Automated Theory Substitution

Toward Proof-Driven Software Development

HCSS May 6, 2024
John Scebold, Jared Ziegler

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

TOC

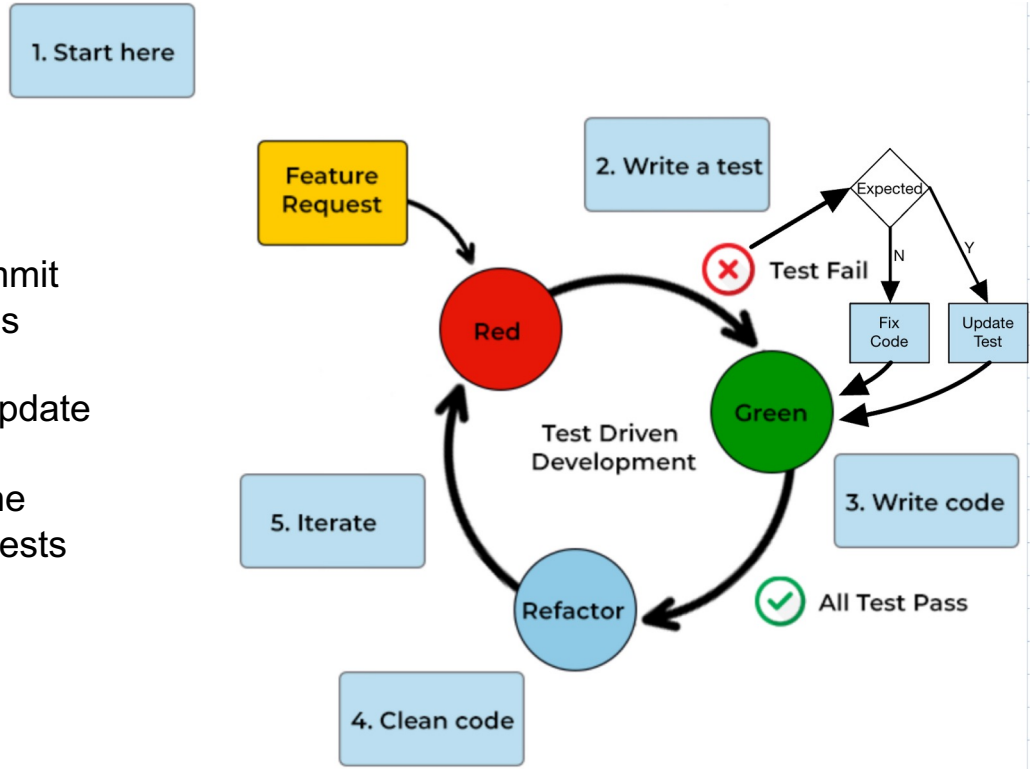
AMETHYST Introduction

- Formal verification of real world software has shown increasing success
  - seL4
  - CompCert
- But still no widespread adoption
  - Expert knowledge required to write and maintain formal proofs
  - Experience with interactive theorem provers (ITP)
- AutoMatEd THeorY SubsTitution (AMETHYST)
  - Artificially intelligent proof assistant for the Isabelle ITP
  - User interaction designed for non-experts
    - Software engineers
    - Integrated into VSCode
  - Move maintenance from the expert into the CI/CD cycle

Empower the novice, Unburden the expert

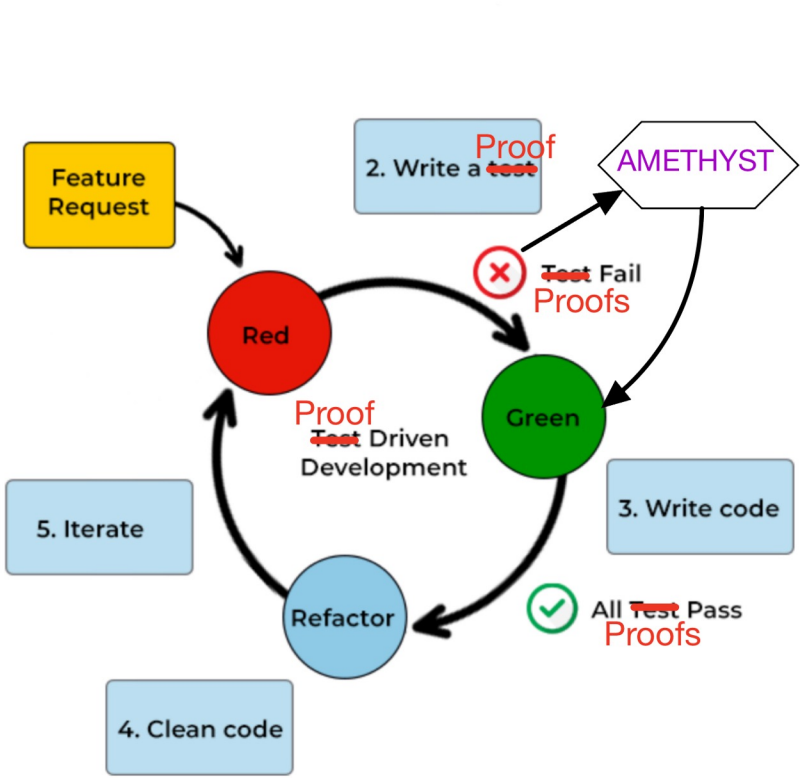
Test-Driven Development

- Tests may be
  - Happy path scenarios
  - Edge case scenarios
  - Regression/unit
- Often run automatically before commit
- Failed tests can go in two directions
  - New bug in code - fix it
  - Expected change in code - update test
- Pro: Tests are written in code so the developer can maintain code and tests
- Con: Can never explicitly cover all cases



