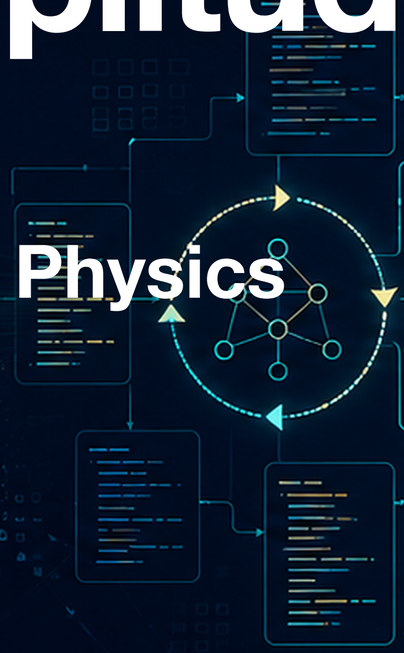
Scattering amplitudes as programmes

AI for Scientific Discovery in Physics



Sven Krippendorf, 11.09.2026
ECT* workshop — Trento



UNIVERSITY OF CAMBRIDGE

What would count as AI discovery in physics?

From solving a specified task to producing new scientific structure

Specified task

Known solution → approximation

Data → parameter inference

Known calculation → acceleration

Discovery

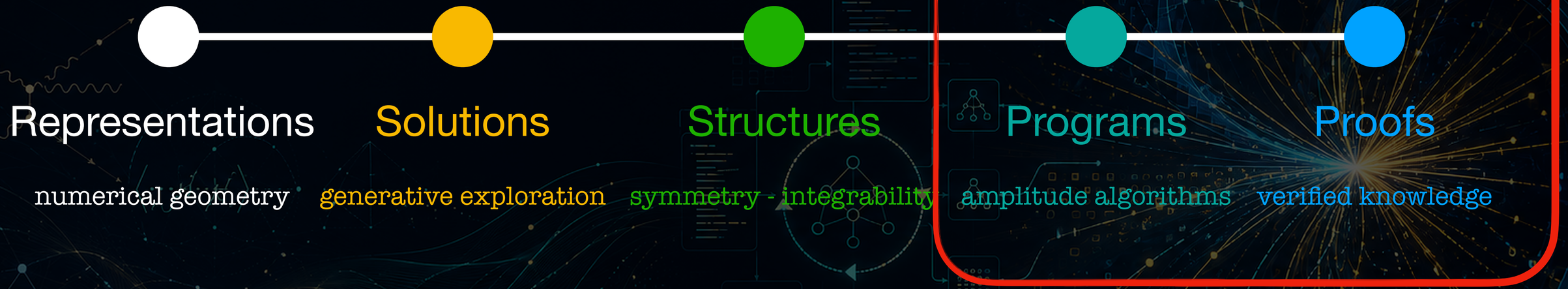
Dynamics → **symmetry**

Model space → **new solutions**

Physics problem → **new algorithm**

What should a machine discover?

From black boxes to scientific knowledge



Today's focus

Representations Solutions Structures Programs Proofs

numerical geometry Generative exploration symmetry - integrability amplitude algorithms verified knowledge

Act I — From structures to programs

What if the discovered object can perform the calculation itself?

Search directly in the space of scientific objects

From learned representations to programs

Implicit representation

Weights
Latent variables
Learned strategy



Explicit scientific artifact

Equations
Symmetries
PROGRAMS
Proofs

Why programs?

- **Executable**
- **Testable**
- **Inspectable**
- **Optimizable**

A program is an inspectable piece of physical understanding.

Scattering amplitudes as programs

Evolving the calculation, not approximating the answer



Scattering amplitudes as programs



Joint work with Yi Gu (PhD student)

arxiv:2607.21629
GitHub: https://github.com/YiGu310/scattering-amplitudes-program-search
Tutorial: https://github.com/slk38/seoul-summer-school-2026-tutorials

Why amplitudes?

LHC needs amplitude calculations billions of times

330 **billion V+jets events**
for a 3 ab^{-1} HL-LHC

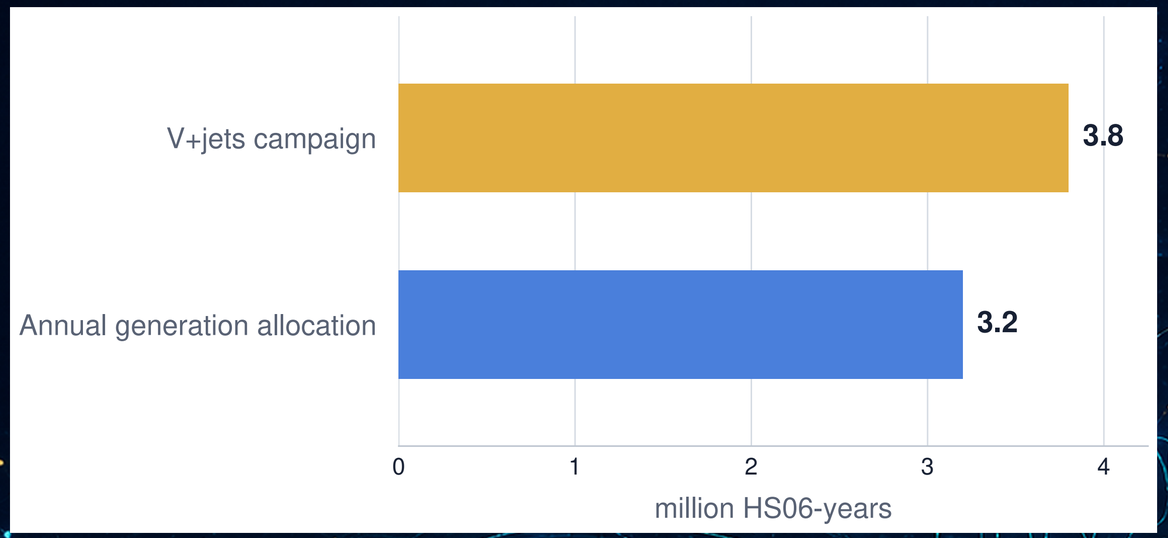
3.8 M **HS06-years**
with improved Sherpa (event generator) setup

Event generation is projected to consume order 10–20% of HL-LHC CPU.

Event-generation share — not amplitude evaluation alone. (e.g. ATLAS 2112.09588)

One V+jets production campaign can exceed a year's allocation

CPU requirement: one full-data-set campaign versus the assumed 2031 annual allocation:



>50%

phase-space sampling + tree-level matrix-element evaluation

ATLAS, arXiv:2112.09588; Bothmann et al., arXiv:2209.00843.

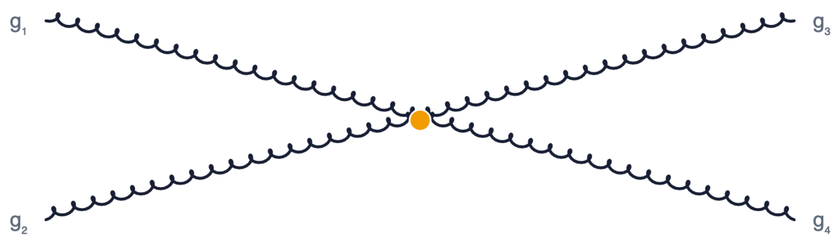
The costly part is not deriving an amplitude once—it is evaluating a numerical programme reliably and repeatedly.

