Automated Program Repair & Verification

The Cost of Poor Software Quality in the US

Software engineers spend 35-50% of their time validating and debugging software.

Cost of debugging, testing, and verification accounts for 50-75% of the software development budgets.

- Devon H. O'Dell, The Debugging Mindset. ACM QUEUE, http://doi.org/10.1145/3055301.3068754

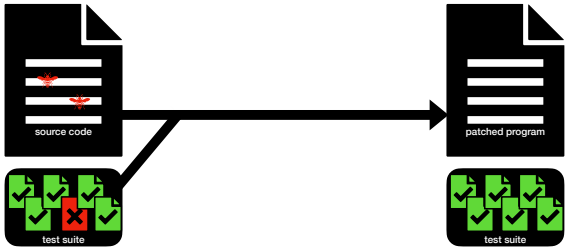
Herb Krasner, The Cost of Poor Software Quality in the US: A 2020 Report. https://www.it-cisq.org/pdf/CPSQ-2020-report.pdf

Machine Learning in Software Engineering

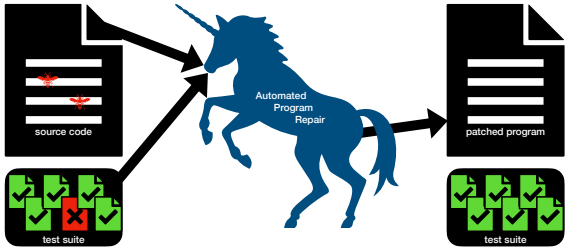


https://github.com/features/copilot

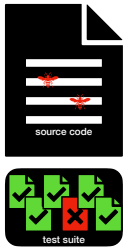
A simpler problem: Automated program repair



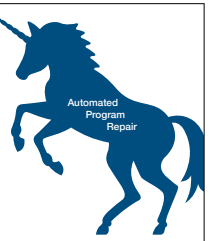
A simpler problem: Automated program repair



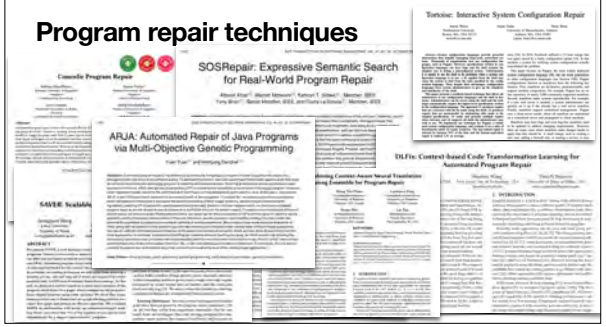
Program repair techniques



- Tweak the program
- Check if tests pass
- If not, repeat

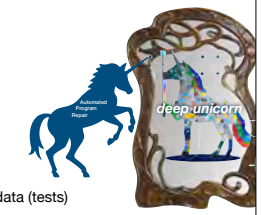


Program repair techniques



APR is a form of machine learning

- first, many techniques rely on ML to learn
  - where to edit the code
  - how to edit the code
  - how to decide which patches are good
- second, the underlying problem is learning a function (program) using training data (tests)



How well does APR work?

- Evaluated 4 techniques
  - GenProg
  - Par
  - TrpAutoRepair
  - SimFix
- Measured patch quality
- Measured what affects patch quality



Quality vs. quantity



technique	defects patched
GenProg	49 (13.7%)
Par	38 (10.6%)
SimFix	68 (19.0%)
TRPAutoRepair	44 (12.3%)
total	106 (29.7%)

When applied to real-world Java code,
APR produces patches for 10.6-19.0% of the defects

Quality vs. quantity

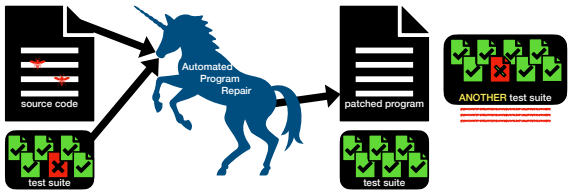


Potential problem: Overfitting

APR uses a set of tests to guide repair.
Tests are inherently partial.
No way APR can know if a patch captures intended behavioral constraints.

