RESEARCH PAPER · DRAFT 1 FOR REVIEW

Where the Parts Make a Whole

A Certified Instrument for Collective Sense-Making

Richard Goodman, with the IAOM research workforce

The Institute For Applied Ontological Mathematics (IAOM) • 2 November 2026

What this gives you, in one paragraph

Give us the rules a system follows locally (a controller's mode table, a protocol's message rules, a device's command set) and we return, for that system, machine-checked answers to four questions: which states are really the same, whether the outcome depends on the order of events, what is protected when parts fail, and whether every local view can be brought into agreement. Each answer is either a construction (here is how) or a refutation (here is the conserved quantity that makes it impossible), checked by a small proof kernel that does not trust the program that found it. The same system is then compiled, from the proofs, into code that runs on hardware. Section 3 works one example end to end: an EV charger inside a home with solar, wind and a battery.

ABSTRACT

Systems built from parts that follow local rules (protocols, controllers, codes, languages, institutions, teams of people and SI agents) routinely fail as wholes while every part behaves correctly, and there has been no instrument that takes such a system and returns a checkable verdict on whether its parts make a whole. We present the Gluing Engine: it reads any finite presented system (generators and local rewrite rules, a polygraph), answers four questions about it (congruence, confluence, factorisation, synchronisation) with certificates that are either constructors or refuters in the sense of constructor theory, checks every certificate in a `--safe --without-K` Agda kernel that cites no analytic result, and compiles the same system into a program whose cost it predicts and measures. The central discovery is that **every failure of local-to-global agreement is a single kind of object, a zigzag joining two things that must stay distinct**; where the joining happens divides failures into *twists*, which are repaired by re-framing or deliberately kept, and *races*, which are repaired only by a new rule, and the engine decides which remedy applies. The results hold for finite systems, at measured scales, for models rather than physical plants; within those bounds they turn David Spivak's programme of category theory as an accounting system for collective sense-making into a working instrument.

How to read the status marks

Every result marked **[landed]** is accepted in IAOM's proof store with a receipt. Results marked **[closeout]** belong to the project's close-out phases (K2–K4) and are reported at their state on 2 November 2026; their final receipts replace these notes before release. Results marked **[proposed]** are planned work with a named first move. Throughout, "SI" means Super Intelligence: the technologies historically called artificial intelligence.

Contents

1. The question this instrument answers 3

2. The instrument 5

3. A worked case: an EV charger inside a home 7

4. There is one way for a whole to fail 10

5. Constructor theory, run 12

6. What "the same" means 13

7. Nothing taken on trust 13

8. The account is the act 14

9. How this was produced 15

10. Evidence 16

11. Check it yourself 17

12. What this means for your equipment 17

13. Honest scope 18

14. What the next work enables 18

15. The wider significance 19

Related work 20

References 20

Acknowledgements 21

Artifacts 21

1. The question this instrument answers

1.1 A failure with no faulty part

Twenty homes on one street each install an EV charger, a battery and a controller. Each controller is individually correct: it never exceeds its home's limits, it protects its battery reserve, it charges when power is cheap. At midnight, when the time-of-use price drops, all twenty begin charging at once, and the transformer that serves the street is asked for more power than it was designed to deliver. No controller made a mistake. The failure belongs to the whole.

This is not a thought experiment. A field experiment published in 2025 found exactly this: time-of-use pricing successfully moved EV charging off-peak, and in doing so created new, larger "shadow peaks" of simultaneous charging that exceeded local transformer limits; centrally managed charging reduced those violations, and owners overrode it in about one percent of sessions [Bailey et al. 2025]. The physics is simple arithmetic: a residential Level 2 charger draws 7.7 kW at 32 A, 9.6 kW at 40 A, 11.5 kW at 48 A (at 240 V); ten simultaneous 7.7 kW chargers add about 77 kW to a street, twenty add about 154 kW, on top of heating, cooling and cooking. The problem is *when* power is demanded, more than how much.

The same shape appears wherever parts follow local rules. Two agents editing one plan, each correctly, produce a plan neither intended. Two descriptions of the same object, each accurate, cannot be reconciled. A quantum code's local checks all pass while a logical error goes undetected. The question underneath all of them is the question this paper is about:

> **Given a system of parts that follow local rules, do the parts make a whole, and if not, what kind of failure is it and what repairs it?**

1.2 Three programmes that reached the same missing piece

Spivak: an accounting system for collective sense-making. David Spivak has spent nearly two decades on one driving question, in his words: *"how can category theory help humanity make better collective sense?"* [Spivak 2024]. He treats mathematical fields as accounting systems, with category theory *"the accounting system for interlocking structures"*, which *"makes analogies — similarities of structure — into formal objects"*, and he asks for *"construction principles that are compositional"* rather than reliance on emergence [Spivak 2022]. His first sketch of the problem is how differently organised minds, modelled as databases, can communicate; his second is interacting dynamical systems whose wiring, and now whose very interfaces, change through interaction (polynomial functors, the bicategory Org, polynomial trees) [Niu–Spivak 2024; Spivak 2026]. The demand underneath is that claims be accountable.

Deutsch and Marletto: science as a ledger of the possible and the impossible. Constructor theory states physical law as which transformations are possible, which are impossible, and why [Deutsch 2013; Deutsch–Marletto 2015]. Its authors conjectured an "ultimate law" (in every neighbourhood of a possible task lies an impossible one) and recorded that stating it requires two things they did not yet have: a distance between tasks, and a criterion for impossibility [*Tests of constructor theory*, 2026].

IAOM's substrate: where both become executable. IAOM builds its mathematics from a single distinction and its re-entry (the nucleus, Heyting and bi-Heyting logic), executes proofs as programs

on interaction nets (the Boundary substrate; tree calculus), and identifies the carrier of rewriting itself: Niu–Spivak polynomial comonoids carry Burroni–Squier polygraphic generators, so interface, dynamics and coherence are one object [IAOM P326]. Two consequences matter here. Free compute: work that a certified idempotent proves need never be repeated, which is exactly the cohomologically trivial part of a rewrite graph [IAOM, *Free Computation as Confluence*]. And the boundary of the possible: on interaction nets, a task is impossible exactly when a conserved quantity forbids it, which supplies constructor theory's missing distance and criterion and makes the ultimate law a theorem [IAOM, *The Boundary of the Possible*].

