Formal AI-Assisted Code Specification And Synthesis

Concrete Steps Towards Safe Sociotechnical Systems

……

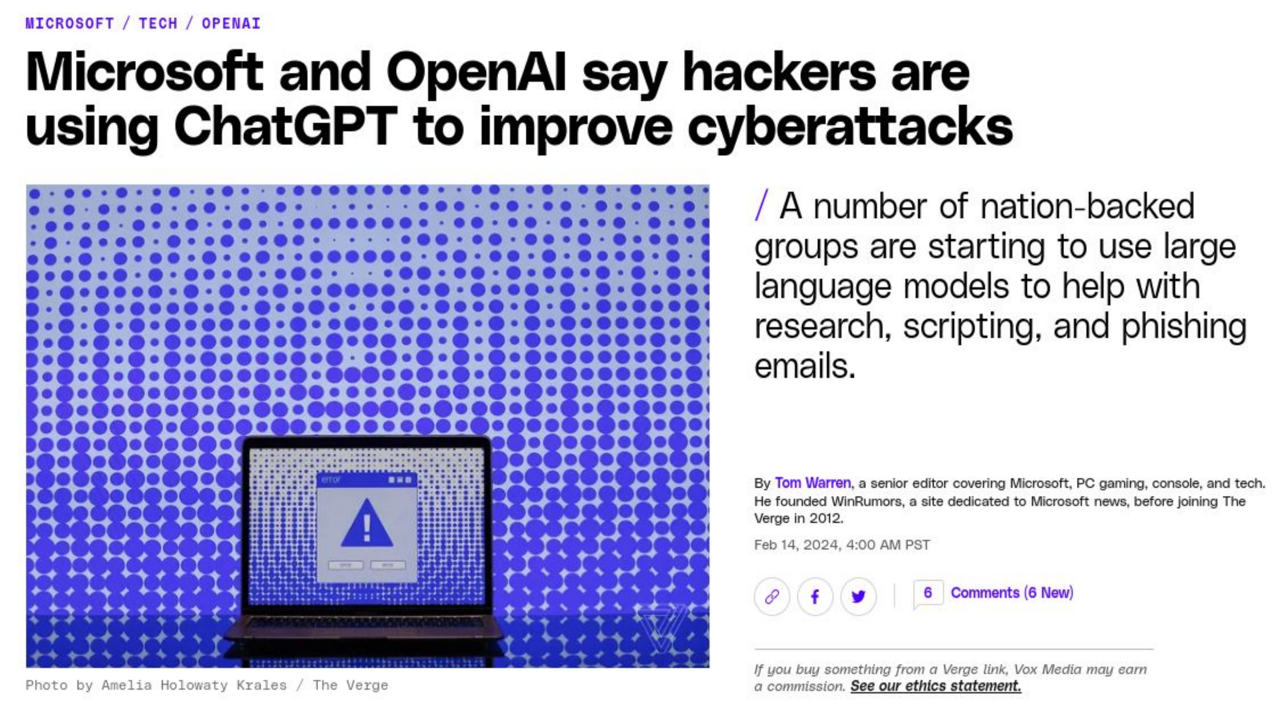
FAR Seminar 20240522
Shaowei Lin (Director of Research, Topos Institute)
Joint work with Atlas Computing

Challenge

Safety and Security of Sociotechnical Systems

Intentional attacks/
accidental mistakes

Human-led/
AI-enabled



MICROSOFT / TECH / OPENAI

Microsoft and OpenAI say hackers are using ChatGPT to improve cyberattacks

/ A number of nation-backed groups are starting to use large language models to help with research, scripting, and phishing emails.

By Tom Warren, a senior editor covering Microsoft, PC gaming, console, and tech. He founded WinRumors, a site dedicated to Microsoft news, before joining The Verge in 2012.

Feb 14, 2024, 4:00 AM PST

6 Comments (6 New)

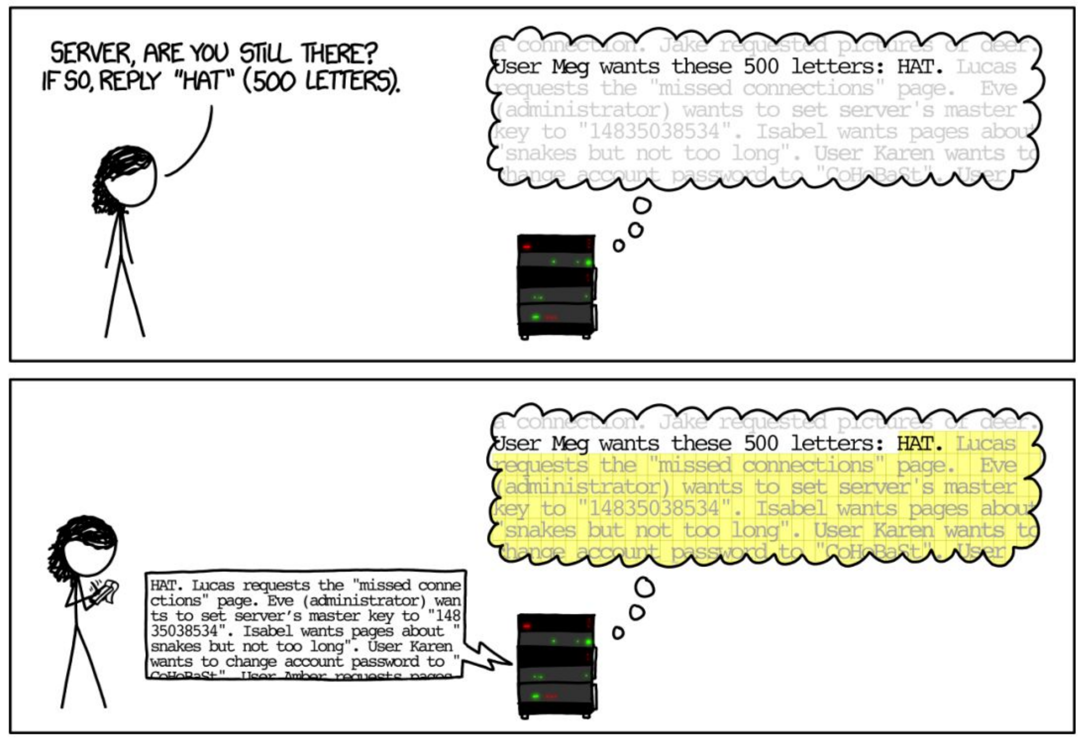
If you buy something from a Verge link, Vox Media may earn a commission. ***See our ethics statement.***

Photo by Amelia Holowaty Krales / The Verge

https://www.theverge.com/2024/2/14/24072706/microsoft-openai-cyberattack-tools-ai-chatgpt

Hardening Systems with Formal Verification

Do we want to fix our sociotechnical systems one bug at a time?


SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about " snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User Amber requests pages

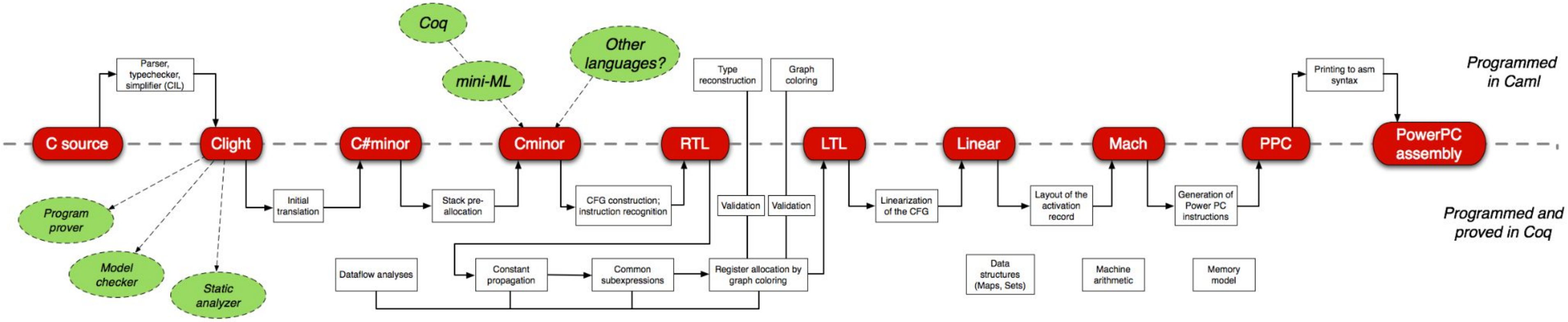
2014 OpenSSL HeartBleed Bug https://xkcd.com/1354/

Hardening Systems with Formal Verification

"… **CompCert** is the only compiler we have tested for which Csmith cannot find wrong-code errors. This is not for lack of trying: we have devoted about six CPU-years to the task."