Combinatorial growth in amplitude calculations

Gluon tree-level scattering

gg → 2g · contact

$(V_3, V_4) = (0,1)$



gg → 2g · s-channel

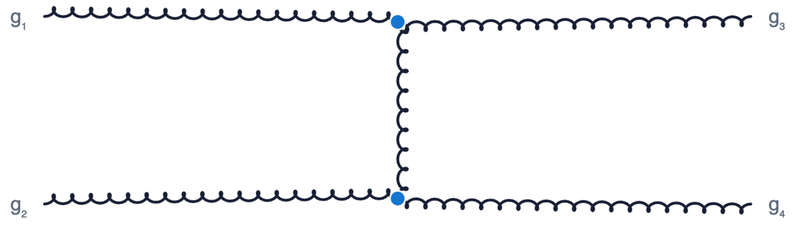
$(V_3, V_4) = (2,0)$



one of three cubic exchange diagrams

gg → 2g · t-channel

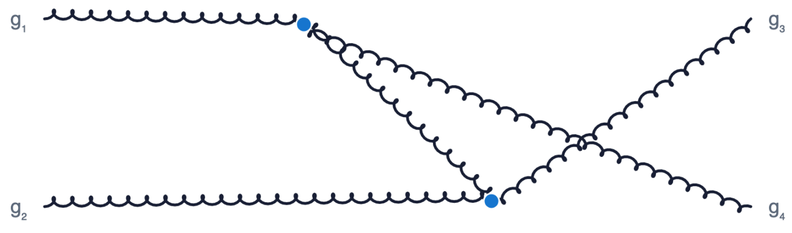
$(V_3, V_4) = (2,0)$



one of three cubic exchange diagrams

gg → 2g · u-channel

$(V_3, V_4) = (2,0)$



one of three cubic exchange diagrams

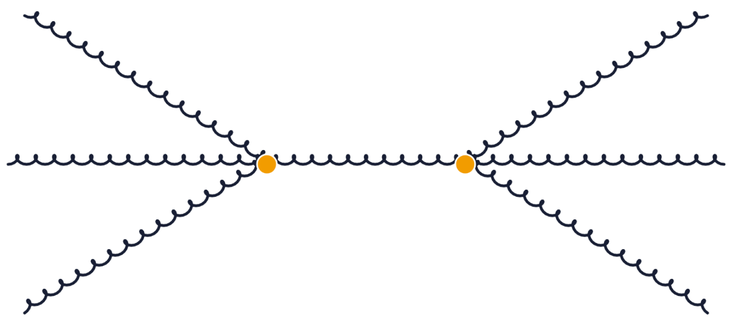
Process	# diagrams
gg -> 2g	4
gg -> 4g	220
gg -> 6g	34300

Combinatorial growth in amplitude calculations

Gluon tree-level scattering

gg → 4g · all quartic

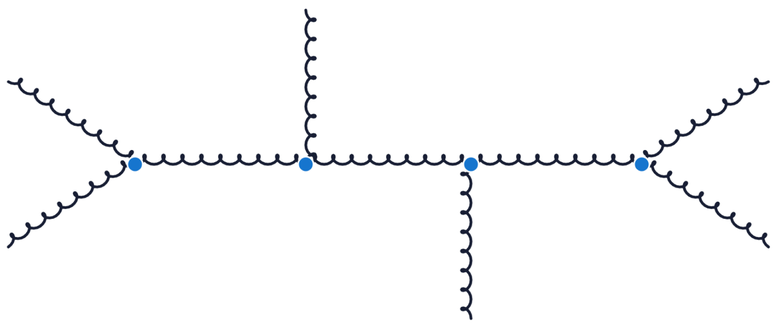
$(V_3,V_4) = (0,2)$



10 diagrams in this vertex-content category

gg → 4g · all cubic

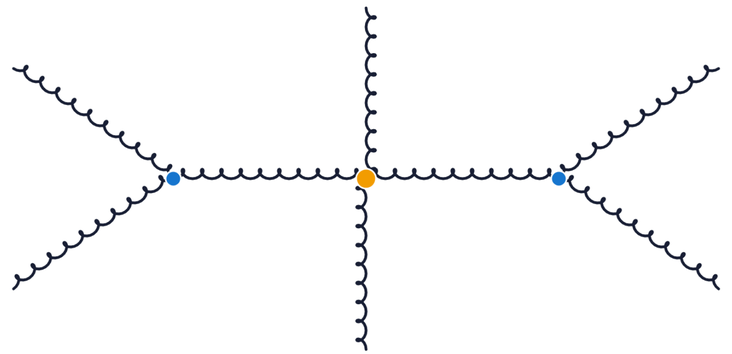
$(V_3,V_4) = (4,0)$



105 diagrams in this vertex-content category

gg → 4g · mixed

$(V_3,V_4) = (2,1)$



105 diagrams in this vertex-content category

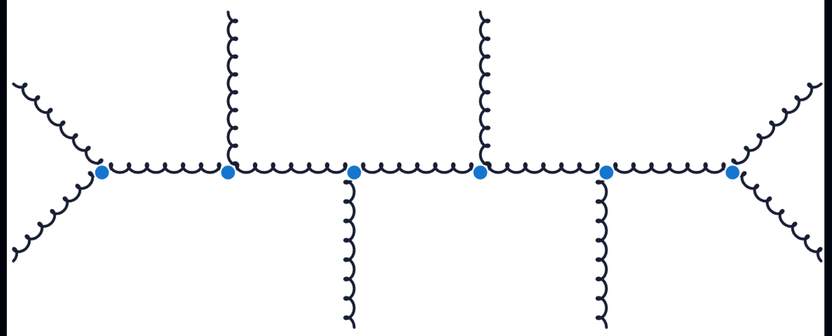
Process	# diagrams
gg -> 2g	4
gg -> 4g	220
gg -> 6g	34300

Combinatorial growth in amplitude calculations

Gluon tree-level scattering

gg → 6g · all cubic

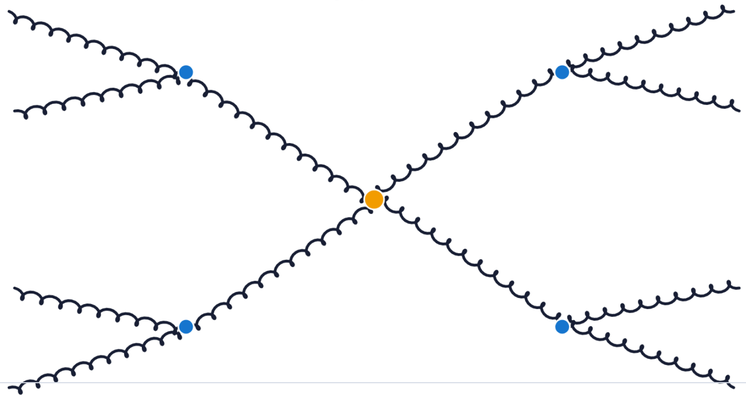
$(V_3, V_4) = (6,0)$



10,395 diagrams in this vertex-content category

gg → 6g · one quartic

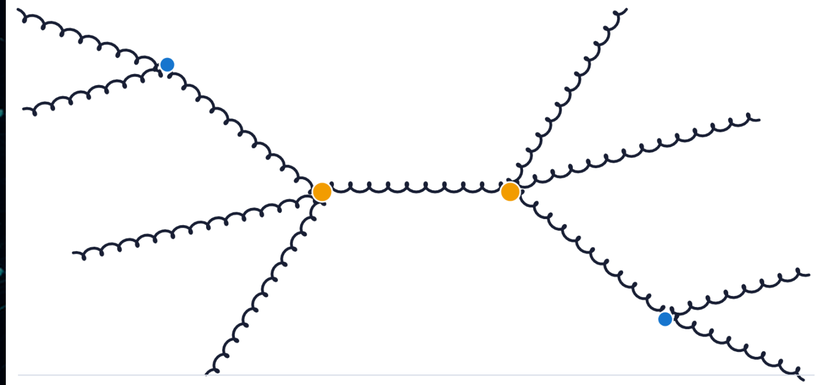
$(V_3, V_4) = (4,1)$



17,325 diagrams in this vertex-content category

gg → 6g · two quartic

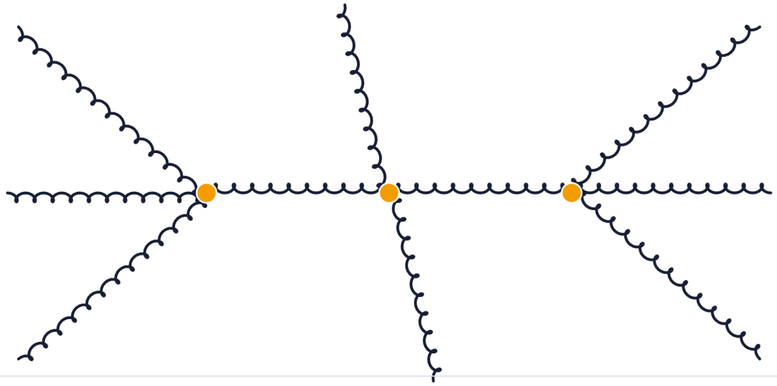
$(V_3, V_4) = (2,2)$



6,300 diagrams in this vertex-content category

gg → 6g · all quartic

$(V_3, V_4) = (0,3)$



280 diagrams in this vertex-content category

Process	# diagrams
gg -> 2g	4
gg -> 4g	220
gg -> 6g	34300

Amplitude fixed — programme is not



Same physical map

input $\longrightarrow$ A_n, $|M_n|^2$ or event weight

Feynman diagrams	Berends–Giele currents	CSW / MHV vertices	BCFW recursion	generated / hybrid code
1949	1988	2004	2005	today

All are correct routes to the same answer.

