A Decade of ATP Tuning

Machine Learning, Strategy Invention, and AI in the Future

Jan Jakubův

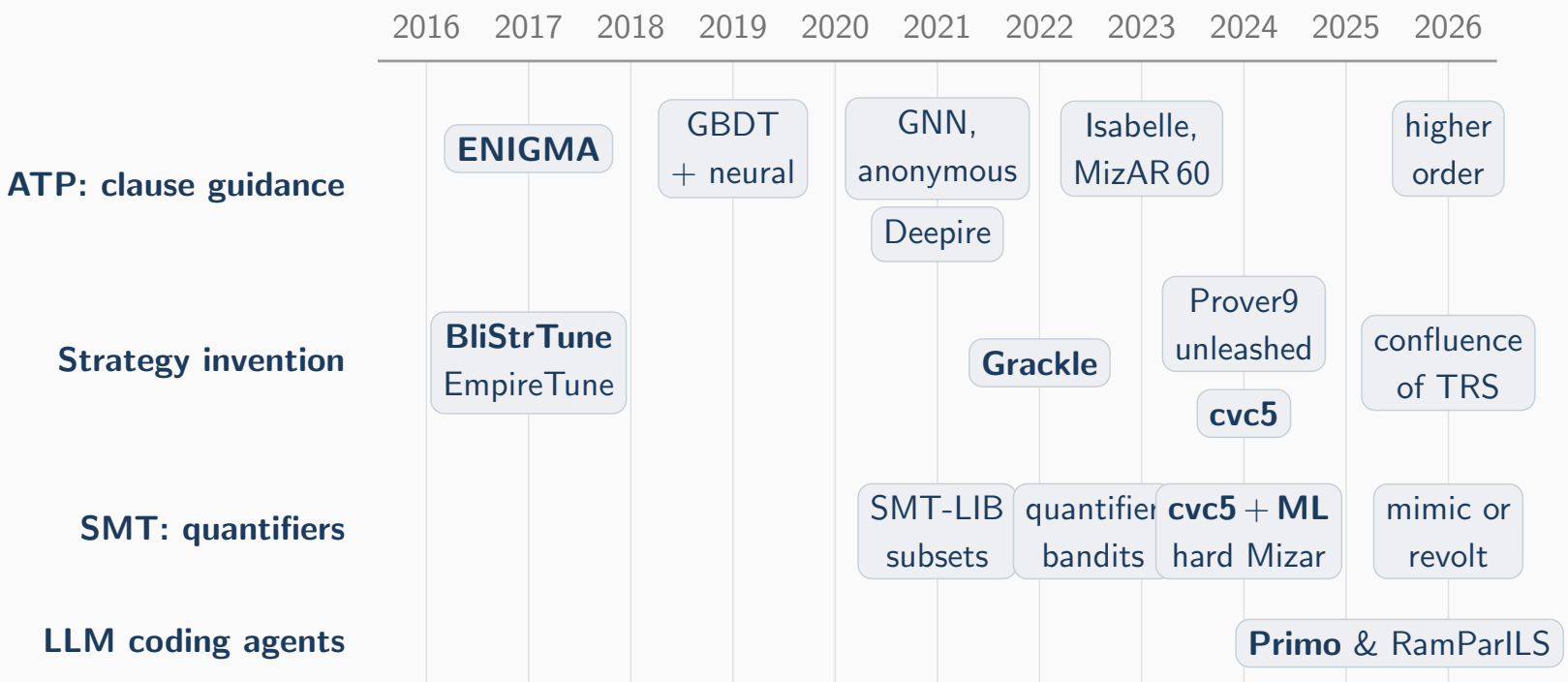
Czech Technical University in Prague, Czech Republic

AI4Math @ CICM 2026, Ljubljana, Slovenia, 21st September 2026

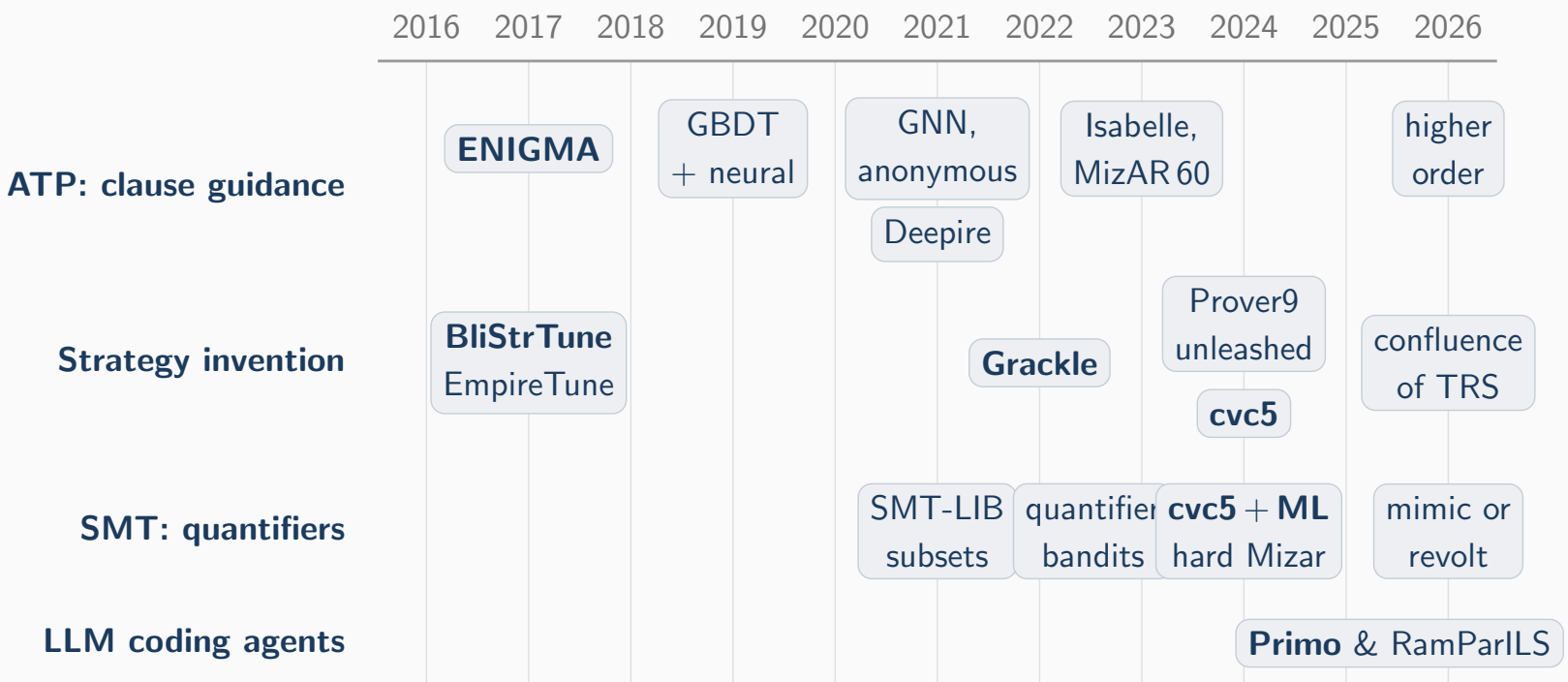
This talk is supported by Renaissance Philanthropy, project DEEPER, and the Czech Science Foundation, project no. 24-12759S.

Outline

Ten Years in One Picture



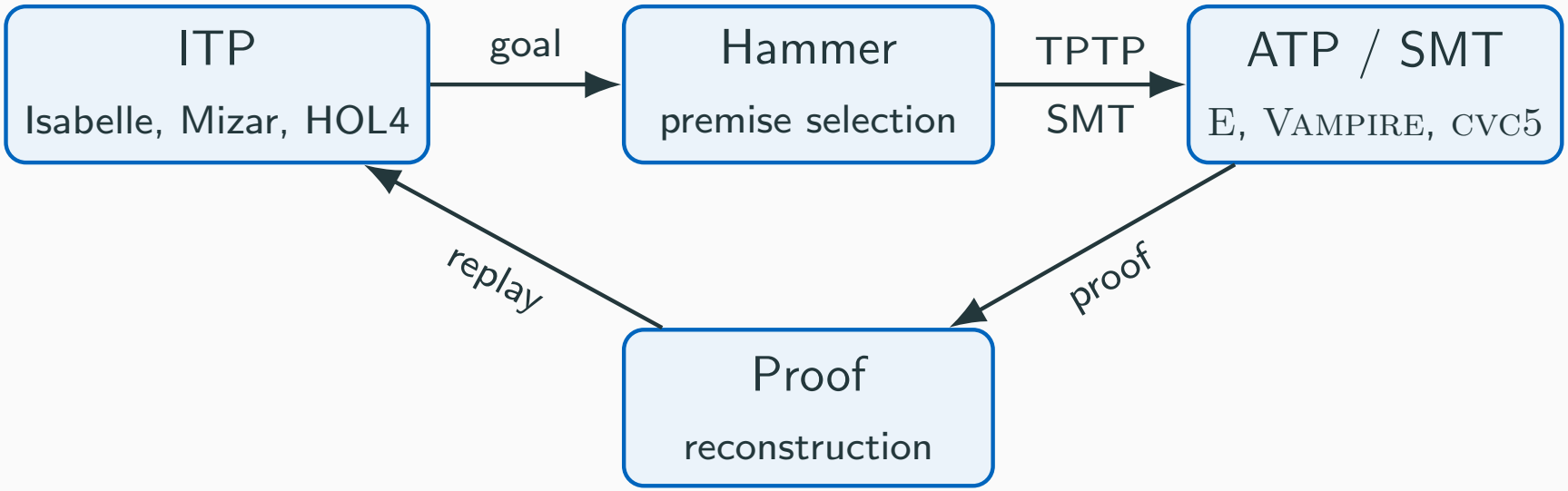
Ten Years in One Picture



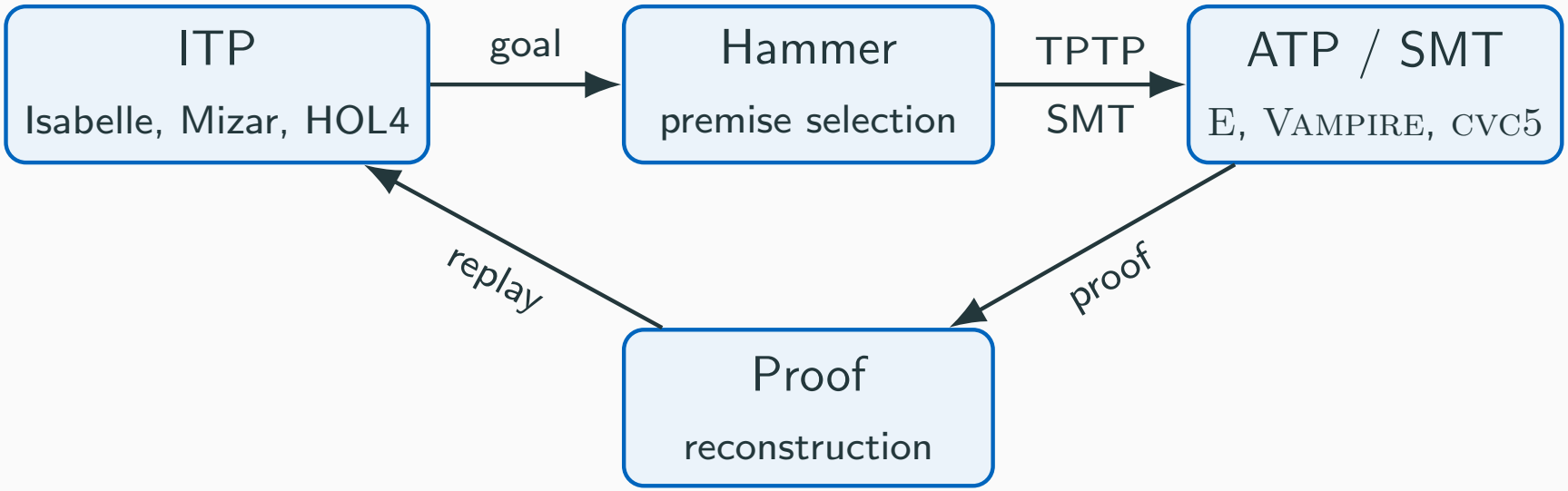
One question, four threads lesson

How do you make a reasoner good at *your* problems?

Where This All Happens



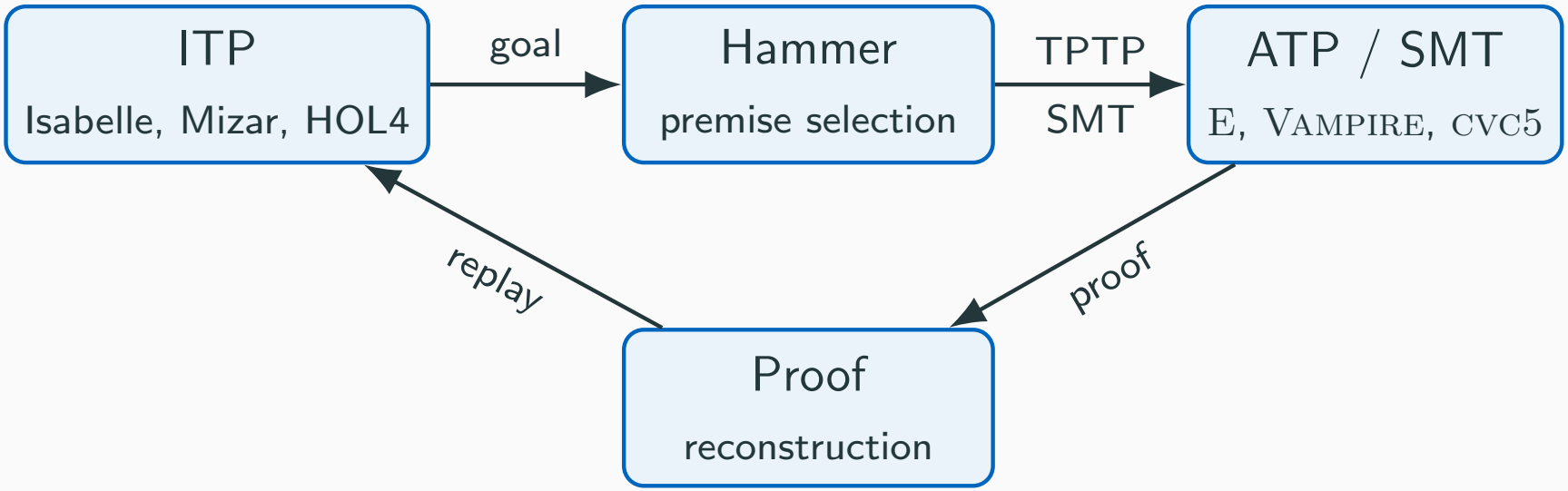
Where This All Happens



This talk is about the box on the right idea

Premise selection has been studied to death. We look *inside* the automated prover.

Where This All Happens



This talk is about the box on the right idea

Premise selection has been studied to death. We look *inside* the automated prover.

The prover is not one algorithm problem

It is a *space* of algorithms — and somebody has to pick a point in it.

Outline

Outline

A Modern Reasoner Is a Pile of Heuristics

A Modern Reasoner Is a Pile of Heuristics

Dozens of techniques, hundreds of options problem

Orderings, selection, simplification, indexing, splitting, instantiation, preprocessing . . . all parametrised — and the right setting needs an insider *and* depends on *your* problems.

