KNWM: A KNOWLEDGE-NEURON ARCHITECTURE FOR WORLD MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper introduces the Knowledge-Neuron World Model (KNWM), an explicit world model inspired by synaptic plasticity and neural signal transmission. KNWM organizes object properties, scientific laws, events, and problems using recursively nested knowledge neurons, stored in corresponding knowledge layers and connected within and across layers through shared knowledge elements. During learning, KNWM proposes candidate relations from changes in object properties over time and through interaction, and tests the influence of variables through active interventions in controlled simulations. Validated knowledge is stored and updated as new evidence becomes available. During problem solving, task-driven knowledge activation combines scientific computation with event simulation to support prediction, counterfactual reasoning, and multi-step planning. Basic knowledge and selected composite knowledge structures serve as semantically explicit signals, distinct from the text tokens of language models. Scientific knowledge neurons provide executable relations and a basis for calculation; event knowledge neurons describe state transitions and resource requirements; and problem knowledge neurons organize objectives and relevant knowledge. Controlled experiments demonstrate law learning, knowledge reuse, and numerical reasoning within planning, while revealing how knowledge validation affects reliability and task completion. By connecting knowledge accumulation with world simulation, KNWM offers an approach to building inspectable, compositional, and continually extensible world models.

1 INTRODUCTION

Learning changes connections between biological neurons: motor learning in mice can form and stabilize dendritic spines, while neural communication uses action potentials and neurotransmitter release (Xu et al., 2009; Gerstner et al., 2014). These observations motivate a software world model in which learning establishes knowledge units and connections, and problem solving activates and combines them. The analogy concerns connection plasticity and signal propagation, not a one-to-one biological replication.

The central question is how knowledge acquired from observations can accumulate and remain usable in later prediction and decision-making. KNWM addresses this through recursively nested knowledge slots, connected layers, and shared semantic elements linking scientific relations, event descriptions, and problem goals. Learning creates or updates these structures; problem solving follows their connections and executes computations and simulations under explicit applicability conditions.

KNWM organizes world-model capabilities around explicit laws: explaining state changes through physical properties, predicting subsequent states, comparing simulated actions, and evaluating counterfactual conditions. Retaining the knowledge, conditions, and numerical dependencies behind an answer makes its reasoning grounds inspectable. The research contribution concerns how knowledge is learned, connected, reused, and updated to support these capabilities.

The main contributions are:

1. A knowledge-neuron representation with recursive slots and layered storage, connecting basic and composite knowledge through shared semantic signals.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2. A mechanism for learning, validating, and updating knowledge from observations and interactions, using prior knowledge to aid new law learning and linking the results to selective activation, scientific computation, and event simulation for prediction and planning.

3. Controlled evaluations of world-model capabilities, including law learning, prediction, causal and counterfactual reasoning, composition, and planning, with validation, noise, difficulty, and equal-information comparisons.

2 Related work

World models. Learned environment models support internal simulation and control (Ha and Schmidhuber, 2018; Hafner et al., 2025). KNWM pursues prediction and planning through persistent, executable knowledge records that retain the relations and applicability conditions used in computation.

Knowledge representation. Frames already organize knowledge through slots, nested structures, and connections (Minsky, 1974). KNWM connects such explicit structures to learning, updating, and execution. Its software knowledge neurons differ from the internal Transformer neurons identified by attribution analysis in Dai et al. (2022). Retrieval-augmented generation conditions language output on retrieved documents (Lewis et al., 2020); KNWM additionally executes retrieved relations and simulates event effects.

Law learning and causality. SINDy identifies dynamical relations through sparse regression (Brunton et al., 2016); PySR searches symbolic expressions and optimizes their parameters (Cranmer, 2023). KNWM uses formula induction within a learning-to-use process that stores checked relations and connects them to later tasks. Controlled interventions and numerical fitting serve different purposes: intervention evidence supports influence judgments, whereas fitting estimates a relation's numerical form (Pearl, 2009).

Planning. STRIPS supplies action preconditions and effects (Fikes and Nilsson, 1971); SHOP2 combines task decomposition with symbolic and numerical computation (Nau et al., 2003). Neuro-symbolic skills also connect operators, samplers, policies, and search (Silver et al., 2023). These provide precedents for KNWM's event representation and numerical planning. This paper studies how scientific and event knowledge connect learning to simulation and how the resulting knowledge changes decisions. Its experiments distinguish these effects from those of conventional computation and indexing.

3 KNWM architecture and layered knowledge base

KNWM connects knowledge learning, layered memory, and knowledge use (Figure 1). Learning creates records and connections; reasoning activates and combines them for computation and simulation.

3.1 Overall architecture

KNWM observes changes in objects and their properties over time and through interaction. Where it can control a simulator, it changes candidate variables and compares their effects on other properties. The learning module identifies relevant properties and induces relations. Stored formulas, mathematical structures, and composition rules supply candidates and foundations for learning more complex knowledge; learning can therefore build from basic to more complex relations.

Learning stores both a resulting formula and its constituent properties and relations in the appropriate layers, connecting them to existing knowledge. New content creates records; corresponding existing knowledge enters the layer's update or merge process.

Problem knowledge neurons organize conditions, goals, constraints, and subproblems. Scientific neurons provide applicable quantitative relations for prediction and event-cost calculation. Event neurons describe conditions, state changes, and subprocesses for simulation and planning. Thus, a learned relation can change an event's predicted consequences and the selected plan. Temporary states remain separate from reusable knowledge.

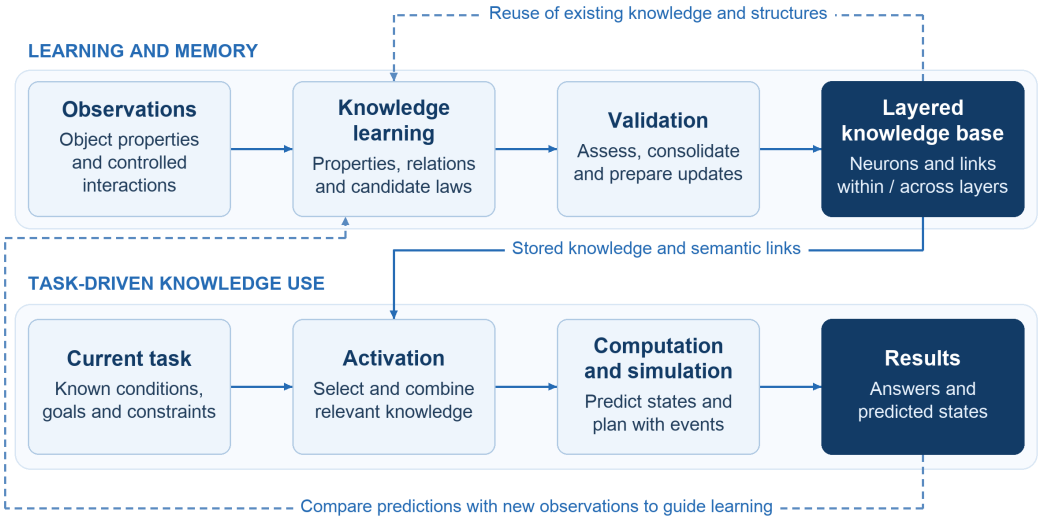

Figure 1: Overall KNWM architecture. Learning creates and updates knowledge, layered memory stores reusable knowledge and its connections, and problem solving activates relevant knowledge as needed. Comparing predicted or simulated outcomes with subsequent observations informs further learning.

3.2 Layered organization and knowledge connections

The knowledge base groups records by responsibility and connects them through semantic relations within and across layers. Layer membership specifies what knowledge represents; connections specify what it can be used with. The layers are therefore not isolated classification folders.

Scientific knowledge is organized into five principal layers. The property layer records objects and their describable or measurable properties. The causal-relation layer records influences between properties. The mathematical-pattern layer records forms such as proportional and inverse-square relations. The composition layer records how elements or effects combine. The physical-law layer stores complete laws connecting the relevant properties, relations, and mathematical structures. Additional records for object names, derived concepts, coordinates, and analytical tools support object identification, spatial representation, and scientific computation.

Methodological knowledge, algorithms and models, planning and metareasoning, and events and processes support knowledge use. Event records include elementary events and composition templates. Planning records describe goal decomposition, candidate organization, and replanning. Successful and unsuccessful paths guide later problem solving but do not directly modify physical laws.

For example, an object's mass can reference a property record, its influences can reference causal records, and a formula can connect these quantities in the physical-law layer. An event can then reference that formula to estimate resources under its current conditions. The same knowledge supports multiple formulas and tasks without a separate copy for each use. Appendix A details these connections.

3.3 Knowledge accumulation and updating

New properties, relations, and laws receive records with provenance and evaluation information. Existing knowledge is supplemented or adjusted according to its layer's rules, expanding both the stored content and its connections.