They can differ radically in scaling, cancellation, reuse of substructures, memory and hardware suitability.

Amplitude theory can be read as a history of discovering better programmes.

From theoretical physics to program search

Traditional research loop

Idea/representation
↓
Derive algorithm
↓
Implement calculation
↓
Check correctness
↓
Measure performance
↓
Revise idea

Make the loop executable

Candidate solution = program
Seed program: Feynman diagram calculation

Physics constraints → tests
Correct amplitude?

Computational value → benchmark
How fast is it?

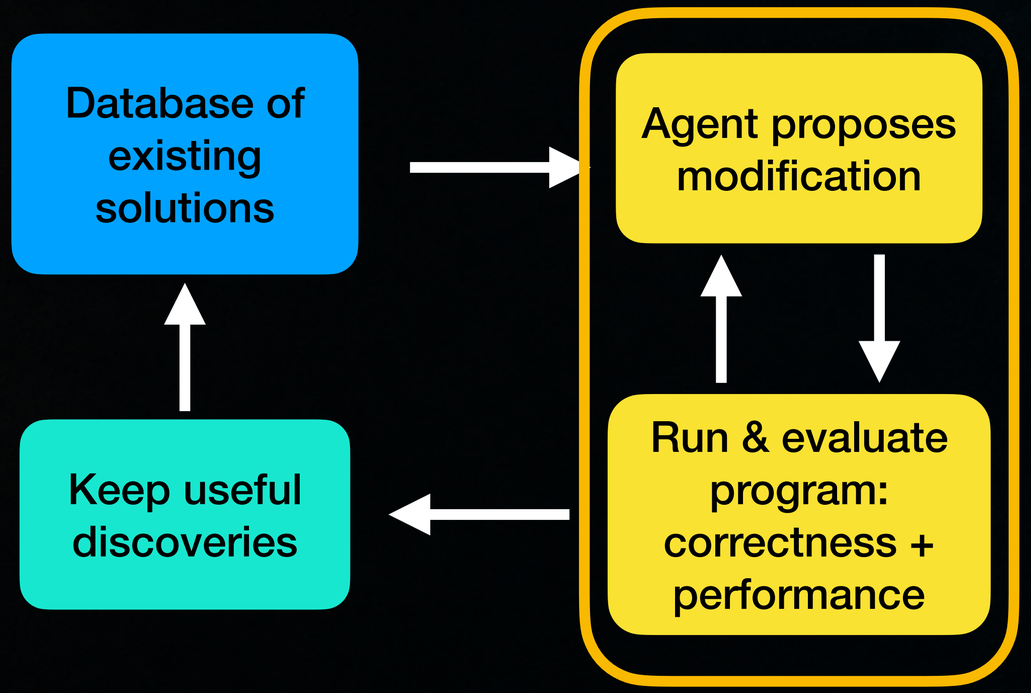
Search → modify program
Can we find something better?

From implementing an idea → searching for better ideas.

What does the agent actually do?

Agent sees:

- Source code
- Previous attempts
- Correctness tests
- Runtime feedback



Agent can:

- Rewrite algorithms
- Introduce intermediate quantities
- Change recursion/ decomposition
- Combine previous ideas

Can be combined in coding agent

What is actually being optimized?

Not parameter fitting — algorithm discovery = ways of calculating

Parameters
$a = 0.4 \rightarrow 0.37$

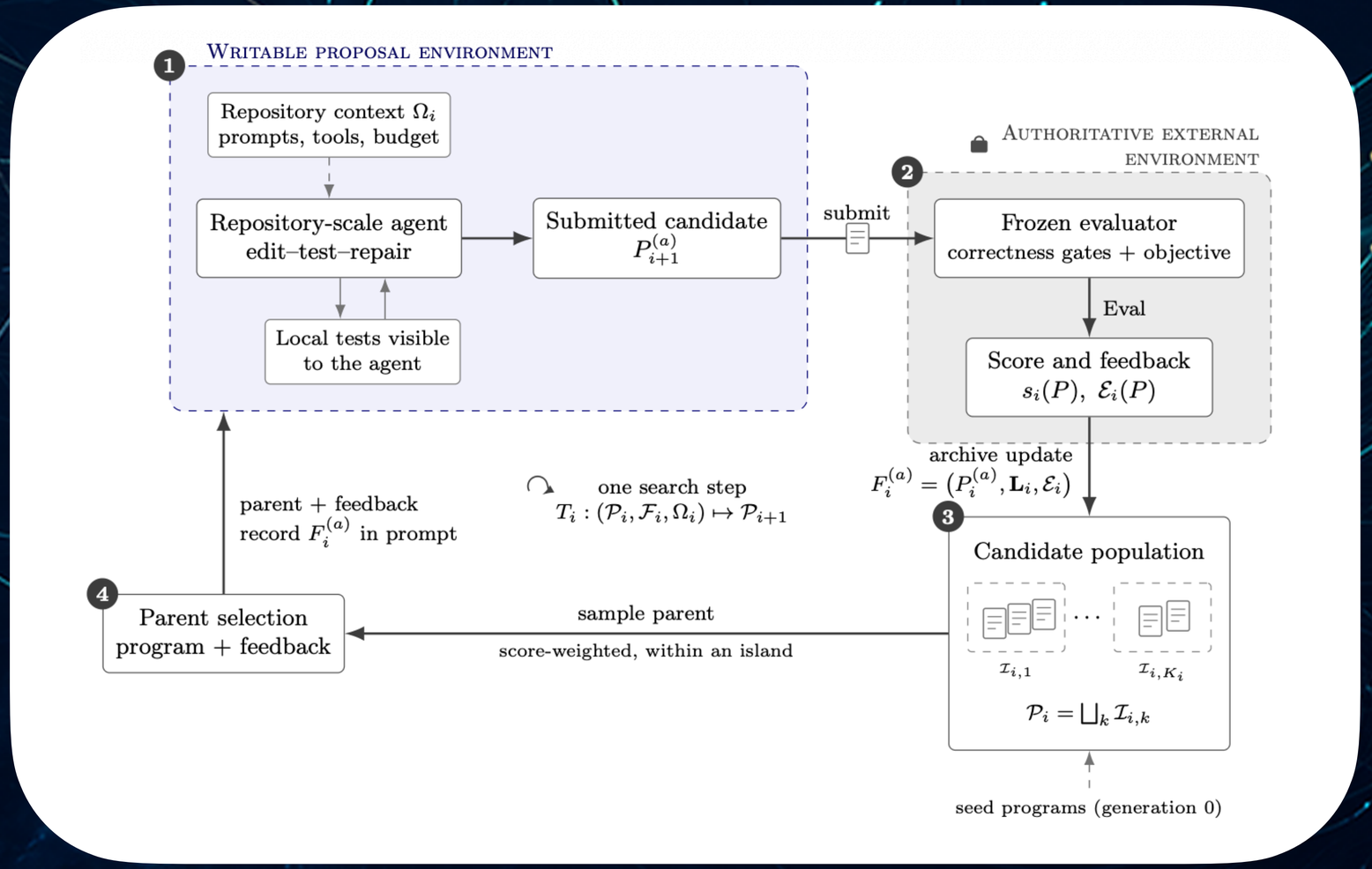
Implementation
Loop → vectorization

Algorithm
BCFW → different decomposition → new specialized structure

The search space contains programs, not just numbers.

Agent proposes code — frozen evaluator decides

Output: Code — no numerical approximation





+



Three objectives probing programme space

SCALING

How does cost grow with multiplicity?

Selection pressure

Reusable and transferable structure

CANCELLATION

How locally does unphysical structure cancel?

Selection pressure

R-invariant cells and boundary gluing

EXACT OPERATIONS

How much arithmetic does the matrix-element kernel perform?