A Modern Reasoner Is a Pile of Heuristics

Dozens of techniques, hundreds of options problem

Orderings, selection, simplification, indexing, splitting, instantiation, preprocessing . . . all parametrised — and the right setting needs an insider *and* depends on *your* problems.

So the typical user gets the default lesson

And tuning is often worth more than a new calculus.

What a Strategy Actually Looks Like

An E strategy, verbatim — you have 5 seconds:

```
--definitional-cnf=24 --destructive-er-aggressive --destructive-er
--prefer-initial-clauses -F1 --delete-bad-limit=150000000 --forward-context-sr
--simul-paramod -tKBO6 -winvfreqrank -c1 -Ginvfreq -WSelectMaxLComplexAvoidPosPred
-H'(4*ConjectureGeneralSymbolWeight(SimulateSOS,100,100,100,50,50,10,50,1.5,1.5,1),
3*ConjectureGeneralSymbolWeight(PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1),
1*Clauseweight(PreferProcessed,1,1,1),1*FIFOWeight(PreferProcessed))'
```

What a Strategy Actually Looks Like

An E strategy, verbatim — you have 5 seconds:

```
--definitional-cnf=24 --destructive-er-aggressive --destructive-er
--prefer-initial-clauses -F1 --delete-bad-limit=150000000 --forward-context-sr
--simul-paramod -tKBO6 -winvfreqrank -c1 -Ginvfreq -WSelectMaxLComplexAvoidPosPred
-H'(4*ConjectureGeneralSymbolWeight(SimulateSOS,100,100,100,50,50,10,50,1.5,1.5,1),
3*ConjectureGeneralSymbolWeight(PreferNonGoals,200,100,200,50,50,1,100,1.5,1.5,1),
1*Clauseweight(PreferProcessed,1,1,1),1*FIFOWeight(PreferProcessed))'
```

Nobody writes this by hand. Nobody reads it either. lesson

And yet the gap between this and the default can be tens of percent of solved problems.

How Big Is the Space?

Not your usual optimisation problem problem

- Discrete options, numeric parameters, *and* a term-structured part (clause weight functions, term orderings) of unbounded size.
- No gradients. One evaluation = a full proof search over a whole benchmark.
- The objective is a *step function*: "solved within T seconds".

How Big Is the Space?

Not your usual optimisation problem problem

- Discrete options, numeric parameters, *and* a term-structured part (clause weight functions, term orderings) of unbounded size.
- No gradients. One evaluation = a full proof search over a whole benchmark.
- The objective is a *step function*: "solved within T seconds".

Classical algorithm configuration — with teeth idea

ParamILS, SMAC and friends apply, but the objective is expensive, discrete and noisy.

How Big Is the Space?

Not your usual optimisation problem problem

- Discrete options, numeric parameters, *and* a term-structured part (clause weight functions, term orderings) of unbounded size.
- No gradients. One evaluation = a full proof search over a whole benchmark.
- The objective is a *step function*: "solved within T seconds".

Classical algorithm configuration — with teeth idea

ParamILS, SMAC and friends apply, but the objective is expensive, discrete and noisy.

And the target moves caveat

The best configuration depends on the benchmark. There may be no such thing as "the best settings".

Three Answers to the Same Question

Learn inside

guide each decision of the proof search

ENIGMA, Deepire

Search outside

invent whole strategies offline

BliStr, Grackle

Choose per problem

predict which strategy to run

scheduling, selection

Three Answers to the Same Question

Learn inside

guide each decision of the proof search

ENIGMA, Deepire

Search outside

invent whole strategies offline

BliStr, Grackle

Choose per problem

predict which strategy to run

scheduling, selection

Orthogonal, and they compose lesson

All three run on the same fuel: data from previous runs.

Three Answers to the Same Question

Learn inside

guide each decision of the proof search

ENIGMA, Deepire

Search outside

invent whole strategies offline

BliStr, Grackle

Choose per problem

predict which strategy to run

scheduling, selection

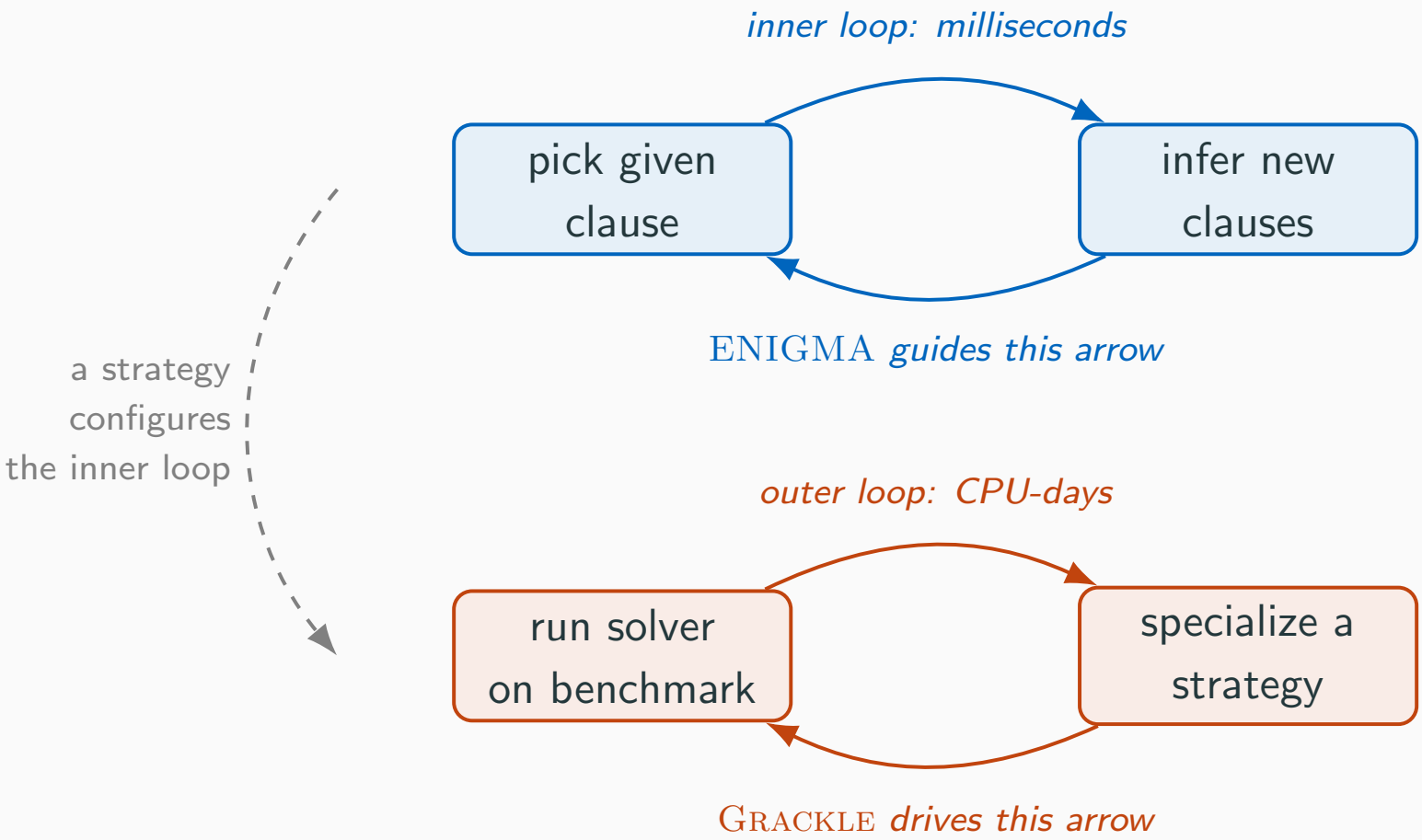
Orthogonal, and they compose lesson

All three run on the same fuel: data from previous runs.

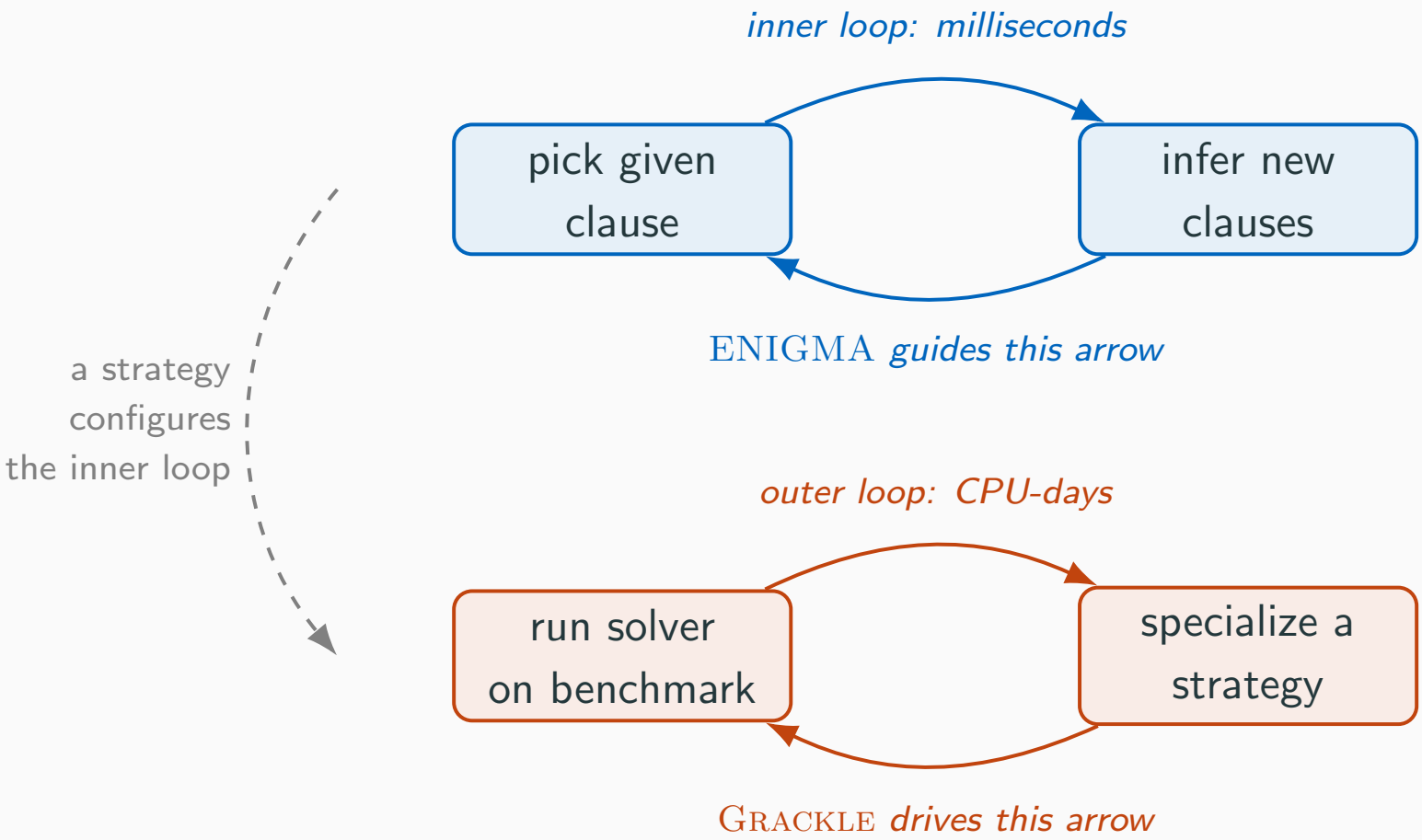
And they share one failure mode caveat

Overfitting to your benchmark.

The Two Loops



The Two Loops

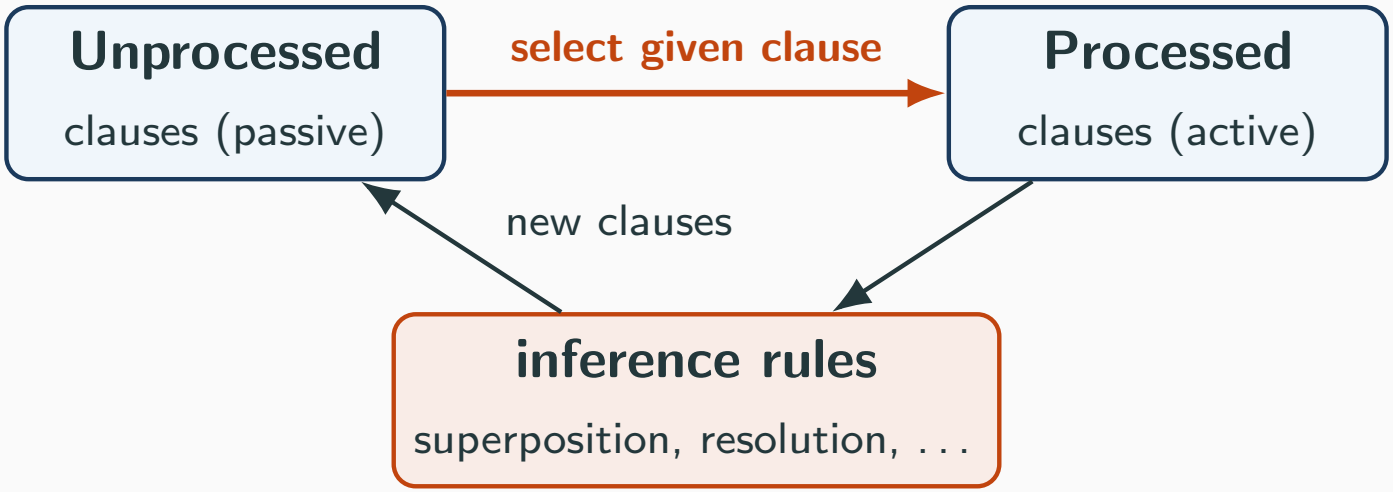


If you remember one slide from this talk lesson

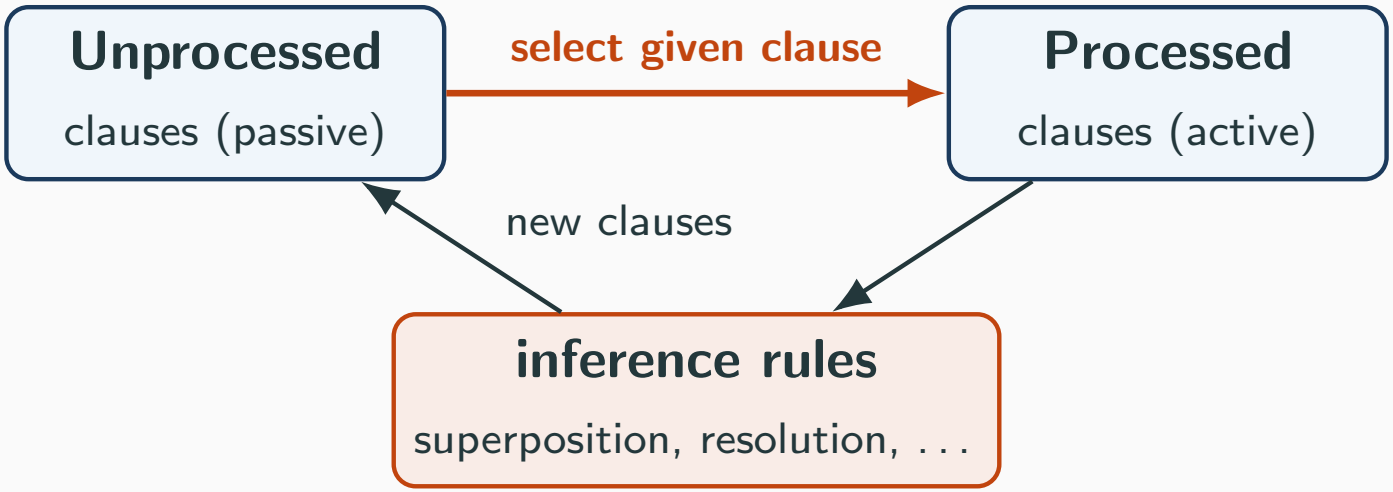
Make it this one. Everything that follows is one of these two loops.