The physical-law layer keeps definitional and causal expressions separately. Within an entry, only validated, lower-loss candidates may replace expressions of the same kind; candidates that fail validation are not accepted. Records meeting formula-matching conditions can be merged with their

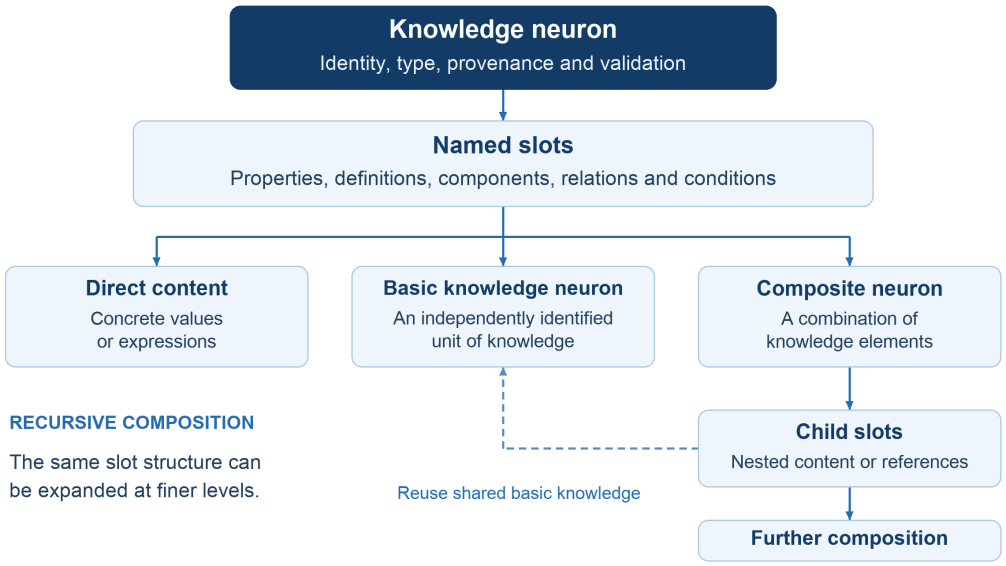

Figure 2: Recursive slot structure of knowledge neurons. Slots hold content or knowledge references, composite knowledge can be expanded further, and shared basic knowledge can be reused by different composite structures.

associated properties, relations, and provenance. Updating is thus knowledge-specific rather than unconditional overwriting.

Updated knowledge remains available through direct invocation, algebraic rearrangement, and compositional derivation. Problem-solving experience remains distinct from domain knowledge: success on one task can raise the priority of a solution path, but cannot by itself establish the general validity of its underlying laws. KNWM thereby connects reusable world knowledge, task-specific reasoning, and problem-solving experience while retaining their different roles and update criteria.

4 KNOWLEDGE-NEURON STRUCTURE AND OPERATING MECHANISMS

A knowledge neuron is a unit of storage, reuse, and composition (Figure 2). Learning creates its content and connections; reasoning activates it and performs computations or state simulations.

4.1 RECURSIVE SLOTS AND KNOWLEDGE TYPES

Knowledge neurons use recursively expandable slots. Each neuron has an explicit identity, type, and knowledge content, together with provenance and validation information. Slots organize properties, definitions, constituents, relations, and conditions. They can hold values or expressions or reference other knowledge neurons. Referenced composite knowledge can be expanded into further subslots, forming a recursive organization from parent structures to children and successively finer levels.

Type hierarchies and compositional hierarchies must be distinguished. Parent types, subtypes, and finer subtypes describe classification, whereas nested slots and knowledge references describe the constituents of a knowledge structure. Together they support representation from general concepts to specific knowledge and from basic elements to complex structures, without requiring all knowledge to fit fixed-length flat records.

By compositional granularity, knowledge is either basic or composite. Basic knowledge is a semantic unit that can be used independently in the current representation, such as mass, distance, or time. Composite knowledge combines these units and their relations into structures such as quantitative expressions, scientific laws, or event processes. This paper uses knowledge token to denote basic knowledge units and selected composite units that participate in knowledge connections and signal

transmission. Their boundaries are determined by semantic content and composition, rather than by text tokenization.

Scientific neurons provide executable relations and applicability conditions; event neurons describe preconditions, effects, costs, and subprocesses; problem neurons organize facts, goals, constraints, and subproblems. All use recursive organization, but their functional content differs (Table A.1). Basic/composite granularity is separate from these functional categories. Temporary task states remain outside reusable knowledge; reusable problem patterns and experience can enter long-term memory.

4.2 Learning scientific knowledge from observation and intervention

Scientific knowledge learning involves variable identification, relation induction, and validation. The system first selects properties associated with changes in the target. In learning routes that can control a physics simulator, it changes selected properties and compares outcomes while holding other conditions fixed. This helps identify which variables affect the target and their input or output roles in the relation under study. Candidate variables can therefore be supported by active intervention as well as by statistical covariation.

KNWM then constructs candidate formulas using three complementary mechanisms.

The first is symbolic expression search. The system constructs expressions from candidate variables and permitted mathematical operations, evaluating their fit to the data. The implementation includes both PySR symbolic regression and bounded expression-search routes. PySR serves as a formula-search tool; KNWM is responsible for organizing, storing, and subsequently invoking the resulting knowledge (Cranmer, 2023).

The second is analogical completion using existing knowledge structures. When a new relation resembles a learned one, the system can reuse the earlier formula structure, map its variable roles to the current properties, and refit its constants. An inverse-square structure, for example, can be a candidate form for another distance-dependent interaction. Analogy proposes a structural hypothesis for testing; the current observations still determine whether it is retained.

The third is incremental composition of formula fragments. The system first searches for simple relations that explain part of the variation, then analyzes the unexplained component and combines relevant fragments into a candidate expression. Fragments can also seed subsequent symbolic search, reducing repeated reconstruction from primitive operations. These mechanisms can be combined and need not all be invoked in a fixed order for every learning episode.

Once a candidate formula has been formed, KNWM holds its structure fixed and treats its constant coefficients as parameters to be estimated. It adjusts simulator inputs to obtain new input-output observations, substitutes these observations into the formula, and re-estimates and calibrates the coefficients to improve parameter accuracy. Depending on the learning route, it then checks fitting error, parameter stability, or predictive performance on independent data and organizes variable meanings and their connections. Intervention-supported influences and relations obtained solely by numerical fitting are recorded separately: a formula describes how variables covary, whereas intervention evidence supports a judgment about the direction of influence. Properties, relations, mathematical structures, and complete laws subsequently enter their corresponding knowledge layers, as described in Section 3, for use in later tasks.

4.3 Shared knowledge signals and selective activation

Knowledge neurons connect through shared knowledge elements. When a quantity produced by one neuron can serve as an input to another, the two can form a reasoning chain. An identical symbol refers here to the same declared semantic concept, not merely the same character sequence. Specific objects, states, and variable roles are bound within each task, preventing same-named properties of different objects from being conflated.

Basic knowledge can participate in multiple composite structures, and outputs of composite knowledge can become inputs to further knowledge. One formula may compute the force on an object, for example, while another uses that force and the object's mass to compute acceleration. Their shared force quantity connects the formulas without requiring a new hard-coded computational chain for

each problem. Composite expressions in existing definitions can likewise be identified and reused, establishing connections at different levels.

The implementation records input, output, and shared relations in a slot-connection graph that indexes knowledge connections without replacing knowledge neurons themselves. During problem solving, the system works backward from the target to identify knowledge that can produce it and the inputs that knowledge requires. It then propagates activation forward from known quantities along these dependencies. When the necessary inputs are available and activation conditions are satisfied, a formula becomes a candidate for execution; its output can support further knowledge activation.

Activation and computation have distinct roles. Activation identifies relevant knowledge and connectable dependencies; computation substitutes task-specific values into the selected relations to obtain intermediate and target results. Relations that support algebraic rearrangement can also be solved for other unknowns. The resulting task-specific knowledge subnetwork makes reuse an executable derivation, rather than merely the retrieval of relevant text.

4.4 Law-based prediction and causal reasoning

When scientific knowledge neurons provide the quantitative relations needed for state changes, KNWM can compute a subsequent state from the current state and reuse the result in the next stage. Position, velocity, time, and other physical properties participate through explicit variables and dependencies, allowing multi-step predictions to continue using the same laws. Stored formulas can be applied to new input values; predictive accuracy depends on their applicability, parameter accuracy, and the numerical methods used.

Causal reasoning can follow direct relations or compose relations to analyze indirect effects. If the knowledge base represents mechanisms in which A affects B and B affects C, the system can, when the relevant conditions hold, compute the effect of changing A on C through the intermediate quantity B. This cross-relation reasoning retains the intervening mechanisms rather than only storing an association between A and C. Algebraic solution direction and physical causal direction must also be distinguished: solving a formula for another variable does not reverse its causal direction.

Counterfactual reasoning changes specified conditions in a temporary state, retains the other assumptions, and recomputes outcomes using the same knowledge. For example, with source mass and distance fixed, doubling a test object's mass doubles its gravitational force but not the acceleration derived from that force. Obtaining this result requires combining gravitational and motion relations. The system changes the conditions being analyzed rather than rewriting a stored law to obtain a different answer.

4.5 Event simulation, nested planning, and cost-based search

Event planning starts from the current state, goals, and constraints organized by problem knowledge neurons. Event knowledge neurons specify when a step can occur and which states it changes. Planning does not begin with a predetermined route from the initial state to the goal. Instead, it generates candidate successor states, compares event sequences, and progressively constructs a path satisfying the goal.

