Automated Sketching and Repairing of Large Mechanised Proofs

C. Tan[1,2], A. F. Donaldson[1], J. Wickerson[1], J. J. Huerta y Munive[3]
[1]Imperial College London, UK [2]Kaihong, China [3]Aalborg University, Denmark
{alastair.donaldson, j.wickerson}@imperial.ac.uk tanchengsong@kaihong.com jjhymuni@cs.aau.dk

Repository **Video demo**

Motivation & Background

- Formal verification with Interactive Theorem Proving (ITP) provides high assurance of software quality, but it is time-consuming which is a barrier to industrial adoption.
- We aim to address the automation challenges that hinder verification workflows by developing tools that streamline interactions among the human expert, the proof assistant, and automation tools.
- **Case study:** A formal model of `CXL.cache` in Isabelle/HOL (74 theory files, ≈310 k LoC), proving **Single-Writer-Multiple-Reader (SWMR)** cache coherence via an inductive invariant P.
- P is an conjunction comprised of $n = 796$ conjuncts that must be preserved by $m = 68$ rules, yielding a **proof obligation matrix** with over 54,000 proof obligations.

$$\begin{pmatrix} \ddots & & \cdots \\ & \begin{pmatrix} P\ \sigma \wedge \\ \texttt{guard}_{R_i}\ \sigma \wedge \\ \sigma \to_{R_i} \sigma' \\ \Longrightarrow \\ \phi_j\ \sigma' \end{pmatrix}_{(i,j)} & \\ \cdots & & \ddots \end{pmatrix}_{m \times n}$$

super_sketch

Generates Isar proof skeletons without generative AI, inspired by Haftmann's `sketch` tool.

How it works:

- Retrieves all proof goals after user-suggested methods
- Concurrently invokes Sledgehammer and user-supplied heuristics
- Applies goal splitting and reduction for harder subgoals before retrying

Results on the 68×796 matrix:

- One full rule lemma (>700 conjuncts) regenerated in **<30 min**
- Full matrix: ≈1 machine-day of compute
- **<100 failures** out of >50,000 goals

super_fix

Automated proof debugging and repair, emulating a proof engineer's debugging workflow.

How it works:

- 1st pass: Parses the .thy file and process the commands sequentially. Detects and fixes **timeouts**, **misaligned goals**, **failed methods** with `sorry` as a placeholder and prioritising upstream errors
- 2nd pass: Invokes Sledgehammer on each `sorry` to find alternative proofs

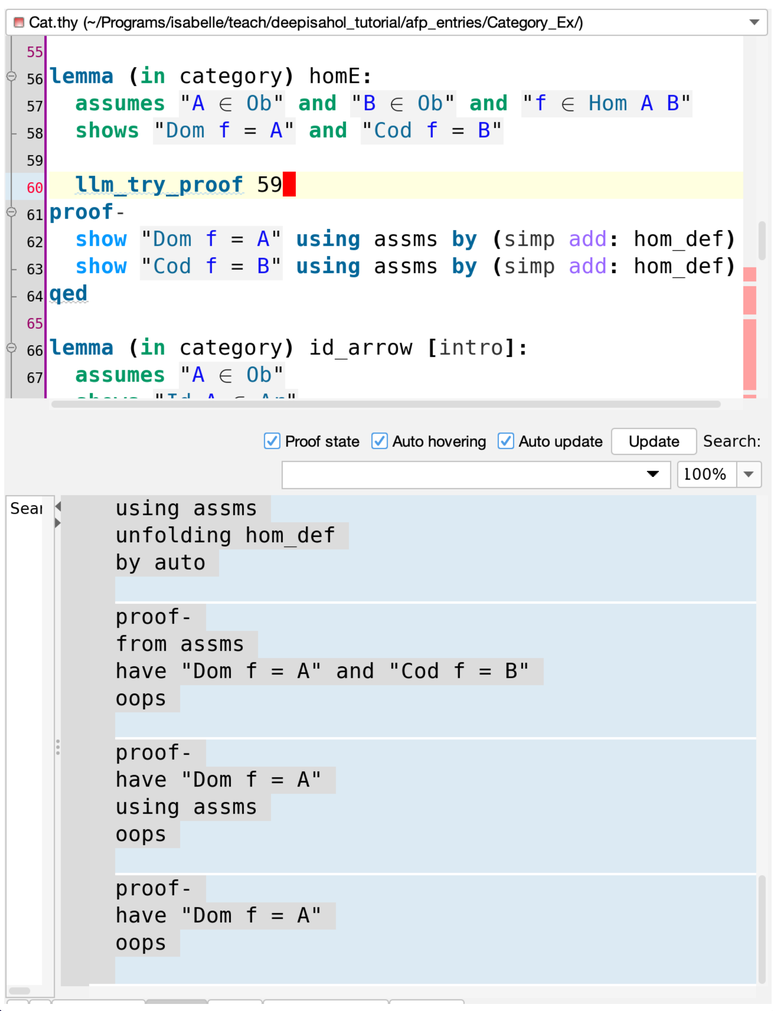
Repair results (after `super_sketch`):

