Machine Learning Across The Solver Stack

Federico Mora

Assistant Professor, University of Waterloo

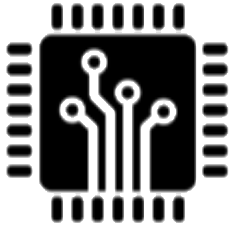
Faculty Affiliate, Vector Institute

(I’m hiring grad students!)

2nd Workshop on Machine Learning for Solvers and Provers (ML4SP)

Combinatorial Reasoning is Everywhere

Solving package dependencies

Finding circuit optimizations

Scheduling sports leagues

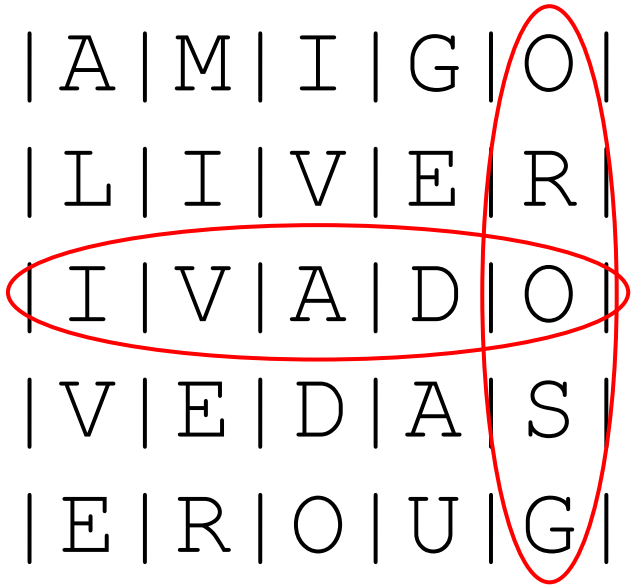
[1]https://en.opensuse.org/openSUSE:Libzypp_satsolver [2]Mischenko et al. (ICCAD 2006) [3]Horbach et al. (Scheduling 2010)

Combinatorial Reasoning Is Hard For LMs

```
Q: Make a 5 by 5 crossword grid with the fewest possible blanks
(use * for blanks) as a markdown table using the following
fills: DIP, IMOUT, SAUCE, CYCLE, HAN, DISC, IMAY, POUCH, UCLA,
TEEN.
A:
| | | | | |
|-|-|-|-|-|
|D|I|P|*|*|
|I|M|O|U|T|
|S|A|U|C|E|
|C|Y|C|L|E|
|*|*|H|A|N|

Q: Make a 5 by 5 crossword grid with the fewest possible blanks
(use * for blanks) as a markdown table using the following
fills: {WORDS}
A:
| | | | | |
|-|-|-|-|-|
```

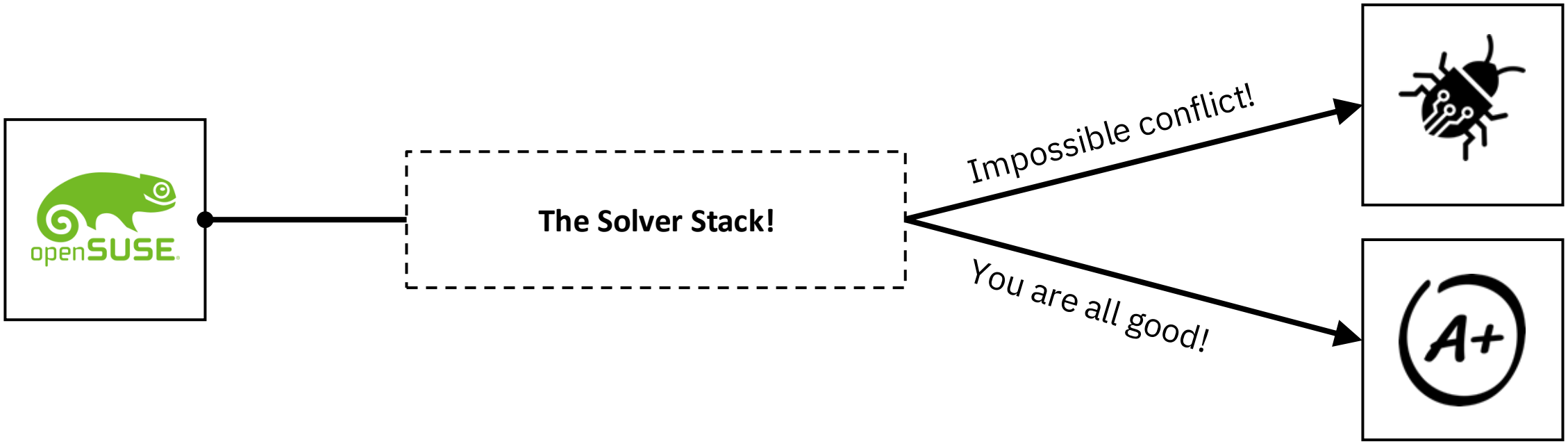
Combinatorial Reasoning Is Hard For LMs



---

[1]Mora (Thesis)

Solvers Are Great At Combinatorial Reasoning!

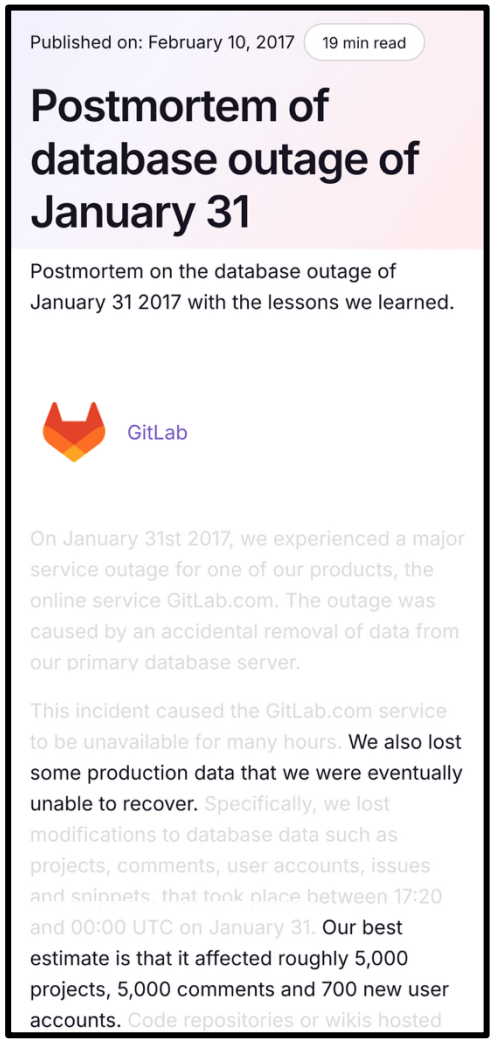


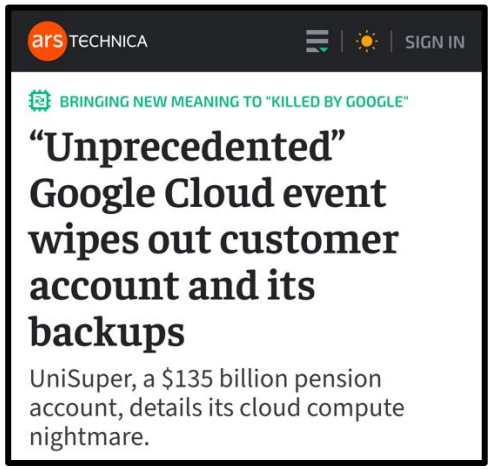
Today: Why/Where to Combine ML with Solvers

- One domain form start to finish
  - Gives concrete examples
  - Presents wholistic view of solvers (the whole solver stack!)

Today’s Focus: Distributed Systems Verification
(Big and Important!)

[1]AWS Global Infrastructure Map (2025) [2]O’Reilly Cloud Adoption Report (2021) [3]Goldman Sachs (2024)



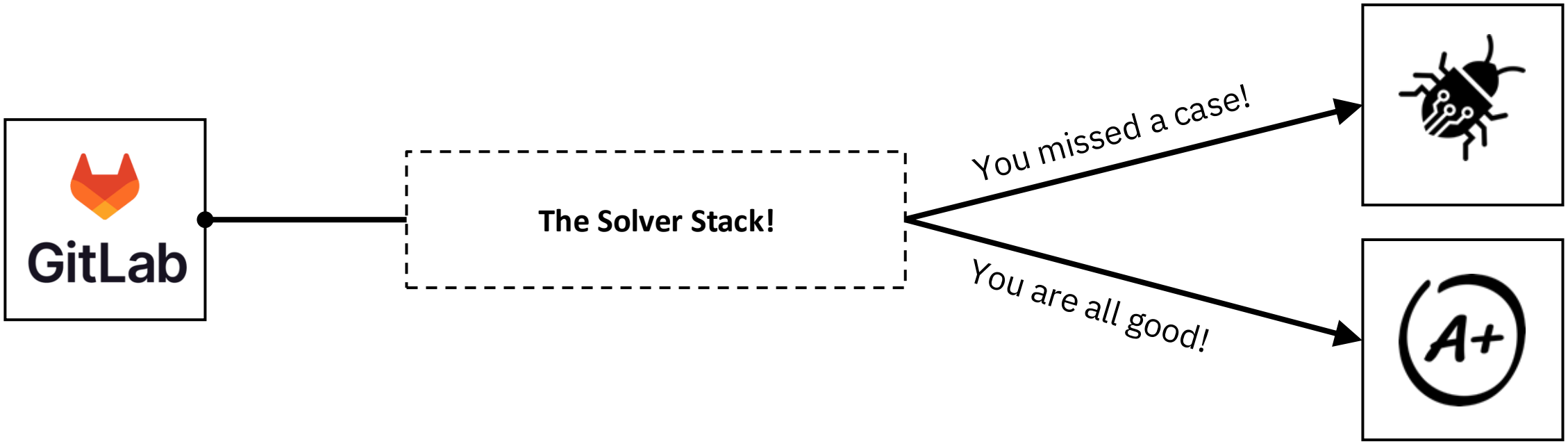


"We examined three diskless recovery protocols… We found that each of these protocols suffers from … serious correctness violations."[1]

Despite extensive testing, fault and timing-related bugs account for 44% of production Microsoft Azure incidents.[2]

[1]Michael et al. (DISC '17) [2]Liu et al. (HotOS '19)

Engineers Need Complete Checks!



Complete Checks Require Expertise!

Goal: Equip software engineers with scalable and usable automated reasoning.

Challenge: Successful users must be experts in their domain **and** automated reasoning.