Outline

The Given-Clause Loop



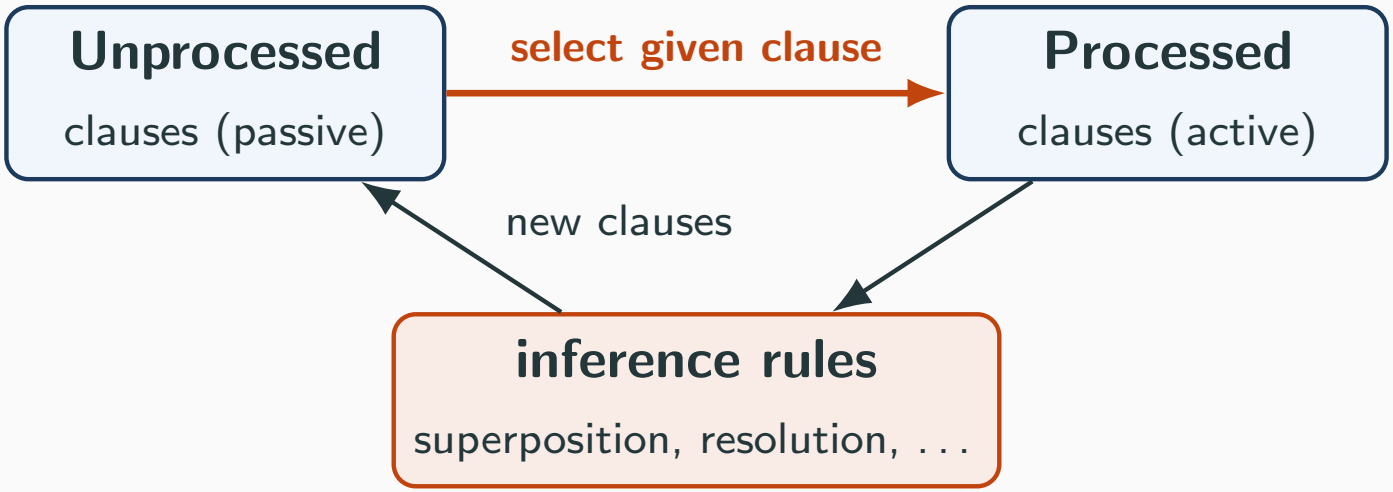
The Given-Clause Loop



One choice point, millions of times problem

Everything else is forced by the calculus. The prover just has to answer, over and over: which clause next?

The Given-Clause Loop



One choice point, millions of times problem

Everything else is forced by the calculus. The prover just has to answer, over and over: which clause next?

The standard answer caveat

A hand-tuned mix of "smallest" and "oldest".

Pre-History (2016): Similarity to the Conjecture

Prefer clauses related to the conjecture idea

Hand-crafted selection heuristics with several notions of "related": shared symbols, weighted symbol overlap, term structure. Implemented in E, evaluated on a large benchmark — and it worked.

Pre-History (2016): Similarity to the Conjecture

Prefer clauses related to the conjecture idea

Hand-crafted selection heuristics with several notions of "related": shared symbols, weighted symbol overlap, term structure. Implemented in E, evaluated on a large benchmark — and it worked.

But every heuristic was another human guess problem

And every guess added another option to tune.

Pre-History (2016): Similarity to the Conjecture

Prefer clauses related to the conjecture idea

Hand-crafted selection heuristics with several notions of "related": shared symbols, weighted symbol overlap, term structure. Implemented in E, evaluated on a large benchmark — and it worked.

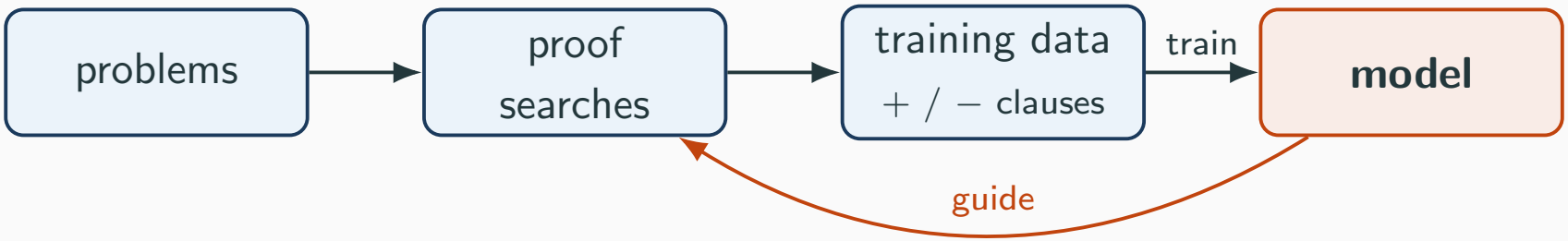
But every heuristic was another human guess problem

And every guess added another option to tune.

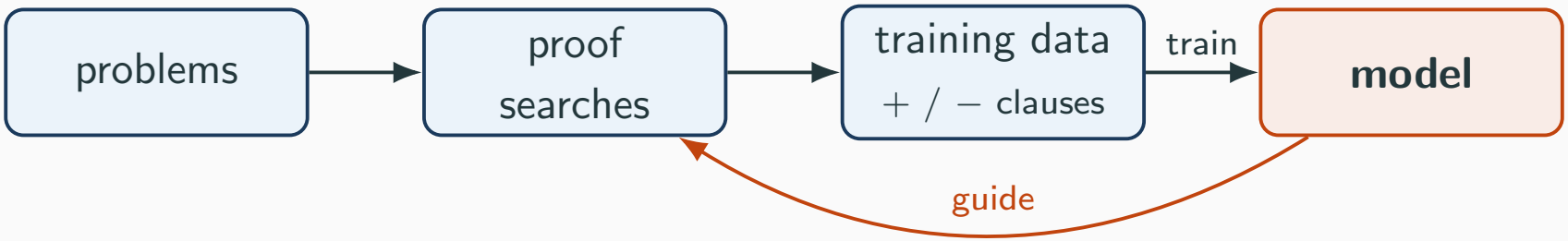
Why guess, when we have thousands of previous proofs?

[Jakubův, Urban. CICM 2016]

ENIGMA (2017): Learn It Instead



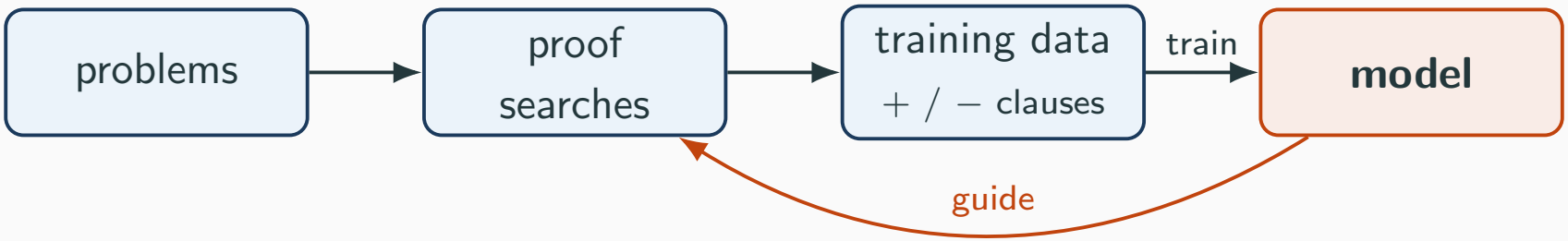
ENIGMA (2017): Learn It Instead



Supervised learning from finished proofs idea

- **Positive** = clauses that appear in the found proof.
- **Negative** = all other clauses that were processed.
- Train a classifier, plug it into E's clause evaluation.

ENIGMA (2017): Learn It Instead



Supervised learning from finished proofs idea

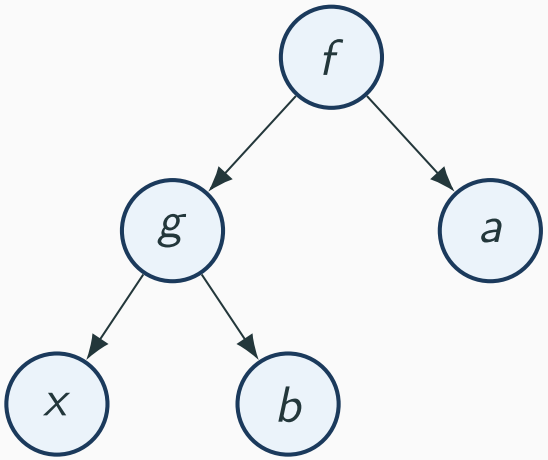
- **Positive** = clauses that appear in the found proof.
- **Negative** = all other clauses that were processed.
- Train a classifier, plug it into E's clause evaluation.

The first model was a linear classifier caveat

Chosen because it is fast, not because it is accurate.

[Jakubův, Urban. CICM 2017]

Clause Features: Making Logic Fit into a Vector

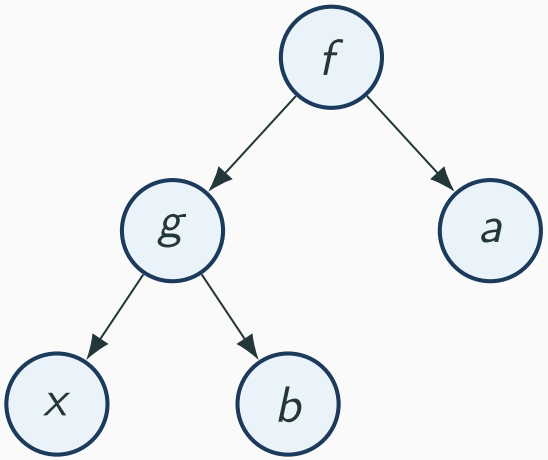


vertical (top-down paths):
(f, g, x), (f, g, b), (f, a)

horizontal (head + arg heads):
$f(g, a)$, $g(x, b)$

plus clause length, symbol counts,
and the same features of the *conjecture*

Clause Features: Making Logic Fit into a Vector



vertical (top-down paths):
(f, g, x), (f, g, b), (f, a)

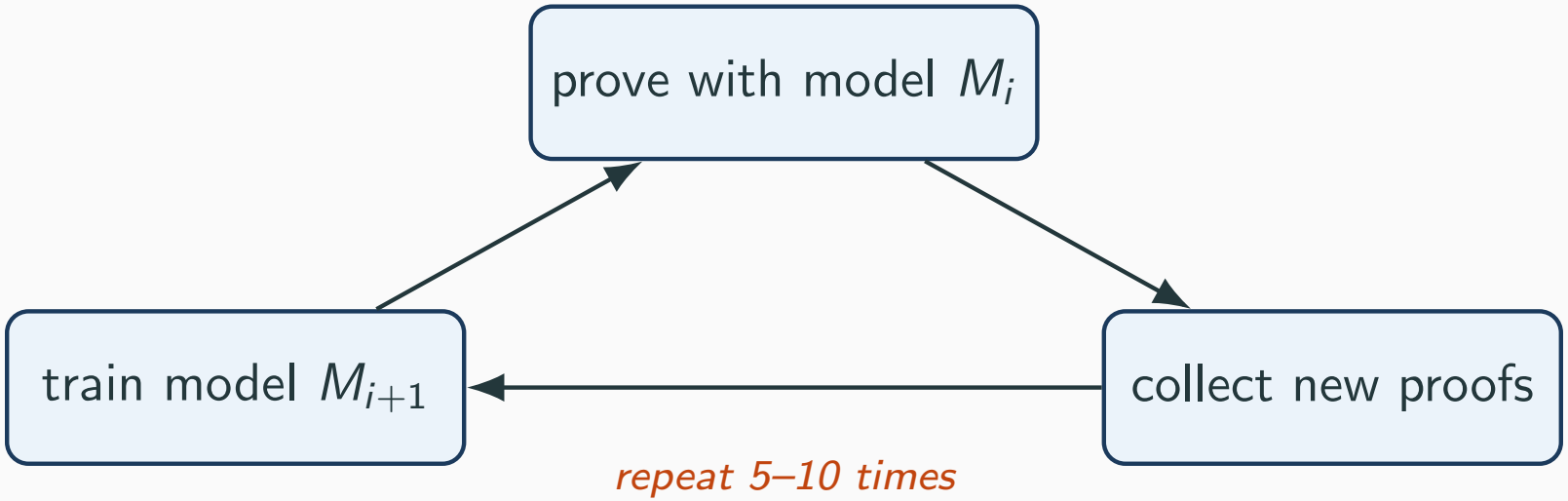
horizontal (head + arg heads):
$f(g, a)$, $g(x, b)$

plus clause length, symbol counts,
and the same features of the *conjecture*

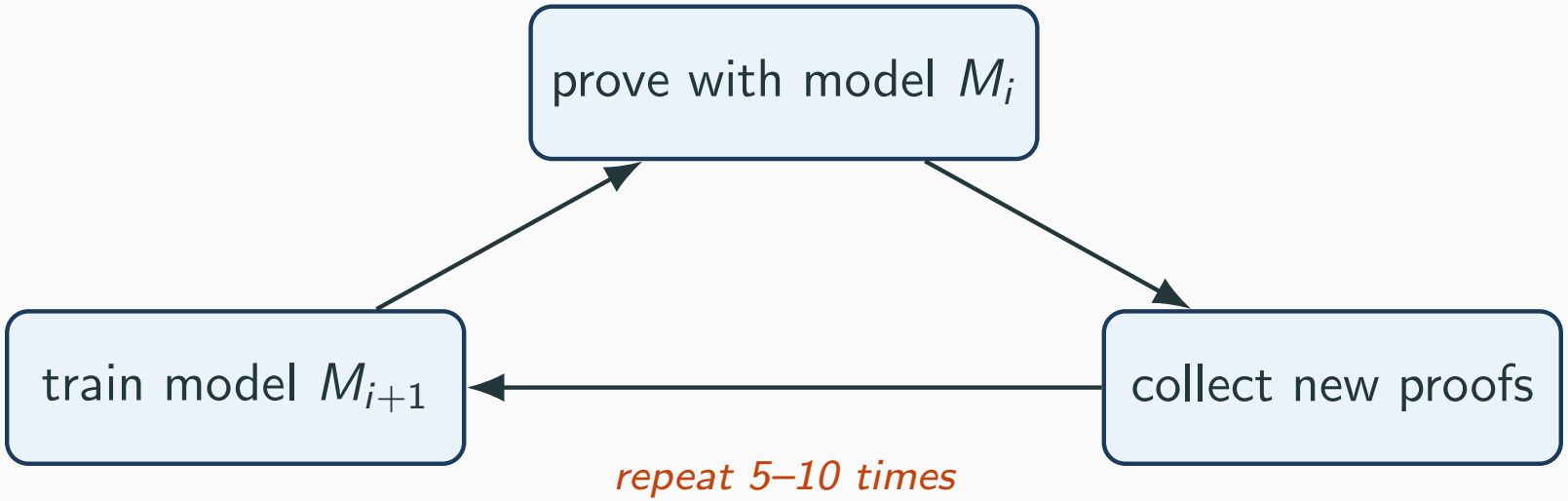
Extraction must cost microseconds problem

It runs on *every* generated clause. Features are hashed into a fixed-size sparse vector.