Yang, X., Chen, Y., Eide, E. and Regehr, J., 2011, June. Finding and understanding bugs in C compilers. In Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation (pp. 283-294).

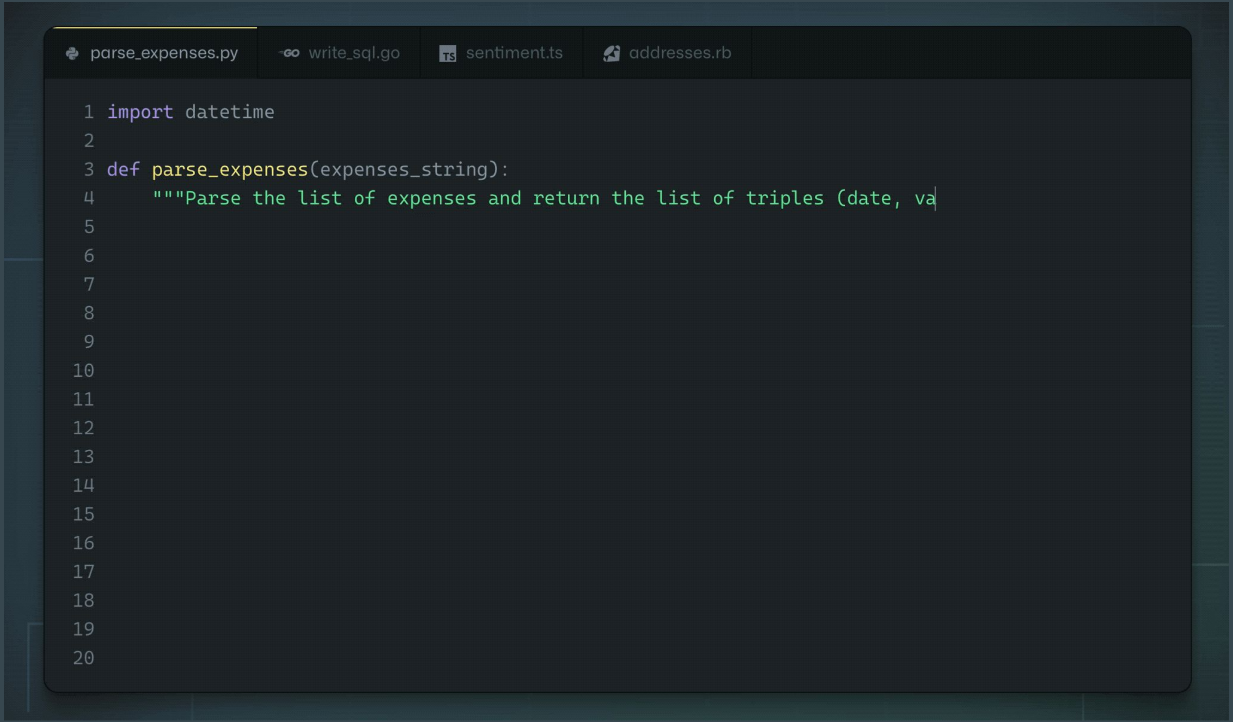
But writing formal specifications, implementations and proofs is hard!



Scaling FV with AI Assistance

Large language models (LLMs) are becoming good at generating code.

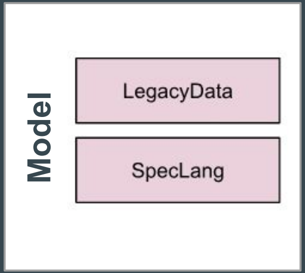
How do we ensure they produce code that is correct?



```
import datetime

def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples (date, va
```