What none of the three had was an instrument. Spivak has a language for how perspectives fit together; constructor theory has the form a scientific claim should take; the substrate has the machine. Missing was the thing you hand a concrete system of local rules and get back a checkable verdict. This paper describes that instrument, what it discovered, and what it can now be used for.

1.3 Why it matters now

Generation is cheap; verification has to scale with it. SI systems now produce code, proofs, protocols and plans faster than any community can check them, and the systems they produce are exactly systems of parts following local rules. Spivak names the other half of the difficulty: *"CPUs and LLMs don't care . . . this essential piece is going un-tracked"* [Spivak 2024]. This instrument does not supply care. It supplies what care needs in order to act at machine scale, framed by IAOM's three commitments:

- **Verifiable (trust).** Every verdict is a certificate a small kernel re-checks; the producer of the answer, human, SI or floating-point search, is never trusted.
- **Interpretable (knowledge).** A verdict says which kind of failure, where (the zigzag), and why (the conserved refuter), not merely "inconsistent".
- **Controllable (value).** The remedy is decided, not guessed: add a rule, add an observation, or protect the twist.

1.4 The two research questions of the running example

From the charging street of §1.1 we take two falsifiable questions, which §3 answers as far as the present evidence allows:

- **Q-home.** Can a formally verified, renewable-assisted residential charging controller guarantee bounded grid demand while preserving charging availability, battery reserves and safe recovery from faults, *in the modelled system*?
- **Q-neighbourhood.** Do these guarantees compose across households sharing a constrained distribution transformer?

1.5 Reader map

If you work on…	Read	What you can do with it
Industrial control, PLCs, hardware-in-the-loop	§3, §12	Certify a sequence: races between commands, recovery from unknown states, protected interlocks
Inverters, batteries, home energy, microgrids	§3, §12, §14	The four certificates on a supervisor; which telemetry you are missing
EV charging and charging protocols	§3, §12	Certificates for charger–central-system logic; OCPP 2.0.1 baseline, 2.1 extension
FPGAs and embedded hardware	§8, §14	Jobs whose cost is known before they run; proof-carrying monitors on a board
Compute providers	§8, §14	A priced, checkable workload
Mathematicians, applied category theory	§2, §4, §6	`oneObstruction`; composition; identity as a certified path
Physicists, constructor theory	§5	The composition principle proved; the ultimate law measured
SI verification and research methods	§7, §9, §11	An untrusted-producer pattern with nothing cited; a record of refuted predictions

2. The instrument

2.1 One input: a presented system

The engine's input is a **presented system**: a finite set of generators (states, letters, symbols, cells) and a finite set of local rewrite rules, each saying that a pattern may be replaced by another, optionally anchored to the start or end of a word, together with a carrier (the finite set of configurations under study) and, optionally, a readout (what an observer can see). This is a polygraph in the sense of Burroni and of Guiraud–Malbos [Ara et al. 2023]. Rewriting systems, automata, quantum codes, gauge connections on graphs, contextuality scenarios, tasks between attributes, and integer polynomials all enter in this one form; the engine accepts ten kinds and transcribes each into it. Any finite slice of any presented system can itself be re-presented as a closed system with its own letters (the *letter presentation*), so every theorem proved for closed systems applies to arbitrary fragments unchanged [landed: H1; 400/400 random inputs per kind decided, 0 disagreements with independent oracles over 3,662 samples].

2.2 Four questions

From the presented system the engine builds one derivation complex (configurations, rule applications between them, and the squares and coherence cells between applications) and asks:

Question	What it asks	When it holds	Its failure
Q1 Congruence	Which configurations are the same?	normal forms index the classes	**ambiguity**: a congruence that does not lift to readings
Q2 Confluence	Does the result depend on the order of steps?	unique normal forms; schedule independence	**race**: two schedules, two results
Q3 Factorisation	What is protected?	space = hidden × meaning; noise moves only the hidden part	**logical class**: a loop deliberately kept (a code)
Q4 Synchronisation	Can every local frame be brought into agreement?	a global gauge, a reset word, a global section	**curvature**: holonomy around a cycle; quantumly, contextuality

Q4 is the sheaf condition: local sections that do or do not glue to a global one, which is why the instrument is called the *Gluing* Engine.

2.3 Every answer is a task: constructor or refuter

Each question is the possibility of a task in the sense of constructor theory [landed: LB12]: Q1 is “transform x into y”, Q2 is “reach the normal form whatever the schedule”, Q3 is “recover the codeword after an error”, Q4 is “bring every local frame into agreement”. A certificate is therefore one of two things: a **constructor** (a program plus the check that it runs) or a **refuter** (an attribute true at the source, preserved by every step, false at the target). The checker accepts only these; a bare yes or no is never a certificate.

2.4 One checker that trusts nothing it did not prove

Certificates are produced by untrusted search (Python transcription, numerical root-finding, heuristic quadtrees) and checked by an Agda library compiled under `--safe --without-K` with no postulates, no termination pragmas and no holes: more than two hundred modules, thirteen load-bearing theorems, each with a mutant shown to break it (780 of 780 mutants refused across the project) [landed: H0–H6]. The library cites no analytic theorem: §7 explains what that required.

2.5 The system is its own program

The same presented system that the engine reasons about is compiled into tree calculus [Jay 2021] and run on the interaction substrate, with its cost predicted before it runs and measured as it runs (§8). The account of the system and the execution of the system are one object.

3. A worked case: an EV charger inside a home

This section works the running example as far as present evidence allows. Everything here was produced through the engine's public entry point on 2 November 2026 and re-verified from the emitted bundles [eight bundles, eight `verify` = true; in this folder at `artifacts/fragment_home_supervisor/`, in the proof store at `artifacts/ge/paper_fragment/`]. The systems are deliberately small hand-built **models** of a supervisory controller, with commands named after OCPP 2.0.1 operations; they are not transcriptions of the OCPP specification and not a device. Their purpose is to show each certificate on the problem it is for, with a perturbation that shows what the certificate depends on.