At each candidate state, the system checks event preconditions and resource requirements, simulates applicable events, and obtains successor states and associated consumption. Event costs can be fixed or determined by computations linked to scientific knowledge. Scientific knowledge thus affects event feasibility and plan ordering, in addition to answering standalone numerical questions.

The current science-event interface computes and binds the numerical quantities required by events under the task conditions. Results for inputs that remain fixed throughout planning can be computed before search and reused. Inputs that change with simulated state instead require state-specific evaluation and cannot reuse a fixed binding indiscriminately. This distinction allows an event to accommodate different task conditions while making the validity of its numerical results explicit.

Composite events expand through subevents or subproblems. A high-level step can contain several finer-grained steps, each of which still requires condition checks, state simulation, and cumulative resource accounting. Shared subevent knowledge can be reused across processes, but each occur-

rence retains its own process position and execution state, preventing reuse of the same knowledge from being mistaken for a single execution. Subprocess states and consumption are then incorporated into the comparison of outer plans.

Path search uses cumulative cost rather than selecting the cheapest current event in isolation. A* organizes candidate-state expansion using

$$f(n) = g(n) + h(n),$$

where $g(n)$ is the cumulative cost from the start to candidate state n, and $h(n)$ estimates the remaining cost to the goal. Scientific computation supplies quantities required for state changes or cost evaluation; search uses these quantities to compare paths. With nonnegative costs, an admissible heuristic, and the appropriate search conditions, A* can return a minimum-cost path within the modeled search space (Hart et al., 1968).

The current nested-process search sets $h(n) = 0$, giving uniform-cost search. Budget-limited results are not globally optimal without sufficient comparison. Planning combines problem goals, event changes, and scientific calculations into a path generated through simulation and cumulative-cost comparison. Appendix C explains the corresponding execution, binding, and physics-solving interfaces.

5 Evaluation of world-model capabilities

Controlled experiments test learning, reasoning, prediction, planning, and adaptation. Each retains its implementation snapshot, input conditions, and scoring unit; task counts and trajectory checkpoints are not pooled. Appendices D–F specify protocols, scoring, and scope.

5.1 Learning and retaining reusable knowledge from observations

Across 40 tasks in six basic relation families, KNWM learns every relation, validates it over an independent interval, and invokes it after storage. A second experiment learns and stores 20 relations and rejects five no-relation controls. These results demonstrate acquisition and reuse rather than retrieval of pre-entered answers. Discovery here concerns relations unknown to the learner, not laws unknown to humanity.

On 36 further tasks, independent validation admits 20 formulas; 19 pass a more distant final-test range at a 5% normalized root mean square error threshold. Training-based admission accepts 30, with ten final-test failures. Validation filters nine of those failures but also rejects one final-test-passing candidate. Targeted new observations, revised structures, or coefficient calibration can address different failure causes; their iterative correction rate is not measured here (Appendix D.1).

5.2 Mechanistic understanding, causal reasoning, and transfer

KNWM passes controlled intervention and counterfactual tests with 100 scenes or instances each. These respectively demonstrate identification of relevant influences and prediction under changed conditions, while retaining the persistent knowledge base.

It also passes 100 object-transfer instances with changed names and irrelevant colors, and 100 structural-analogy tasks selecting and validating predefined candidates. These results support reuse based on physical properties and bounded structural transfer, rather than dependence on object names.

KNWM forms successful derivation chains on all 300 composed-relation tasks, and the conventional symbolic comparator also solves all tasks. These results demonstrate KNWM's knowledge composition and multi-step derivation capabilities on this test set (Appendix D.2).

5.3 Sustained prediction and spatiotemporal state simulation

KNWM and a fully informed symbolic comparator pass all 6,000 checkpoints across 200 independent trajectories using supplied constant-velocity and constant-acceleration laws. This tests repeated application of stored knowledge to evolving states.

With hidden 5% gravitational-parameter mismatch, median normalized root mean square error rises from 0.48% to 2.26% over 10–200 steps in five-object scenes (20 paired scenes per horizon). Explicit laws support low-error multi-step prediction here, although model error still accumulates.

Separate calibrated-triangulation and synthetic image-to-state tests assess spatial representation; the latter passes 96/100 scenes, retaining detection failures in the denominator. Appendix D.2 reports their distinct error measures.

5.4 How scientific knowledge changes event planning

In 24 scenarios, KNWM learns relations, stores them, reloads them in a new process, and calculates event requirements. Validated reuse yields 24 feasible and 17 truly optimal plans, compared with 24 feasible but only six optimal plans without reuse. Learned knowledge changes 11 choices to optimal ones, demonstrating a connection from learning to later decisions.

Training-based admission gives 18 optimal but three infeasible plans. An equivalent target index makes the same choices as validated reuse with the same knowledge. The benefit demonstrated here comes from knowledge acquisition and reuse, not a unique retrieval algorithm (Appendix D.3).

A nested-process test combines 16 datasets and 12 conditions into 192 tasks with two candidate processes each. Of these, 107 have a feasible process; 85 have none under resource, evidence, or risk constraints. With bounded additional learning, KNWM completes 81 feasible tasks, rejects 26, and correctly rejects all 85 infeasible tasks. Training-only admission completes 87 feasible tasks but accepts 13 infeasible ones.

New-seed tasks yield 83/108 feasible completions and 84/84 correct infeasible rejections. Additional learning uses more observations and computation, so this is not an equal-budget comparison. These results quantify reliability and coverage; search cannot correct an inaccurate underlying relation. Appendix D.4 gives budgets and all outcomes.

5.5 Knowledge updating and adaptation under uncertainty

Validation protects accumulated knowledge against erroneous updates: it retains 98/100 correct records at 5% noise, versus 19 under direct overwriting; at 10% noise, the counts are 76 versus 15. Observation quality affects this protection.

Under simultaneous observation noise, missing observations, and stochastic motion disturbances, closed-loop control completes 78/100 tasks, versus 52 without state estimation and two without replanning. Feedback therefore supports task completion under these uncertainties. All 100 system trajectories meet the tested safety conditions; 22 nevertheless miss terminal position or velocity requirements.

At 5% noise, 39/40 structures are recovered but only 15/40 pass strict held-out prediction scoring; test targets are also noisy. Planning success falls from 100% with three obstacles to 25% with six. Appendix D.5 reports intermediate difficulties and additional task groups.

5.6 Knowledge-access efficiency and representation overhead

The proposed knowledge-neuron storage structure supports efficient indexed lookup and rich knowledge organization. On 100,000 records, warmed-up single-record lookup is about 7,753.6 times faster than sequential scanning; separately, full neuron serialization is 4.36 times the size of simplified records. These measure lookup and representation overhead, not whole-system speed or memory.

With 10,000 rules and matched dependencies and executors, median retrieval-plus-execution takes 0.6542 ms for KNWM, 0.0183 ms for an inverted index, and 5.1395 ms for scanning. All solve 144 queries and reject 36 with missing dependencies. KNWM avoids scanning overhead, while typed activation and diagnostics cost more than equivalent indexing. Loading and index construction are excluded (Appendix D.6).

5.7 Comparison with language models and overall findings

On 900 historical structured tasks, KNWM and a task-routed symbolic comparator score 900/900; GPT-5.6 Sol (Codex CLI) and Qwen2.5-VL-7B (Ollama) score 895/900 and 417/900, respectively (Appendix E.3). These results support reliable explicit computation under the tested conditions, without establishing superiority over language models with matched tools. Local reproduction shows that KNWM can execute these structured tasks on a computer with 16 GB of RAM without calling a large language model during problem solving.

On 93 validity-reviewed development problems, KNWM combines scientific and event knowledge from manually assisted structured inputs and physical conditions. Project-internal review confirms 82/93 correct (88.17%); four further outputs hold under specific interpretations or additional conditions and are reported separately, not included in this accuracy. These results demonstrate KNWM's knowledge composition and multi-step solving on this set. This is not end-to-end language accuracy or an unseen-task evaluation (Appendix E).

Together, these experiments connect observable capabilities to the design: learned relations remain usable after storage, knowledge reuse changes plans, and validation improves reliability. Difficulty tests identify where prediction and task completion require further development.

6 Discussion and conclusion

6.1 Knowledge neurons as basic units of world modeling

KNWM turns learned laws into reusable computational resources through recursive structures, layered memory, and shared knowledge signals. Scientific neurons calculate consequences; event neurons organize simulatable steps; problem neurons connect these operations to goals. This makes learning consequential for subsequent prediction and planning.

Reliability depends on both a valid reasoning path and knowledge applicable to its conditions. Equal-information implementations reproduce some results, so the contributions of knowledge organization and validation must be distinguished from those of solvers, indexes, and search.

6.2 Scope and further development

Controlled results demonstrate working learning, prediction, and planning mechanisms. Their present scope is structured inputs, supplied actions, and bounded search. Broader independent tasks, repeated trials, and matched-information and matched-tool comparisons should test transfer beyond these conditions.

Further work should build on the existing learning and validation mechanisms. Independent observations and interventions spanning wider parameter ranges can test applicability and reduce overfitting risk. Better perceptual and language grounding, cross-domain structural reuse, and active experiment design can extend knowledge acquisition. Updates should preserve provenance and older laws that remain valid under different conditions.