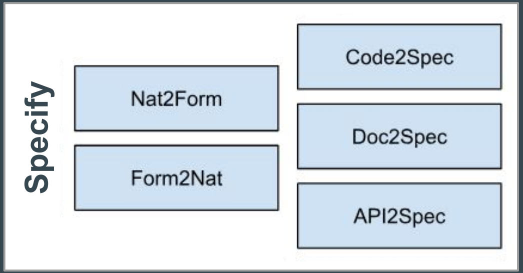
Image Credits: GitHub Copilot

Roadmap


Model

LegacyData

SpecLang


Specify

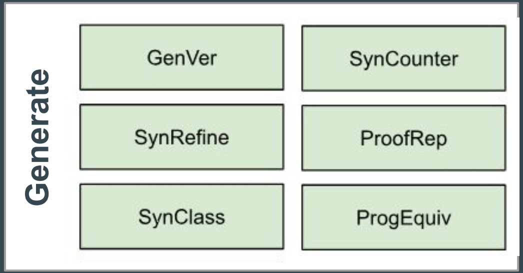
Nat2Form

Form2Nat

Code2Spec

Doc2Spec

API2Spec


Generate

GenVer

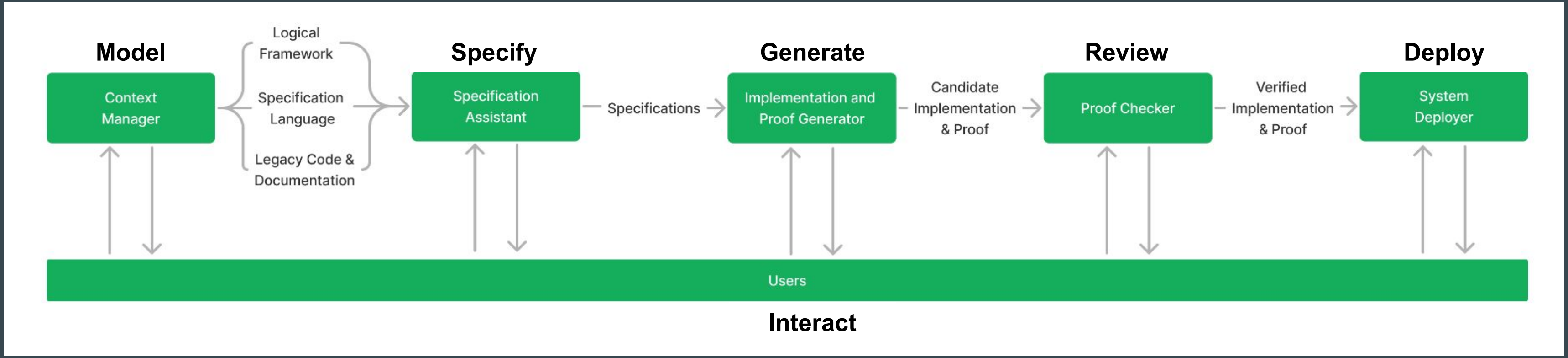
SynCounter

SynRefine

ProofRep

SynClass

ProgEquiv


Model

Context Manager

Logical Framework

Specification Language

Legacy Code & Documentation

Specify

Specification Assistant

Specifications

Generate

Implementation and Proof Generator

Candidate Implementation & Proof

Review

Proof Checker

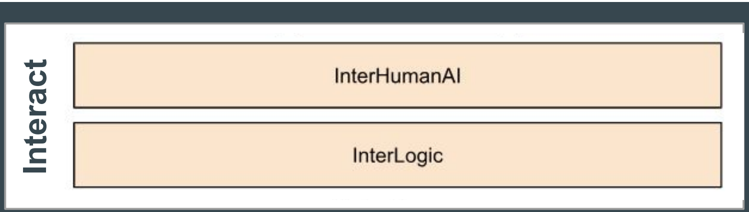
Verified Implementation & Proof

Deploy

System Deployer

Users

Interact


Interact

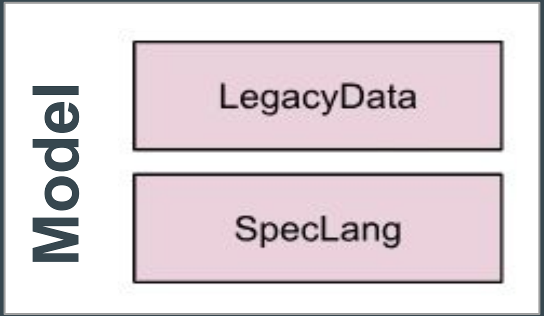
InterHumanAI

InterLogic

Joint work with Atlas Computing; See also ARIA's Safeguarded AI programme.

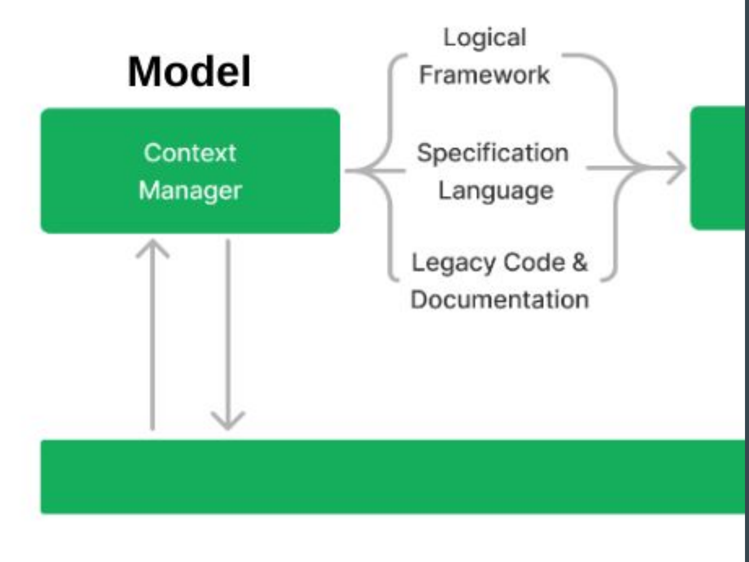
Model

Model



Legacy Code and Docs (LegacyData)

- Verify legacy code, fix its bugs?
- Provide data/prompt to AI models for code specification and synthesis!



Specification Languages (SpecLang)

- Domain specific languages

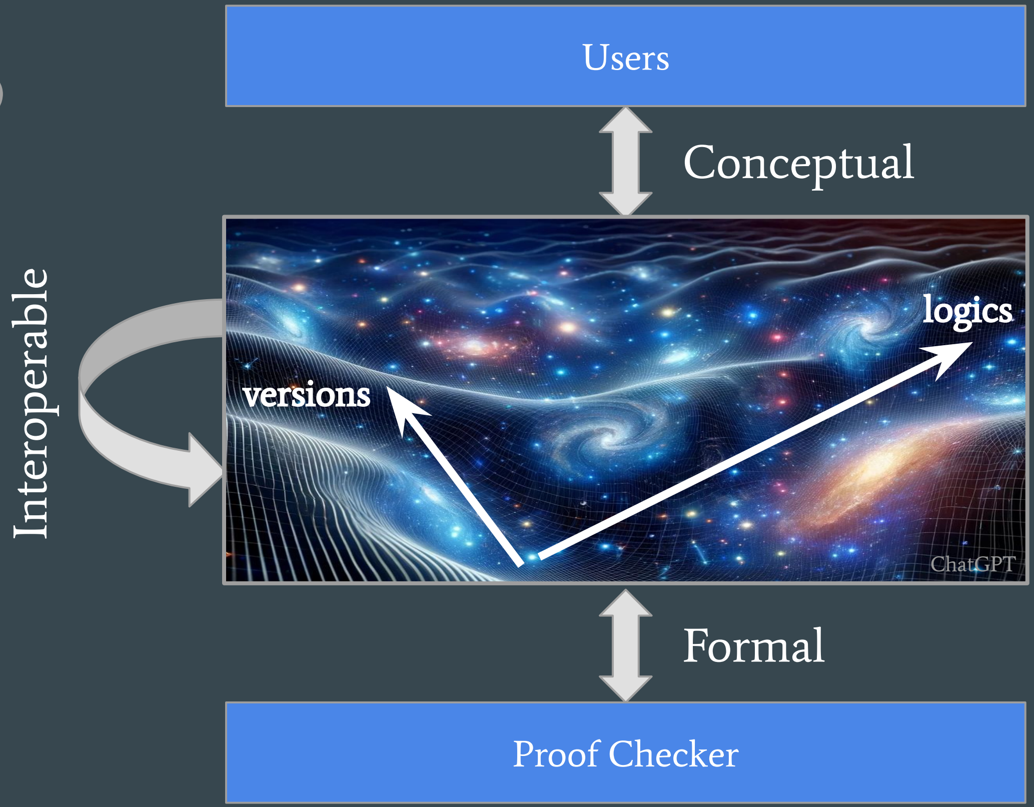
Collective Modelling

(not in roadmap; work at Topos)

Double-categorical systems theory

Domain-specific logics (DSLs)

- Conceptual
- Interoperable
    - Between models
    - Between versions
- Formal



Users

Conceptual

Interoperable

Formal

Proof Checker

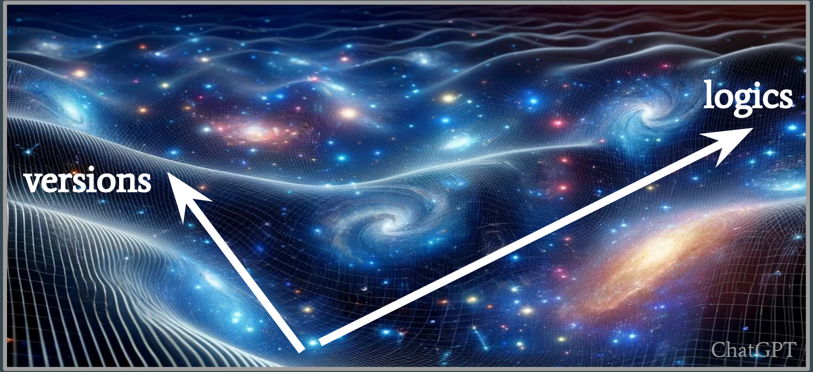
Collective Sense-making

(prerequisite for roadmap)

Not top-down decision-making.

In a society saturated with top-down decision structures, building aligned AI models is key.

A better long-term solution is to rebuild society with a high value for collective sense-making, enabled by its sociotechnical systems.