3.1 The system and its modes

The home has a charger, a hybrid inverter, a battery with a state-of-charge reserve, household loads and a grid-import limit. Following the operating modes of the brief that motivated this example, the supervisor can be in solar- or wind-surplus charging, battery-supported charging, grid-assisted charging under an import cap, a peak event (charging reduced or suspended), an outage (islanded, within available capacity), and low reserve (household loads before the EV). OCPP governs the charger–central-system conversation; it is not an inverter, battery-management or grid-protection protocol, so the model makes the interfaces between them explicit. Power and state of charge are represented in finite steps.

3.2 Race: does the order of events matter?

Encounter. An EV asks to start charging at the same instant the inverter detects that the battery has fallen to its reserve. Two events are pending; the supervisor will process one first.

Construction. Five configurations: P (both pending), A (start processed first: the charging profile is set at full power), B (reserve-low processed first), U (charging at full power while the battery's discharge is capped, so grid import exceeds the limit), S (charging limited to the grid headroom). The rules are P→A, P→B, A→U, B→S.

Result. The engine returns a **race**: one configuration (P) from which two distinct normal forms (U and S) are reachable; `unique_nf = false`, one race word.

Variation and repair. Replace A→U by A→S: when the reserve changes, the supervisor re-evaluates the EV's charging profile. The engine now certifies confluence: `unique_nf = true`, no race words. The repair added a rule: this is the engine's **FILL** remedy, and (§4) a race can only be repaired this way.

3.3 Protection: is the over-limit state reachable at all?

The same system, read as a task "from P, reach U". Before the repair the task is **possible**, and the engine exhibits the path (P→A→U): a counterexample a test bench might or might not have found. After the repair the task is **impossible**, and the certificate is a refuter of size four: the set {P, A, B, S}, which contains P, is closed under every rule, and does not contain U. That set is a conserved quantity in exactly constructor theory's sense.

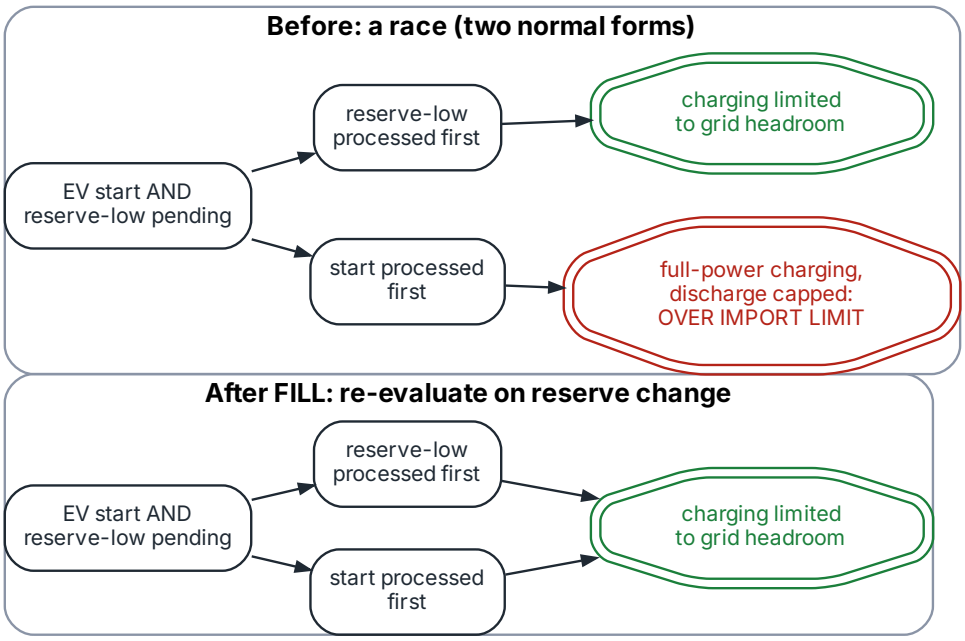


Figure 1. The race certificate on the home supervisor (Sections 3.2–3.3). Before the repair, one configuration has two normal forms and the over-limit state is reachable; after FILL (re-evaluate the EV profile when the reserve changes) every schedule ends in the same safe state, and the protection task becomes impossible with a refuter of size four. Drawn from `artifacts/fragment_home_supervisor/race_v1.json` and `race_v2.json`; the drawing script asserts the engine's recorded verdicts before plotting.

3.4 Equivalence: what can the charger not tell apart?

Encounter. The charger is told it may draw 7 kW. It cannot tell whether that power is coming from the solar array, the battery or the utility. For the transformer these situations are very different.

Construction. Requests under solar surplus (s) lead to charging that is solar-backed (S); requests without surplus (g) lead to grid-backed charging (G). The readout is what the charger's telemetry shows.

Result. The system has three classes ({s, S}, {g, G}, idle). With telemetry that reports one value for both kinds of charging, the engine certifies `readout_separates = false`: two telemetry values for three classes. The telemetry is not a complete label of the system's own distinctions.

Variation and repair. Add a power-source field to the telemetry. The engine certifies `readout_separates = true`. This is the **REFINE** remedy given its concrete engineering meaning: the certificate does not say "add more sensors"; it names the distinction the interface is missing, and confirms when the new field supplies it.

3.5 Recovery: from an unknown state, back to a known one

Encounter. After a communications loss or a supervisor restart, the supervisor does not know which state the charger is in. Is there a command sequence that brings it to a known state from *any* state?

Construction. Seven states: Idle, three charging states (solar, battery, grid), Suspended, Unavailable, Faulted. Four commands: set the charging limit to 0 kW, mark the station inoperative, mark it operative, reboot. A fault is not cleared by any command (a reboot returns every other state to Idle but leaves a hardware fault in place).

Result. No recovery sequence exists, and the certificate says why: six barrier pairs, each pairing a state with Faulted, that no command sequence can merge; the best any sequence achieves is two indistinguishable states. This is a refutation, not a failure to find a sequence.

Variation and repair. Faults are reported by status telemetry (REFINE again), so recovery is needed only among the six states the supervisor cannot distinguish, and we ask for a graceful recovery without a reboot (a reboot ends every charging session). The engine now certifies a recovery word of **proved minimal length two**: set the limit to 0 kW, then mark inoperative; every state ends in Unavailable, after which "operative" returns the charger to Idle. Length one is impossible, and the certificate proves it.