The Feedback Loop



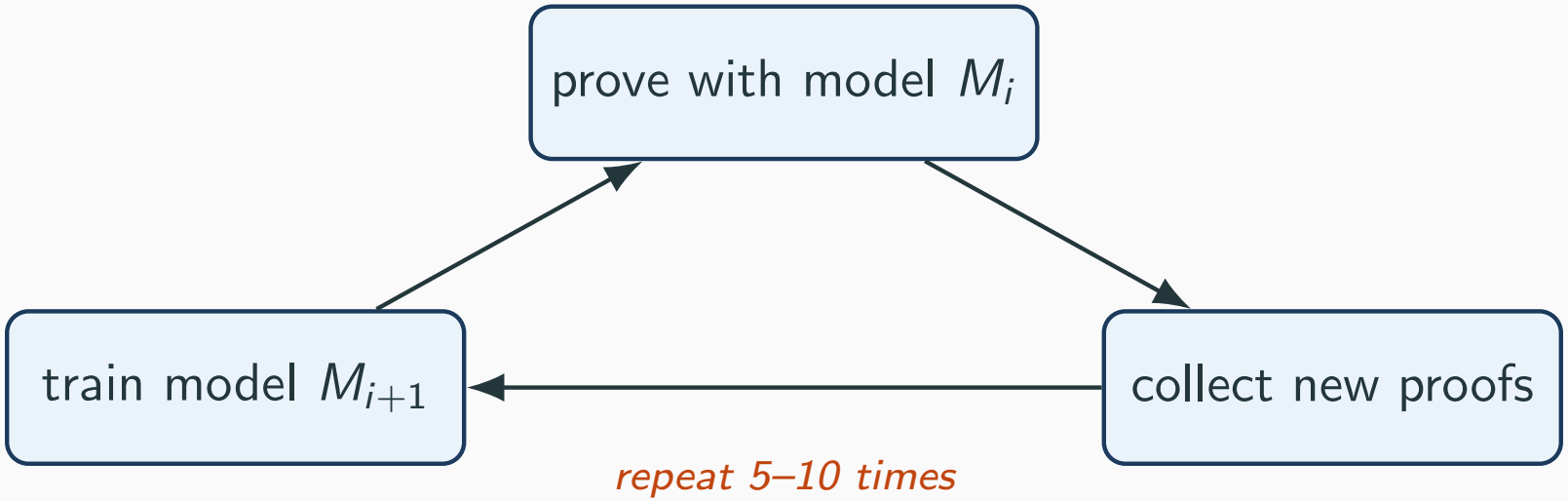
The Feedback Loop



Each round generates its own next training set idea

Problems solved for the first time become new positive data. Gains are largest in the first 2–3 rounds, then flatten.

The Feedback Loop



Each round generates its own next training set idea

Problems solved for the first time become new positive data. Gains are largest in the first 2–3 rounds, then flatten.

This loop reappears everywhere in this talk lesson

The single most reusable idea we have.

[Jakubův, Urban. CICM 2018]

ENIGMA-NG (2019): Better Models, Same Time Budget

Replace the linear classifier idea

- **Gradient-boosted decision trees** (XGBoost, later LightGBM): much more accurate, still fast enough. The workhorse to this day.
- **Recursive neural networks**: more accurate still — and far too slow, naively.

ENIGMA-NG (2019): Better Models, Same Time Budget

Replace the linear classifier idea

- **Gradient-boosted decision trees** (XGBoost, later LightGBM): much more accurate, still fast enough. The workhorse to this day.
- **Recursive neural networks**: more accurate still — and far too slow, naively.

Making the neural version affordable idea

Deep integration with E's data structures: cache term embeddings across clauses, amortize evaluation.

ENIGMA-NG (2019): Better Models, Same Time Budget

Replace the linear classifier idea

- **Gradient-boosted decision trees** (XGBoost, later LightGBM): much more accurate, still fast enough. The workhorse to this day.
- **Recursive neural networks**: more accurate still — and far too slow, naively.

Making the neural version affordable idea

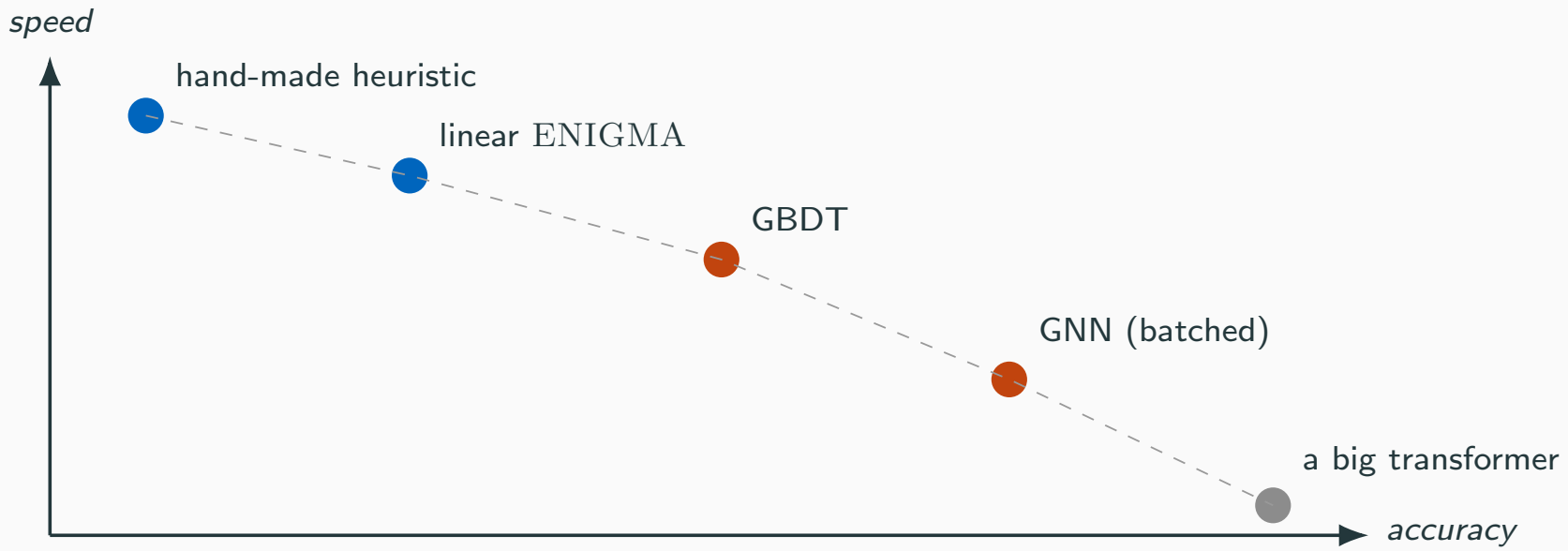
Deep integration with E's data structures: cache term embeddings across clauses, amortize evaluation.

First practically convincing neural clause guidance result

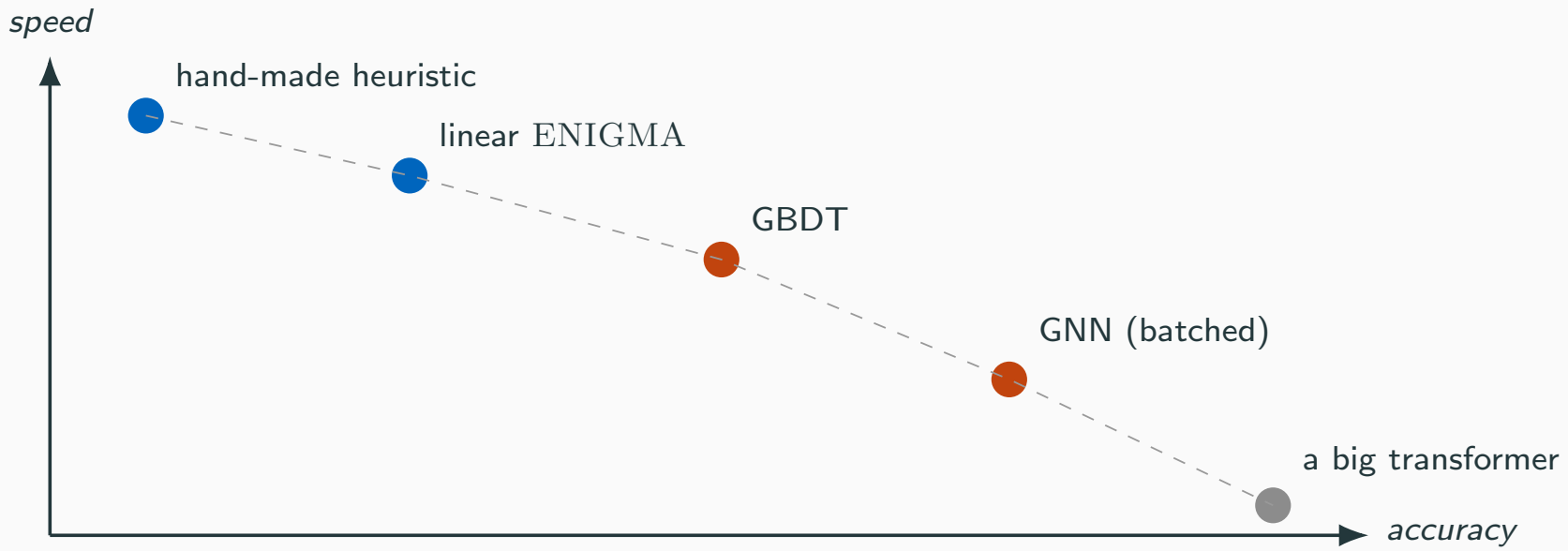
In a saturation-style prover.

[Chvalovský, Jakubův, Suda, Urban. CADE 2019]

The Tyranny of the Inner Loop



The Tyranny of the Inner Loop



A model 10× better and 100× slower loses lesson

Much of our work is really about moving up and right on this curve: engineering, not just learning.

ENIGMA Anonymous (2020): Forget the Names

Symbol names do not transfer problem

A model trained on Mizar knows nothing about Isabelle — and breaks even inside one library when definitions are renamed.

ENIGMA Anonymous (2020): Forget the Names

Symbol names do not transfer problem

A model trained on Mizar knows nothing about Isabelle — and breaks even inside one library when definitions are renamed.

Two symbol-independent answers idea

- **Arity-based abstraction** of symbols for the GBDT features.
- **Graph neural networks** embedding terms and clauses structurally — and scoring a *batch* of candidates against the clauses already selected, rather than each clause in isolation.

ENIGMA Anonymous (2020): Forget the Names

Symbol names do not transfer problem

A model trained on Mizar knows nothing about Isabelle — and breaks even inside one library when definitions are renamed.

Two symbol-independent answers idea

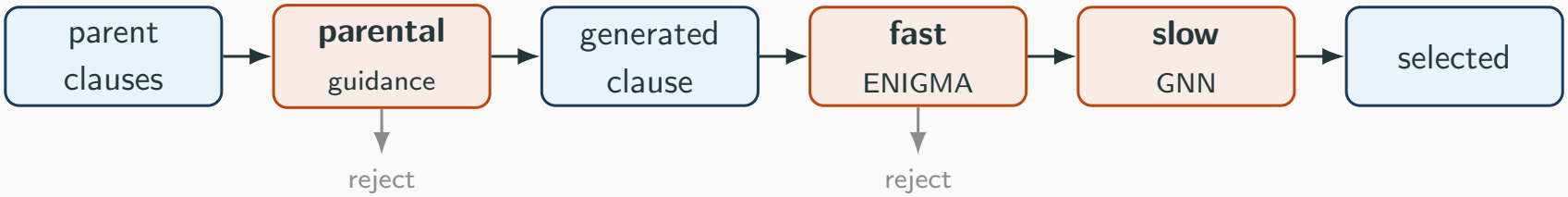
- **Arity-based abstraction** of symbols for the GBDT features.
- **Graph neural networks** embedding terms and clauses structurally — and scoring a *batch* of candidates against the clauses already selected, rather than each clause in isolation.

The GNN learns what a symbol does lesson

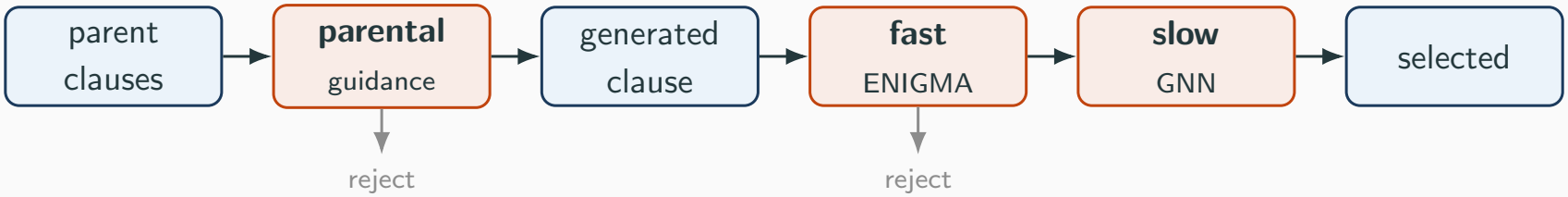
From how it is used. No name required.

[Jakubův et al. IJCAR 2020 • Chvalovský, Jakubův, Olšák, Urban. TABLEAUX 2021]

Fast and Slow — and Parental Guidance (2021)



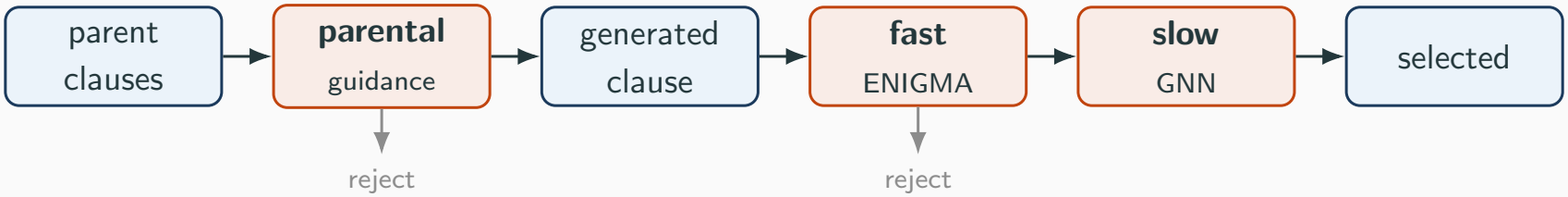
Fast and Slow — and Parental Guidance (2021)



Two cheap tricks that both pay idea

- **Fast & slow**: a cheap model pre-filters for an expensive one — beats both fast-only and slow-only.
- **Parental guidance**: judge the *parents*, reject the inference before the clause exists.

Fast and Slow — and Parental Guidance (2021)



Two cheap tricks that both pay idea

- **Fast & slow**: a cheap model pre-filters for an expensive one — beats both fast-only and slow-only.
- **Parental guidance**: judge the *parents*, reject the inference before the clause exists.

"There is always a cost to producing all possible offspring."

[Goertzel, Chvalovský, Jakubův, Olšák, Urban. FroCoS 2021]

Does It Actually Work? Mizar

$+70\,\%$

Real-time improvement of E over the whole Mizar library result

Same prover, same time budget, learned clause selection — a single strategy, not a portfolio.