Quality vs. quantity



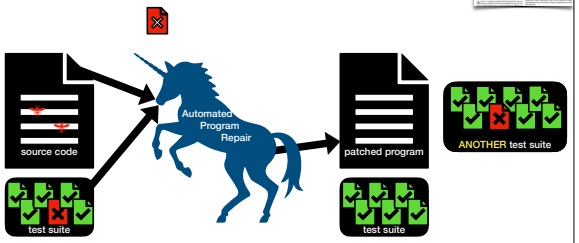


Quality vs. quantity

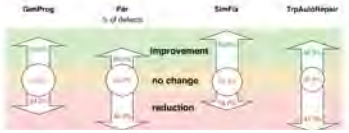
technique	patch quality minimum	mean	median	maximum	100%-quality patches
GenProg	64.8%	95.7%	98.4%	100.0%	24.3%
Par	64.8%	96.1%	98.5%	100.0%	13.8%
SimFix	65.0%	96.3%	99.9%	100.0%	46.1%
TrpAutoRepair	64.8%	96.4%	98.4%	100.0%	19.5%

Less than half (14-46%)
of the patches are correct

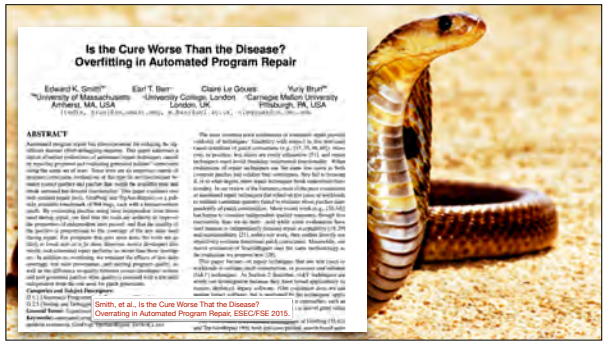
Does APR at least improve things a bit?



Does APR at least improve things a bit?



technique	change in quality due to patch minimum	mean	median	maximum
GenProg	−30.9%	−1.7%	0.0%	2.6%
Par	−30.9%	−2.8%	0.0%	1.5%
SimFix	−24.9%	0.2%	0.0%	35.0%
TrpAutoRepair	−30.9%	−2.1%	0.0%	3.8%



Takeaway: Tests are an imperfect oracle, so APR suffers, producing low-quality patches.

Can we find a domain with better oracles?

Formal verification: proving software correct





AIRBUS

android

Formal verification works!

(2011)

Csmith

AbsInt/CompCert

(The CompCert C verified compiler)

GCC

79

LLVM

202

CompCert was the only one for which Csmith could not find bugs!

"As of early 2011, the under-development version of CompCert is the only compiler we have tested for which Csmith cannot find wrong-code errors."
[Yang et al., Finding and Understanding Bugs in C Compilers, 2011]

Industrial impact of theorem proving

AIRBUS

BEDROCK Systems Inc

galois

android

aws

CERTORA

Virtually all software that ships today is unverified.

How: Interactive theorem provers

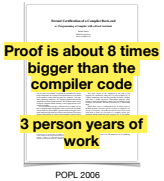
Isabelle

```
Lemma inDom :
 forall am (an: AssetName) (a: Asset),
isMappable am an a -> set_In an (dom am).
Proof.
intros am0 an0 a HMpb.
induction am0.
 - simpl in HMpb. contradiction.
 - Search "isMappable".
   apply isMappable_elim in HMpb.
   inversion HMpb. clear HMpb.
   destruct H as [Heql1 Heql2].
   + rewrite Heql1. simpl. Search "set_add".
     apply set_add_intro2. reflexivity.
   + simpl. apply set_add_intro1.
     apply IHam0. apply H.
   + intuition.
   + intuition.
Qed.
```