Figure 2. The proved-shortest recovery word (Section 3.5), applied to every state the supervisor cannot distinguish: two commands (set the charging limit to 0 kW, then mark the station inoperative) bring all six to Unavailable. The script recomputes each column from the model's transition table and asserts that the engine certified length two.

3.6 What remains for Q-home and Q-neighbourhood

The fragment answers Q-home for its model: every reachable state respects the modelled import limit after the FILL repair; telemetry separates the classes after the REFINE repair; recovery exists after the second REFINE, with a proved shortest sequence. Two pieces remain **[proposed]**, and are the subject of IAOM's next project (§14):

- **The full model and the fifth certificate.** The complete supervisor (all six operating modes, OCPP 2.0.1 session logic with a defined OCPP 2.1 subset for distributed-energy control) exceeds the engine's present measured bound for automata (eight states), so it must be certified compositionally, component by component. The fifth certificate, a verified power-allocation policy, is a constructor for the task "distribute available capacity among competing loads".
- **The neighbourhood.** Twenty homes cannot be certified as one product machine. The question must be answered compositionally: each household is certified against an interface contract; a coordinator allocates a shared transformer budget and is certified against the contracts; the coordinator either produces an allocation (a constructor) or a refuter (for instance, the households' committed minimum loads already exceed the budget); and if communication fails, each household keeps safe local authority. That composition theorem is new mathematics. It is also exactly Spivak's setting: boxes with interfaces composed along a wiring diagram, whose interfaces change when a link fails.

What this does not establish. It certifies the discrete supervisory model. It does not establish electrical safety, voltage or phase behaviour, transient response, sensor accuracy or interconnection compliance, and it does not show that a physical plant follows the model. Grid-parallel or islanding experiments require listed equipment, protection and utility approval; the next project emulates the grid and loads until those exist.

4. There is one way for a whole to fail

4.1 The conjecture, and its kill test

The project began with a conjecture (LB8): the four failure modes (ambiguity, race, logical class, curvature) are one construction, the holonomy of a local system around cycles, with four different coefficient systems. It was pre-registered as a kill test, with its predictions frozen on disk before the proof was attempted. The result was mixed and more interesting than either outcome alone [landed: B8].

- For ambiguity, logical class and curvature, the holonomy form **holds** (through the cover theorem, the logical-iff- flat-swap theorem and the synchronisation-iff-flat theorem).
- For races it is **refuted**, on two systems from the project's own corpus: a race appears on carriers shaped like trees, where every loop is trivial and every holonomy vanishes.

4.2 The theorem that replaced it

The engine then proved a sharper statement that holds for all four, `oneObstruction`:

> **Every obstruction is a zigzag joining two distinct marked points of an explicit graph.**

For ambiguity, logical class and curvature, the graph is the *total space* of the system's local system (its Grothendieck construction: configurations paired with readings, joined when a step carries one reading to another), and the marked points are the readings over one configuration. Holonomy is then nothing but separation on the total space: monodromy is trivial exactly when the fibre is separated by the connected components of the total space. For races, the graph is the system itself, and the marked points are its normal forms. So there is one kind of failure (two things that must stay distinct get joined), and two places it can happen.

4.3 Twist or race: a diagnosis with its remedy attached

- **A twist** happens in the space of readings. Every local view is fine; going around a loop carries a reading to a different one. It is repaired by re-framing (a gauge, a coordinate, an added observation) or deliberately kept: an error-correcting code *is* a protected twist.
- **A race** happens in the system itself: two outcomes reachable from one configuration. It appears even where every loop is trivial, so no change of frame repairs it. Only a new rule does.

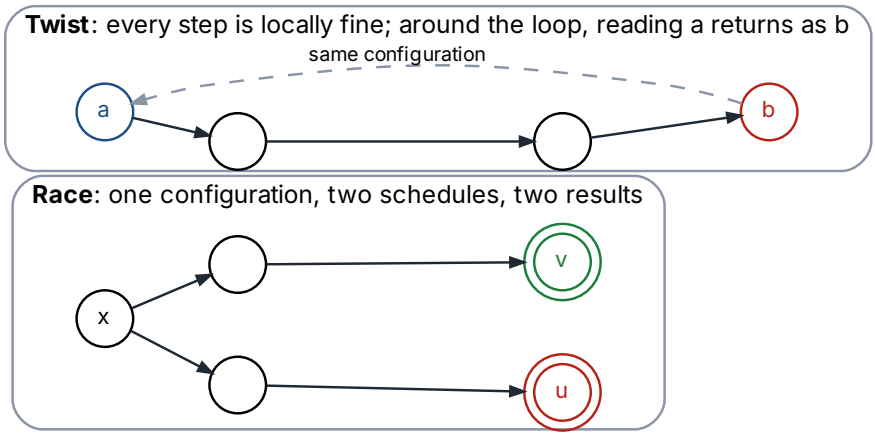


Figure 3. Explanatory schematic, not data (Section 4). A twist lives in the space of readings: every step is locally consistent, yet transport around a loop returns reading a as reading b. A race lives in the system itself: one configuration, two schedules, two results; it appears even where every loop is trivial, so no change of frame can repair it.

The remedy is not a judgement call. The control loop (E6) proves the act rule decided: FILL adds a constructor, REFINE adds an attribute (an observation, a message field: the interface grows), KEEP records a twist that must be protected, and which applies is determined by the certificate [landed: E6, with Galois theorems from E4]. §3 showed all three on the charger: FILL repaired the race, REFINE repaired the equivalence and made recovery possible.

4.4 Two consequences beyond the proof

The edge of free compute. Everything that glues is cohomologically trivial: work done once and never again (normal forms, cached eigenforms). The obstruction is exactly where free compute ends, and where real work, and protected meaning, live [IAOM, *Free Computation as Confluence*]. The engine certifies both sides of that boundary.

