCISE 2.3. Prove that for any compact Hausdorff space S, the condensed abelian group $\mathbb{Z}[S]$ can naturally be written as a countable union $\bigcup_n \mathbb{Z}[S]_{\leq n}$ of compact Hausdorff spaces $\mathbb{Z}[S]_{\leq n}$, and comes from a topological abelian group $\mathbb{Z}[S]_{\text{top}}$.

The idea of solid modules is that we would want to enlarge $\mathbb{Z}[S]$, allowing more sums deemed "convergent", and an obvious possibility presents itself:

$$\mathbb{Z}[S]^\blacksquare := \varprojlim_i \mathbb{Z}[S_i].$$

In particular, this ensures that

$$\mathbb{Z}[\mathbb{N}\cup\{\infty\}]^\blacksquare/([\infty]=0) = \varprojlim_n \mathbb{Z}[\{0,1,\dots,n-1,\infty\}]/([\infty]=0) = \varprojlim_n \mathbb{Z}[T]/T^n = \mathbb{Z}[[T]]$$

is the power series algebra.[6]

In other words, we want to pass to a subcategory Solid $\subset$ Cond(Ab) with the property that the free solid abelian group on a profinite set S is $\mathbb{Z}[S]^\blacksquare$. This is codified in the following definition.

DEFINITION 2.4. A condensed abelian group M is solid if for any profinite set S with a map $f : S \to M$, there is a unique map of condensed abelian groups $\widetilde{f} : \mathbb{Z}[S]^\blacksquare \to M$ such that the composite $S \to \mathbb{Z}[S]^\blacksquare \to M$ is the given map f.

Let us immediately state the main theorem on solid abelian groups.

[6]Here, $\mathbb{Z}[T]$ denotes the polynomial algebra in a variable T, not the free condensed module on a profinite set T.

Very complex!

Highly specialized

Did I **do everything** right?

x100



Real-world mathematical verification

```
variables (p' p : ℝ≥0) [fact (0 < p')] [fact (p' < p)] [fact (p ≤ 1)]

theorem liquid_tensor_experiment
  (S : Profinite) (V : pBanach p) :
  ∀ i > 0, Ext i (𝓜_{p'} S) V ≅ 0 :=
-- the proof ...
```



x100

Formalization

mathlib

The Liquid Tensor Experiment

Solver

> 12 specialists
~ 18 months

- material inferences
- incomplete
- potentially incorrect

https://github.com/leanprover-community/lean-liquid

Two major reasoning regimes

> Discovery

- Identifying **conceptual tensions**
- Producing new **hypotheses** and **abstractions**
- Emphasis is on **cohered** mathematical arguments
- **Bridging** existing concepts, theorems, axioms, etc.

Reasoning signature

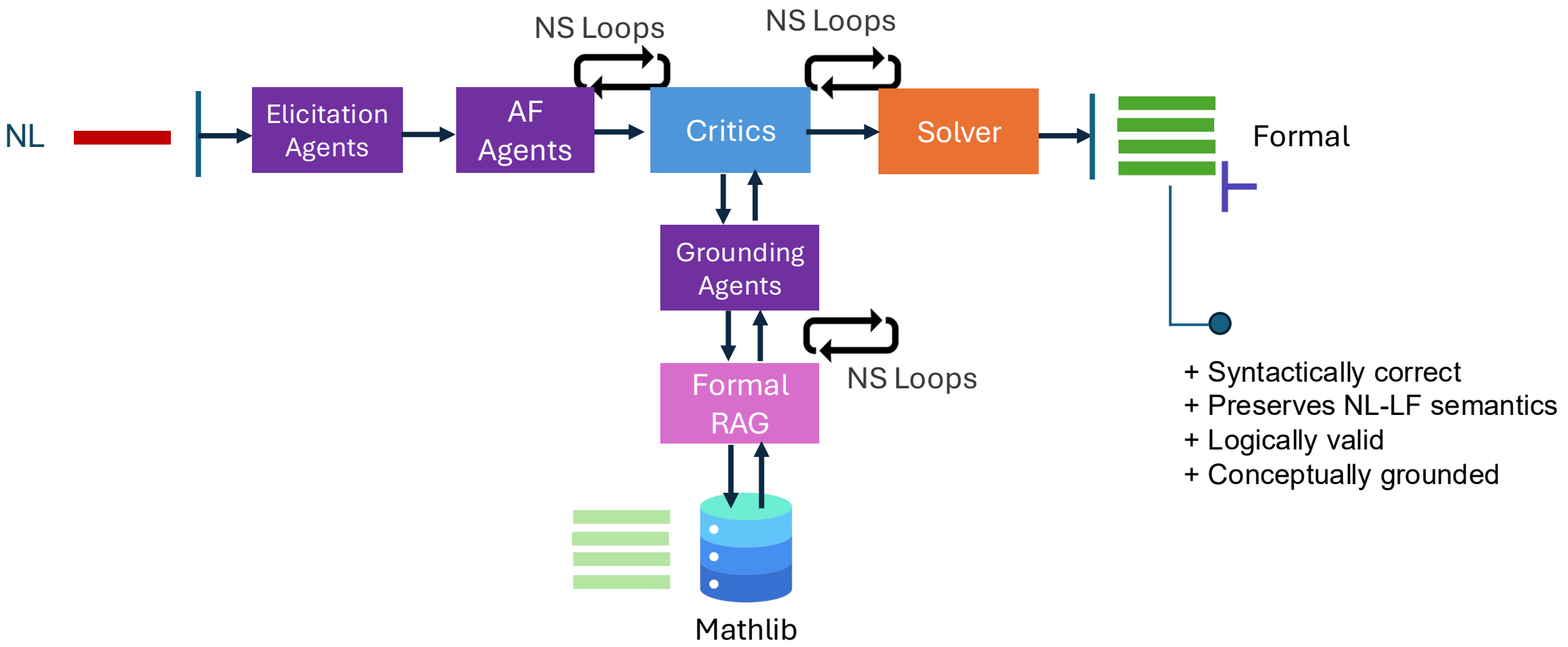
(Generative dominated)

Content-based, conceptual interpretation, broader retrieval, explanatory/abductive

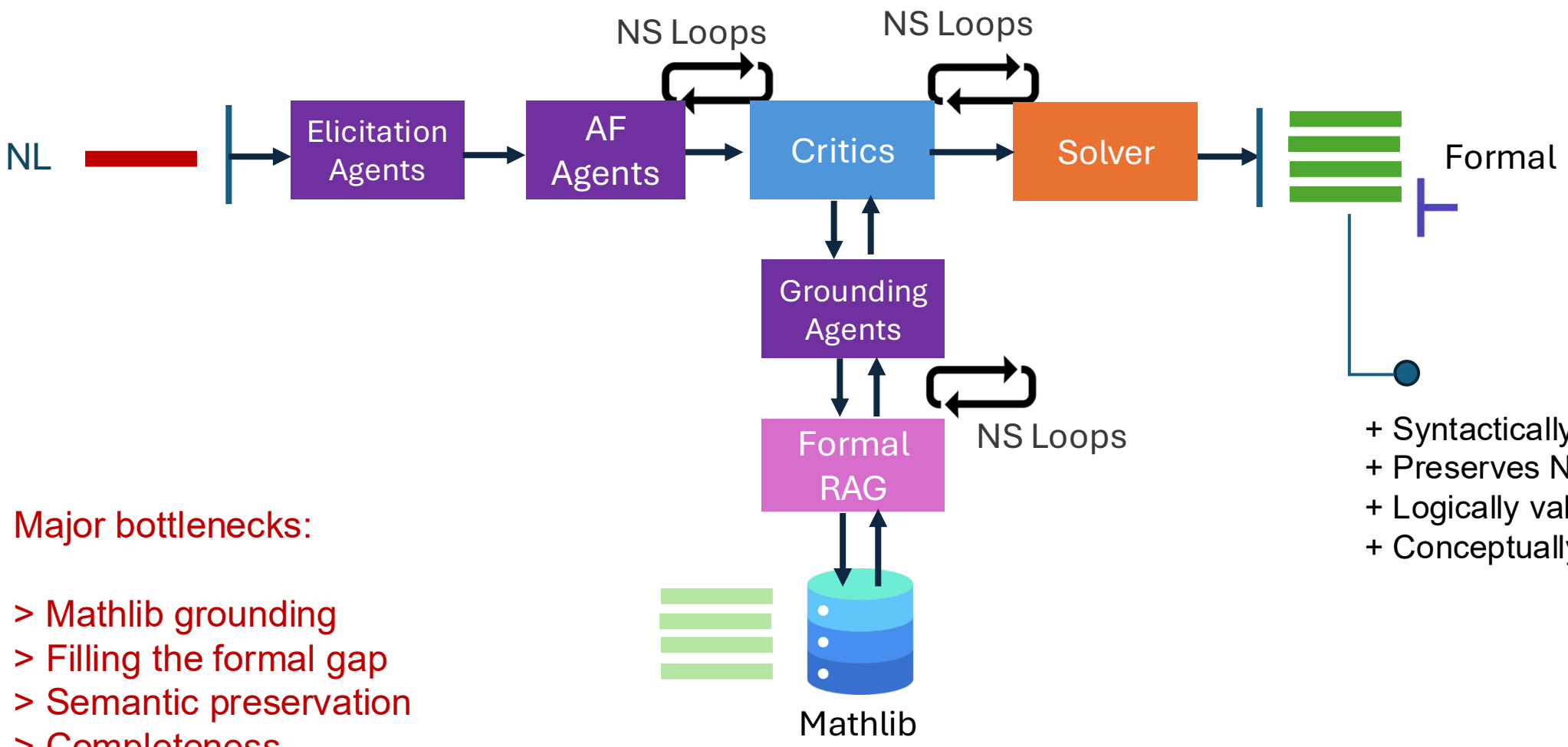
Architectural paradigms

- Multi-agent inference-time
- Model training/steering
  - Pre-training/SFT/RL
  - Steering methods

Verification & mathematical autoformalization (broad strokes)



Verification & mathematical autoformalization (broad strokes)



+ Syntactically correct
+ Preserves NL-LF semantics
+ Logically valid
+ Conceptually grounded

Major bottlenecks:

> Mathlib grounding
> Filling the formal gap
> Semantic preservation
> Completeness

lunar

Package version

v7 · Matl

Runtime: ready

- Home
- Chat
- Data Sources
- AI Prompts
- Reports
- Templates
- AI Models
- Repository
- Prism
- Skills

D danilo@lunarbas...

Skills

Search skills...

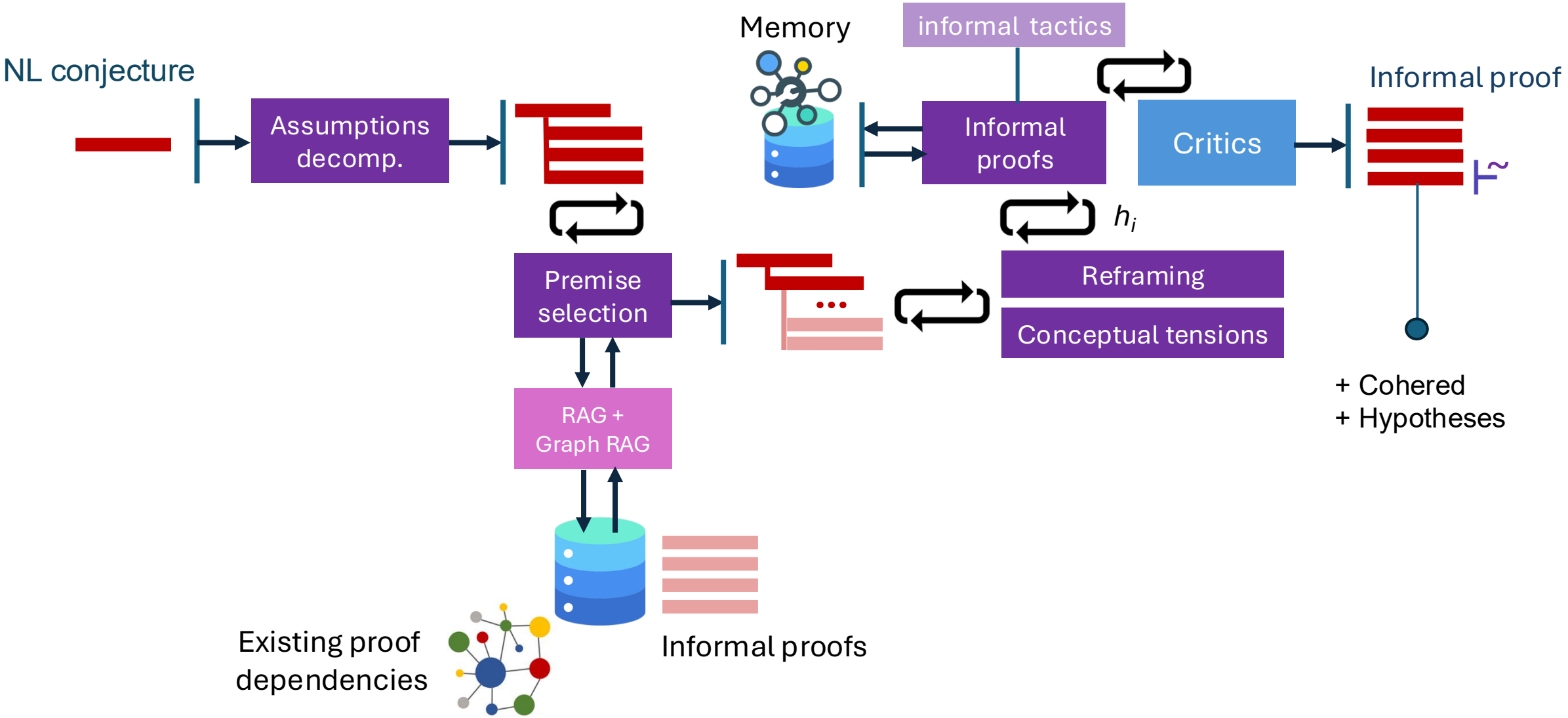
188 skills

Atp	2 skills
Autoformalization	13 skills
Benchmarks	1 skill
Control	8 skills
Isabelle	4 skills
Latex Textbook	8 skills
Lean	21 skills
Mathformerlib	4 skills
Mathformerphysicslib	13 skills
Mathlib	22 skills
Paper	3 skills
Physics Common	24 skills
Physics Isabelle	6 skills
Physics Lean	24 skills

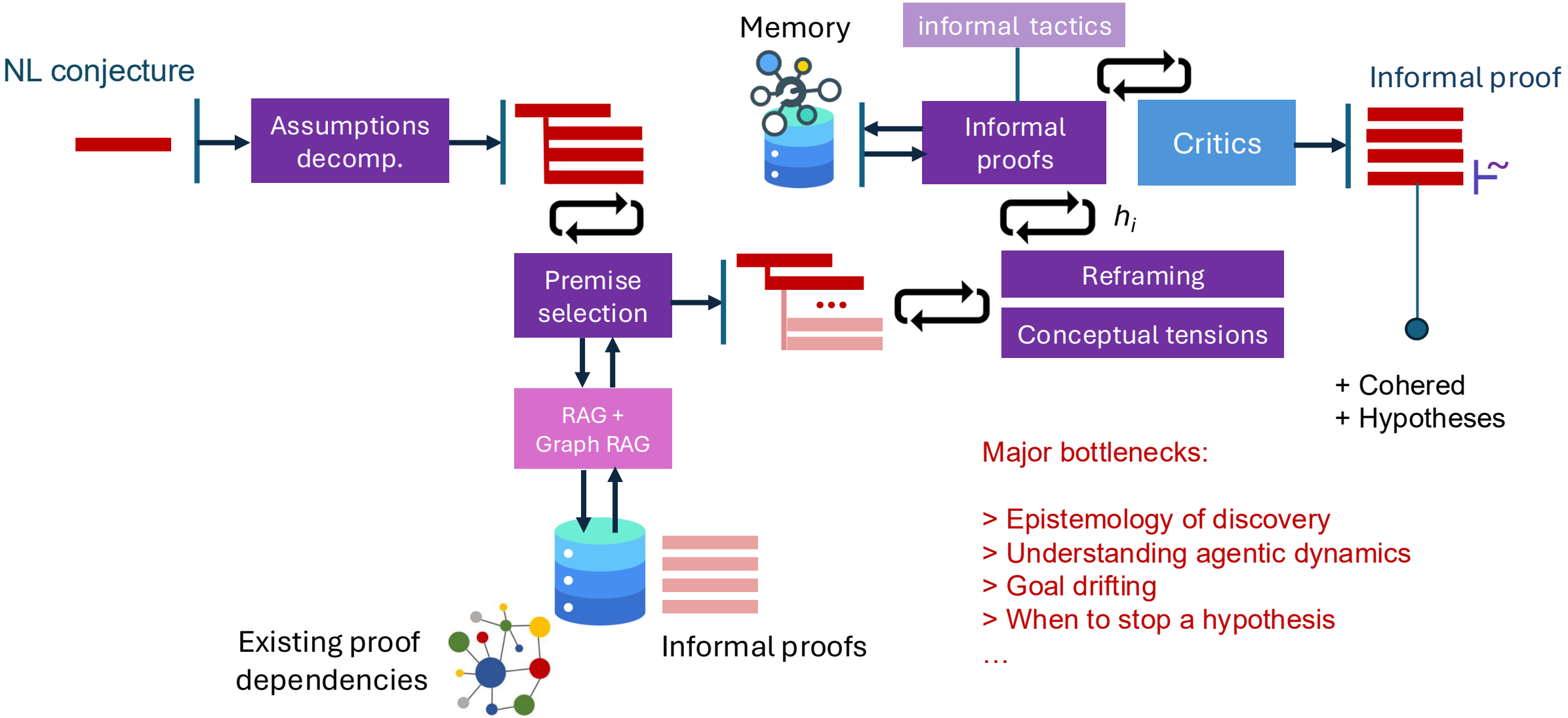


https://github.com/idiap/lunar

Discovery (broad strokes)



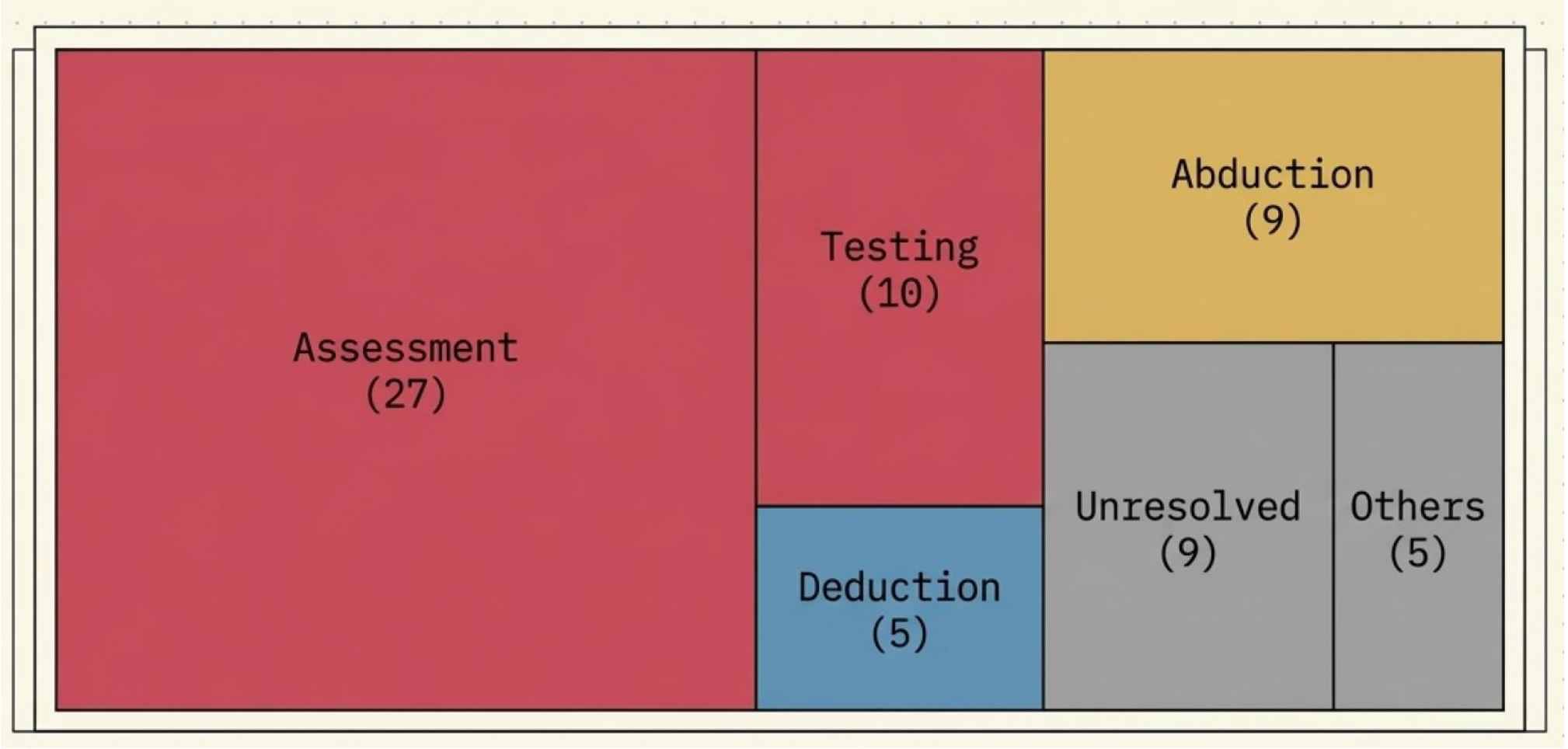
Discovery
(broad strokes)



Major bottlenecks:

