CDRL: Certification-Driven Reinforcement Learning for Neutrino Flavor Model Discovery

Piyush Jha[1,2] **Jake Rudolph**[3] **Victoria Knapp-Pérez**[3,6] **Max Fieg**[4]
Aishik Ghosh[2,5] **Vijay Ganesh**[1]

[1]School of Computer Science, Georgia Institute of Technology, USA
[2]School of Physics, Georgia Institute of Technology, USA
[3]University of California, Irvine, USA [4]Particle Theory Department, Fermilab, USA
[5]Lawrence Berkeley National Laboratory, USA
[6]Halluminate, Department of Research, USA

piyush.jha@gatech.edu, jirudolp@uci.edu, vknapppe@uci.edu,
mfieg@fnal.gov, aishikghosh@physics.gatech.edu, vganesh@gatech.edu

Abstract

Many scientific discovery problems require searching combinatorial hypothesis spaces under complex domain constraints. Reinforcement learning (RL) offers a promising approach, but existing methods rely on scalar rewards that provide limited information about why candidate solutions fail, leading agents to repeatedly explore invalid regions. We introduce Certification-Driven Reinforcement Learning (CDRL), a framework that leverages structured feedback from symbolic reasoning tools. When a candidate violates domain constraints, these tools produce certificates identifying the *actions* responsible for failure. CDRL converts these certificates into reusable constraints that eliminate classes of invalid solutions and guide exploration toward valid regions. We evaluate CDRL on neutrino flavor model discovery in theoretical particle physics, where the hypothesis space exceeds 10^{26} possible models, and compare it with the state-of-the-art RL approach previously used for this task. Across three theory spaces, CDRL achieves up to $1.95\times$ higher valid model rates and up to $6.33\times$ higher neutrino model rates while evaluating up to $4\times$ fewer candidates. We further extract 40 interpretable rules from search trajectories using a post-hoc decision-tree framework and show that reusing them as soft constraints yields gains of up to $2\times$ in valid model rates and $3\times$ in neutrino model discovery across all three theory spaces. These results suggest that CDRL uncovers reusable structure in combinatorial search spaces and provides a general framework for scientific model discovery.

1 Introduction

Scientific discovery often requires exploring large spaces of candidate hypotheses. Researchers propose mathematical models, derive predictions, and compare them with experimental observations. This iterative process lies at the core of many scientific disciplines, including particle physics [1, 2, 3, 4], computational biology [5], chemistry [6], and materials science [7]. In many such settings, the space of candidate models is combinatorial, and small changes in model structure can lead to dramatically different predictions. Systematically exploring these large theoretical spaces therefore becomes a major computational challenge, motivating the development of AI systems that assist scientific discovery and exploration [8, 9, 10].

Preprint.

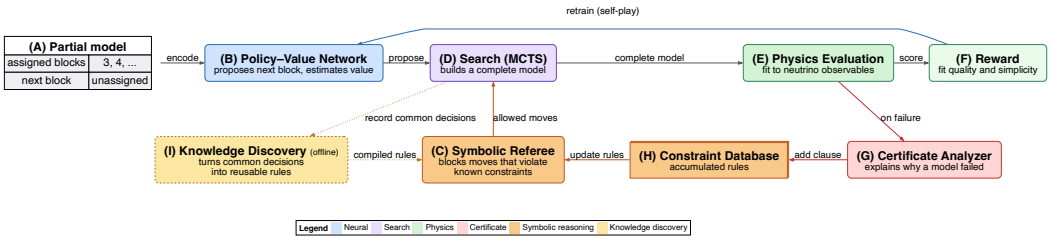


Figure 1: **The CDRL loop.** CDRL builds each model one block at a time (A) using a policy-value network and search (B, D) [18, 19]. Before the search commits to a block, a symbolic referee (C) rules out moves that are already known to violate a domain constraint, so search time is spent only on moves that remain possible. Once a model is complete, it is scored by the physics pipeline (E, F; Section 4.4 defines the score and what counts as a good model). When a model fails, the certificate analyzer (G) identifies which specific block assignments caused the failure and compiles that into a reusable rule stored in the constraint database (H), so the referee (C) blocks the same failure pattern, and every equivalent model that would have made it, everywhere it recurs in future search. Separately, and only after training, search knowledge discovery (I) mines the most common decisions the search made into simple reusable rules that also feed the referee (C). Section 4.3 gives the formal, SAT-based definition of the referee and constraint database, and Table 2 works out the ways a certificate can arise. Figure 7 (Appendix A) walks through this loop step by step on a worked example.

A common approach to exploring large hypothesis spaces is reinforcement learning (RL) [11], which formulates discovery as a sequential decision-making problem. In these frameworks, an agent incrementally constructs candidate models and receives rewards based on whether they satisfy domain-specific criteria. RL has been applied successfully to automated theorem proving [12], program synthesis [13], and scientific discovery [14, 15, 16]. However, these methods typically rely on scalar reward signals that provide little information about why a candidate fails [17]. Consequently, agents often waste substantial effort repeatedly exploring similar invalid regions of the combinatorial search space.

In many scientific domains, structured feedback beyond scalar rewards is available, yet traditional RL algorithms do not fully exploit it. External reasoning tools such as theorem provers, constraint solvers, computer algebra systems, program analysis tools, and simulators can analyze candidate solutions and identify the causes of failure. Because these systems operate on formally defined objects, they can often produce *certificates* [1] that describe the outcome of their analysis and pinpoint the components responsible for violated constraints. Such certificates provide structured information that can be leveraged to guide exploration more effectively than scalar reward signals alone. Crucially, the two signals are complementary rather than competing (Figure 2).

Further, scientists have traditionally been able to distill useful heuristic rules from the problem-solving process, and these insights can accelerate future exploration of the underlying mathematical space. An RL approach that enables the extraction of such heuristic rules from the agent's decisions would therefore be valuable, both for interpretability and for improving computational efficiency in subsequent runs.

There is a growing interest in using RL to construct theories in particle physics [20, 1, 21, 22]. A viable theory, or model, is subject to a number of consistency checks such as requiring that Lorentz invariance and other internal symmetries are respected, as well as demands that the theory reproduce certain observed qualities like the mass of a certain particle. When an RL agent attempts to construct a model, it will generically fail these checks. However, if the RL agent had feedback in the form of a certificate concerning the failure of a candidate model, it could be trained to build viable models more quickly. To demonstrate the potential advantages of fully incorporating such feedback, we consider models that can be constructed for neutrino masses [1], which enables rapid exploration of mathematical models where physicists have limited intuition.

[1] A certificate is a structured explanation produced by a reasoning tool that identifies the subset of decisions responsible for satisfying or violating domain constraints. See Section 3 for a formal definition and discussion.

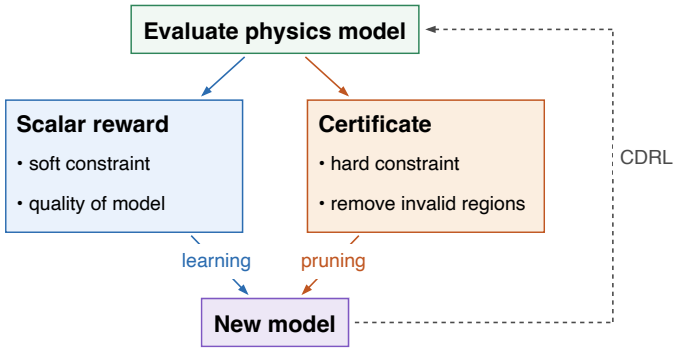


Figure 2: **The two feedback signals CDRL gets from each physics evaluation are complementary rather than competing.** Evaluating a candidate model yields a *scalar reward*, a *soft* constraint that scores the quality of the model, and a *certificate*, a *hard* constraint that names the exact components responsible for a failure and forbids them via a clause that Boolean Constraint Propagation (BCP) then enforces everywhere that pattern would recur. The reward shapes the network through *learning*, while the certificate *prunes* the search space; together they determine the new model the search proposes next, and the cycle repeats. Section 3 gives the formal definition of a certificate.

In our setup, an agent selects the particle content of a model and the symmetries it must satisfy. These constraints are encoded via irreducible representations of a symmetry group, yielding a Lagrangian density that defines a quantum field theory. The resulting models are evaluated using existing symbolic and numerical software for Lagrangian construction, mass-matrix extraction, and parameter fitting, followed by statistical comparison to experimental data. Exhaustive search over more than 10^{26} possible models is infeasible, as evaluating a single candidate requires non-trivial symbolic and numerical computation.

Certification-Driven Reinforcement Learning (CDRL) is a framework that leverages structured feedback from external reasoning tools to improve exploration in constrained combinatorial search spaces. When a candidate violates a domain constraint, the reasoning tool returns a certificate identifying the subset of assignments responsible for the failure. CDRL analyzes this certificate and converts the responsible assignments into a symbolic conflict clause, which is added to the global SAT constraint database [23]. During search, these learned constraints are enforced by a lightweight symbolic reasoning layer that immediately eliminates partial candidates violating known domain rules, allowing the RL agent to avoid exploring impossible regions of the search space.

During the MCTS rollouts that follow [18, 19], this symbolic reasoning layer is implemented via Boolean Constraint Propagation (BCP) [24, 23], which enforces the accumulated clauses as the agent builds new candidates, blocking exploration of any partial assignment that repeats a previously seen conflicting decision pattern. This differs from a negative reward in *kind*, not degree. A negative reward is a scalar that adjusts the network's weights by a small gradient step; it lowers the *probability* of a decision but never rules it out, so the agent can, and empirically does, revisit the same dead end many times, and the penalty says nothing about *which* of the decisions was to blame. A conflict clause instead operates on the search space itself: it is a hard logical constraint naming the exact assignments responsible, and once added, BCP makes every partial assignment containing that pattern *unreachable* for all agents sharing the clause database, with no dependence on how well the network has been trained. In effect, a scalar reward changes where the agent *prefers* to go, whereas a certificate changes where it *can* go. As more certificates accumulate, the feasible search space is progressively pruned, allowing the agent to avoid rediscovering equivalent failure modes and focus expensive physics evaluations on regions more likely to contain valid theories.

As a concrete example of this structured feedback in our setting, consider a candidate neutrino model whose non-Abelian A_4 representation assignment is invalid for *any* choice of $\mathbb{Z}_N$ charge (Figure 1). Rather than emitting only a scalar penalty for this one candidate, our system extracts a *certificate* identifying the representation pattern responsible and compiles it into a logical constraint that blocks that pattern across the whole charge sector, so it is masked the moment it reappears

Table 1: **Example neutrino flavor model discovered by CDRL in the $T_{19} \times \mathbb{Z}_4$ theory space.** The table shows particle representation and charge assignments for one of the viable neutrino models discovered by CDRL. This model achieves a strong χ^2 fit with a minimal number of free parameters, illustrating the ability of CDRL to discover compact and high-quality theories. Additional discovered neutrino models across all three theory spaces are provided in Section C.

	L	E_1	E_2	E_3	N	H_u	H_d	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6
T_{19}	$\mathbf{3}_1$	$\mathbf{1}''$	$\mathbf{1}$	$\mathbf{1}'$	$\bar{\mathbf{3}}_2$	$\mathbf{1}'$	$\mathbf{1}$	$\mathbf{3}_1$	$\bar{\mathbf{3}}_1$	$\mathbf{3}_1$	$\bar{\mathbf{3}}_1$	$\mathbf{3}_2$	$\mathbf{3}_2$
$\mathbb{Z}_4$	4	2	2	2	4	2	4	2	2	3	4	4	1

elsewhere in the search tree, before any physics evaluation is run on it (worked out in detail in Section 4). Concretely, we realize this loop with well-established tools: candidate models are encoded as a Boolean satisfiability (SAT) formula in conjunctive normal form; certificates become *conflict clauses* that are enforced during search by Boolean Constraint Propagation (BCP) [24, 23], the same unit-propagation mechanism used inside modern SAT solvers [23]; and the search itself is an AlphaZero-style Monte Carlo Tree Search (MCTS) [18, 19] guided by a policy-value network. These three components play distinct, complementary roles at every decision: neural guidance *proposes* the next assignment, symbolic reasoning *prunes* the assignments a certificate has ruled out, and MCTS *balances* exploration against exploitation over what remains. We quantify the payoff of this structured feedback empirically across three neutrino theory spaces (Section 5), and further show that the constraints and decision patterns CDRL accumulates expose reusable structure in the theory space (Section 5.4).

Across three theory spaces for neutrino flavor model discovery, CDRL achieves up to $1.95\times$ higher valid model rates and up to $6.33\times$ higher neutrino model rates than AMBer [1], while evaluating up to $4\times$ fewer candidates. These gains are expected to scale with the complexity of the problem and indicate substantially more efficient navigation of the underlying combinatorial hypothesis space, similar in spirit to recent advances in algorithm discovery and search-based scientific reasoning [25, 26, 10, 9].

Beyond improved search efficiency, CDRL enables search knowledge discovery from search trajectories. Using a post-hoc decision-tree analysis of high-confidence MCTS states, we extract 40 interpretable rules that capture recurring dependencies between particle assignments. Reusing these rules as soft constraints further improves performance, suggesting that they capture meaningful and reusable structure within the theory space.

Our contributions are therefore: (i) CDRL, a certificate-driven RL framework that transforms symbolic failure certificates into reusable search constraints, progressively eliminating large invalid regions; (ii) state-of-the-art results on neutrino flavor model discovery, achieving substantially higher discovery rates with fewer evaluations than prior RL approaches; and (iii) a framework for extracting and reusing interpretable rules from search trajectories, demonstrating that knowledge acquired during exploration can improve future search.

2 Background and Related Work

Neutrino Flavor Model Construction. The Standard Model (SM) of particle physics describes the fundamental particles and their interactions and has exquisite agreement with experiment, with some exceptions. Perhaps the most notable exception is the prediction of massless neutrinos, but experiments [27, 28, 29, 30] have determined that they have a mass, which necessitates the presence of new particles beyond the SM. Because these new particles must couple to neutrinos, and because neutrinos and charged leptons are components of the same electroweak doublets, one generally expects them to couple to charged leptons as well. For this reason, solutions to the massive neutrino problem are typically connected to the question of flavor in the SM [31, 2, 32, 33]: what is the explanation for the three flavors of fermions and what is the origin for the observed levels of mixing between them?