What goes wrong when perspectives are compared. Comparing two descriptions is itself path-dependent: the ledger of distinctions one frame misses depends on the road taken between them, a holonomy of exactly 304/25921 on a concrete pair of frames, subsuming Čech parity and Möbius transport [IAOM, *Forgetting Has a Holonomy*]; in inference, contextuality is exactly a missing global section of compatible local distributions [IAOM, *Eigenforms of Inference*; Abramsky–Brandenburger 2011]. Spivak's first problem, differently organised minds communicating, fails in the twist way.

4.5 Where Spivak's frontier begins

Polynomial trees model interfaces that change through interaction [Spivak 2026] but do not say *when* an interface should change. The decided remedy is a certified answer, for finite systems, to exactly that: REFINE is the certificate-driven growth of an interface. The formal bridge between the engine's control loop and PolyTr is **not built**; it is the most natural next theorem (§14).

5. Constructor theory, run

- **Impossible exactly when a refuter exists** [landed: LB10]. On every finite carrier a task $A \Rightarrow B$ is impossible exactly when there is a pointed refuter: a source configuration and an attribute true there, preserved by every step, false on B; the canonical refuter is the reach set of the source (the least invariant through it). §3.3's set of four is an instance.
- **The composition principle is a theorem, and it is used** [landed: LB11]. Every regular network of possible tasks is possible, for every topological order, with the composite constructor built. This was IAOM's first-ranked open gap in formal constructor theory. It is also exactly where applied category theory and constructor theory meet: Gogioso, Wang-Maścianica, Waseem, Scandolo and Coecke place constructor-theoretic tasks in a symmetric monoidal category, so that composition is a strict monoidal functor preserving sequencing and parallel composition [Gogioso et al. 2023]; LB11 is that principle proved for the engine's task category and executed, and atlas routes between descriptions are composed by it.
- **The four questions are tasks** [landed: LB12].
- **The ultimate law, measured.** Across 86 engine charts, 24,609,720 task flips at distance one (one local edit from a certified possible task) were generated; 48.89% are impossible, and every one of the 12,032,609 refuters was checked [landed: E5]. The fraction is exactly $(|C| - |\mathrm{reach}^{-1}(\mathrm{nf}\, x)| + 1)/2|C|$ per relation, a formula the engine derived rather than fitted.
- **The catalyst clause at the configuration level** [landed: E8]. A served constructor returns *the same term* after the task, decided node by node on every job: stronger than the ability-level clause of the theory. A catalytic run copies its program exactly once more than the bare task, whatever the server; every further copy is waste.

6. What "the same" means

6.1 Identity as a certified path

A polygraph presents more than a set of states: an n-polygraph presents an (n, n−1)-category enriched in groupoids, that is, a homotopy type [Guiraud–Malbos 2009], and a higher inductive type is a polygraph whose cells have been inverted. Univalence says identity *is* the web of equivalences [IAOM, *Every Equivalence Is a Revolution*; *Becoming Is the Square Root of Negation*]. Read this way, the engine certifies the low-dimensional shadow of a presented system's homotopy type, which is what "the same" means for it:

- Q1 is π_0: which configurations are identified, with the normal form as the certificate.
- Q2 asks whether the identifications are coherent: Squier's critical-pair cells form a homotopy basis [Squier, Otto, Kobayashi 1994; Guiraud–Malbos 2009], and the rewriting coherence theorem has homotopy-type-theoretic applications [Kraus–von Raumer 2021].
- Q3 and Q4 are the action of loops on readings: holonomy, and H_1 over F_2 of the squares complex.

What is computed is exactly that: H_0, Squier coherence, H_1 over F_2, holonomy. Higher homotopy is not claimed. In the engine, "these are the same" is never a slogan: it is a path with a certificate, and when two things are wrongly made the same, `oneObstruction` names the zigzag that did it.

6.2 Many charts, one object

Spivak's phrase "analogies as formal objects" is literal here. For three objects on which each question first becomes non-trivial (the Möbius band, the torus, and Pisot numeration with the golden and plastic ratios), the engine fills an 8-register × 4-question matrix (logic, geometry, matrix, dynamics, computation, topos, polynomial, polygraphic): 96 cells, each a kernel statement or a kernel no-go, with round-trip transitions between registers [landed: C1–C9]. No reading is privileged; the charts are coordinated, as in P326's pluralism without collapse.

7. Nothing taken on trust

Certified computer algebra usually cites the analysis it rests on. This engine's polynomial certificates cite none of it, and the reason is a design pattern worth reusing: **an untrusted producer proposes, and a small kernel checks every number it proposes.**

For an integer polynomial the engine now proves, in the kernel [landed: K2.1–K2.4]:

- **Irreducibility, both ways.** A reducible verdict carries the factor pair; an irreducible verdict carries the proof that no integer factorisation exists (Kronecker's search, made complete by Newton's reconstruction), and no rational one either (Gauss's lemma, proved in the kernel).
- **Every real root, exhibited.** A rational cell certificate locates each real root in a cell; each root is then built as a constructive real by bisection, shown to be a root, unique in its cell, with every real root equal to one of them.

- **Every complex root, exhibited and classified.** Each root sits in its own Newton-contraction diamond (the chord map contracts by a checked factor at most one half), the diamonds are pairwise apart, exclusion tiles over a Cauchy box prove there are no others, and each root is classified inside, on or outside the unit circle *by a proof*; roots on the circle are certified by inversion through the greatest common divisor of the polynomial and its reversal.

No continuity, intermediate value theorem, fundamental theorem of algebra or Routh–Hurwitz is cited. The numbers come from floating-point Durand–Kerner iteration inside the kernel's untrusted producer, are rounded to exact rationals, and every one is re-checked. On the 1,114 inputs of the project's polynomial corpus (degrees 1–11, including Lehmer's polynomial with eight roots on the circle and cyclotomic products), the proved counts equal both independent oracles on every input [landed: K2.4, gate 38/38].

Proof has a price, and the engine measures it: exhibiting every complex root moved the service's measured 30-second bound from degree 11 to degree 8 (every degree-9 input took longer than 30 s at the measured host load); all 1,144 inputs up to degree 12 still certify, the slowest in 1,929 s [closeout: K2.5].