Proof-Driven Development

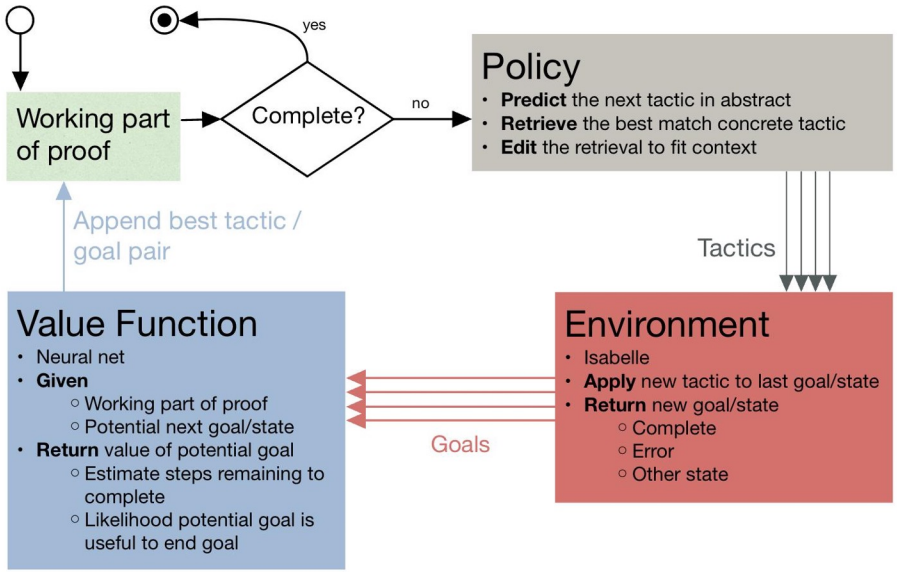
- Proofs implicitly cover all cases
- Should run automatically before commit
- It is easier to break a test than to break a proof
- Fixing a proof is less straightforward - may be a combination of code and proof edits
- Code is easy, proofs are hard
- Amethyst can help with the proof



AMETHYST AI

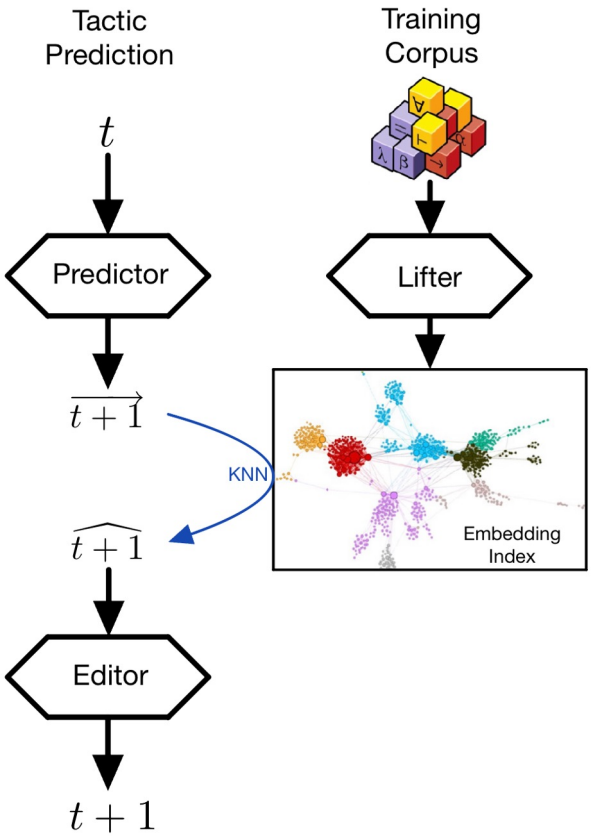
Amethyst AI

- Two machine learning models
  - **Policy** and **Value Function**
  - Pretrained offline on seL4 and Isabelle proofs
- Isabelle/HOL proof assistant
- Coordinated by a tree search agent
  - Finishes a partial proof
  - Updates models in online learning fashion



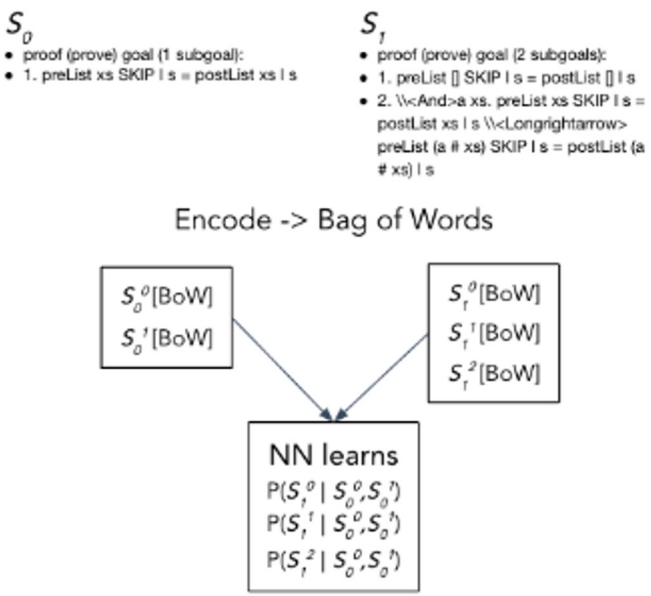
Policy

- Lifter
  - Pretrained to embed lexically similar tactics near each other
  - Embeddings are indexed for NN lookup
- Predictor
  - Given text of a tactic, predict embedding of next tactic
  - Lookup NN in index
- Editor
  - GPT edits the retrieval to fit context of current proof
  - This is remedial RAG