> "This limits the reach of formal verification: scalability will require teaching many more people to engineer proofs"[1]

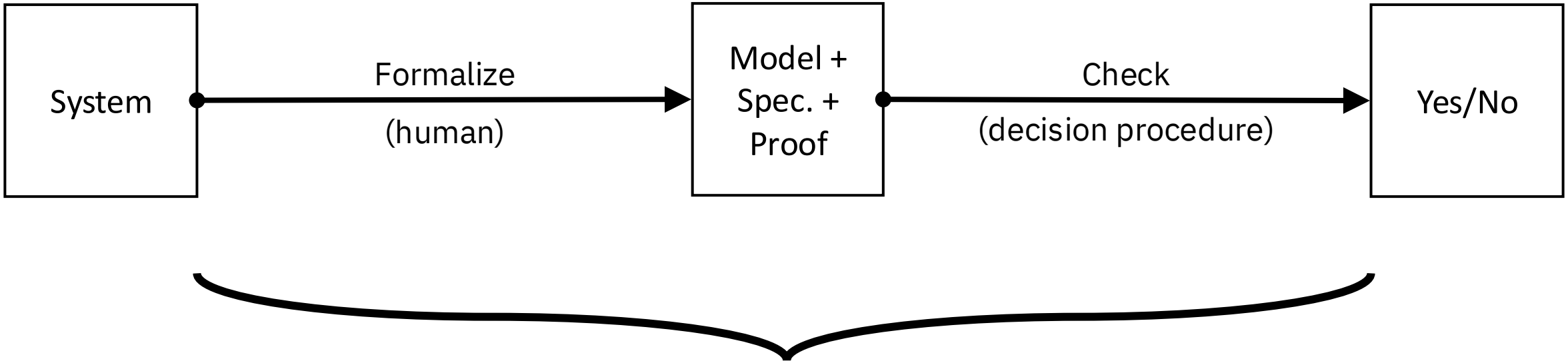
> "The full power of automated reasoning is not yet available to everyone because today's tools are either difficult to use or weak"[2]

> "We cannot expect all AWS users to be experts in formal methods, have the time to be trained in the use of formal methods tools, or even be experts in the cloud domain"[3]

---

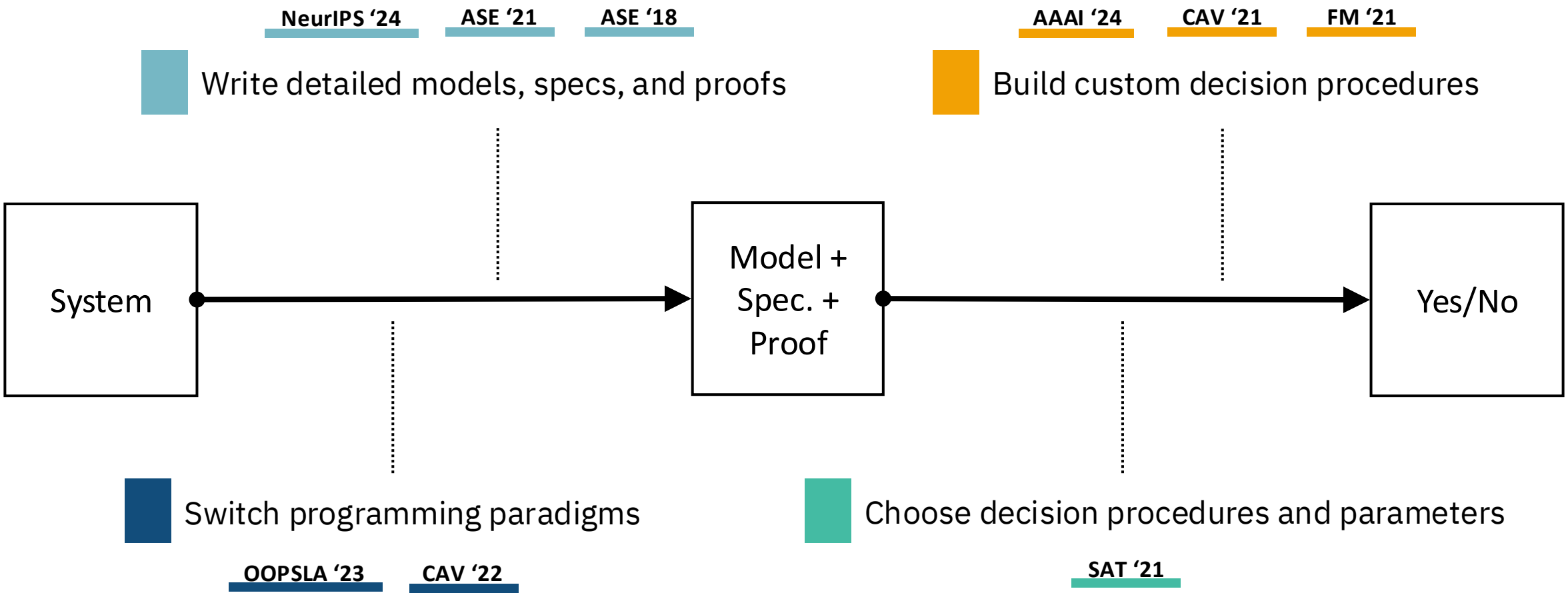
[1]Dodds (IEEE Security & Privacy '22) [2]Cook (AWS Security Blog '19) [3]Rungta (CAV '22)

Solver Stack Sketch



Usability and scalability concerns:
both steps require automated reasoning expertise for success at scale

Users must know their domain *and* ...

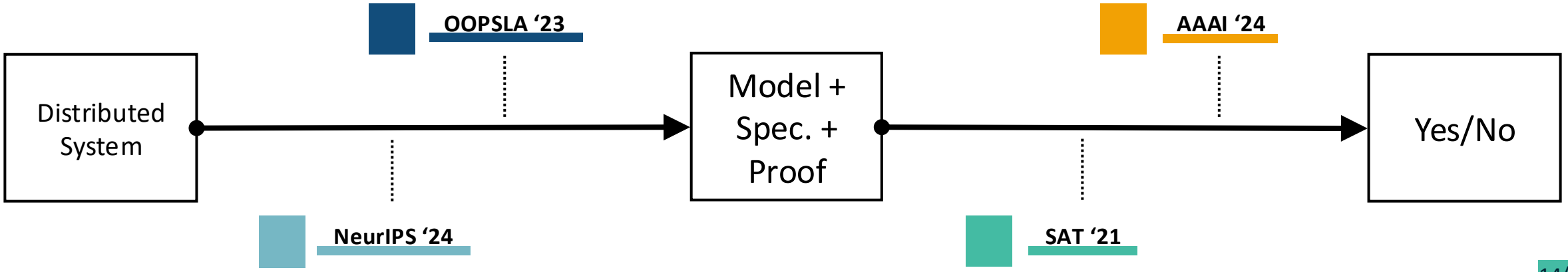


Contributions: Distributed Systems Solver Stack

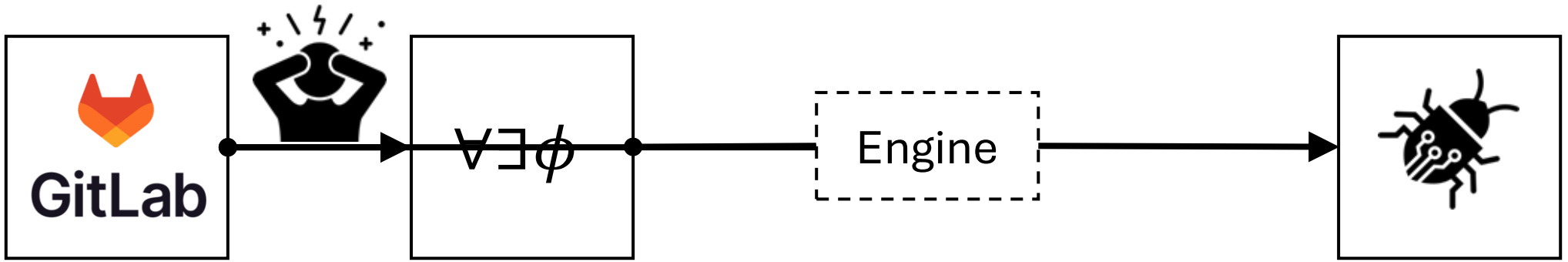
Challenge: Successful users must be experts in their domain **and** automated reasoning (AR).

Solution: Reduce the need for automated reasoning expertise by tailoring every level of the AR stack.

- Avoid programming paradigm switch with domain-specific verification framework
- Reduce formalization burden with neuro-symbolic automation
- Reduce need for custom decision procedures with solver for user-defined algebraic data types
- Automate choice for decision procedures and parameters with reinforcement learning



Why Switch Programming Paradigms?

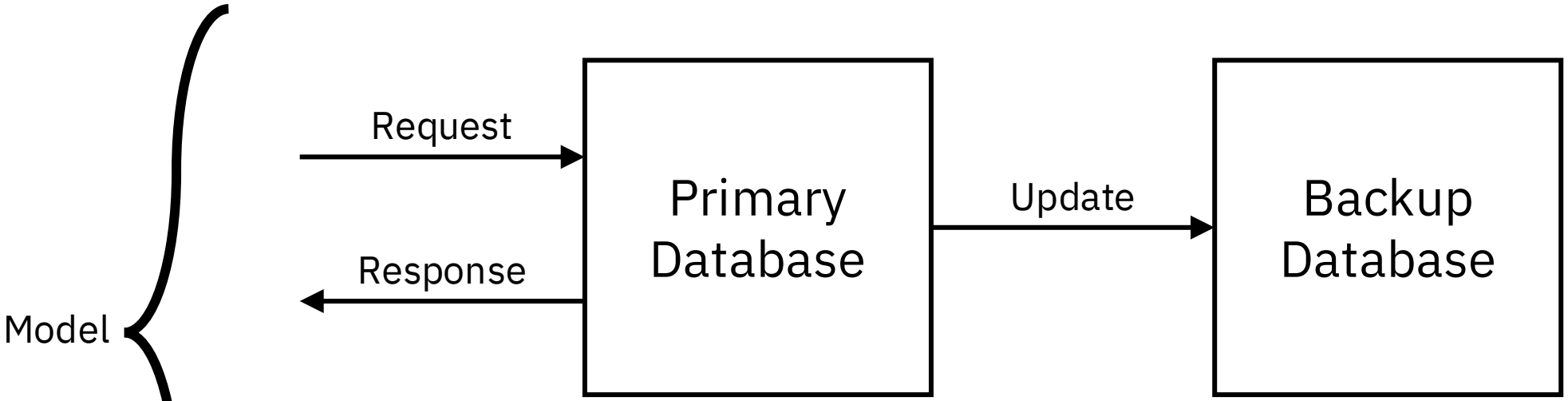


Actor Paradigm
(-like)

First-order logic
(-like)[1,2,3]

[1]Wilcox et al. (CAV '24) [2]Lamport (TOPLAS '94) [3]McMillan and Padon (CAV '20)

Formalization Example: GitLab Async



- "GitLab.com currently uses a single primary and a single secondary in hot-standby mode."[1]
- Using asynchronous replication[2]

Spec.