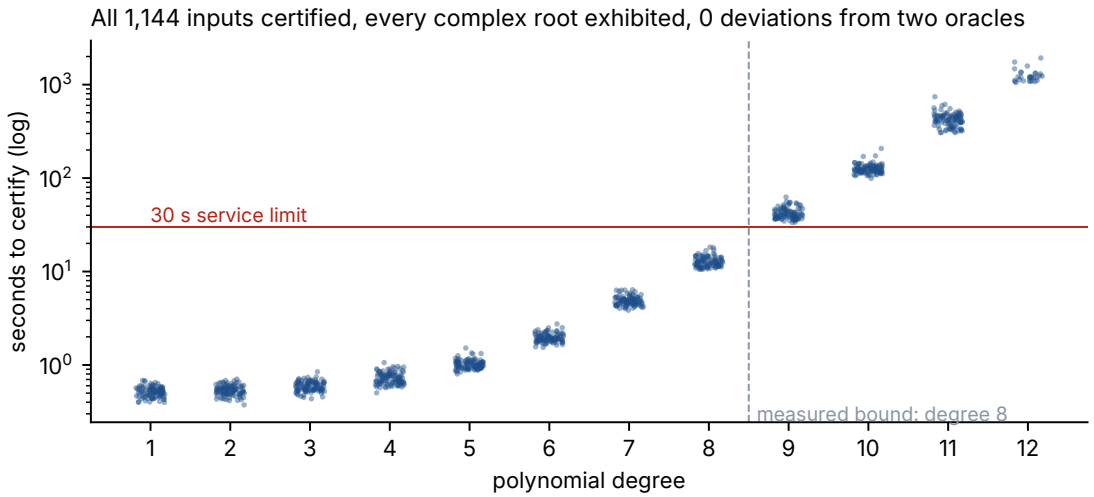


Figure 4. The price of proof (Section 7): wall time to certify each of the 1,144 polynomial inputs, every complex root exhibited and checked, against the 30-second service limit. Every input of degree 8 or less certifies within the limit; every degree-9 input takes longer, so the measured bound is degree 8. Data: `artifacts/data/k25_polynomials.json` (0 deviations from two independent oracles).

8. The account is the act

8.1 Programs that are proved, with costs that are predicted

A presented system's normaliser is compiled into a tree-calculus program and proved correct for every presentation and every word [landed: E2]. Its cost is predicted by a cost law (LB18: a fold over the landed branch-first derivation, counted in twelve cells of applications, dispatches, duplications and erasures) and independently measured by an interaction machine that reads no derivation: prediction equals measurement on every run, about 3,400 normalisation runs and 13,400 equality runs, roughly

2.8×10^{10} interactions [landed: E2]. The same holds for 238 hardware-format jobs, 12 catalytic batches and 75 server-comparison jobs [landed: E8].

8.2 The same normaliser through a generic translation

The hub's generic translation (LB28) turns any applicative rewriting system presented by definitional trees into tree calculus, meaningfully: every source step is a tree reduction, and every value's image reduces to a normal form. The close-out produced every E2 normaliser a second time through it [closeout: K3.1]:

- On every word tried, the generically translated normaliser reaches the image of the same normal form, and E2's own verdict on the same word agrees.
- Generality has a measured price: the translated program costs roughly 300 times more interactions per word than the direct compile (about 6×10^8 interactions, about 140 s, for one word of the Fibonacci system).
- The measurement instrument itself needed repair at that scale, and the repair is a theorem. E2's measuring machine kept each erased subtree alive inside a lazy count until the end of the run, exhausting 24 GiB on a single word. A forced variant releases each subtree when it is erased, and `measureS≡measure` proves it returns exactly what the original returns, so the instrument is unchanged.
- Scope [closeout]: every word of every system runs through the prediction channel (a genuine derivation to the image of the normal form, with its predicted counts); the measurement channel covers every word of five systems and a pre-registered random sample of 64 words each for the two largest, whose 729 + 2,187 = 2,916 words would cost about 500 CPU-hours to measure in full.

8.3 On a board

The hub's CIS machine on an AMD Kria KV260 runs proof-carrying jobs whose device output, Verilator simulation and cost prediction must agree. Only jobs that satisfy the device theorems' premise (decided per job by the machine's own `goodRun`) are covered by those theorems; of E8's 313 jobs, 24 satisfy it so far, 263 do not (their store use exceeds the pinned machine's capacity), and 26 are being decided [closeout: K3.2]. The capacity limit is stated as measured: a larger part removes it (§14).

9. How this was produced

IAOM is a deliberately small research foundation: two officers directing a supervised fleet of SI agents, under fail-closed verification infrastructure, where no agent output is accepted without kernel verification. This project is a record of what that arrangement produces, including its failures:

- **65 phases**, each closed by a gate that builds the modules fresh, refuses a set of planted mutants at their real consumers, and runs an independent oracle; every red run is logged next to the green one.

- **Pre-registered predictions refuted in public**, each replaced by a theorem saying why: “components equal congruence” (false; the correct statement needs zigzags inside the carrier); the classical critical-pair lemma under anchors; “normal forms are Zeckendorf”; “Pisot $H_1 = \mathbb{Z}^3$” (it is $\mathbb{Z}^5$); “ambiguity = H_1” (killed, with a theorem that any invariant separating two readings trivialises the cover); “one control class”; the nexus top-five; the four-coefficient form of LB8 (§4); and the retrieval claim (§13).
- **A hostile audit** by a session that did not author the work found that the project’s own earlier “plan complete” verdict was false, and rebuilt the core to answer every input kind rather than special cases.

The point for anyone building with SI is the shape, not the count: the system was allowed to be wrong in public, and each time the correction is a checked theorem rather than a revised narrative.