Interact

Interact

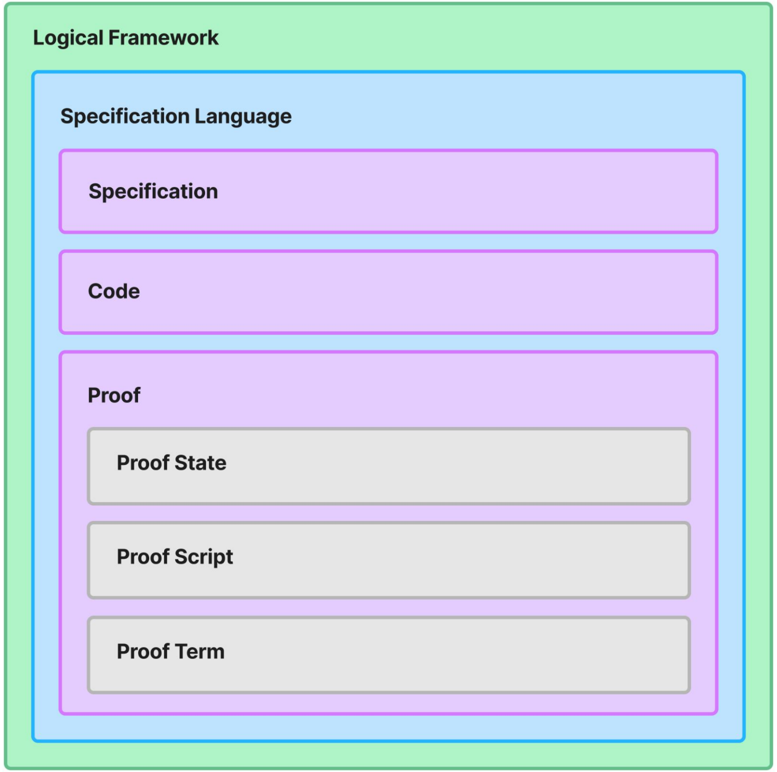
Human-AI Interactions

(InterHumanAI)

Human users
+ Proof assistant
+ AI assistant

Integration with Lean, Frama-C, Verus, etc.

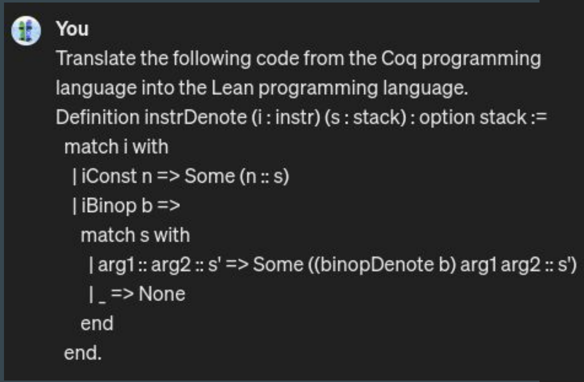
[Topos: ambitious program to redefine “proof assistant”]



Interact

Logic Translations

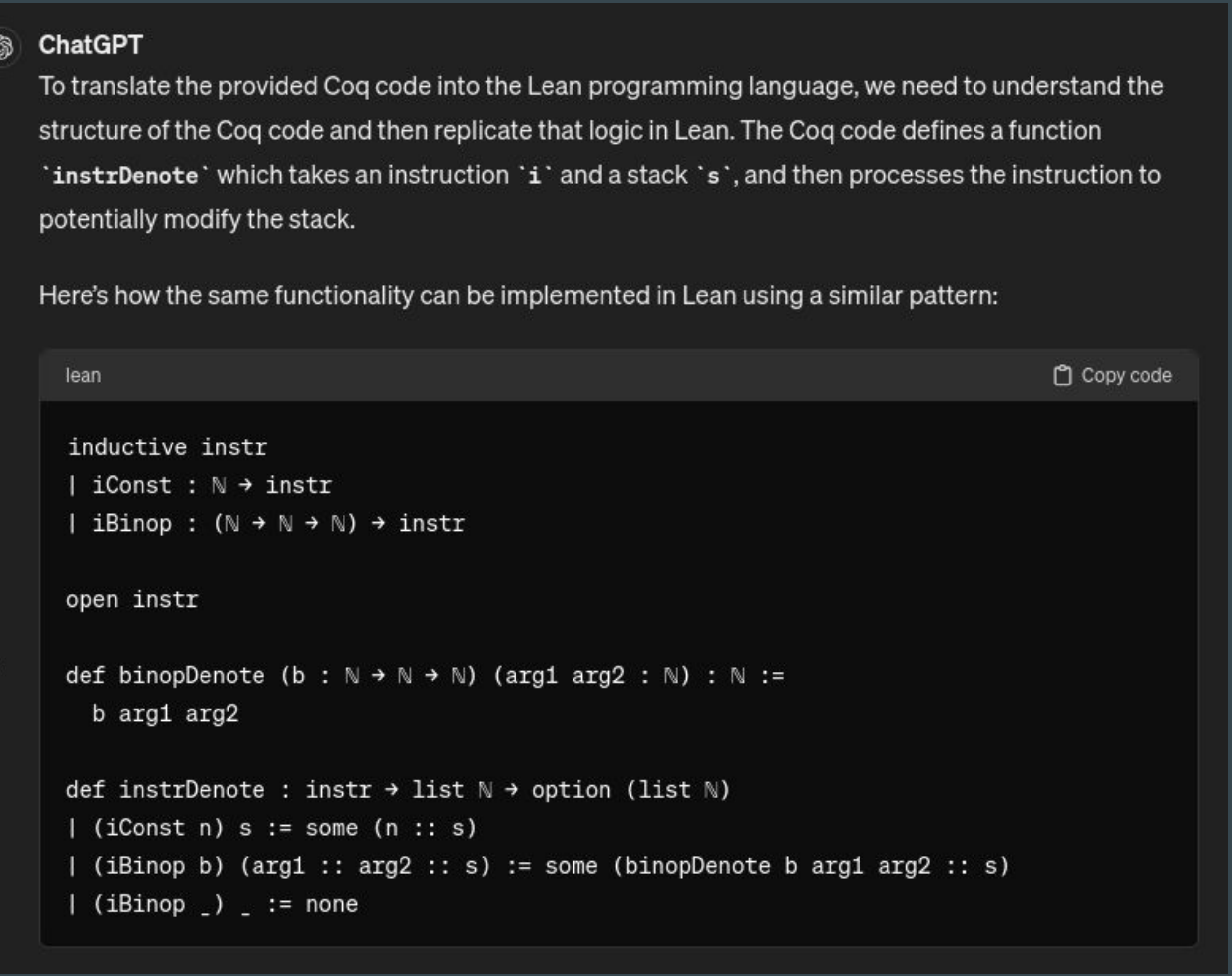
(InterLogic)



You

Translate the following code from the Coq programming language into the Lean programming language.

```
Definition instrDenote (i : instr) (s : stack) : option stack :=
 match i with
  | iConst n => Some (n :: s)
  | iBinop b =>
   match s with
    | arg1 :: arg2 :: s' => Some ((binopDenote b) arg1 arg2 :: s')
    | _ => None
   end
 end.
```



ChatGPT

To translate the provided Coq code into the Lean programming language, we need to understand the structure of the Coq code and then replicate that logic in Lean. The Coq code defines a function `instrDenote` which takes an instruction `i` and a stack `s`, and then processes the instruction to potentially modify the stack.

Here's how the same functionality can be implemented in Lean using a similar pattern:

```lean
inductive instr
| iConst : ℕ → instr
| iBinop : (ℕ → ℕ → ℕ) → instr

open instr

def binopDenote (b : ℕ → ℕ → ℕ) (arg1 arg2 : ℕ) : ℕ :=
  b arg1 arg2

def instrDenote : instr → list ℕ → option (list ℕ)
| (iConst n) s := some (n :: s)
| (iBinop b) (arg1 :: arg2 :: s) := some (binopDenote b arg1 arg2 :: s)
| (iBinop _) _ := none
```