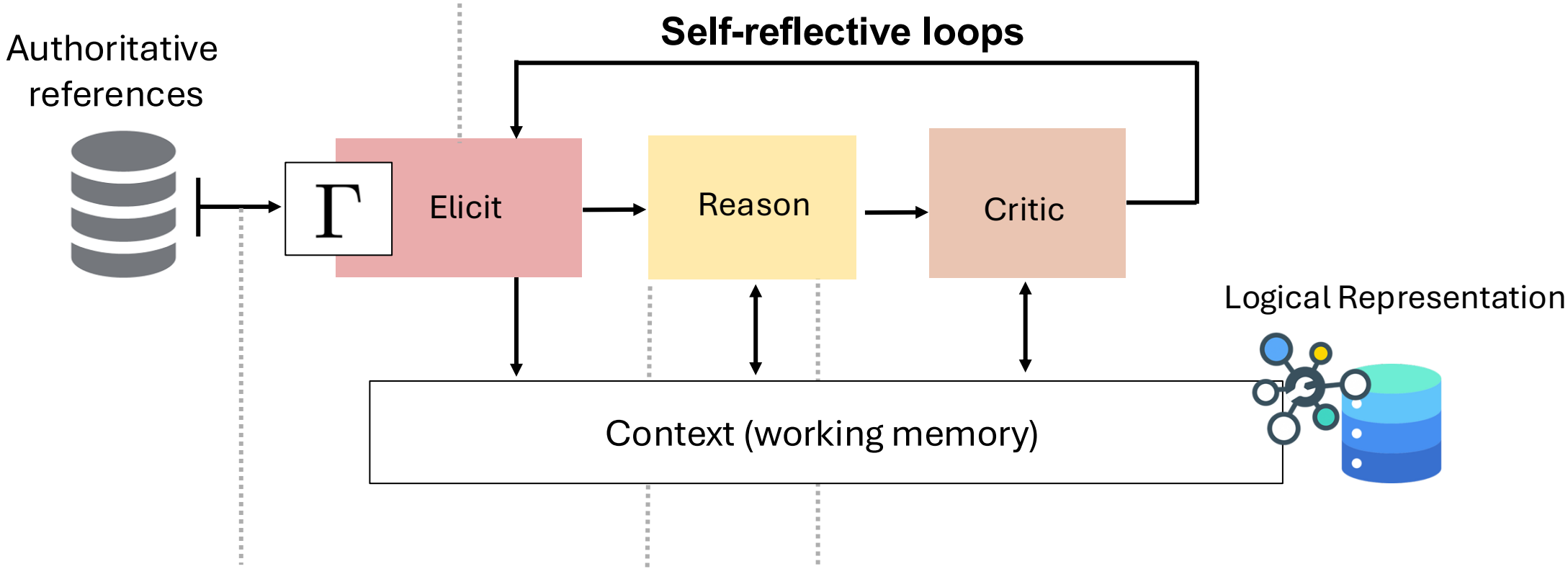
> Epistemology of discovery
> Understanding agentic dynamics
> Goal drifting
> When to stop a hypothesis
…

Most agentic workload is on verification



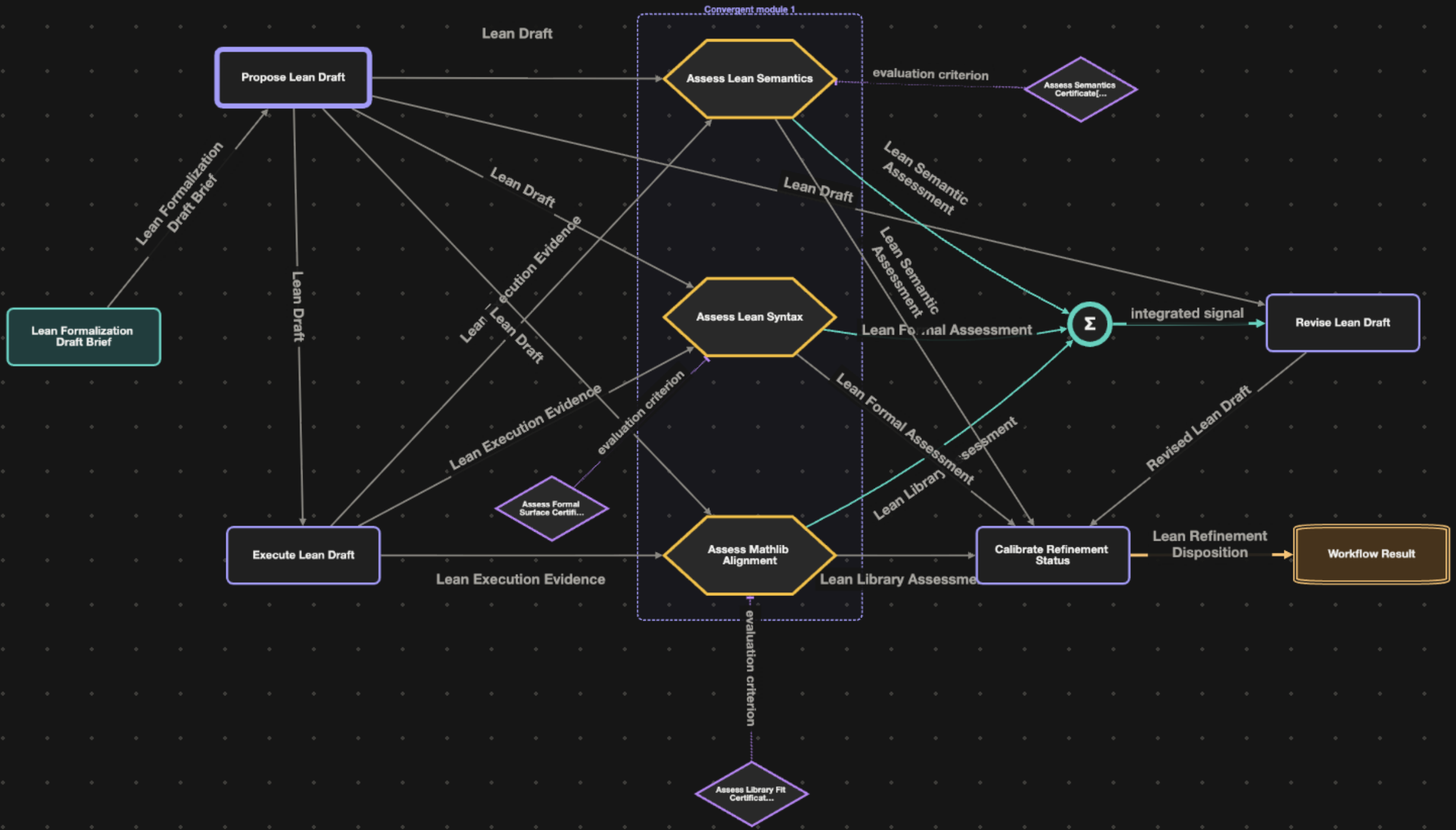
CQs:
- Is all background knowledge present?
- Is it necessary and sufficient?

Agentic systems are self-critical

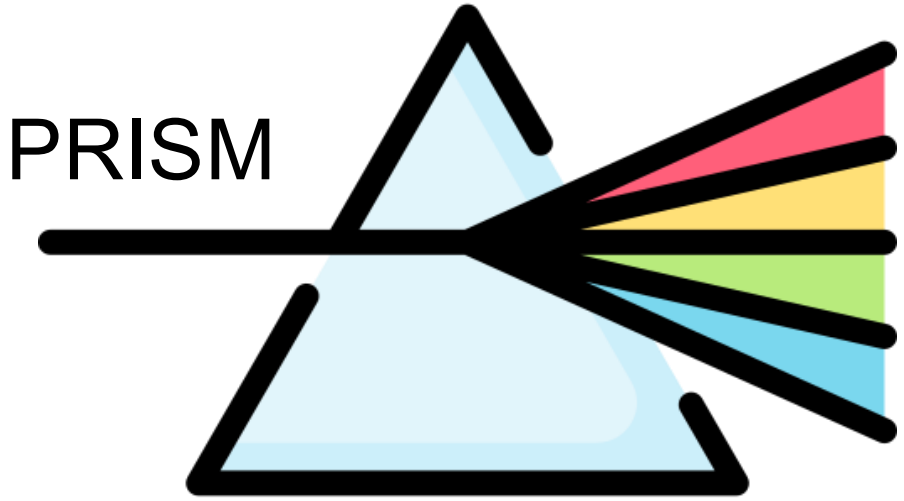


CQs:
- Are all sources for facts and definitions available?
- Are the sources authoritative and up to date?

CQs:
- Have we **decomposed** all specialized inferences?
- Have we specified and integrated all external tools?
- Are they described systematically and completely?
- Have we critically defined the validity conditions?


Convergent module 1
Lean Draft
Propose Lean Draft
Assess Lean Semantics
evaluation criterion
Assess Semantics Certificate[...
Lean Formalization Draft Brief
Lean Formalization Draft Brief
Lean Draft
Lean Draft
Lean Draft
Lean Semantic Assessment
Lean Semantic Assessment
Lean Execution Evidence
Assess Lean Syntax
Σ
integrated signal
Revise Lean Draft
Lean Formal Assessment
Lean Draft
Lean Execution Evidence
evaluation criterion
Lean Formal Assessment
Revised Lean Draft
Assess Formal Surface Certifi...
Lean Library
Execute Lean Draft
Assess Mathlib Alignment
Calibrate Refinement Status
Lean Refinement Disposition
Workflow Result
Lean Execution Evidence
Lean Library Assessme
evaluation criterion
Assess Library Fit Certificat...

Conceptualizing & managing complex inferences



https://github.com/idiap/prism

```
esolve a Lean concept demand against a fixed mathlib revision, verify the
eclarations that actually fit, and produce a usable API and import plan.
soning GroundLeanAPIs(
 workflow_source: LeanFormalizationConceptDemand,
> Generated[GroundingDisposition]:
 sequence:
     parallel:
         # Resolve requested concepts, ambiguity, proof roles, and navigation bounds.
         [concept_model: ModelConceptDemand(workflow_source)]
         # Observe a version-stamped declaration and import snapshot.
         [mathlib_snapshot: InspectMathlibSnapshot(workflow_source)]
     # Search local indexes, hierarchy neighborhoods, and official-source fallbacks.
     [candidates: RetrieveMathlibCandidates(MathlibSearchBasis(
         concept_model = concept_model.value,
         mathlib_snapshot = mathlib_snapshot.value,
     ))] by RetrieveCandidates
     # Verify snapshot-bound declaration applicability and retain the supporting evidence.
     [grounding_evidence: VerifyMathlibGrounding(GroundingVerificationBasis(
         concept_model = concept_model.value,
         mathlib_snapshot = mathlib_snapshot.value,
         candidates = candidates.value,
     ))] by GroundCandidates
     # Assess whether every demanded concept has verified coverage or an explicit gap.
     [coverage_check: AssessConceptCoverage(MathlibCoverageCase(
         concept_model = concept_model.value,
```

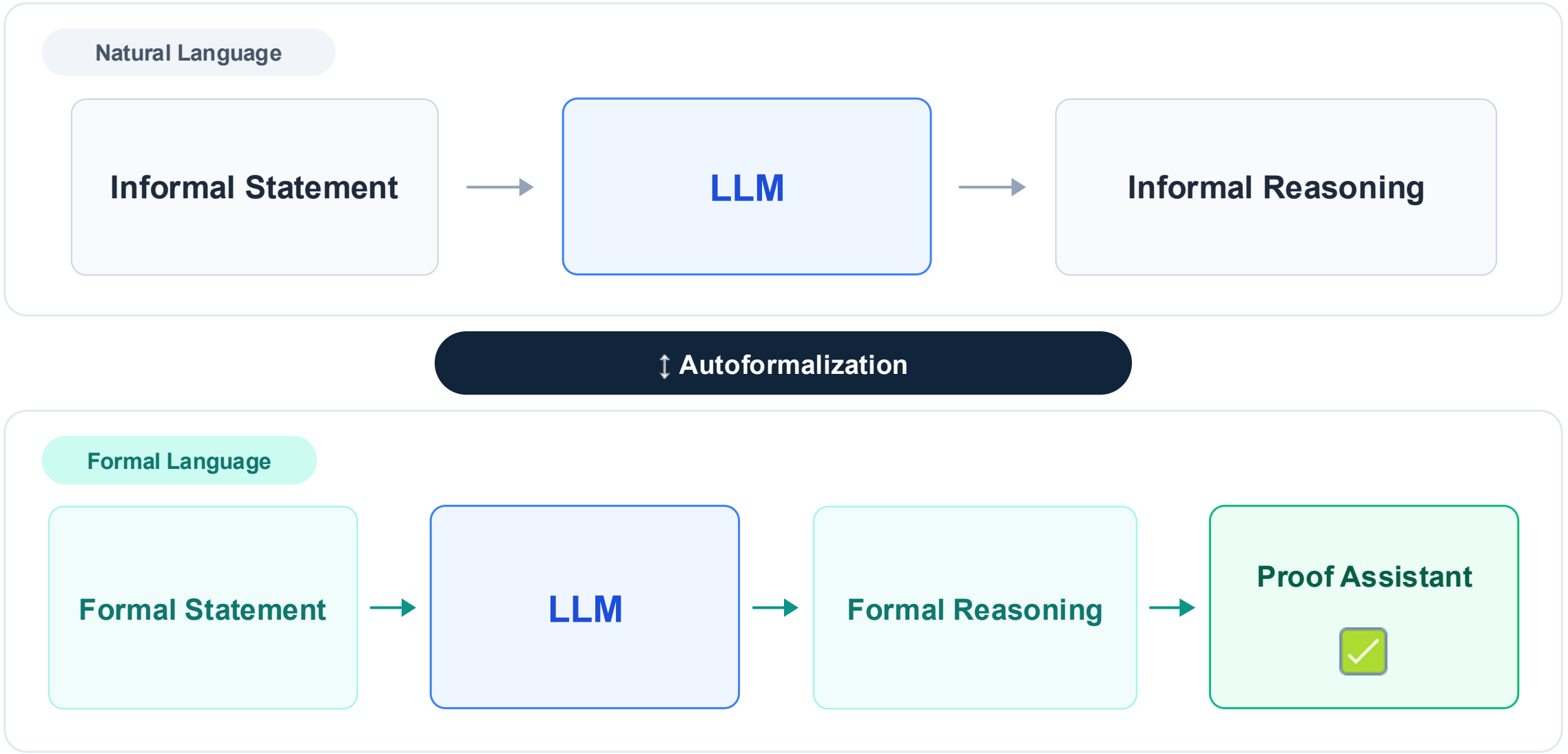


⊢ Demonstration

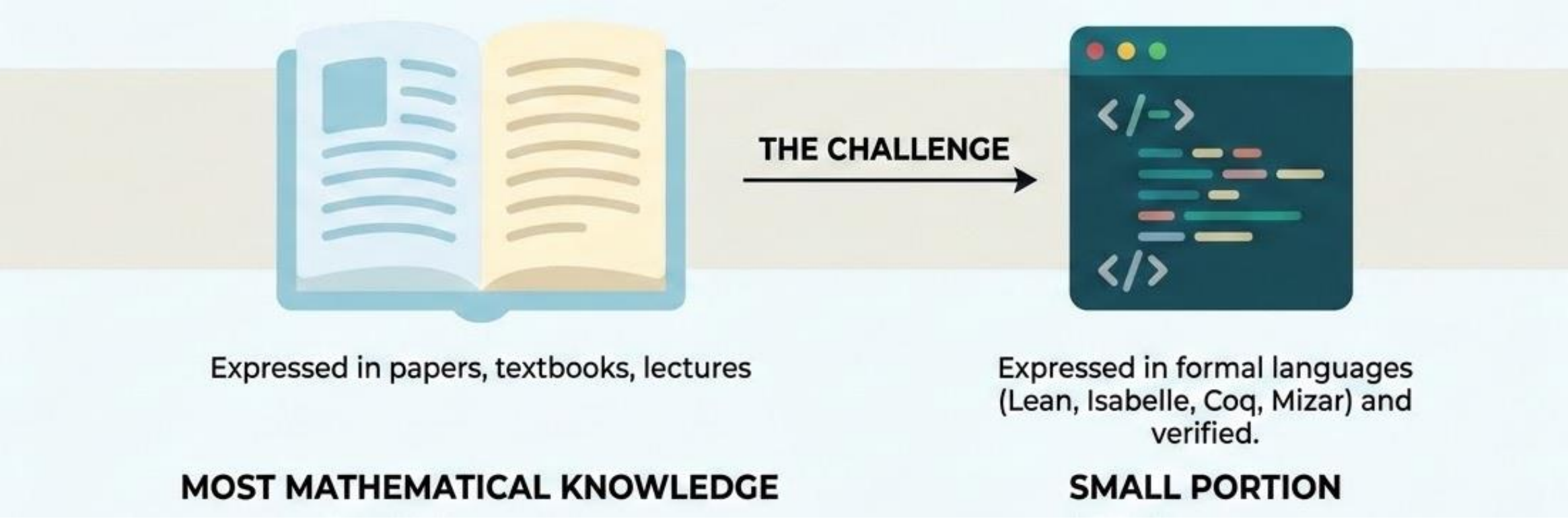


Methods

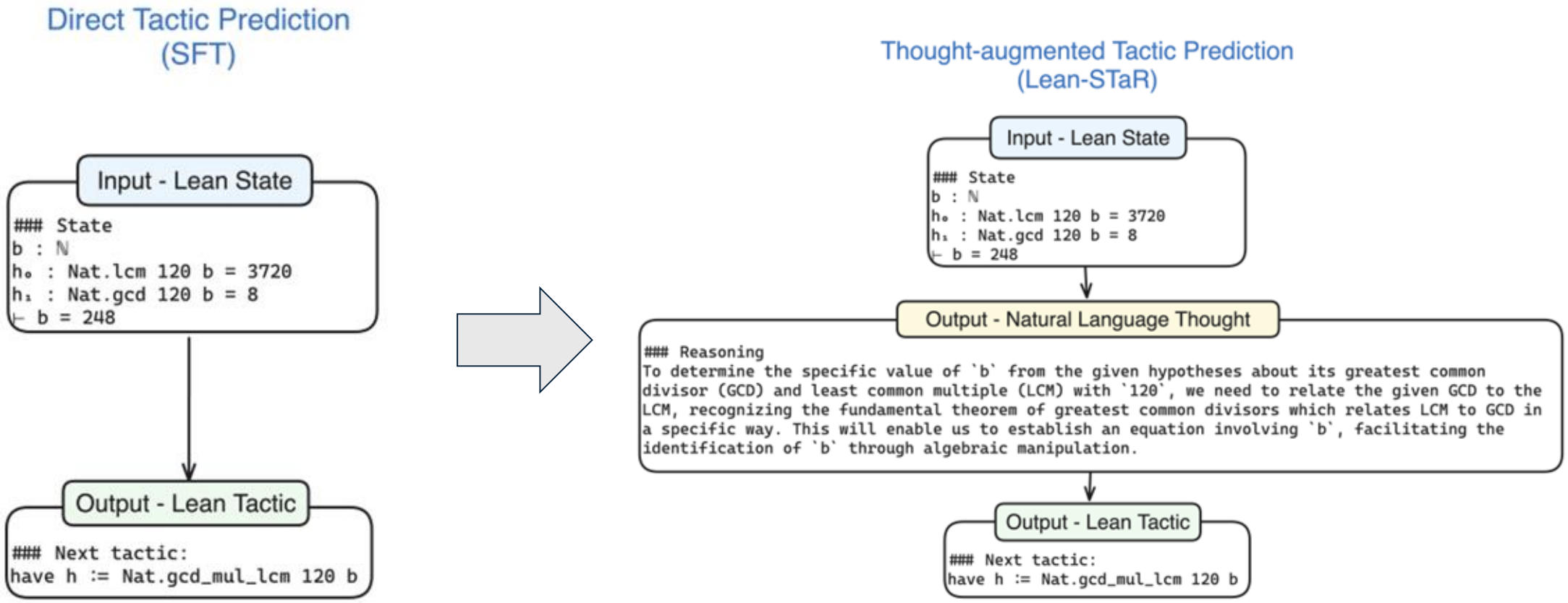
The Interplay Between Informal and Formal Reasoning



The Interplay Between Informal and Formal Reasoning

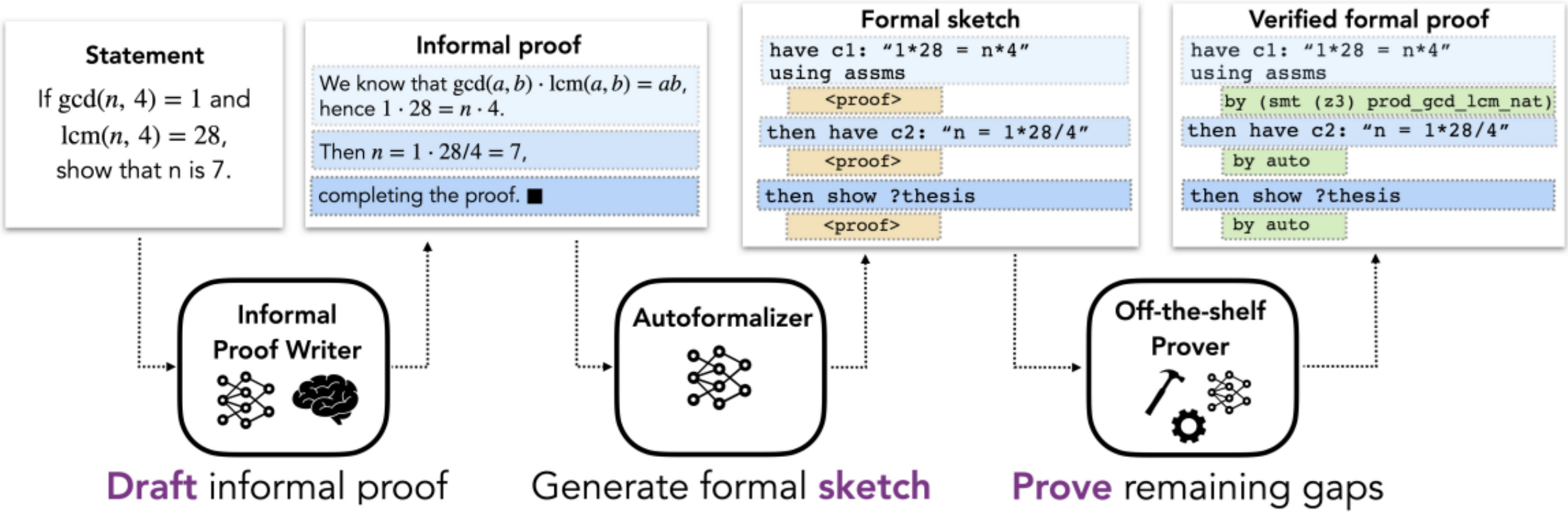


The Interplay Between Informal and Formal Reasoning



(Lin et al., 2025)

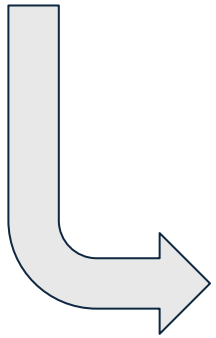
The Interplay Between Informal and Formal Reasoning



(Jiang et al., 2023)

Informal Statement: Show that for any real number a, $10a \leq 28a^2 + 1$.