One general framework to address these shortcomings is to add right-handed neutrinos to generate small neutrino masses via the seesaw mechanism [34, 35, 36, 37], as well as a number of scalar particles, "flavons", whose interactions give structure to the level of lepton mixing after they acquire a certain vacuum expectation value (VEV). Adding these new particles generally results in an even

larger number of free parameters than was already present in the SM, which limits the predictivity of the model. However, the parameters in the flavor sector can be strongly constrained if the model is subject to a global flavor symmetry, which gives structure to the observed mixing; in the ideal case it explains the observed values with minimal free parameters. A given theory under this framework is thus described by: the set of particles, the irreducible representations they take under the flavor symmetry, the VEV of the flavons[2], and the remaining free parameters. The number of models that exist for a given choice of flavor symmetry grows rapidly with the number of particles, and identifying solutions that fit to experiment is highly nontrivial.

Recent work [38, 39, 40, 41, 42] has demonstrated RL as a tool for searching analogous spaces of models for various areas in particle physics. In particular, the AMBer framework [1] integrates domain specific physics tools which allows the agent to evaluate candidate models using software used by physicists for conventional model-building. CDRL goes further: rather than using these tools only to evaluate candidates one at a time, it extracts structured certificates from them and converts these into reusable constraints that prune invalid regions of the search space, letting the system progressively accumulate knowledge about the structure of valid theories. This approach can be extended to several other applications of agent-guided workflows to scientific discovery.

AI for scientific discovery. Artificial intelligence is increasingly used across the scientific discovery pipeline, including hypothesis generation, experimental design, and large-scale data analysis [14, 43]. Recent advances highlight the ability of AI systems to explore large scientific search spaces, such as materials discovery using graph neural networks and generative models [8, 44], and biomolecular structure prediction using deep learning systems like AlphaFold [9]. Machine learning is also widely used in particle physics for analyzing experimental data and identifying potential new phenomena [45]. Alongside these developments, there is growing interest in incorporating domain knowledge and reasoning into AI-driven discovery processes [46, 47]. In this work, we take a step in this direction by introducing CDRL, which integrates symbolic reasoning tools into the learning loop to provide structured and interpretable feedback during exploration.

Reinforcement learning for algorithm and scientific discovery. Reinforcement learning (RL) has discovered algorithms and optimized complex scientific systems, including AlphaTensor for tensor decomposition [25], AlphaTensor-Quantum for fault-tolerant quantum circuits [26], and AlphaDev for faster sorting [10]. It has also controlled physical systems and experiments such as tokamak plasmas [48], stratospheric balloons [49], and quantum experiment design [50]. These results show RL can navigate large combinatorial search spaces, but they depend on reward signals from a simulator or environment. CDRL instead adds symbolic reasoning tools that emit structured certificates, giving the agent richer feedback to guide exploration.

Neuro-symbolic AI. Neuro-symbolic AI combines neural learning with symbolic reasoning to strengthen reasoning, interpretability, and reliability [51, 52]. Integration strategies range from embedding symbolic rules within neural architectures and differentiable reasoning modules to modular systems that call external reasoning tools [53]. CDRL takes the modular route: a learning agent queries external symbolic systems that analyze candidate solutions and return structured feedback. In the taxonomy of [53], CDRL sits at the center of the neuro-symbolic design space, uniting learning, symbolic reasoning, and structured knowledge within a single decision loop.

Oracle-augmented learning. Recent work has explored integrating external reasoning tools into AI systems. Large language models have been combined with symbolic tools during inference [54, 55, 56, 57], while other approaches incorporate external feedback during RL [58, 17]. Related ideas also appear in automated reasoning and synthesis. Modern SAT solvers use conflict feedback to guide search [59], and frameworks such as CEGIS and Satisfiability Modulo Oracles refine candidates using counterexamples or constraints from external verifiers [60, 61, 62]. CDRL builds on these directions by converting certificates from reasoning systems into reusable constraints, enabling RL agents to progressively prune invalid regions of large scientific search spaces.

Together, these observations motivate Certification-Driven Reinforcement Learning (CDRL), a paradigm in which symbolic reasoning systems act as structured feedback generators that guide RL over large scientific search spaces.

[2] In a complete model, the VEV is a dynamical result of the theory, and achieving a certain VEV pattern is a problem in its own right. Here, as in Ref [1], we do not attempt to solve this problem, but assume that a certain set of VEVs can be achieved

Algorithm 1 Certification-Driven Reinforcement Learning (CDRL)

Notation: π_θ policy-value network; V verifier (the physics evaluation pipeline); $\mathcal{A}$ certificate analyzer; $\mathcal{K}_0, \mathcal{K}$ initial and accumulated constraint sets; H space of candidate hypotheses (models); $h \in H$ one candidate; $\mathcal{H}_{\text{valid}}$ set of discovered valid hypotheses; σ a certificate returned by V on failure; c the constraint derived from σ.

Require: Policy π_θ, verifier V, certificate analyzer $\mathcal{A}$, initial constraints $\mathcal{K}_0$
Ensure: Set of valid theories $\mathcal{H}_{\text{valid}}$

$\mathcal{K} \leftarrow \mathcal{K}_0$ ▷ initialize with known domain constraints
$\mathcal{H}_{\text{valid}} \leftarrow \emptyset$ ▷ store discovered valid solutions
for each training episode **do**
  $h \sim \textsc{Plan}(\pi_\theta, \mathcal{K})$ ▷ construct candidate while respecting accumulated constraints
  $(status, \sigma) \leftarrow V(h)$ ▷ evaluate candidate and optionally return failure certificate σ
  if $status =$ valid **then**
    $\mathcal{H}_{\text{valid}} \leftarrow \mathcal{H}_{\text{valid}} \cup \{h\}$ ▷ record valid hypothesis
    Store trajectory and reward in replay buffer
  else
    $c \leftarrow \mathcal{A}(\sigma)$ ▷ convert certificate into reusable constraint
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{c\}$ ▷ prune entire class of invalid solutions
    Store trajectory and penalty in replay buffer
  end if
  Periodically update π_θ from replay-buffer mini-batches
end for

3 Certification-Driven Reinforcement Learning (CDRL)

Many scientific discovery tasks can be viewed as structured search problems over a large space of candidate hypotheses. An agent incrementally constructs candidate solutions while interacting with an environment that verifies whether the candidate satisfies domain constraints.

Certificate-Driven Reinforcement Learning (CDRL) augments this RL loop with certificate-based constraint learning. Each time the verifier rejects a candidate, the certificate it returns is also converted into a reusable constraint that rules out the whole class of similarly invalid candidates, not just the one just tried.

Formally, let H denote the space of candidate hypotheses and let $\mathcal{K}_0$ denote a set of initial domain constraints known a priori. The agent maintains a growing set of constraints $\mathcal{K}$ initialized with $\mathcal{K}_0$. During training, the agent proposes candidate hypotheses using a planning procedure $\textsc{Plan}(\pi_\theta, \mathcal{K})$ (such as MCTS), which incrementally constructs candidates while enforcing constraints $\mathcal{K}$ and using π_θ to guide exploration. Each candidate is verified by an external reasoning tool. If the candidate satisfies all constraints, the agent receives a positive reward. Otherwise, the verifier produces a certificate describing the failure. This certificate is analyzed to derive a new constraint, which is added to $\mathcal{K}$ and used to prune the search space.

The overall CDRL procedure is shown in Algorithm 1. As training progresses, the learned constraints accumulate and progressively restrict exploration to regions of the hypothesis space that remain consistent with previously discovered knowledge. This allows the RL agent to avoid rediscovering similar failure modes and focus exploration on configurations that are more likely to yield valid solutions.

Certificates and Constraint Learning. A certificate is a structured explanation produced by a reasoning tool when a candidate hypothesis violates domain constraints. It identifies the subset of decisions responsible for the failure. Let $h \in H$ denote a candidate hypothesis. When h is invalid or redundant with an already-covered equivalent hypothesis, the verifier returns a certificate σ, which is analyzed to derive a constraint c that prevents similar configurations. This constraint is added to $\mathcal{K}$ and enforced during future exploration, progressively pruning the search space. This mechanism is conceptually related to conflict-driven learning in SAT solvers, where conflicts yield clauses that eliminate inconsistent assignments. In CDRL, certificates play an analogous role by enabling reusable constraint learning.

4 Application: Neutrino Flavor Model Discovery

We evaluate CDRL on the discovery of viable neutrino flavor models, a central problem in particle physics. The task involves constructing models that explain neutrino masses and mixing while satisfying domain constraints. We follow a common flavor symmetry model building framework, where candidate models are defined by non-Abelian symmetry choices, particle irreducible representation assignments, Abelian charges, and vacuum expectation values. This results in a large combinatorial search space, making efficient exploration critical. Figure 1 shows how the components below fit together; further background is provided in Section 2.

4.1 Matrix Representation of Candidate Models

Following the AMBer framework [1], we encode a candidate theory as a matrix representation that captures the assignment of particles to symmetry representations and charges. Rows correspond to particles in the model, while columns correspond to irreducible representation and charge assignments under the symmetry groups. For example, rows include the three charged lepton doublets (L_1, L_2, L_3), charged lepton singlets (E_1, E_2, E_3), right-handed neutrinos (N_1, N_2, N_3), Higgs fields (H_u, H_d), and several flavon fields. Columns represent discrete non-Abelian representations (e.g., A_4 irreducible representations), Abelian $\mathbb{Z}_N$ charges, and VEV configurations for flavon fields. Columns are encoded using one-hot vectors. Unassigned entries are represented by -1, allowing the state to represent partially constructed models. This matrix therefore forms the state representation for the learning agent.

4.2 Sequential Construction of Models

Model construction is formulated as a sequential decision process. Instead of assigning every matrix entry independently, decisions are organized into blocks corresponding to logical modeling choices. The agent starts by assigning the active $\mathbb{Z}_N$ symmetry as one block, then for each particle in turn assigns its representation and then its Abelian charge as two separate blocks, and finally assigns each flavon's VEV configuration as its own block. At each step, the next unassigned block is selected, and the agent chooses from a small alphabet consisting of either an all-zero vector (inactive block) or a one-hot vector indicating the assignment. This structured action space reduces the branching factor and enables incremental construction of candidate theories until completion.

4.3 SAT-Based Constraint Layer

Rather than allowing the RL agent to freely construct candidate models, discovering only at the end that they violate domain rules, we incorporate a symbolic constraint layer that continuously enforces known scientific constraints during search. This layer immediately rejects partial assignments that cannot be extended into a valid theory, allowing the agent to focus its exploration on feasible regions of the combinatorial search space.

We encode the domain knowledge as a set of Boolean constraints, including exactly one representation per particle, a single active $\mathbb{Z}_N$ symmetry (a modeling choice that reduces the search space), exactly one charge per particle for that active $\mathbb{Z}_N$ symmetry, structural constraints for triplet representations, and exactly one vacuum expectation value (VEV) assignment per flavon. These constraints are compiled once into conjunctive normal form (CNF).

During the MCTS search, we use Boolean Constraint Propagation (BCP) [24, 23] at every node to maintain consistency between the current partial assignment and the encoded constraints. Whenever the agent makes a new assignment, BCP efficiently detects whether the partial model already violates a constraint and automatically infers any assignments that are logically forced. As a result, invalid branches are pruned immediately, while the state presented to the neural network always satisfies all currently known hard constraints. We therefore use BCP as a lightweight symbolic reasoning layer that filters illegal actions without requiring a full SAT solve after every decision.

4.4 RL environment and reward

We use the same particle physics evaluation pipeline as AMBer [1] for direct comparison. A terminal state corresponds to a fully specified model matrix, which is passed to the physics tools. Using

Table 2: **The three certificate types**, each worked through on a common $\mathbb{Z}_6$ candidate; the full derivation follows below. Certificates 2 and 3 depend only on the $\mathbb{Z}_N$ charges, independent of representations or VEVs. Every clause is $\neg$(the equivalent or excluded pattern); once added to the shared constraint database, it is enforced across all parallel searches with no physics evaluation spent on models it already rules out.

Description	Example	Clause
Insufficient-rank representation		
Model already rank-deficient without $\mathbb{Z}_N$ symmetry	Reps and VEVs with $\mathbb{Z}_6$ charges wildcarded already rank-deficient	$\neg$(reps & VEV, any charges)
Abelian symmetry reducibility		
$\mathbb{Z}_N[q_i] \cong \mathbb{Z}_{N/d}[q_i/d]$	$\mathbb{Z}_6\ [2,4,6] \mapsto \mathbb{Z}_3\ [1,2,3]$	$\neg$(other representative)
Charge negation		
$\mathbb{Z}_N[q_i] \cong \mathbb{Z}_N[-q_i \bmod \mathrm{N}]$	$\mathbb{Z}_6\ [2,4,6] \mapsto \mathbb{Z}_6\ [4,2,6]$	$\neg$(negated charges)

`Model2Mass`, the particle assignment matrix is compiled into a Lagrangian and the resulting Dirac and Majorana mass matrices are constructed, from which the light-neutrino mass matrix is obtained through the seesaw mechanism. Using `flavorpy`, the model's free parameters are then fit to neutrino observables from NuFit 5.3 [63] and scored by a chi-squared goodness of fit χ^2. A model is *invalid* if it fails a structural check (a rank-deficient mass matrix, so no physical spectrum exists) or is inelegant (more than 50 free parameters); such models receive a strong penalty. Valid models are scored by a reward that combines (i) the chi-squared fit quality and (ii) the number of free parameters n_p, normalized and clipped to a bounded range so that better fits with fewer parameters score higher (an equal weighting of the two terms by default). We designate a valid model a *neutrino model* when it additionally satisfies $\chi^2 \leq 10$ and $n_p \leq 7$. Because each evaluation requires expensive numerical optimization and dominates runtime (approx. 98%; Section 5), avoiding invalid and redundant regions of the search space before evaluation is critical. Full experimental and compute details are given in Appendix B.