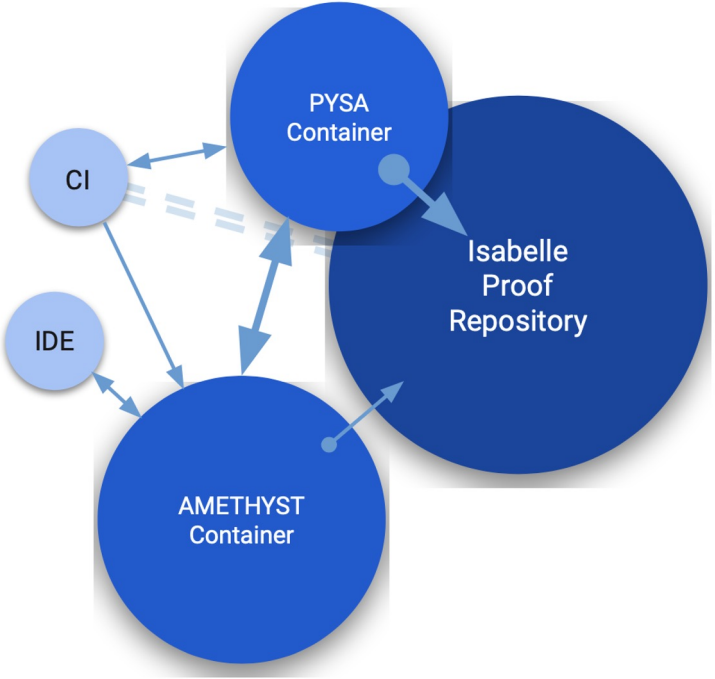
Value Function

- Proof: series of interleaved states and tactics
  - A state s is composed of one or more sub goals
  - Tactic t_n operates on s_n to produce s_{n+1}
- Measure: given states 0-n, how useful is s_{n+1} to completing the proof?
- Train a neural network to predict if the sub goals of s_{n+1} truly follow sub goals of s_n
  - Positive example - select s_n s_{n+1} from contiguous proofs
  - Negative example - select s_n from one proof, select s_{n+1} from random other proof



PYSA

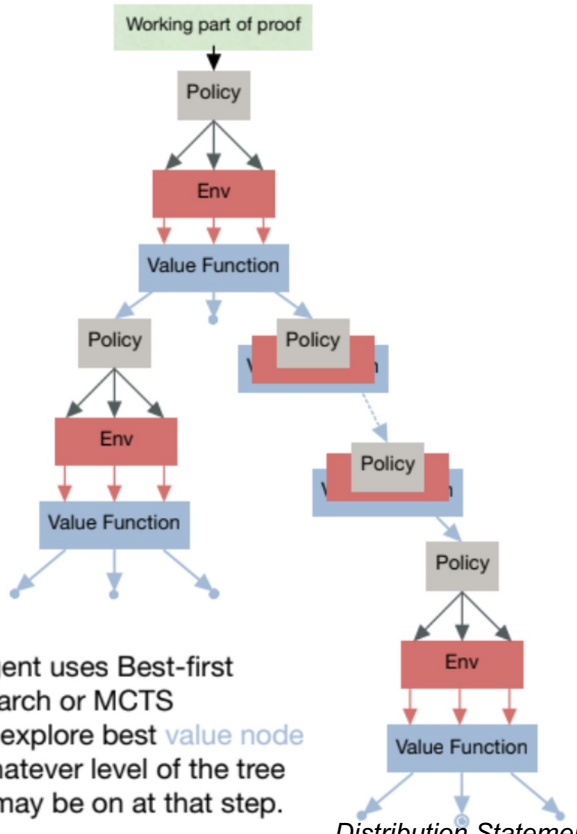
- Provide Python API for Isabelle actions
  - Inspired by Portal to Isabelle (PISA)
  - Parse and navigate theory files
  - Build heap images
  - Gather curriculum data from git history
- Get proof state from Isabelle
  - Lesson plans for AMETHYST training
  - Feedback for AMETHYST evaluation
  - Proof code testing for CI



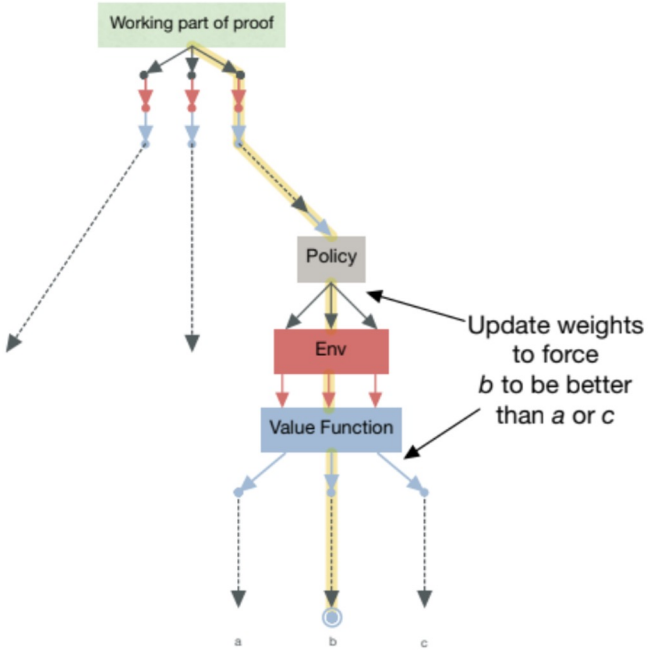
Tree Search Agent

Agent "owns" all modules

- Receives partial working proof
- Policy predicts possible next tactics
- Execute tactics via PYSA
- Value function scores valid PYSA returns
- Iterate until complete proof or max depth
- Take user feedback and back propagate up the tree from best leaf updating models.