Informal Proof:
It suffices to show $0 < 28a^2 - 10a + 1$. First, consider completing the square for $28a^2 - 10a$ and observe that $\left(a - \frac{5}{28}\right)^2 = a^2 - \frac{10}{28}a + (5/28)^2$. Since $0 \leq \left(a - \frac{5}{28}\right)^2$, we get $0 \leq a^2 - \frac{10}{28}a + (5/28)^2$. Multiplying by 28 and simplifying gives $0 \leq 28a^2 - 10a + (25/28)$. Since $25/28 < 1$, the result follows.



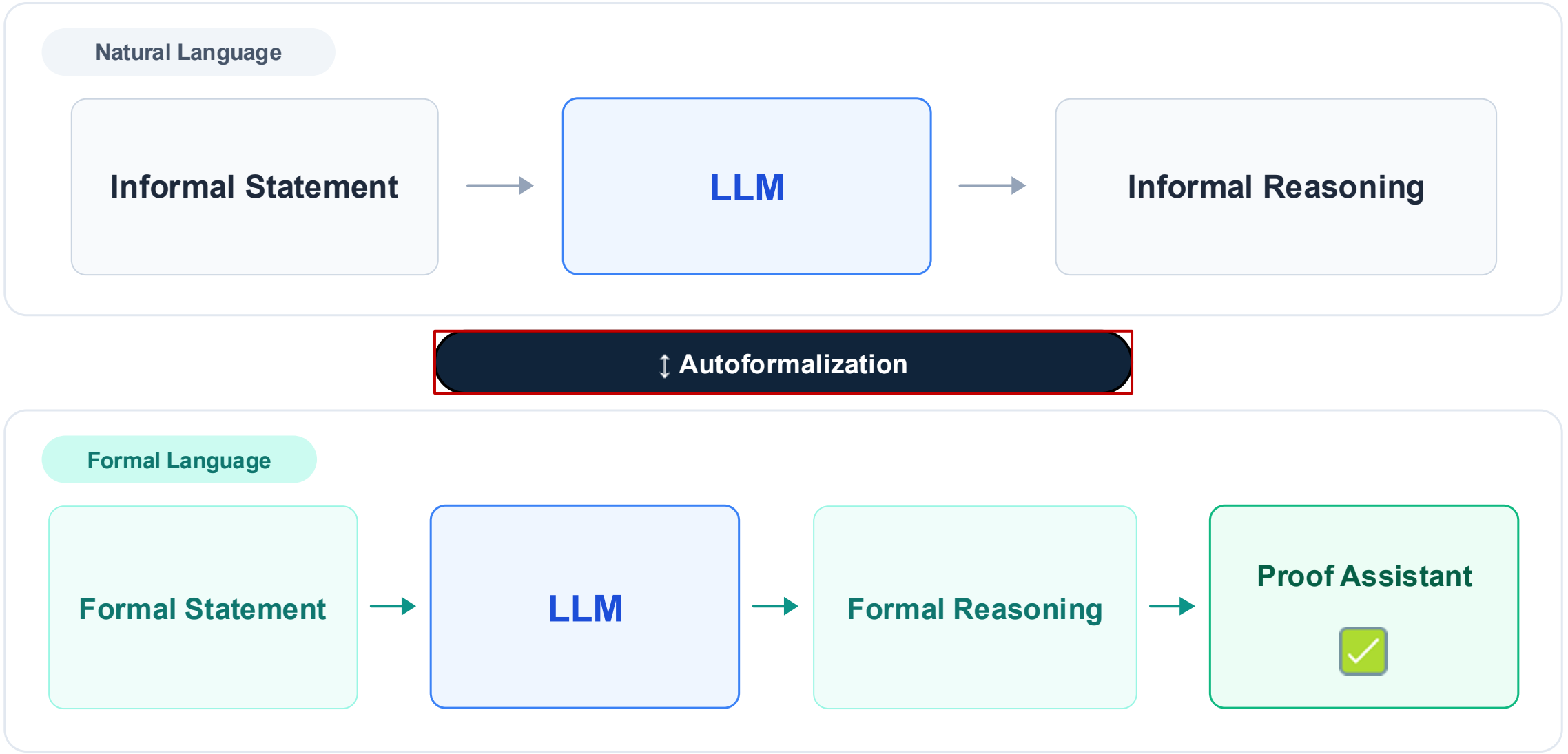
Autoformalization

Formal Statement:

```
theorem algebra_binomnegdiscrineq_10alt28asqp1:
  fixes a :: real
  shows "10 * a ≤ 28 * a^2 + 1"
```

Formal Proof Sketch:

```
proof - (* it suffices to show 0 <= 28a^2 - 10a + 1 *)
  have c0: "0 ≤ 28a^2 - 10a + 1"
  proof - (* observe that (a - (5/28))^2 = a^2 - (10/28)a + (5/28)^2 *)
    have c1: "(a - (5/28))^2 = a^2 - 10/28a + (5/28)^2" < ··· >
    (* we get 0 <= a^2 - (10/28)a + (5/28)^2 *)
    have c2: "0 ≤ a^2 - 10/28a + (5/28)^2" using c1 < ··· >
    (* Multiplying by 28 and simplifying gives 0 <= 28a^2 - 10a + (25/28) *)
    have c3: "0 ≤ 28a^2 - 10a + 28((5/28)^2)" using c2 < ··· >
    have c4: "0 ≤ 28a^2 - 10a + 28((5/28)*(5/28))" using c3 < ··· >
    have c5: "0 ≤ 28a^2 - 10a + (25/28)" using c4 < ··· >
    (* Since 25/28 < 1, the result follows. *)
    show ?thesis using c5 < ··· >
  qed
  show ?thesis < ··· >
qed
```



Autoformalization with LLMs

Target Properties of Autoformalization

In general, methods tend to optimize for:

Syntactic Correctness
Ensure that generated formalizations are accepted by the target formal environment.

Semantic Preservation
Maintain the intended meaning despite implicit assumptions and non-standard notations.

Grounding Maximization
Whenever possible, link new statements to existing formalizations and mathematical libraries.

Semantic Preservation

Definition of Bradley–Terry model: Given a pair of items i and j drawn from some population, the Bradley–Terry model estimates the probability that the pairwise comparison turns out true, as

$$\Pr(i > j) = \frac{p_i}{p_i + p_j}$$

where p_i is a positive real-valued score assigned to individual i.

```
theory test
imports Main
begin
definition bradley_terry :: "real ⇒ real
  ⇒ real" where "bradley_terry p_i p_j =
     p_i / (p_i + p_j)"
end
```

An example of using GPT-4o in a zero-shot setting to formalize the definition of Bradley–Terry model.

The term “probability” does not appear in the formal code.

1. The phrase “p_i” is a positive real number” is omitted.
2. The formalization should represent “i>j”.

These mismatches highlight the importance of maintaining semantics in the informal mathematics during autoformalization.

(Zhang et al., 2025)

Grounding Maximization

Definition of Bradley–Terry model: Given a pair of items i and j drawn from some population, the Bradley–Terry model estimates the probability that the pairwise comparison turns out true, as

$$\Pr(i > j) = \frac{p_i}{p_i + p_j}$$

where p_i is a positive real-valued score assigned to individual i.

```
theory test
imports Main
begin
definition bradley_terry :: "real ⇒ real
  ⇒ real" where "bradley_terry p_i p_j =
     p_i / (p_i + p_j)"
end
```

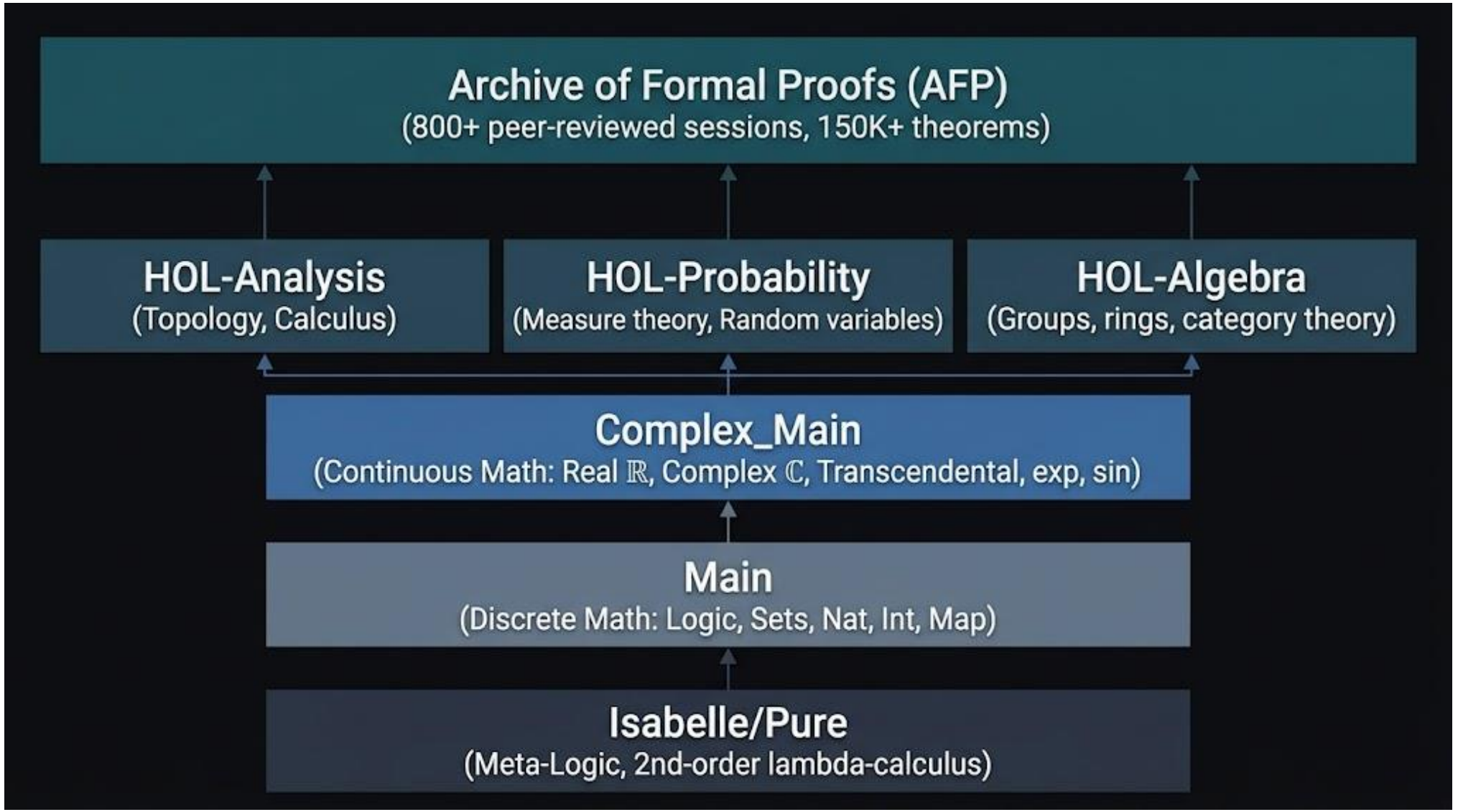
"import Main" does not contain the formalization of "real", making the formal code invalid.

Change the imports to "HOL.Real" solves the problem.

This failure to identify the correct preambles exposes limitations in the autoformalization capabilities of LLMs.

(Zhang et al., 2025)

Grounding Maximization



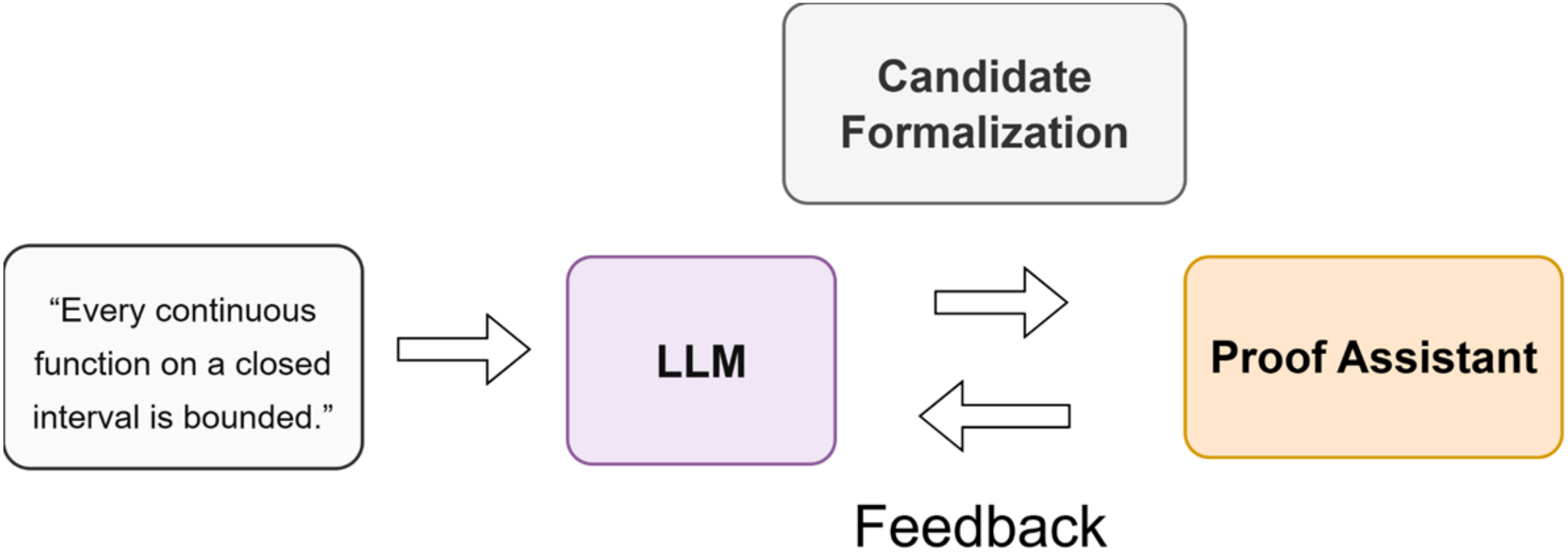
Recurring Methodological Paradigms

Iterative Refinement
Leverage expert feedback to iteratively improve formalizations.

Retrieval-Augmented Approaches
Retrieve relevant definitions, theorems, and dependencies from formal libraries.

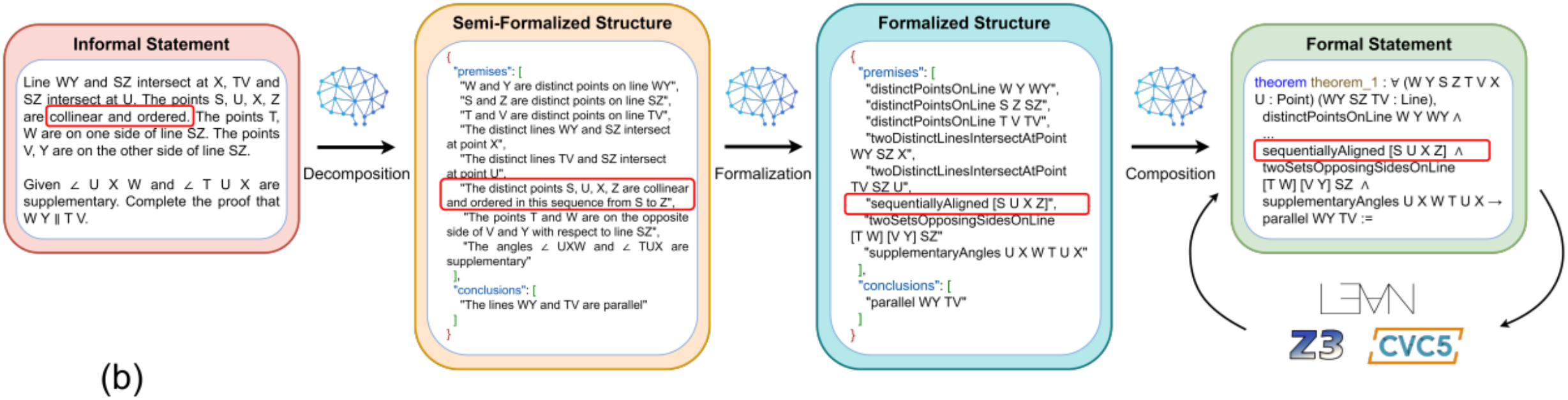
Search & Ranking
Generate multiple candidate formalizations and select the best candidate.

Iterative Refinement



Iterative Refinement

Decomposes statements into components and uses symbolic validation feedback to improve formalization.



Phase II: Statement Formalization via Decomposition and Self-Refine

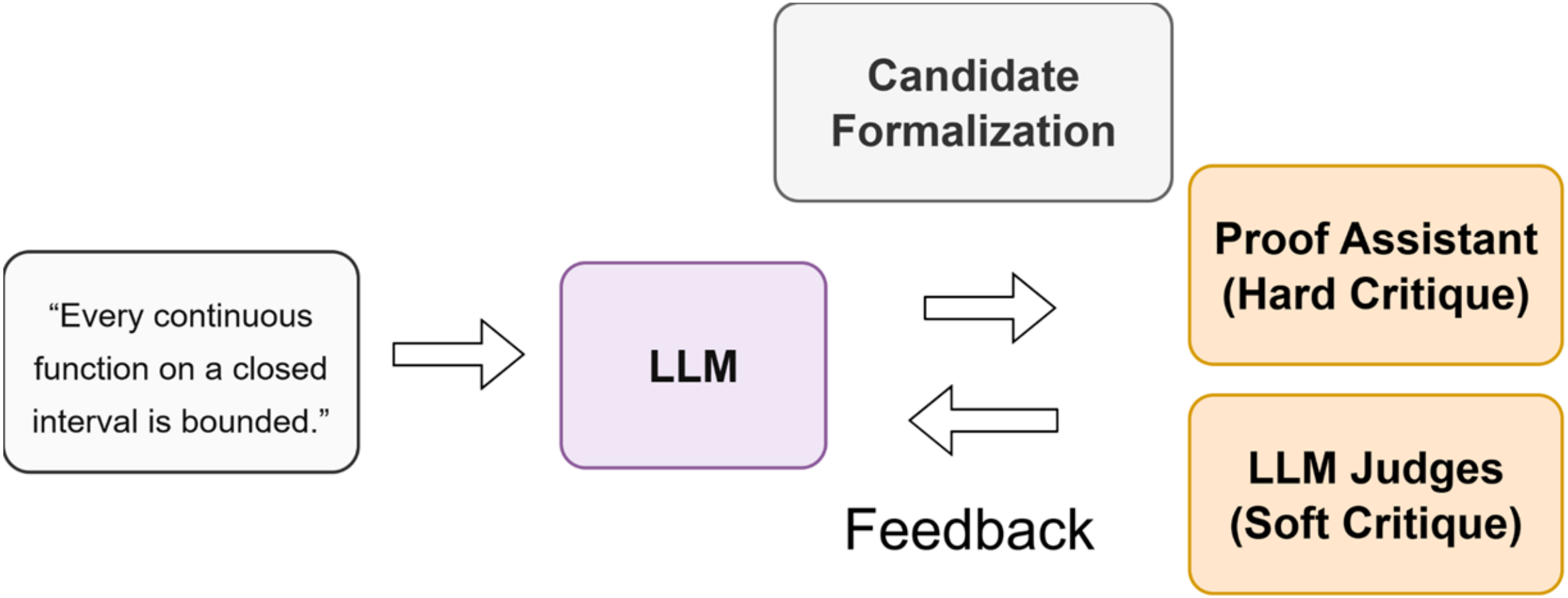
(Min et al., 2026)

Iterative Refinement

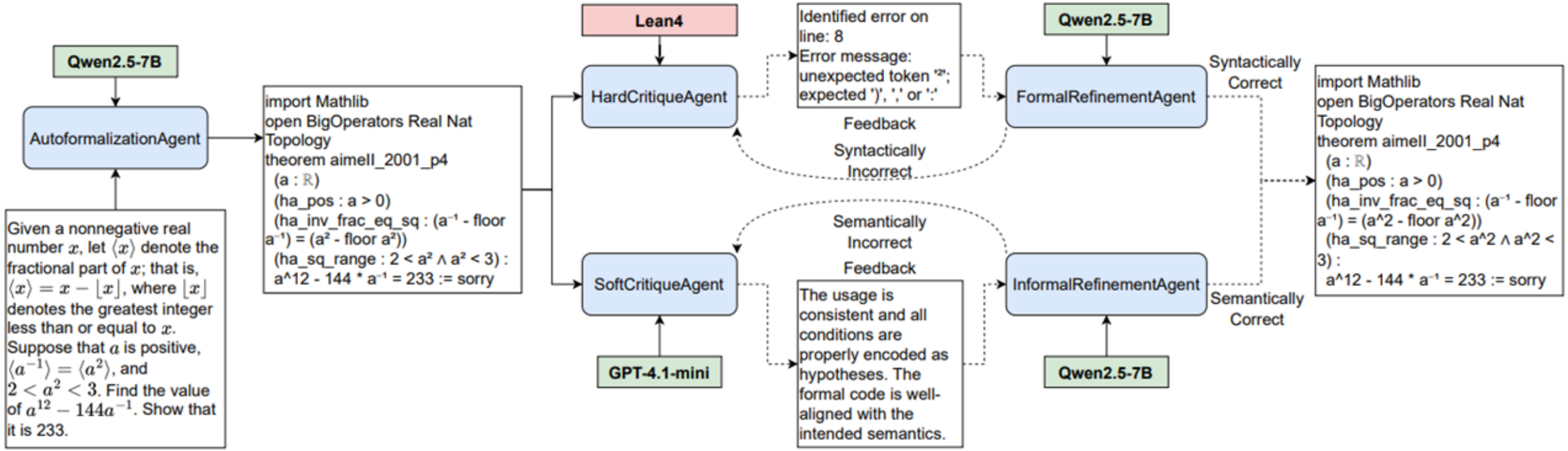
(Min et al., 2026)