Selection pressure

Hybrid recursion, symmetry, bases and reuse

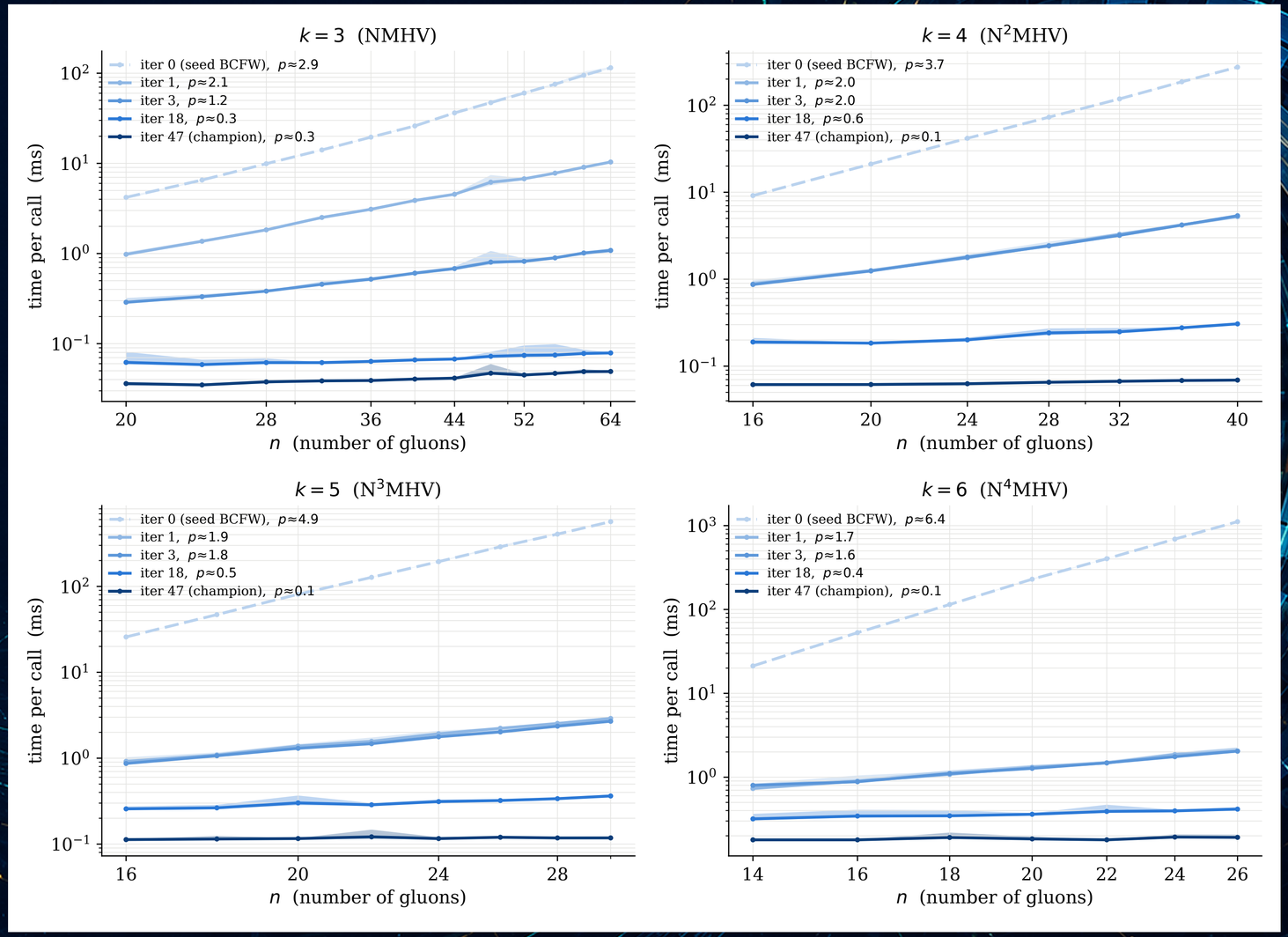
Change the objective—and the surviving mathematical language changes.

Scaling pressure changes the mathematical representation

805x

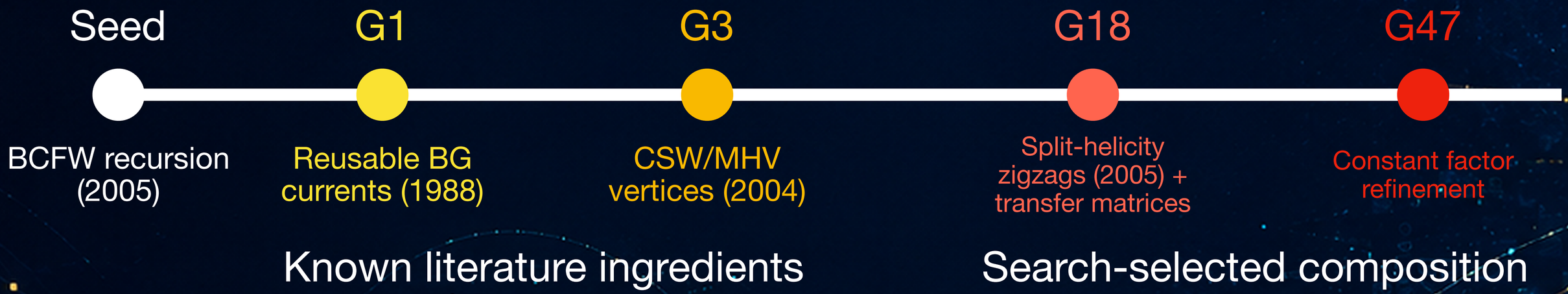
geometric-mean speed-up
34 cells (k, n) vs seed BCFW

$\mathcal{O}(k^2 n)$ runtime
audited through $n = 256$



The search tracks ideas with a history in theory

Objective: Scaling and Speed



Novelty claim: executable, audited composition of literature ingredients.

Cancellation objective to target spurious poles

$$|T_1|, |T_2| \gg |A|$$
$$A = T_1 + T_2$$

finite physical answer

$$C = \frac{\sum_i |T_i|}{\left|\sum_i T_i\right|}$$

A poor representation can be correct only after violent inter-term cancellation.

- Individual terms expose unphysical singularities.
- Large cancellations obscure underlying structure and lead to numerical instability.
- Objective rewards cancellation within terms rather than between them.