- Pass 1: 18 files with errors → 9 files (1 error each)
- Pass 2: → **proof complete (0 errors)**

try_sketch example

```
(* example proof produced by try_sketch *)
lemma "(∑i≤(n::nat). 2*i-1) = n^2"
try_sketch SORRYS[simp] (induct n)
proof(induct n)
  show goal0:"(∑i::nat≤0. 2*i-1)=0^2"
    by simp
next
  show goal1:"(∑i≤Suc n. 2*i-1)=(Suc n)^2"
    if "(∑i≤n. 2*i- 1) = n^2"
    for n :: "nat"
    using that
    sorry
qed
```

llm_try_proof



Ongoing Extensions

`meta_sketch`: recursive DSL-driven generalisation of `super_sketch`.

- Users specify a hierarchy of goal splitters and a prioritised solver portfolio
- Three-stage leaf engine: *racing*, *rescuing*, *last-resort splitting*
- Isabelle/ML `Future` orchestration replaces ad-hoc Python scripting

LLM integration in Isabelle:

- Python client–server bridge to Hugging Face models, Ollama, ChatGPT, and Gemini
- `llm_recommend`: suggests the next proof step
- `llm_try_proof`: DFS-based incremental proof completion with configurable depth/breadth

Fixer functions:

- Treats tool actions as functions $f : S \to \Sigma^*$ over ITP states
- `try_sketch` reimplements `super_sketch` inside this framework
- Provides a modular basis for combining LLM actions with classical automation

super_fix example

```
theory Super_Fix_Demo
  imports Complex_Main

begin


lemma real_sqrt_le_iff': "x ≥ 0 ⟹ y ≥ 0
  ⟹ sqrt x ≤ y ⟷ x ≤ y ^ 2"
  apply blast
  done

lemma sqrt_le_itself: "1 ≤ x ⟹ sqrt x ≤ x"
  sorry

lemma sqrt_real_nat_le:"sqrt (real n) ≤ real n"
  apply simp
  done

lemma semiring_factor_left:
  fixes b:: "'a::semiring"
  shows "a * b + a * c = a * (b + c)"
  apply simp
```

```
theory Super_Fix_Demo_Fixed
  imports Complex_Main

begin


lemma real_sqrt_le_iff': "x ≥ 0 ⟹ y ≥ 0
  ⟹ sqrt x ≤ y ⟷ x ≤ y ^ 2"
  by (smt (verit, ccfv_SIG) real_le_lsqrt sqrt_le

lemma sqrt_le_itself: "1 ≤ x ⟹ sqrt x ≤ x"
  sorry

lemma sqrt_real_nat_le:"sqrt (real n) ≤ real n"
  by (smt (verit, ccfv_SIG) less_Suc0 nat_1 nat_

lemma semiring_factor_left:
  fixes b:: "'a::semiring"
  shows "a * b + a * c = a * (b + c)"
  by (smt (verit, del_insts) distrib_left)
```

Future work

Evaluation. Stress-tested on 54,128 obligations (68×796): avg. ≈0.90 s/goal. Excessive parallelism causes *timeout cascades* (16%–40% failure rate). Next: adaptive timeouts & concurrency control.

ITP Automation Survey identifying which proof engineering tasks experts most want to automate informing case studies and tooling directions.

Acknowledgements. EPSRC grant EP/R006865/1; Horizon MCSA 2022 Postdoctoral Fellowship (project 101102608); National S&T Major Project of China (Grant No. 2024ZD0803002).