Model	LeanEuclidPlus						ProofNet-Hard					
	Baseline	Divide	Abstract	DNA	OracleA	DNOracleA	Baseline	Divide	Abstract	DNA	OracleA	DNOracleA
Fine-tuned Models												
Kimina Autoformalizer 7B	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**8.4**	1.1	8.4	7.4
Mathesis Autoformalizer 7B	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**5.3**	**5.3**	7.4	5.3
Non-Reasoning Models												
GPT-4.1 mini	4.8	25.4	6.2	**42.4** ↑7.83×	3.0	48.8 ↑9.17×	0.0	0.0	5.3	**7.4**	10.5	13.7
GPT-4.1	26.8	35.2	28.6	**48.4** ↑80.6%	34.2	61.2 ↑1.28×	0.0	0.0	0.0	**10.5**	14.7	23.2
Claude 4 Sonnet	34.2	46.6	36.2	**58.2** ↑70.2%	36.6	60.4 ↑76.6%	0.0	0.0	**14.7**	12.6	12.6	25.3
Qwen3 14B	1.0	5.0	4.8	**9.6** ↑8.60×	3.0	13.4 ↑12.40×	0.0	0.0	1.1	**3.2**	0.0	0.0
Qwen3 32B	4.6	17.7	4.0	**20.6** ↑3.48×	9.2	26.4 ↑4.74×	0.0	0.0	1.1	**4.2**	5.3	4.2
Qwen3 235B Instruct 2507	40.8	43.6	45.4	**64.4** ↑57.8%	55.4	71.4 ↑75.0%	0.0	0.0	1.1	**9.5**	2.1	16.8
Average	18.7	28.9	20.9	**40.6** ↑1.17×	23.6	46.9 ↑1.51×	0.0	0.0	3.9	**7.9**	7.6	13.9
Reasoning Models												
GPT-5 mini	25.8	32.0	47.2	**58.2** ↑1.26×	58.0	71.2 ↑1.76×	0.0	0.0	8.4	**10.5**	19.0	22.1
GPT-5	35.4	37.0	53.2	**55.8** ↑57.6%	60.2	68.4 ↑93.2%	0.0	0.0	12.6	**15.8**	12.6	28.4
Claude 4 Sonnet (Thinking)	32.8	45.4	42.0	**57.4** ↑75.0%	50.8	64.4 ↑96.3%	0.0	0.0	11.6	**12.6**	10.5	30.5
Qwen3 14B (Thinking)	25.4	27.8	33.8	**38.6** ↑52.0%	33.0	51.6 ↑1.03×	0.0	0.0	**1.1**	**1.1**	4.2	9.5
Qwen3 32B (Thinking)	29.6	31.6	39.4	**40.8** ↑37.8%	46.8	58.2 ↑96.6%	0.0	0.0	**2.1**	**2.1**	8.4	14.7
Qwen3 235B Thinking 2507	40.2	45.4	41.6	**55.6** ↑38.3%	58.4	70.4 ↑75.1%	0.0	0.0	**5.3**	**5.3**	9.5	19.0
Average	31.5	36.5	42.9	**51.1** ↑62.2%	42.9	64.0 ↑1.03×	0.0	0.0	6.8	**7.9**	10.7	20.7

Iterative Refinement

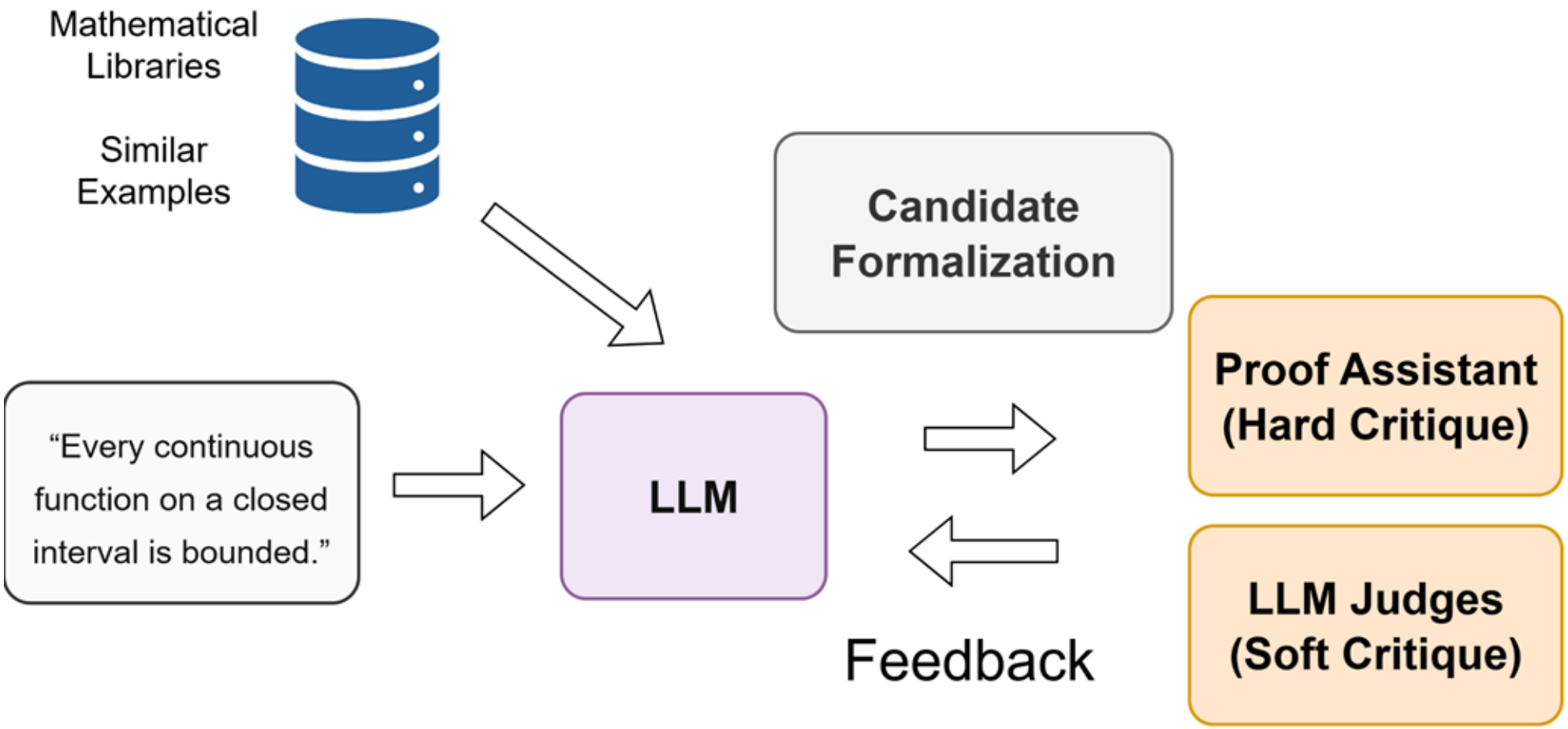


Iterative Refinement

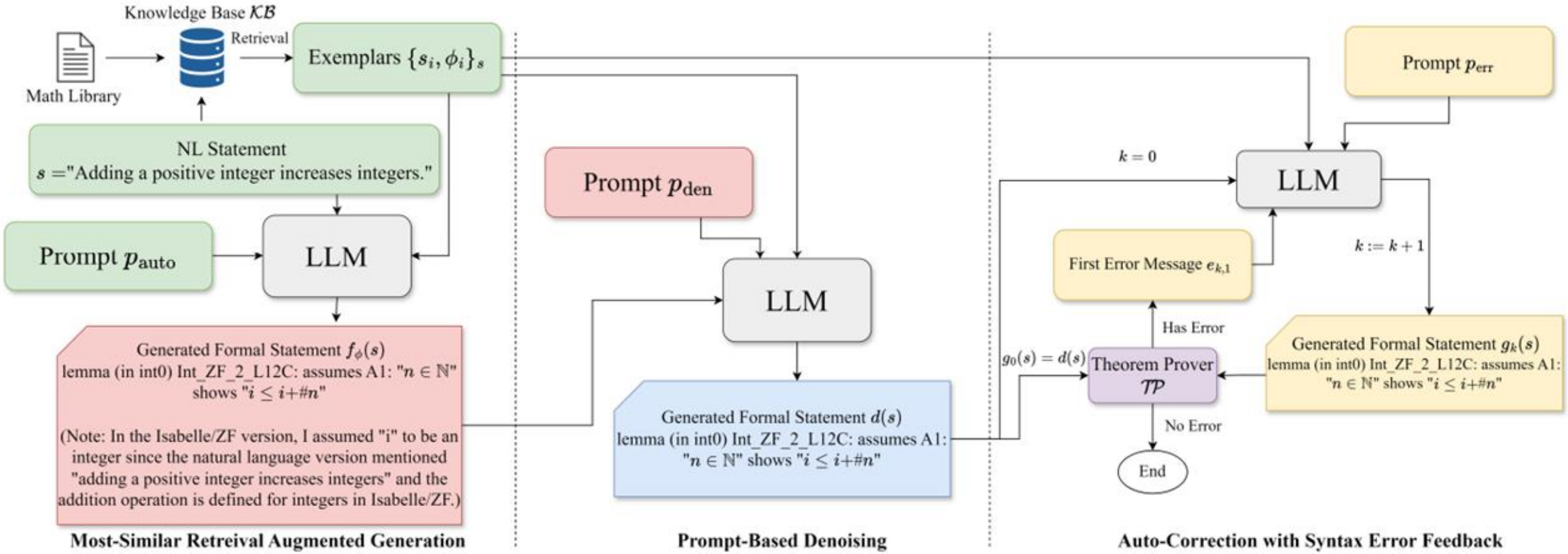


(Zhang et al., 2025)

Retrieval-Augmented Approaches



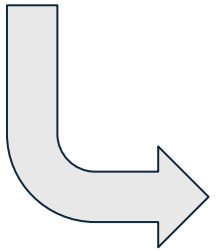
Retrieval-Augmented Approaches



(Zhang et al., 2024)

Retrieval-Augmented Approaches

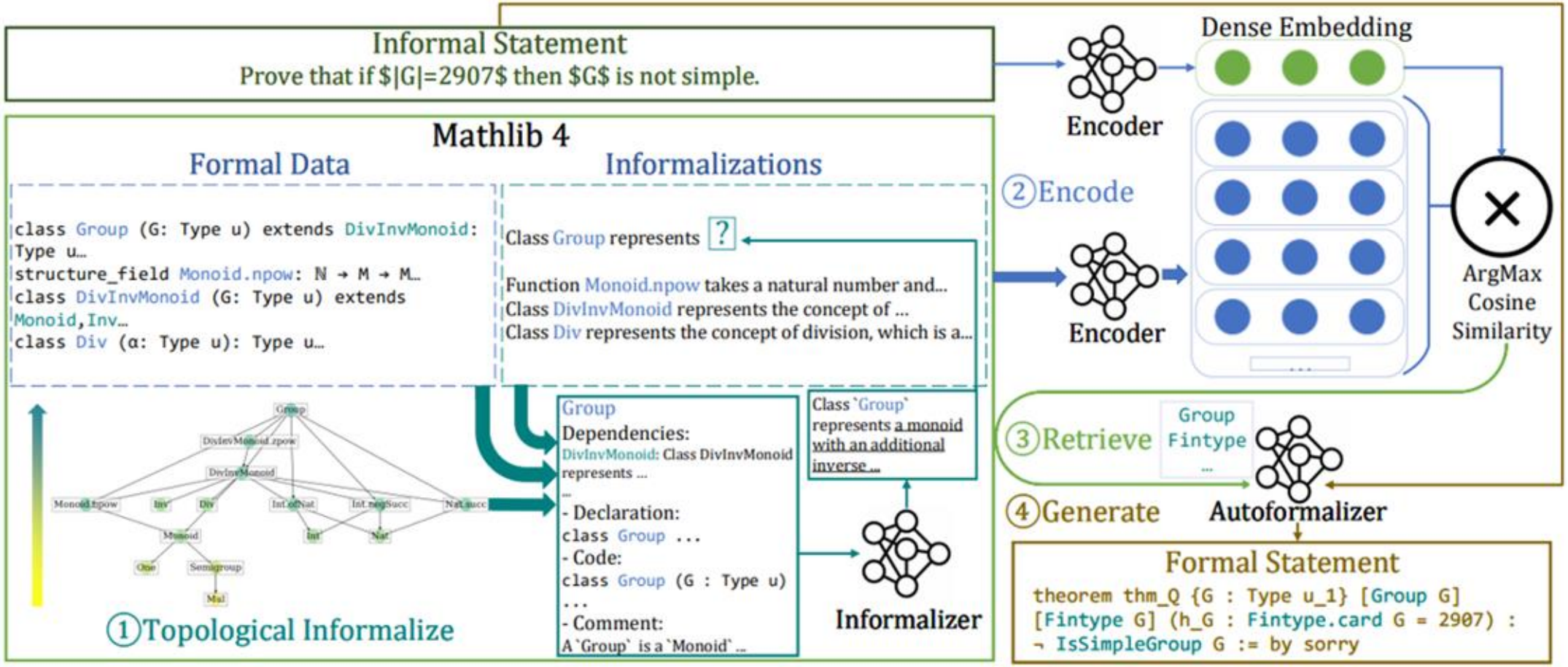
LLM	Method	BLEU-2	ChrF	RUBY	CBS	Pass
Baselines						
Mistral	Zero-Shot	0.30	17.14	16.13	51.13	0.0
Mistral	3-Shot	1.77	27.30	24.02	62.73	5.47
Llemma 7B	Zero-Shot	0.91	16.67	14.77	47.74	9.12
Llemma 7B	3-Shot	2.43	28.81	21.93	66.68	8.76
Mixtral	Zero-Shot	0.65	16.33	17.97	51.07	0.36
Mixtral	3-Shot	5.37	30.53	28.51	62.86	1.09
GPT-3.5-Turbo	Zero-Shot	2.15	17.81	21.93	51.69	40.51
GPT-3.5-Turbo	3-Shot	14.23	37.95	39.13	67.26	38.69



Retrieval Augmented Autoformalization						
Mistral	Query: **T** Index: **T**	10.05	51.38	44.82	76.93	21.53
Mistral	Query: **T** Index: **T+S**	9.96	50.79	43.92	76.21	19.71
Mistral	Query: **T** Index: **I+S**	5.65	36.92	32.23	67.47	8.76
Mistral	Query: **T** Index: **T+I+S**	10.53	49.61	43.28	75.17	22.26
Mistral	Query: **T+ZS** Index: **T**	10.14	46.89	40.76	73.69	12.77
Mistral	Query: **T+ZS** Index: **T+S**	8.40	46.26	39.91	73.40	14.96
Mistral	Query: **T+ZS** Index: **I+S**	5.51	36.71	31.94	66.91	10.95
Mistral	Query: **T+ZS** Index: **T+I+S**	8.85	45.14	39.27	72.47	16.06
Llemma 7B	Query: **T** Index: **T**	4.18	36.93	28.68	69.93	12.77
Llemma 7B	Query: **T** Index: **T+S**	4.61	37.48	29.39	69.56	14.23
GPT-3.5-Turbo	Query: **T** Index: **T**	36.32	**59.63**	**58.51**	**79.14**	**64.60**
GPT-3.5-Turbo	Query: **T** Index: **T+S**	**37.11**	58.56	57.71	78.89	62.77

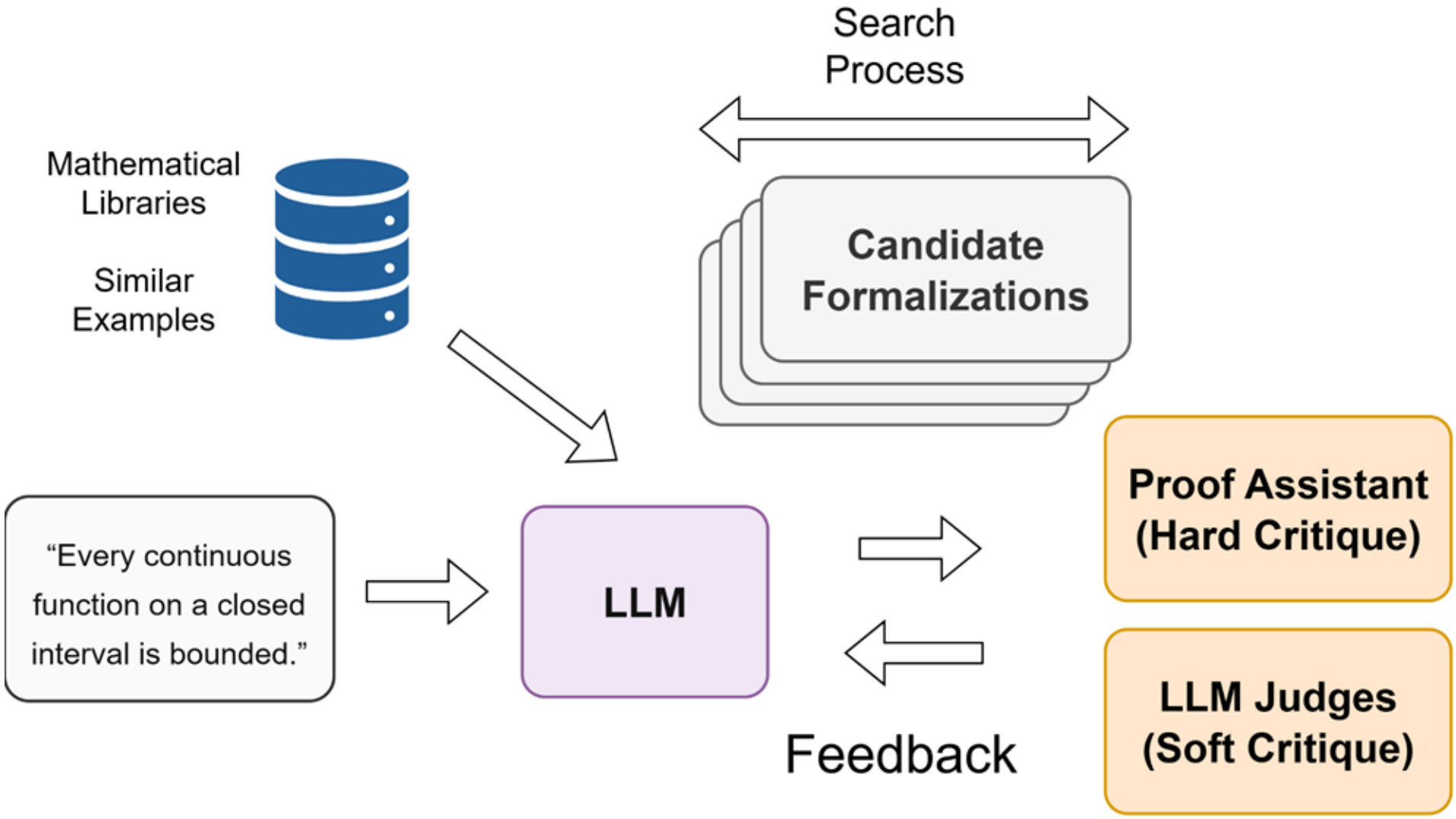
(Zhang et al., 2024)

Retrieval-Augmented Approaches



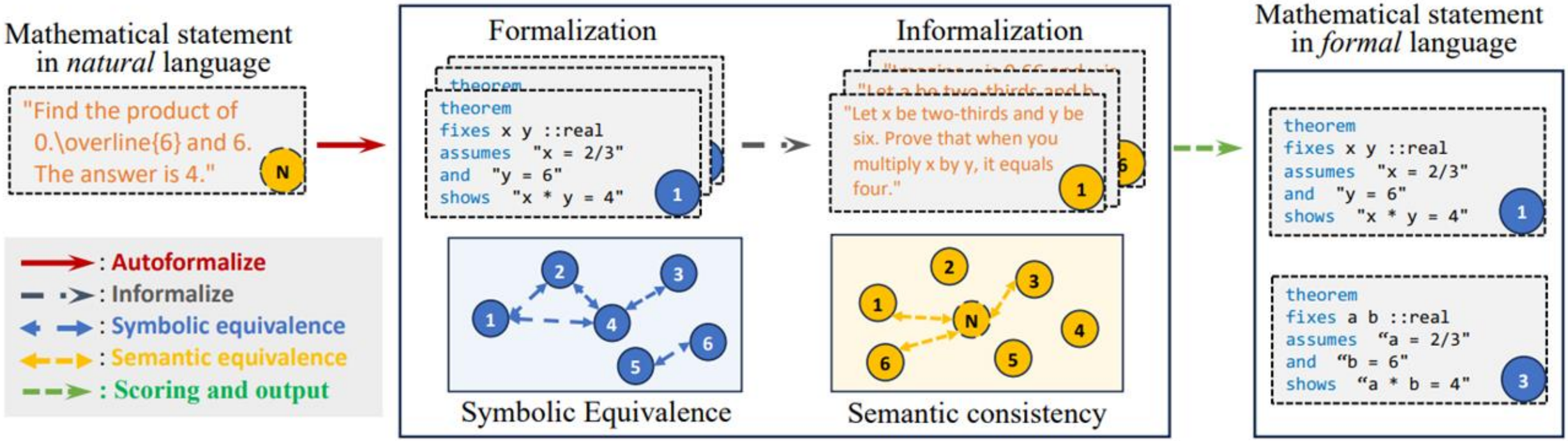
(Liu et al, 2025)

Search & Ranking



Search & Ranking

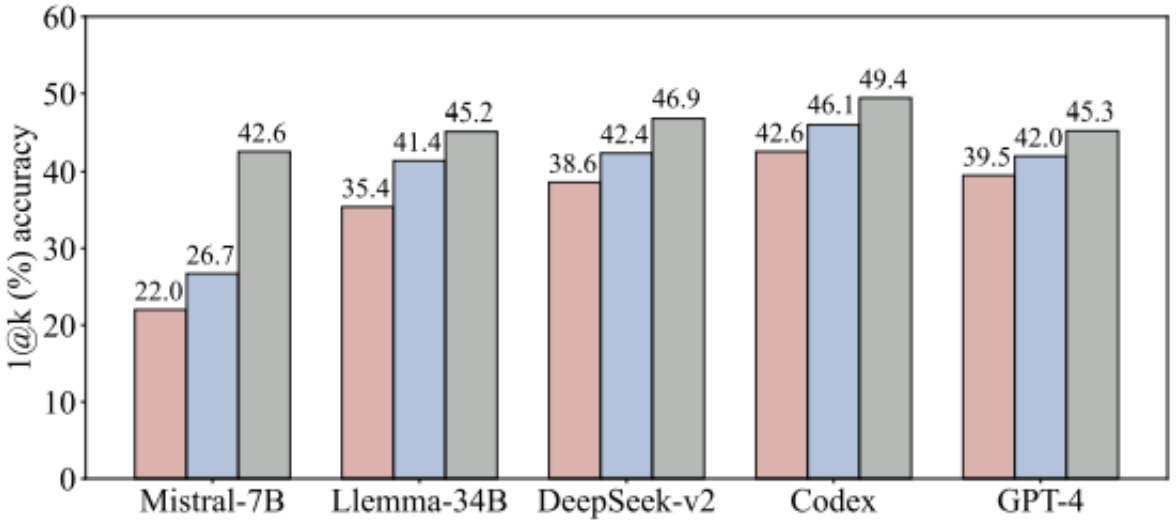
Ranks candidate formal statements using symbolic equivalence (via equivalence class determined by ATP) and semantic consistency (via backtranslation and similarity).