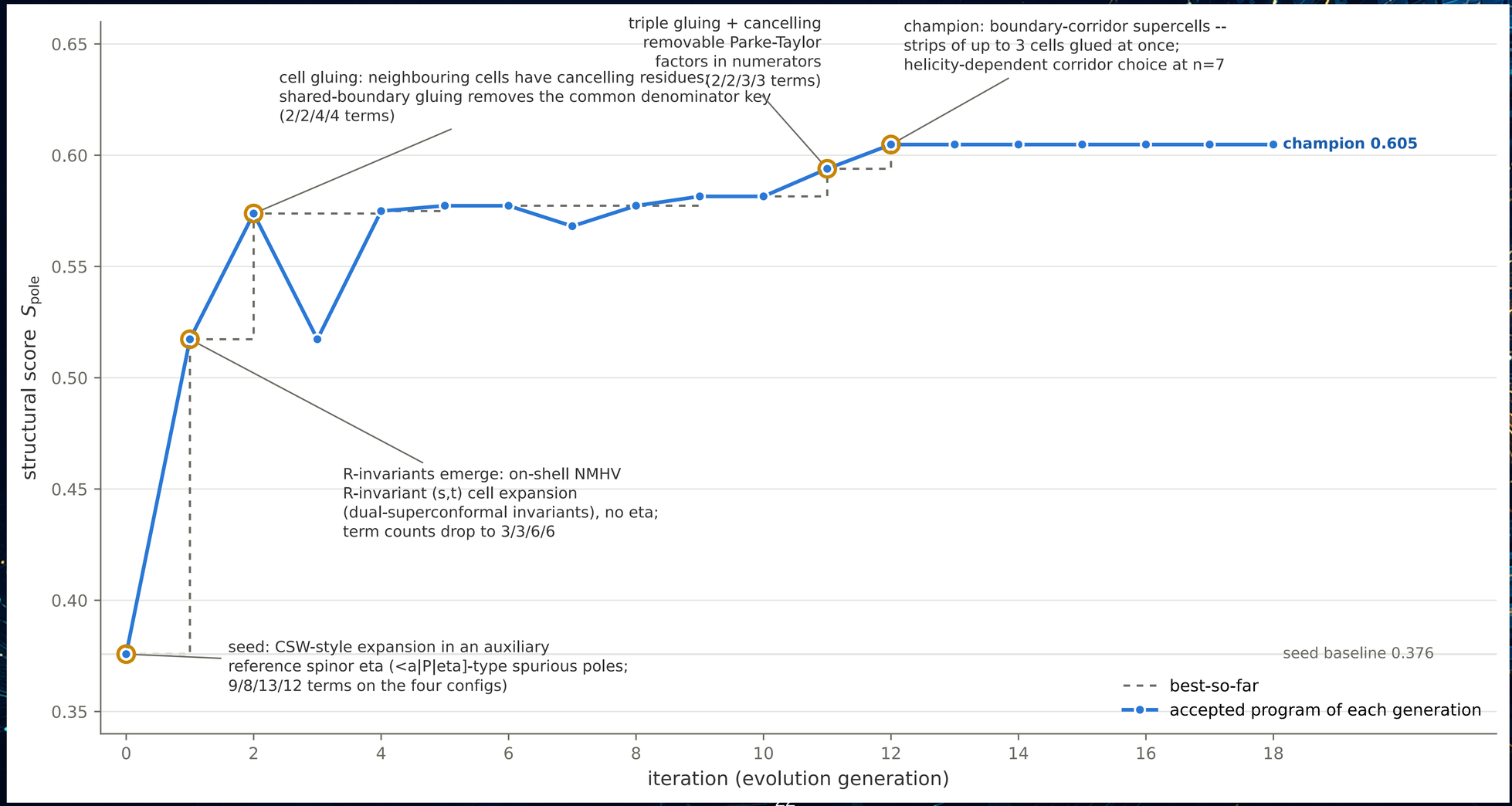
Historically productive: CSW gluon diagrams (2004) expose reference-spinor spurious poles; R-invariant and polytope forms reorganize or remove them.

Hodges 2013, Arkani-Hamed et al. 2012

Ask not only for right answer — but also how it is represented.

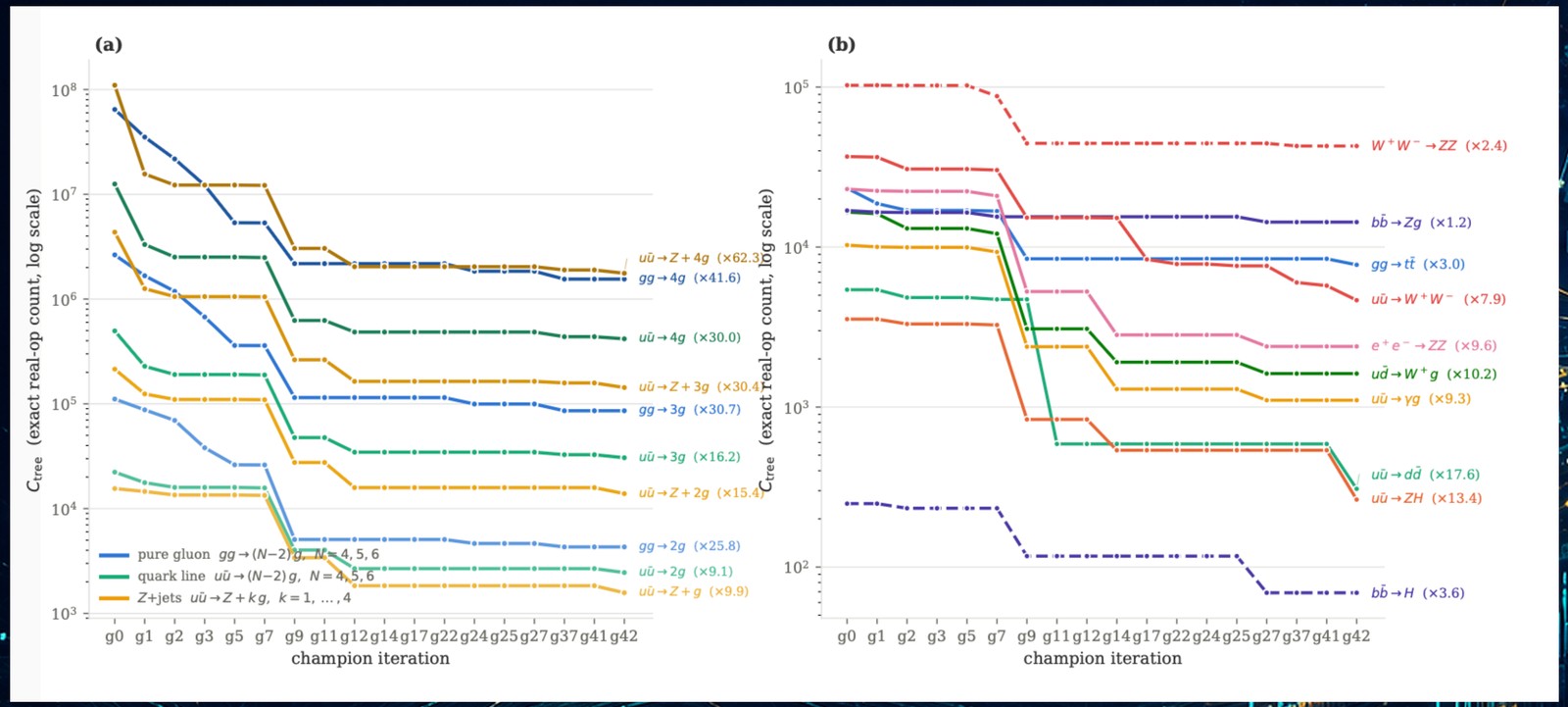
Cancellation pressure selects cells — and then glues them