1. Write a formal specification
2. Write the code
3. Write a deductive proof
4. A theorem prover machine-checks the proof

Prohibitively difficult

Verified software requires a lot of time and a lot of proofs in relation to code



Proof is about 8 times bigger than the compiler code

3 person years of work

POPL 2006



11 person years of work

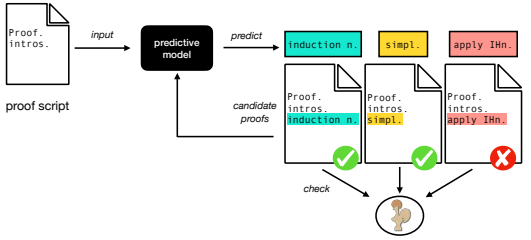
ACM Transactions on Computer Systems (2014)

Virtually all software that ships today is unverified.

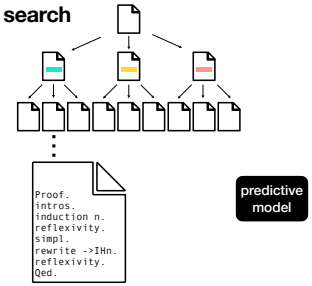
AUTOMATE!!!

1. Write a formal specification
2. Write the code
3. Write a deductive proof
4. A theorem prover machine-checks the proof

next tactic prediction



metaheuristic search

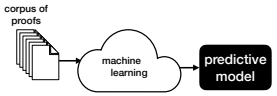


predictive model

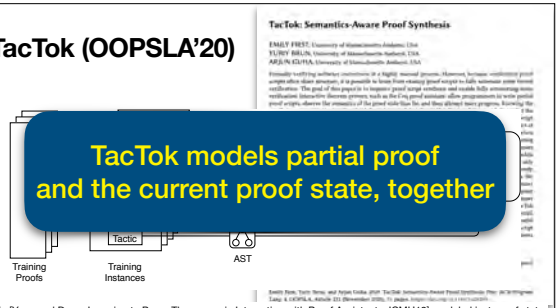
two todos

1. learn a precise predictive model
2. make search efficient

how to learn a predictive model



TacTok (OOPSLA'20)



TacTok models partial proof and the current proof state, together

ASTactic [Yang and Deng, Learning to Prove Theorems via Interacting with Proof Assistants, ICML'19] modeled just proof state. [Hellendoorn, Devanbu, Alipour, On the naturalness of proofs, ESEC/FSE NIER'18] looked at predictability of proof sequences.

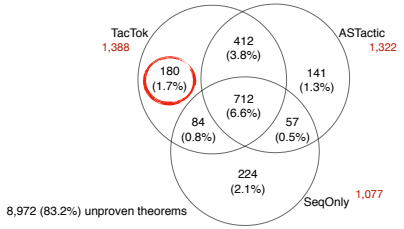
CoqGym Dataset

- 123 open-source software projets in Coq
- 70,856 theorems
- Broken down into 96 projects (57,719 proofs) for training and 27 projects (13,137 theorems) for testing

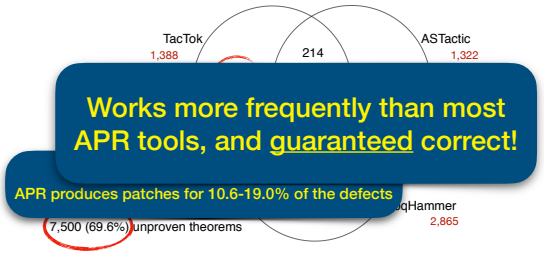
https://github.com/princeton-vl/CoqGym

[Yang and Deng, Learning to Prove Theorems via Interacting with Proof Assistants, ICML'19]