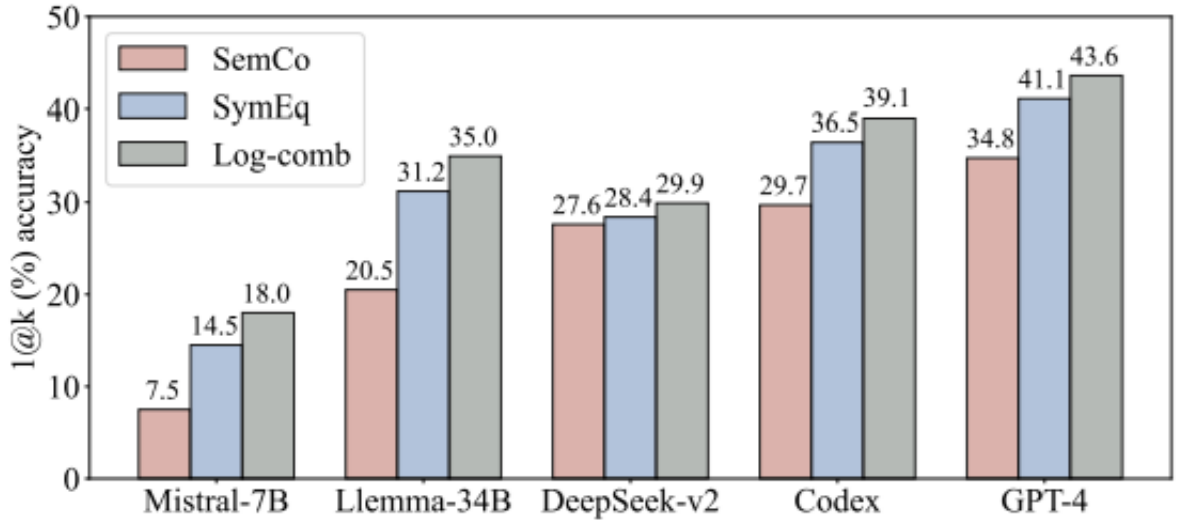


(Li et al., 2024)

Search & Ranking

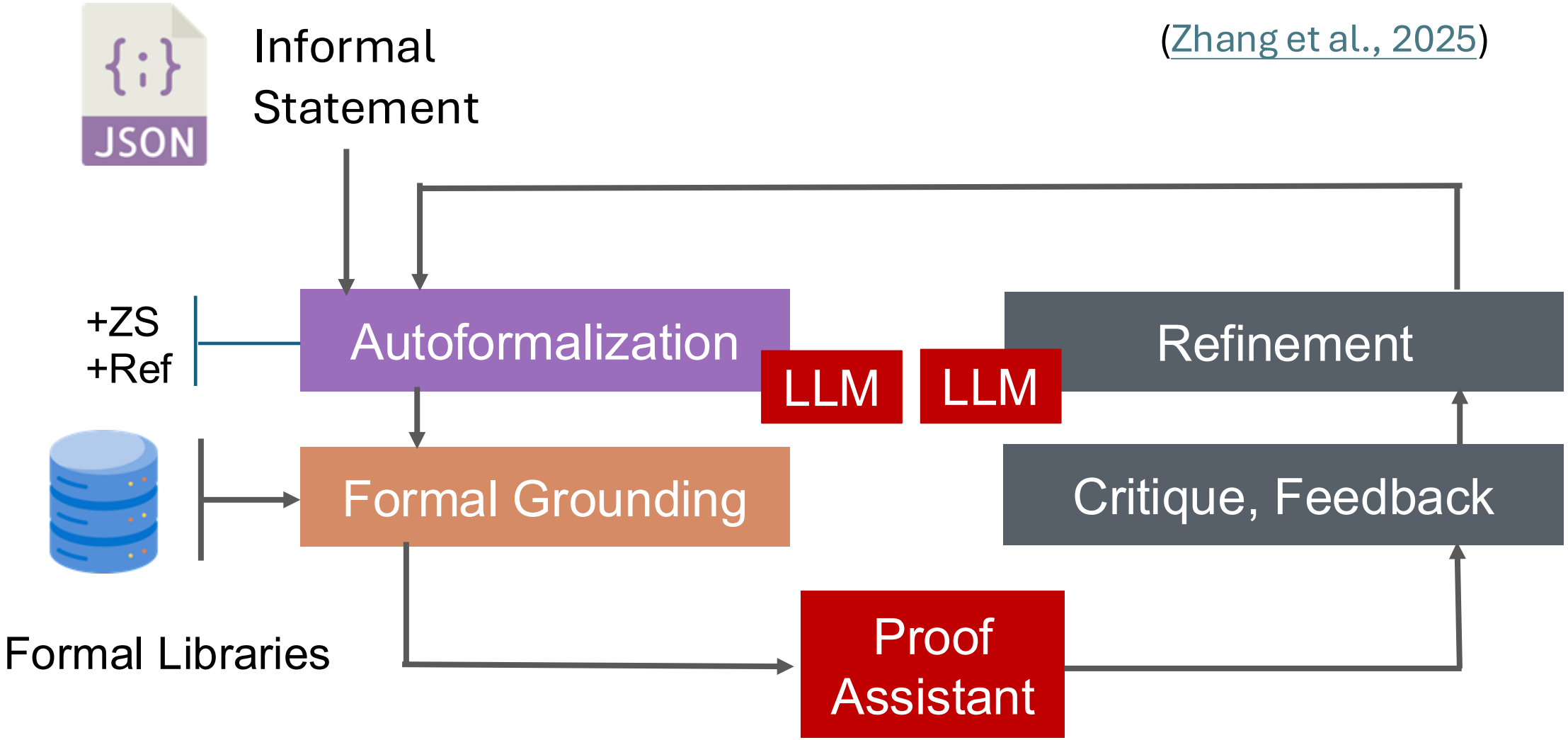
Ranks candidate formal statements using symbolic equivalence (via equivalence class determined by ATP) and semantic consistency (via backtranslation and similarity).





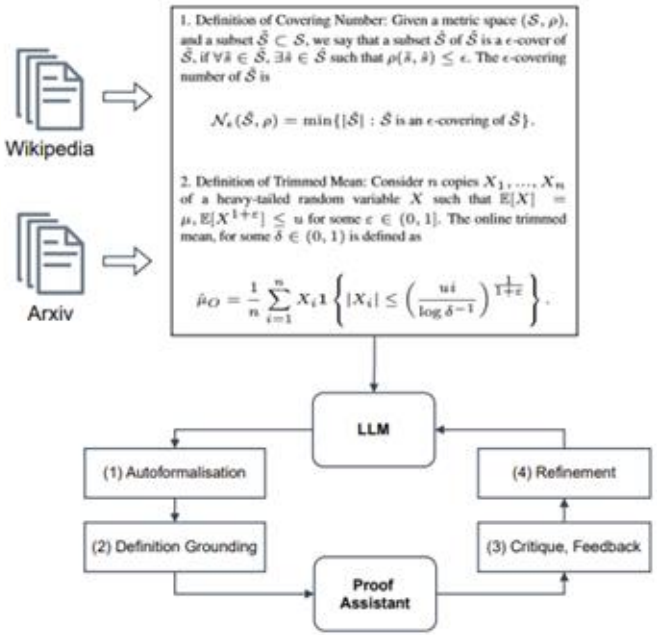
(Li et al., 2024)

Combining Multiple Paradigms



Combining Multiple Paradigms

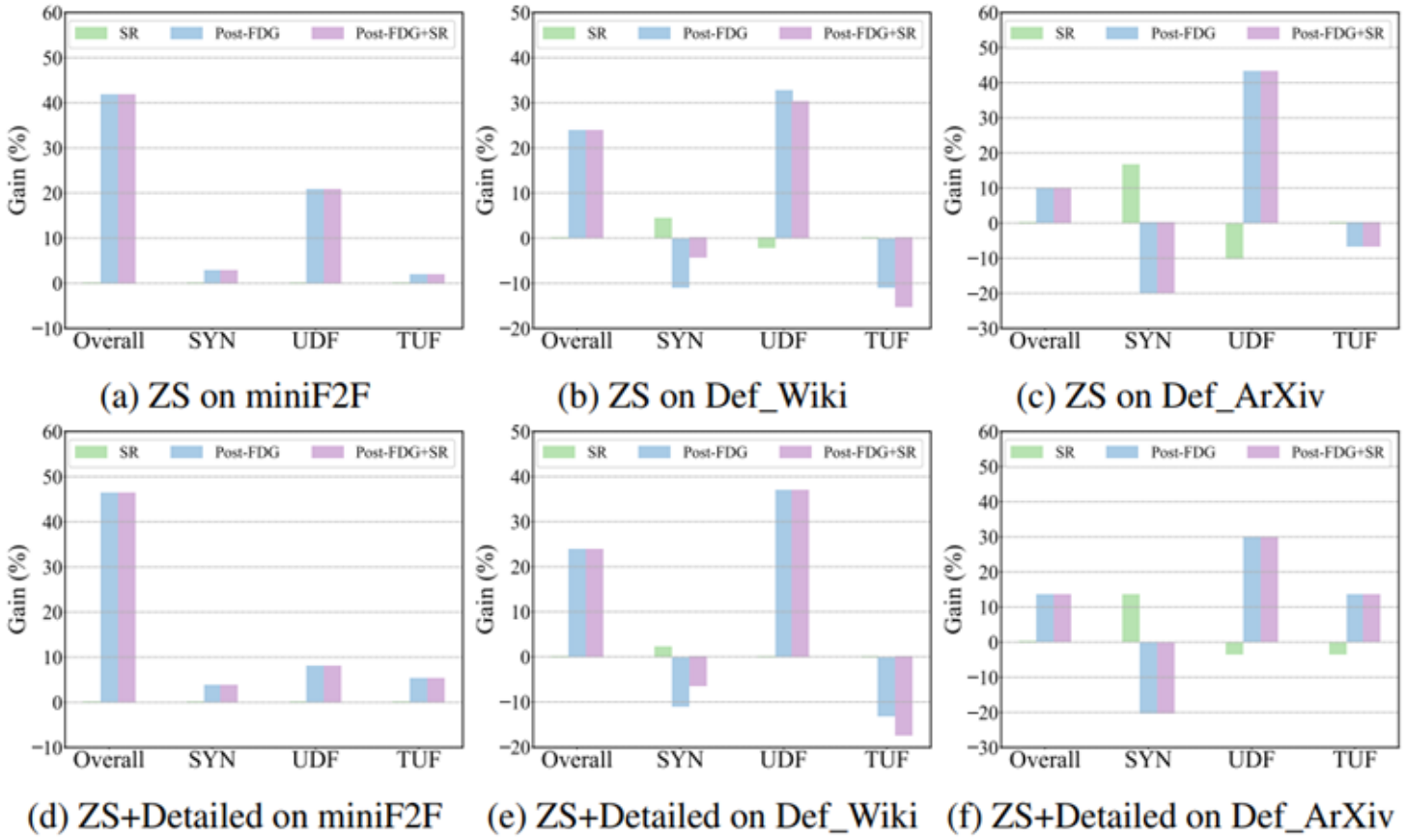
miniF2F	Def_Wiki	Def_ArXiv
1. Suppose that $\sec x + \tan x = \frac{22}{7}$ and that $\csc x + \cot x = \frac{m}{n}$, where $\frac{m}{n}$ is in lowest terms. Find $m+n$. Show that it is 044. 2. What is the sum of the two values of x for which $(x+3)^2 = 121$? Show that it is -6. 3. The product of two positive whole numbers is 2005. If neither number is 1, what is the sum of the two numbers? Show that it is 406. 4. The expression $10x^2 - x - 24$ can be written as $(Ax-8)(Bx+3)$, where A and B are integers. What is $AB+B$? Show that it is 12.	1. Definition of Rademacher Complexity: Given a set $A \subseteq \mathbb{R}^m$, the Rademacher complexity of A is defined as follows: $$\mathrm{Rad}(A) := \frac{1}{m}\mathbb{E}_\sigma\left[\sup_{a\in A}\sum_{i=1}^{m}\sigma_i a_i\right]$$ where $\sigma_1, \sigma_2, \ldots, \sigma_m$ are independent random variables drawn from the Rademacher distribution (i.e. $\Pr(\sigma_i = +1) = \Pr(\sigma_i = -1) = 1/2$ for $i = 1, 2, \ldots, m$), and $a = (a_1, \ldots, a_m)$. 2. Definition of Polynomial Kernel: For degree-d polynomials, the polynomial kernel is defined as $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\mathsf{T}\mathbf{y} + c)^d$ where $\mathbf{x}$ and $\mathbf{y}$ are vectors of size n in the input space, i.e. vectors of features computed from training or test samples and $c \geq 0$ is a free parameter trading off the influence of higher-order versus lower-order terms in the polynomial.	1. Definition of Covering Number: Given a metric space $(\mathcal{S}, \rho)$, and a subset $\tilde{\mathcal{S}} \subset$ subset $\hat{\mathcal{S}}$ of $\tilde{\mathcal{S}}$ is a ϵ-cover of $\tilde{\mathcal{S}}$, if such that $\rho(\tilde{s}, \hat{s}) \leq \epsilon$. The ϵ-cov $\tilde{\mathcal{S}}$ is $$\mathcal{N}_\epsilon(\tilde{\mathcal{S}}, \rho) = \min\{|\hat{\mathcal{S}}| : \hat{\mathcal{S}} \text{ is an } \epsilon\text{-}$$ 2. Definition of Trimmed Mean: Consider n copies $X_1, ..., X_n$ of a heavy-tailed random variable X such that $\mathbb{E}[X] = \mu, \mathbb{E}[X^{1+\varepsilon}] \leq u$ for some $\varepsilon \in (0, 1]$. The online trimmed mean, for some $\delta \in (0, 1)$ is defined as $$\hat{\mu}_O = \frac{1}{n}\sum_{i=1}^{n} X_i \mathbf{1}\left\{|X_i| \leq \left(\frac{ui}{\log \delta^{-1}}\right)^{\frac{1}{1+\varepsilon}}\right\}.$$



(Zhang et al., 2025)

Combining Multiple Paradigms

Real-world definitions define a challenging frontier. However, existing strategies yeld notable improvements.



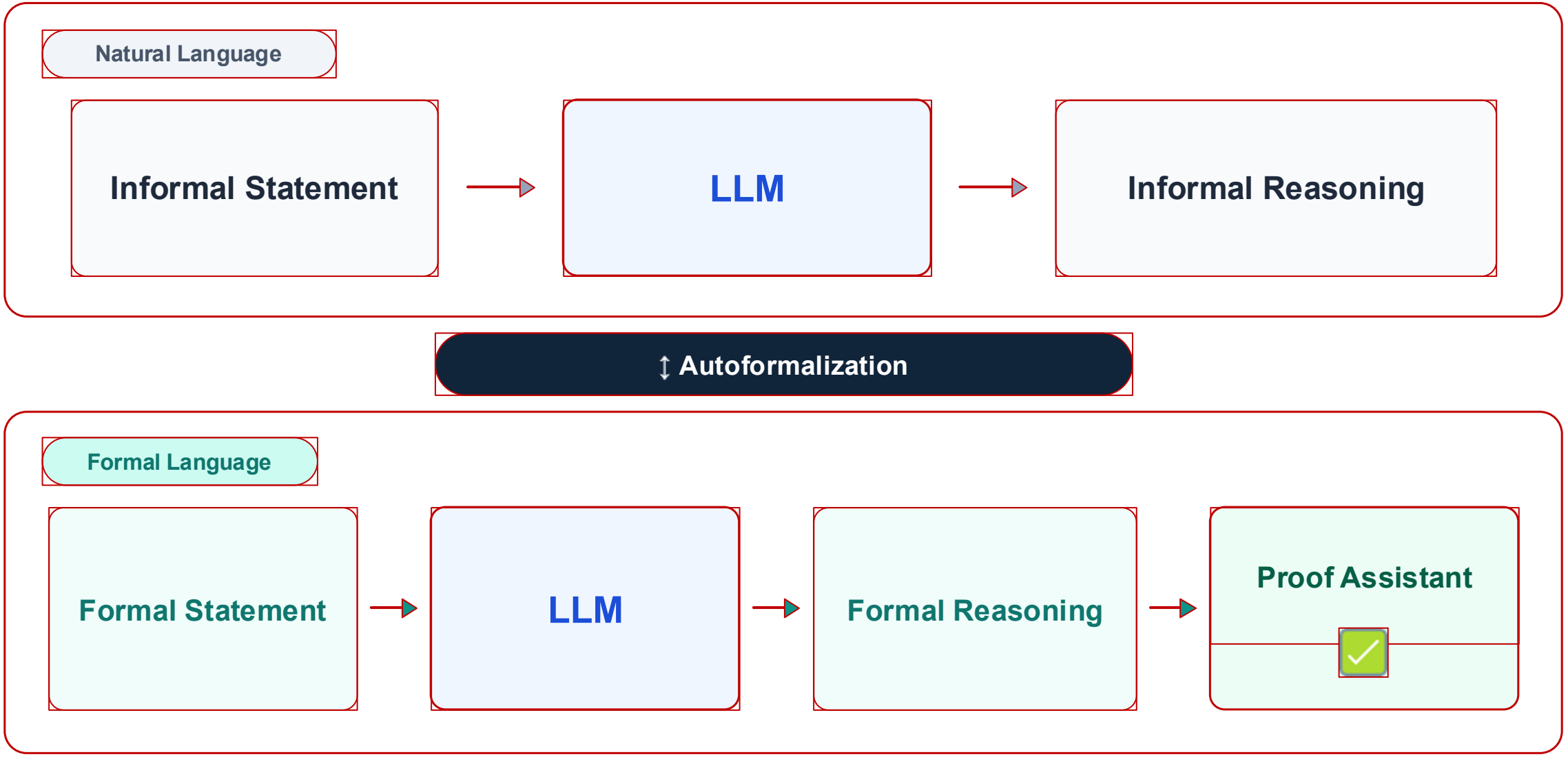
(Zhang et al., 2025)

Method Trade-offs: No Universal Winner

Method	Advantages	Limitations
Retrieval-Augmented Approaches	Grounds outputs in formal libraries; reduces missing or invented dependencies; requires no retraining.	Depends on retrieval quality and library coverage; irrelevant context may mislead generation.
Iterative Refinement	Uses verifier or semantic feedback to repair errors; improves syntactic validity through repeated attempts.	Repeated calls raise computational cost; syntactic fixes may introduce semantic drift.
Search & Ranking	Explores alternative formalizations; selection can consider both correctness and semantic fidelity.	Candidate generation and scoring are costly; unreliable ranking may select wrong statements.

Key Takeaways

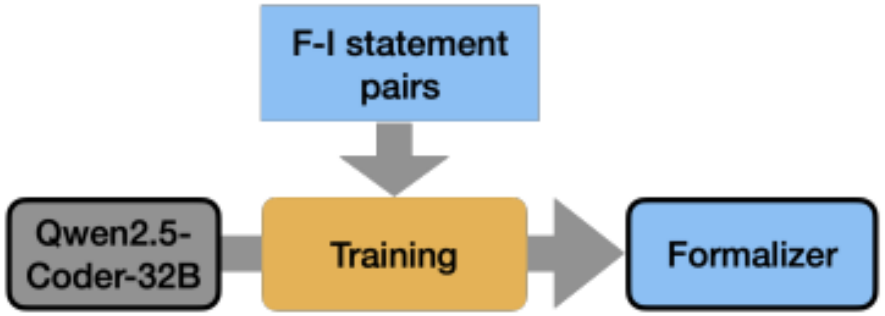
- **Complementary methods:** Different approaches address distinct challenges and can be combined rather than treated as competing alternatives.
- **Hybrid systems:** Modern autoformalization systems often integrate multiple techniques (e.g., RAG + iterative refinement).
- **Multi-dimensional improvements:** Combining methods can improve grounding, syntactic correctness, and semantic preservation, but gains in one dimension do not guarantee gains in others.
- **Remaining challenges:** Hybrid systems introduce additional computational costs and complexity. Formalising real-world mathematical statements is very hard. Obtaining high-quality data is a major challenge (mostly rely on synthetic data).