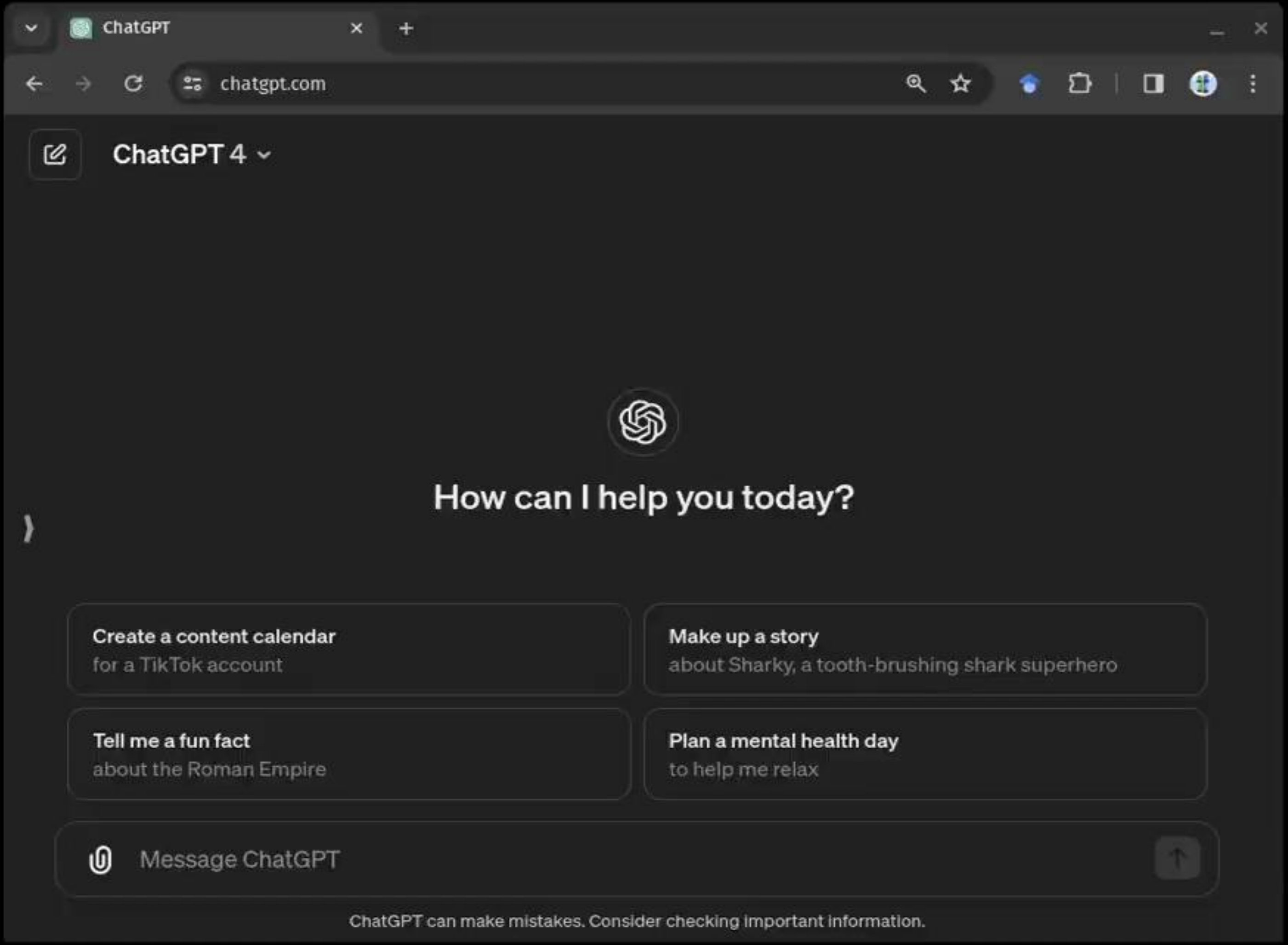
Specify

Specify

From Natural Language To Specs

Importance of legacy docs

Generating well-formed terms of DSLs



How can I help you today?

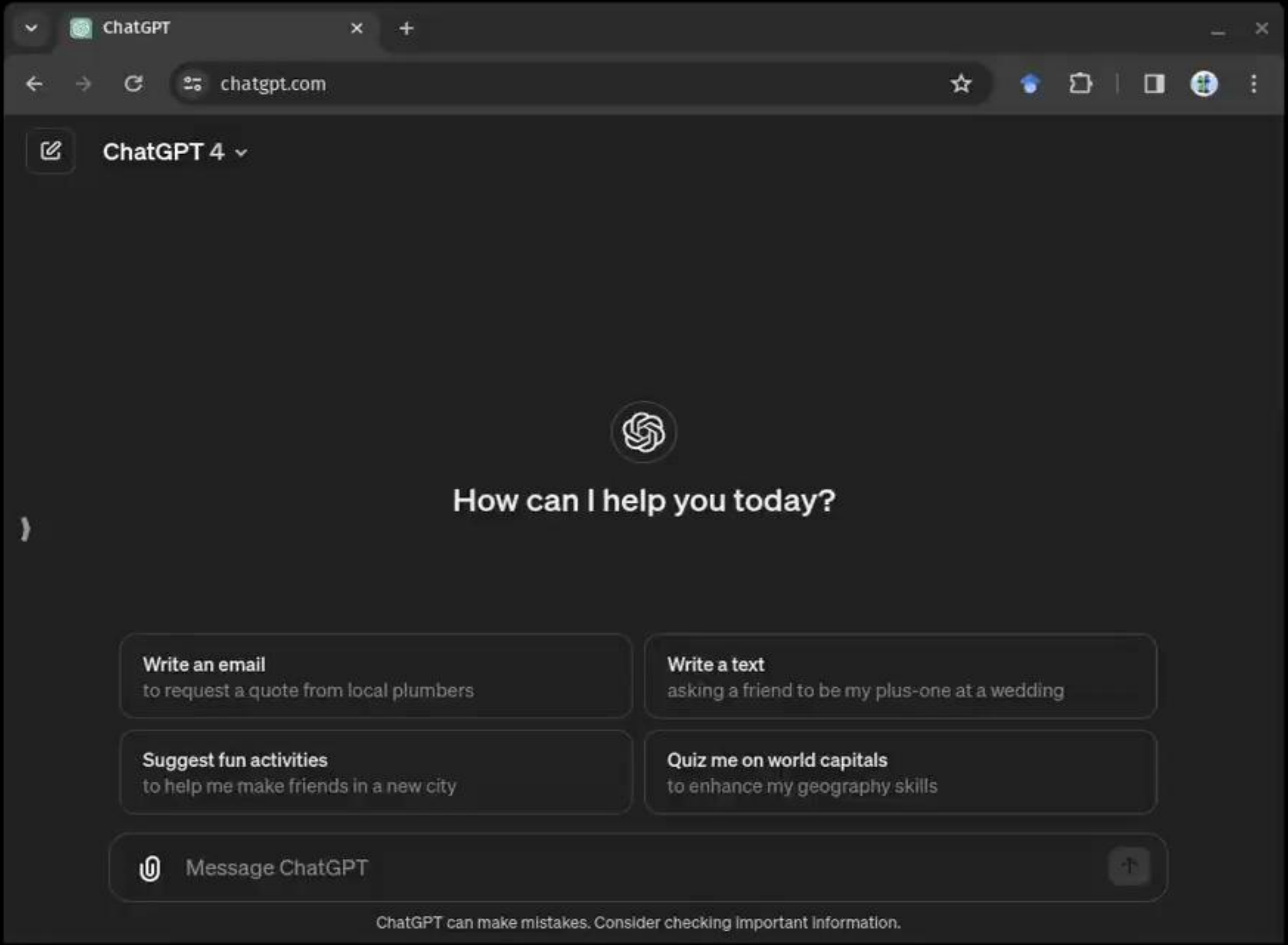
ChatGPT can make mistakes. Consider checking important information.

Brandfonbrener, D., Raja, S., Prasad, T., Loughridge, C., Yang, J., Henniger, S., Byrd, W.E., Zinkov, R. and Amin, N., 2024. Verified Multi-Step Synthesis using Large Language Models and Monte Carlo Tree Search. arXiv preprint arXiv:2402.08147

Specify

From Code To Specs

Importance of legacy codes



ChatGPT
chatgpt.com
ChatGPT 4

How can I help you today?

Write an email
to request a quote from local plumbers
Write a text
asking a friend to be my plus-one at a wedding
Suggest fun activities
to help me make friends in a new city
Quiz me on world capitals
to enhance my geography skills
Message ChatGPT

ChatGPT can make mistakes. Consider checking important information.