4.5 Certificate Extraction and Conflict Clauses

Beyond scalar rewards, CDRL extracts structured certificates from invalid candidate models and converts them into reusable *conflict clauses* added to the global constraint set $\mathcal{K}$. We implement this with a dedicated certificate-analysis routine that runs on every terminal model. It inspects the completed model matrix, tests it against the criteria below, and, when a criterion fires, identifies the exact subset of assignments responsible and emits their negation as a clause. Because a clause names the responsible assignments rather than the single candidate, one certificate prunes an entire class of models and lets the agent avoid revisiting similar configurations. We consider two main classes: (i) *insufficient-rank representation certificates*, which arise when a candidate's non-Abelian representations and VEV choice yield a rank-deficient mass matrix for every choice of Abelian charge, and eliminate that specific representation-and-VEV combination independent of the charges; and (ii) $\mathbb{Z}_N$ *symmetry certificates*, which depend only on the Abelian charges and capture two group-theoretic equivalences: charge assignments reducible to a smaller symmetry group by a shared divisor d (checked in both directions between the smaller and larger group), and charge assignments related by negation $q \mapsto -q \bmod N$, an automorphism of order 2, both of which preserve the theory's selection rule. Table 2 demonstrates how each type of certificate works on a common $\mathbb{Z}_6$ candidate. Together, these certificates encode reusable symbolic knowledge that progressively removes invalid and redundant regions from the search space.

We now give the full construction of each certificate type. The first class arises when a candidate model's non-Abelian representations and VEV choice cannot support a full-rank mass matrix under any Abelian charge assignment. After constructing a candidate matrix, the system evaluates the model using the physics validity checker. If the model fails, the algorithm performs a reduced analysis in which the Abelian charge sector is temporarily wildcarded and only the non-Abelian representation and VEV assignments are retained. This has the effect of removing the Abelian symmetry entirely. When the representation-and-VEV-only model is already rank-deficient, no

Abelian charge assignment layered on top of it can restore full rank, so the failure must be attributed solely to the non-Abelian representations and VEV choice. The system extracts the corresponding representation and VEV assignments and generates a conflict clause blocking this combination; other VEV choices for the same representations remain unaffected and available for exploration. This allows the solver to prune large classes of models that share the same insufficient-rank structure.

The second class arises from symmetry relations in the Abelian $\mathbb{Z}_N$ charge sector, independent of the non-Abelian representations or VEV choice, and captures two group-theoretic equivalences between charge assignments that describe the same physics. The first is reducibility. If all charges in a $\mathbb{Z}_N$ model share a common divisor d, then the model is reducible to a $\mathbb{Z}_{N/d}$ model. Formally, if

$$q_i = da_i, \qquad N = dM,$$

then the selection rule

$$q_1 + q_2 + q_3 = 0 \pmod{N}$$

is equivalent to

$$a_1 + a_2 + a_3 = 0 \pmod{M}.$$

This implies that the larger symmetry provides no additional structure beyond the reduced group $\mathbb{Z}_M$. Whichever model is evaluated first, the certificate blocks the other, and the relation is checked in both directions (a smaller model also blocks the larger models it embeds into). The system generates a conflict clause by reconstructing the equivalent model in matrix form and negating the literals corresponding to that assignment. The second equivalence is relation by an automorphism, specifically the order-2 automorphism of negation. Replacing every charge with its additive inverse, $q \mapsto -q \bmod N$, preserves the selection rule $\sum_i q_i \equiv 0 \pmod{N}$ (negating every term preserves a sum of zero), so the negated charge assignment describes the same physics and need not be explored separately. Higher order automorphisms could also be included in future applications.

4.6 Learned Clause Injection

The conflict clauses generated during evaluation are accumulated globally and injected into the SAT solver before each new search episode, requiring no recompilation since each clause is already in CNF. Detection then falls out of BCP itself: at every MCTS node the current partial assignment is fed to the solver, and to score a candidate block assignment we tentatively add its literals and propagate. If propagation drives any learned clause to a conflict, that block assignment is removed from the legal-action mask before the policy network ever ranks it (Section 4.3). Instead, a forced literal implied by the clauses is propagated into the state. Figure 1 traces this loop end to end: a conflict at a completed model is analyzed into a clause, appended to the shared database, and the clause then masks the same failing pattern the moment it recurs elsewhere in the tree, before any physics evaluation is spent on it. This is precisely where the neuro-symbolic design pays off over a purely neural agent: the learned clause makes the whole equivalence class logically unreachable for every worker (Figure 2), so large regions of the search space are pruned permanently rather than merely down-weighted.

4.7 MCTS, Neural Guidance, and Portfolio Parallelization

We search the space of partial model matrices using MCTS [18, 19]. Each node corresponds to a partially constructed candidate. At each step, a policy-value network proposes the next assignment and estimates the state's value. The policy is masked by BCP's legal-action vector (Section 4.3), ensuring only constraint-consistent actions are explored, and the masked priors guide tree traversal via PUCT [64]. At terminal states, the physics evaluation pipeline computes a reward that is backpropagated through the tree and used to retrain the network via self-play. This tightly couples all three components at every decision: neural guidance proposes, symbolic reasoning prunes, and MCTS balances exploration against exploitation. Removing any one causes substantial degradation and neutrino discovery collapses to near zero without either the network or MCTS (Table 5).

To scale search, we employ portfolio-style parallelization inspired by parallel SAT solvers [65, 66] and distributed RL [67]. Multiple workers independently perform MCTS self-play while sharing (i) a replay buffer for network retraining and (ii) a global conflict clause database. Any certificate discovered by one worker is immediately enforced via BCP in all others, a direct analogue of clause sharing in portfolio SAT solving. Terminal evaluations are also cached globally to avoid redundant

Table 3: **Comparison with Baselines across Theory Spaces.** Valid (%) denotes structurally valid models, Neutrino (%) denotes discovered neutrino flavor models, and Min Params denotes the minimum number of free parameters found (lower is better). DNN: deep neural network (the policy-value network, Section 4.3); PPO: Proximal Policy Optimization, the RL algorithm AMBer uses in place of CDRL's MCTS. Parenthetical multipliers under CDRL's Valid (%) and Neutrino (%) entries show improvement relative to AMBer, the prior state of the art; all other baselines (Random, ChatGPT, and the DNN/MCTS/symbolic ablations below AMBer) are shown for context and are not the basis for these multipliers. Each ablated or hybrid baseline (MCTS only, DNN only, Symbolic only, AlphaZero) is CDRL itself with the corresponding component removed, evaluated on the same physics pipeline described in Appendix B.3; this isolates that component's contribution rather than testing an independent reimplementation.

Method	Category	$A_4 \times \mathbb{Z}_4$			$A_4 \times \mathbb{Z}_N$			$T_{19} \times \mathbb{Z}_4$		
		Valid (%)	Neutrino (%)	Min Params	Valid (%)	Neutrino (%)	Min Params	Valid (%)	Neutrino (%)	Min Params
Uninformed and LLM baselines										
Random	Random search	0.1	0.00	12	0.1	0.00	14	0.1	0.00	16
ChatGPT (GPT-5.3) [68]	Few-shot prompting	0.2	0.00	13	0.2	0.00	18	0.1	0.00	15
Search and learning hybrids										
AlphaZero [18]	DNN + MCTS	14.9	0.01	7	16.2	0.03	7	5.4	0.04	5
MCTS only	Tree search only	10.2	0.00	9	9.8	0.00	10	1.1	0.00	7
DNN only	Policy/value network	5.1	0.00	13	4.5	0.00	9	2.2	0.01	7
Symbolic only [59]	CDCL SAT solver	0.2	0.00	12	0.2	0.00	12	0.1	0.00	10
State-of-the-art										
AMBer [1]	DNN + PPO	19.53	0.02	**5**	13.79	0.03	**5**	11.02	0.03	**4**
CDRL (Ours)	**DNN + MCTS + certificates**	**28.1 ± 3.3** (1.44×)	**0.10 ± 0.01** (5.00×)	**5**	**26.9 ± 3.7** (1.95×)	**0.19 ± 0.05** (6.33×)	**5**	**18.2 ± 2.1** (1.65×)	**0.09 ± 0.02** (3.00×)	**4**

physics computation. Removing this sharing increases variance and reduces neutrino discovery by up to $2\times$ (Table 5). Additional details are in Appendices B.5 and B.6.

4.8 Benchmark Theory Spaces

We evaluate CDRL on neutrino flavor model discovery using the same theory spaces as AMBer [1] to ensure direct comparability. The search space consists of discrete flavor symmetry models defined by non-Abelian representations, Abelian $\mathbb{Z}_N$ charge assignments, and flavon VEV configurations. We consider three theory spaces used in prior work, defined by their symmetry structure and the number of flavons n_ϕ: (i) $A_4 \times \mathbb{Z}_4$ with $n_\phi \leq 5$ flavons, (ii) $A_4 \times \mathbb{Z}_N$ where $N \in [2, 10]$ and $n_\phi \leq 5$, and (iii) $T_{19} \times \mathbb{Z}_4$ with $n_\phi \leq 6$. These spaces increase in symmetry richness and combinatorial complexity, providing progressively harder benchmarks.

5 Experimental Results

5.1 Evaluation Metrics

We adopt the same evaluation protocol as AMBer to ensure consistency. Each candidate model is evaluated using the physics pipeline, which produces a χ^2 fit to experimental observables and the number of free parameters n_p. A model is considered valid if it satisfies structural constraints, and is considered a neutrino model if it achieves both a good fit and low complexity, defined as $\chi^2 \leq 10$ and $n_p \leq 7$. We report the percentage of valid physics models (Valid %), the percentage of viable neutrino models (Neutrino %), and the minimum number of parameters discovered.

5.2 Main Results

CDRL achieves the highest Valid (%) and Neutrino (%) of any method in every theory space, matching on AMBer's Min Params. As shown in Table 3, on $A_4 \times \mathbb{Z}_N$, CDRL achieves 26.9% valid models compared to 13.79% for AMBer (a $1.95\times$ improvement), and 0.19% neutrino models compared to 0.03% (a $6.33\times$ improvement). Similar gains are observed in other theory spaces, including up to $1.65\times$ improvements in valid model rates and up to $3\times$ improvements in neutrino discovery. These gains are achieved with substantially higher sample efficiency. As shown in Table 4, CDRL discovers more neutrino models while evaluating up to $4\times$ fewer candidates. In combinatorial spaces exceeding

Table 4: **Search Efficiency Comparison.** CDRL discovers more neutrino models while evaluating significantly fewer candidates (up to $4\times$ fewer) across all theory spaces, demonstrating substantially improved sample efficiency over AMBer and random search.

Space	Method	Models evaluated	Neutrino models found
$A_4 \times \mathbb{Z}_4$	Random	4,000,000	28
	AMBer	4,000,000	683
	CDRL	1,000,000	**1090**
$A_4 \times \mathbb{Z}_N$	Random	4,000,000	21
	AMBer	4,000,000	1394
	CDRL	1,000,000	**2343**
$T_{19} \times \mathbb{Z}_4$	Random	24,000,000	409
	AMBer	24,000,000	6439
	CDRL	6,000,000	**7019**

10^{26} possible models, these reductions correspond to substantially more efficient exploration. The mechanism behind these gains is concrete: Figure 1 traces how a single certificate blocks an entire class of candidates before any physics is run, and Figure 5 shows an example rule recovered from the search itself (full analysis in Section 5.4).

Importance of certificate-driven pruning. The additional baselines (implementation details in Appendix B.3) are CDRL itself with one component removed, and highlight the importance of combining learning, planning, and symbolic reasoning. Few-shot prompting with GPT-5.3 performs near-random search, indicating that current large language models offer little advantage in these underexplored theory spaces: the relevant structure is largely absent from pretraining data, so the model cannot substitute for guided search with symbolic feedback. Pure symbolic search, DNN-only policies, MCTS-only search, and AlphaZero-style baselines all substantially underperform CDRL. On $A_4 \times \mathbb{Z}_N$ the AlphaZero-style baseline (MCTS and the policy-value network, without certificates) already exceeds AMBer's valid model rate (16.2% vs. 13.79%), so planning and neural guidance alone recover part of AMBer's gap to CDRL. Certificate-driven pruning (Table 5) accounts for the remainder and for most of the gain in neutrino discovery. Future work may study the additional performance metrics for such rule-guided search, beyond raw number of neutrino models found.

Qualitative trends. Aggregate rates measure how often a search succeeds, but for model building the *diversity* of what is found matters just as much: physicists want a broad set of distinct candidate theories to study, not many copies of the same one, and coverage across different symmetry groups indicates that the search is not confined to a narrow corner of the space. We therefore also examine the distribution of discovered models, not only their counts. Figure 3 shows the clearest evidence of this: CDRL discovers viable models spread across a substantially broader range of $\mathbb{Z}_N$ symmetry groups than AMBer, which concentrates around a few specific values of N. This indicates that CDRL explores different regions of the theory space rather than only different instances within the same region. Models are binned by the $\mathbb{Z}_N$ they were constructed under rather than any reduced or canonical form, so this spread is not an artifact of the same underlying model being counted under multiple symmetry groups; the reducibility certificate (Table 2) instead prevents a redundant representative from being explored and counted a second time under a different N. Figure 4 shows where CDRL's models concentrate in (χ^2, n_p) space in the right column compared to AMBer in the left column. We also present a strong example model discovered by CDRL in the $T_{19} \times \mathbb{Z}_4$ space (Table 1). The neutrino models discovered by CDRL do not exactly match any of those found by AMBer. Combined with the broader $\mathbb{Z}_N$ coverage in Figure 3, this is consistent with CDRL exploring complementary regions of the theory space rather than merely finding different individual models within the same region AMBer already covers.