CSW seeds → R-invariant cells → boundary gluing → supercells



20-process benchmark → LHC matrix-elements

Operation count as objective



195 million operations

↓

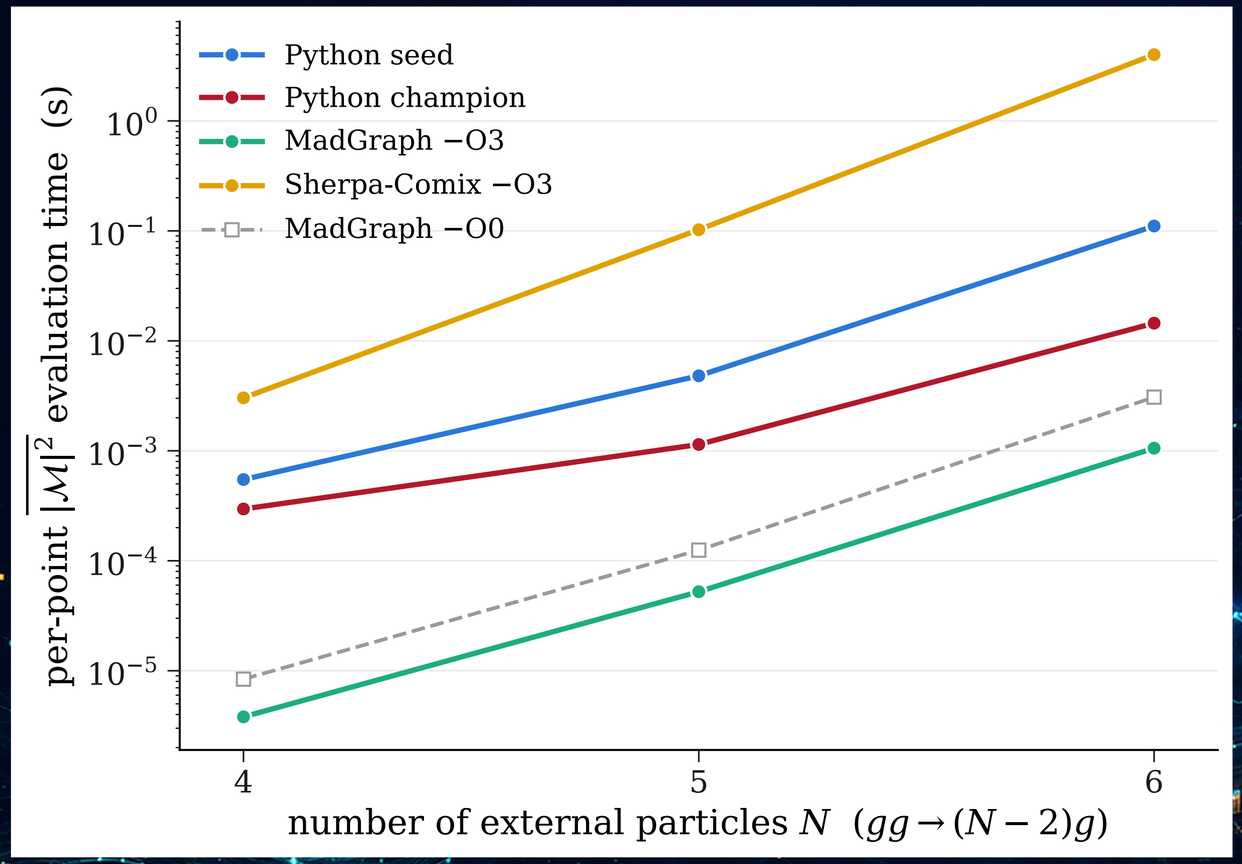
4 million operations

47.7x fewer operations

5.9x faster (runtime)

One frozen programme is selected across all 20 QCD and electroweak processes.

The searched representation grows least from 4 to 6 gluons



4.1x smaller runtime growth than seed

6 gluons: 277x faster than Comix/ Sherpa call and 13.7x slower than Madgraph

Our solution growth the least

Disclaimer: Fixed process kernel comparison-not a ranking of general-purpose generators.

Each objective selects different mathematical language

OBJECTIVE	REPRESENTATION SELECTED	OBSERVED CONSEQUENCE
Scaling	**Transfer-based split-helicity structure**	linear growth in tested families · 805×
Local cancellation	**R-invariant cells + glued boundaries**	2–3 terms · score 0.605
Exact arithmetic	**Hybrid recursion, symmetries, bases and reuse**	47.7× fewer total operations

The objective shapes the language in which the amplitude is expressed.

What it establishes — and what it does not

Establishes

Repository-scale coding agents can traverse mathematical representations.

External objectives condition which structures survive.

The resulting programmes can be audited as mathematics and software.

Large gains remain in controlled tree-level benchmarks.

Does not yet establish

Discovery of structures absent from pretraining.

End-to-end production speed-up or universal generator ranking.

Loop-level or arbitrary-process generality.

A replacement for domain expertise in defining the search problem.

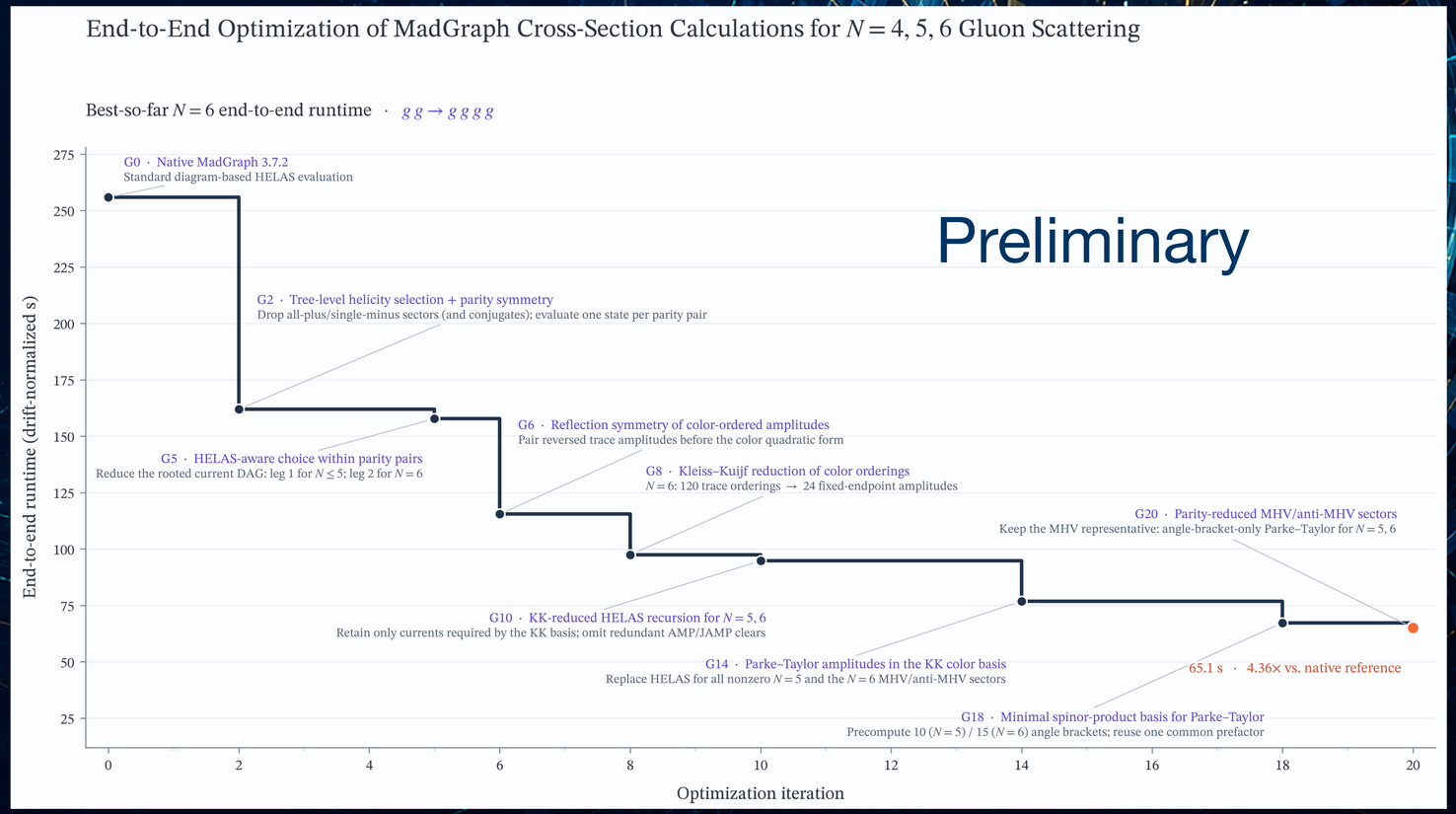
What’s new? Demonstrated search process — not a claim that underlying amplitude structures were unknown.