Sun, C., Sheng, Y., Padon, O. and Barrett, C., 2023. Clover: Closed-Loop Verifiable Code Generation. arXiv preprint arXiv:2310.17807.

Generate

Generate

Decidable First-order Theories

Hoare logic,
Separation logic

Automated Theorem Provers (ATP), e.g. Dafny, Frama-C, Verus

```
method maxArray(a: array<int>) returns (m: int)
    requires a != null && a.Length > 0
    ensures forall k :: 0 <= k < a.Length ==> a[k] <= m
    ensures exists k :: 0 <= k < a.Length && a[k] == m
{
    m := a[0];
    var i := 1;
    while (i < a.Length)
        invariant 1 <= i <= a.Length
        invariant forall j :: 0 <= j < i ==> a[j] <= m
        invariant exists j :: 0 <= j < i && a[j] == m
    {
        m := if m > a[i] then m else a[i];
        i := i + 1;
    }
}
```

Sun, C., Sheng, Y., Padon, O. and Barrett, C., 2023. Clover: Closed-Loop Verifiable Code Generation. arXiv preprint arXiv:2310.17807.

Generate

Expressive Higher-Order Theories

Dependent type theory

Interactive Theorem Provers (ITP), e.g. Coq, Lean, Agda

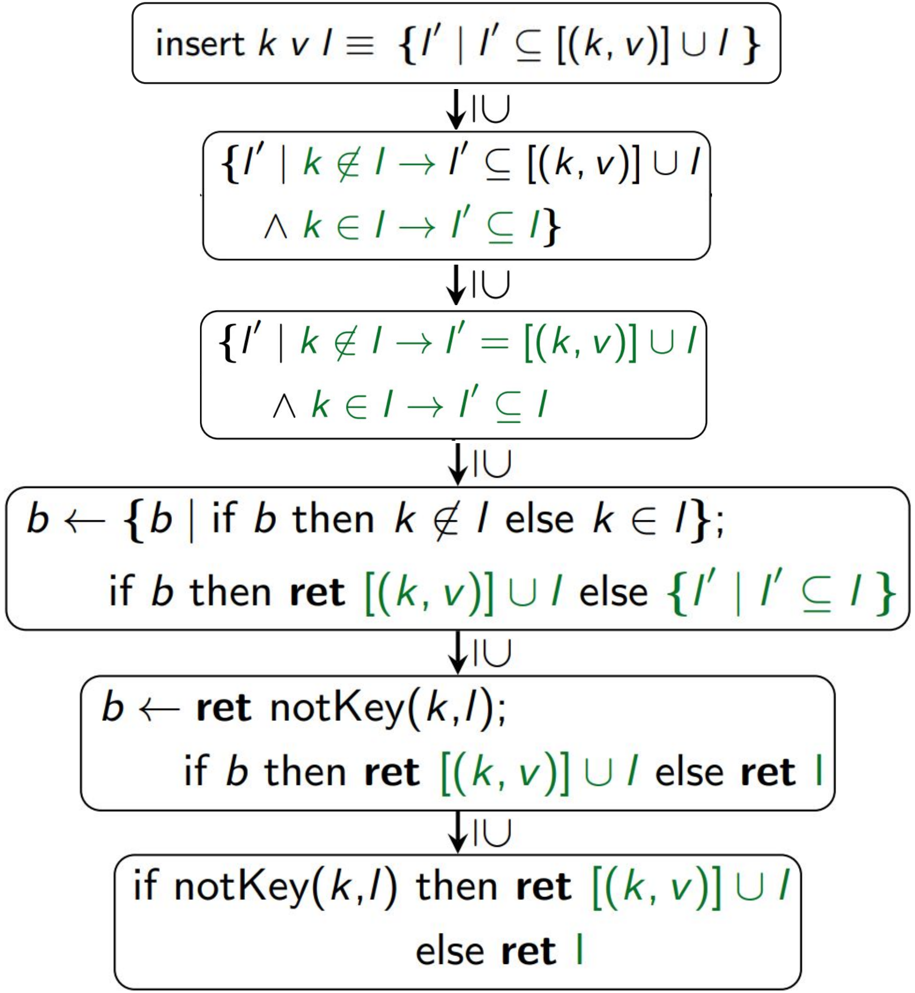
```
-- Import necessary libraries for proving
import data.nat.basic
import tactic

-- Using the previously defined factorial function
def factorial : ℕ → ℕ
| 0     := 1
| (n+1) := (n + 1) * factorial n

-- Proof that factorial is always positive
theorem factorial_pos : ∀ n : ℕ, factorial n > 0 :=
begin
  -- Apply mathematical induction on n
  apply nat.rec,
  -- Base case: show that factorial 0 is 1, which is positive
  { show factorial 0 > 0,
    rw factorial,  -- factorial 0 = 1 by definition
    exact nat.zero_lt_one, },
  -- Inductive step: assume factorial n is positive,
  -- prove factorial (n+1) is positive
  { intros n h,
    show factorial (n + 1) > 0,
    rw factorial,  -- expand factorial (n + 1)
    apply nat.mul_pos,  -- product of two positive numbers is positive
    show n + 1 > 0,
    exact nat.succ_pos n,
    exact h, }
end
```

Generate

Refinement-based Synthesis

Functionality without sacrificing *performance*

e.g. fiat-crypto library: Estimated that over 95% of HTTPS connections by browsers run the generated algorithm.



$$\text{insert } k\ v\ l \equiv \{l' \mid l' \subseteq [(k,v)] \cup l\}$$

$$\downarrow|\cup$$

$$\{l' \mid k \notin l \rightarrow l' \subseteq [(k,v)] \cup l \;\wedge\; k \in l \rightarrow l' \subseteq l\}$$

$$\downarrow|\cup$$

$$\{l' \mid k \notin l \rightarrow l' = [(k,v)] \cup l \;\wedge\; k \in l \rightarrow l' \subseteq l$$

$$\downarrow|\cup$$

$$b \leftarrow \{b \mid \text{if } b \text{ then } k \notin l \text{ else } k \in l\};\quad \text{if } b \text{ then } \textbf{ret}\ [(k,v)] \cup l \text{ else } \{l' \mid l' \subseteq l\}$$

$$\downarrow|\cup$$

$$b \leftarrow \textbf{ret}\ \text{notKey}(k,l);\quad \text{if } b \text{ then } \textbf{ret}\ [(k,v)] \cup l \text{ else } \textbf{ret}\ l$$

$$\downarrow|\cup$$

$$\text{if notKey}(k,l) \text{ then } \textbf{ret}\ [(k,v)] \cup l \text{ else } \textbf{ret}\ l$$

Delaware, B., Pit-Claudel, C., Gross, J. and Chlipala, A., 2015. Fiat: Deductive synthesis of abstract data types in a proof assistant. Acm Sigplan Notices, 50(1), pp.689-700.

Generate

Correct By Construction

Tactic-driven

Code-carrying proof

Extracting implementations

Insight 1

Construct tactic by tactic,
not sample token by token

```
hone method insert. {
      StartMethod.
      setoid_rewrite refine_ReplaceUsedKeyAdd.
      setoid_rewrite refine_SubEnsembleInsert.
      autorewrite with monad laws.
      setoid_rewrite refine_pick_KeyToBeReplaced min.
      setoid_rewrite refine_If_Then_Else_Bind.
      autorewrite with monad laws.
      setoid_rewrite refine_If_Opt_Then_Else_Bind.
      autorewrite with monad laws.
      setoid_rewrite refine_pick_CacheADTwLogIndex AbsR.
      setoid_rewrite refine_pick_KVEnsembleInsertRemove
                            with (1 := EquivKeys H).
      setoid_rewrite refine_pick_KVEnsembleInsert
                            with (1 := EquivKeys H).
      autorewrite with monad laws; simpl.
      finish honing. }
```

Delaware, B., Pit-Claudel, C., Gross, J. and Chlipala, A., 2015. Fiat: Deductive synthesis of abstract data types in a proof assistant. Acm Sigplan Notices, 50(1), pp.689-700.