Following the analysis protocol of AMBer [1], we can look more closely at why these distributions differ. For the distribution of models shown in Figure 4, CDRL reaches neutrino models with the smallest parameter counts observed for either method (Table 3 reports this as Min Params) while maintaining competitive or lower χ^2 values. For the $\mathbb{Z}_N$ distribution of Figure 3, AMBer concentrates its discoveries around specific symmetry groups (e.g., $N = 5$), whereas CDRL spreads

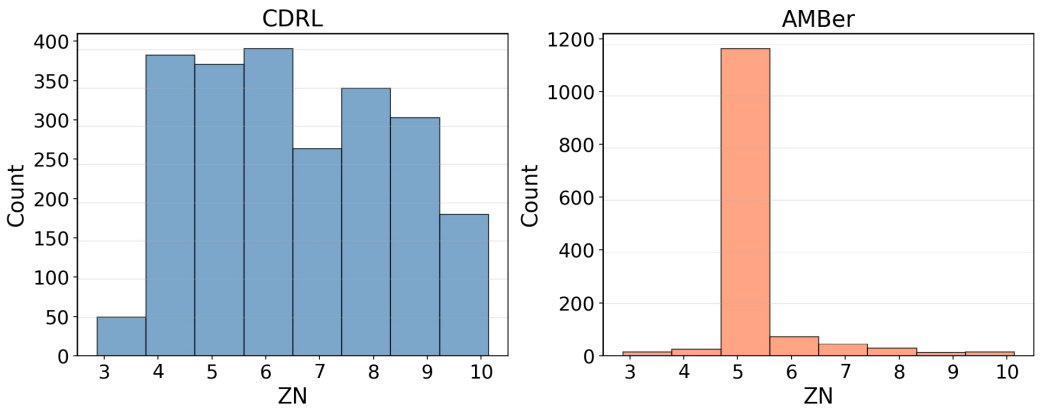


Figure 3: Distribution of successfully discovered neutrino models across $\mathbb{Z}_N$ symmetry groups for AMBer and CDRL. CDRL discovers viable models more uniformly across different symmetry choices, while AMBer concentrates more heavily around specific values of N.

more uniformly across the available $\mathbb{Z}_N$ choices. This trend is consistent with progressively stronger symbolic pruning during search. As the learned clause database grows, the legal-action mask becomes increasingly selective, allowing Boolean constraint propagation to eliminate larger regions of invalid or redundant search trajectories before expensive physics evaluation is performed. This mechanism both concentrates evaluation on predictive low-parameter models and, because whole redundant equivalence classes are removed rather than merely down-weighted, pushes the search into $\mathbb{Z}_N$ regions AMBer rarely reaches. CDRL's reward contains no term that rewards higher-order $\mathbb{Z}_N$ symmetries (Section 4.4 gives the full reward, which depends only on χ^2 fit quality and the number of free parameters); AMBer's scalar reward includes such a term to steer search toward the more complex, higher-order groups [1]. CDRL reaches broader coverage without an equivalent incentive. Certificates prune redundant equivalence classes uniformly across the theory space, so exploration spreads more evenly as a byproduct of pruning rather than of reward shaping. This difference in reward design does not affect the evaluation protocol itself (Section 5), which scores both methods identically.

Although CDRL introduces slight additional overhead from MCTS, certificate analysis, and symbolic propagation, the dominant computational cost (approx. 98%) arises from the physics evaluation pipeline, which is the same as AMBer. Since experiments use different parallelization settings and compute configurations (CDRL uses 32 CPU cores, while AMBer uses 250 parallel environments), we report candidate evaluations as the primary normalized measure of search efficiency.

5.3 Ablation Study

An ablation study measures each component's importance by removing it from the full system and observing the resulting drop in performance, so a larger drop indicates a more important component. Table 5 isolates the contribution of each component this way. Error bars are the standard deviation across 4 runs with different random seeds. Every ablation degrades performance, confirming that the gains come from integrating learning, planning, and symbolic reasoning rather than from any single part. Removing either the neural network or MCTS collapses neutrino discovery to near zero, showing that guided exploration is necessary at all in a space this large. Removing symbolic certificates instead leaves a functioning search, with valid model rates falling from 26.9% to 16.2% and neutrino discovery from 0.19% to 0.03% on $A_4 \times \mathbb{Z}_N$, so certificates contribute a substantial improvement on top of the neural and planning components. Removing portfolio sharing lowers performance and raises variance, indicating that shared constraints and experience are critical for scaling. In short, each component contributes measurably, and the full system is required for strong performance.

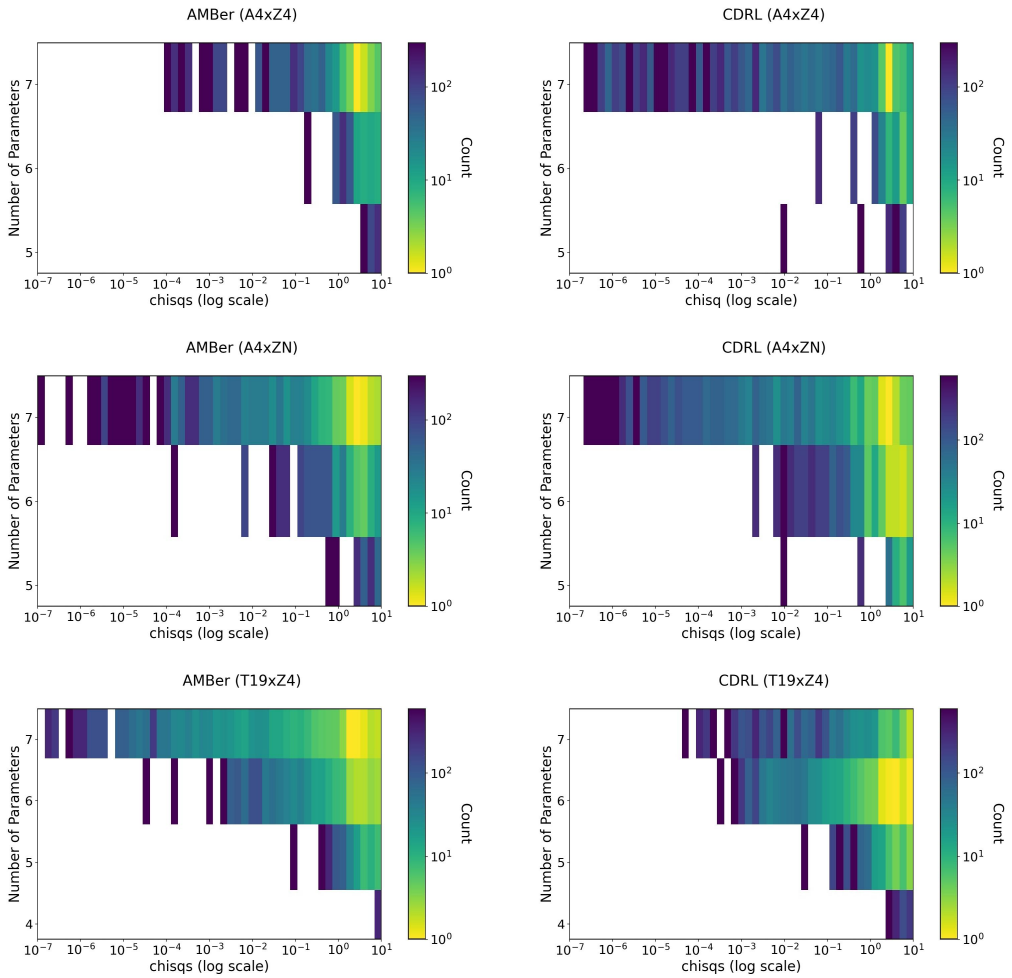


Figure 4: Distribution of discovered models over χ^2 (fit quality) and n_p (number of parameters). Each bin shows model density, with the bottom-left region corresponding to well-fitting, low-complexity models. CDRL places more mass in the low-n_p areas. Within that region, χ^2 differences at very low values do not indicate meaningfully different fit quality, so Table 3 and Table 5 remain the primary quantitative comparison, with this figure illustrating where in (χ^2, n_p) space that advantage is concentrated.

Table 5: **Ablation Study of CDRL Components.** Valid (%) denotes the percentage of structurally valid models, Neutrino (%) denotes the percentage of discovered neutrino flavor models, and Min Params denotes the minimum number of free parameters found.

Method / Ablation	Description	$A_4 \times \mathbb{Z}_4$			$A_4 \times \mathbb{Z}_N$			$T_{19} \times \mathbb{Z}_4$		
		Valid (%)	Neutrino (%)	Min Params	Valid (%)	Neutrino (%)	Min Params	Valid (%)	Neutrino (%)	Min Params
CDRL (Full)	DNN + MCTS + symbolic certificates + portfolio	**28.1 ± 3.3**	**0.10 ± 0.01**	**5**	**26.9 ± 3.7**	**0.19 ± 0.05**	**5**	**18.2 ± 2.1**	**0.09 ± 0.02**	**4**
No DNN	No policy/value network	10.2 ± 5.1	0.00 ± 0.00	9	9.8 ± 4.6	0.00 ± 0.00	10	1.1 ± 1.0	0.00 ± 0.00	7
No MCTS	No planning	5.1 ± 8.4	0.00 ± 0.00	13	4.5 ± 9.0	0.00 ± 0.00	9	2.2 ± 3.6	0.01 ± 0.00	7
No symbolic	No certificate	14.9 ± 3.1	0.01 ± 0.02	7	16.2 ± 2.1	0.03 ± 0.01	7	5.4 ± 4.9	0.04 ± 0.02	5
No portfolio	No clause/replay shared	26.9 ± 6.5	0.04 ± 0.02	6	25.1 ± 7.2	0.10 ± 0.09	5	12.1 ± 7.8	0.06 ± 0.04	5

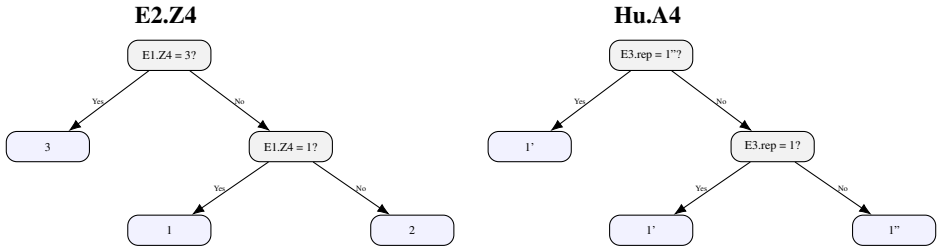


Figure 5: **Example decision trees.** *Left:* Determination of the E_2's $\mathbb{Z}_4$ charge in $A_4 \times \mathbb{Z}_4$. Among high-confidence states, this decision tree predicts the E_2's $\mathbb{Z}_4$ charge from E_1's $\mathbb{Z}_4$ charge. This demonstrates how CDRL has learned to match the $\mathbb{Z}_4$ charges of the charged lepton singlets, a reasonable result in order to construct physically realizable mass matrices. This observation also shows how some rules exclude possible choices (in this case the $\mathbb{Z}_4$ charge value of 4). *Right:* Determination of the H_u's A_4 representation in $A_4 \times \mathbb{Z}_N$. This decision tree is harder to interpret physically. The up-flavor Higgs is not directly correlated with the charged lepton sector, so there is either some non-trivial relationship, or this is a spurious correlation within the CDRL search. The full set of such observations for all three theory spaces is in Appendix C.

5.4 Search Knowledge Discovery

A by-product of the search is a record of the decisions CDRL makes on its way to good models. We examine this record post-hoc to recover common decision patterns from the MCTS trajectories and cast them into human-readable decision trees. These decision patterns both illuminate how the agent is exploring, and with sufficient interpretation by a human physicist can provide insight on common modes of viable models in the theory space. Our approach is related to prior work on extracting interpretable policies from RL systems [69]. However, in addition to using extracted rules purely for explanation or verification, we inject these rules as soft symbolic constraints in fresh, smaller-scale, "rule-guided" CDRL runs to test their utility in deepening exploration of fruitful regions of the theory space. Effectively, these soft constraints encourage the agent towards known viable regions of the theory space while still permitting exploration of the entire space. As shown in Figure 6, incorporating these rules consistently increases the number of discovered models in all three theory spaces. In $A_4 \times \mathbb{Z}_4$ and $T_{19} \times \mathbb{Z}_4$, rule-guided runs achieve $\sim 2\times$ higher valid model rates and 2–3$\times$ higher neutrino model discovery, while in $A_4 \times \mathbb{Z}_N$ we observe $\sim 1.5\times$ improvements in validity and up to $3\times$ in neutrino discovery.

We describe the procedure used to extract soft constraints from MCTS trajectories in Algorithm 2 and incorporate them into the search process. We retain high-confidence decision points from MCTS trajectories, *i.e.* states where the dominant action receives more than 95% of visits and has a positive value estimate, and group these states by construction progress, defined as the number of assigned entries in the state before the first unassigned position. For each progress bucket with enough samples, we train one shallow decision tree that predicts the dominant action from the partial model's earlier assignments; hyperparameters are given in Appendix B. Because blocks are assigned in a fixed top-down order (Section 4), only blocks assigned earlier in that order are available as features. We extract these high-confidence root-to-leaf paths as rules of the form conditions $\rightarrow a$, where the conditions are conjunctions over binary state features, and retain only rules expressible as binary conditions for compatibility with the symbolic representation. These rules can then be injected into a new CDRL run as soft constraints. At inference time, we identify all rules whose conditions are satisfied by the current state, let each activated rule vote for its preferred action to form a bias vector $r(s)$, and reweight the neural policy as $\tilde{P}(a \mid s) \propto P(a \mid s)\,(1 + \beta\, r_a(s))$, where $\beta = 0.5$ in all experiments. The reweighted policy is then masked by the legality constraints and renormalized before being used in MCTS, so soft constraints guide the search without overriding hard constraints or eliminating exploration.