Does It Actually Work? Mizar

$+70\,\%$

Real-time improvement of E over the whole Mizar library result

Same prover, same time budget, learned clause selection — a single strategy, not a portfolio.

MizAR 60 for Mizar 50 result

The hammer built on top of this reaches roughly $75\,\%$ of the Mizar library.

Does It Actually Work? Mizar

$+70\,\%$

Real-time improvement of E over the whole Mizar library result

Same prover, same time budget, learned clause selection — a single strategy, not a portfolio.

MizAR 60 for Mizar 50 result

The hammer built on top of this reaches roughly 75 % of the Mizar library.

Meanwhile, in Vampire: Deepire idea

Martin Suda's neural guidance judges a clause by its *derivation history* alone — where it came from, not what it says. Cheap to evaluate, and it proves many Mizar theorems ENIGMA does not.

[Jakubův, Urban. ITP 2019 • Jakubův et al. ITP 2023 • Suda. FroCoS 2021]

Does It Actually Work? Isabelle

$+25.3\,\%$

Over the best previous E on Sledgehammer problems, at 15 s result

Outperformed all other previous ATP and SMT systems on this benchmark.

Does It Actually Work? Isabelle

$+25.3\,\%$

Over the best previous E on Sledgehammer problems, at 15 s result

Outperformed all other previous ATP and SMT systems on this benchmark.

Nobody wins alone lesson

Three targeted ingredients together: ENIGMA guidance + neural premise selection + invented E strategies.

[Goertzel, Jakubův, Kaliszyk, Olšák, Piepenbrock, Urban. ITP 2022]

2026: Going Higher-Order

Isabelle is higher-order; we threw that away problem

Until now everything went through a first-order encoding.

2026: Going Higher-Order

Isabelle is higher-order; we threw that away problem

Until now everything went through a first-order encoding.

Native HO guidance idea

ENIGMA features for λ-terms and partial applications in λE; Deepire's GNN extended to *polymorphic* HO. 246,277 problems exported from Isabelle in three TPTP dialects.

2026: Going Higher-Order

Isabelle is higher-order; we threw that away problem

Until now everything went through a first-order encoding.

Native HO guidance idea

ENIGMA features for λ-terms and partial applications in λE; Deepire's GNN extended to *polymorphic* HO. 246,277 problems exported from Isabelle in three TPTP dialects.

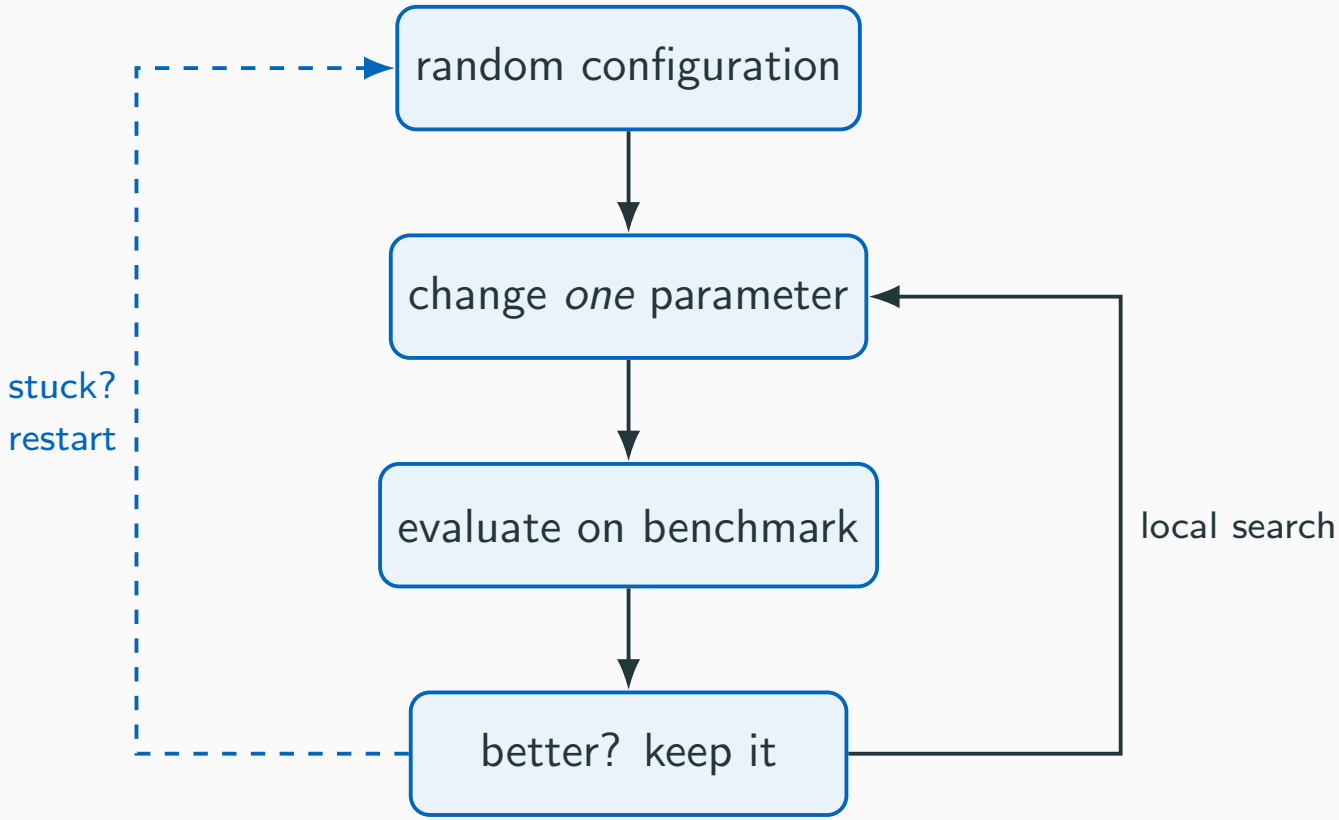
The gains hold up natively — and are complementary result

λENIGMA on TH0: 48.9 → 53.4 %. Deepire-HO on TH1: 31.8 → 40.8 % (+28 % rel.). HO strategies add +4.2 points over the best FO-only combination.

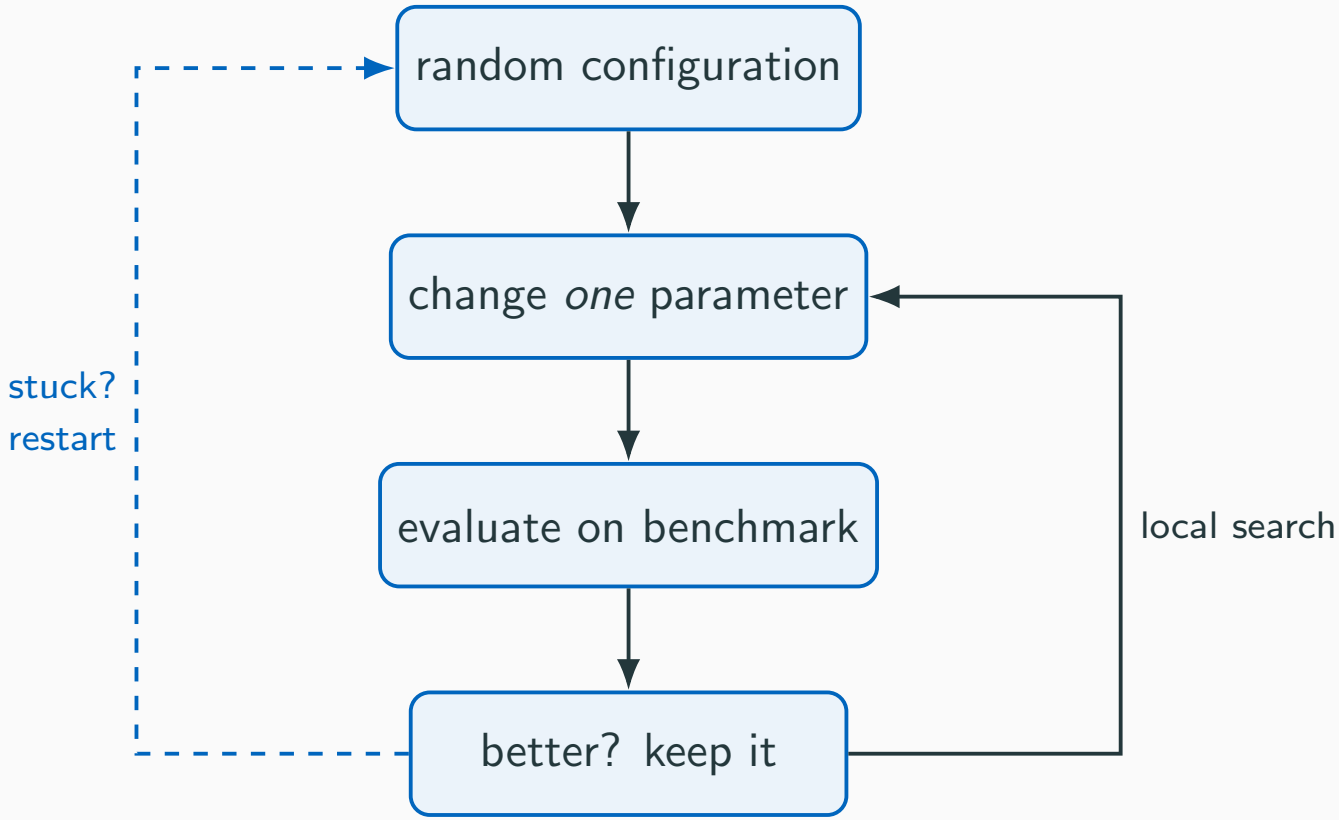
[Jakubův, Kaliszyk, Suda. AITP / CICM 2026 — see the main track]

Outline

Algorithm Configuration in One Slide



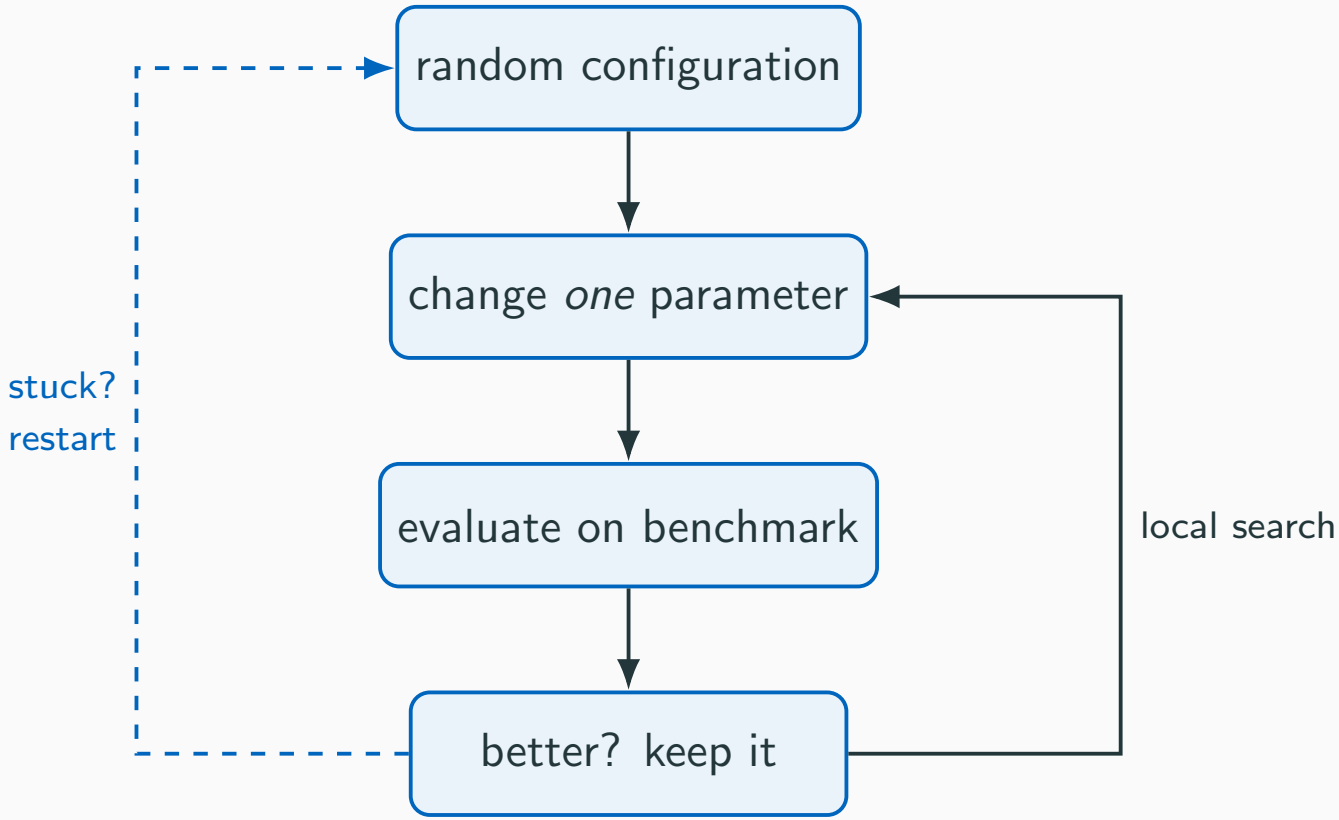
Algorithm Configuration in One Slide



Iterated local search — the core of ParamILS idea

Cheap, dumb, embarrassingly parallel, and surprisingly effective.

Algorithm Configuration in One Slide



Iterated local search — the core of ParamILS idea

Cheap, dumb, embarrassingly parallel, and surprisingly effective.

But it gives you one configuration

One Champion Is the Wrong Goal

Benchmarks are heterogeneous problem

A configuration that is best on average is best at *nothing in particular*.

One Champion Is the Wrong Goal

Benchmarks are heterogeneous problem

A configuration that is best on average is best at *nothing in particular*.

What a prover ships is a portfolio idea

Several strategies, run in sequence or in parallel. What matters is not individual strength but complementarity.

One Champion Is the Wrong Goal

Benchmarks are heterogeneous problem

A configuration that is best on average is best at *nothing in particular*.

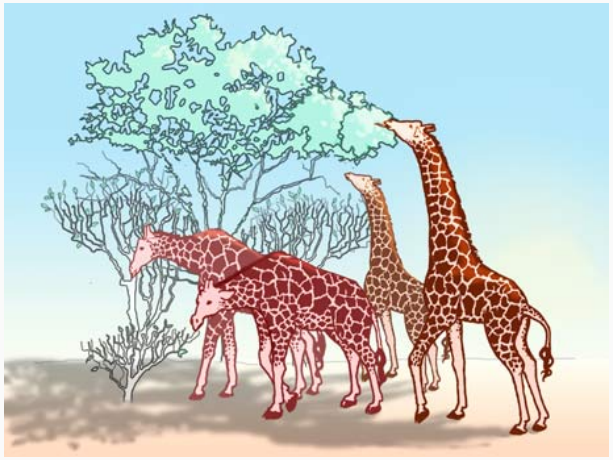
What a prover ships is a portfolio idea

Several strategies, run in sequence or in parallel. What matters is not individual strength but complementarity.