- "Losing production data is unacceptable."[1] (If they accept your write, they have two copies)

---

[1]https://about.gitlab.com/blog/2017/02/10/postmortem-of-database-outage-of-january-31/
[2]https://gitlab.com/gitlab-com/gl-infra/production-engineering/-/issues/7282

PVerifier: Domain-Specific Verification

Message Chains for Distributed System Verification (OOPSLA '23)

- P is a domain-specific language for modeling async (event-driven) state machines.[1,2]

P Model

```
machine PrimaryDatabase {
  on eRequest q do {
    ...
    send q.source, eResponse(r)
  } ...
} ...
```

```
spec TwoCopies observes eResponse {
  on eResponse r do {
    assert primary.db == backup.db
  }
}
```

P Spec.

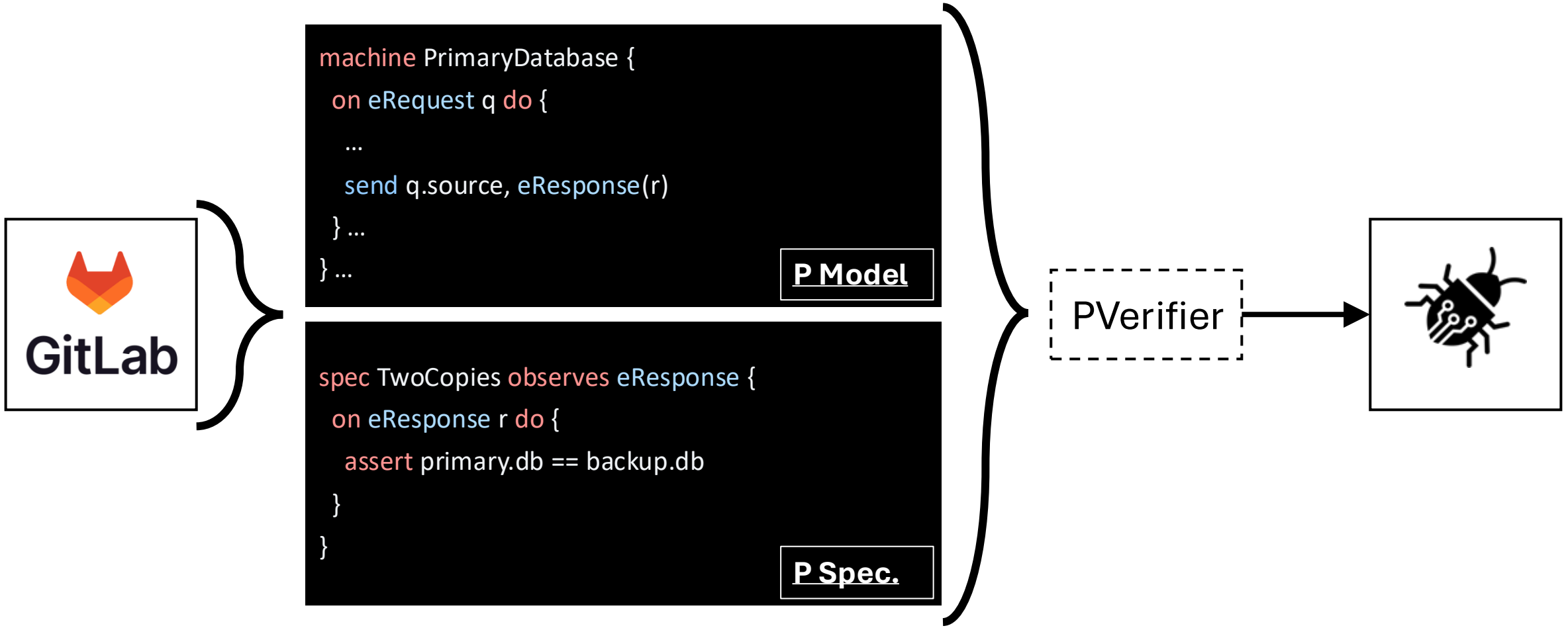
- Engineers are using P for systematic testing of distributed systems.[3,4]
- **We add formal verification to P!**

---

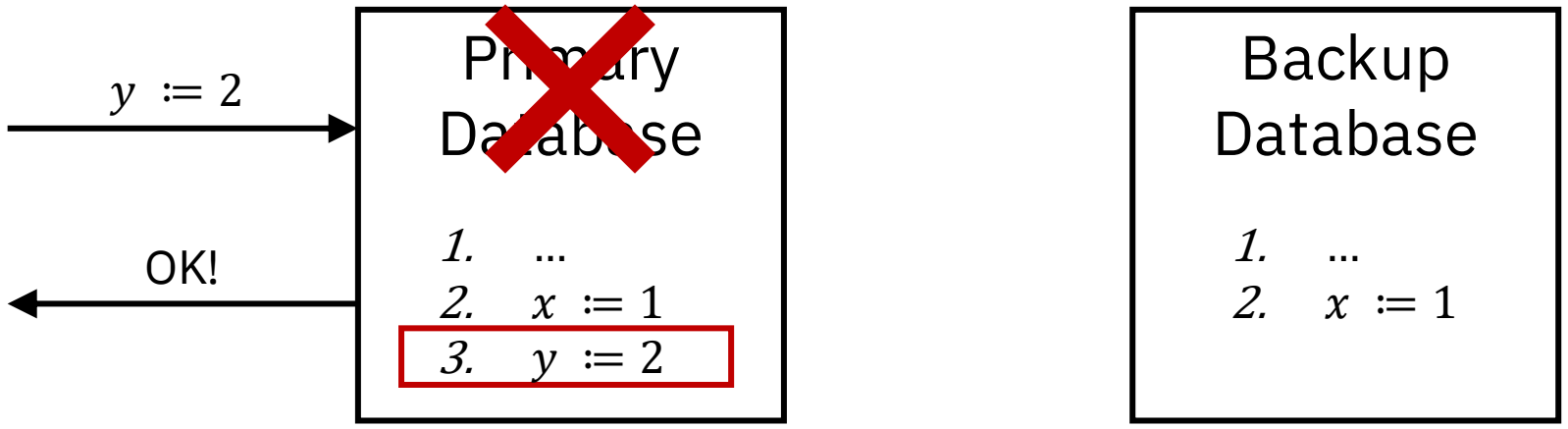
[1]Desai et al. (PLDI '13) [2]Desai et al. (OOPSLA '18) [3]Desai (Amazon '23) [4]Terry (Amazon '24)

PVerifier: Domain-Specific Verification

Message Chains for Distributed System Verification (OOPSLA '23)



GitLab Bug



- “Losing production data is unacceptable” (but there was only one copy of y := 2!)

PVerifier Evaluation

- Reproduced 16 verification efforts from related work.
  - Compared proof size and solver performance.
  - Empirically, **our domain-specific abstractions are zero-cost for verification**!
- Officially released in the P compiler in collaboration with AWS![2]
- Beta version already used inside AWS before release.[3]
  - A co-author replicated two existing proofs.
  - **A PVerifier user found a bug in a handwritten proof for an internal protocol!**

---

[1]https://github.com/uclid-org/uclid [2]https://github.com/p-org/P

Key Idea: Message Chains (PL Design, Not ML)

- The hardest part of proving correctness: auxiliary invariants (lemmas)
- Hard to write lemmas over individual messages!
- Enter message chains: sequences of causally related messages
- PVerifier lets users write higher-level proofs using message chain invariants

```
machine PrimaryDatabase {
 on eRequest q do {
   …
   send q.source, eResponse(r)
 } …
} …
```

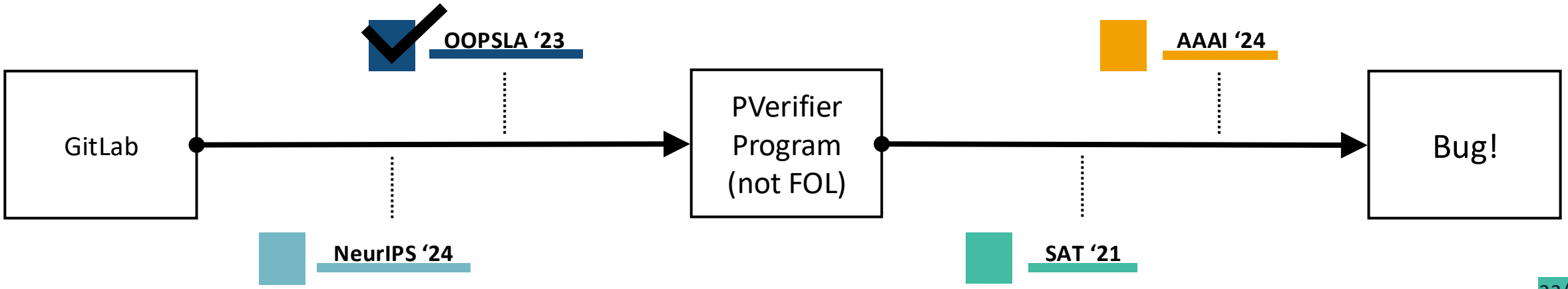
E.g., a specific eRequest causes a specific eResponse

Contributions: Distributed Systems Solver Stack

Challenge: Successful users must be experts in their domain **and** automated reasoning (AR).

Solution: Reduce the need for automated reasoning expertise by tailoring every level of the AR stack.

- ✓ Avoid programming paradigm switch with domain-specific verification framework (**No ML**)
- Reduce formalization burden with neuro-symbolic automation
- Reduce need for custom decision procedures with solver for user-defined algebraic data types
- Automate choice for decision procedures and parameters with reinforcement learning



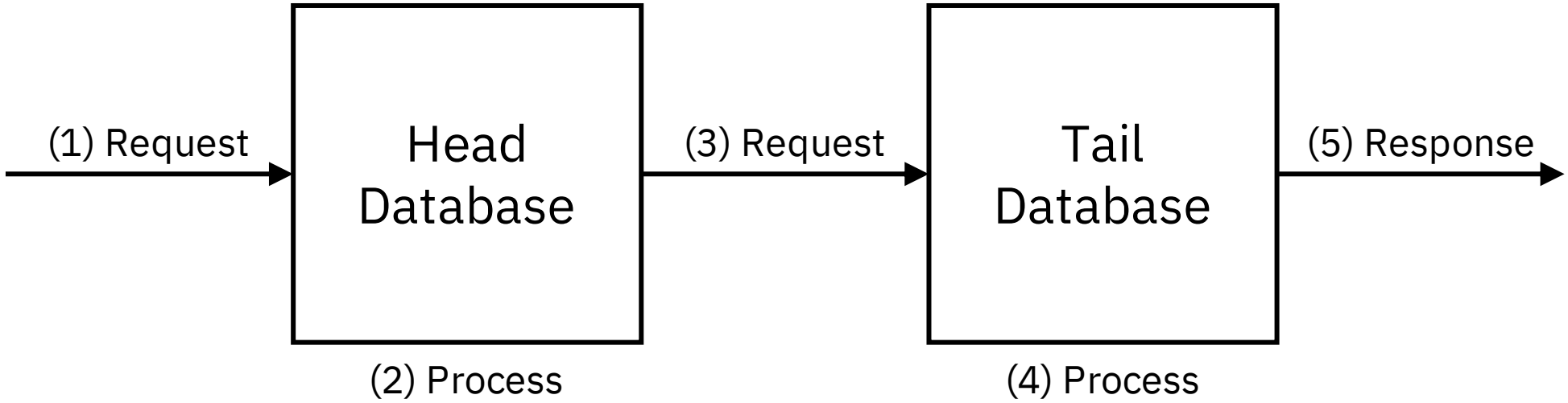
Why Is There A Formalization Burden?