The extracted rules are simple if-else conditions over particle assignments that reveal recurring structure in successful models. For example, assignments of charged lepton singlets often follow consistent patterns (e.g., E_1 influencing E_2, E_3 through identical or systematically shifted $\mathbb{Z}_4$ charges). We also observe coordinated triplet assignments across L, N, and ϕ, mirroring common design

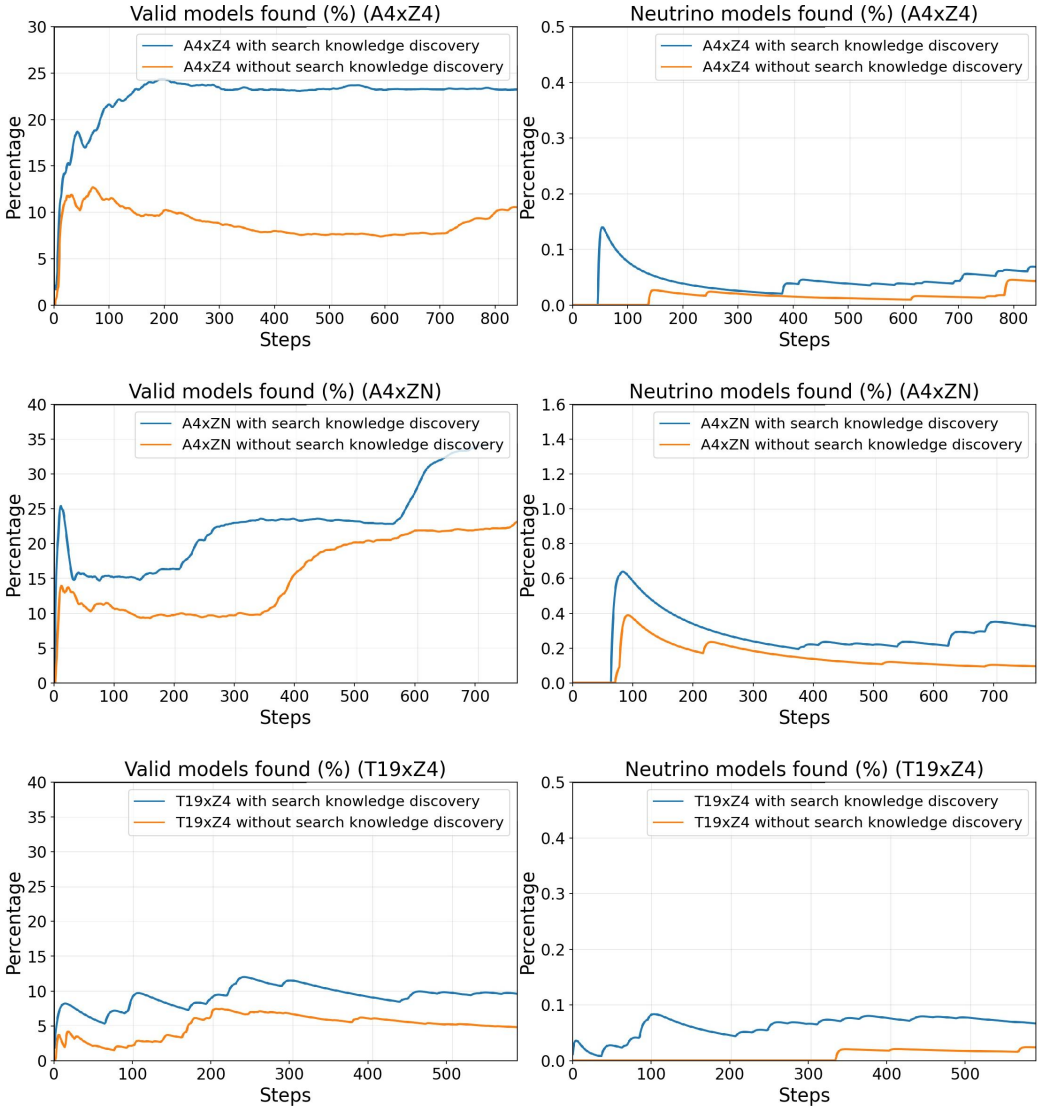


Figure 6: Impact of search knowledge discovery (KD) across theory spaces. Each panel is titled by metric and theory space. We compare CDRL with and without rule-based soft constraints on $A_4 \times \mathbb{Z}_4$, $A_4 \times \mathbb{Z}_N$, and $T_{19} \times \mathbb{Z}_4$. Incorporating rules extracted from high-confidence search trajectories consistently improves both valid model rates and neutrino model discovery. Rule-guided runs achieve higher final performance and faster convergence, with gains of roughly 1.5–$2\times$ in valid models and 2–$3\times$ in neutrino models across the three spaces, demonstrating that the extracted rules provide useful and generalizable guidance for exploration. These experiments are performed using smaller-scale KD evaluation runs, so absolute values are not directly comparable to the full-scale results reported in Table 3.

Algorithm 2 Soft Constraint Extraction and Policy Reweighting

Require: MCTS traces $\mathcal{D}$, progress buckets $\mathcal{B}$
Require: Minimum samples m, tree hyperparameters (d, s, ℓ)
Require: Soft bias strength β
Collect state-action samples (x_i, a_i) from MCTS
Retain high-confidence samples based on visit concentration and value estimates
Group samples into buckets $\{\mathcal{D}_b\}_{b \in \mathcal{B}}$ based on progress
for each bucket $b \in \mathcal{B}$ **do**
    if $|\mathcal{D}_b| < m$ **then**
        continue
    end if
    Train decision tree T_b with depth d, split s, leaf size ℓ
    Extract leaf paths corresponding to high-confidence predictions
    Convert each path into a rule (conditions $\rightarrow a$)
    Store rules with support and purity
end for
Build rule table indexed by progress and decision block

Policy Reweighting during MCTS:
Given state s, compute neural policy $P(\cdot \mid s)$
Identify rules satisfied by s and compute bias vector $r(s)$
Reweight policy:

$$\tilde{P}(a \mid s) \propto P(a \mid s)\,(1 + \beta\, r_a(s))$$

Apply legality mask and renormalize $\tilde{P}$

patterns used by human model builders [70]. Although extracted post-hoc, these rules provide reusable and generalizable structure. The full set of 40 rules is listed in Section C.

Crucially, interpreting each rule within the model building process requires additional context not always captured within the decision tree, and many rules are more representative of the agent's search process than of correlations between parameters of the model. For example, several of the observed rules for the $\mathbb{Z}_N$ charges of the charged lepton singlets (E_1, E_2, E_3) show them favoring identical or closely related charges. This is a reasonable pattern for correctly producing their masses if other particles have already been given appropriate representation assignments. However, some rules restrict themselves to only a subset of the charge values the symmetry allows. This demonstrates the limitation of interpreting these rules physically. Each tree is fit only to the high-confidence trajectories within a single progress bucket, a small, selective sample. Therefore, a tree can look like a high-confidence rule simply because the values outside it were rare in that particular sample, not because the wider search avoids them. While we introduce a tool for search knowledge discovery, further work in this direction may enable more robust interpretation of these discovered correlations in terms of useful model-building rules. Surfacing this kind of observation is still useful. It either flags a rule that will not hold up once tested on more data, or, when a pattern holds up under closer inspection, a regularity of the theory space a human model builder had not already anticipated.

6 Conclusion, Limitation, and Future Work

We introduced CDRL, a framework for scientific discovery in large constrained hypothesis spaces. Unlike standard RL methods that rely primarily on scalar rewards, CDRL uses structured certificates from symbolic domain-specific reasoning tools to identify why candidates fail and convert these failures into reusable constraints. On neutrino flavor model discovery, CDRL consistently improves search efficiency across all theory spaces. Per candidate evaluated, it achieves up to 1.95× higher valid model rates and 6.33× higher neutrino model discovery than AMBer. Separately, reaching a comparable number of neutrino models requires up to 4× fewer physics evaluations (Table 4). CDRL also improves coverage of the theory space, matches AMBer's best model complexity while discovering far more of them, and enables post-hoc search knowledge discovery through interpretable rules extracted from search trajectories.

The core idea of CDRL is not specific to neutrino physics, nor to particle physics. Many scientific domains pair a large discrete design space with an expensive verifier and an external tool that can explain failures, including chemistry, materials science, and automated theorem proving. In each, the dominant cost is the verification step, so the same computational bottleneck we address here applies: a scalar reward wastes that expensive budget by letting the agent revisit equivalent failures, whereas a certificate removes the whole failing class once and for all. Wherever a reasoning tool can name the assignments responsible for a violation, CDRL can turn that explanation into a constraint that is reused across the search. Finally, the rules recovered through search knowledge discovery are useful beyond efficiency, as they compress what the agent learned into human-readable structure, which both accelerates future runs and gives scientists a first look at the regularities of a mathematical space they are only beginning to explore. We have not yet independently validated these rules as physically meaningful. At this stage, they describe regularities in how the search itself explores the space, which are themselves shaped by our design choices, and confirming which of them reflect genuine physical structure is left to future work with domain experts.

The main limitation of CDRL is that it depends on the availability of useful certificate-generating tools and is currently best suited to structured, discrete search spaces where constraints can be represented and reused effectively. The framework could also introduce overhead in larger search spaces from certificate analysis and clause management. Future work will extend CDRL to broader scientific domains such as chemistry and materials science, where external verifiers can provide rich feedback. Another promising direction is to learn when and how to generalize certificates into reusable constraints, and to combine CDRL with foundation models so that neural priors, symbolic pruning, and scientific verification can work together in a unified discovery loop.

Acknowledgments

PJ was supported by a seed grant from the AI4Science Center in the College of Sciences at Georgia Tech. MF is supported by Fermilab which is administered by Fermi Forward Discovery Group, LLC under Contract No. 89243024CSC000002 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. JR is supported by a fellowship from the WATCHEP training grant from the U.S. Department of Energy, Office of Science. The work of VKP was supported in part by the U.S. National Science Foundation under Grant PHY-221028.

References

[1] J. B. Baretz, M. Fieg, V. Ganesh, A. Ghosh, V. Knapp-Perez, J. Rudolph, and D. Whiteson, "Towards ai-assisted neutrino flavor theory design," *arXiv preprint arXiv:2506.08080*, 2025.

[2] F. Feruglio and A. Romanino, "Lepton flavour symmetries," 2021.

[3] Y. Almumin, M.-C. Chen, M. Cheng, V. Knapp-Perez, Y. Li, A. Mondol, S. Ramos-Sanchez, M. Ratz, and S. Shukla, "Neutrino Flavor Model Building and the Origins of Flavor and CP Violation," *Universe*, vol. 9, no. 12, p. 512, 2023.

[4] A. de Gouvêa, I. Mocioiu, S. Pastore, L. E. Strigari, L. Alvarez-Ruso, A. M. Ankowski, A. B. Balantekin, V. Brdar, M. Cadeddu, S. Carey, J. Carlson, M. C. Chen, V. Cirigliano, W. Dekens, P. B. Denton, R. Dharmapalan, L. Everett, H. Gallagher, S. Gardiner, J. Gehrlein, L. Graf, W. C. Haxton, O. Hen, H. Hergert, S. Horiuchi, P. Q. Hung, J. Isaacson, N. Jachowicz, L. Jin, A. N. Khan, A. Lovato, P. A. N. Machado, K. Mahn, D. Marfatia, C. Mariani, E. Mereghetti, J. G. Morfín, A. Nicholson, G. Paz, R. Plestid, N. Rocco, I. Sarcevic, R. Schiavilla, A. Sousa, J. Tena-Vidal, M. L. Wagman, and A. Walker-Loud, "Theory of neutrino physics – snowmass tf11 (aka nf08) topical group report," 2022.

[5] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, "Highly accurate protein structure prediction with AlphaFold," *nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[6] M. H. Segler, M. Preuss, and M. P. Waller, "Planning chemical syntheses with deep neural networks and symbolic ai," *Nature*, vol. 555, no. 7698, pp. 604–610, 2018.

[7] X. Bai and X. Zhang, "Artificial intelligence-powered materials science," *Nano-micro letters*, vol. 17, no. 1, p. 135, 2025.

[8] A. Merchant, S. Batzner, S. S. Schoenholz, M. Aykol, G. Cheon, and E. D. Cubuk, "Scaling deep learning for materials discovery," *Nature*, vol. 624, no. 7990, pp. 80–85, 2023.

[9] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, *et al.*, "Accurate structure prediction of biomolecular interactions with AlphaFold 3," *Nature*, vol. 630, no. 8016, pp. 493–500, 2024.

[10] D. J. Mankowitz, A. Michi, A. Zhernov, M. Gelmi, M. Selvi, C. Paduraru, E. Leurent, S. Iqbal, J.-B. Lespiau, A. Ahern, *et al.*, "Faster sorting algorithms discovered using deep reinforcement learning," *Nature*, vol. 618, no. 7964, pp. 257–263, 2023.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[12] Z. Li, J. Sun, L. Murphy, Q. Su, Z. Li, X. Zhang, K. Yang, and X. Si, "A survey on deep learning for theorem proving," *arXiv preprint arXiv:2404.09939*, 2024.

[13] J. Zhang and B. Gu, "Advances in reinforcement learning for intelligent program synthesis," in *2025 5th International Conference on Neural Networks, Information and Communication Engineering (NNICE)*, pp. 1734–1739, IEEE, 2025.

[14] H. Wang, T. Fu, Y. Du, W. Gao, K. Huang, Z. Liu, P. Chandak, S. Liu, P. Van Katwyk, A. Deac, *et al.*, "Scientific discovery in the age of artificial intelligence," *Nature*, vol. 620, no. 7972, pp. 47–60, 2023.

[15] J. D. Martín-Guerrero and L. Lamata, "Reinforcement learning and physics," *Applied Sciences*, vol. 11, no. 18, p. 8589, 2021.

[16] S. Gow, M. Niranjan, S. Kanza, and J. G. Frey, "A review of reinforcement learning in chemistry," *Digital Discovery*, vol. 1, no. 5, pp. 551–567, 2022.

[17] P. Jha, P. Jana, P. Suresh, A. Arora, and V. Ganesh, "RLSF: Fine-tuning LLMs via symbolic feedback," *28th European Conference on Artificial Intelligence*, 2025.

[18] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[19] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, pp. 354–359, Oct 2017.

[20] G. N. Wojcik, S. T. Eu, and L. L. Everett, "Graph reinforcement learning for exploring model spaces beyond the standard model," *Phys. Rev. D*, vol. 111, no. 3, p. 035007, 2025.

[21] S. Alexander, B. Bradley, L. Gouskos, and C. Niu, "Autonomous Discovery of Particle Physics Theories from Experimental Data," 3 2026.

[22] A. Giarnetti and D. Meloni, "Reinforcement Learning Techniques for the Flavor Problem in Particle Physics," *Symmetry*, vol. 18, no. 1, p. 131, 2026.

[23] A. Biere, H. van Maaren, and T. Walsh, *Handbook of satisfiability*. SAGE Publications Limited, 2009.

[24] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem-proving," *Communications of the ACM*, vol. 5, no. 7, pp. 394–397, 1962.

[25] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R. Ruiz, J. Schrittwieser, G. Swirszcz, *et al.*, "Discovering faster matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, no. 7930, pp. 47–53, 2022.

[26] F. J. Ruiz, T. Laakkonen, J. Bausch, M. Balog, M. Barekatain, F. J. Heras, A. Novikov, N. Fitzpatrick, B. Romera-Paredes, J. Van De Wetering, *et al.*, "Quantum circuit optimization with AlphaTensor," *Nature Machine Intelligence*, vol. 7, no. 3, pp. 374–385, 2025.

[27] Y. Fukuda *et al.*, "Evidence for oscillation of atmospheric neutrinos," *Phys. Rev. Lett.*, vol. 81, pp. 1562–1567, 1998.

[28] Q. R. Ahmad *et al.*, "Direct evidence for neutrino flavor transformation from neutral current interactions in the Sudbury Neutrino Observatory," *Phys. Rev. Lett.*, vol. 89, p. 011301, 2002.

[29] B. T. Cleveland, T. Daily, R. Davis, Jr., J. R. Distel, K. Lande, C. K. Lee, P. S. Wildenhain, and J. Ullman, "Measurement of the solar electron neutrino flux with the Homestake chlorine detector," *Astrophys. J.*, vol. 496, pp. 505–526, 1998.