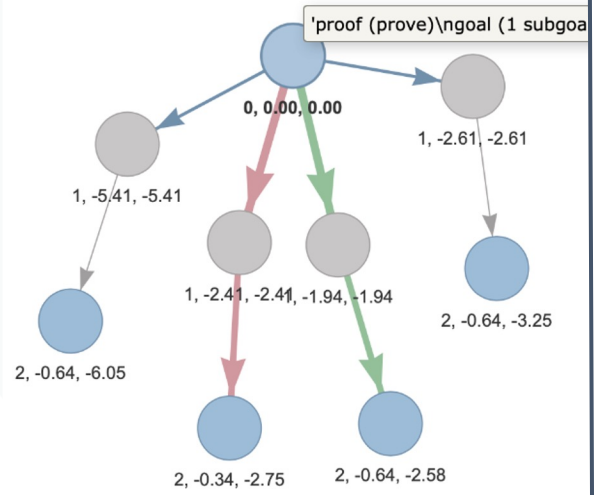


Agent uses Best-first search or MCTS to explore best value node whatever level of the tree it may be on at that step.

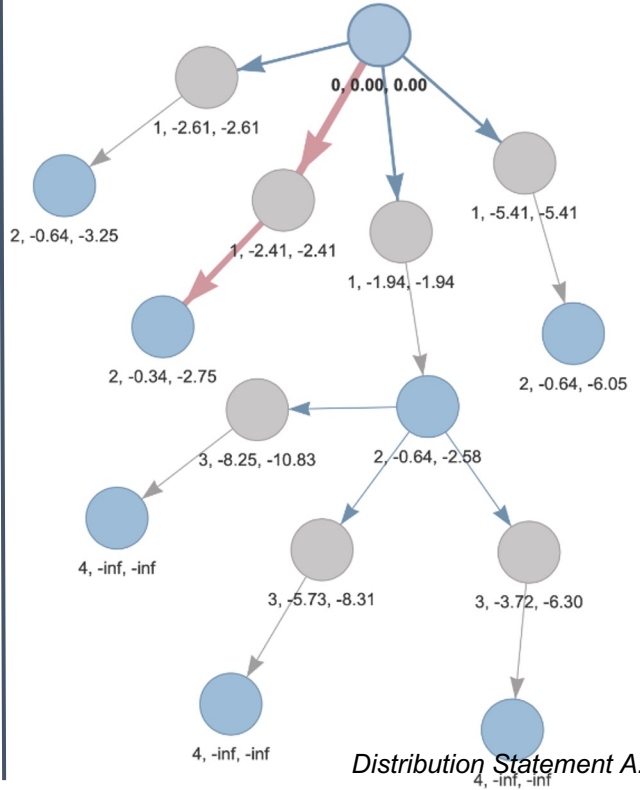


Tree Search Teacher-Forced Learning

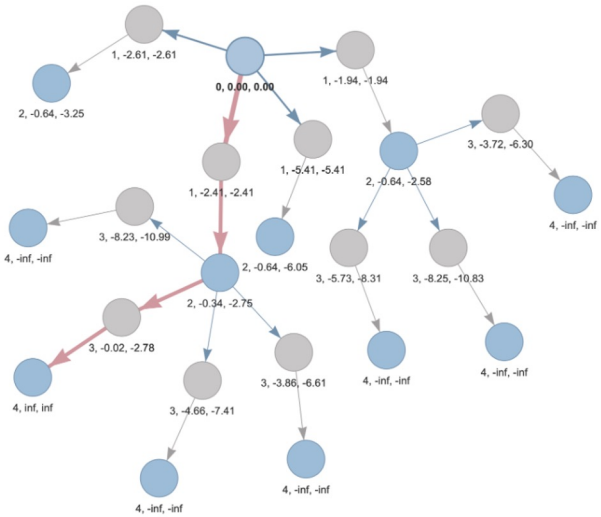
Step 1



Step 2



Step 3



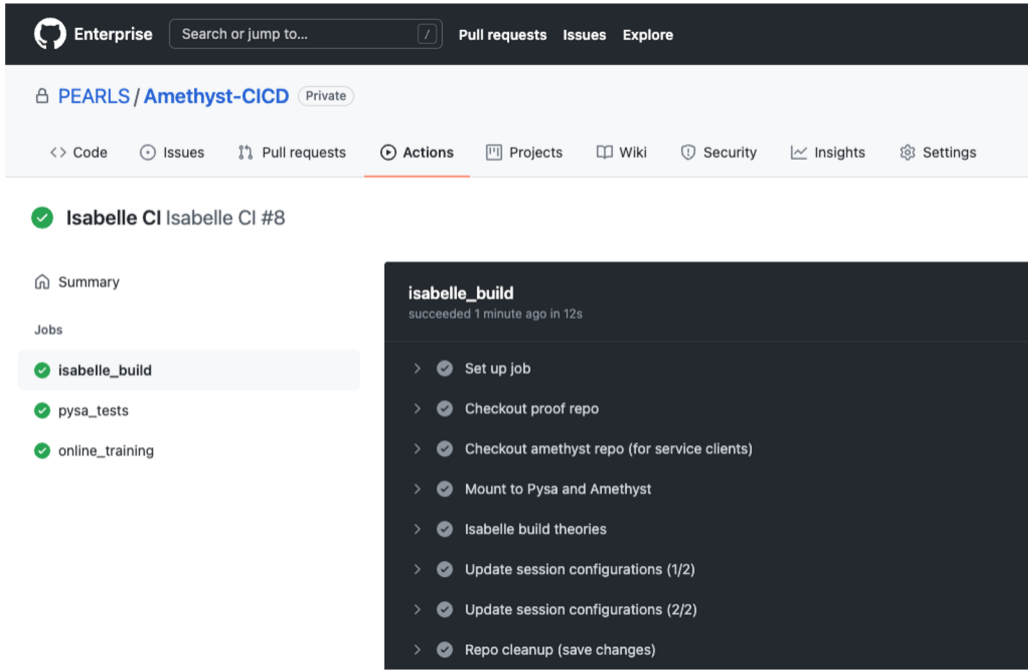
- Blue node - proof state
- Grey node - generated tactic
- Green edge - best path
- Red Edge - teacher's correct path