> "Microsoft was able to write 5,000 lines of verified Dafny code [IronFleet[1]] in only 3.7 person-years! That's a blazing-fast rate of *four whole lines a day*."[2]

Any automation will have a big impact!

[1]Hawblitzel et al. (SOSP '15) [2]Wayne (Blog '19)

GitLab Fix: (Mini) Chain Replication[1]



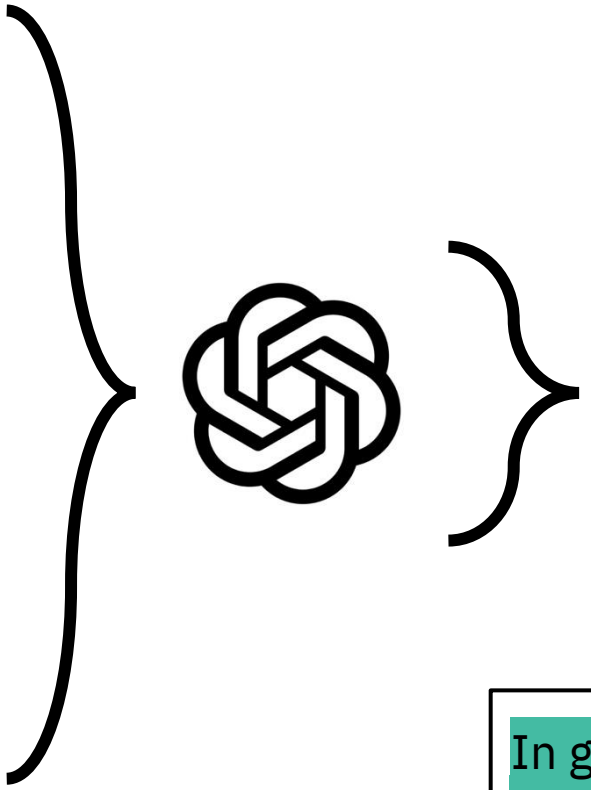
- “Losing production data is unacceptable.” (If we accept your write, we have two copies)

---

[1]van Renesse and Schneider (OSDI ‘04)

ChatGPT: (Mini) Chain Replication Example

“ Write a chain replication protocol in UCLID5 with just two nodes. The primary node receives requests from the client, forwards them to the backup node, the backup node responds to the primary, and the primary responds to the client. Both nodes should have a memory that is updated with each request. The specification is that after every response sent to the client, the primary and backup memories should always be equal. ”



```
module ChainReplication2Nodes {
  ...
  fairness {
    request -> (primary_memory == backup_memory);
    response -> (primary_memory == backup_memory);
  }
  ...
}
```

In general, LLM baselines (including prompting, self-repair, …) struggle with formal languages

Eudoxus: (Mini) Chain Replication Example

Synthetic Programming Elicitation for Text-to-Code in Very Low-Resource Programming and Formal Languages (NeurIPS '24)

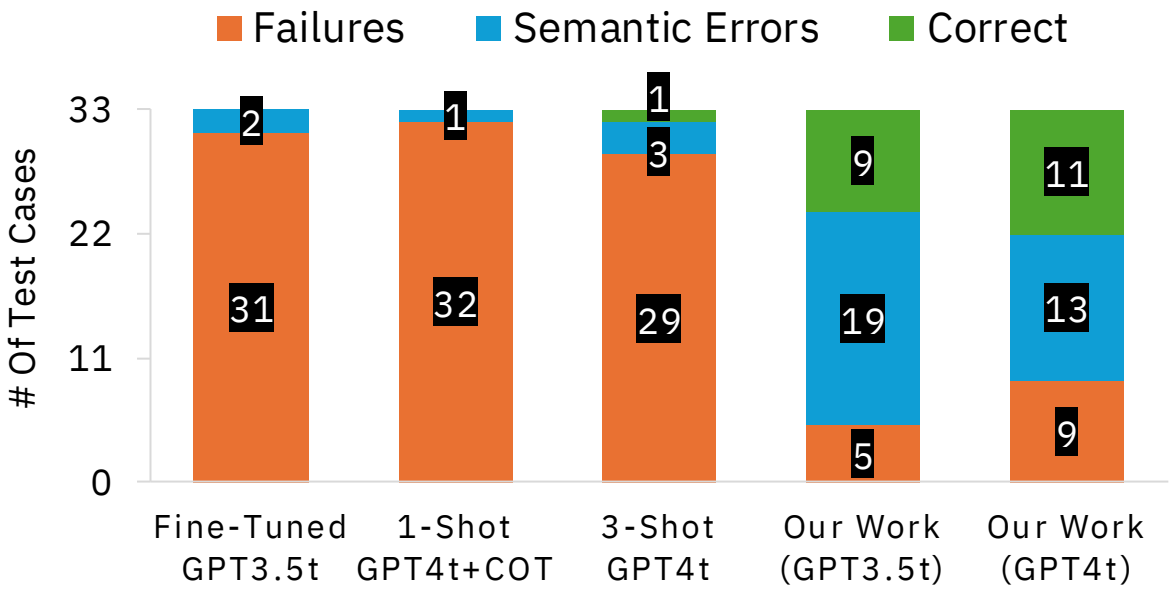
" Write a chain replication protocol in UCLID5 with just two nodes. The primary node receives requests from the client, forwards them to the backup node, the backup node responds to the primary, and the primary responds to the client. Both nodes should have a memory that is updated with each request. The specification is that after every response sent to the client, the primary and backup memories should always be equal. "

Eudoxus
(MAX-SMT + LLM)

```
module Chain {
  ...
  next {
    ...
    node1.client_request' = client_request;
    node2.client_request' = node1.response;
    ...
  }
}
```

Manually add the specification and run the verification engine. It's correct!

Eudoxus Evaluation



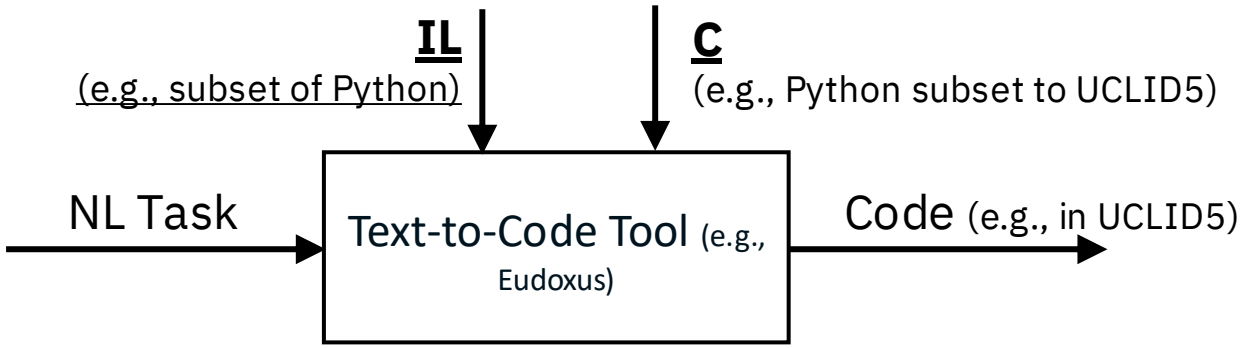
- 33 exercises from 3 textbooks.[1,2,3]
- 5 iterations maximum (15% reach this).
- **85% vs. max 12%** pass compiler with LLM baselines.
- The majority are also semantically correct or almost correct (missing specification is a semantic error).
- Piloting in UC Berkeley's EECS 219C: Formal Methods

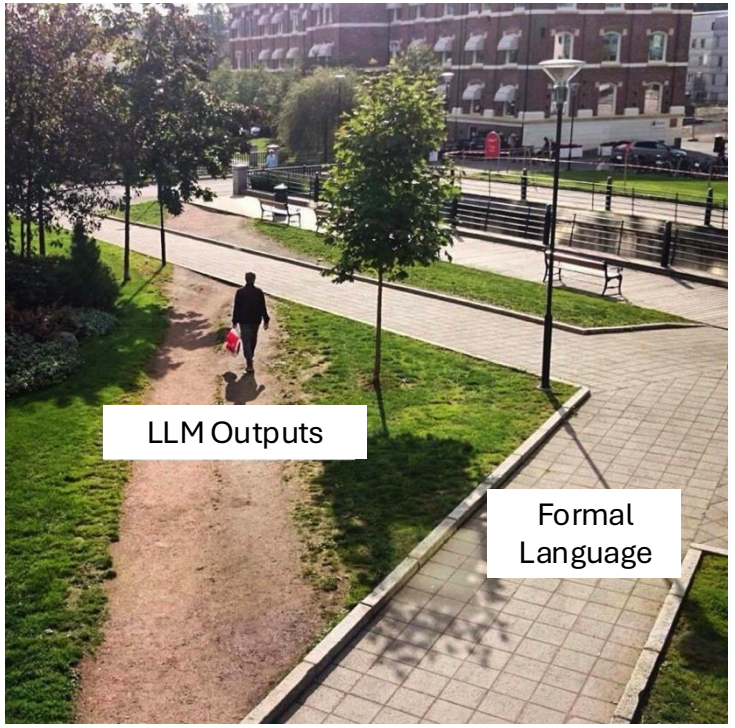
[1]Lee and Seshia ('16) [2]Huth and Ryan ('04) [3]Baier and Katoen ('08)

Key Idea: Parse, Don't Prompt!