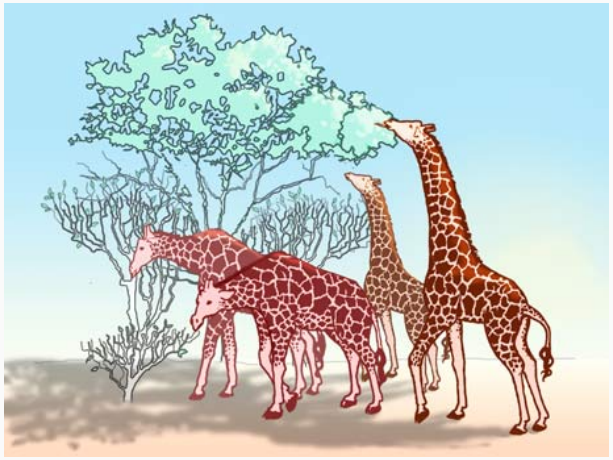
A weak strategy solving 30 unique problems. . . lesson

. . . beats a strong one solving 300 already-covered ones. So: invent a *set* of strategies, together.

BliStr's Idea: Specialize What Is Already Good



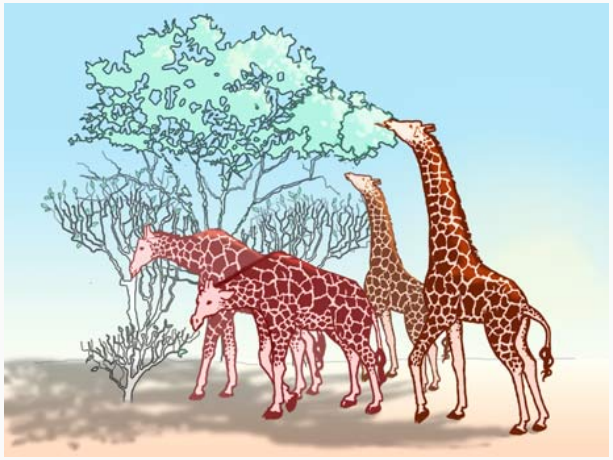
BliStr's Idea: Specialize What Is Already Good



Let each strategy stretch towards its own niche idea

Take a strategy, look at the problems it is *already* best at, and tune it only on those. The niches then partition the benchmark.

BliStr's Idea: Specialize What Is Already Good



Let each strategy stretch towards its own niche idea

Take a strategy, look at the problems it is *already* best at, and tune it only on those. The niches then partition the benchmark.

Evolution — but Lamarckian lesson

The acquired improvement is inherited directly.

BliStrTune (2017): Mind the Hierarchy

BliStr treated the clause weight function as a blob problem

Which is exactly where most of the power lives:

```
Clauseweight(PreferProcessed,1,1,1)
```

BliStrTune (2017): Mind the Hierarchy

BliStr treated the clause weight function as a blob problem

Which is exactly where most of the power lives:

```
Clauseweight(PreferProcessed,1,1,1)
```

Interleave two levels of search idea

Pick the *shape* of the strategy (which weight functions, how many), fine-tune their numeric parameters, then go back up.

BliStrTune (2017): Mind the Hierarchy

BliStr treated the clause weight function as a blob problem

Which is exactly where most of the power lives:

```
Clauseweight(PreferProcessed,1,1,1)
```

Interleave two levels of search idea

Pick the *shape* of the strategy (which weight functions, how many), fine-tune their numeric parameters, then go back up.

A far larger space, and significant gains on Mizar/MML result

With new conjecture-aware weight functions targeted at ITP libraries.

[Jakubův, Urban. CPP 2017 / AI Communications 2018]

EmpireTune (2017): Vampire, and Inventing Orderings

Do not marry the tuner to one prover idea

Added Vampire support alongside E.

EmpireTune (2017): Vampire, and Inventing Orderings

Do not marry the tuner to one prover idea

Added Vampire support alongside E.

Invent the term ordering itself idea

Symbol precedences and KBO weights become tunable parameters — which required extending Vampire with a way to specify them. Term orderings sit deep in the calculus: the kind of thing only an implementer would touch. Very good results on the AIM (loop theory) benchmark.

EmpireTune (2017): Vampire, and Inventing Orderings

Do not marry the tuner to one prover idea

Added Vampire support alongside E.

Invent the term ordering itself idea

Symbol precedences and KBO weights become tunable parameters — which required extending Vampire with a way to specify them. Term orderings sit deep in the calculus: the kind of thing only an implementer would touch. Very good results on the AIM (loop theory) benchmark.

Side quest: weighted path orderings caveat

A richer, relaxed ordering space, designed to give the tuner more room.

[Jakubův, Suda, Urban. GCAI 2017 • Jakubův, Kaliszyk. ICMS 2018]

Grackle (2022): The Generic Successor

Solver-agnostic strategy invention idea

You describe how to *run* the solver, the *parameter space*, and the benchmark. GRACKLE does the rest.

Grackle (2022): The Generic Successor

Solver-agnostic strategy invention idea

You describe how to *run* the solver, the *parameter space*, and the benchmark. GRACKLE does the rest.

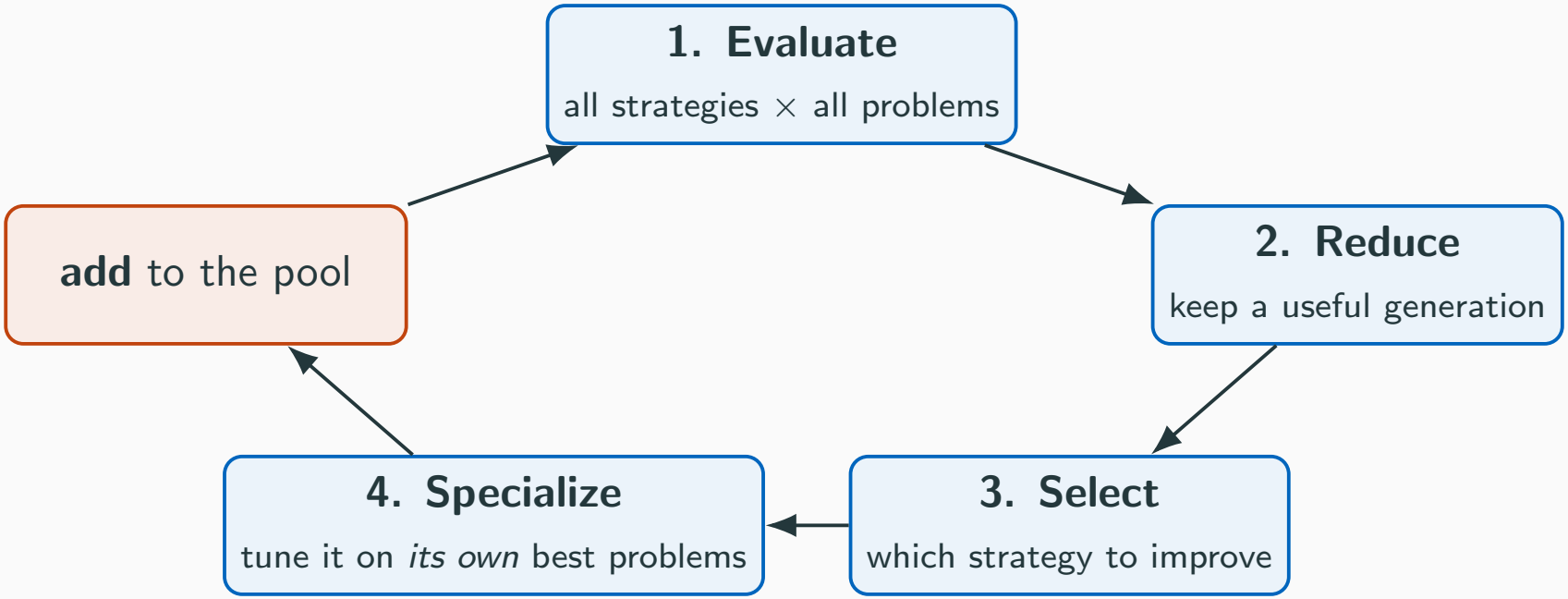
Pluggable external tuner idea

The actual configuration search is delegated to PARAMILS, SMAC3, ... Parallel, resumable, explicitly complementarity-driven.

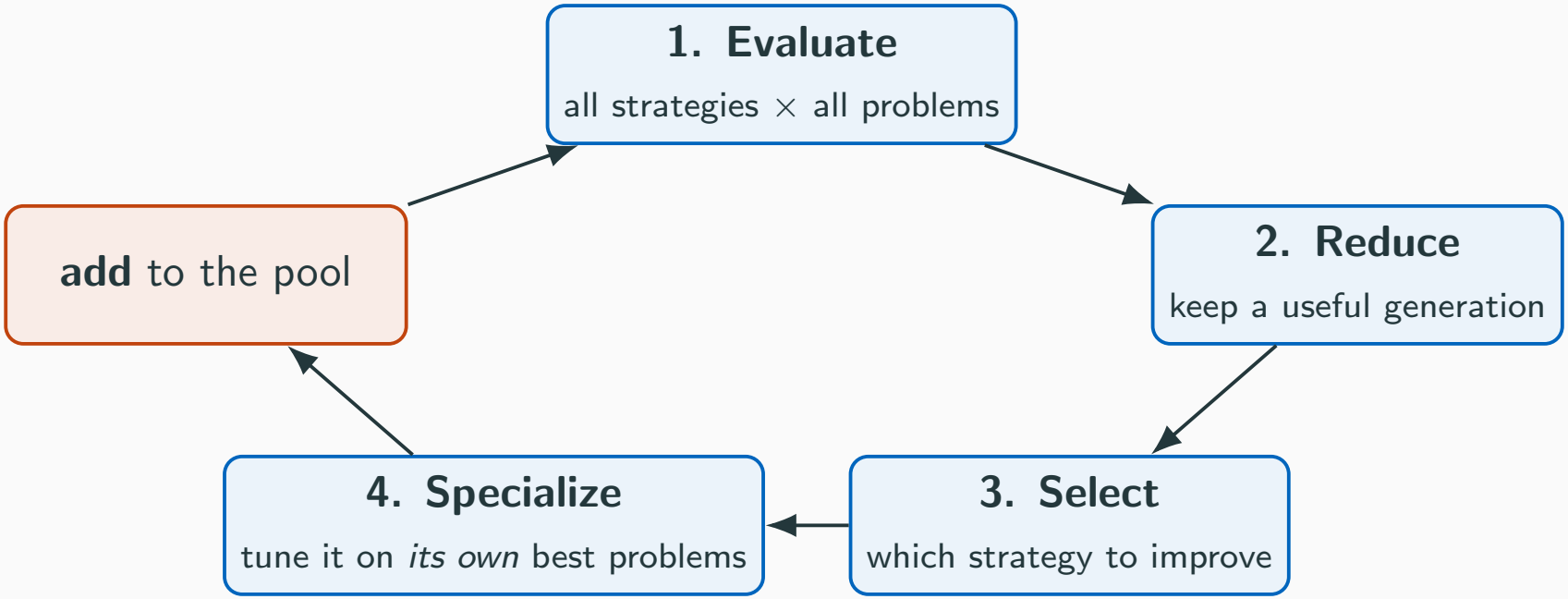
A grackle: clever, adaptable, and known for stealing other birds' food.

[Hůla, Jakubův, Janota, Kubej. CICM 2022]

The Grackle Loop



The Grackle Loop



It terminates — in theory caveat

No strategy is specialized on the same problems twice, so the loop must end. In practice you stop it with a wall-clock budget.

Grackle on Bitwuzla: Numbers

The setup idea

3,000 SMT-COMP problems, including all 342 that BITWUZLA failed to solve in the competition. 1 second per problem while tuning (the competition used 1200).

Grackle on Bitwuzla: Numbers

The setup idea

3,000 SMT-COMP problems, including all 342 that Bitwuzla failed to solve in the competition. 1 second per problem while tuning (the competition used 1200).

Solved result

20 initial (hand-made) strategies	1,345
Grackle, three tuner runs	1,481
Grackle, six runs	1,500

Plus a substantial *drop in runtime* on problems already solved.

Grackle on Bitwuzla: Numbers

The setup idea

3,000 SMT-COMP problems, including all 342 that Bitwuzla failed to solve in the competition. 1 second per problem while tuning (the competition used 1200).

Solved result

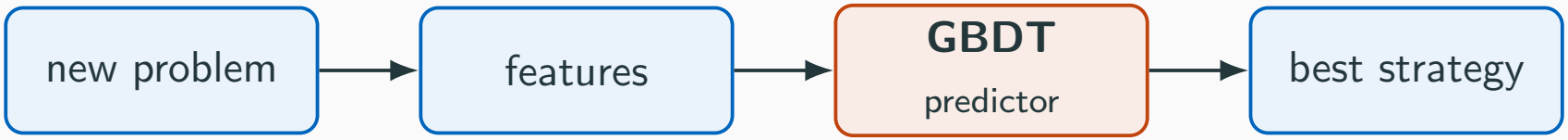
20 initial (hand-made) strategies	1,345
Grackle, three tuner runs	1,481
Grackle, six runs	1,500

Plus a substantial *drop in runtime* on problems already solved.

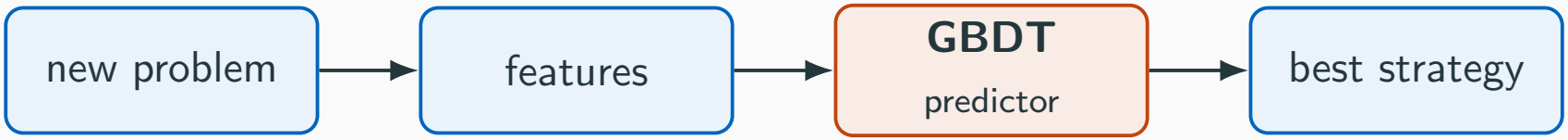
The loop matters more than the tuner lesson

ParamILS and SMAC3 end up at similar totals.

Then: Pick the Right One per Problem



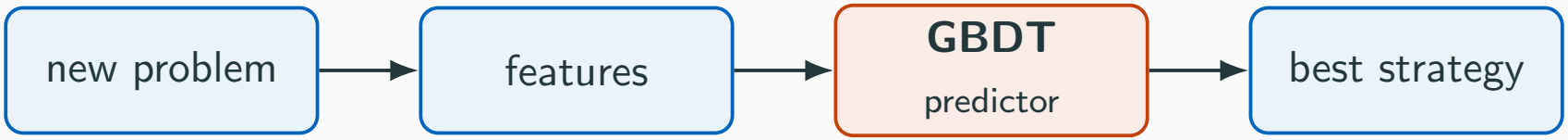
Then: Pick the Right One per Problem



The portfolio is also a labelled dataset idea

For each problem: which strategy solved it fastest? Train a model to predict that from problem features — turning a portfolio into a scheduler.

Then: Pick the Right One per Problem



The portfolio is also a labelled dataset idea

For each problem: which strategy solved it fastest? Train a model to predict that from problem features — turning a portfolio into a scheduler.

Much of the virtual-best performance at single-strategy cost

[Hůla, Jakubův, Janota, Kubej. CICM 2022]

Prover9 Unleashed (2024): An Old Prover, New Tricks

Prover9 never got the decade of schedule tuning problem

Its `auto` mode is weak — yet it offers more varied proof control than its competitors. The knobs were there; nobody turned them.

Prover9 Unleashed (2024): An Old Prover, New Tricks

Prover9 never got the decade of schedule tuning problem

Its `auto` mode is weak — yet it offers more varied proof control than its competitors. The knobs were there; nobody turned them.

Grackle + Prover9 result

	AIM	*TPTP/NUM*
Prover9 `auto` mode	41	512
+ Grackle portfolio	91	619

Prover9 Unleashed (2024): An Old Prover, New Tricks

Prover9 never got the decade of schedule tuning problem

Its `auto` mode is weak — yet it offers more varied proof control than its competitors. The knobs were there; nobody turned them.