When simulated actions change formula inputs, scientific quantities and subsequent resource feasibility must be reevaluated. Efficiency evaluation should cover knowledge selection, computation, state expansion, and maintenance against equivalent indexes and tools, separating benefits of reuse from representation overhead.

6.3 Conclusion

KNWM connects knowledge learning and accumulation to explicit computation and internal simulation. Recursive scientific, event, and problem neurons organize reusable reasoning with inspectable dependencies. The controlled results support this design's ability to carry knowledge from learning into later prediction and decisions, providing a basis for further work on transfer, adaptation, and knowledge discovery.

AI USE STATEMENT

The author proposed the KNWM research concept and core knowledge-neuron design and directed the system's development. Generative AI tools, including OpenAI Codex, assisted with four aspects of the work: (1) academic English translation and language editing; (2) literature searches and identification of related work; (3) author-directed research development and execution; and (4) manuscript drafting and restructuring, figure preparation, and LaTeX formatting.

The author led problem assessment and solution development. Codex helped organise issues at the author's request, implement solutions proposed or approved by the author, and refine technical details. Assistance included code implementation and debugging, refinement and execution of experimental designs, structured-input preparation, knowledge curation, and result analysis and interpretation. Codex also assisted with scripts for generating and validating knowledge-base practice examples and simulated experimental data; the author was responsible for reviewing and revising the relevant materials.

Reported numerical results derive from executable experiments and retained evaluation records. Verification involved automated tests and project-internal review, with AI assistance, as described in the experimental sections and appendices. The author takes responsibility for the final methods, results, interpretations, and manuscript, including all AI-assisted content.

REFERENCES

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences, 113(15), 3932–3937. https://doi.org/10.1073/pnas.1517384113

Cranmer, M. (2023). Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl. arXiv:2305.01582. https://doi.org/10.48550/arXiv.2305.01582

Dai, D., Dong, L., Hao, Y., Sui, Z., Chang, B., and Wei, F. (2022). Knowledge Neurons in Pretrained Transformers. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 8493–8502. https://doi.org/10.18653/v1/2022.acl-long.581

Fikes, R. E., and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3–4), 189–208. https://doi.org/10.1016/0004-3702(71)90010-5

Gerstner, W., Kistler, W. M., Naud, R., and Paninski, L. (2014). Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition. Cambridge University Press. https://neuronaldynamics.epfl.ch/

Ha, D., and Schmidhuber, J. (2018). Recurrent World Models Facilitate Policy Evolution. Advances in Neural Information Processing Systems, 31. https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. (2025). Mastering diverse control tasks through world models. Nature, 640, 647–653. https://doi.org/10.1038/s41586-025-08744-2

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33, 9459–9474. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

maximus-21. PhysicsQA [Dataset]. Hugging Face. Fixed revision used in this study: 0b8218d957947dc0f5d776ef018cb41dc10abf15. https://huggingface.co/datasets/maximus-21/PhysicsQA/tree/0b8218d957947dc0f5d776ef018cb41dc10abf15

Minsky, M. (1974). A Framework for Representing Knowledge. MIT Artificial Intelligence Laboratory, AI Memo 306. https://dspace.mit.edu/handle/1721.1/6089

Nau, D., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. (2003). SHOP2: An HTN Planning System. Journal of Artificial Intelligence Research, 20, 379–404. https://doi.org/10.1613/jair.1141

Pearl, J. (2009). Causality: Models, Reasoning, and Inference. Second edition. Cambridge University Press. https://doi.org/10.1017/CBO9780511803161

Silver, T., Athalye, A., Tenenbaum, J. B., Lozano-Pérez, T., and Kaelbling, L. P. (2023). Learning Neuro-Symbolic Skills for Bilevel Planning. Proceedings of the 6th Conference on Robot Learning, Proceedings of Machine Learning Research, 205, 701–714. https://proceedings.mlr.press/v205/silver23a.html

Xu, T., Yu, X., Perlik, A. J., Tobin, W. F., Zweig, J. A., Tennant, K., Jones, T., and Zuo, Y. (2009). Rapid formation and selective stabilization of synapses for enduring motor memories. Nature, 462, 915–919. https://doi.org/10.1038/nature08389

A KNOWLEDGE NEURONS AND LAYERED MEMORY

This appendix expands on Sections 3 and 4.1, describing knowledge-neuron composition, type hierarchies, and cross-layer connections. Figures A.1 and A.2 are conceptual diagrams: their boxes describe the semantic responsibilities of records rather than requiring every implementation to use the same fixed set of fields.

A.1 GENERAL STRUCTURE AND RECURSIVE SLOTS

A knowledge neuron consists of identifiable knowledge content and its conditions of use. Identity distinguishes knowledge records; type specifies what the knowledge represents; slots hold properties, expressions, conditions, or references to other knowledge; and provenance and validation information document its basis. References allow multiple composite structures to share basic knowledge instead of duplicating a common concept into unrelated copies.

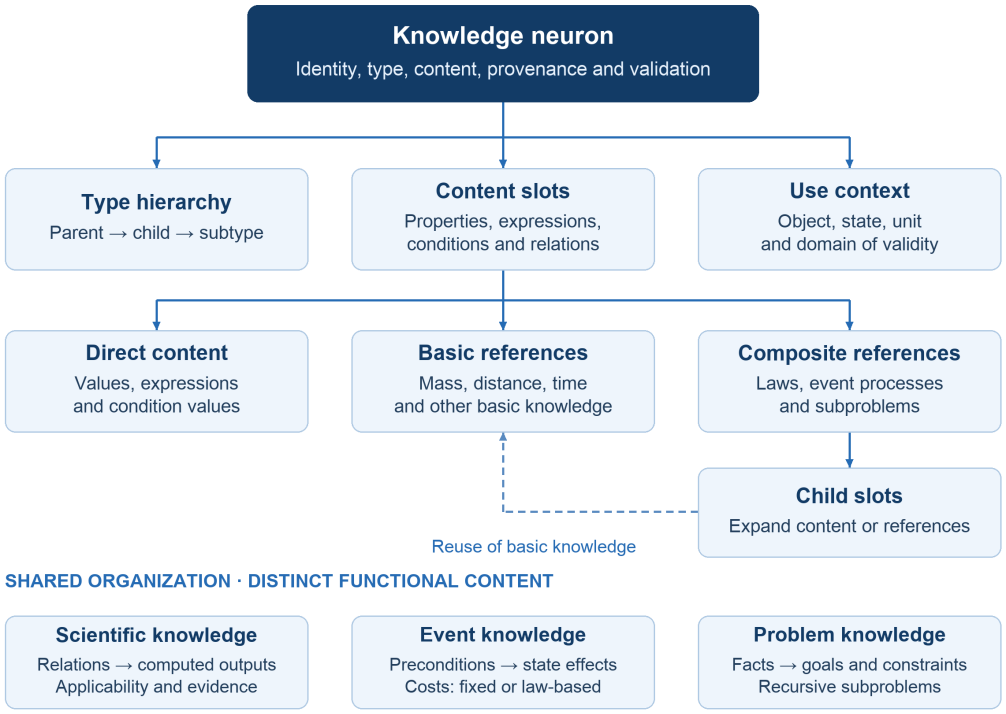

Figure A.1: Recursive knowledge-neuron structure. Type relations determine applicability, whereas composition relations expand knowledge content; these relations are recorded separately. Scientific, event, and problem knowledge share an organizational principle but not every functional field.

Parent types, subtypes, and finer subtypes describe classification scope, such as physical objects, wave systems, and electromagnetic waves. Composite knowledge, subslots, and basic knowledge describe composition, such as how a law references mass, distance, and mathematical relations. Knowledge use requires both locating the applicable type and checking the current object, state, units, and premises. A shared property name alone does not justify treating a general mechanical wave as a photon.

Basic and composite knowledge distinguish compositional granularity; scientific, event, and problem knowledge distinguish function. A knowledge token is basic knowledge or selected composite knowledge that identifies and conveys explicit semantics, rather than a result of text tokenization. A signal can carry a bound quantity such as the force on a particular object in a particular state, connecting the relation that produces it to a relation that requires it.

Table A.1: Content and execution contracts of the three knowledge types.