From Auformalization to Formal Mathematical Reasoning

Autoformalization and Expert Iteration

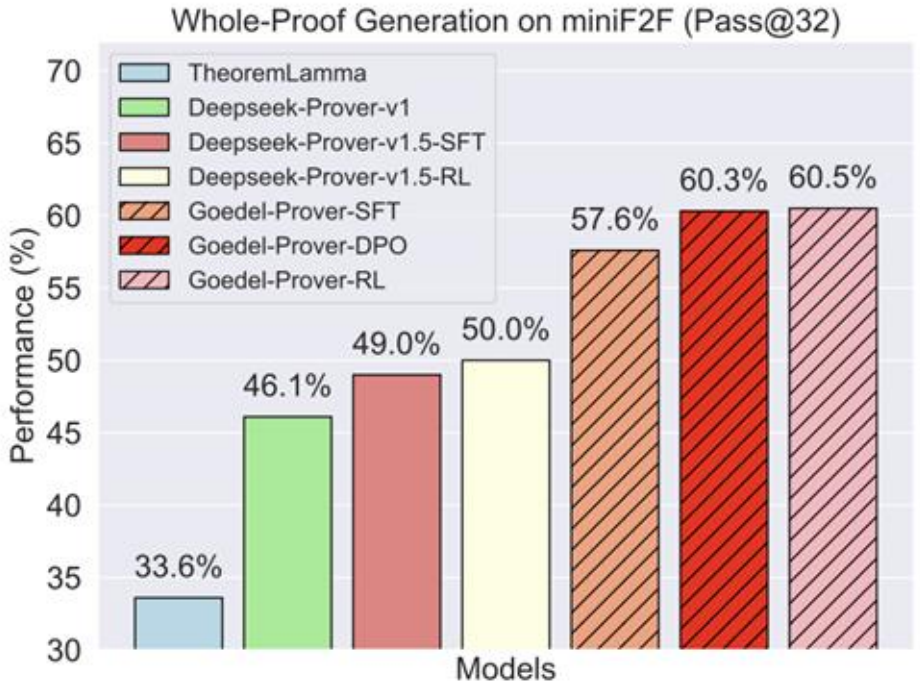
(Lin et al., 2025)

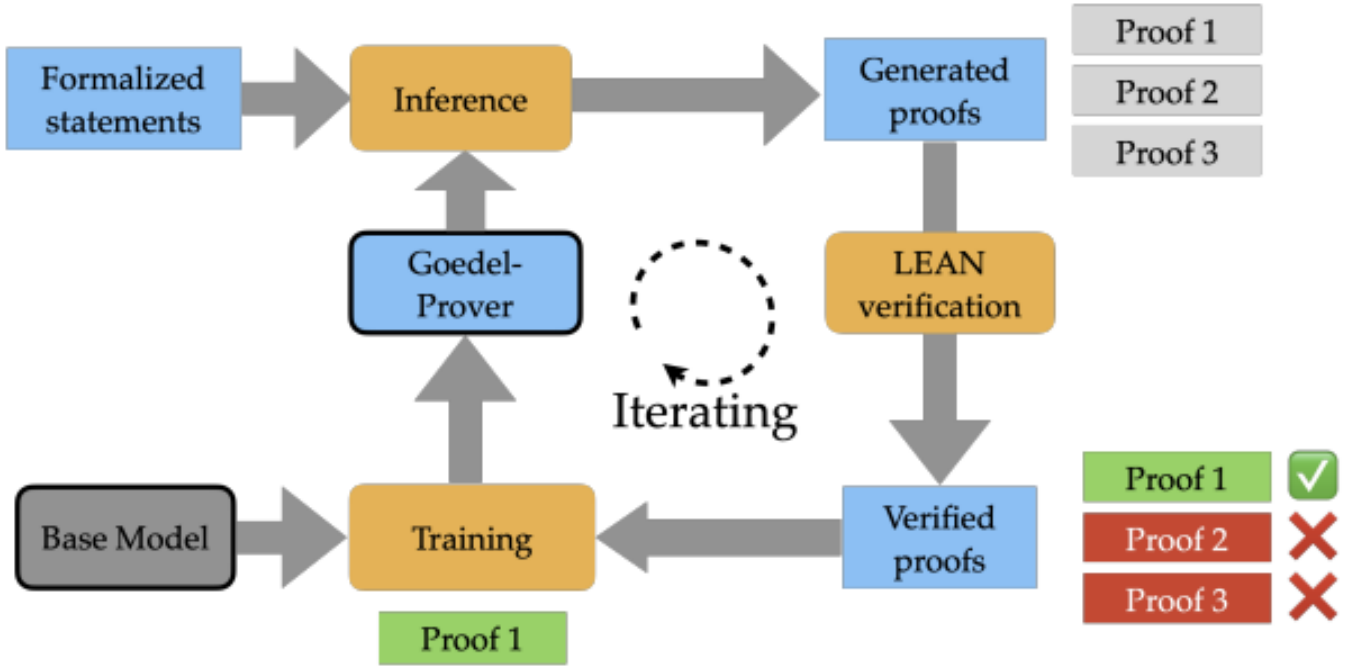


Formal and Informal (F-I) statement pair

Informal statement	Let a, b, and c be real numbers that satisfying $3a+b+c=-3$, $a+3b+c=9$, and $a+b+3c=19$. Prove that the product $abc=-56$.
Formal statement	`theorem mathd_algebra_338 (a b c : ℝ) (h₀ : 3 * a + b + c = -3) (h₁ : a + 3 * b + c = 9) (h₂ : a + b + 3 * c = 19) : a * b * c = -56 := by`

1.64 million formal statements





Combining Informal and Formal Reasoning for Large-Scale RL

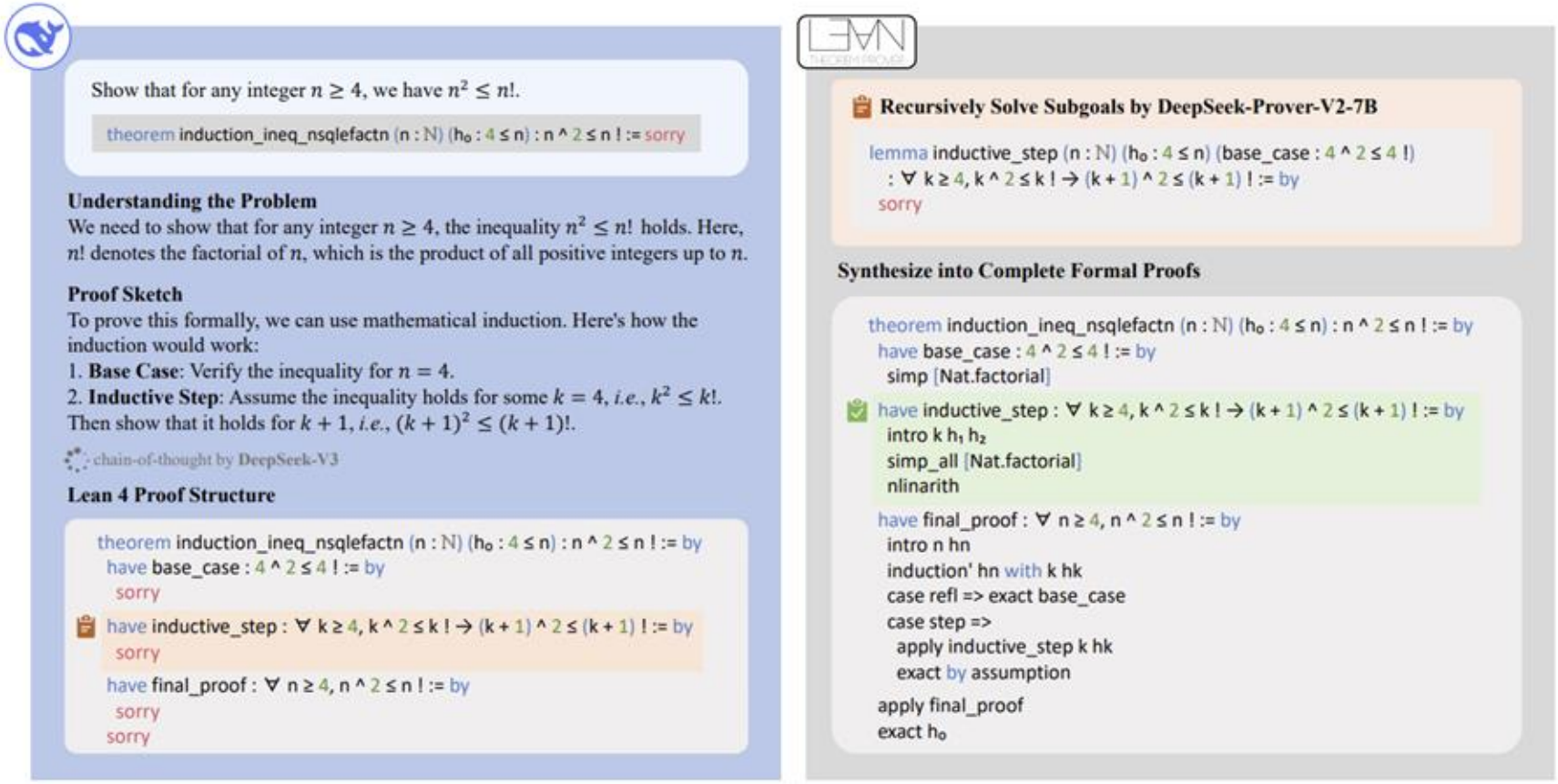
DeepSeek-Prover-V2:

Stage 1:

- **DeepSeek-V3** drafts informal natural-language sketches and autoformalizes them into Lean subgoals.
- A small fine-tuned **7B model** proves each individual subgoal.
- Stitches the informal thoughts and verified formal steps into unified end-to-end proofs.

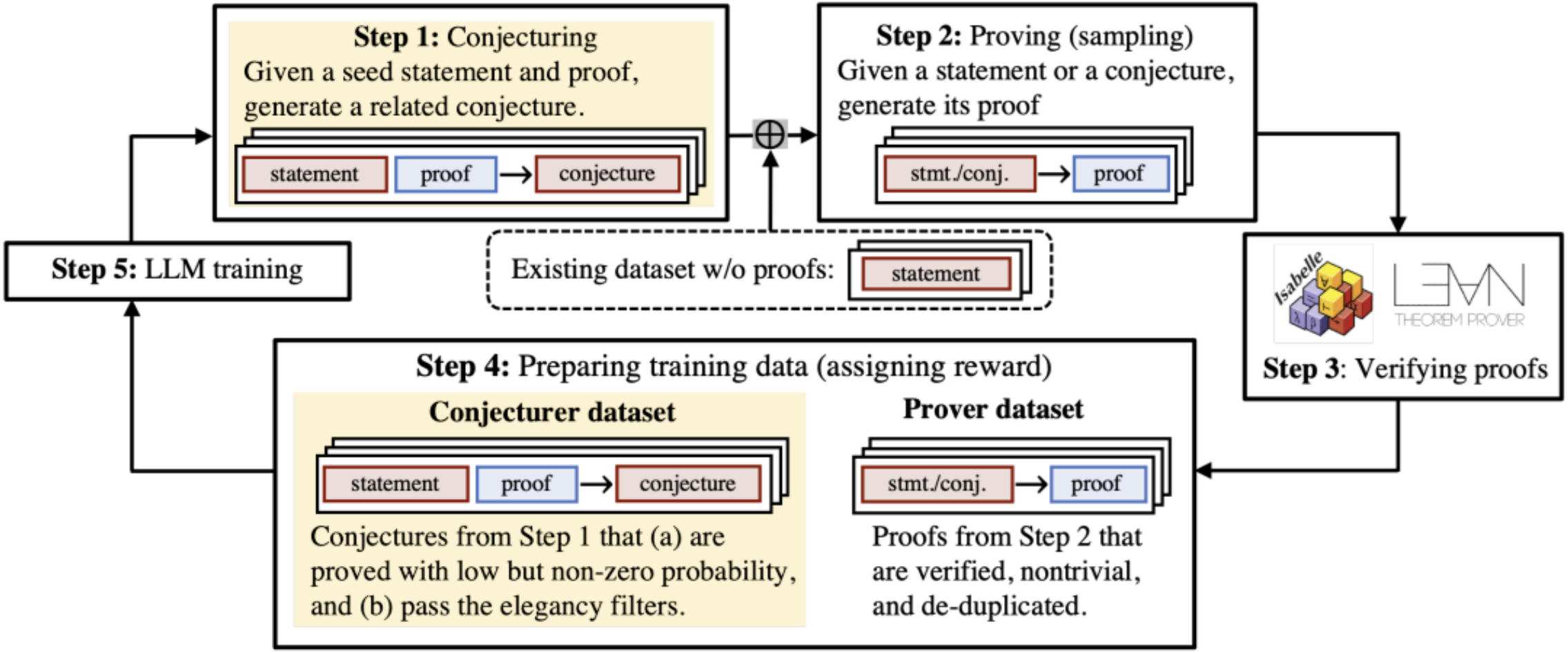
Stage 2:

- Fine-tunes the a **671B** MoE on these trajectories and then use RL with Lean kernel feedback.



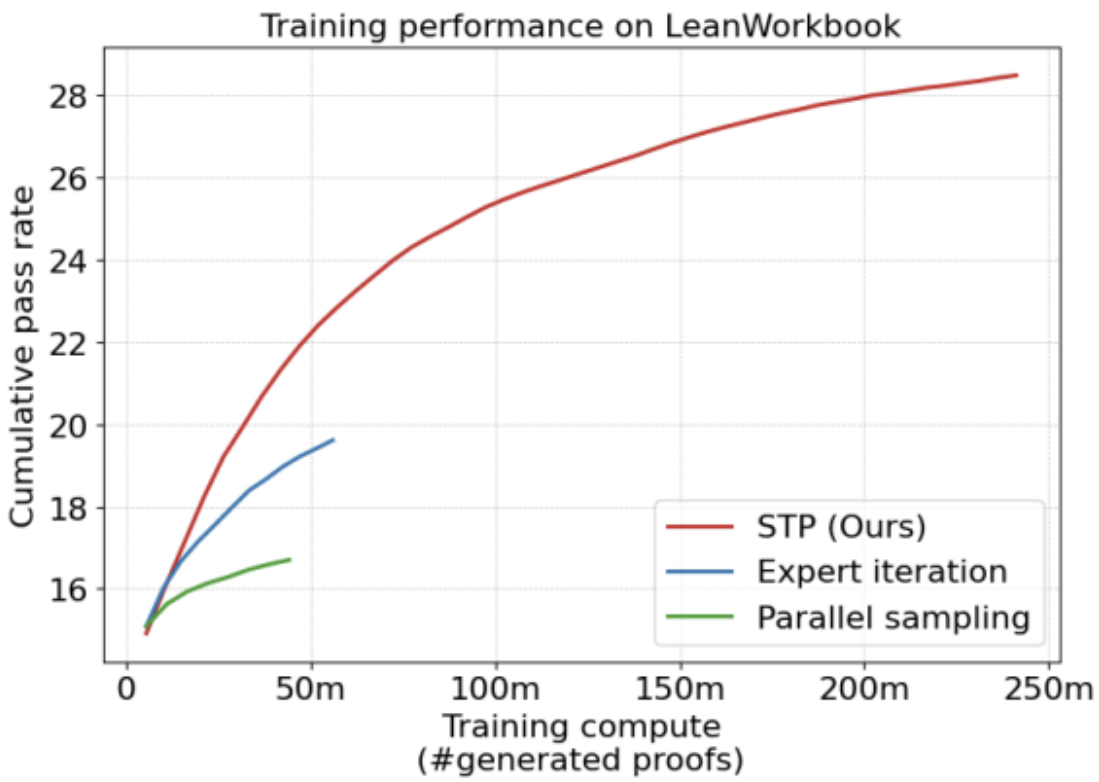
(Ren et al., 2025)

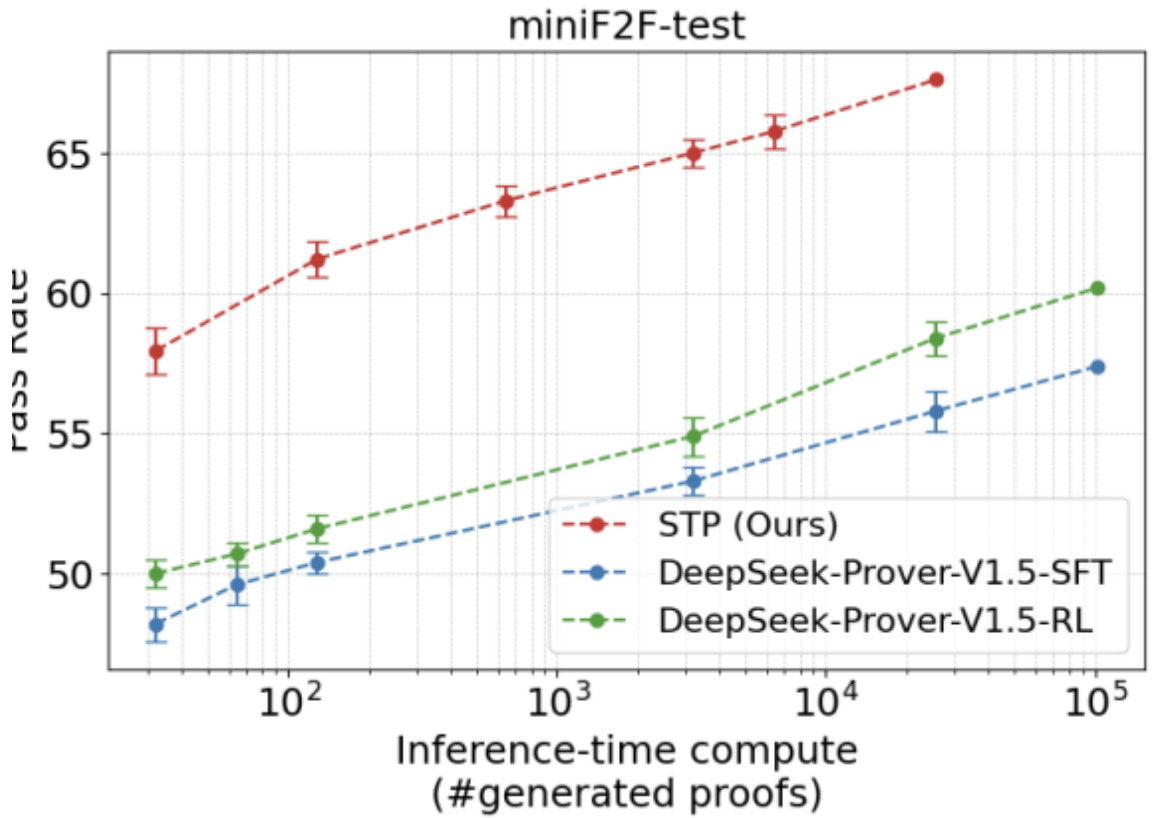
Toward Self-Play LLM Theorem Provers



(Dong et al, 2025)

Toward Self-Play LLM Theorem Provers





(Dong et al, 2025)

Takeaways

- **Informal-formal synergy:** SOTA models succeed by drafting informal proof sketches and autoformalizing them into verifiable subgoals.
- **Scalable Environment:** feeback from proof assistants and theorem provers provide verifiable rewards that unlock scalable RL and Expert Iteration.

Persisting challenges:

- **Compute, data & search bottlenecks:** Proving deep theorems suffers from massive sample rejection rates and exponential search spaces.
- **Prover overspecialisation:** Heavy community focus on Lean 4 limits generalization and cross-system transfer (Isabelle, Coq).
- **Adaptability & abstraction:** Models struggle to efficiently adapt to new formal languages and generalise effectively across domains.



Evaluation

Two Axes: Syntactic Validity ≠ Semantic Alignment

SYNTACTIC VALIDITY

Is the formalization well-formed and accepted by the target theorem prover?

Parsing · Elaboration · Type Checking
Verified Proof (when required)

SEMANTIC ALIGNMENT

Does the generated formalization preserve the intended mathematical meaning of the informal source?

Quantifiers · Hypotheses · Objects
Implicit Assumptions · Library Grounding

A theorem prover verifies syntactic validity, not semantic alignment. Even a fully verified theorem may formalize the wrong mathematical claim.

Why Syntactic Validity Is Not Enough

Informal source

For every real x, x^2 is nonnegative.

Correct statement (Lean-style)

```
theorem sq_nonneg (x : ℝ) : 0 ≤ x ^ 2 := by
sorry
```

Correctly compiled statement

```
theorem wrong (x : ℝ) : 0 < x ^ 2 := by
  sorry
```

Teaching point

The formalization can parse and elaborate, however, at x = 0 it is false. Compilation of a statement alone does not establish semantic alignment.

Why Syntactic Validity Is Not Enough

Informal source

Every natural number has a greater natural number.

Correct formalization (Lean-style)

```
theorem example : ∀ n : ℕ, ∃ m : ℕ, m > n :=
by
intro n
exact ⟨n + 1, by omega⟩
```

Correctly compiled formalization

```
theorem example : ∀ n : ℕ, ∃ m : ℕ, m ≥
n := by
  intro n
  exact ⟨n, le_refl n⟩
```

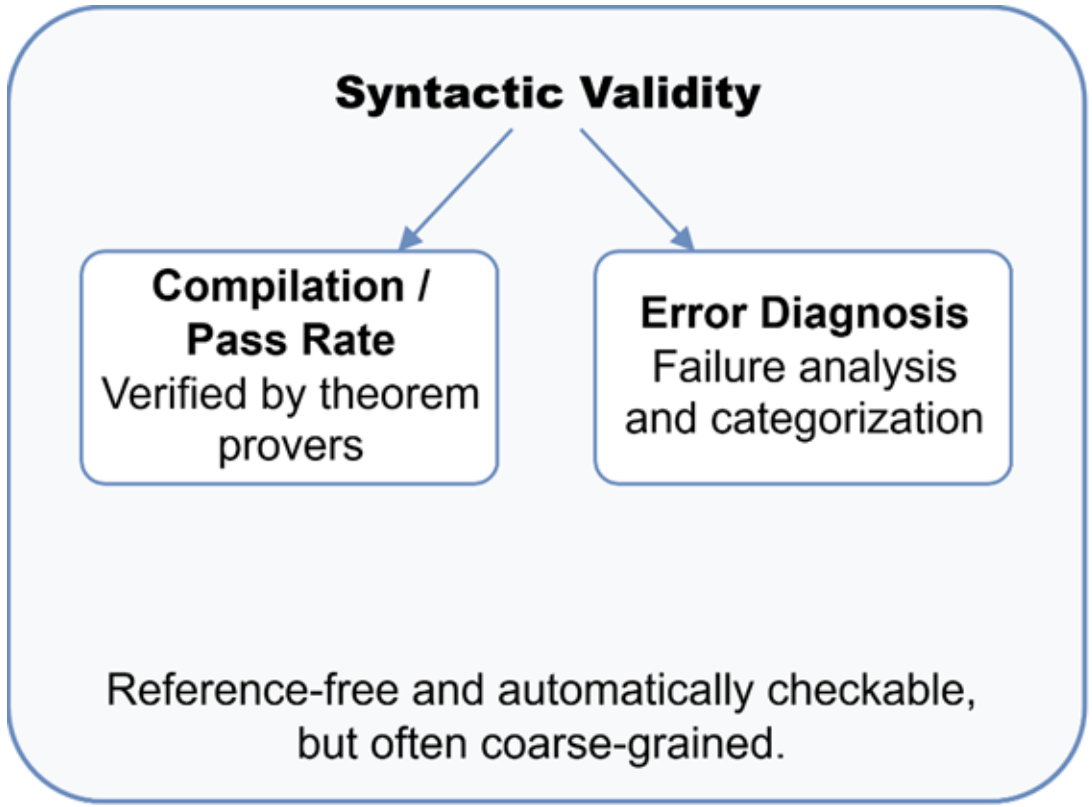
Teaching point

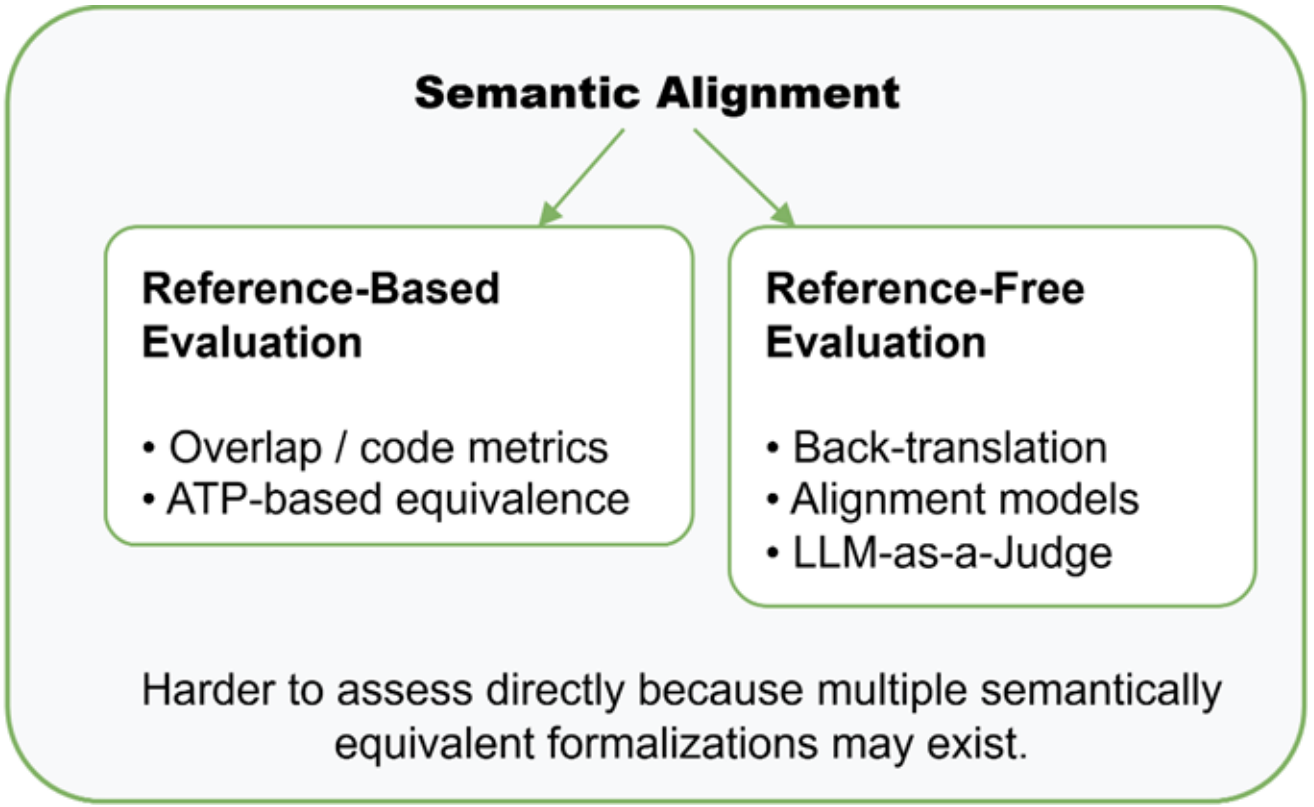
The formalization replaces strict inequality (>) with non-strict inequality (≥), thereby weakening the original mathematical claim. It becomes trivial.