[30] M. H. Ahn *et al.*, "Indications of neutrino oscillation in a 250 km long baseline experiment," *Phys. Rev. Lett.*, vol. 90, p. 041801, 2003.

[31] K. S. Babu, "TASI Lectures on Flavor Physics," in *Theoretical Advanced Study Institute in Elementary Particle Physics: The Dawn of the LHC Era*, pp. 49–123, 2010.

[32] W. Altmannshofer and A. Greljo, "Recent Progress in Flavor Model Building," *Ann. Rev. Nucl. Part. Sci.*, vol. 75, no. 1, pp. 201–322, 2025.

[33] S. F. King, "Models of Neutrino Mass, Mixing and CP Violation," *J. Phys. G*, vol. 42, p. 123001, 2015.

[34] P. Minkowski, "$\mu \to e\gamma$ at a Rate of One Out of 10^9 Muon Decays?," *Phys. Lett. B*, vol. 67, pp. 421–428, 1977.

[35] T. Yanagida, "Horizontal gauge symmetry and masses of neutrinos," *Conf. Proc. C*, vol. 7902131, pp. 95–99, 1979.

[36] S. L. Glashow, "The Future of Elementary Particle Physics," *NATO Sci. Ser. B*, vol. 61, p. 687, 1980.

[37] M. Gell-Mann, P. Ramond, and R. Slansky, "Complex Spinors and Unified Theories," *Conf. Proc. C*, vol. 790927, pp. 315–321, 1979.

[38] J. Halverson, B. Nelson, and F. Ruehle, "Branes with brains: exploring string vacua with deep reinforcement learning," *Journal of High Energy Physics*, vol. 2019, June 2019.

[39] T. R. Harvey and A. Lukas, "Quark Mass Models and Reinforcement Learning," *JHEP*, vol. 08, p. 161, 2021.

[40] S. Nishimura, C. Miyao, and H. Otsuka, "Exploring the flavor structure of quarks and leptons with reinforcement learning," *JHEP*, vol. 23, p. 021, 2020.

[41] S. Nishimura, C. Miyao, and H. Otsuka, "Reinforcement learning-based statistical search strategy for an axion model from flavor," , 9 2024.

[42] F. Carta, A. Gauntlett, F. Griffin, and Y.-H. He, "BPS spectroscopy with reinforcement learning," , 1 2025.

[43] Y. Xu, X. Liu, X. Cao, C. Huang, E. Liu, S. Qian, X. Liu, Y. Wu, F. Dong, C.-W. Qiu, *et al.*, "Artificial intelligence: A powerful paradigm for scientific research," *The innovation*, vol. 2, no. 4, 2021.

[44] Y. Liu, Z. Yang, Z. Yu, Z. Liu, D. Liu, H. Lin, M. Li, S. Ma, M. Avdeev, and S. Shi, "Generative artificial intelligence and its applications in materials science: Current situation and future perspectives," *Journal of Materiomics*, vol. 9, no. 4, pp. 798–816, 2023.

[45] G. Karagiorgi, G. Kasieczka, S. Kravitz, B. Nachman, and D. Shih, "Machine learning in the search for new fundamental physics," *Nature Reviews Physics*, vol. 4, no. 6, pp. 399–412, 2022.

[46] L. Messeri and M. J. Crockett, "Artificial intelligence and illusions of understanding in scientific research," *Nature*, vol. 627, no. 8002, pp. 49–58, 2024.

[47] R. Van Noorden and J. M. Perkel, "Ai and science: what 1,600 researchers think," *Nature*, vol. 621, no. 7980, pp. 672–675, 2023.

[48] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, *et al.*, "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, no. 7897, pp. 414–419, 2022.

[49] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, 2020.

[50] A. A. Melnikov, H. Poulsen Nautrup, M. Krenn, V. Dunjko, M. Tiersch, A. Zeilinger, and H. J. Briegel, "Active learning machine learns to create new quantum experiments," *Proceedings of the National Academy of Sciences*, vol. 115, no. 6, pp. 1221–1226, 2018.

[51] P. Hitzler, M. K. Sarker, and A. Eberhart, *Compendium of Neurosymbolic Artificial Intelligence*, vol. 369. IOS Press, 2023.

[52] K. Acharya, W. Raza, C. Dourado, A. Velasquez, and H. H. Song, "Neurosymbolic reinforcement learning and planning: A survey," *IEEE Transactions on Artificial Intelligence*, 2023.

[53] B. C. Colelough and W. Regli, "Neuro-symbolic ai in 2024: A systematic review," *arXiv preprint arXiv:2501.05435*, 2025.

[54] S. Kambhampati, K. Valmeekam, L. Guan, K. Stechly, M. Verma, S. Bhambri, L. Saldyt, and A. Murthy, "LLMs can't plan, but can help planning in llm-modulo frameworks," *arXiv preprint arXiv:2402.01817*, 2024.

[55] L. A. Agrawal, A. Kanade, N. Goyal, S. Lahiri, and S. Rajamani, "Monitor-guided decoding of code LMs with static analysis of repository context," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[56] Y. Chen, C. Wang, O. Bastani, I. Dillig, and Y. Feng, "Program synthesis using deduction-guided reinforcement learning," in *International Conference on Computer Aided Verification*, pp. 587–610, Springer, 2020.

[57] P. Jana, P. Jha, H. Ju, G. Kishore, A. Mahajan, and V. Ganesh, "Cotran: An llm-based code translator using reinforcement learning with feedback from compiler and symbolic execution," *arXiv preprint arXiv:2306.06755*, 2023.

[58] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, *et al.*, "Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[59] J. H. Liang, V. Ganesh, P. Poupart, and K. Czarnecki, "Learning rate based branching heuristic for sat solvers," in *Theory and Applications of Satisfiability Testing–SAT 2016: 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings 19*, pp. 123–140, Springer, 2016.

[60] S. Jha, S. Gulwani, S. A. Seshia, and A. Tiwari, "Oracle-guided component-based program synthesis," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pp. 215–224, 2010.

[61] E. Polgreen, A. Reynolds, and S. A. Seshia, "Satisfiability and synthesis modulo oracles," in *International Conference on Verification, Model Checking, and Abstract Interpretation*, pp. 263–284, Springer, 2022.

[62] S. Jha and S. A. Seshia, "A theory of formal synthesis via inductive learning," *Acta Informatica*, vol. 54, no. 7, pp. 693–726, 2017.

[63] I. Esteban, M. C. Gonzalez-Garcia, M. Maltoni, T. Schwetz, and A. Zhou, "The fate of hints: updated global analysis of three-flavor neutrino oscillations," *JHEP*, vol. 09, p. 178, 2020.

[64] C. D. Rosin, "Multi-armed bandits with episode context," *Annals of Mathematics and Artificial Intelligence*, vol. 61, no. 3, pp. 203–230, 2011.

[65] T. Balyo, P. Sanders, and C. Sinz, "Hordesat: A massively parallel portfolio sat solver," in *International Conference on Theory and Applications of Satisfiability Testing*, pp. 156–172, Springer, 2015.

[66] T. Menouer and S. Baarir, "Parallel learning portfolio-based solvers," *Procedia Computer Science*, vol. 108, pp. 335–344, 2017.

[67] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay. corr abs/1803.00933 (2018)," *arXiv preprint arXiv:1803.00933*, 2018.

[68] A. Singh, A. Fry, A. Perelman, A. Tart, A. Ganesh, A. El-Kishky, A. McLaughlin, A. Low, A. Ostrow, A. Ananthram, *et al.*, "Openai gpt-5 system card," *arXiv preprint arXiv:2601.03267*, 2025.

[69] O. Bastani, Y. Pu, and A. Solar-Lezama, "Verifiable reinforcement learning via policy extraction," *Advances in neural information processing systems*, vol. 31, 2018.

[70] G.-J. Ding and D. Meloni, "A Model for Tri-bimaximal Mixing from a Completely Broken A_4," *Nucl. Phys. B*, vol. 855, pp. 21–45, 2012.

[71] A. Ignatiev, A. Morgado, and J. Marques-Silva, "PySAT: A Python toolkit for prototyping with SAT oracles," in *SAT*, pp. 428–437, 2018.

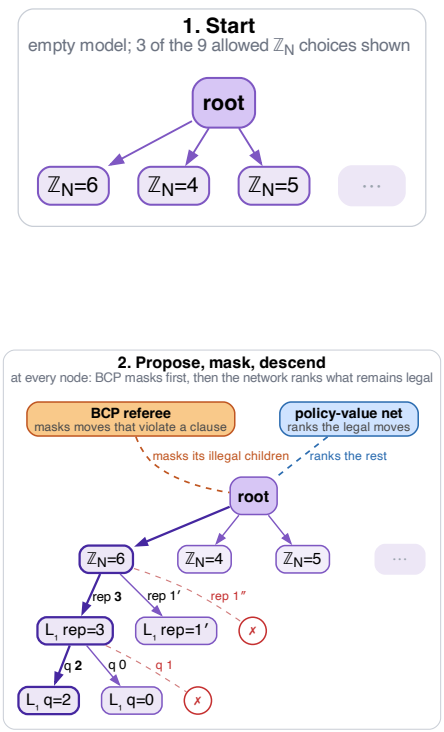
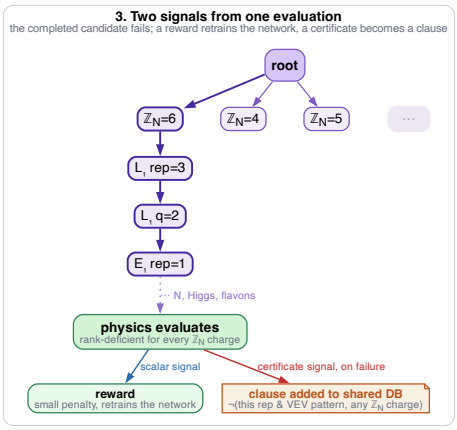


Figure 7: **The CDRL mechanism, shown as four snapshots of the same tree. (1) Start:** the empty model; three of the nine allowed $\mathbb{Z}_N$ choices are shown at the root. **(2) Propose, mask, descend:** at the root, and again at every node MCTS visits, the BCP referee masks the illegal children first and the policy-value network then ranks what remains; MCTS commits to $\mathbb{Z}_N = 6$ and descends, and the same red $\times$ masking recurs at each node along the way. Continued below.

A The CDRL Loop: A Step-by-Step Walkthrough

Figure 7 below walks through a worked example as four snapshots of the same tree, taken as it is actually built, covering the same components as Figure 1 (the policy-value network, the BCP referee, MCTS, physics evaluation, the certificate analyzer, and the clause database) except search knowledge discovery, which operates offline after training and is not part of this per-candidate loop. Showing the same tree across panels also makes the key claim concrete rather than only described: a clause learned from one branch permanently prunes a matching pattern anywhere else in the tree.

At every node, not only the root, the BCP referee first masks any child that would violate an already-learned clause, and the policy-value network then ranks the remaining legal children; this repeats at each subsequent decision (the $\mathbb{Z}_N$ choice, then each particle's representation, then its charge, and so on). What Figure 7 shows as a single selected path is itself the outcome of many such simulations from the current state, each one descending through already-expanded nodes by upper-confidence bound and expanding one new leaf; the move actually played is the one visited most often across these simulations, not simply the network's top-ranked action. As in Algorithm 1, the network is updated periodically from replay-buffer mini-batches rather than immediately after every single evaluation, and the certificate path in Step 3 only triggers when the completed candidate fails; a valid candidate produces a reward alone. The three $\mathbb{Z}_N$ choices at the root are an illustrative subset of the nine allowed in this theory space ($N \in \{2, \ldots, 10\}$).



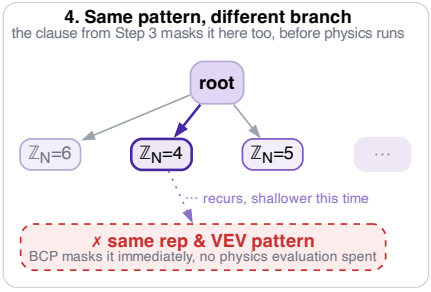


Figure 7: **(continued) (3) Two signals from one evaluation:** the branch completes and the candidate fails, rank-deficient for every $\mathbb{Z}_N$ charge; this single evaluation produces a scalar reward that retrains the network and, separately, a certificate that the analyzer converts into a clause added to the shared database. **(4) Same pattern, different branch:** MCTS next explores the sibling $\mathbb{Z}_N = 4$ branch, and the clause learned in step 3 masks the same representation-and-VEV pattern as soon as it recurs, before the branch is built out further or any physics evaluation is spent on it.

B Additional Experimental Details

B.1 Compute Infrastructure

All experiments are conducted on a machine with Intel Xeon Gold 6448Y CPUs (32 cores, 2.10 GHz), 32 GB RAM, and an NVIDIA H100 GPU (80GB, MIG 20GB partition). Parallel self-play is executed across CPU workers, while neural network training is performed on the GPU.

The dominant computational cost in our setting arises from the physics evaluation pipeline, which involves symbolic model construction, numerical optimization, and parameter fitting for candidate theories. MCTS traversal and SAT-based propagation introduce additional overhead, but are comparatively lightweight relative to terminal model evaluation. Since different methods use different parallelization settings and compute configurations (CDRL uses 32 CPU cores, while AMBer uses 250 parallel environments), we report candidate evaluations as the primary normalized efficiency metric.

B.2 Additional Implementation Details

Our framework is implemented in PyTorch. We integrate symbolic constraint reasoning using a SAT solver backend, leveraging MiniSAT-style Boolean Constraint Propagation (BCP) via the PySAT library [71].

Domain constraints are compiled into conjunctive normal form (CNF) and enforced during search. After each action, the solver applies BCP to propagate implied assignments, ensuring that all intermediate states remain consistent with hard constraints.

For search knowledge discovery rule extraction (Section 5.4), we treat an MCTS decision as high-confidence when the dominant action receives more than 95% of visits and has a positive value estimate. Progress buckets require at least 10 such samples; each decision tree has maximum depth 3, requires at least 10 samples to split a node and 5 samples per leaf, and is fit with an 80/20 train-test split when enough data is available.