Knowledge type	Reusable content	Task binding and output
Scientific	Variable roles, expressions, applicability conditions, units, and validation range	Bind objects and states; substitute known quantities; return target quantities or conditional relations
Event	Preconditions, state effects, resource requirements, fixed or computable costs, and subevent references	Check conditions; simulate effects; update state, resources, and cumulative cost
Problem	Goals, constraints, subgoals, and reusable solution patterns	Bind task facts and goals; organize dependencies; check actual goal completion

A.2 Layered organization and cross-layer references

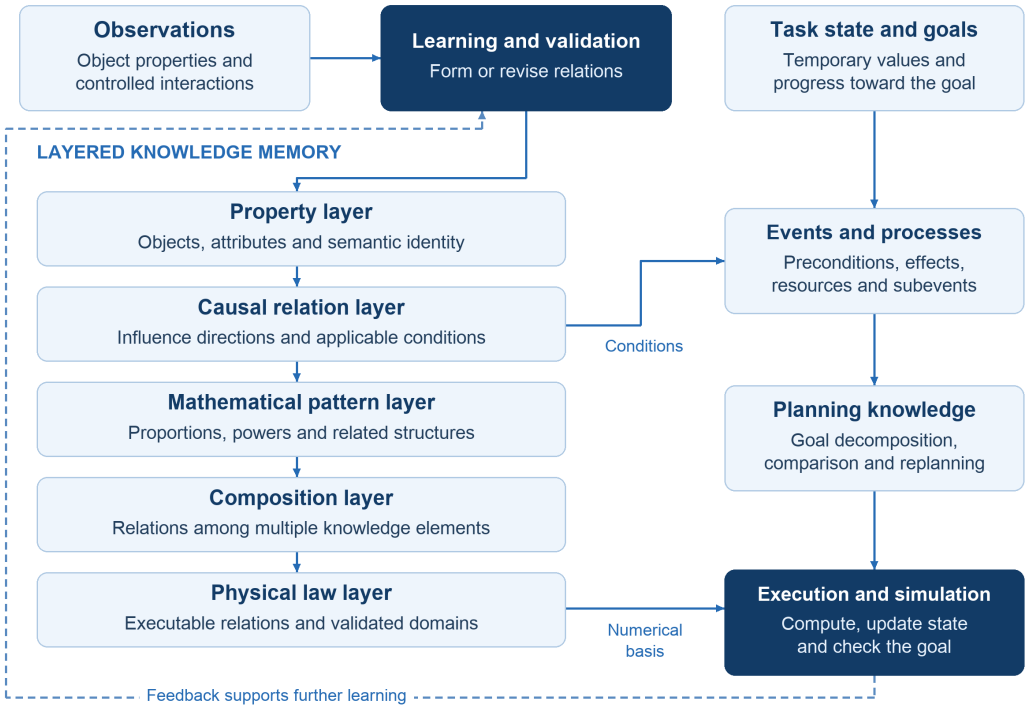

Figure A.2: Layered knowledge base and its connections. Arrows illustrate dependencies, calls, or feedback, not a mandatory sequential execution of layers. References between knowledge units do not imply reversible physical causality. Learning creates or updates records in the appropriate layers, while temporary values remain in task state.

The property layer stores objects and their describable or measurable properties; the causal-relation layer stores directions and conditions of influence; the mathematical-pattern layer stores structures such as proportionality and powers; the composition layer describes connections among elements; and the physical-law layer stores complete computational relations. Event and process knowledge and planning and metareasoning knowledge reference these resources to support goal decomposition and process simulation. Layer membership and cross-layer connections are complementary: layered storage does not prohibit connections between layers.

A.3 Adding and updating knowledge

New knowledge enters the appropriate layer and establishes references; updates to existing records follow that layer's rules. Physical-law records can maintain separate definitional and causal expressions, selecting expressions by fitting loss or merging records when the corresponding conditions are met. Lower fitting loss indicates closer agreement with the data used, not an automatic extension of a law's applicability. The experimental comparison between validated integration and direct overwriting tests update strategies; it does not imply that every storage interface uses identical validation rules.

Reusable knowledge and task state are maintained separately. A mass, velocity, or event-completion marker within a task belongs to a specific object, state, or process occurrence. Only knowledge processed through the relevant learning and recording procedures enters long-term memory. This separation allows general knowledge to be updated without temporary results contaminating later tasks.

B Scientific knowledge learning and coefficient calibration

This appendix supports Section 4.2. Figure B.1 shows the learning process from observations of object properties to knowledge use. Individual experiments instantiate the relevant branches and do not all execute the complete simulator-intervention process.

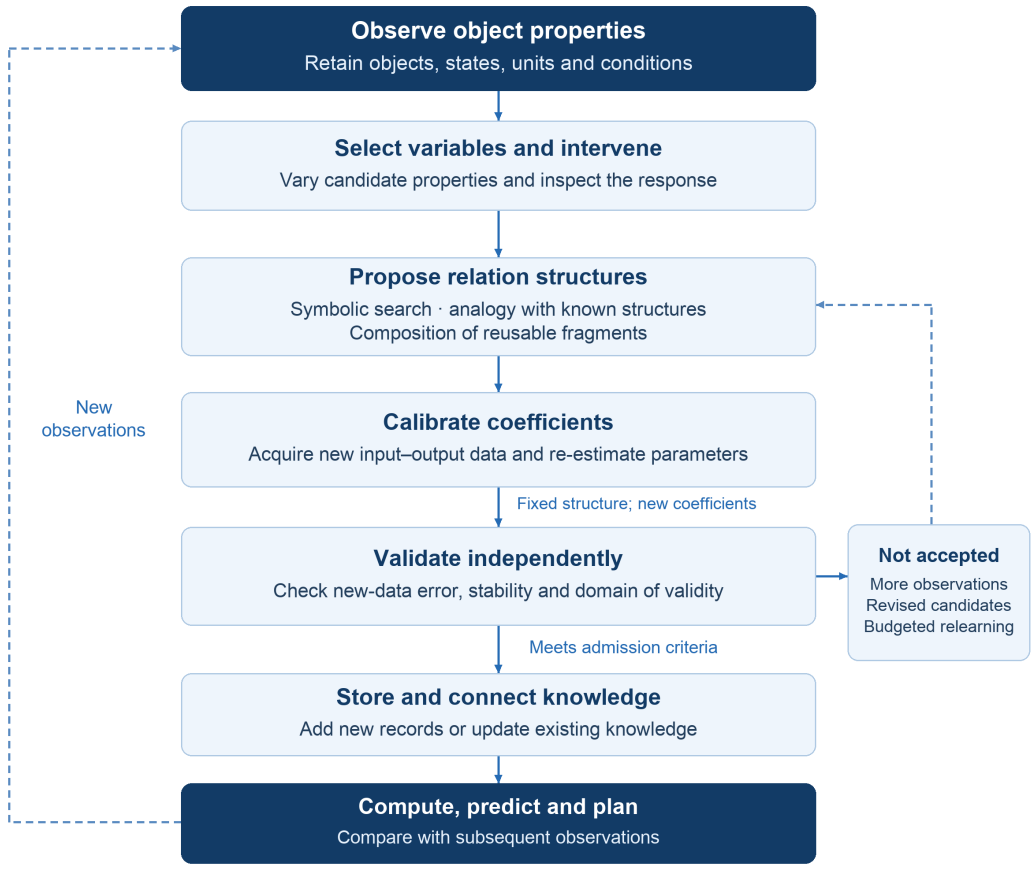

Figure B.1: Scientific knowledge learning and validation. Relation structure and coefficients are treated separately. Data used to adjust either cannot simultaneously serve as independent validation evidence. Additional learning after failure is bounded by observation and search budgets, and the final test supplies no feedback to the learner.

B.1 VARIABLE IDENTIFICATION AND CANDIDATE STRUCTURES

Learning starts from changes in object properties over time and through interaction. In controllable simulations, the system changes candidate properties while holding other conditions as constant as possible, testing their influence through changes in the response. Records distinguish observed associations, intervention-supported directions of influence, and algebraic computation direction. Solving for a variable does not establish reverse causality (Pearl, 2009).

Candidate relations can be proposed through symbolic expression search, analogical completion of known structures, or formula-fragment composition. PySR is an optional symbolic-search tool; open-law beam search uses its own declared operator set. Existing knowledge supplies reusable structures and variable roles, allowing basic knowledge to support more complex learning, but analogically proposed structures still require testing against new observations (Cranmer, 2023).

B.2 COEFFICIENT CALIBRATION WITH A FIXED STRUCTURE

Once a relation structure has been selected, calibration treats its constant coefficients as parameters to be estimated. The system obtains new input-output observations under the relevant simulation conditions and refits the coefficients without changing the structure. For example, suppose the selected relation is

$$y = k\varphi(x) \tag{B.1}$$

where φ denotes the selected structure and k is the coefficient to calibrate. Evaluating φ for each new input and using the observed outputs estimates k. With no intercept, equally weighted squared error, and a nonzero denominator, the estimate is

$$k^* = \frac{\sum_i \varphi(x_i) y_i}{\sum_i \varphi(x_i)^2} \tag{B.2}$$

Models with an intercept or multiple coefficients require joint parameter estimation rather than forcing a constant from a single observation. Inputs should vary sufficiently to avoid parameters becoming indistinguishable when variables always change together. Calibration improves parameter estimates within a given structure. Incorrect variable selection, missing operations, or violations of applicability conditions require reconsideration of the relation structure.

B.3 INDEPENDENT VALIDATION AND BOUNDED ADDITIONAL LEARNING

Fitting, validation, and final-test data have distinct roles. Fitting selects structures or estimates coefficients, validation determines admission, and the final test measures performance after admission. Data already used to select a structure or adjust coefficients cannot still be described as independent validation data. Agreement across widely separated validation samples supports generalization within the covered range, but cannot establish the absence of overfitting under all possible conditions.

The 36-task experiment in Section 5.1 evaluates candidate admission under a fixed budget and does not include the full process of actively controlling a simulator. The additional-learning experiment in Section 5.4 obtains independent new training and validation observations after rejection, then searches and validates within the original expression space. It permits structural reselection rather than coefficient calibration alone; the two experiments are reported separately.

Additional learning targets the cause of failure. If variables cannot be distinguished, new observations should break their covariation; if coefficients are unstable, observations should increase the effective input range and sample size; if the candidate structure is unsuitable, search must be repeated or transferred to a learning route that supports the required structure. Only candidates that subsequently satisfy admission criteria can support further computation.