Grackle + Prover9 result

	AIM	*TPTP/NUM*
Prover9 `auto` mode	41	512
+ Grackle portfolio	91	619

The tuned Prover9 beats E and Vampire on AIM lesson

And solves everything they solve. On TPTP/NUM it is highly complementary: together they reach 770.

[Aleksandrova, Jakubův, Kaliszyk. LPAR 2024]

What Invented Strategies Look Like

Ugly, unexplainable, brittle caveat

Nobody would write them. We usually cannot say *why* a flag combination wins. Rebuild the solver, re-tune.

What Invented Strategies Look Like

Ugly, unexplainable, brittle caveat

Nobody would write them. We usually cannot say *why* a flag combination wins. Rebuild the solver, re-tune.

Occasionally interpretable afterwards result

In Primo, one option (`monotone-elimination`) accounted for 62 of 77 new solutions — all from *one problem family*.

What Invented Strategies Look Like

Ugly, unexplainable, brittle caveat

Nobody would write them. We usually cannot say *why* a flag combination wins. Rebuild the solver, re-tune.

Occasionally interpretable afterwards result

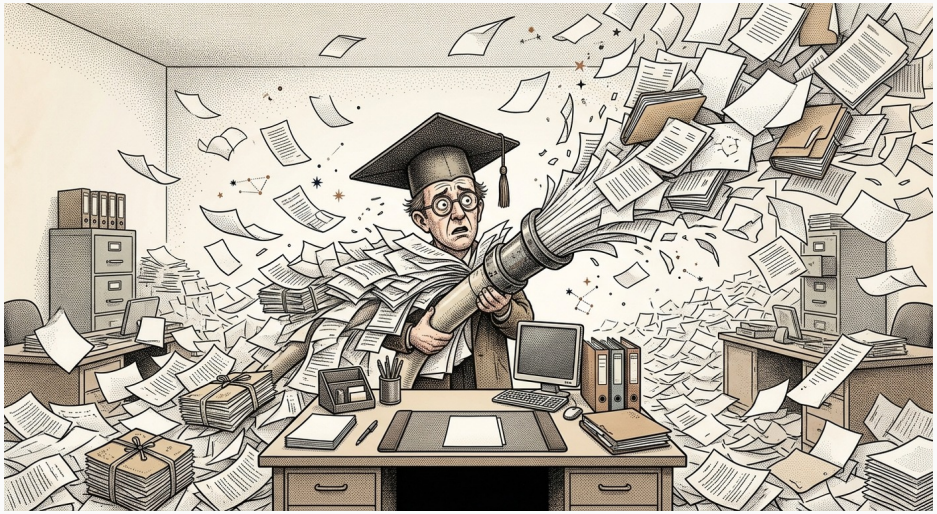
In Primo, one option (`monotone-elimination`) accounted for 62 of 77 new solutions — all from *one problem family*.

And they are better than ours lesson

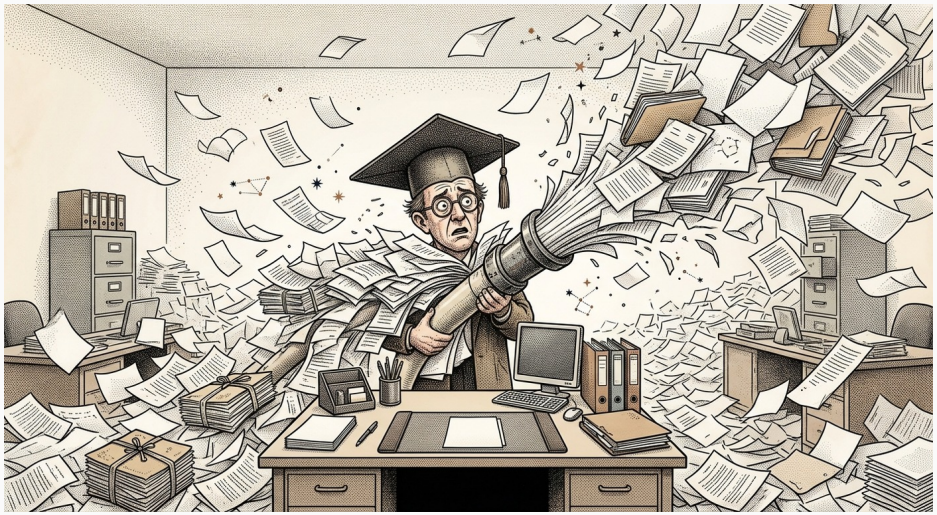
Consistently, for ten years, across seven different solvers. *This is the most reliable finding in the whole talk.*

Outline

SMT Is a Different Animal



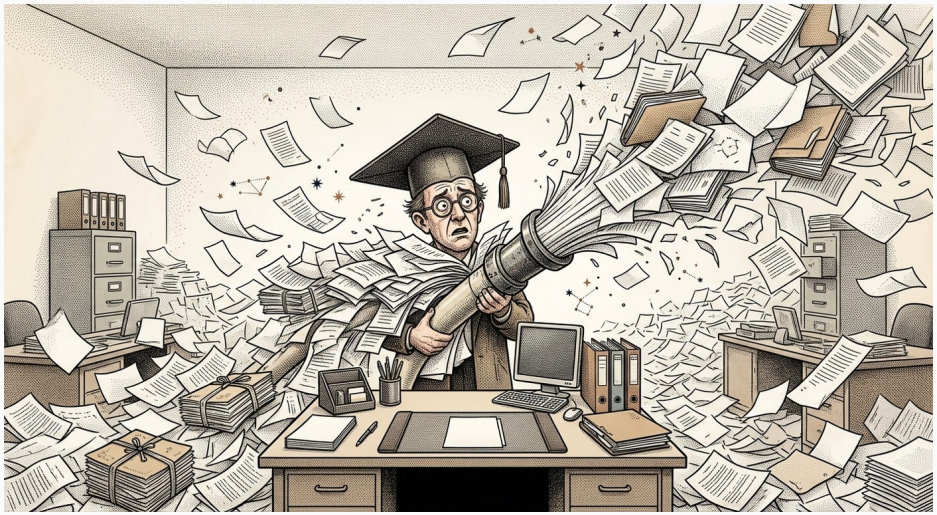
SMT Is a Different Animal



Ground reasoning is fast — then you add quantifiers problem

SAT plus theory propagation is (in our case) decidable and quick. Quantifiers cost you decidability, completeness and peace of mind: the standard answer is to *instantiate* them into ground formulas — e-matching, model-based, conflict-based, syntax-guided, enumerative.

SMT Is a Different Animal

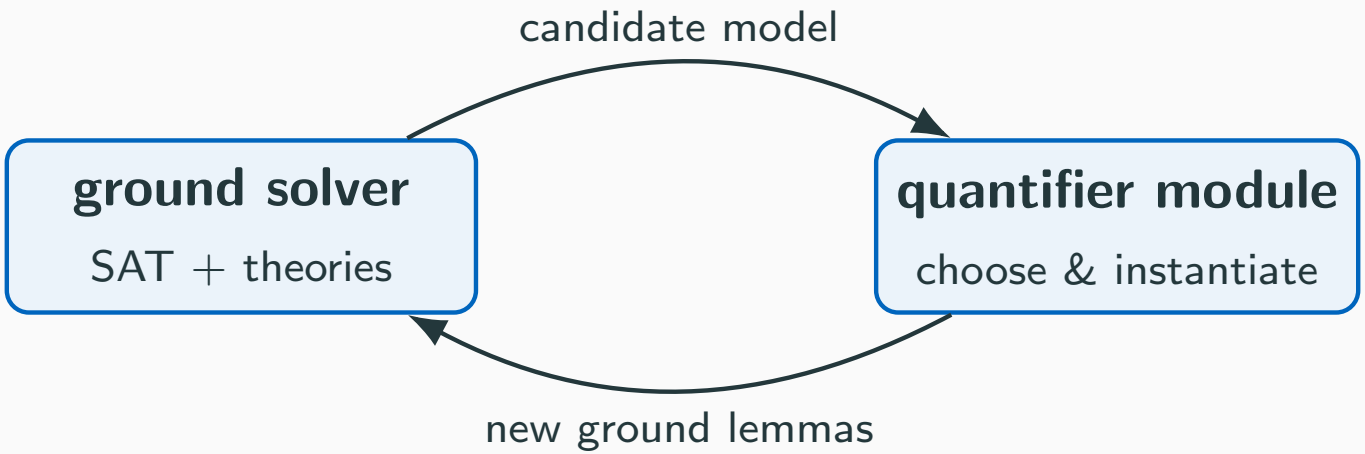


Ground reasoning is fast — then you add quantifiers problem

SAT plus theory propagation is (in our case) decidable and quick. Quantifiers cost you decidability, completeness and peace of mind: the standard answer is to *instantiate* them into ground formulas — e-matching, model-based, conflict-based, syntax-guided, enumerative.

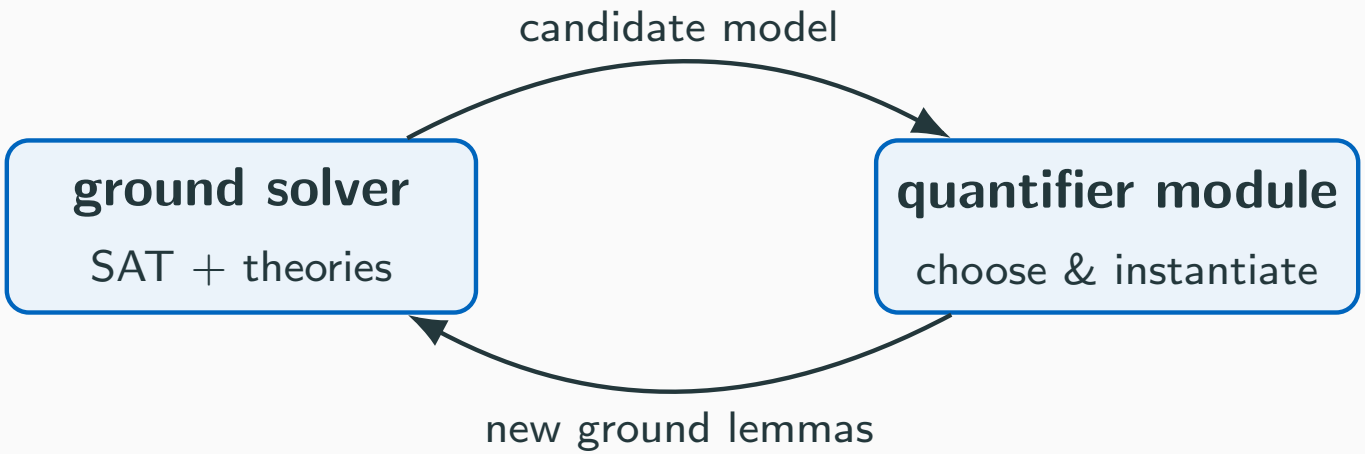
Many schemes, all complementary, all generating terms — the ground solver drowns

The Instantiation Loop



each round adds terms ⇒ the ground problem grows

The Instantiation Loop

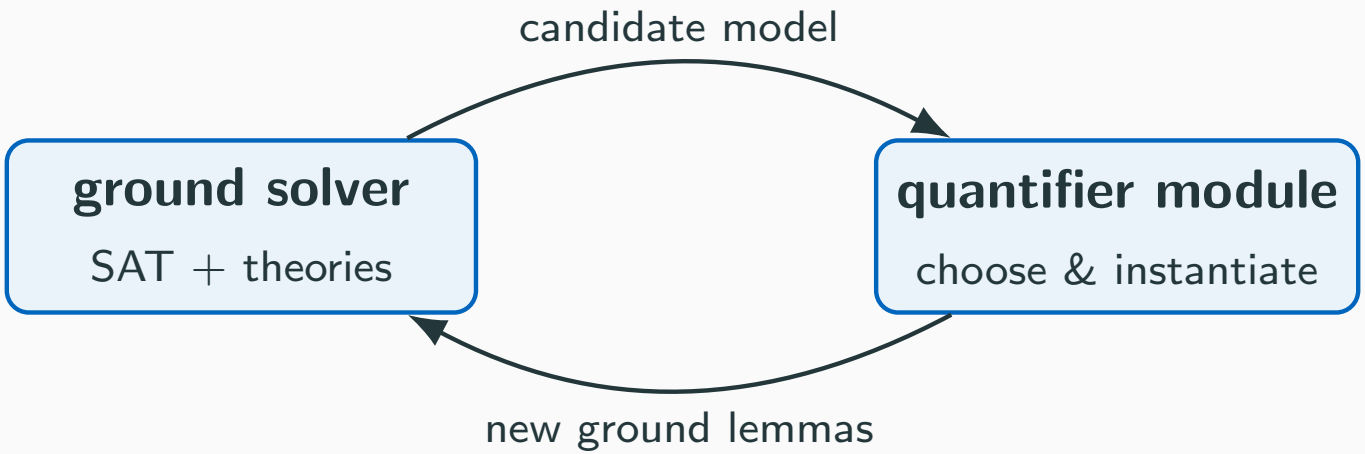


each round adds terms ⇒ the ground problem grows

The real question problem

Not *how* to instantiate, but *which quantifiers* to instantiate at all.

The Instantiation Loop



each round adds terms ⇒ the ground problem grows

The real question problem

Not *how* to instantiate, but *which quantifiers* to instantiate at all.

The same shape as given-clause selection lesson

A hot inner loop with a choice point.

First Attempt (2023): Multi-Armed Bandits

Model the solver as a bandit idea

Each quantifier is a lever; pulling it = instantiating it; reward = the instantiation was useful.

First Attempt (2023): Multi-Armed Bandits

Model the solver as a bandit idea

Each quantifier is a lever; pulling it = instantiating it; reward = the instantiation was useful.

Credit assignment is the hard part problem

You do not know whether an instantiation was useful *until the problem is proved* — and then it is too late. We explored proxy heuristics: does the instance propagate? conflict? survive?

First Attempt (2023): Multi-Armed Bandits

Model the solver as a bandit idea

Each quantifier is a lever; pulling it = instantiating it; reward = the instantiation was useful.

Credit assignment is the hard part problem

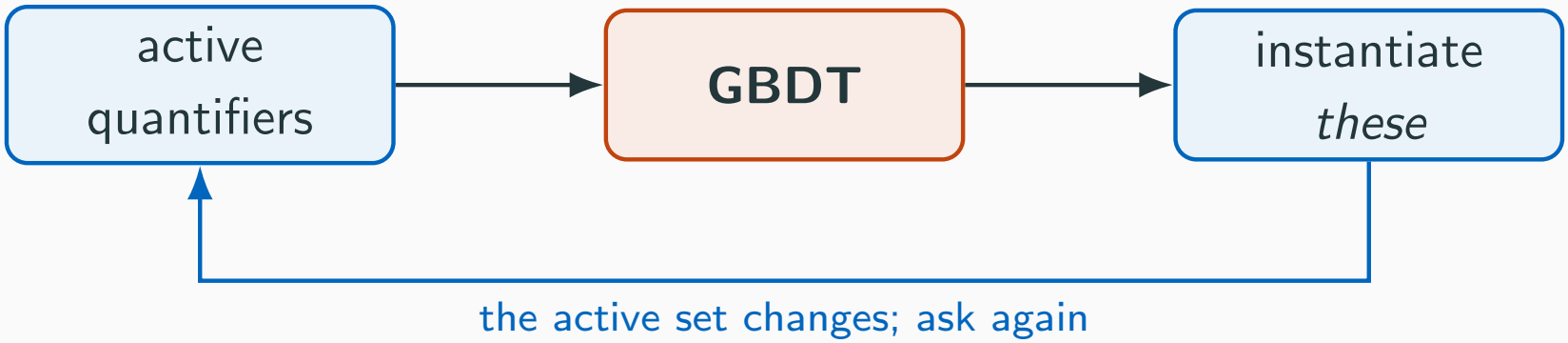
You do not know whether an instantiation was useful *until the problem is proved* — and then it is too late. We explored proxy heuristics: does the instance propagate? conflict? survive?