- Natural programming elicitation[1]
  - Understand what programmers find "natural"
  - Design a programming language or tool for that

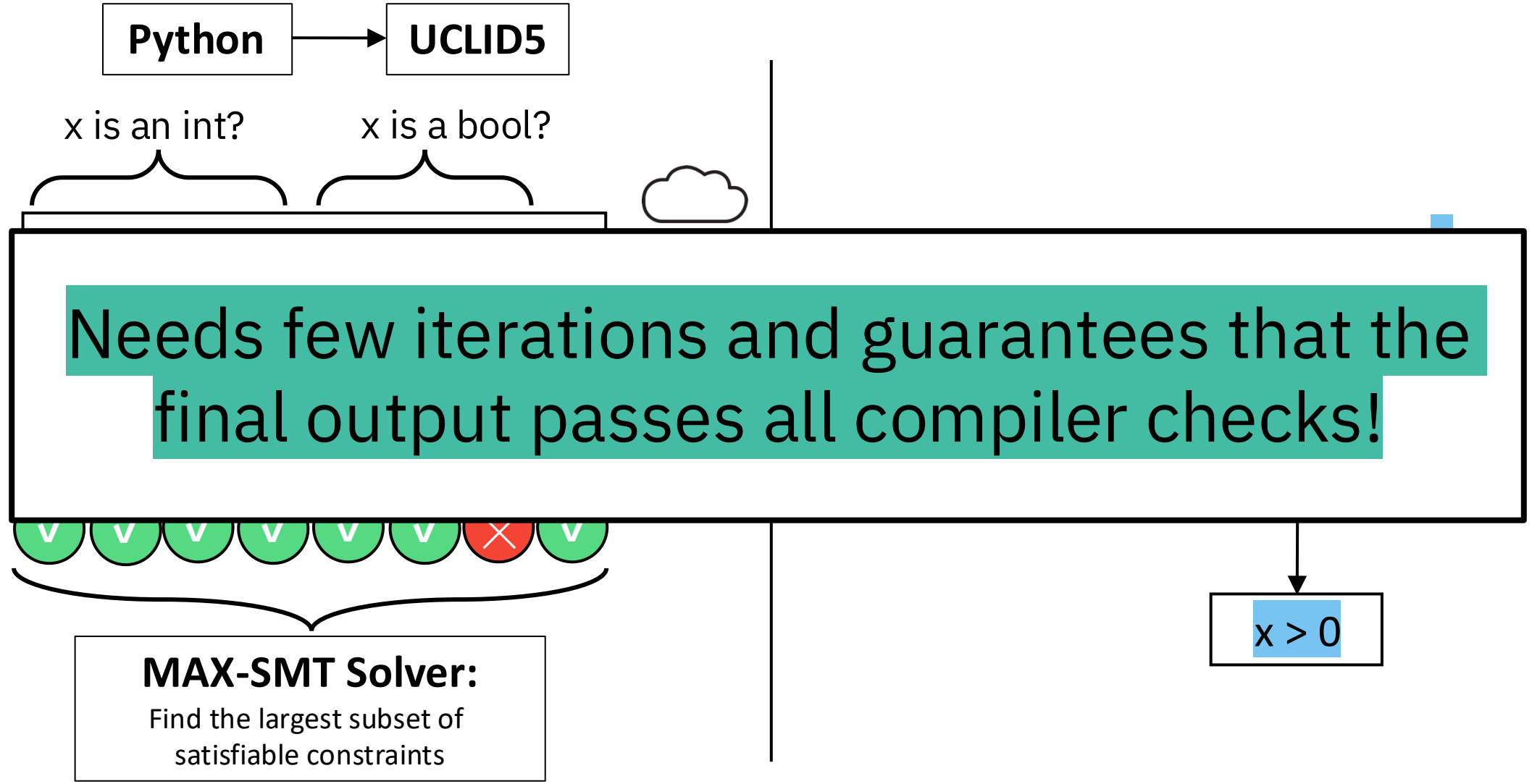
- Our work: synthetic programming elicitation.
  - Understand what LLM finds "natural" in your domain.
  - Design an intermediate language (**IL**) for that.
  - Write a compiler (**C**) from **IL** to target language.





[1]Myers, Pane, and Ko (CACM '04)

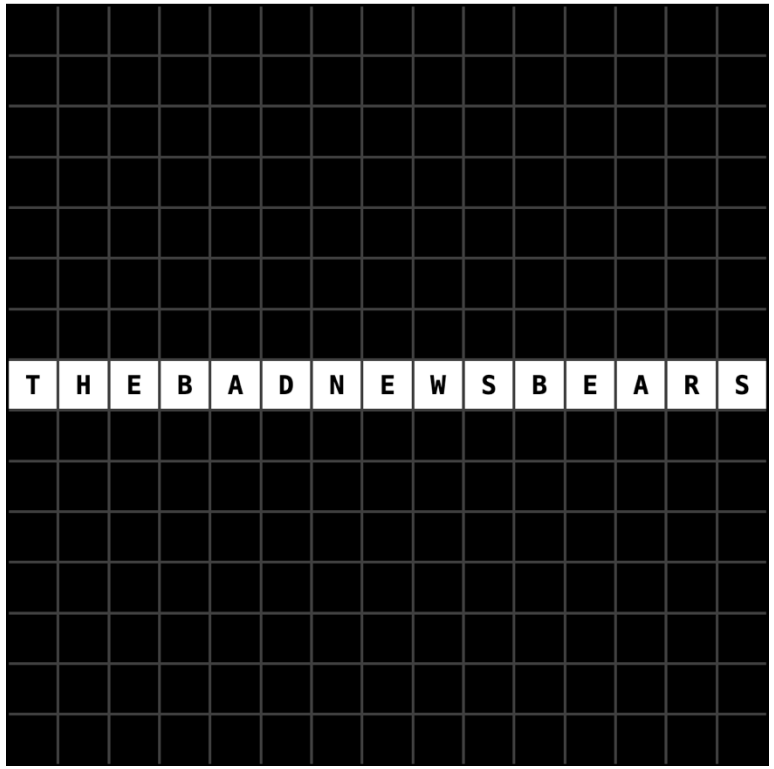
Key Idea: Repair, Don't Despair!



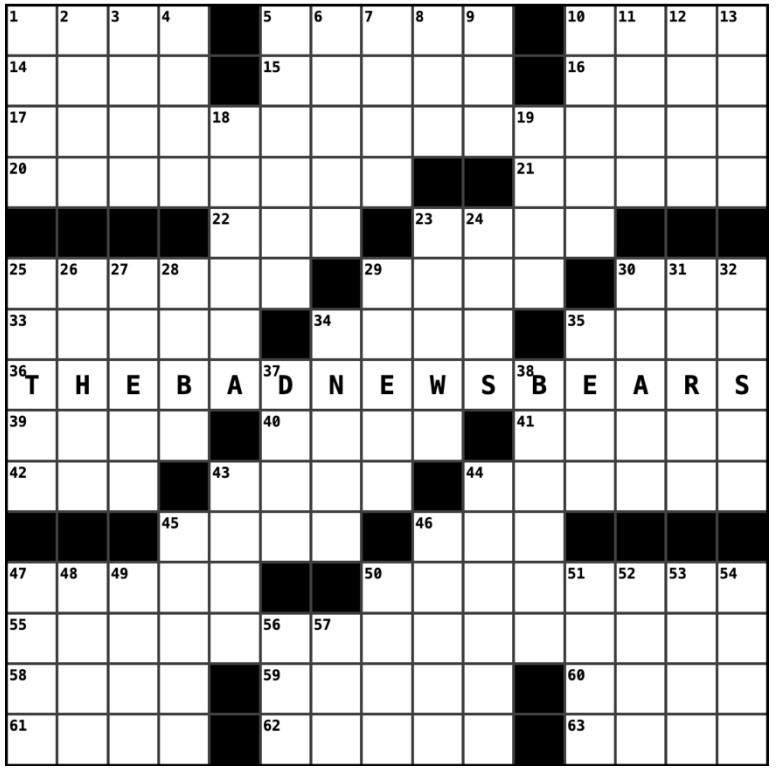
MAX-SMT encoding inspired by Pavlinovic, King, and Wies (OOPSLA '14)

Crossword Construction Analogy

Start with one entry
(~write a line of code)



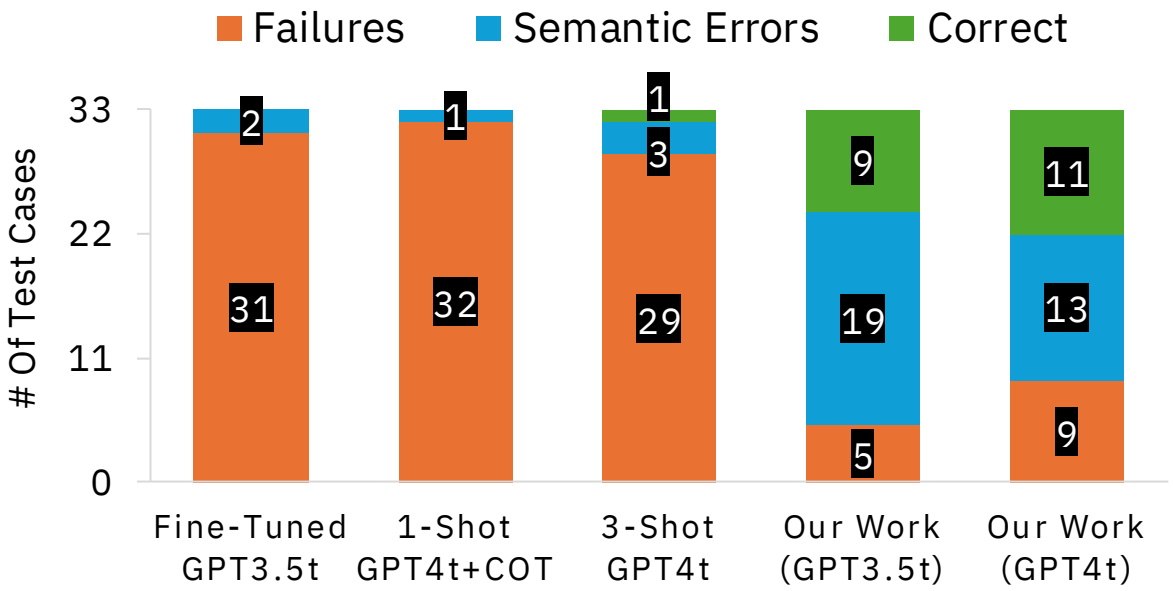
Fill in words until you have a puzzle
(~write the rest of your program)



Every entry has implications down the line. We probably need to backtrack! But to where?!
(MAX-SMT solver tells us where!)

New York Times Crossword 03/23/1995

Eudoxus Evaluation (ML Can Help!)