- Search terminates when PYSA returns zero subgoals.
- Value function assigns score of infinity.
- Teacher path is highest scoring

AMETHYST UX

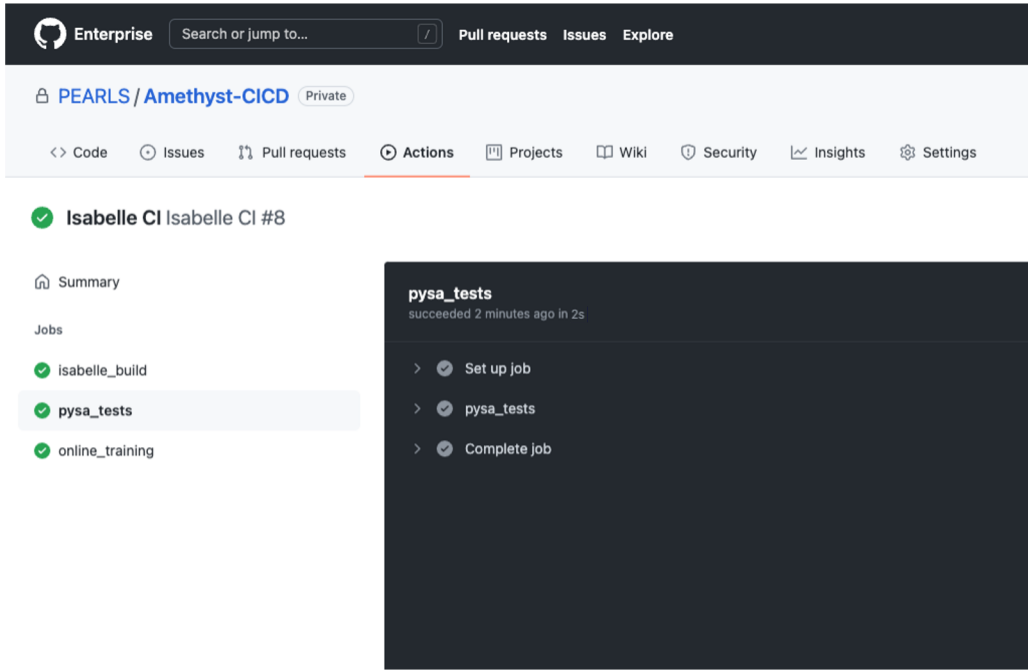
CI with GitHub Actions

- Builds modified theories, reports meaningful deltas



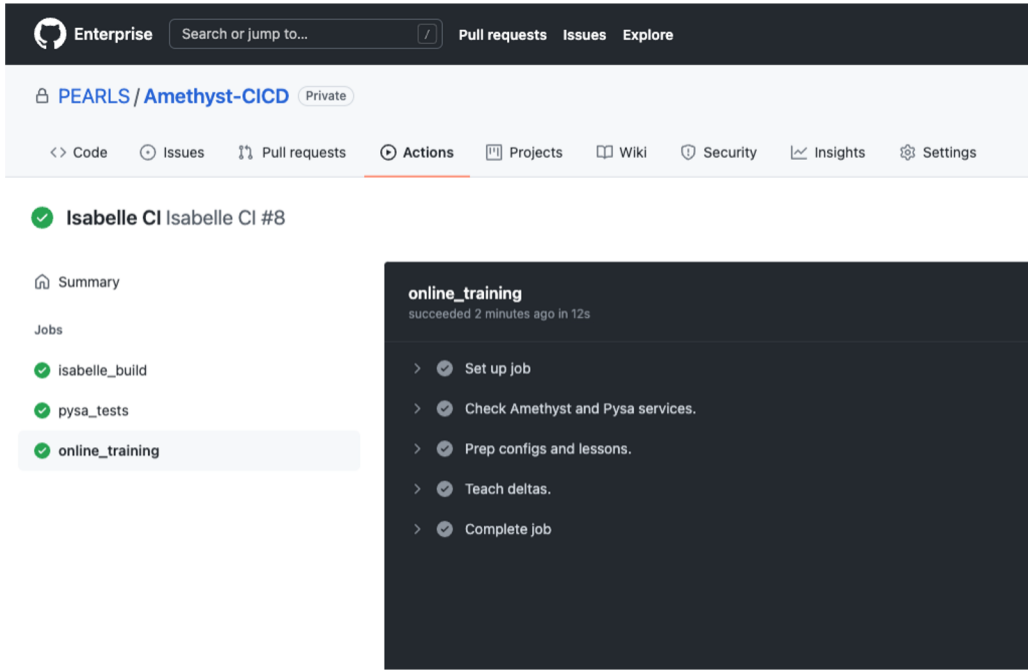
CI with GitHub Actions

- Builds modified theories, reports meaningful deltas
- Tests all proofs, checking PYSA capabilities.