C KNOWLEDGE ACTIVATION AND EVENT PLANNING

C.1 FROM SHARED KNOWLEDGE SIGNALS TO EXECUTABLE DEPENDENCIES

Selective activation in Section 4.3 combines backward goal dependencies with forward propagation from known quantities. The task first distinguishes objects, states, quantity roles, units, and applicability conditions. The system then retrieves knowledge capable of producing the target and creates subgoals for missing inputs. A relation executes only after all necessary inputs are available, and its output becomes an input to downstream knowledge.

For example, a momentum relation can bind separately to the masses and velocities of two objects before passing their momenta to a total-momentum relation. These are distinct instances of one reusable relation, not unassigned values formed by merging properties across objects. Types and premises constrain admissible relations; circular references cannot create missing known quantities.

C.2 EVENT-STATE SEARCH AND NESTED PROCESSES

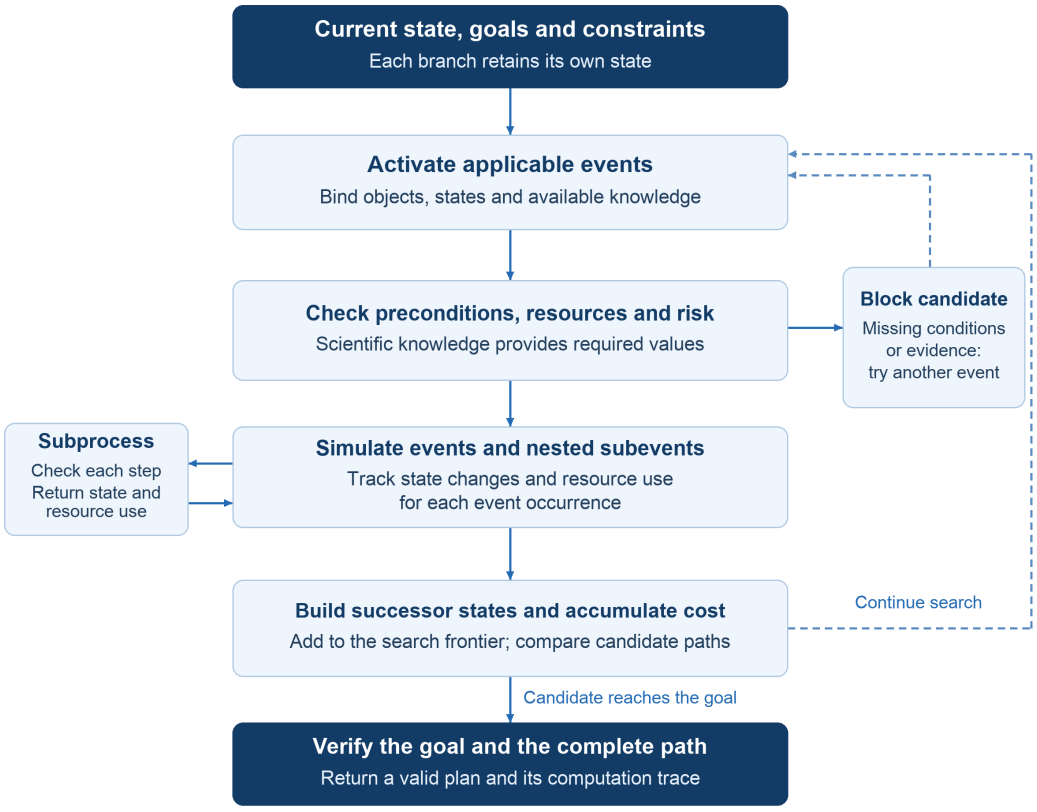

Figure C.1: Nested event planning with scientific computation. Each branch checks premises and resources in its own state; completed subprocesses return state and consumption to the outer process. Failure to find a plan within a search budget is recorded as unresolved with a reason, rather than as proof that no path can reach the goal.

Event state effects and costs are represented separately. A step may consume fuel, increase elapsed time, or establish a fact, whereas cost ranks candidate paths. A resource requirement alone is an applicability check; only an explicit consumption effect changes available resources in the successor state. A parent event can declare global conditions, while leaf-event consumption is accumulated for each occurrence without being charged again through parent-level aggregation.

The same subevent can be referenced at multiple positions in a nested structure, with separate execution position and progress for each occurrence. A parent completes only after its required child

steps have completed. Shared knowledge identity does not imply a shared execution result. Ordering, hierarchy, revision, and progress constraints jointly prevent unintended duplicate use, omitted steps, and the use of an old process's completion record to satisfy a new process.

Path comparison combines cumulative cost from the start with an estimate of remaining cost. The nested-event search in Section 4.5 uses a zero heuristic and is therefore uniform-cost search. Its optimality claim applies only to its modeled costs, conditions, and complete search space (Hart et al., 1968). Numerical action control and physics problem solving use different state and operation interfaces, so a search guarantee for one interface cannot automatically be extended to another.

C.3 Binding scientific quantities to event conditions

The science-event interface binds the knowledge record, target quantity, formula inputs, and corresponding task state. Numerical results can be computed once and reused when inputs remain fixed during planning. If events change those inputs, valid values for the corresponding successor state are required; old-state results cannot continue to be used. Section 5.4 reports numerical prediction and nested-process experiments connected through experimental adapters, rather than a complete autonomous execution system for arbitrary dynamic environments.

Evidence bindings identify the knowledge, revision, and conditions associated with a numerical value or judgment, helping locate invalidated dependencies after updates. A matching record digest establishes reference consistency only. It does not by itself prove that an external experiment occurred, a physical law holds universally, or a real-world action succeeded.

C.4 Event composition in physics problem solving

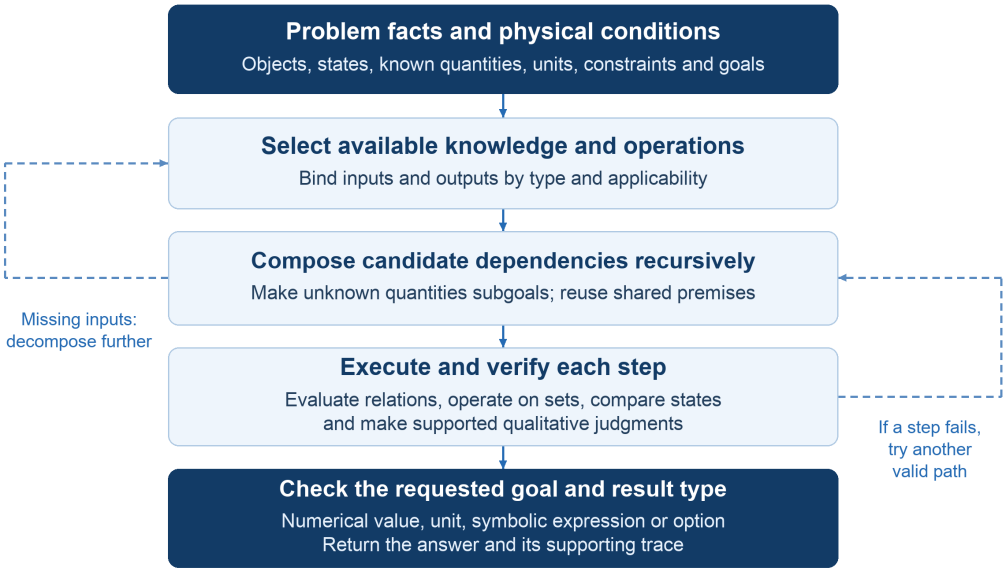

Figure C.2: Knowledge-composition route for the physics problems in Section 5.7. Inputs provide facts, conditions, and goals; knowledge records provide individual relations and operation contracts; and the composer constructs and executes dependency paths. Result checks distinguish a target answer from insufficient conditions, exhausted budget, or failed validation. Calling a solver once is not treated as having constructed a complete solution.

This route represents known or established quantities as state and considers relation evaluation, set operations, state comparison, and justified judgments as candidate steps. Event steps declare their required inputs and produced outputs. When a goal depends on an unavailable quantity, the planner continues decomposition and, after failure, tries other admissible paths within the overall budget. Each step's inputs, outputs, and goal dependencies are retained in its trace, making its role in problem solving inspectable.

The composition route invokes KNWM's existing problem-planning mechanisms using bounded AND/OR search and an A*-style priority queue. Structural step costs in this physics experiment are not physical resources such as fuel or time, and the experiment does not claim globally shortest proofs for every problem. Individual relations and general operations curated with human and AI assistance form the available resources. The solver does not retrieve prewritten complete solutions or reference answers by problem identifier. The evaluation tests the composition and execution of existing knowledge, not autonomous discovery during knowledge curation.

D EXPERIMENTAL SETTINGS AND SCORING

This appendix supports Section 5. Each experiment retains its own statistical units, input provenance, and implementation snapshot. Repeated timings of a query, multiple checkpoints within a trajectory, and task conditions sharing a dataset are not counted as additional independent samples.