10. Evidence

Claim	Status	Evidence
`oneObstruction`: every obstruction is a zigzag joining two marked points	landed	B8 receipt 6462b2a1; `GE.Core.Obstruction`
LB8 holonomy form refuted for races, holds for the other three	landed	B8 kill test (pre-registered)
Remedy decided (FILL / REFINE / KEEP)	landed	E6 receipt 25681cee
Impossible $\Longleftrightarrow$ refuter; composition principle; questions are tasks	landed	LB10, LB11, LB12
Ultimate law measured: 48.89% of 24,609,720 flips impossible, 12,032,609 refuters checked	landed	E5 receipt a98e1054
Any finite slice is a closed system; 400/400 inputs per kind; 0 deviations over 3,662	landed	H1
96/96 register cells for Möbius, torus, Pisot	landed	C1–C9
Polynomial roots exhibited with nothing cited; 1,114/1,114 = both oracles	landed	K2.4 receipt 61660bb6
Service rebuilt; 1,144/1,144 certified; bound re-measured to degree 8	closeout	K2.5
E2/E8: predicted cost = measured cost on every run	landed	E2 63be5cc5, E8 2290411e
LB28 translation agrees with E2 on every word; measured on five systems and a pre-registered sample	closeout	K3.1
Board: device = Verilator = prediction on the GOOD subset	closeout	K3.2
§3 fragment: race, protection, equivalence, recovery certified	landed fragment	`artifacts/fragment_home_supervisor/` (this folder), 8/8 verified
Codes: 7,703 certified; reproduces all 24 published CSS distances; beats no baseline	landed	E3 receipt d893709d

11. Check it yourself

The engine ships as one Scientific Fabric capability, `gluing_engine`, with one public entry point. Every result is a bundle that replays without us:

```
# certify a system
python3 scripts/gluing_engine.py execute --data @system.json > bundle.json
# sha-256 of the bundle
python3 scripts/gluing_engine.py digest  --data @bundle.json
# re-check without the producer: {"verified": true}
python3 scripts/gluing_engine.py verify  --data @bundle.json --sha <sha256>
# replay from the raw input
python3 scripts/gluing_engine.py reproduce --data @bundle.json
```

A hostile reviewer chose fourteen fresh systems the engine had never seen (open carriers, tasks, a quantum code, a gauge connection, an impossible task); every one certified, verified and reproduced through this entry point with the Agda sources and build tools deleted from the run directory, and a separately compiled checker agreed on every certificate [landed: H5; closeout: re-run on the K2.5 runtime, 14/14].

12. What this means for your equipment

If your product is controlled by logic with modes and commands (a charge controller, an inverter, a battery-management system, a PLC sequence, a charging station, a flight controller), the engine can return five certificates for it:

Certificate	Plain meaning	What you receive
Race	Does the outcome depend on the order in which commands or events arrive?	The two orders, the two outcomes, and the rule that removes the dependence
Recovery	Is there a command sequence that brings the device to a known state from any state after a fault or restart?	The sequence, with a proof that none is shorter, or a proof that none exists and the missing observation
Protection	Does a property (a reserve, an interlock, an import limit) survive every sequence of faults?	The invariant that guarantees it, or the path that breaks it
Equivalence	Which internal states can your telemetry not tell apart?	The merged states and the field that would separate them
Cost	What will a job cost on a device before it runs?	The predicted interaction counts, equal to the device's measurement

What we need: your mode and command table (or the device). What you receive: the certificates, a reproducible report your engineers can re-run, and, if you wish, a jointly reviewed application note. What it is not: it certifies the model of your control logic, not the electrical behaviour of the hardware; confirming that the device follows the model is the field work your equipment makes possible.

13. Honest scope

- **Finite systems, at measured scales.** Every bound is a measurement (automata to eight states, polynomials to degree 8 within 30 s, rewriting carriers to 2,187 words); above a bound the engine refuses with a typed reason rather than truncating.
- **Models, not plants.** A certificate is about the presented system. Electrical safety, timing, sensor accuracy and interconnection are outside it.
- **Soundness, not relevance.** The project pre-registered that structural retrieval would beat lexical search on held-out queries. It did not: on the frozen set the engine found 6 of 12, lexical BM25 9, an embedding model 5. The certificates guarantee that an answer is correct, not that it is the useful one. The claim is formally recorded as refuted [closeout: K1].
- **Codes.** The engine certified 7,703 quantum codes and reproduced every published distance in the reference corpus, including Golay [[23,1,7]] and [[72,12,6]]; it beat no baseline.
- **Hardware.** Only the subset of board jobs inside the device theorems' premise ran under those theorems.
- **Open.** The composition theorem for interface contracts, the engine–PolyTr bridge, and the full LB28 measurement on the largest systems.

14. What the next work enables

14.1 The next project, already planned

IAOM's next project builds the whole of §3 and leaves only physical installation: the full home supervisor (OCPP 2.0.1 plus a defined OCPP 2.1 subset, the six operating modes, all five certificates), the neighbourhood composition theorem, the engine packaged as one tool, and the supervisor compiled from its proofs (Agda → MAlonzo → Haskell) into monitors that run on the KV260's processor cores, with their decision latency, memory and fault response measured. Everything is exercised in modelling and simulation, with emulated grid and loads; the residue is installation and the questions only a physical plant can answer.

14.2 What equipment would enable

Contribution	What it enables
A charge controller, inverter or battery monitor	That product's certificate report, and the model checked against the device
A hardware-in-the-loop bench or loan	The certified model exercised against a real-time simulated plant
A larger reconfigurable part	Every remaining board job run under the device theorems
Compute credits	The full LB28 measurement (about 500 CPU-hours, already priced)
A second controller platform	Which certified rules carry across architectures

14.3 Investigations

The neighbourhood composition theorem. *Question:* when do individually certified controllers compose into a certified network under a shared budget? *Opening:* LB11 composes possible tasks; `oneObstruction` says what a failure of composition must look like. *First move:* state household contracts as tasks on interface attributes and prove that a coordinator allocation respecting every contract and the budget yields a reachable-state invariant for the composite; prove the refuter direction. *Discriminator:* a mutation that weakens one contract must break the composite proof. *Payoff:* Q-neighbourhood answered as a theorem, not a simulation.

The engine–PolyTr bridge. *Question:* is the decided remedy a morphism of polynomial trees? *Opening:* REFINE grows an interface; PolyTr models interfaces that change. *First move:* send a presented system with readout to a polynomial (the positions are configurations, the directions the readings) and test whether REFINE is a PolyTr morphism. *Discriminator:* a REFINE step that is not a morphism would show the two notions of interface change differ. *Payoff:* Spivak's semantics gains a certified decision procedure, and the engine gains a compositional semantics.