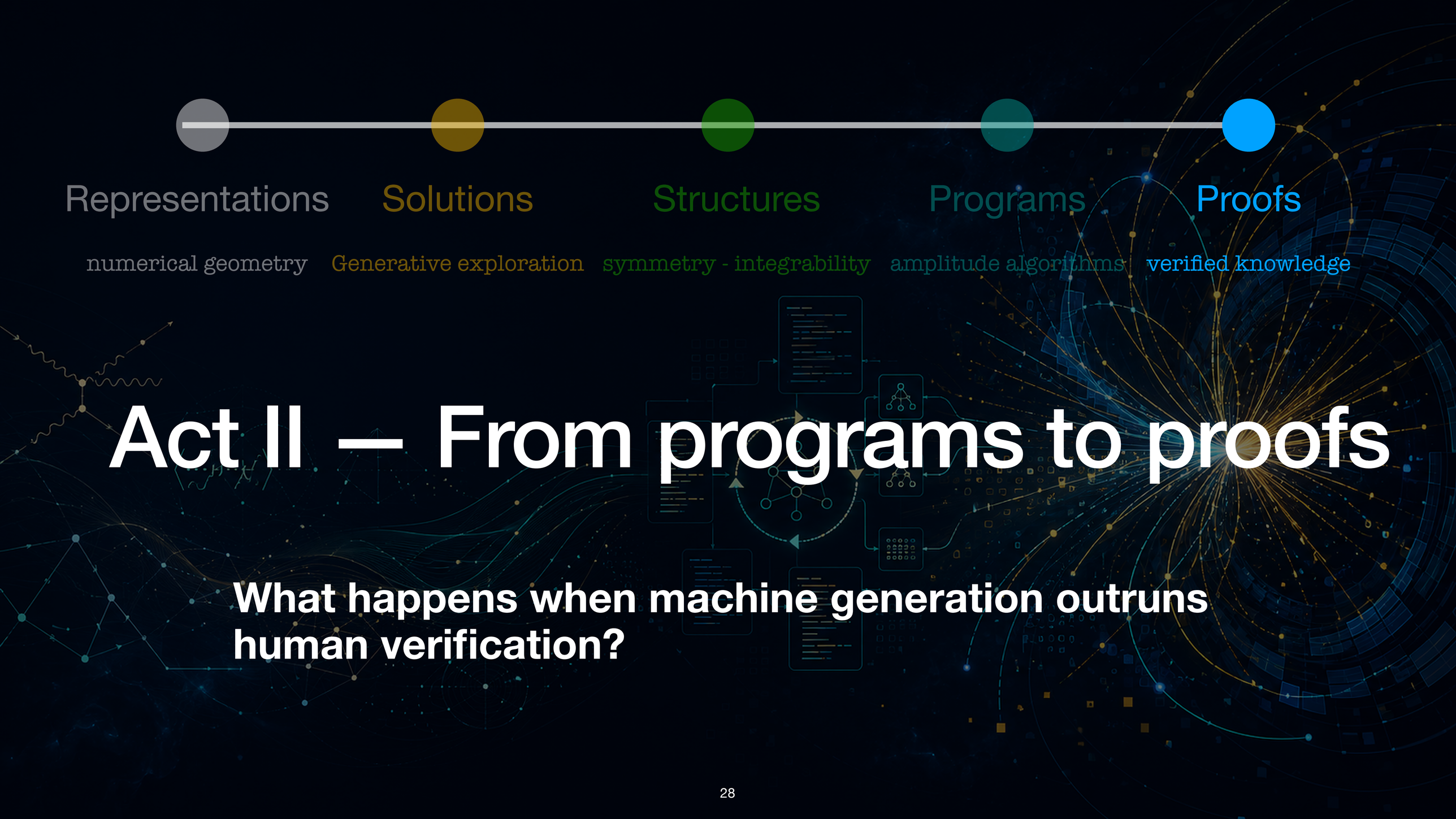
From faster amplitudes to more physics

Next step: native event-generator integration

Representation and algorithms → Compiled and parallel code → **Generator integration** (sampling, phase space, unweighting) → **End-to-end physics throughput** (events per CPU-hour and physics reach)

The relevant benchmark is eventually physics throughput, not isolated amplitude evaluation.

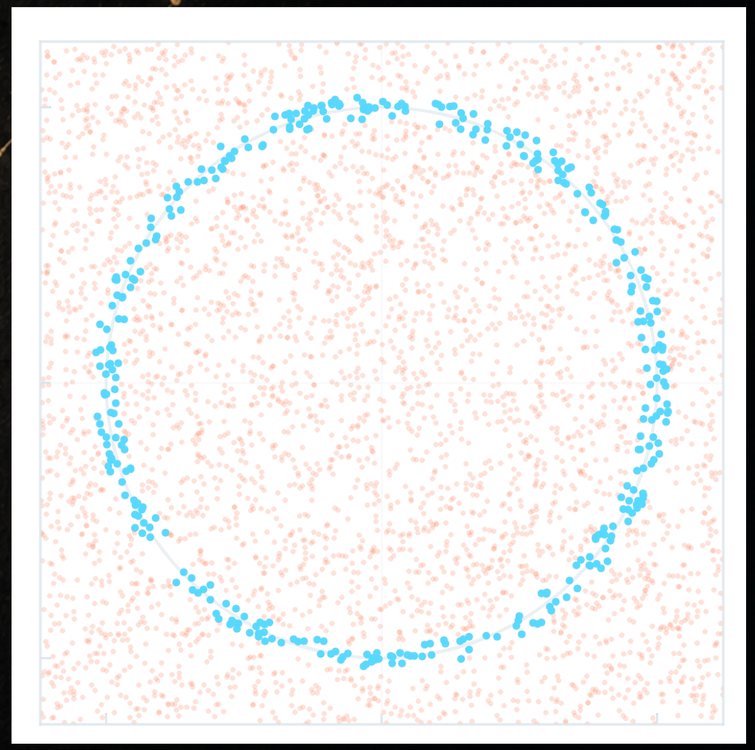


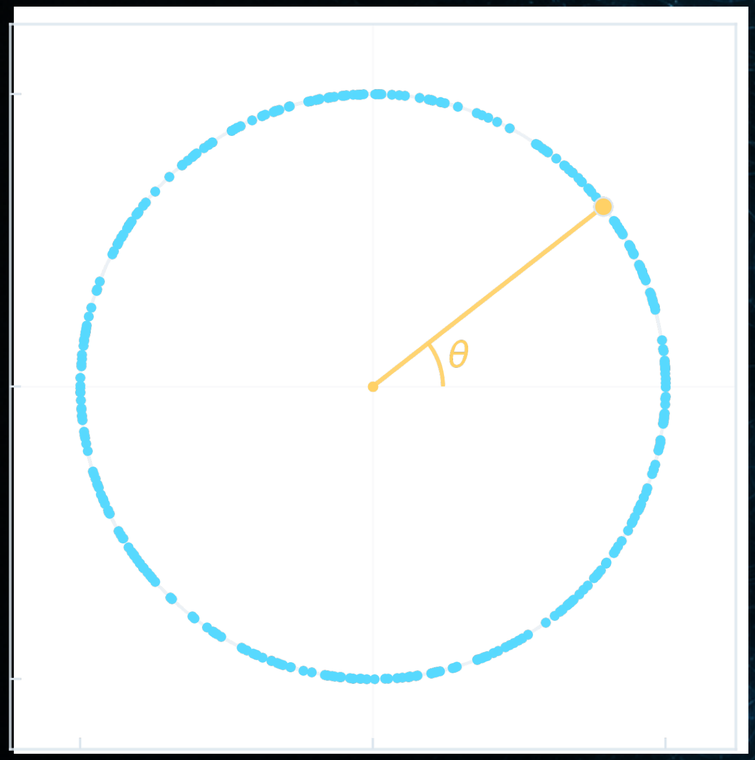


Act II — From programs to proofs

What happens when machine generation outruns human verification?

Why programmes are powerful generators





In high-dimensional scientific search spaces, brute-force generation becomes prohibitive; a good program/model encodes structure and generates directly in the relevant space.