Can We Fool Lean into Accepting an Unverified Proof?

- **Proof Placeholders:** Use *"sorry" or "admit"* to bypass unfinished proof obligations.
- **Unproven Axioms:** Introduce assumptions without proving them.
- **Hidden Dependencies:** Rely on previously established theorems containing unverified assumptions.
- **Inconsistent Axioms:** Introduce contradictory assumptions, making arbitrary conclusions derivable.
- **Trusted Computation:** Rely on external or native computation, extending the trusted computing base.
- **Verification Infrastructure:** Exploit bugs or weaknesses in trusted components, such as the kernel or checker.

Evaluation Taxonomy for Mathematical Autoformalization





Compilation / Pass Rate

Measures the proportion of generated formalizations accepted by the theorem prover. For statement autoformalization, this usually means successful elaboration and type checking; for proof autoformalization, it additionally requires formal proof verification.

Representative studies:
Ying et al. (2024) — Lean Workbook: Introduces Compile Pass Number (CPN) for evaluating Lean 4 statement formalizations.

Zhang et al. (2024) — Consistent Autoformalization: Evaluates syntactic correctness in Isabelle/ZF and investigates compiler-feedback correction.

Lu et al. (2024) — Process-Driven Autoformalization: Uses Lean 4 compiler feedback in evaluating and improving statement and proof formalization.

Coarse-grained: it distinguishes accepted from rejected outputs but provides limited information about the nature or severity of failures.

Error Analysis / Diagnosis

Uses Isabelle diagnostic feedback to categorize errors in automatically generated formal mathematical definitions, revealing different classes of syntactic failures that can be used to guide refinement.

Category	Reasons
SYN	1. **Invalid Symbol Format.** Isabelle uses symbols like "\<Rightarrow>" to represent "\Rightarrow ($\Rightarrow$)" in LaTeX. GPT-4o does not strictly follow this behaviour. A symbol in its formalized code starting with "\<" can miss ">" at the end so that the relevant symbol is not valid. 2. **Confusion of Mapping between LaTeX Mathematical Symbols and Isabelle Symbols.** Not all natural language symbols in LaTeX have a similar corresponding version in Isabelle symbols. In natural language mathematics we use different mathematical fonts such as "\mathcal ($\mathcal{A}$)" to distinguish items. Isabelle uses "\<A>" to represent this LaTeX symbol. However, GPT-4o would pretend the existence of a symbol named \<mathcal> and use it for autoformalization. 3. **Unaware of Name Conflict.** Some keywords such as "instance" are reserved by Isabelle/HOL and they cannot be used as the name of a new item. 4. **Incorrect Stylistic Usage of Symbols or Operators.** Some symbols or operators require specific usage which is not in the same style as in natural language. The incorrect usage of them in formalized code generated by GPT-4o can lead to syntax errors.
UDF	1. **Items not defined.** Formalization requires every mentioned item to be clearly defined in the local context or preambles. During autoformalization, GPT-4o could refer to items that are not defined in both sources.
TUF	1. **Mismatch between Types in Definition and Types in Actual Usage.** There are some operators or functions which have been clearly defined about the types of their operands or parameters. When using these operators or functions, the types of actual operands or parameters need to match the types in the definitions exactly. GPT-4o would produce mismatched types in the formalized codes and introduce TUF errors.

Autoformalization in the Wild: Assessing LLMs on Real-World Mathematical Definitions, Zhang et al., 2025

Human Evaluation

Human experts assess whether generated formalizations faithfully preserve the intended mathematical meaning of the informal source.

Representative studies:
Wu et al. (2022) — Autoformalization with Large Language Models: Manually evaluates 150 generated formalizations, identifying correct translations and categorizing errors.

Gao et al. (2025) — Herald: Human experts classify generated formalizations as correct, containing minor errors, or containing major errors.

Ying et al. (2024) — Lean Workbook: Uses human inspection to assess the quality of generated formalizations and identify translation errors.

Limitation: Human evaluation is expensive, difficult to scale, and potentially subjective.

n-gram / Code Similarity Metrics

Evaluate generated formalizations by measuring textual or structural similarity to reference formalizations, using existing n-gram and code metrics.

Representative studies:
Wu et al. (2022) — Autoformalization with Large Language Models: Uses BLEU to compare generated formalizations with reference statements.

Zhang et al. (2024) — Consistent Autoformalization for Constructing Mathematical Libraries: Applies BLEU, ChrF, RUBY, CodeBERTScore to evaluate generated Isabelle/ZF formalizations against references.

Limitation: Syntactically different formalizations may express the same mathematical meaning, while superficially similar formalizations may differ in important details.

ATP-Based Equivalence Checking

Uses symbolic equivalence checking to evaluate whether generated formalizations are equivalent to each other.

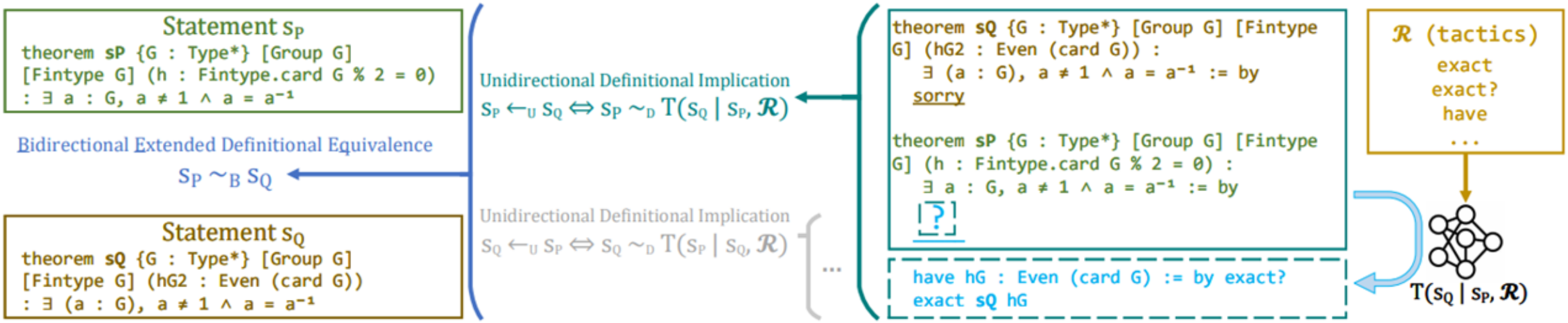
Definition 1 (Symbolic equivalence) *Let two mathematical statements Ψ_1 and Ψ_2 in formal language be expressed as $\mathcal{P}_1 \to \mathcal{Q}_1$ and $\mathcal{P}_2 \to \mathcal{Q}_2$, and suppose their premises $\mathcal{P}_1$ and $\mathcal{P}_2$ are not tautologies. Then, the two statements are called symbolically equivalent if the two logical equivalences, i.e., $\mathcal{P}_1 \equiv \mathcal{P}_2$ and $\mathcal{Q}_1 \equiv \mathcal{Q}_2$, both hold.*

The two logical equivalences induced by the symbolic equivalence can be determined through existing automated theorem provers (ATPs). Additionally, the validity of the premises can also be checked by replacing the conclusion $\mathcal{Q}$ with a contradictory result (e.g., $0 = 1$) and verifying $\mathcal{P} \to \mathcal{Q}$. If this vacuous form can be proved, then the corresponding premise $\mathcal{P}$ is a contradiction.

Autoformalizing Mathematical Statements by Symbolic Equivalence and Semantic Consistency, Li et al., 2024

ATP-Based Equivalence Checking

Introduces BEq, an automated equivalence-based metric for evaluating formal statements against references.



Rethinking and Improving Autoformalization, Liu et al., 2025

Theorem Prover Tactics

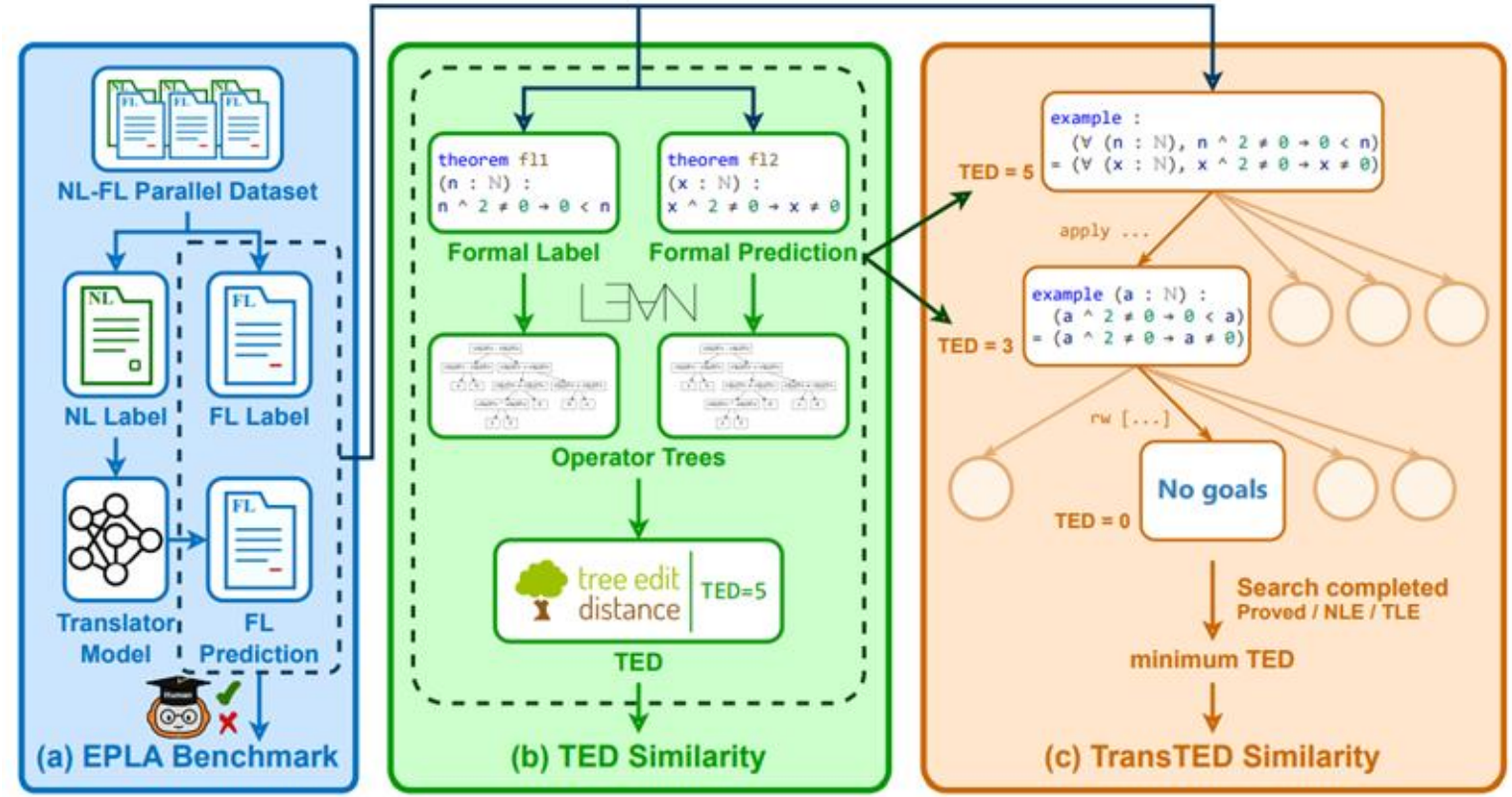
Introduces BEq+, using theorem-prover tactics to improve equivalence evaluation and investigate reference reliability.

Input: Theorem formalizations t_1 and t_2
Output: Whether t_2 can be derived from t_1
1. Run BEq_L
if `exact?` closes proof by using t_1 **then**
  return TRUE
2. Leverage conclusion matching
if `apply t_1` or `convert t_1` succeeds **then**
  Proving t_1 assumptions can be derived from t_2
  if repeated applications of `tauto`, `simp_all_arith!`, `noncomm_ring`, or `exact?` close proof **then**
    return TRUE
3. Attempt direct assumption of t_1
if `have : goal(t_1) := by apply_rules [t_1]` succeeds **then**
  if repeated applications of `tauto`, `simp_all_arith!`, `noncomm_ring`, or `exact?` close the subproof introduced by `have` **then**
    if repeated applications of `tauto`, `simp_all_arith!`, or `exact? using this` close the main proof **then**
      return TRUE
return FALSE

Reliable Evaluation and Benchmarks for Statement Autoformalization, Poiroux et al., 2025

Syntax Tree Comparison

Introduces a framework for evaluating statement similarity using semantic and structural information, including syntax-tree comparisons.



ASSESS: A Semantic and Structural Evaluation Framework for Statement Similarity, Liu et al., 2026

Limitations of Reference-Based Evaluation Methods

- **Reference Correctness:** Ground-truth formalizations may contain errors, even in established benchmarks (Faults in Our Formal Benchmarking, Ammanamanchi et al., ICML 2026).
- **One-to-Many Nature:** Multiple syntactically different formalizations may express the same mathematical meaning. A single reference may therefore underestimate correctness.
- **Imperfect Equivalence Estimation:** Automatic equivalence checkers may produce incorrect or inconclusive judgments.
- **Limited Applicability:** Formal equivalence checking generally cannot evaluate syntactically invalid outputs, even when they preserve much of the intended meaning.

Back-Translation+Embedding Similarity

Evaluates informal–formal consistency by back-translating generated statements and measuring embedding similarity.

Definition 2 (Semantic consistency) *Let the original mathematical statement in natural language and its formalization candidate be* Φ *and* Ψ*, and suppose that* Ψ *is further informalized into a new natural language statement* $\tilde{\Psi}$*. Then, the formal statement* Ψ *is* τ*-semantically consistent with the original statement* Φ *if the embedding similarity between* $\tilde{\Psi}$ *and* Φ *satisfies* $\mathrm{Sim}(\tilde{\Psi}, \Phi) \geq \tau$.

Autoformalizing Mathematical Statements by Symbolic Equivalence and Semantic Consistency, Li et al., 2024

Back-Translation+NLI

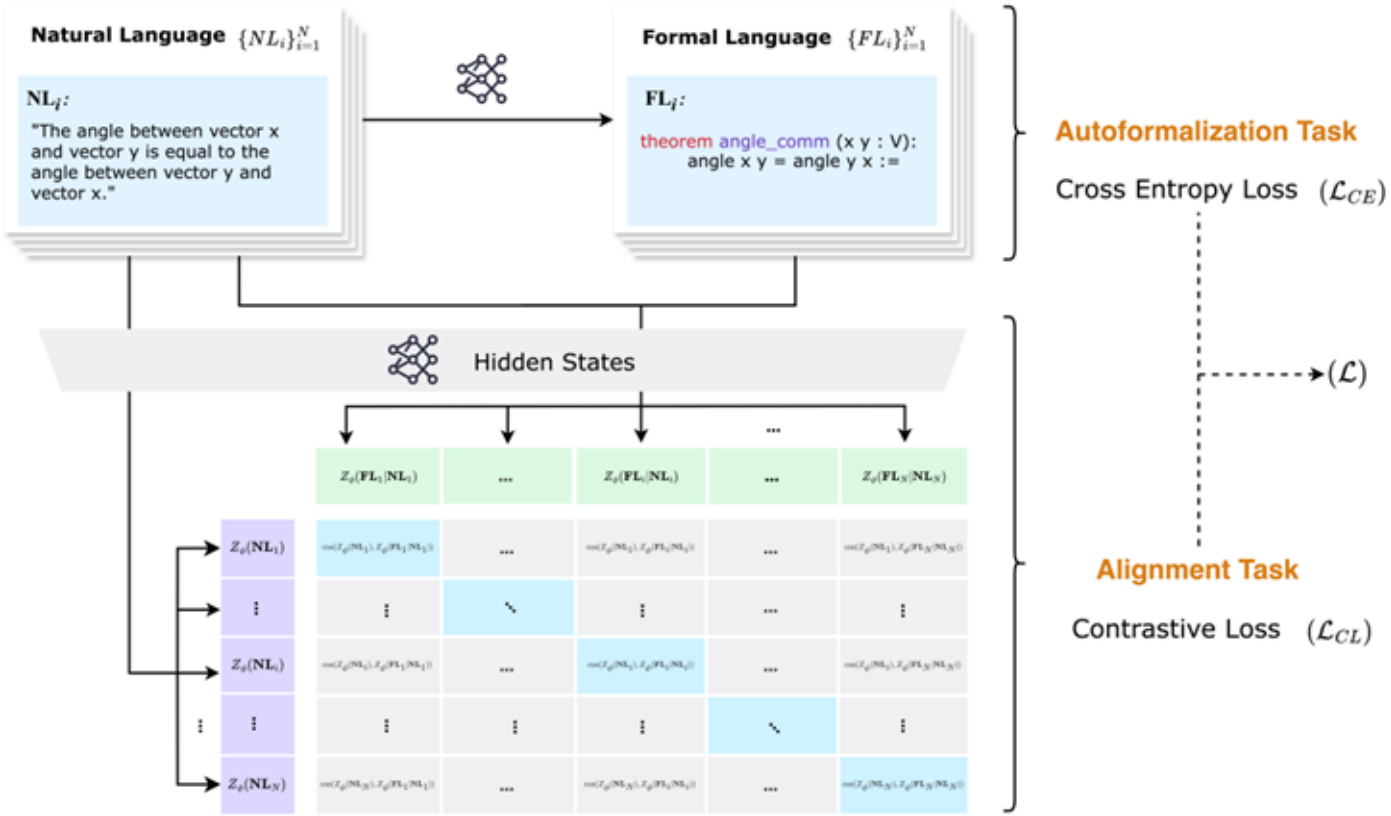
Uses back-translation and natural-language inference (NLI) to assess the correspondence between original statements and generated formalizations. Different studies employ different NLI models.

3. **Back-translation**: For statements that pass the validation in step 2, we used InternLM2-Math-Plus-7B to translate the formal statements back to natural language to assess the preservation of meaning.
4. **NLI check**: We use the DeepSeek Chat v2.5 model to compare the back-translated statements with the original informal statements, ensuring that our translations are mathematically accurate and preserve the intended meaning.

Ying et al. (2024), Gao et al. (2025), Liu et al. (2025d), Lu et al. (2026)

Learned Alignment Models

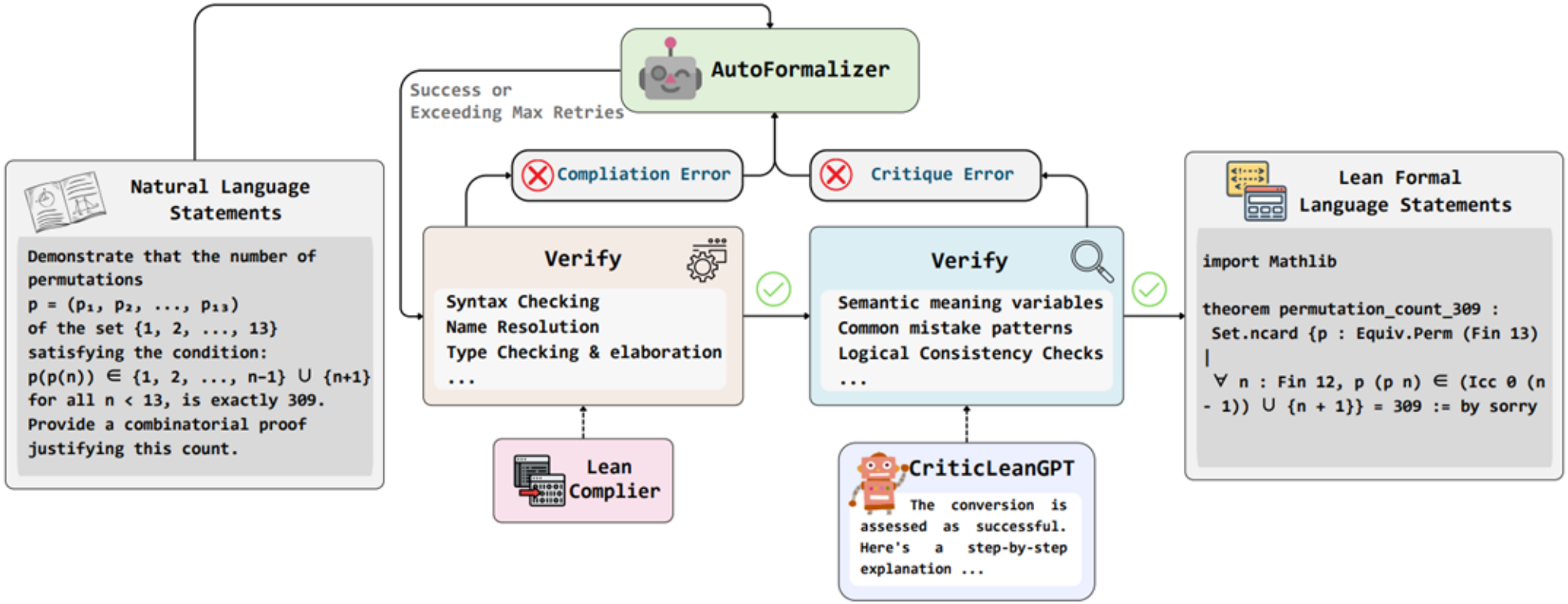
Introduces an automatic alignment evaluator that estimates correspondence between informal mathematics and generated formalizations.