B.3 Baselines

We compare against **AMBer** (Autonomous Model Builder) [1], an RL framework for neutrino flavor model discovery. AMBer performs iterative model refinement using scalar rewards derived from the physics evaluation pipeline, including χ^2 fit quality, the number of free parameters, and a bonus for exploring higher-order $\mathbb{Z}_N$ symmetry groups. Unlike CDRL, AMBer does not use certificate-driven symbolic pruning or reusable constraint learning.

We additionally compare against several ablated and hybrid baselines, including pure MCTS search, DNN-only policies, symbolic-only SAT-based search [59], and a DNN+MCTS framework [18]. These baselines isolate the contributions of planning, neural guidance, and symbolic reasoning.

B.3.1 LLM Baseline Details

We evaluate a prompting-based baseline using GPT-5.3 [68]. The model is prompted to directly generate candidate neutrino flavor model matrices satisfying the required representation, charge, and VEV assignment format used throughout the paper.

The prompt contains: (i) a task description explaining the flavor model construction problem, (ii) formatting instructions specifying the matrix structure and allowed assignments, and (iii) 3 few-shot examples of valid candidate model matrices drawn from the corresponding theory space. The model is then asked to generate new candidate matrices autoregressively.

Generated outputs are parsed into matrix form and evaluated using the same symbolic constraints and physics evaluation pipeline used for all other methods. Invalid, incomplete, or unparsable outputs are counted as failed candidates.

We use temperature 0.8 and top-p 0.95 decoding for generation. The baseline does not use search, SAT-based propagation, iterative refinement, or certificate-driven feedback, and therefore serves as a reference point for pure prompting-based generation without structured exploration.

B.4 Neural Network Architecture

The policy-value network operates on a flattened matrix representation of the state. The architecture consists of:

- Fully connected layer: 512 units with BatchNorm and ReLU
- Fully connected layer: 256 units with BatchNorm and ReLU
- Policy head: linear layer to action dimension followed by log-softmax
- Value head: linear layer followed by $\tanh$

B.5 Additional Training Details

We provide the concrete hyperparameters for the MCTS and network training described in Section 4. We use:

- Max iterations: 30
- Episodes per iteration: 100
- MCTS simulations per move: 150
- MCTS c_{puct}: 0.5
- Batch size: 512
- Training epochs per iteration: 20
- Optimizer: Adam

The policy head is trained using cross-entropy loss against MCTS visit counts, while the value head is trained using mean squared error against normalized terminal rewards.

After each training iteration, the updated model is evaluated against the previous checkpoint in a single-agent self-play evaluation setting. Both models independently generate candidate solutions using the same environment and evaluation pipeline, and we compare their cumulative terminal rewards across a fixed number of evaluation episodes. The checkpoint is updated only if the new model achieves higher cumulative reward than the previous model. Unlike standard AlphaZero pit-play, this comparison does not involve a competitive two-player game, since neutrino model discovery is a single-agent environment.

B.6 Parallelization

We employ a portfolio-style parallelization strategy inspired by parallel SAT solvers. Multiple workers independently perform MCTS-guided self-play while sharing:

- neural network replay buffers
- a global database of learned conflict clauses

Conflict clauses discovered by any worker are synchronized globally and reused by all workers. Additionally, expensive terminal evaluations are cached to avoid redundant computation.

This combination of parallel search and clause sharing significantly improves exploration efficiency and sample efficiency.

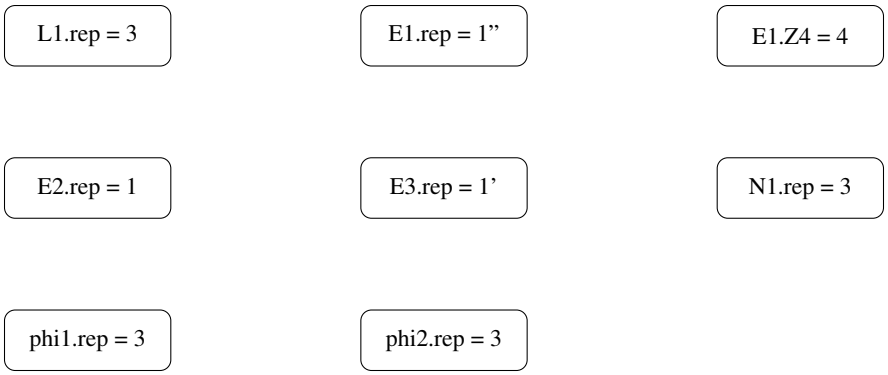
In our current implementation, learned clauses are accumulated monotonically and shared across workers, and we do not employ CDCL-style clause deletion. In practice, this remains computationally manageable because SAT propagation constitutes only a small fraction of the total runtime, while the dominant cost arises from the physics evaluation pipeline. Although the number of learned certificates grows over training, we did not observe clause management becoming a bottleneck in the theory spaces considered in this work. For larger domains, clause forgetting or activity-based clause management would be a natural extension.

C Search Knowledge Discovery Rule Galleries

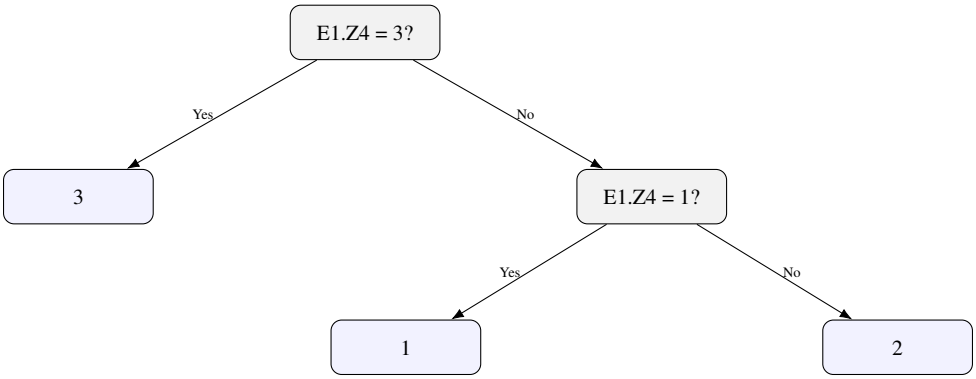
Below we list the full set of observations extracted from the high-confidence MCTS trajectories in each theory space, using the procedure described in Section 5.4. Each observation is a shallow decision tree that predicts one block assignment (a representation or a $\mathbb{Z}_N$ charge) from earlier assignments in the same partial model; an observation that collapses to a single leaf is a *constant assignment*, always taking the same value in the high-confidence set. For every tree we report the train and test accuracy of the predicted assignment. In the two-way trees, the left branch is the *Yes* outcome of the condition and the right branch is *No*. These rules are extracted post-hoc and reused as the soft policy bias of Algorithm 2; the aggregate effect of reusing them is reported in Figure 6.