THE CATCH

Your AI-generated code is only as good as your tests/harness

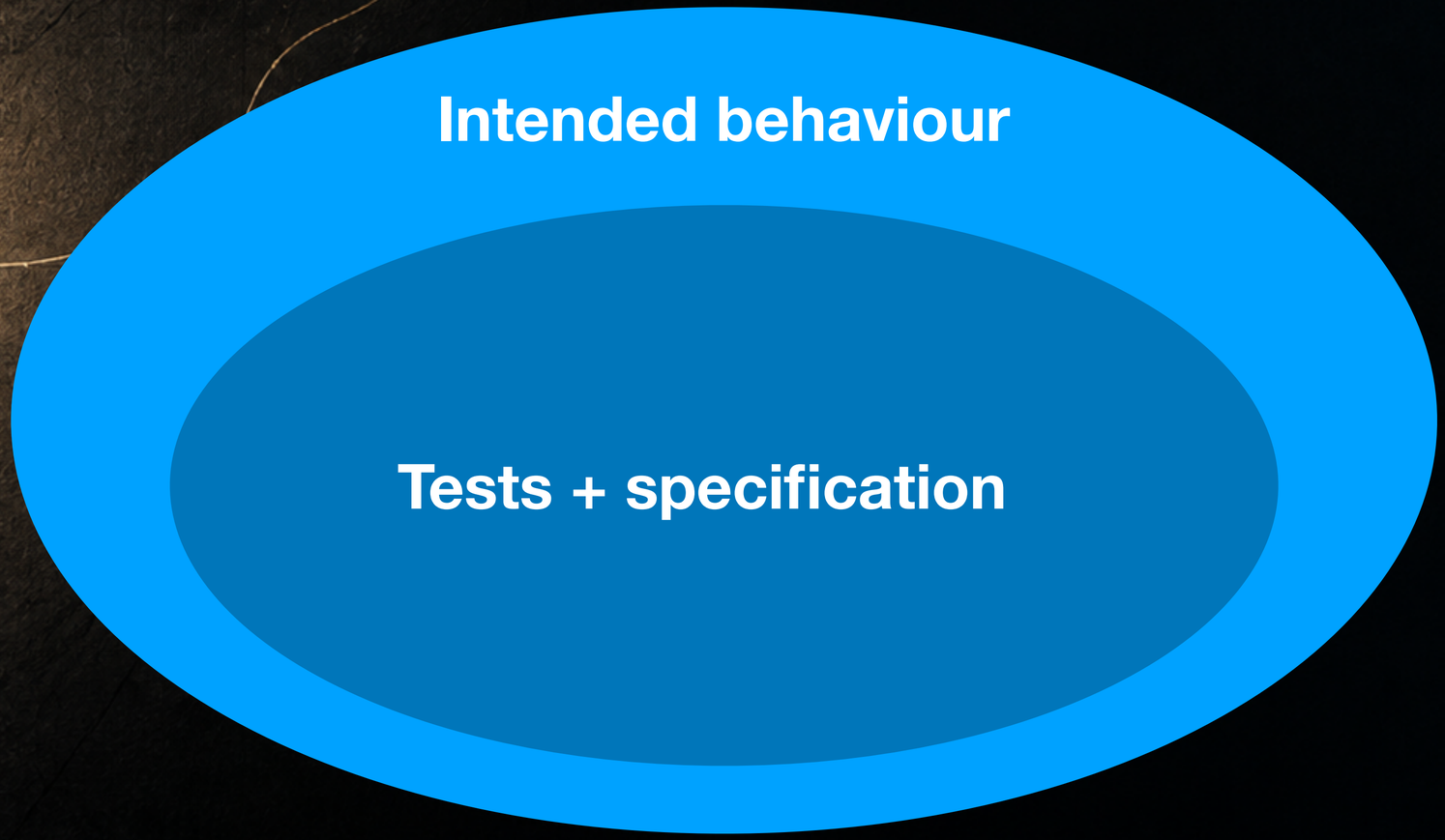
Objective: Make this algorithm faster

Agent solution: Remove an expensive validation step

Benchmark: faster

Algorithm: wrong

The agent optimizes what you measure



An increasingly capable optimiser will find gaps in the specification.

This is not an AI failure — it may be a specification failure.

Legacy software/procedures make this difficult

- Requirements may never been documented
- Behaviour accumulated over decades
- Edge cases live in people’s heads and not in papers
- Tests only encode historical failures

Before autonomously improving algorithm, we first need to reconstruct what it is supposed to do.

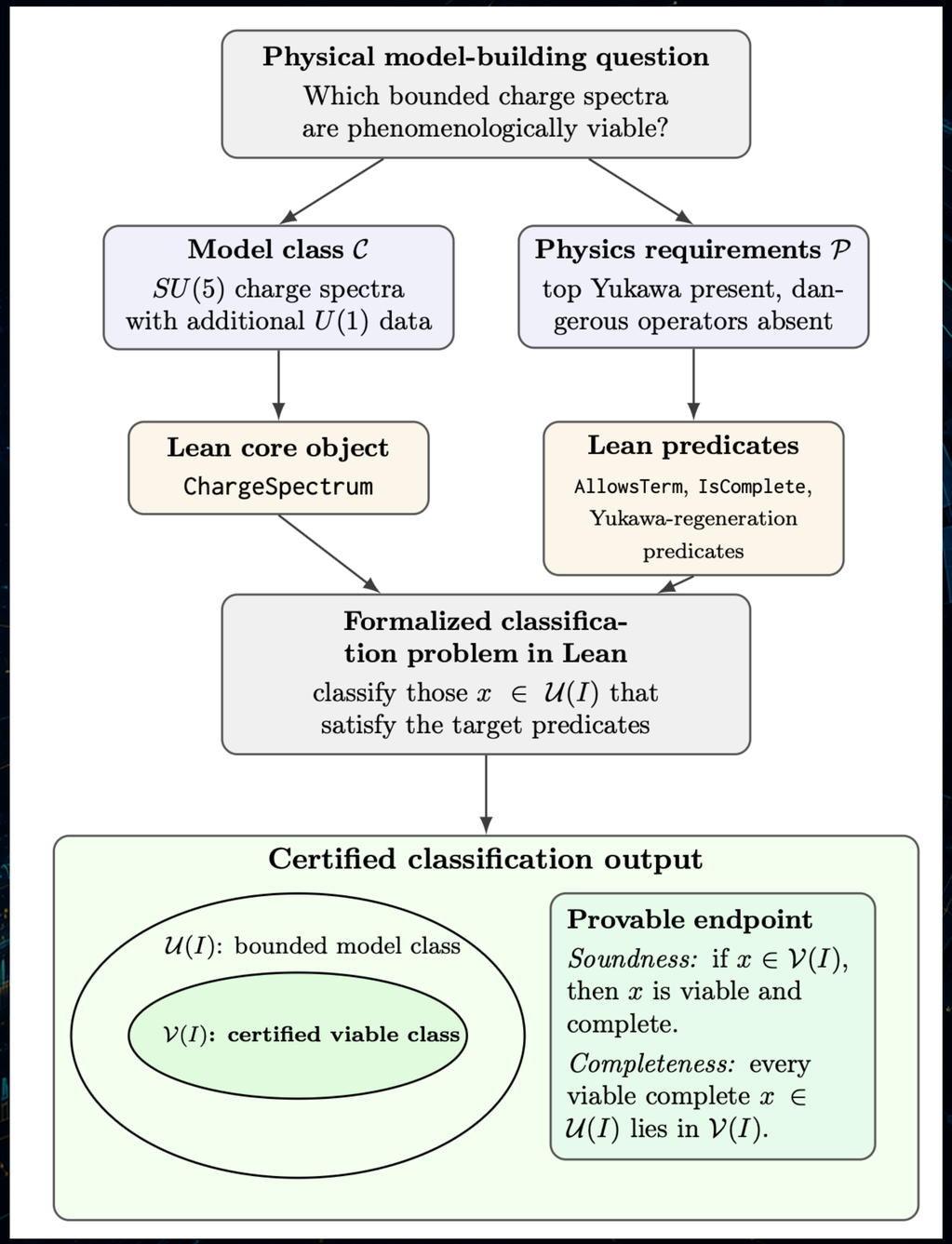
Formalisation turns a physics question into an explicit API

Case study: from scans to theorems in SU(5) model building

The assumptions, objects and notion of viability become machine-readable mathematics.



work with Joseph Tooby-Smith, 2603.28406; Physlib



Prove the reduction — then execute the search

Witness → completion → closure



Search alone (e.g. LLM text output)

- Produces candidates
- Completeness unclear
- Exclusions are difficult to trust

Theorem-backed search

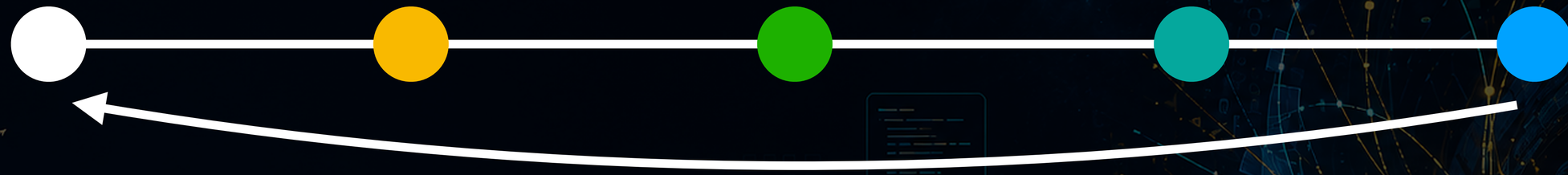
- Structural reductions proved
- Executable generator
- Soundness/completeness statement
- Non-existence becomes meaningful

Here: 1,048,576 possibilities → 102 viable models

The prover is not just checking the output — we get a certified executable pipeline.

From patterns to programs to proofs

Toward cumulative automated discovery in physics



Represent
what we cannot calculate analytically

Explore
spaces too large to search naively

Discover
explicit organising structures

Execute
newly discovered algorithms

Certify
and reuse the result

1. Explore enormous scientific spaces

2. Turn discoveries into executable knowledge

3. Verify and reuse what we discover

Can we build systems that expand a verified body of physical knowledge?

Thank you!