FormalAlign: Automated Alignment Evaluation for Autoformalization, Lu et al., 2025

Learned Alignment Models

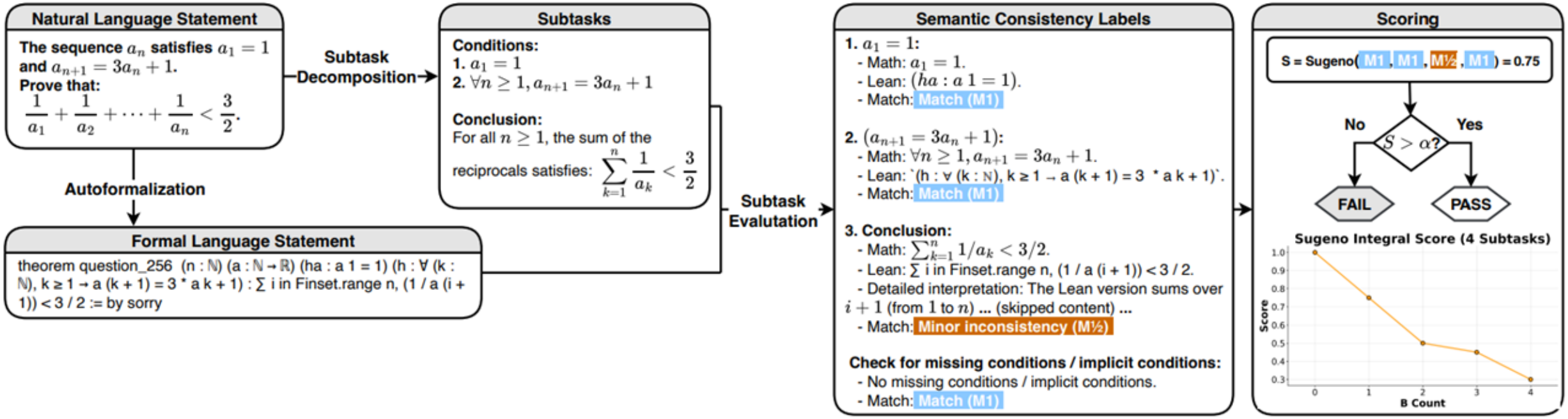
Uses a learned critic to assess formalization quality and provide an evaluation signal for improving mathematical formalization.



CriticLean: Critic-Guided Reinforcement Learning for Mathematical Formalization, Peng et al., 2025

LLM-as-a-Judge

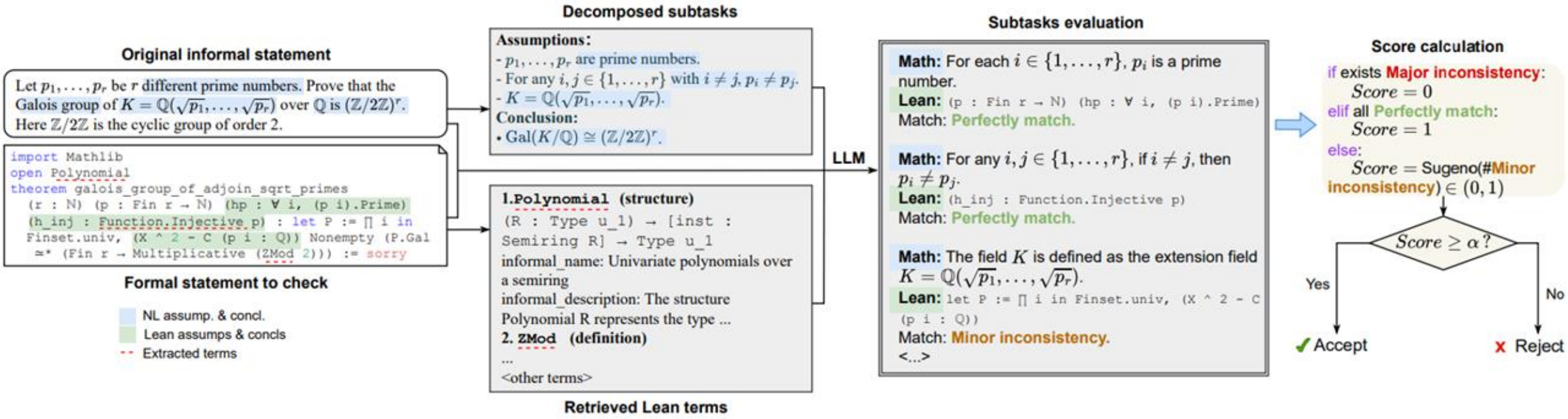
Introduces LeanScorer, which combines multiple assessment signals, including LLM-based evaluation, to judge generated formalizations.



Mathesis: Towards Formal Theorem Proving from Natural Languages, Xuejun et al., 2026

LLM-as-a-Judge

Introduces AriaScorer, an LLM-based evaluator that assesses mathematical concept grounding and formalization correctness.



Aria: an Agent for Retrieval and Iterative Auto-Formalization via Dependency Graph, Wang et al., 2026

LLM-as-a-Judge: Is Judging Alignment Alone Enough?

Simply asking an LLM to judge semantic alignment may produce **overly coarse-grained evaluations**, even with subtask decomposition. **Autoformalization quality is inherently multidimensional** and cannot be reduced to whether a formalization faithfully captures its informal counterpart.

Case 1: miniF2F Isabelle
Unpreserved Logic
Find the minimum value of $\frac{9x^2 \sin^2 x + 4}{x \sin x}$ for $0 < x < \pi$. Show that it is 012.

```
theorem aime_1983_p9:
  fixes x::real
  assumes "0<x" "x<pi"
  shows "12 ≤ ((9 * (x^2 *
    (sin x)^2)) + 4) / (x * sin x)"
    sorry
```

Case 2: Proofnet Lean4
Inconsistent Mathematics
Let A be a nonempty set of real numbers which is bounded below. Let $-A$ be the set of all numbers $-x$, where $x \in A$. Prove that $\inf A = -\sup(-A)$.

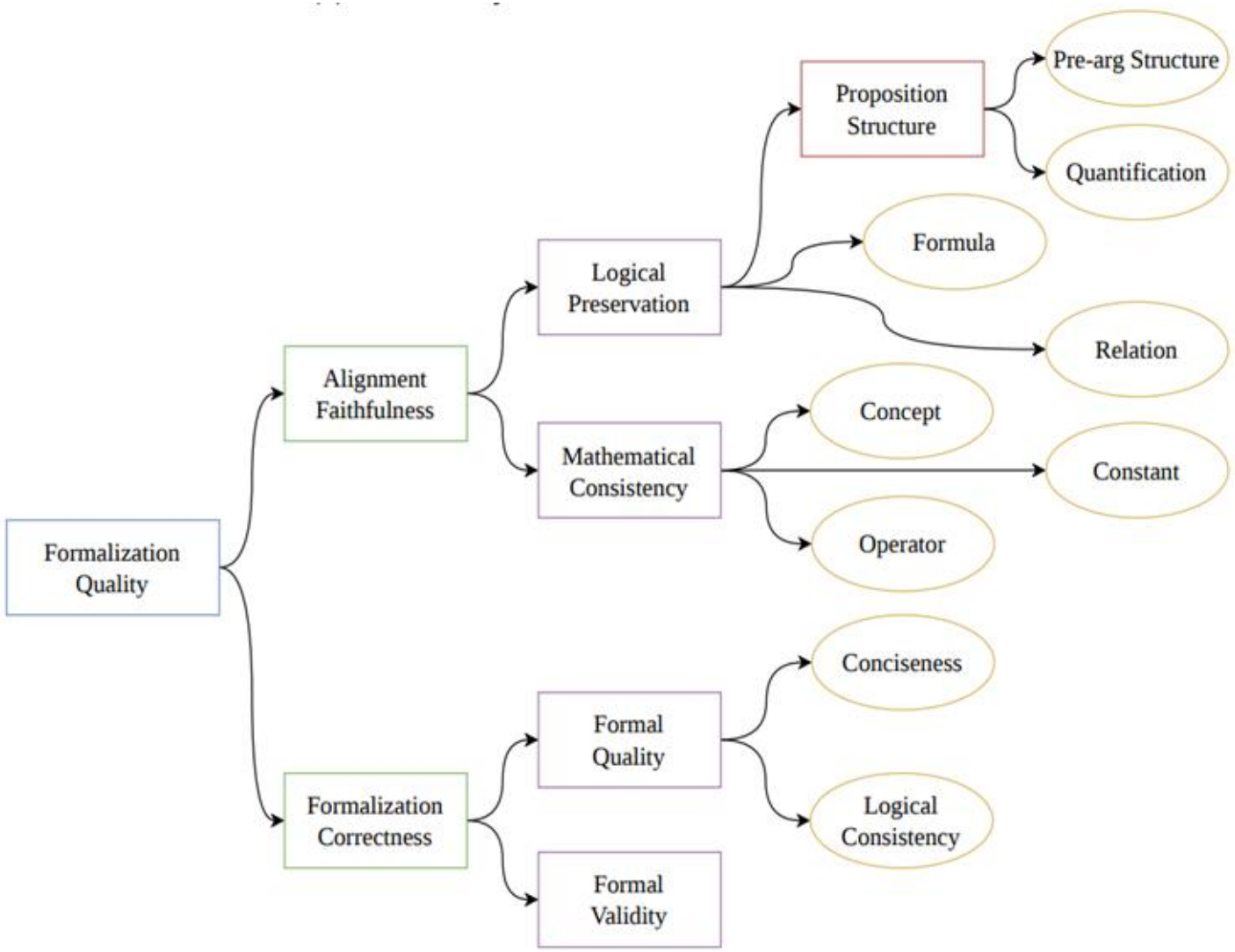
```
theorem exercise_1_5
  (A minus_A : Set ℝ) (hA : A.Nonempty)
  (hA_bdd_below : BddBelow A)
  (hminus_A : minus_A = {x | -x ∈ A}) :
  Inf A = Sup minus_A :=sorry
```

Case 3: miniF2F Lean4
Redundant Hypothesis
Simplify $(2 - 2i)(5 + 5i)$, where $i^2 = -1$. Show that it is 20.

```
theorem mathd_algebra_110
  (q e : ℂ)
  (h₀ : q = 2 - 2 * Complex.I)
  (h₁ : e = 5 + 5 * Complex.I) :
  q * e = 20 := sorry
```

Beyond Gold Standards, Zhang et al., 2026

LLM-as-a-Judge: An Evaluation Taxonomy



Decompose autoformalization quality into four dimensions of autoformalization quality, and operationalize 9 interpretable atomic properties for aspects beyond theorem-prover checking.

Beyond Gold Standards, Zhang et al., 2026

Limitations of Reference-Free Evaluation Methods

- **Back-Translation:** May introduce information loss or linguistic variation, making it difficult to distinguish translation errors from formalization errors.
- **Learned Alignment Models:** Require supervised training data and may generalize poorly across mathematical domains, formal languages, and formalization styles.
- **LLM-as-a-Judge:** Reliability, calibration, and robustness remain insufficiently understood, particularly for complex or out-of-distribution mathematics.
- **Reference-free evaluation reduces dependence on ground-truth formalizations, but introduces uncertainty about the reliability of the evaluator itself.**

Key Takeaways

- **Syntactic validity is necessary but insufficient.** Theorem-prover acceptance verifies formal well-formedness or proof correctness, but cannot by itself establish semantic alignment.
- **Semantic evaluation involves different trade-offs.** Human evaluation is costly; similarity metrics can miss mathematical equivalence; formal equivalence depends on executable candidates; learned evaluators can be unreliable.
- **Evaluation methods are complementary, not mutually exclusive.** A single paper may combine compilation checks, human judgments, formal equivalence, back-translation, and LLM evaluation.
- **No universally reliable semantic evaluation metric exists.** We therefore recommends reporting syntactic validity and semantic alignment separately, with explicit benchmark and theorem-prover settings.

Are Reported Performance Across Studies Really Comparable?

Paper	Task	Formal Language	Version	Tested Formal Bench.	Reported Eval.
Wu et al. (2022)	S	Isabelle/HOL	Unknown	miniF2F	GCM, HE
Zhang et al. (2024)	S	Isabelle/ZF	2023	MathLibForm	PCR, GCM
Li et al. (2024)	S	Isabelle/HOL	2023	miniF2F	ATPE, BT
Ying et al. (2024)	S	Lean4	v4.7.0	miniF2F, Lean Workbook	PCR, BT, HE
Gao et al. (2025)	S	Lean4	v4.11.0	miniF2F, ProofNet	PCR, BT, HE
Liu et al. (2025b)	S	Lean4	4.7.0-rc2	ProofNet, Con-NF	PCR, ATPE, HE
Liu et al. (2025d)	S	Lean4	v4.19.0-rc2	ProofNet, PutnamBench	PCR, BT, HE
Lu et al. (2026)	S	Lean4	4.20.0-rc5	miniF2F, ProofNet	PCR, BT
Wang et al. (2026)	S	Lean4	Unknown	ProofNet, Fate	PCR, LLM
Xuejun et al. (2026)	S	Lean4	Unknown	miniF2F, PutnamBench, Gaokao-Formal	PCR, LLM, HE
Zhang et al. (2026b)	S	Lean4	4.7.0, 4.18.0	miniF2F, ProofNet, Con-NF	PCR, ATPE
Jiang et al. (2023)	P	Isabelle/HOL	2021	miniF2F	PCR
Tarrach et al. (2024)	P	Isabelle/HOL	2022	miniF2F	PCR
Lu et al. (2024)	F	Lean4	v4.7.0-rc2	FORML4	PCR, HE
Zhang et al. (2026a)	F	Lean4	v4.19.0-rc3	miniF2F, ProofNet	PCR, LLM, HE

Task differences: S: Statement Only; **P**: Proof Only; **F**: Statement+Proof

Are Reported Performance Across Studies Really Comparable?

Paper	Task	Formal Language	Version	Tested Formal Bench.	Reported Eval.
Wu et al. (2022)	S	Isabelle/HOL	Unknown	miniF2F	GCM, HE
Zhang et al. (2024)	S	Isabelle/ZF	2023	MathLibForm	PCR, GCM
Li et al. (2024)	S	Isabelle/HOL	2023	miniF2F	ATPE, BT
Ying et al. (2024)	S	Lean4	v4.7.0	miniF2F, Lean Workbook	PCR, BT, HE
Gao et al. (2025)	S	Lean4	v4.11.0	miniF2F, ProofNet	PCR, BT, HE
Liu et al. (2025b)	S	Lean4	4.7.0-rc2	ProofNet, Con-NF	PCR, ATPE, HE
Liu et al. (2025d)	S	Lean4	v4.19.0-rc2	ProofNet, PutnamBench	PCR, BT, HE
Lu et al. (2026)	S	Lean4	4.20.0-rc5	miniF2F, ProofNet	PCR, BT
Wang et al. (2026)	S	Lean4	Unknown	ProofNet, Fate	PCR, LLM
Xuejun et al. (2026)	S	Lean4	Unknown	miniF2F, PutnamBench, Gaokao-Formal	PCR, LLM, HE
Zhang et al. (2026b)	S	Lean4	4.7.0, 4.18.0	miniF2F, ProofNet, Con-NF	PCR, ATPE
Jiang et al. (2023)	P	Isabelle/HOL	2021	miniF2F	PCR
Tarrach et al. (2024)	P	Isabelle/HOL	2022	miniF2F	PCR
Lu et al. (2024)	F	Lean4	v4.7.0-rc2	FORML4	PCR, HE
Zhang et al. (2026a)	F	Lean4	v4.19.0-rc3	miniF2F, ProofNet	PCR, LLM, HE

Formal-environment differences: Isabelle/HOL, Isabelle/ZF, and Lean4 with various versions.

Are Reported Performance Across Studies Really Comparable?

Paper	Task	Formal Language	Version	Tested Formal Bench.	Reported Eval.
Wu et al. (2022)	S	Isabelle/HOL	Unknown	miniF2F	GCM, HE
Zhang et al. (2024)	S	Isabelle/ZF	2023	MathLibForm	PCR, GCM
Li et al. (2024)	S	Isabelle/HOL	2023	miniF2F	ATPE, BT
Ying et al. (2024)	S	Lean4	v4.7.0	miniF2F, Lean Workbook	PCR, BT, HE
Gao et al. (2025)	S	Lean4	v4.11.0	miniF2F, ProofNet	PCR, BT, HE
Liu et al. (2025b)	S	Lean4	4.7.0-rc2	ProofNet, Con-NF	PCR, ATPE, HE
Liu et al. (2025d)	S	Lean4	v4.19.0-rc2	ProofNet, PutnamBench	PCR, BT, HE
Lu et al. (2026)	S	Lean4	4.20.0-rc5	miniF2F, ProofNet	PCR, BT
Wang et al. (2026)	S	Lean4	Unknown	ProofNet, Fate	PCR, LLM
Xuejun et al. (2026)	S	Lean4	Unknown	miniF2F, PutnamBench, Gaokao-Formal	PCR, LLM, HE
Zhang et al. (2026b)	S	Lean4	4.7.0, 4.18.0	miniF2F, ProofNet, Con-NF	PCR, ATPE
Jiang et al. (2023)	P	Isabelle/HOL	2021	miniF2F	PCR
Tarrach et al. (2024)	P	Isabelle/HOL	2022	miniF2F	PCR
Lu et al. (2024)	F	Lean4	v4.7.0-rc2	FORML4	PCR, HE
Zhang et al. (2026a)	F	Lean4	v4.19.0-rc3	miniF2F, ProofNet	PCR, LLM, HE

Benchmark differences: studies evaluate different subsets or additional datasets.

Are Reported Performance Across Studies Really Comparable?

Paper	Task	Formal Language	Version	Tested Formal Bench.	Reported Eval.
Wu et al. (2022)	S	Isabelle/HOL	Unknown	miniF2F	GCM, HE
Zhang et al. (2024)	S	Isabelle/ZF	2023	MathLibForm	PCR, GCM
Li et al. (2024)	S	Isabelle/HOL	2023	miniF2F	ATPE, BT
Ying et al. (2024)	S	Lean4	v4.7.0	miniF2F, Lean Workbook	PCR, BT, HE
Gao et al. (2025)	S	Lean4	v4.11.0	miniF2F, ProofNet	PCR, BT, HE
Liu et al. (2025b)	S	Lean4	4.7.0-rc2	ProofNet, Con-NF	PCR, ATPE, HE
Liu et al. (2025d)	S	Lean4	v4.19.0-rc2	ProofNet, PutnamBench	PCR, BT, HE
Lu et al. (2026)	S	Lean4	4.20.0-rc5	miniF2F, ProofNet	PCR, BT
Wang et al. (2026)	S	Lean4	Unknown	ProofNet, Fate	PCR, LLM
Xuejun et al. (2026)	S	Lean4	Unknown	miniF2F, PutnamBench, Gaokao-Formal	PCR, LLM, HE
Zhang et al. (2026b)	S	Lean4	4.7.0, 4.18.0	miniF2F, ProofNet, Con-NF	PCR, ATPE
Jiang et al. (2023)	P	Isabelle/HOL	2021	miniF2F	PCR
Tarrach et al. (2024)	P	Isabelle/HOL	2022	miniF2F	PCR
Lu et al. (2024)	F	Lean4	v4.7.0-rc2	FORML4	PCR, HE
Zhang et al. (2026a)	F	Lean4	v4.19.0-rc3	miniF2F, ProofNet	PCR, LLM, HE

Evaluation differences: semantic alignment is assessed using heterogeneous methods measuring different aspects of semantic quality and rely on different assumptions.

What Should Autoformalization Studies Report?

To improve reproducibility and comparability, studies should at minimum explicitly:

1. Specify task, dataset, exact split and exclusions.
2. Pin prover, math library, dependencies and version.
3. Disclose prompts, decoding configuration, search and compute budget.
4. Report syntax and semantic metrics separately with uncertainty.
5. Release traces, failed examples and evaluation scripts when feasible.

However, reporting these remains insufficient to ensure fair comparisons.

What We Urgently Need: A Standardized Evaluation Testbed

A unified testbed with consistent benchmarks, controlled formal environments, and standardized evaluation protocols is essential for reliably comparing autoformalization methods and identifying genuine methodological improvements.

Even Frontier Models Face Evaluation Challenges

August 1, 2026: OpenAI announced "Ten advances in mathematics and theoretical computer science", supported by publicly released formalizations.

On the same date: J. L. Nielsen (2026) challenged the claimed counterexample to Connes' rigidity conjecture and presented a competing mathematical argument.

September 8, 2026: OpenAI announced "On the Navier–Stokes Millennium Prize Problem", claiming a solution to the Navier–Stokes Millennium Prize Problem, supported by a Lean formalization.

October 6, 2026: Bastounis et al. (2026) identified alleged discrepancies between OpenAI's Navier–Stokes manuscript and its Lean formalization.

October 6, 2026: OpenAI announced "Sharing AI progress in mathematics":

- Originally 722 mathematical manuscripts, organized into 372 result families.
- Approximately 42% of top-line results have Lean formalizations.

October 7, 2026: Within 24 hours, OpenAI withdrew three manuscripts due to a mathematical error and revised 14 others. The withdrawn results did not have Lean-formalized main theorems.

Autoformalization offers a promising way to **verify AI-generated mathematical advances**. However, **manually** assessing formalizations at scale is **impractical and prohibitively time-consuming**.

As AI accelerates mathematical discovery, **automated**, **rigorous**, and **standardized evaluation** of autoformalization must keep pace. Otherwise, we risk mistaking reported progress for genuine mathematical advances.

⊢ Concluding remarks