- 33 exercises from 3 textbooks.[1,2,3]
- 5 iterations maximum (15% reach this).
- **85% vs. max 12%** pass compiler with LLM baselines.
- The majority are also semantically correct or almost correct (missing specification is a semantic error).
- Piloting in UC Berkeley's EECS 219C: Formal Methods

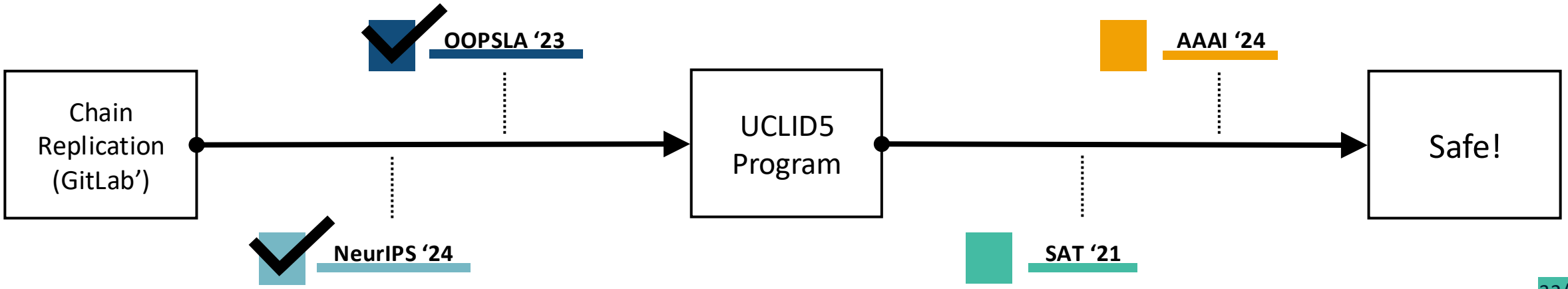
[1]Lee and Seshia ('16) [2]Huth and Ryan ('04) [3]Baier and Katoen ('08)

Contributions: Distributed Systems Solver Stack

Challenge: Successful users must be experts in their domain **and** automated reasoning (AR).

Solution: Reduce the need for automated reasoning expertise by tailoring every level of the AR stack.

- ✓ Avoid programming paradigm switch with domain-specific verification framework (No ML)
- ✓ Reduce formalization burden with neuro-symbolic automation (**Yes ML**)
- Reduce need for custom decision procedures with solver for user-defined algebraic data types
- Automate choice for decision procedures and parameters with reinforcement learning



Why Custom Decision Procedures?

Need to answer yes or no questions

e.g., “does this proof by induction work?” (PVerifier)

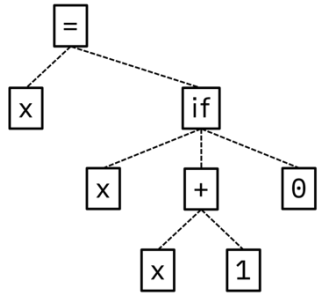
e.g., “can I compile this intermediate program to the target language”? (Eudoxus)

Satisfiability Modulo Theories (SMT)

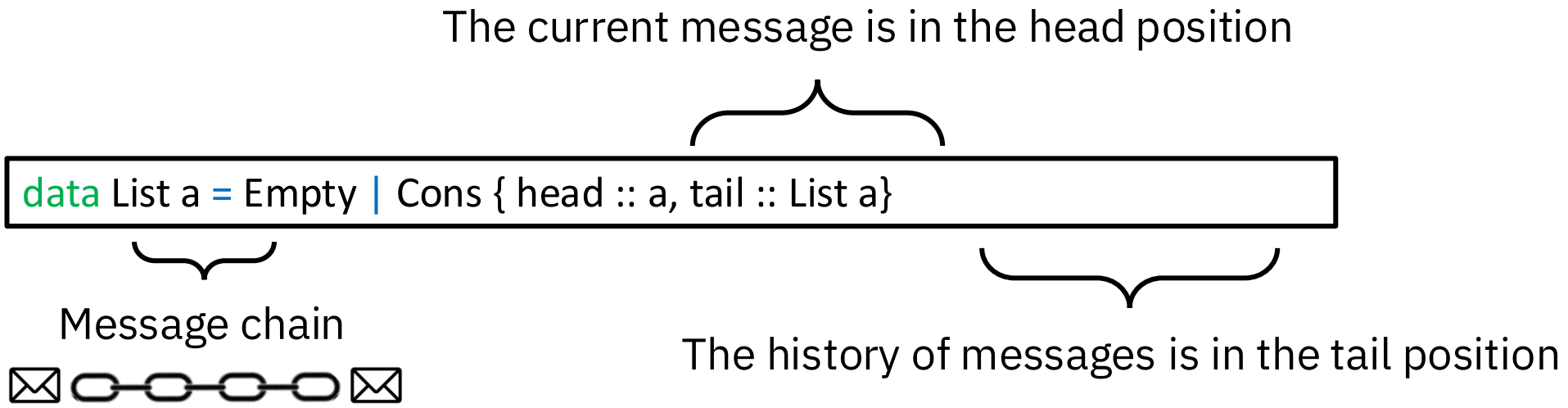
(like Boolean SAT, but with richer structures defined by background theories)

Do we need three new fast solvers?!

```
struct machine_state
{
    …
};
```



No! Just One Solver for Algebraic Data Types!



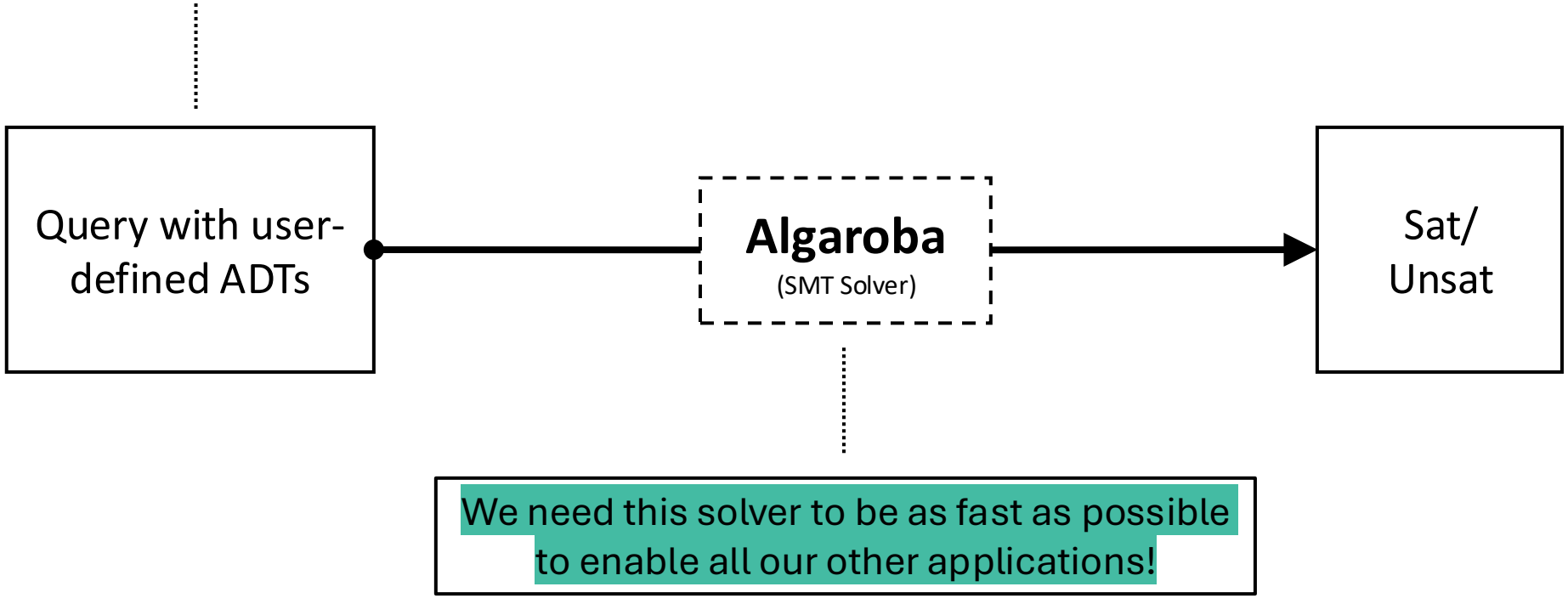
Algebraic data types (e.g., List) consist of

- constructors (e.g., Empty and Cons),
- selectors (e.g., head and tail),
- testers (e.g., is Empty and is Cons).

Algaroba: Solver for Algebraic Data Types

An Eager Satisfiability Modulo Theories Solver for Algebraic Datatypes (AAAI '24)

Generated by PVerifier, UCLID5, Eudoxus, ...



Theory of Algebraic Data Types

Well-Foundedness Axiom

Let u and v be two ADT values and let s_i be selectors.

If $u = v.s_1.s_2 \ldots s_n \wedge \theta$ then $u \neq v$,

where θ asserts that all s_i are correctly applied.

Input Query

```
var x: List;
assert x = x.tail;
assert is_Cons(x);
```

+

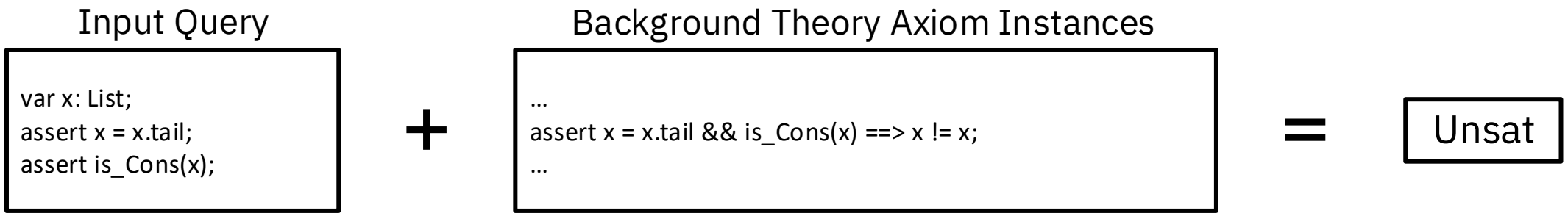
Background Theory Axiom Instances

```
...
assert x = x.tail && is_Cons(x) ==> x != x;
...
```

=

Unsat

Related Work: How to Introduce Axioms



- *Lazy* approaches (add axiom instances as needed):
  - SMTInterpol,[1] cvc5,[2] and Z3.[3]

- We take an *eager* approach (add all necessary axiom instances upfront)!
  - Turns out to be faster!

---

[1]Christ, Hoenicke, and Nutz (MCS '12) [2]Barbosa et al. (TACAS '22) [3]de Moura and Bjørner (TACAS '08)

Eager Reduction Challenge

Let $\phi(x)$ be the input query.

Eager solver must instantiate well-foundedness axiom for

$$x \neq x.s_i$$

$$x \neq x.s_i.s_j$$

$$\ldots$$

upfront, where s_l are selectors.

When do we stop?

We define a measure $k(\phi)$.

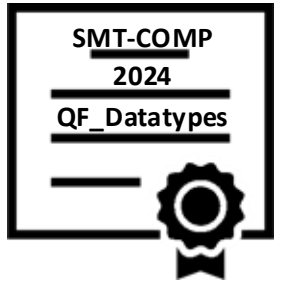
Our eager solver instantiates well-foundedness axiom up to

$$x \neq \underbrace{x.s_i \ldots s_j}_{k(\phi) \text{ applications}}$$

We prove this is sound and complete.