CI with GitHub Actions

- Builds modified theories, reports meaningful deltas
- Tests all proofs, checking Pysa capabilities.
- Online training: teaches model new proofs automatically



VSCode Extension

- Indexer
  - Search tool based on KNN in an embedding space
  - Helps find similar lemmas or tactics
- Interactive Proof Search
  - Visualize AI driven proof search
  - See which branches are promising
    - Proof states ranked by a score
  - Direct agent to explore certain paths

Indexer

Equivalence_Relations.thy

```
le_lang_equiv: "(\<simeq>) \<le> (\<simeq>\<^sub>L)"
begin

lemma eq_implies_ltl_equiv:
  "\<phi> \<simeq> \<psi> \<Longrightarrow> w \<Turnstile>\<^sub>n \<phi> = w \<Turnstile>\<^sub>n \<psi>"
  using le_lang_equiv ltl_lang_equiv_def by blast

lemma const_implies_eq:
  "\<phi> \<simeq>\<^sub>C \<psi> \<Longrightarrow> \<phi> \<simeq> \<psi>"
  using ge_const_equiv by blast

lemma eq_implies_lang: "\<phi> \<simeq> \<psi> \<Longrightarrow> \<phi> \<simeq>\<^sub>L \<psi>"
using le_lang_equiv
by blast

lemma eq_refl[simp]:
  "\<phi> \<simeq> \<phi>"
  by (meson eq_equivp equivp_reflp)

lemma eq_sym[sym]:
  "\<phi> \<simeq> \<psi> \<Longrightarrow
  by (meson eq_equivp equivp_symp)
```

- Pysa Indexer
- Pysa Tree Visualizer
- Go to Definition
- Peek
- Change All Occurrences
- Refactor...
- Cut
- Copy
- Paste
- Command Palette...

Indexer

[Extension Development Host] Pysa Indexer — LTL

Equivalence_Relations.thy

```
520     le_lang_equiv: "(\<sim>) \<le> (\<sim>\<^sub>L)"
521   begin
522
523   lemma eq_implies_ltl_equiv:
524     "\<phi> \<sim> \<psi> \<Longrightarrow> w \<Turnstile>\<^sub>n \<phi> = w
525     using le_lang_equiv ltl_lang_equiv_def by blast
526
527   lemma const_implies_eq:
528     "\<phi> \<sim>\<^sub>C \<psi> \<Longrightarrow> \<phi> \<sim> \<psi>"
529     using ge_const_equiv by blast
530
531   lemma eq_implies_lang:  "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<si
532   using le_lang_equiv
533   by blast
534
535
536
537
538   lemma eq_refl[simp]:
539     "\<phi> \<sim> \<phi>"
540     by (meson eq_equivp equivp_reflp)
541
542   lemma eq_sym[sym]:
543     "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<phi>"
544     by (meson eq_equivp equivp_symp)
545
546   lemma eq_trans[trans]:
```

Pysa Indexer

Pysa Tree Visualizer

Search

lemma eq_implies_lang: "\<phi> \<sim> \<psi> \<Longrigh Search

Results

lemma eq_implies_lang:
"\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<sim>\<^sub>L \<psi>"

lemma const_implies_eq:
"\<phi> \<sim>\<^sub>C \<psi> \<Longrightarrow> \<phi> \<sim> \<psi>"

lemma af_lang_congruent:
"\<phi> \<sim>\<^sub>L \<psi> \<Longrightarrow> af \<phi> w \<sim>\<^sub>L af \<psi> w"

lemma ltl_prop_equiv_implies_ltl_lang_equiv:
"\<phi> \<sim>\<^sub>P \<psi> \<Longrightarrow> \<phi> \<sim>\<^sub>L \<psi>"