TacTok vs. ASTactic vs. SeqOnly



TacTok vs. ASTactic vs. CoqHammer

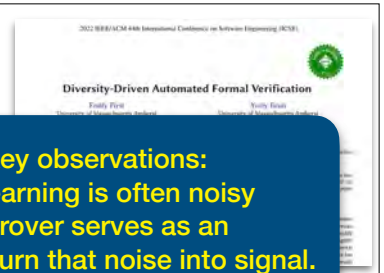


Building Precise Predictive Models

- TacTok: Semantics-Award Proof Synthesis (OOPSLA'20)
  - Combined predictions using proof state and partial proof
- "Passport: Improving Automated Formal Verification Using Identifiers" (TOPLAS'23)
  - Using semantic information in identifiers to improve prediction
- Diva: "Diversity-Driven Automated Formal Verification" (ICSE'22)
  - Combined multiple predictive models to make more diverse predictions and increase proving power
  - Together with CoqHammer, proved 33% of the properties fully automatically



Diva (ICSE'22)



2 key observations:
- Machine learning is often noisy
- Theorem prover serves as an oracle to turn that noise into signal.

Diva (ICSE'22)



- Vary:
  - proof tactic and token depth
  - learning rate
  - embedding size
  - number of layers
  - training order
  - access to proof state, partial proof, Gallina proof term

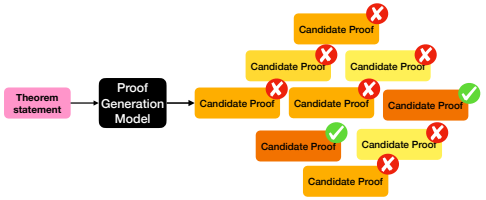
Diva vs. state-of-the-art



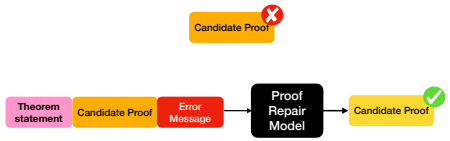
Diversity inherent in ML increases the proving power 68%-77% over prior search-based synthesis tools, and 27% over CoqHammer.

https://github.com/LASER-UMASS/Diva/

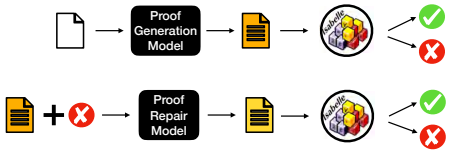
Baldur whole-proof generation



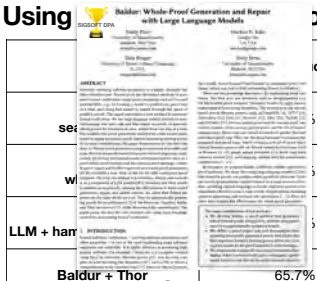
proof repair



Baldur: whole proof generation & repair



Baldur: Using [...] oof Synthesis

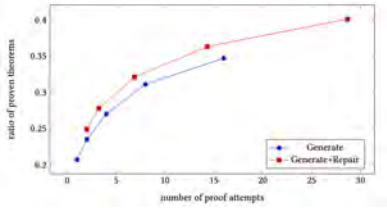


FSE'23

Baldur + Thor prove nearly 2/3 of PISA test set!

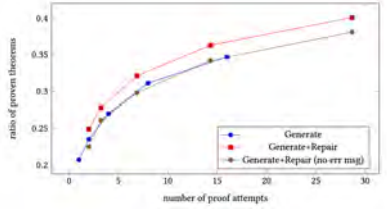
Jiang et al. (2021) "LISA: Language models of ISAbelle proofs" AITP
Jiang et al. (2022) "Thor: Wielding Hammers to Integrate Language Models and Automated Theorem Provers" NeurIPS 2022

Generate vs Generate+Repair



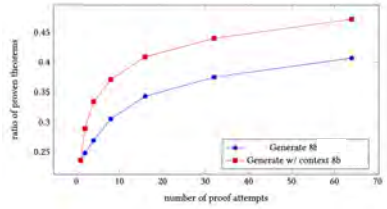
Generate+Repair outperforms Generate only