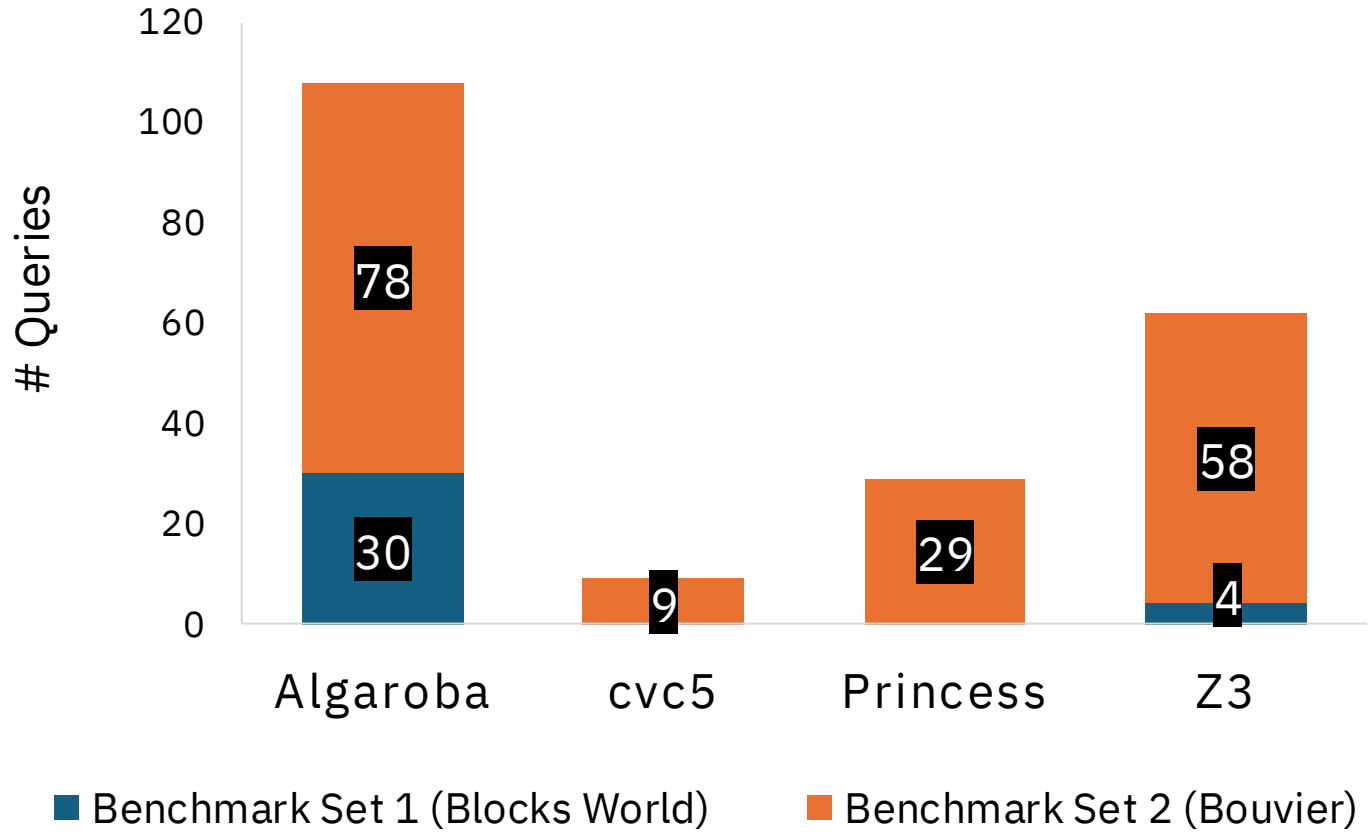
Algaroba Evaluation

Algaroba is the fastest solver in the world for (quantifier-free) algebraic data type queries!



Number of Queries Uniquely Solved per Solver

(1200 second timeout, 900 queries total)



[1]Bouvier (INRIA TR '21)

Need More Fast and Correct Solvers!

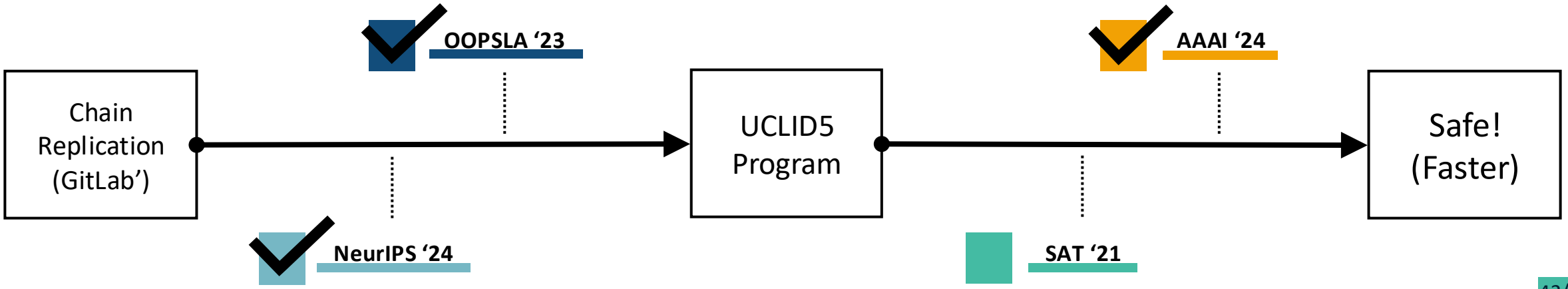
- New domains will benefit from new solvers!
  - e.g., quantified but not inductive ADTs are particularly useful in hardware formal verification.
- Bad news:
  - Building a tool like Algaroba requires a lot of engineering.
- Good news:
  - Algaroba and similar tools are term rewriting systems.
- **Future work**: build up infrastructure and metatheory to be able to quickly
  - create new eager solvers by declaring rewrite rules, and
  - prove that these solvers are sound and complete.
  - **Use LLMs in both steps!**

Contributions: Distributed Systems Solver Stack

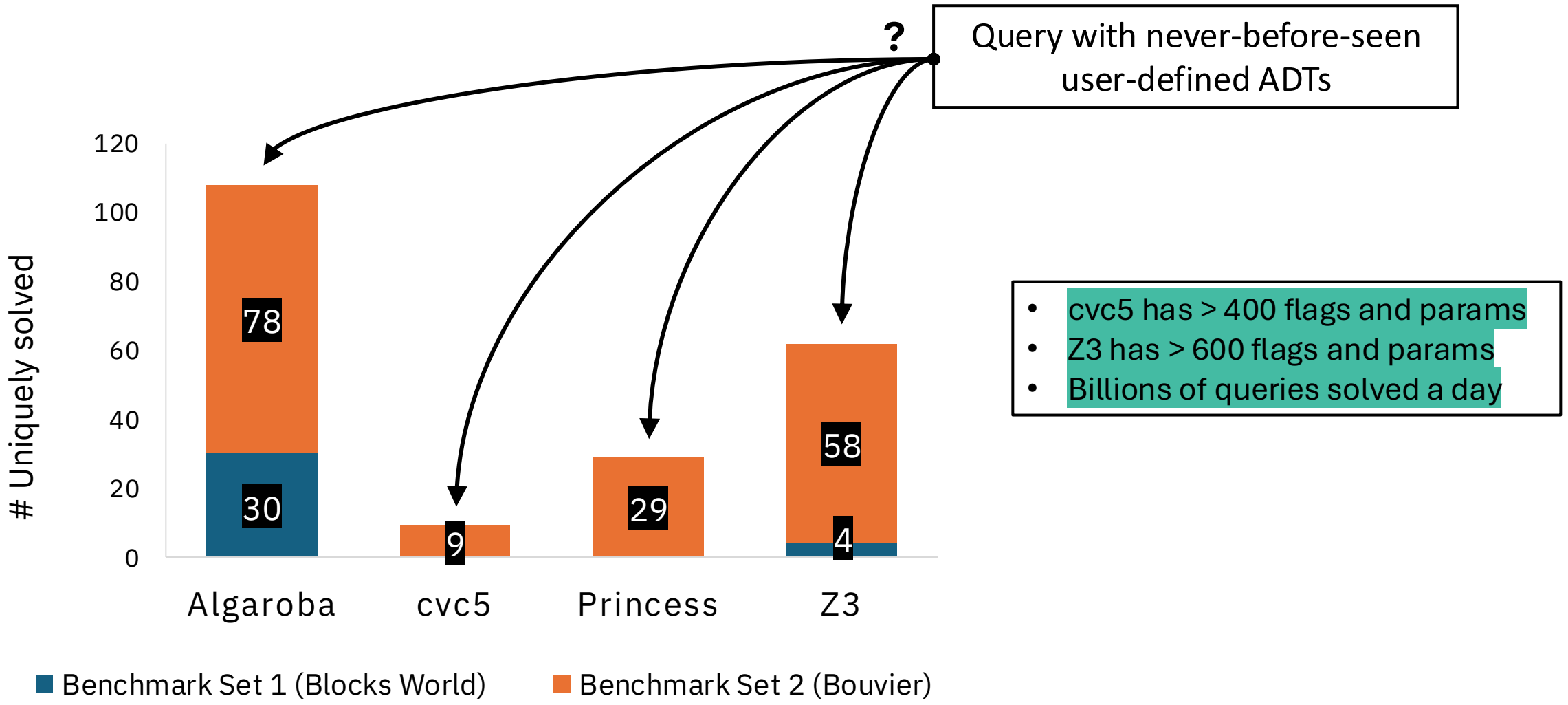
Challenge: Successful users must be experts in their domain **and** automated reasoning (AR).

Solution: Reduce the need for automated reasoning expertise by tailoring every level of the AR stack.