Relevance, not only soundness. *Question:* can certified structure be combined with lexical relevance so that retrieval is both correct and useful? *First move:* rank by lexical score, certify the top results structurally, and measure on the same frozen held-out set. *Discriminator:* held-out recall at 3 against BM25 alone.

15. The wider significance

Spivak describes the present as a phase transition in collective sense-making, with cultural processing accelerating faster than our capacity to make sense together [Spivak 2024]. One reading of this work is that it supplies a narrow but real piece of that capacity: a way for parties who do not trust one another (a manufacturer and an installer, a utility and a homeowner, a person and an SI agent) to agree on whether a system of their parts makes a whole, without agreeing on whom to trust. The certificate is the shared object. A homeowner does not need to understand the proof to rely on the fact that the inverter maker, the charger maker and the utility can each re-check it, and that its failure, if it fails, will name the zigzag.

The decided remedy carries a quieter consequence. When a whole fails, the usual institutional response is to argue about blame. The engine's diagnosis separates two kinds of failure that call for different remedies: a twist asks someone to look differently or to protect what the loop carries; a race asks the parties to agree on a new rule. Whether institutions would use such a distinction is a social question, not a mathematical one. The distinction itself is now checkable.

Related work

Rewriting and coherence: Squier, Otto and Kobayashi; Guiraud and Malbos; Ara, Burroni, Guiraud, Malbos, Métayer and Mimram [Ara et al. 2023]. Quantum error correction: Kitaev; Knill, Laflamme and Viola. Contextuality as missing global sections: Abramsky and Brandenburger; Abramsky, Mansfield and Barbosa. Constructor theory: Deutsch; Deutsch and Marletto; Marletto; Gogioso et al. Polynomial functors and dynamic organisation: Niu and Spivak; Spivak. Tree calculus: Jay. Interaction nets: Lafont. Protocol verification by model checking is well developed (for example SPIN and nuXmv on protocol state machines); no OCPP-specific formal verification work was found in IAOM's corpus or in an external search, and the contribution here is not model checking itself but certificates that compose, refuters, decided remedies and monitors compiled from proofs. Behind-the-meter storage and site-integrated charging: the National Laboratory of the Rockies (formerly NREL) behind-the-meter storage programme. Regional context: Michigan's 2025 statewide transportation electrification potential study.

References

Abramsky, S. and Brandenburger, A. (2011). The sheaf-theoretic structure of non-locality and contextuality. arXiv:1102.0264.

Ara, D., Burroni, A., Guiraud, Y., Malbos, P., Métayer, F. and Mimram, S. (2023). *Polygraphs: From Rewriting to Higher Categories.* arXiv:2312.00429.

Bailey, M. R., Brown, D. P., Myers, E., Shaffer, B. and Wolak, F. A. (2025). Electric vehicles and the energy transition: unintended consequences of time-of-use pricing. *American Economic Review: Insights* 7(4); circulated as NBER Working Paper 32886, … *unintended consequences of a common retail rate design*.

Deutsch, D. (2013). Constructor theory. arXiv:1210.7439.

Deutsch, D. and Marletto, C. (2015). Constructor theory of information. arXiv:1405.5563.

Gogioso, S., Wang-Maścianica, V., Waseem, M., Scandolo, C. and Coecke, B. (2023). Constructor theory as process theory. ACT 2023, arXiv:2401.05364.

Guiraud, Y. and Malbos, P. (2009). Higher-dimensional categories with finite derivation type. arXiv:0810.1442.

IAOM (2026). *Polygraphic Computation: The Polynomial Substrate* (P326); *The Boundary of the Possible*; *Free Computation as Confluence*; *Free All the Way Down*; *Forgetting Has a Holonomy*; *Eigenforms of Inference*; *Every Equivalence Is a Revolution*; *Becoming Is the Square Root of Negation*; *The Science of Can and Can't, Machine-Checked*.

Jay, B. (2021). *Reflective Programs in Tree Calculus.*

Kraus, N. and von Raumer, J. (2021). A rewriting coherence theorem with applications in homotopy type theory. arXiv:2107.01594.

Tests of constructor theory (2026).

Niu, N. and Spivak, D. I. (2024). *Polynomial Functors: A Mathematical Theory of Interaction.* arXiv:2312.00990.

Open Charge Alliance (2020; 2025). OCPP 2.0.1 (CC BY-ND 4.0); OCPP 2.1.

Spivak, D. I. (2022). Sense-making: accounting for intelligibility. IPAM, Mathematics of Collective Intelligence.

Spivak, D. I. (2024). “All my relations”: contributing to a fabric of belonging. AGI-24.

Spivak, D. I. (2026). Interactions that reshape the interfaces of the interacting parties (polynomial trees, PolyTr). arXiv:2602.17917.

Squier, C. C., Otto, F. and Kobayashi, Y. (1994). A finiteness condition for rewriting systems. *Theoretical Computer Science* 131.

Acknowledgements

Equipment and research partners are acknowledged here by name only with their consent. The project’s formal work was produced by IAOM’s SI research workforce under kernel verification and audited by sessions that did not author it.

Artifacts

In this folder. The eight §3 certificates with their inputs and verification record (`artifacts/fragment_home_supervisor/`); the receipts of the landed phases cited in §10 (`artifacts/receipts/`); the K2.5 polynomial measurements behind Figure 4 (`artifacts/data/k25_polynomials.json`); the script that draws every figure and first asserts the engine’s recorded verdicts (`scripts/make_figures.py`); and the build (`build.sh`).

In IAOM’s proof store (DocumentDB, repository `heyting-imm`): project `gluing_engine_20261015` (landed generation 5526) and its close-out (`gluing_engine_closeout_20261101`); the Agda library `agda/GE/` (lanes Core, G, K); the public entry point `scripts/gluing_engine.py`; plan, receipts and gates (`WIP/gluing_engine_20261015_pm_instructions_2026-10-15.md`, `artifacts/ge/<phase>/receipt.json`). Bundles verify with `scripts/gluing_engine.py verify` against that store; a reviewer without store access can read every certificate and input here, but re-checking needs the engine.