Generate

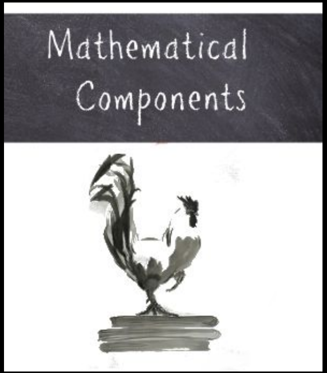
Correct By Construction

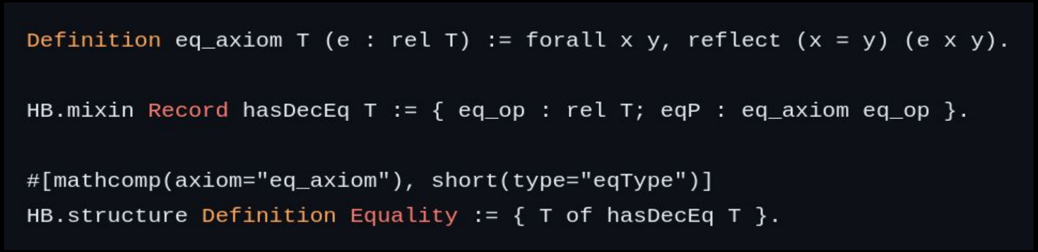
Insight 2

For AI models to generalize well, we need **generic tactics**

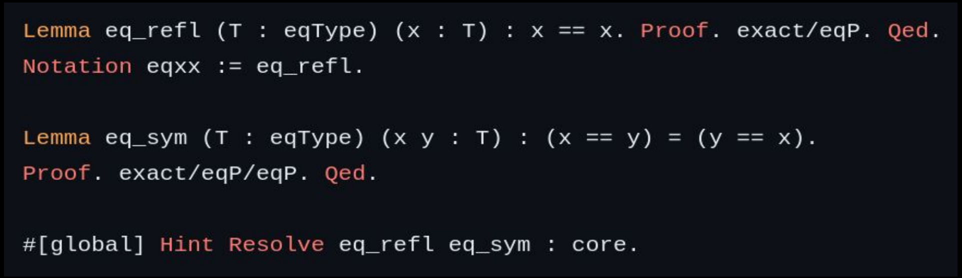
Develop tactics for categories – more precisely, algebraic theories in the form of **typeclasses**

e.g. proof of Four Color Thm and Odd Order Thm with mathematical components





```
Definition eq_axiom T (e : rel T) := forall x y, reflect (x = y) (e x y).

HB.mixin Record hasDecEq T := { eq_op : rel T; eqP : eq_axiom eq_op }.

#[mathcomp(axiom="eq_axiom"), short(type="eqType")]
HB.structure Definition Equality := { T of hasDecEq T }.
```



```
Lemma eq_refl (T : eqType) (x : T) : x == x. Proof. exact/eqP. Qed.
Notation eqxx := eq_refl.

Lemma eq_sym (T : eqType) (x y : T) : (x == y) = (y == x).
Proof. exact/eqP/eqP. Qed.

#[global] Hint Resolve eq_refl eq_sym : core.
```

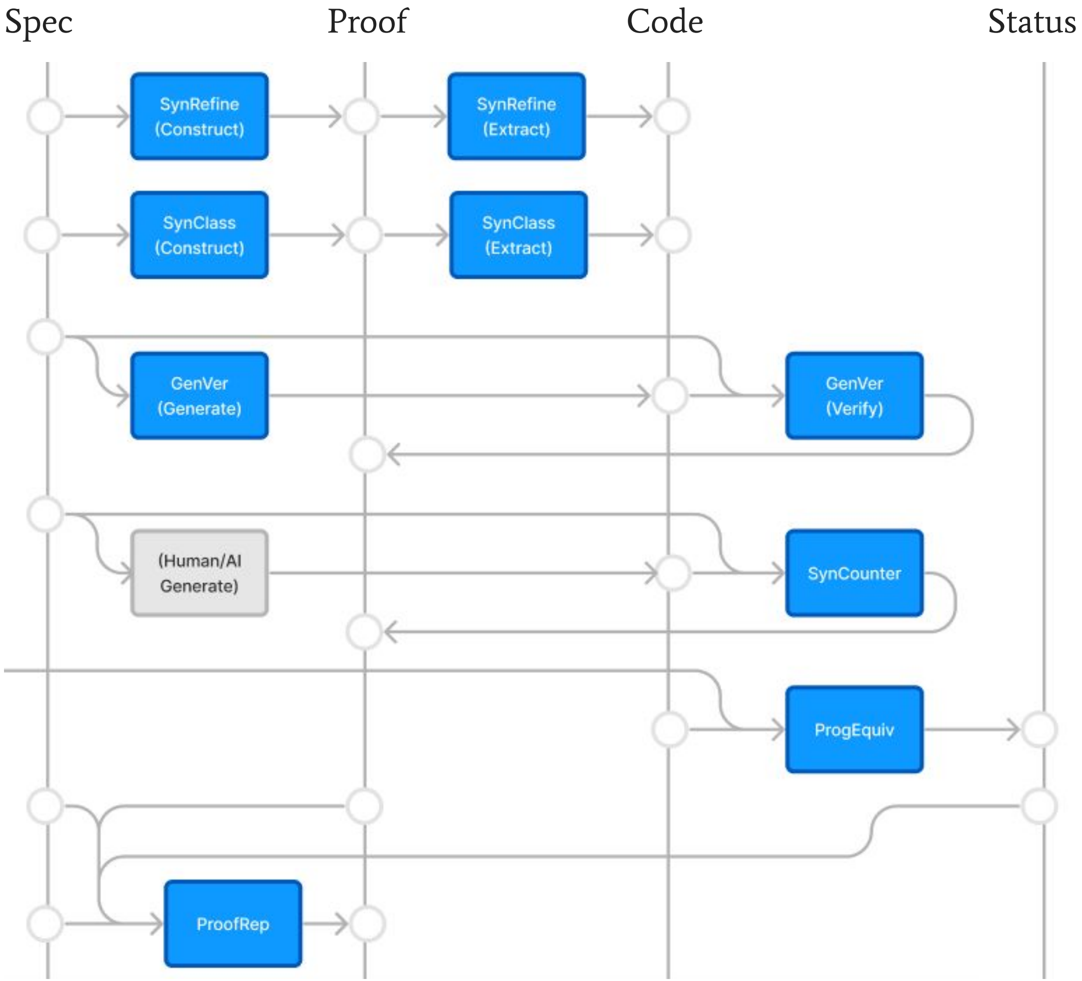
Generate

Program Equivalence

- Check if two implementations have the same behavior (ProgEquiv)
- Otherwise give a counterexample (ProgCounter)

Proof Repair

- Fix proofs when libraries, specs, implementations change (ProofRep)



Summary

Challenge	Scaling formal verification with AI assistance
Model	Feeding AI models with legacy code and doc
Interact	Human users + proof assistant + AI assistant
Specify	From natural language to formal structures
Generate	Correct by construction, not by generation