- Avoid programming paradigm switch with domain-specific verification framework (No ML)
- Reduce formalization burden with neuro-symbolic automation (Yes ML)
- Reduce need for custom decision procedures with solver for user-defined algebraic data types (**Need More ML**)
- Automate choice for decision procedures and parameters with reinforcement learning



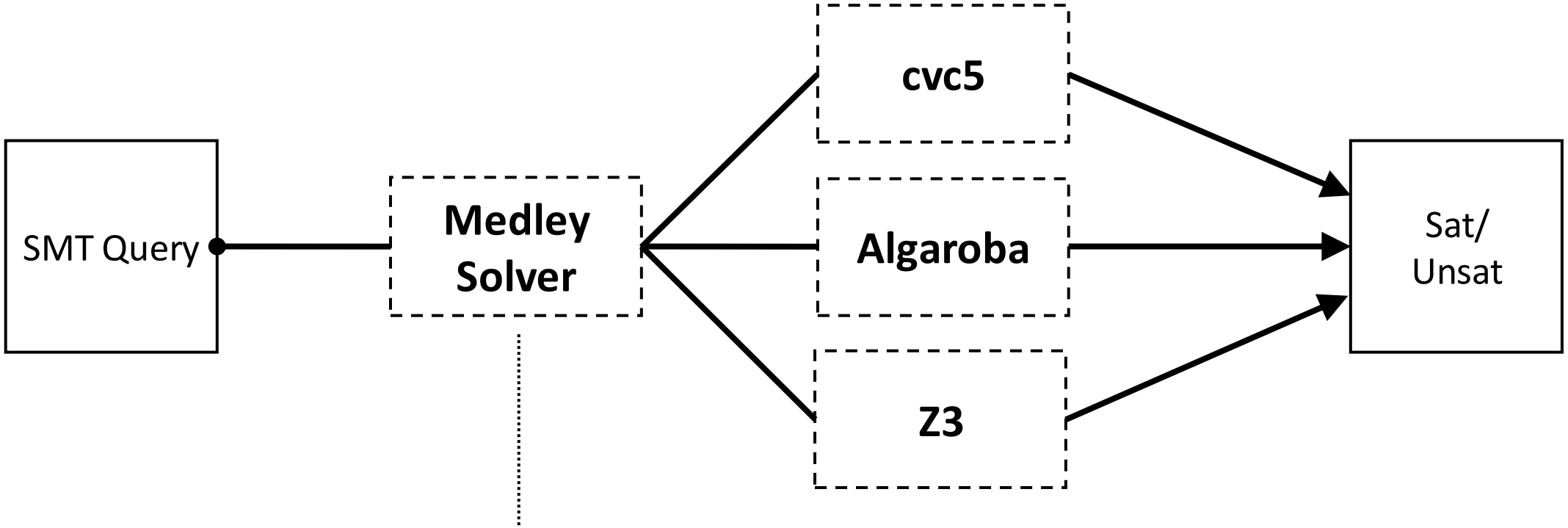
Why Choose Decision Procedures and Parameters?



- cvc5 has > 400 flags and params
- Z3 has > 600 flags and params
- Billions of queries solved a day

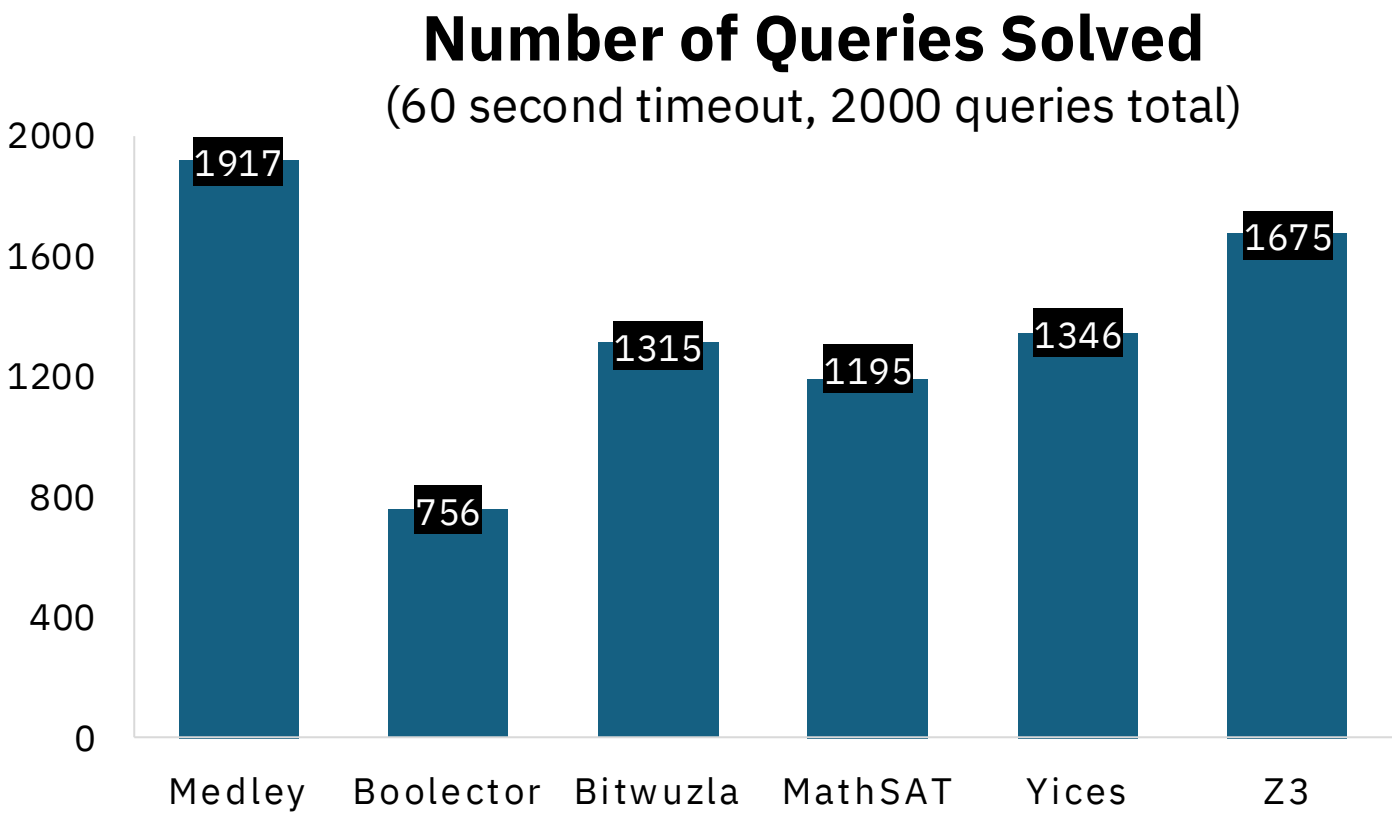
MedleySolver Picks For You!

MedleySolver: Online SMT Algorithm Selection (SAT '21)



First **online** (learn as you go) solver selection technique!
Uses a multi-armed bandit formulation to adjust to any new domain you throw at it!

MedleySolver Evaluation



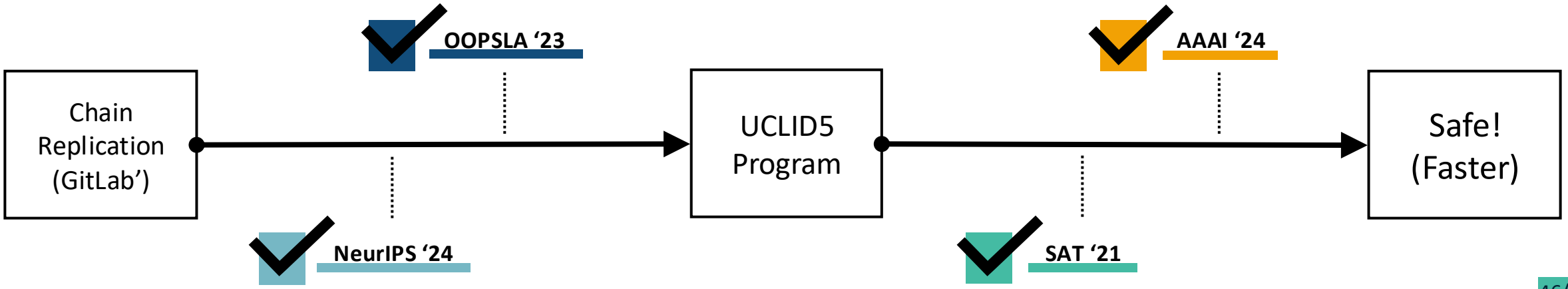
- 5 individual SMT solvers.
- 4 benchmark sets, each with 500 queries (2000 total).
- MedleySolver outperforms all individual solvers! (96% vs. 84% max)
- No training!

Contributions: Distributed Systems Solver Stack

Challenge: Successful users must be experts in their domain **and** automated reasoning (AR).

Solution: Reduce the need for automated reasoning expertise by tailoring every level of the AR stack.

- Avoid programming paradigm switch with domain-specific verification framework (No ML)
- Reduce formalization burden with neuro-symbolic automation (Yes ML)
- Reduce need for custom decision procedures with solver for user-defined algebraic data types (Need More ML)
- Automate choice for decision procedures and parameters with reinforcement learning (**Yes ML**)



Published on: February 10, 2017 19 min read

Postmortem of database outage of January 31

Postmortem on the database outage of January 31 2017 with the lessons we learned.

GitLab

ars TECHNICA SIGN IN

BRINGING NEW MEANING TO "KILLED BY GOOGLE"

"Unprecedented" Google Cloud event wipes out customer account and its backups

UniSuper, a $135 billion pension account, details its cloud compute nightmare.

GADGETS NOW Read ePaper

Microsoft confirms that it lost weeks of data for its Cloud customers: "A bug in one of Microsoft's..... resulted in malfunction"

TOI Tech Desk / TIMESOFINDIA.COM / Updated: Oct 19, 2024, 13:51 IST

December 5, 2022: AWS Outage in US-East 2 Availability Zone

In December 2022, a 40-minute outage hit AWS's US-East 2 region, the second outage for the zone in 2022. AWS published no incident summary for this outage, and a spokesperson told *Data Center Knowledge* the company does "not publish Post-Event Summaries ... for every service event."

June 13, 2023: Amazon's Cloud Computing Service Experiences Regional Outage

Distributed System

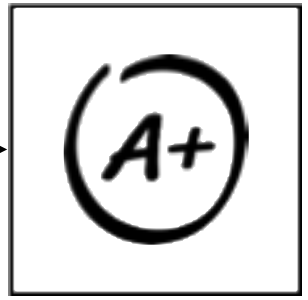
Despite extensive testing, fault and timing-related bugs account for 44% of production Microsoft Azure incidents.[2]

node, the backup node responds to the primary, and the primary responds to the client. Both nodes should have a memory that is updated with each request. The specification is that after every response sent to the client, the primary and backup memories should always be equal. "

```
} …

spec TwoCopies observes eResponse {
  on eResponse r do {
    assert primary.db == backup.db
  }
}
```

SMT Query



aws

Eudoxus — NeurIPS '24

PVerifier — OOPSLA '23

Medley + Algaroba — SAT '21, AAAI '24

Today: Why/Where to Combine ML with Solvers

- One domain form start to finish
  - Gives concrete examples
  - Presents wholistic view of solvers (the whole solver stack!)

Big and Important Reasoning Everywhere

Vision: Domain-specific automated reasoning.

Build up metatheory and frameworks to be able to quickly and easily apply domain-specific automated reasoning approach to new domains.

1. LM-generated bespoke solvers and solver components
   - e.g., some very exciting work at this workshop!
2. Getting LMs to use solvers in each domain
   - i.e., a more wholistic view of the solver stack

Thank you!!

Challenge: Successful users must be experts in their domain **and** automated reasoning (AR).

Solution: Reduce the need for automated reasoning expertise by tailoring every level of the AR stack.

- Avoid programming paradigm switch with domain-specific verification framework (**No ML**)
- Reduce formalization burden with neuro-symbolic automation (**Yes ML**)
- Reduce need for custom decision procedures with solver for user-defined algebraic data types (**Need More ML**)
- Automate choice for decision procedures and parameters with reinforcement learning (**Yes ML**)