D.1 LAW LEARNING AND INDEPENDENT VALIDATION

The 40 primitive-discovery tasks cover six relation families and test relation recovery, independent-interval validation, and invocation after storage. A separate group of 20 relations and five no-relation controls tests learning, input identification, and rejection. The generator specifies target structures without supplying answers directly to the learner. These experiments measure the recovery and reuse of relations unknown to the learner, not the discovery of natural laws previously unknown to humanity.

The 36 numerical-discovery tasks use three seeds, each generating 12 relation types. Training, validation, and test input ranges are [0.7, 2.0], [2.3, 3.5], and [3.8, 5.2], respectively. Six anonymous input columns contain relevant variables and distractors. In half the tasks, a distractor covaries with a true input during training but varies independently during validation and testing, assessing variable selection rather than fit alone.

Search depth is three, beam width 64, partner width 12, and the complexity penalty 10^{-7}. Permitted operators are addition, subtraction, multiplication, division, square, cube, square root of absolute value, and reciprocal. Sinusoidal, exponential, and piecewise relations test cases beyond the current expression space. Admission requires normalized training error no greater than 3%, validation error no greater than 5%, and a coefficient-stability ratio no greater than 0.20. The stability ratio is the standard deviation of slope estimates from five fold-excluded fits divided by their mean absolute magnitude; it is not a probabilistic confidence interval.

Numerical scoring divides prediction root mean square error by the root mean square of target values, with a denominator floor of 10^{-12}. Final-test target values are withheld from candidate search. An admitted candidate whose final-test error exceeds 5% is counted as a numerical false acceptance.

Table D.1: Admission outcomes for the same 36 candidates.

Admission rule	Admitted	Admitted and test passed	Admitted but test failed
Retain independent validation	20	19	1
Same candidates; training and stability thresholds only	30	20	10

Both admission conditions reuse the same selected candidates. Removing validation does not trigger a new search for more favorable formulas. Of the ten candidates no longer admitted with validation, nine fail the final test; one passes the test threshold but fails validation. That candidate still substitutes a distractor for a true variable, so low test error does not establish recovery of the correct mechanism. The experiment evaluates candidate admission only and does not store these candidates in the persistent knowledge base.

Table D.2: Tasks and assessment units for Sections 5.2 and 5.3.

Experiment	Independent tasks or scenes	Assessment
Controlled causal intervention	100 two-object scenes	Change candidate properties; distinguish influences from irrelevant properties
Counterfactual reasoning	100 instances	Original, modified, and restored conditions; persistent knowledge base unchanged
Object transfer	100 instances	Change names and irrelevant colors; retain physical conditions
Structural analogy	100 tasks	Select and validate structures from given candidate relations
Knowledge composition	300 tasks across six generated topology families	Test whether supplied relations form valid derivation chains
State prediction	200 trajectories	6,000 checkpoints in total, not 6,000 independent trajectories
Model mismatch	20 paired scenes per horizon	Five-object systems with hidden 5% gravitational-parameter mismatch
Synthetic visual prediction	100 two-frame scenes	96 passes; four detection failures retained in the denominator

D.2 Statistical units for mechanistic reasoning and prediction

This paper operationalizes mechanistic understanding through three testable behaviors: identifying properties that affect an outcome, predicting consequences of changed conditions, and continuing to use the same laws when irrelevant surface features change.

Controlled causal-intervention tests comprise 100 two-object scenes, each with multiple property interventions. The system uses intervention outcomes to distinguish relevant influences from irrelevant properties. A separate counterfactual test comprises 100 instances, each checked under original, temporarily modified, and restored conditions. The system satisfies the scoring criteria throughout these tests while leaving the persistent knowledge base unchanged. The two experiments respectively support identifying influences through intervention and computing alternative-condition outcomes using known mechanisms.

Knowledge-transfer tests change object names and irrelevant colors while retaining the physical properties required by the task. Correct results across 100 instances show that these computations depend on bound knowledge properties rather than specific object names. Another 100 bounded structural-analogy tasks require selecting a relation structure from predefined candidates and validating it on held-out observations. All tasks pass, supporting transfer of known structures within a restricted candidate space rather than mere repetition of an existing formula.

A composition experiment further tests whether multiple relations can jointly solve a problem. KNWM completes all 300 tasks across six families of generated dependency topologies; the conventional symbolic comparator also passes all tasks. Knowledge neurons can therefore form effective derivation chains, while equal accuracy indicates that successful composition and the distinctive benefits of a representation should be evaluated separately.

Prediction experiments test whether stored laws can be applied repeatedly to evolving states rather than only in one-off calculations. Across 200 trajectories governed by supplied constant-velocity and constant-acceleration relations, KNWM passes all 6,000 state checkpoints; a symbolic-reasoning comparator with the complete knowledge base also returns correct results. There are 200 independent trajectories, with checkpoints within each trajectory assessing the consistency of sequential state propagation.

To examine imperfect models, increased-difficulty experiments introduce a hidden 5% gravitational-parameter mismatch in five-object scenes and progressively extend the prediction horizon. Each horizon uses 20 paired scenes. As the horizon increases from 10 to 200 steps, median normal-

ized root mean square error rises from 0.48% to 2.26%. Explicit laws support low-error multi-step prediction under these conditions, but parameter mismatch still produces accumulating error.

A spatial-representation experiment uses known camera calibration and scene scale, yielding a triangulation root mean square error of 0.013782 scene-coordinate units. A separate synthetic image-to-state prediction experiment passes 96 of 100 two-frame sphere scenes. Median future-position error among successfully detected scenes is 1.06 pixels; the four detection failures remain in the total sample count. These experiments separately assess geometric localization and prediction from visual state estimates, supporting the connection between knowledge representation and spatial observations.

Numerical prediction, geometric localization, and visual detection use different metrics and assessment conditions. Neither scene-coordinate error nor image-plane pixel error directly measures real-world robot localization accuracy.

D.3 Effects of learned knowledge on subsequent planning

The 24-task experiment uses three seeds and eight numerical-relation families. Each task has 48 training, 32 validation, and 64 test observations and uses isolated storage followed by reloading in a new process. Plans are compared using numerical requirements for three candidate processes and a known fallback process. The conventional target index receives identical knowledge and predictions and changes only the retrieval method.

Independent scoring uses hidden ground-truth relations to assess resource feasibility and actual cost rather than treating the planner's success status as sufficient. Validating and reusing knowledge yields 24 feasible and 17 truly optimal plans, matching the conventional same-knowledge index. Eleven choices change from nonoptimal fallback plans to optimal plans, supporting the value of subsequent knowledge use but not attributing these changes to a unique retrieval-speed advantage.

Table D.3: Feasibility and optimality in 24 learning-to-planning scenarios.

Configuration	**Feasible plans**	**Optimal under the true relation**
Validate and reuse learned knowledge	24/24	17/24
Admit by training fit and reuse knowledge	21/24	18/24
No reuse of newly learned relations; fixed fallback	24/24	6/24
Conventional target index with identical knowledge	24/24	17/24

D.4 Nested processes and additional learning

Combining 16 numerical datasets with 12 process conditions produces 192 tasks, each with two candidate processes. Conditions include balanced or skewed hierarchies of depth three or four, repeated or shared subprocesses, and constraints on fuel, parent-event evidence, and risk. Ground-truth numerical relations and independent constraint checks determine process feasibility, not KNWM's admission decision.

Additional learning allows at most two rounds for relations not yet admitted, each adding 64 training and 32 independent validation observations. Training, validation, and stability thresholds and the expression space remain unchanged. New observations come from an independent synthetic-environment generator; final scoring values do not participate in fitting. The experiment permits additional data and computation and is neither an equal-budget comparison nor an intervention experiment involving autonomous control of real equipment.

The 85 infeasible development tasks test constraint compliance: 32 involve evidence or risk restrictions and 53 involve insufficient cumulative fuel. The 26 uncompleted feasible tasks are a different category: 20 depend on sinusoidal relations and six on cross-threshold piecewise relations, structures not reliably supported within the current learning space. The new-seed group uses the same relation families and does not establish generalization to unknown domains.

Table D.4: Nested-task outcomes with bounded additional learning.

Task group and method	Feasible completed	Feasible rejected	Infeasible accepted	Infeasible rejected
Development: validation and additional learning	81/107	26	0	85/85
Development: training-only admission	87/107	20	13	72/85
New seeds: validation and additional learning	83/108	25	0	84/84
New seeds: training-only admission	88/108	20	14	70/84

D.5 KNOWLEDGE UPDATING AND CLOSED-LOOP ADAPTATION

The updating comparison starts with 100 correct records and then introduces constructed erroneous candidates. At 5% and 10% noise, validated integration retains 98 and 76 correct records, respectively, while direct overwriting retains 19 and 15. The experiment measures resistance to incorrect updates, not equivalent correctness guarantees for every naturally acquired input.

Closed-loop control uses a 64-step horizon, actions in $[-2.5, 2.5]$, position bounds of $[-20, 20]$, and velocity bounds of $[-5, 5]$. Terminal position and velocity tolerances are 1.5 and 0.75. The controller receives only observable inputs, without access to hidden simulator states or action parameters. Completion and full-trajectory safety are scored separately, with safety checked against the realized simulator trajectory.

Table D.5: Closed-loop control and ablations under the same implementation version.