Generate+Repair with no error message



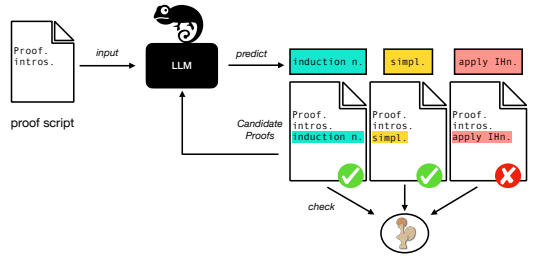
Error message is crucial for repair approach

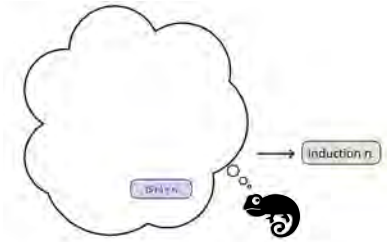
Generate with context

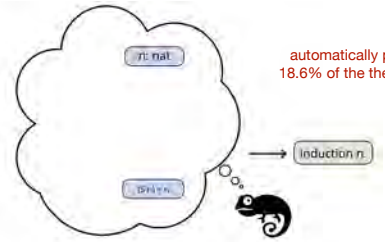


Proof context helps improve proof generation

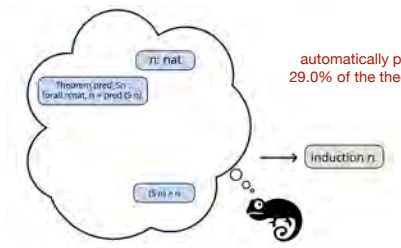
Rango



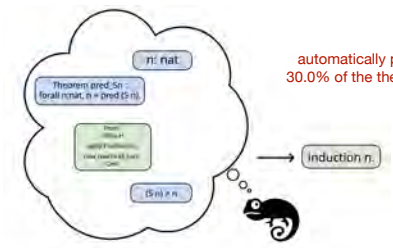




automatically prove
18.6% of the theorems



automatically prove
29.0% of the theorems



automatically prove
30.0% of the theorems



CoqHammer

+



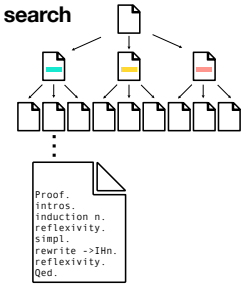
automatically prove
of the theorems

ICSE'25

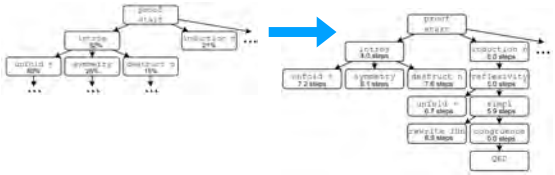
two todos

1. learn a precise predictive model
2. make search efficient

metaheuristic search



Idea: Use reinforcement learning to learn how far a partial proof is from QED



Without knowing where to go, we have to use Dijkstra's algorithm.

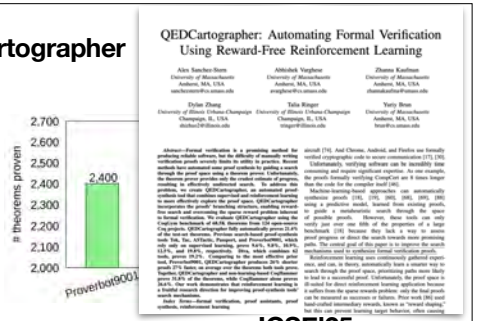


Without knowing where to go, we have to use Dijkstra's algorithm.

If we can estimate if we're getting close, we can use A*.



QEDCartographer



ICSE'25

Fully Automated Formal Verification

Machine learning and meta-heuristic search
can fully automate
some bug-repair and formal verification.

While APR underperforms because it is driven by an unreliable oracle,
formal verification is a killer app for APR because
the theorem prover provides a reliable oracle.