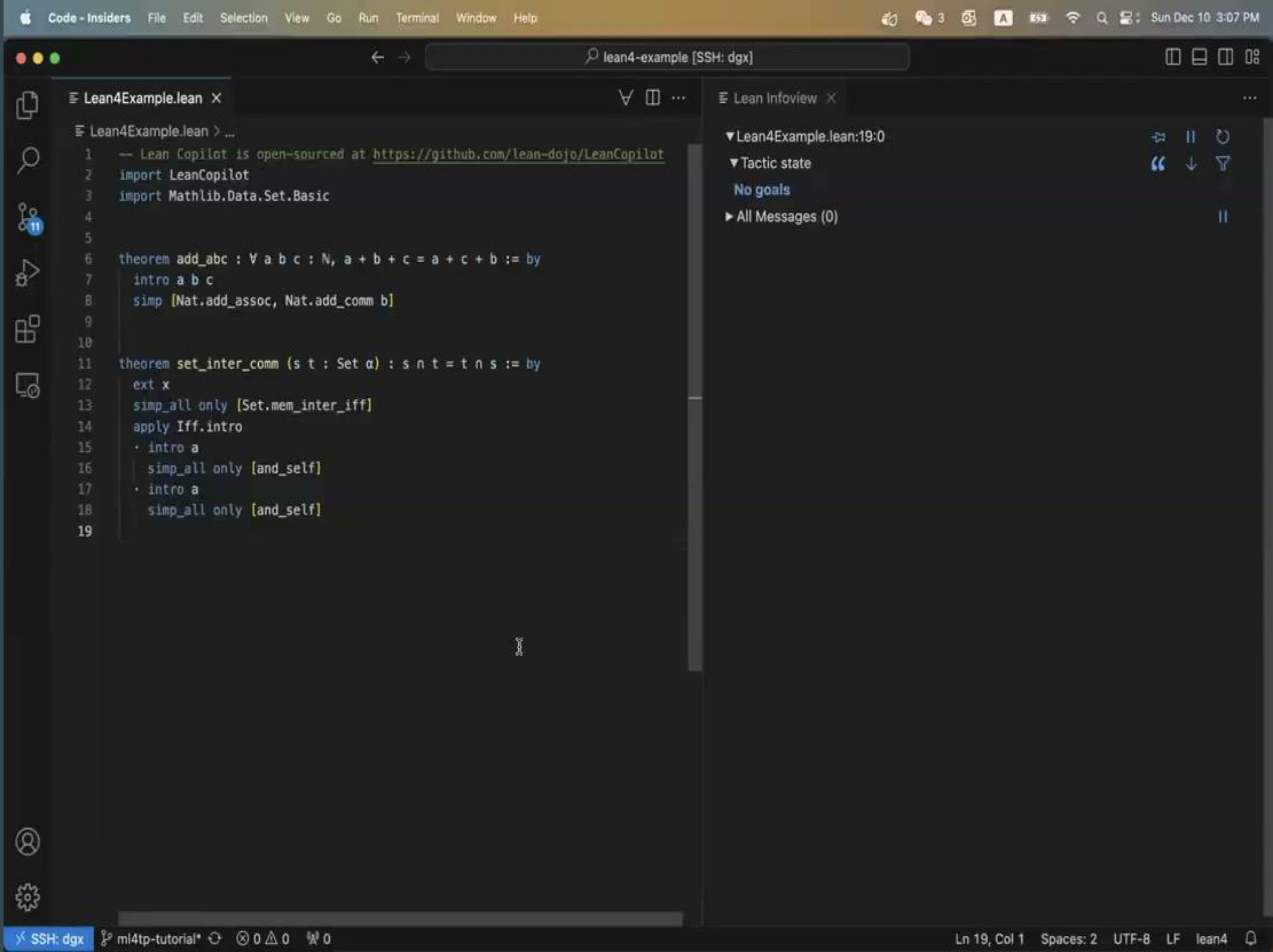
Thank you!

Lean Copilot

suggest_tactics

search_proof

select_premises



```
-- Lean Copilot is open-sourced at https://github.com/lean-dojo/LeanCopilot
import LeanCopilot
import Mathlib.Data.Set.Basic


theorem add_abc : ∀ a b c : ℕ, a + b + c = a + c + b := by
  intro a b c
  simp [Nat.add_assoc, Nat.add_comm b]


theorem set_inter_comm (s t : Set α) : s ∩ t = t ∩ s := by
  ext x
  simp_all only [Set.mem_inter_iff]
  apply Iff.intro
  · intro a
    simp_all only [and_self]
  · intro a
    simp_all only [and_self]

```

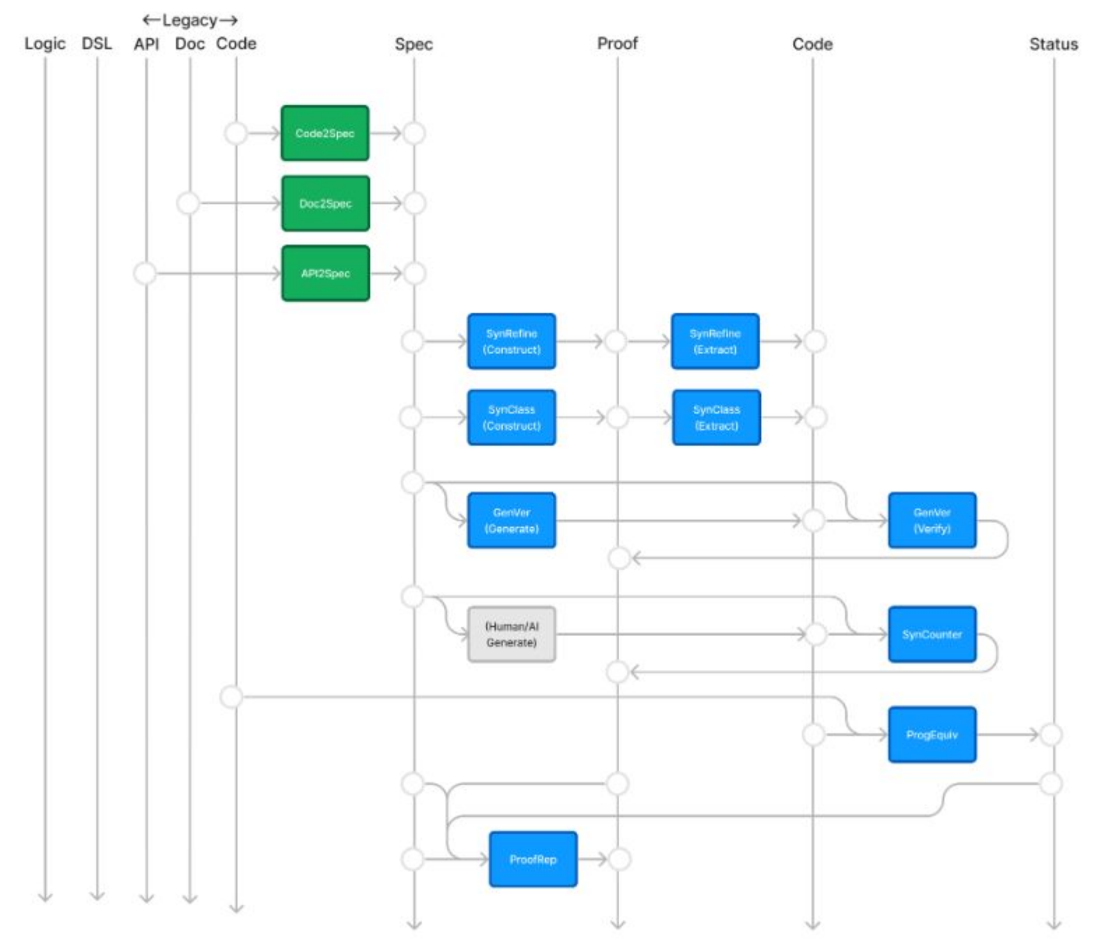
▼ Lean4Example.lean:19:0

▼ Tactic state

No goals

▶ All Messages (0)

Dependencies


←Legacy→
Logic
DSL
API
Doc
Code
Spec
Proof
Code
Status
Code2Spec
Doc2Spec
API2Spec
SynRefine (Construct)
SynRefine (Extract)
SynClass (Construct)
SynClass (Extract)
GenVer (Generate)
GenVer (Verify)
(Human/AI Generate)
SynCounter
ProgEquiv
ProofRep