Condition	Tasks completed	Safe throughout trajectory
KNWM under combined uncertainty	78/100	100/100
Without state estimation	52/100	89/100
Without replanning	2/100	22/100

All three conditions use the same observable tasks, environment seeds, bounds, noise, missing observations, and scoring rules. Both ablations have 100-task reruns under the same source version. Completing 78 tasks does not imply that the remaining 22 violate bounds: every trajectory in this group remains within bounds, but 22 fail the terminal-goal requirements.

Table D.6: Additional task groups under combined uncertainty.

Task group	Hidden action gain; bias	Tasks completed	Safe throughout
New-seed rerun	1.0; 0.0	83/100	100/100
Parameter-shift rerun	1.12; 0.08	80/100	99/100
Post-freeze held-out group	1.0; 0.0	77/100	100/100
Post-freeze parameter-shift held-out group	0.8; -0.08	88/100	100/100

One trajectory in the parameter-shift rerun violates the velocity bound. The 100 safe trajectories reported in Section 5.5 therefore refer only to that task group, not every tested condition. State- and parameter-estimation errors affect prediction intervals, and Gaussian disturbances are not strictly bounded. Finite-sample compliance cannot replace a safety guarantee.

Section 5.5 additionally reports two difficulty extensions. Each noise level in law discovery contains 40 paired tasks, with noise added to both training and held-out targets. Strict success requires both

the correct structure and held-out normalized error no greater than 5%. Thus, the 39 structural recoveries and 15 strict successes at 5% noise measure different properties. Each obstacle-count condition contains 20 paired generated scenes with an 80-step horizon. Three through six obstacles yield 20, 18, six, and five completed tasks, respectively, with every task retained in the denominator.

D.6 Lookup efficiency and representation size

The historical 100,000-record experiment times warmed-up lookup of one record by identifier: approximately 0.167 μs for indexed lookup and 1,294.854 μs for sequential scanning, a ratio of approximately 7,753.6. This excludes learning, formula solving, and multi-step planning. The 4.36-fold serialization-size comparison concerns complete knowledge-neuron representations versus simplified records; it measures record-representation overhead, not peak process memory or total application size.

The multi-step retrieval experiment uses seeds 101, 202, and 303, each generating 60 queries covering single-hop, chained, shared-premise, competing-output, and missing-dependency cases. The 180 queries are evaluated in knowledge bases of 1,000 and 10,000 rules. Each scale has 144 answerable and 36 unanswerable queries. Methods use identical dependencies and numerical executors; loading, index construction, and shared expression preparation are excluded from warmed-up query timing.

Table D.7: Retrieval-plus-execution time with 10,000 rules.

Method	**Warm median time (ms)**	**Correctly solved / rejected**
KNWM typed activation	0.6542	144 / 36
Conventional inverted index	0.0183	144 / 36
Exhaustive-scan reasoning	5.1395	144 / 36

The conventional inverted index uses target outputs to locate relations that can produce them, then recursively expands their required inputs. The exhaustive method passes all visible relations to the same executor. Each warmed-up query is repeated three times with randomized method order, and timings are first aggregated within each query. This separates the benefits of avoiding scans from the overhead of typed activation, record handling, and diagnostics.

E Physics problem inputs and adjudication

E.1 Data source and executable inputs

The 93 development-test problems in Section 5.7 come from a fixed snapshot of PhysicsQA (maximus-21, n.d.). Human- and AI-assisted transcription represents each problem as objects, states, known quantities, units, constraints, and goals. Physical conditions are made explicit for some tasks, with provenance retained. The solving stage uses KNWM's knowledge-and-operation composition route without calling a language model or accessing private reference solutions.

Physical conditions can be reusable knowledge, but conditions explicitly supplied during this transcription must not be described as autonomously inferred from the problem statement. Phase alignment, communication-band ranges, or the absence of guard bands, for example, can affect applicable relations and final results. Original problem information, supplementary conditions, and executable inputs are recorded separately to distinguish externally supplied conditions from system-derived conclusions.

These problems were used during development and failure analysis. The total solving budget is 30 seconds per problem, with a 45-second outer timeout. Executed inputs, predictions, and call traces are sealed before scoring. The results characterize knowledge composition given structured inputs; they are neither natural-language end-to-end accuracy nor an independent evaluation on wholly unseen tasks.

E.2 Eligible problems and answer adjudication

Eligibility excludes records with missing essential information or clear conflicts between the question and its source, not records excluded merely because solving failed. Original identifiers are retained: problems 35, 46, 52, and 97 lack necessary text or figures; problem 10 has a unit-scale conflict; problem 63 has an inconsistent subtraction direction and anomalous units; and problem 70 has a typographic or numerical conflict between 106 and 10^6. The latter three were excluded after review, not through a pretest eligibility decision.

Table E.1: Outcomes for 93 eligible development-test problems.

Outcome	Problems	Counted as correct
Answer content confirmed correct	82	Yes
Valid under a specific interpretation or supplementary conditions	4	No
Disagreement with source reference not fully adjudicated	5	No
Unsolved	2	No
Total	93	82

The main result is 82/93, or 88.17%. It combines sealed outputs with project-internal content review: 68 results passed the original scoring, and 14 of 21 flagged results were subsequently confirmed correct in answer content. This is not a new solver run or a claim that all 93 answers underwent independent external review. Symbolic expressions, units, directions, and choices are checked against each problem's requirements rather than scored solely by extracting a number.

The 14 confirmed results are problems 3, 18, 27, 28, 36, 41, 48, 57, 60, 71, 80, 83, 92, and 93. Conditional results are problems 9, 15, 33, and 84; unresolved reference discrepancies are problems 6, 25, 30, 32, and 73; unsolved problems are 8 and 91. Problem 36 concerns an exact value versus a truncated reference, and problem 71 contains duplicate choices. Their adjudication is documented by answer content without overwriting the original scores.

E.3 Distinction from the historical language-model comparison

The 900-task historical comparison has nine categories of 100 tasks each. KNWM, GPT-5.6 Sol, Qwen2.5-VL-7B, and the task-routed symbolic-tool comparator pass 900, 895, 417, and 900 tasks, respectively. This comparison shows the importance of access to tools and explicit relations. Its questions, inputs, and scoring differ from the 93-problem evaluation, so the results cannot be pooled into one overall accuracy or used to infer superiority over all language models.

Task generation used the fixed random seed 20260714. Each language model received 210 requests with identical public task instructions and evidence. Batches contained five cases for discovery, retrieval, and prediction; ten for composition, object-label generalization, counterfactual simulation, inverse planning, and causal-evidence interpretation; and one for analogy. Application tasks supplied the required laws; causal tasks supplied fixed intervention evidence. Neither model had solver or simulator access.

The hosted baseline requested gpt-5.6-sol through ChatGPT-authenticated Codex CLI 0.144.0-alpha.4 with medium reasoning effort. Independent ephemeral sessions ignored user configuration and rules and used an empty read-only workspace. Task-specific JSON schemas constrained output; the 210 sessions recorded no tool calls. No explicit temperature or output-token limit was set. Only transport failures without model output or token usage could be retried; answer-bearing responses were not retried.

The local baseline called Ollama tag qwen2.5vl:7b via /api/generate: temperature 0, JSON output, streaming disabled, and think=false. Output limits (num_predict) were 1,024 tokens for discovery and retrieval, 2,048 for prediction, 512 for analogy, and 1,536 otherwise. Context length and sampling seed were not overridden. Retained metadata identifies neither an immutable hosted checkpoint nor the local weight digest, quantization, or Ollama version; the configurations are therefore not fully pinned model snapshots.

All 900 cases remained in the denominator. Numeric checks used absolute and relative tolerances of 0.001; all trajectory and counterfactual values had to pass, and retrieval required the exact minimal law set. Missing IDs and invalid answers counted as incorrect; Qwen omitted 22 IDs. The five GPT errors involved retrieval-slot aliases incompletely specified in its prompt, so the gap cannot be attributed solely to reasoning. The symbolic comparator directly dispatched tools by task type without new model calls; its 900/900 does not demonstrate autonomous tool selection by Qwen.

F Evidence traceability and implementation scope

F.1 Distinguishing architectural descriptions from experimental instances

Figures A.1-C.2 describe knowledge organization and operations; the experiments in Section 5 instantiate different parts of that design. Simulator intervention, numerical-candidate validation, reloading after learning, nested-event execution, and physics problem solving use different input interfaces. Sharing KNWM knowledge structures does not mean that every experiment exercises the entire learning, perception, and action process.

A knowledge record's validation status must correspond to the evidence supporting it. Held-out numerical validation supports prediction under tested conditions; controlled interventions support judgments about influence; and event-simulation receipts support transition checks in the simulated state. Imported definitions, manually curated basic relations, and programmatically checked synthetic solution trajectories retain their distinct provenance. They are not treated as real student experiences or as natural laws independently discovered by the system.

F.2 Experimental records and reproduction procedure

Reproduction checks verify frozen source and protocol fingerprints, then recompute aggregates from stored inputs, predictions, trajectories, and independent scores. Reruns retain the corresponding source, environment, seeds, budgets, inputs, and scoring. New-seed evaluations, development retests, and software regressions remain separate.

Reproduction materials link results to configurations, code, inputs, predictions, scoring, and environments. Personal knowledge stores, credentials, facial data, and the full desktop application are not required.