Honest summary: instructive, not a breakthrough caveat

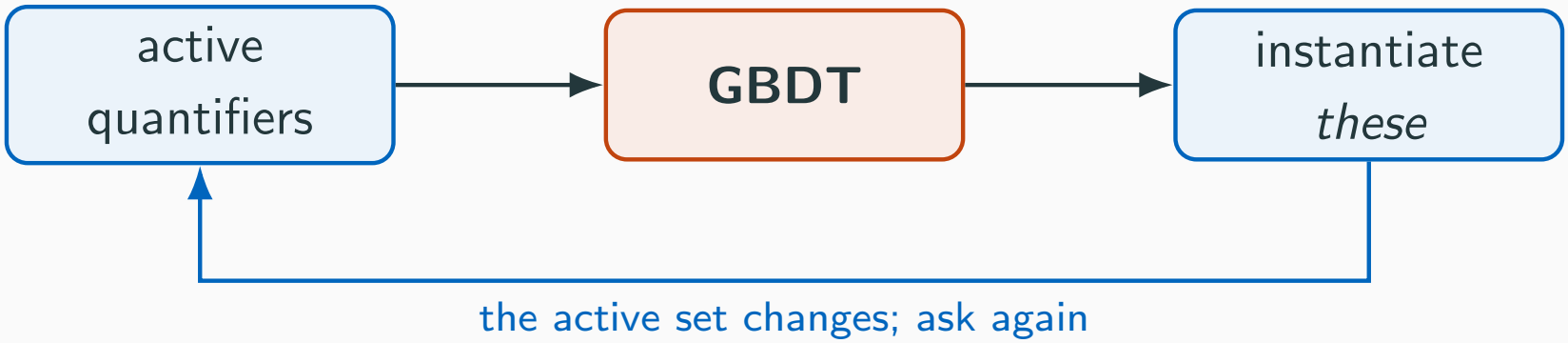
So we went back to what worked for ENIGMA: supervised learning from finished proofs.

[Jakubův, Janota, Piotrowski, Piepenbrock, Reynolds. SMT 2023]

ML Quantifier Selection in cvc5 (2024)



ML Quantifier Selection in cvc5 (2024)



Exactly the ENIGMA recipe, in a different architecture idea

Positives = quantifiers whose instances end up in the proof. The predictor runs *many times per solve* — hence GBDTs, not transformers.

Quantifier Selection: Results

Over state-of-the-art cvc5 result

+20 % on Mizar

+10 % over CVC5's CASC competition portfolio

Validated on *two* independent corpora — Mizar/MML and HOL4 — in both single-strategy and portfolio settings.

Quantifier Selection: Results

Over state-of-the-art cvc5 result

+20 % on Mizar

+10 % over cvc5's CASC competition portfolio

Validated on *two* independent corpora — Mizar/MML and HOL4 — in both single-strategy and portfolio settings.

Neural cvc5 (LPAR 2024): the same idea with a GNN caveat

More accurate, and — as always — the cost is the problem.

[Jakubův, Janota, Piepenbrock, Urban. ECAI 2024 / IJAR 2026]

Everything at Once: the Hard Mizar Problems (2024)

Target: the 14,163 Mizar problems no ATP had solved

Everything at Once: the Hard Mizar Problems (2024)

Target: the 14,163 Mizar problems no ATP had solved

Three ingredients, none of them new by itself idea

- CVC5: instantiation-based, *different* from superposition ⇒ new solutions for free;
- GRACKLE: invented CVC5 strategies — the best solves +14 % over the best pre-existing one;
- **clausification variants**: a boring implementation detail that turns out to matter a lot.

Everything at Once: the Hard Mizar Problems (2024)

Target: the 14,163 Mizar problems no ATP had solved

Three ingredients, none of them new by itself idea

- CVC5: instantiation-based, *different* from superposition ⇒ new solutions for free;
- GRACKLE: invented CVC5 strategies — the best solves +14 % over the best pre-existing one;
- **clausification variants**: a boring implementation detail that turns out to matter a lot.

3,021 hard problems solved (21.3 % of them) result

Mizar ATP coverage: 75 % ⟶ **80 +** %

[Jakubův, Janota, Urban. CICM 2024 • extended version in preparation for JAR]

Outline

What Changed

What Changed

For ten years the human wrote the solver, the machine tuned it idea

Since roughly 2025, LLM coding agents write code that actually compiles, runs, and is not obviously wrong.

What Changed

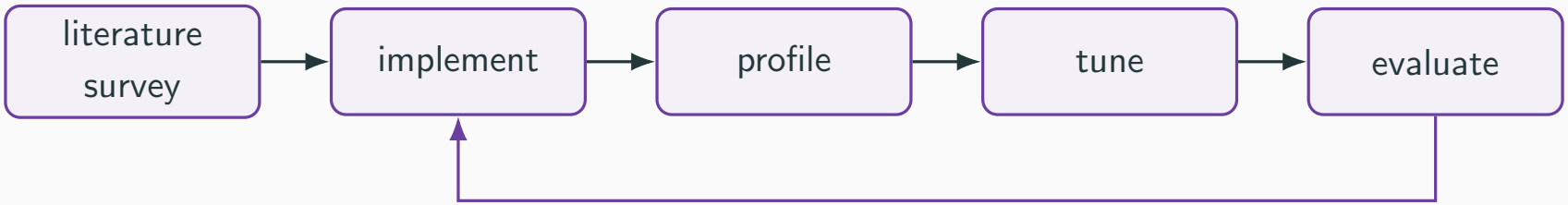
For ten years the human wrote the solver, the machine tuned it idea

Since roughly 2025, LLM coding agents write code that actually compiles, runs, and is not obviously wrong.

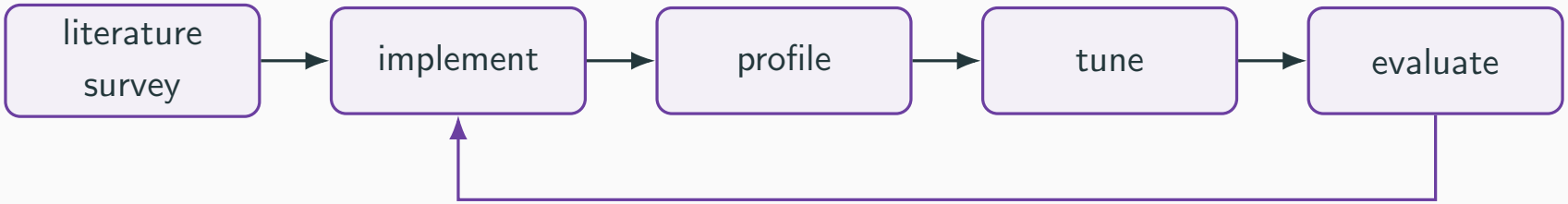
The obvious question for us lesson

Not "can an LLM prove theorems" — that is a different talk. Can an LLM build the tool that proves theorems? We tried it. Twice.

Primo: a Vibe-Coded SMT Solver



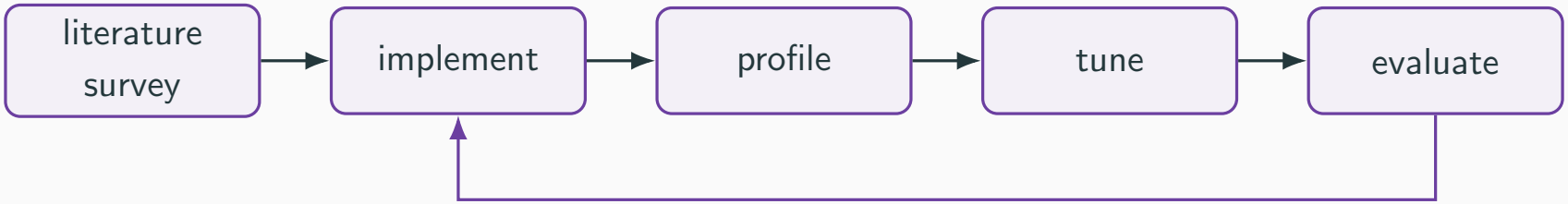
Primo: a Vibe-Coded SMT Solver



An SMT solver for QF-LRA, entirely written by a coding agent idea

The human role: pick the target fragment, supply the literature list, insist on benchmarks, and say "that number looks too good".

Primo: a Vibe-Coded SMT Solver



An SMT solver for QF-LRA, entirely written by a coding agent idea

The human role: pick the target fragment, supply the literature list, insist on benchmarks, and say "that number looks too good".

Then the old machinery kicks in — also vibe-coded lesson

RamParILS: ParamILS rewritten in Rust by the agent, with parallel evaluation and a result cache. It even built Primo's parameter space by reading the option parser — which used to be the most tedious and error-prone part of every project in this talk.

[LPAR 2026 • https://deeper4ai.github.io/ramparils]

Primo: Results on QF-LRA

Full SMT-LIB QF-LRA, 1,753 instances, 30 s result

configuration	*solved*	*%*	*PAR2*
Primo (tuned)	**1,599**	91.2	**12,806**
OpenSMT (default)	1,578	90.0	14,193
Yices (default)	1,563	89.2	13,886
Primo (default)	1,536	87.6	16,597

Tuning is worth +63 instances — exactly what lifts Primo past OpenSMT and Yices. It also beats the SMT-COMP 2026 QF-LRA winner.

Primo: Results on QF-LRA

Full SMT-LIB QF-LRA, 1,753 instances, 30 s result

configuration	*solved*	*%*	*PAR2*
Primo (tuned)	**1,599**	91.2	**12,806**
OpenSMT (default)	1,578	90.0	14,193
Yices (default)	1,563	89.2	13,886
Primo (default)	1,536	87.6	16,597

Tuning is worth +63 instances — exactly what lifts Primo past OpenSMT and Yices. It also beats the SMT-COMP 2026 QF-LRA winner.

The honest version caveat

+77 gained, −14 lost, and 62 of the 77 come from *one* problem family. The headline is true; the story is narrower.

What the Agents Are Good and Bad At

Good at result

- Implementing a known algorithm from a paper.
- Parameter spaces extracted from source.
- Profiling loops, micro-optimization.
- Parsers, harnesses, plots.
- Being tireless.

What the Agents Are Good and Bad At

Good at — result

- Implementing a known algorithm from a paper.
- Parameter spaces extracted from source.
- Profiling loops, micro-optimization.
- Parsers, harnesses, plots.
- Being tireless.

Bad at — caveat

- Knowing when a result is *too good*.
- Benchmark hygiene — it will happily test on the training set.
- Architectural taste; what *not* to build.
- Deciding what is worth doing at all.

What the Agents Are Good and Bad At

Good at result

- Implementing a known algorithm from a paper.
- Parameter spaces extracted from source.
- Profiling loops, micro-optimization.
- Parsers, harnesses, plots.
- Being tireless.

Bad at caveat

- Knowing when a result is *too good*.
- Benchmark hygiene — it will happily test on the training set.
- Architectural taste; what *not* to build.
- Deciding what is worth doing at all.

The scarce resource moved lesson

From *writing code* to *judging results*.

Outline

Lessons

The default is a local optimum lesson

Every solver we touched — E, Vampire, Prover9, cvc5, Bitwuzla, Primo — got substantially better from tuning alone.

Lessons

The default is a local optimum lesson

Every solver we touched — E, Vampire, Prover9, cvc5, Bitwuzla, Primo — got substantially better from tuning alone.

Complementarity beats strength; iterate; watch the clock lesson

- The best portfolio is never the top-k individual strategies.
- Every method here is a feedback loop. One-shot training leaves most of the gain behind.
- Inference speed is a research problem, not an afterthought.

Lessons

The default is a local optimum lesson

Every solver we touched — E, Vampire, Prover9, cvc5, Bitwuzla, Primo — got substantially better from tuning alone.

Complementarity beats strength; iterate; watch the clock lesson

- The best portfolio is never the top-k individual strategies.
- Every method here is a feedback loop. One-shot training leaves most of the gain behind.
- Inference speed is a research problem, not an afterthought.

Nothing wins alone lesson

Guidance + strategies + premise selection + a second solver. Every headline number in this talk is a combination.

The Uncomfortable Ones

We almost never know why an invented strategy works caveat

Ten years in, interpretability is barely started.

The Uncomfortable Ones

We almost never know why an invented strategy works caveat

Ten years in, interpretability is barely started.

Everything overfits — and we cannot say what will transfer caveat

Ours, and everybody else's competition schedules too. Worse: we do not know what makes a benchmark respond to learning at all. The same methods that give +20 % on Mizar do much less on TPTP, SMT-COMP or Isabelle. We have guesses, not answers.

The Uncomfortable Ones

We almost never know why an invented strategy works caveat

Ten years in, interpretability is barely started.

Everything overfits — and we cannot say what will transfer caveat

Ours, and everybody else's competition schedules too. Worse: we do not know what makes a benchmark respond to learning at all. The same methods that give +20 % on Mizar do much less on TPTP, SMT-COMP or Isabelle. We have guesses, not answers.

Most of these systems are hard to reuse caveat

GRACKLE is the exception — and only because we were forced to generalize it.

Where This Is Going

Near term idea

- **Higher-order** reasoning as a first-class target, not an encoding.
- **Configuration on the fly**: stop shipping one schedule; configure per problem, at runtime.

Where This Is Going

Near term

idea

- **Higher-order** reasoning as a first-class target, not an encoding.
- **Configuration on the fly**: stop shipping one schedule; configure per problem, at runtime.

Further out

idea

- **Agents in the loop**: not just writing the solver, but reading the profiles and proposing the next experiment.
- **Self-improving solvers**: the two loops of this talk, closed around each other, running unattended.

Where This Is Going

Near term idea

- **Higher-order** reasoning as a first-class target, not an encoding.
- **Configuration on the fly**: stop shipping one schedule; configure per problem, at runtime.

Further out idea

- **Agents in the loop**: not just writing the solver, but reading the profiles and proposing the next experiment.
- **Self-improving solvers**: the two loops of this talk, closed around each other, running unattended.

And the question we still cannot answer caveat

What makes a problem hard?

Thanks

This is all joint work. In rough chronological order:

Josef Urban • Cezary Kaliszyk • Bob Veroff
Stephan Schulz • Martin Suda • Karel Chvalovský
Zarathustra Goertzel • Miroslav Olšák • Bartosz Piotrowski
Mikoláš Janota • Jelle Piepenbrock • Andrew Reynolds
Jan Hůla • Lukáš Kubej • Kristina Aleksandrova

CTU CZECH TECHNICAL UNIVERSITY IN PRAGUE
erc
Ministry of Education, Youth and Sports of the Czech Republic
GAČR CZECH SCIENCE FOUNDATION
Renaissance Philanthropy

AI contribution to this talk: slides drafted and typeset with Claude, illustrations generated with Gemini.

Thank you!

Questions?

The default is a local optimum.

Go and tune something.