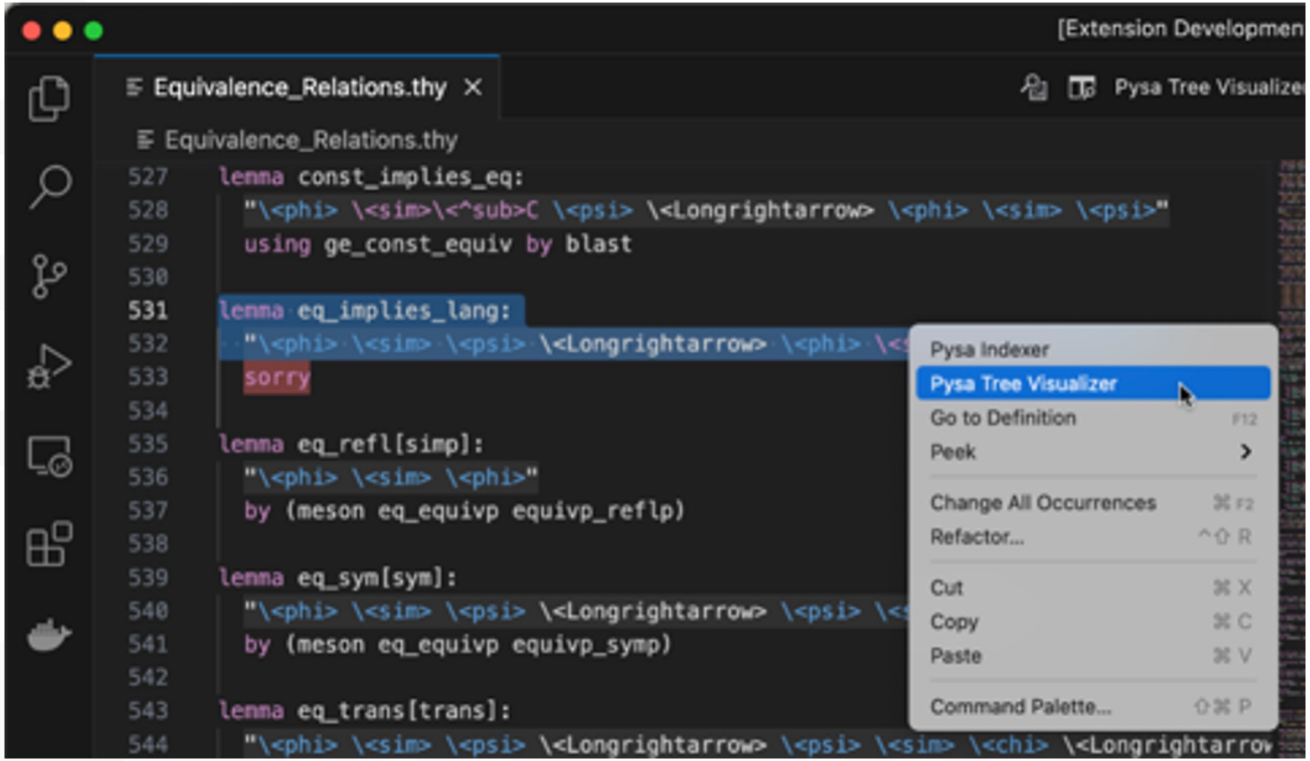
lemma af_const_congruent:
"\<phi> \<sim>\<^sub>C \<psi> \
\<phi> w \<sim>\<^sub>C af \<psi

Copied to clipboard

VSCode Extension

- Indexer
  - Search tool based on KNN in an embedding space
  - Helps find similar lemmas or tactics
- Interactive Proof Search
  - Visualize AI driven proof search
  - See which branches are promising
    - Proof states ranked by a score
  - Direct agent to explore certain paths

Proof Search Visualizer



Proof Search Visualizer

[Extension Development Host] Pysa Tree Visualizer — LTL

Equivalence_Relations.thy

```
527  lemma const_implies_eq:
528    "\<phi> \<sim>\<^sub>C \<psi> \<Longrightarrow> \<phi> \<sim> \<psi>"
529    using ge_const_equiv by blast
530
531  lemma eq_implies_lang:
532    "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<sim>\<^sub>L \<psi>"
533    sorry
534
535  lemma eq_refl[simp]:
536    "\<phi> \<sim> \<phi>"
537    by (meson eq_equivp equivp_reflp)
538
539  lemma eq_sym[sym]:
540    "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<phi>"
541    by (meson eq_equivp equivp_symp)
542
543  lemma eq_trans[trans]:
544    "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<chi> \<Longrightarrow
545    by (meson eq_equivp equivp_transp)
546
547  end
548
549  interpretation ltl_lang_equivalence: ltl_equivalence "(\<sim>\<^sub>L)"
550    using ltl_lang_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv_
551    by unfold_locales blast+
552
553  interpretation ltl_prop_equivalence: ltl_equivalence "(\<sim>\<^sub>P)"
554    using ltl_prop_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv_
555    by unfold_locales blast+
556
```

Pysa Tree Visualizer

Lemma	lemma eq_implies_lang: "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<s
Tactics	
Session	LTL
Theory	/workspace/LTL/Equivalence_Relations.thy

Start | Full Proof | Step | Trace | Refresh

Proof Search Visualizer

[Extension Development Host] Pysa Tree Visualizer — LTL

Equivalence_Relations.thy

```
527  lemma const_implies_eq:
528    "\<phi> \<sim>\<^sub>C \<psi> \<Longrightarrow> \<phi> \<sim> \<psi>"
529    using ge_const_equiv by blast
530
531  lemma eq_implies_lang:
532    "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<sim>\<^sub>L \<psi>"
533    sorry
534
535  lemma eq_refl[simp]:
536    "\<phi> \<sim> \<phi>"
537    by (meson eq_equivp equivp_reflp)
538
539  lemma eq_sym[sym]:
540    "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<phi>"
541    by (meson eq_equivp equivp_symp)
542
543  lemma eq_trans[trans]:
544    "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<chi> \<Longrightarrow
545    by (meson eq_equivp equivp_transp)
546
547  end
548
549  interpretation ltl_lang_equivalence: ltl_equivalence "(\<sim>\<^sub>L)"
550    using ltl_lang_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv_
551    by unfold_locales blast+
552
553  interpretation ltl_prop_equivalence: ltl_equivalence "(\<sim>\<^sub>P)"
554    using ltl_prop_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv_
555    by unfold_locales blast+
556
```

Pysa Tree Visualizer

Lemma: lemma eq_implies_lang: "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<s

Tactics:

Session: LTL

Theory: /workspace/LTL/Equivalence_Relations.thy

Start | Full Proof | Step | Trace | Refresh

```
proof (prove)
goal (1 subgoal):
 1. ϕ ~ ψ ⟹ ϕ ~⇩L ψ
```

Proof Search Visualizer

[Extension Development Host] Pysa Tree Visualizer — LTL

Equivalence_Relations.thy

```
527  lemma const_implies_eq:
528    "\<phi> \<sim>\<^sub>C \<psi> \<Longrightarrow> \<phi> \<sim> \<psi>"
529    using ge_const_equiv by blast
530
531  lemma eq_implies_lang:
532    "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<sim>\<^sub>L \<psi>"
533    sorry
534
535  lemma eq_refl[simp]:
536    "\<phi> \<sim> \<phi>"
537    by (meson eq_equivp equivp_reflp)
538
539  lemma eq_sym[sym]:
540    "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<phi>"
541    by (meson eq_equivp equivp_symp)
542
543  lemma eq_trans[trans]:
544    "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<chi> \<Longrightarrow
545    by (meson eq_equivp equivp_transp)
546
547  end
548
549  interpretation ltl_lang_equivalence: ltl_equivalence "(\<sim>\<^sub>L)"
550    using ltl_lang_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv_
551    by unfold_locales blast+
552
553  interpretation ltl_prop_equivalence: ltl_equivalence "(\<sim>\<^sub>P)"
554    using ltl_prop_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv_
555    by unfold_locales blast+
556
```

Pysa Tree Visualizer

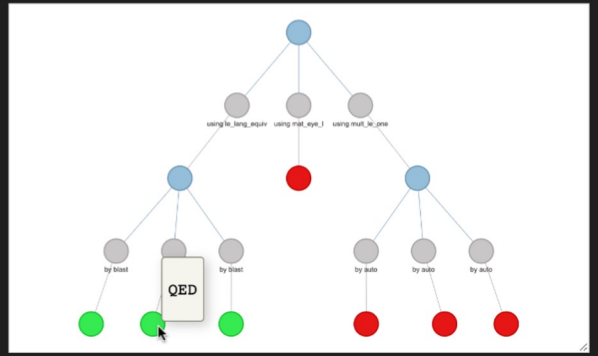
Lemma: lemma eq_implies_lang: "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<s

Tactics:

Session: LTL

Theory: /workspace/LTL/Equivalence_Relations.thy

Start | Full Proof | Step | Trace | Refresh



Proof Search Visualizer

[Extension Development Host] Pysa Tree Visualizer — LTL

Equivalence_Relations.thy

```
527  lemma const_implies_eq:
528    "\<phi> \<sim>\<^sub>C \<psi> \<Longrightarrow> \<phi> \<sim> \<psi>"
529    using ge_const_equiv by blast
530
531  lemma eq_implies_lang:
532    "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<sim>\<^sub>L \<psi>"
533    sorry
534
535  lemma eq_refl[simp]:
536    "\<phi> \<sim> \<phi>"
537    by (meson eq_equivp equivp_reflp)
538
539  lemma eq_sym[sym]:
540    "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<phi>"
541    by (meson eq_equivp equivp_symp)
542
543  lemma eq_trans[trans]:
544    "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<chi> \<Longrightarrow
545    by (meson eq_equivp equivp_transp)
546
547  end
548
549  interpretation ltl_lang_equivalence: ltl_equivalence "(\<sim>\<^sub>L)"
550    using ltl_lang_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv_
551    by unfold_locales blast+
552
553  interpretation ltl_prop_equivalence: ltl_equivalence "(\<sim>\<^sub>P)"
554    using ltl_prop_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv_
555    by unfold_locales blast+
556
557  interpretation ltl_const_equivalence: ltl_equivalence "(\<sim>\<^sub>C)"
558    using ltl_const_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv
559    by unfold_locales blast+
```

Pysa Tree Visualizer

Lemma: lemma eq_implies_lang: "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<

Tactics:

Session: LTL

Theory: /workspace/LTL/Equivalence_Relations.thy

Start | Full Proof | Step | Trace | Refresh

Trace copied to clipboard

Proof Search Visualizer

[Extension Development Host] Equivalence_Relations.thy — LTL

Equivalence_Relations.thy

```
527  lemma const_implies_eq:
528    "\<phi> \<sim>\<^sub>C \<psi> \<Longrightarrow> \<phi> \<sim> \<psi>"
529    using eq_const_equiv by blast
530   command "lemma"
531  lemma eq_implies_lang: "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<sim>\<
532  using le_lang_equiv
533  by blast
534
535
536  lemma eq_refl[simp]:
537    "\<phi> \<sim> \<phi>"
538    by (meson eq_equivp equivp_reflp)
539
540  lemma eq_sym[sym]:
541    "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<phi>"
542    by (meson eq_equivp equivp_symp)
543
544  lemma eq_trans[trans]:
545    "\<phi> \<sim> \<psi> \<Longrightarrow> \<psi> \<sim> \<chi> \<Longrightarrow
546    by (meson eq_equivp equivp_transp)
547
548  end
549
550  interpretation ltl_lang_equivalence: ltl_equivalence "(\<sim>\<^sub>L)"
551    using ltl_lang_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv_
552    by unfold_locales blast+
553
554  interpretation ltl_prop_equivalence: ltl_equivalence "(\<sim>\<^sub>P)"
555    using ltl_prop_equiv_equivp ltl_const_equiv_lt_ltl_prop_equiv ltl_prop_equiv_
556    by unfold_locales blast+
```

Pysa Tree Visualizer

Lemma: lemma eq_implies_lang: "\<phi> \<sim> \<psi> \<Longrightarrow> \<phi> \<s

Tactics:

Session: LTL

Theory: /workspace/LTL/Equivalence_Relations.thy

Start | Full Proof | Step | Trace | Refresh