C.1 $A_4 \times \mathbb{Z}_4$

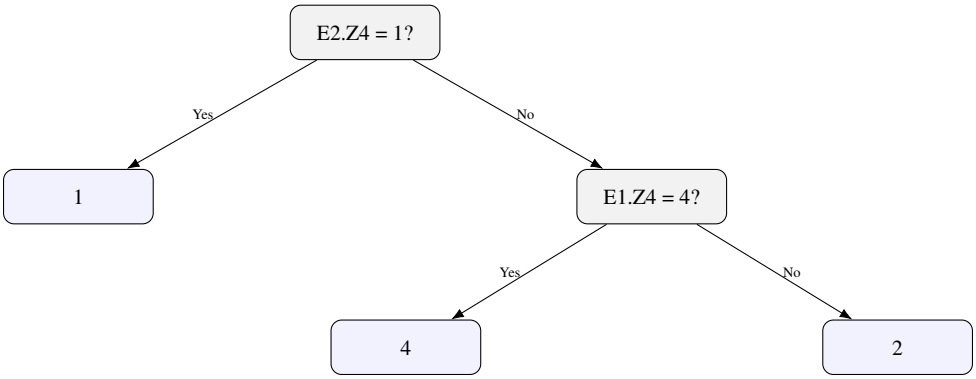
Constant assignments



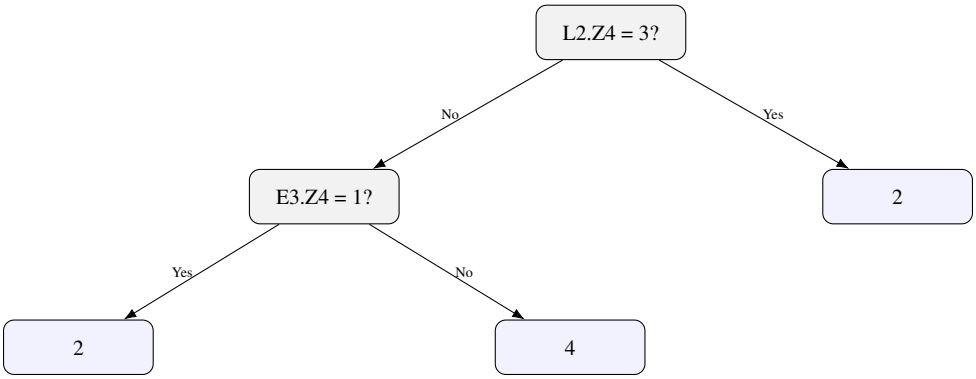
E2.Z4_charge Train: 0.917 Test: 0.917



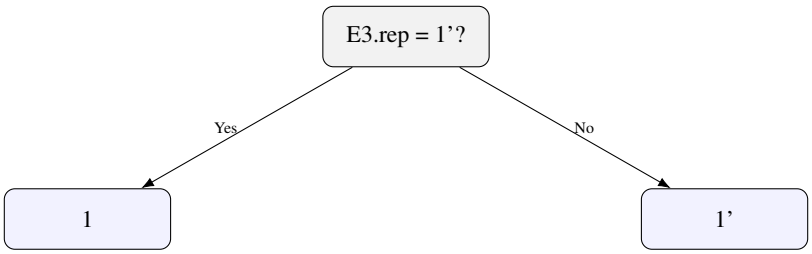
E3.Z4_charge Train: 0.972 Test: 0.944



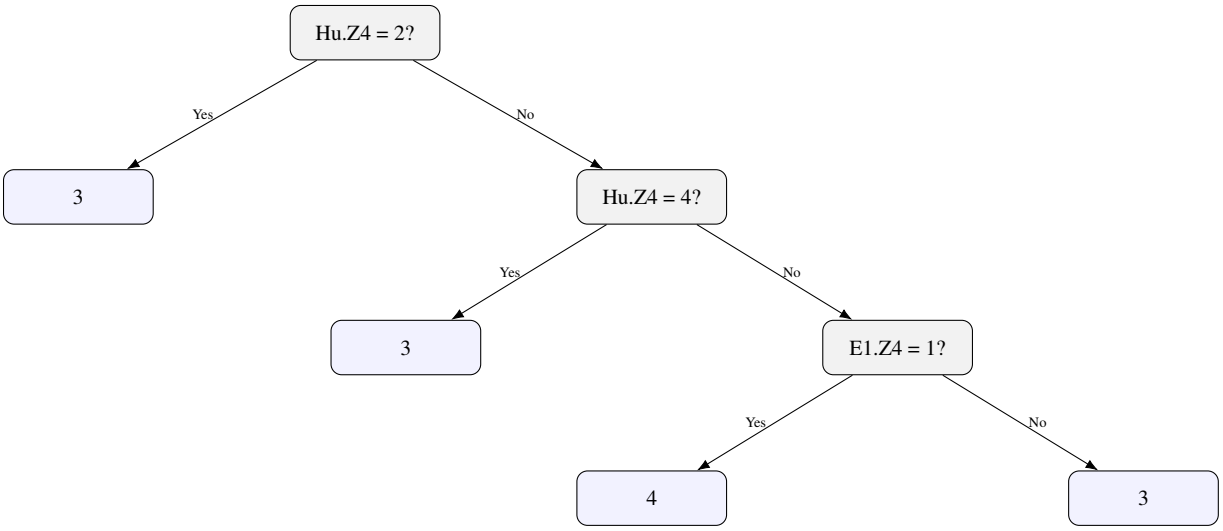
N1.Z4_charge Train: 0.911 Test: 1.000



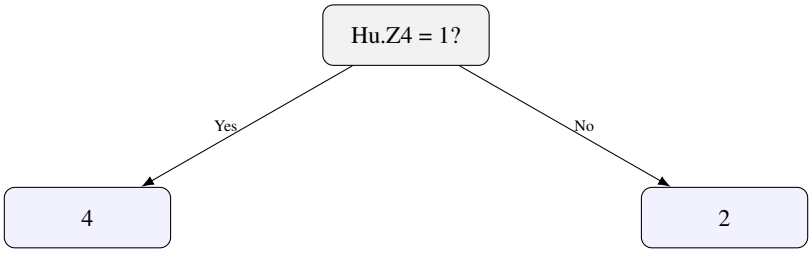
Hd.rep Train: 0.615 Test: NA



phi1.Z4_charge Train: 0.654 Test: 0.429

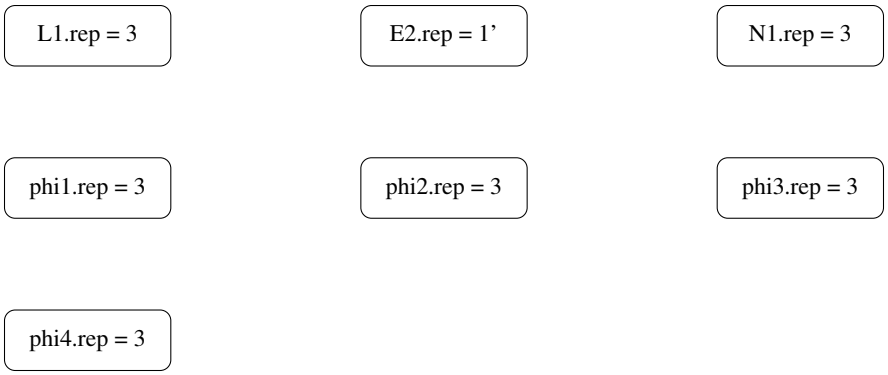


phi2.Z4_charge Train: 0.719 Test: 0.444

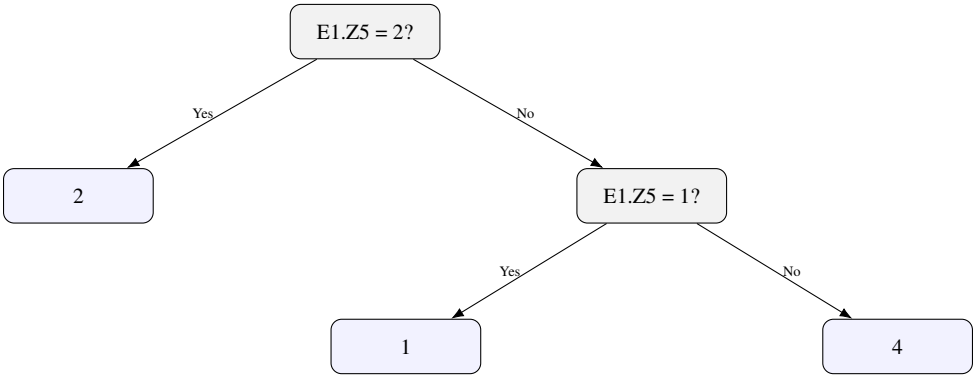


C.2 $A_4 \times \mathbb{Z}_N$

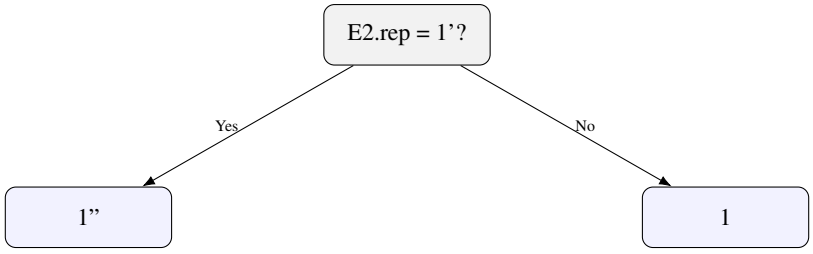
Constant assignments



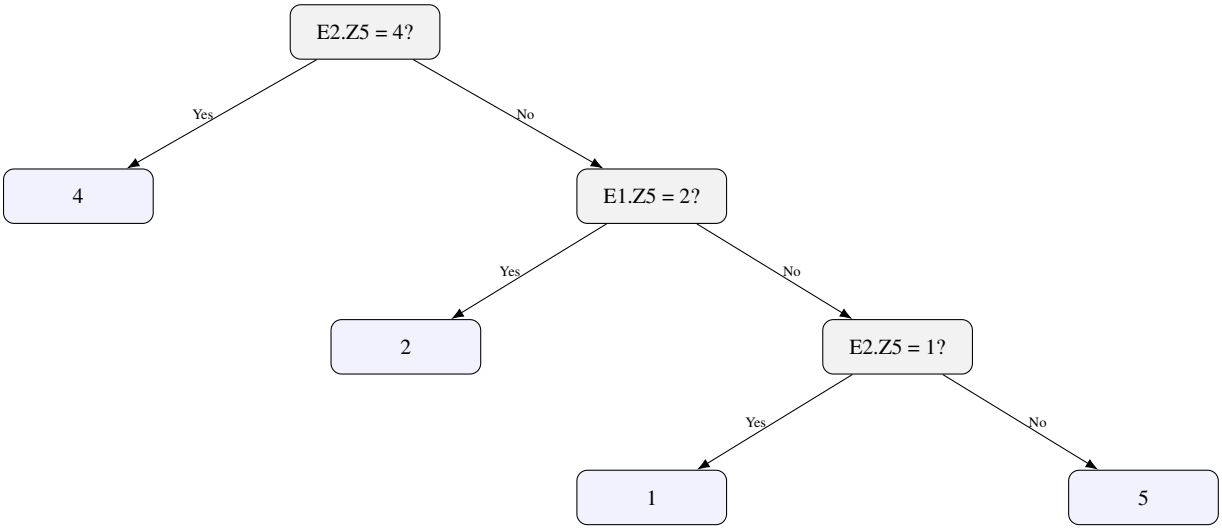
E2.Z5_charge Train: 0.938 Test: 0.882



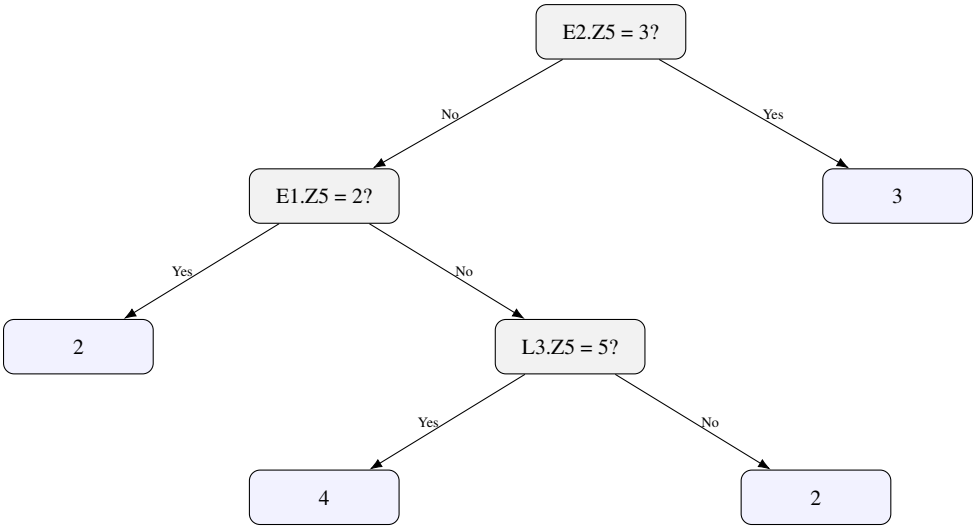
E3.rep Train: 0.707 Test: 0.727



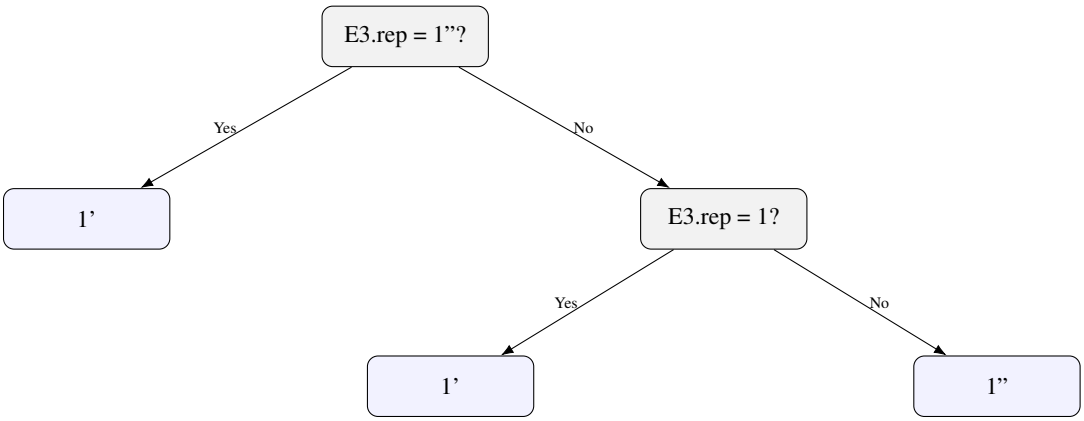
E3.Z5_charge Train: 0.926 Test: 0.968



N1.Z5_charge Train: 0.861 Test: 0.900



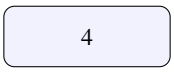
Hu.rep Train: 0.889 Test: 0.600



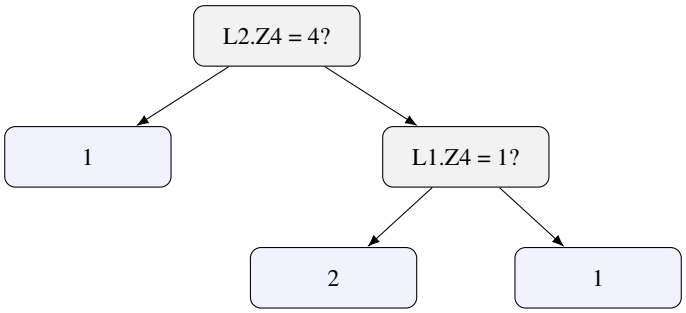
C.3 $T_{19} \times \mathbb{Z}_4$

Left edge: Yes Right edge: No

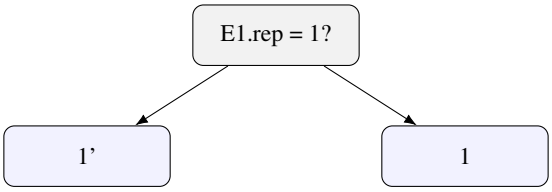
L1.Z4_charge Train: 0.792 Test: 0.714



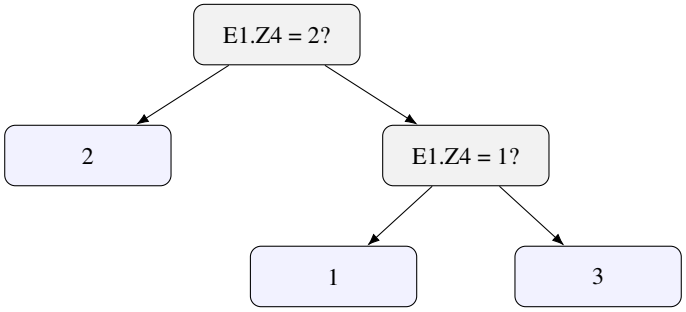
E1.Z4_charge Train: 0.611 Test: 0.600



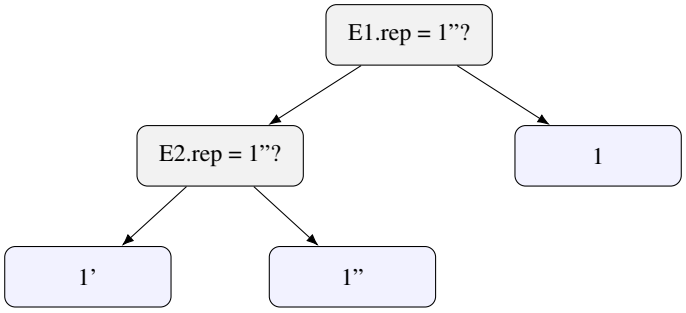
E2.rep Train: 0.950 Test: 0.905



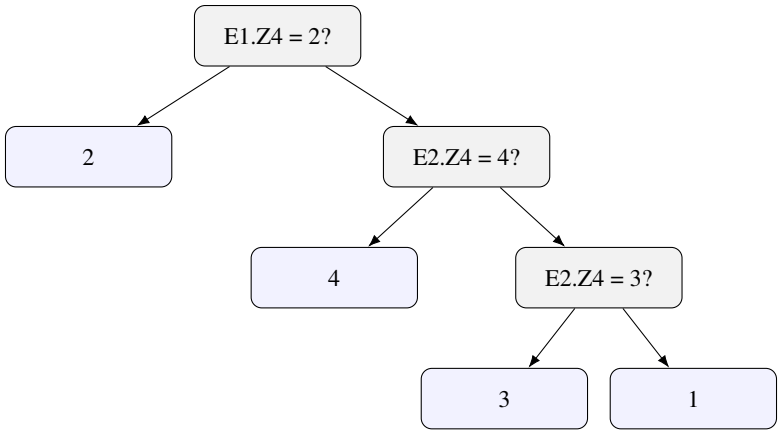
E2.Z4_charge Train: 0.938 Test: 0.857



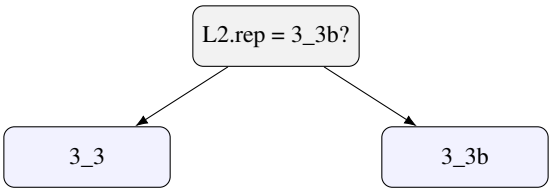
E3.rep Train: 0.685 Test: 0.694



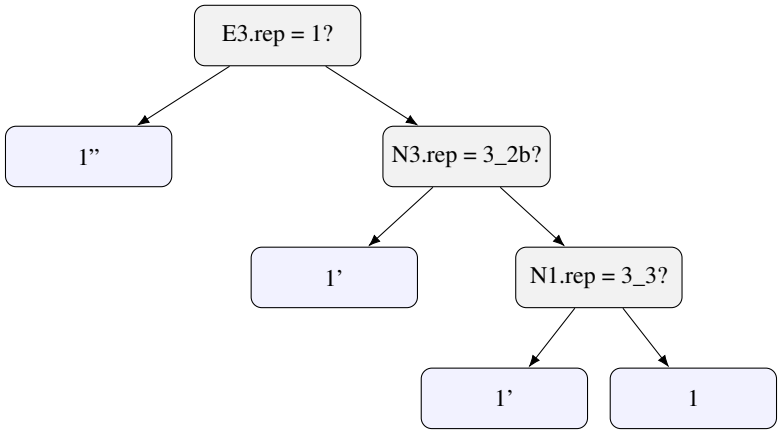
E3.Z4_charge Train: 0.924 Test: 0.926



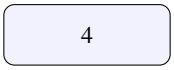
N1.rep Train: 0.960 Test: 0.932



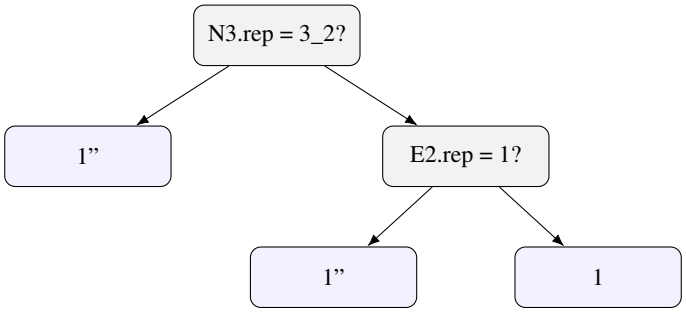
Hu.rep Train: 0.522 Test: 0.667



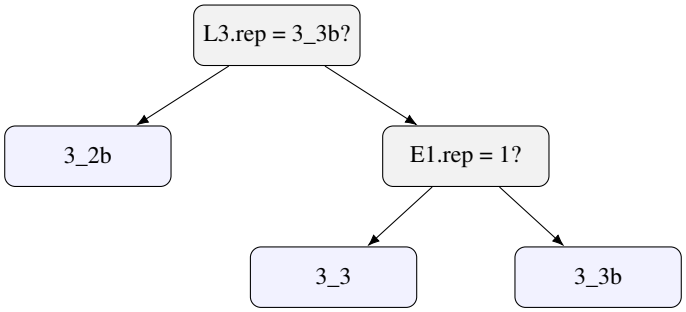
Hu.Z4_charge Train: 0.978 Test: 0.833



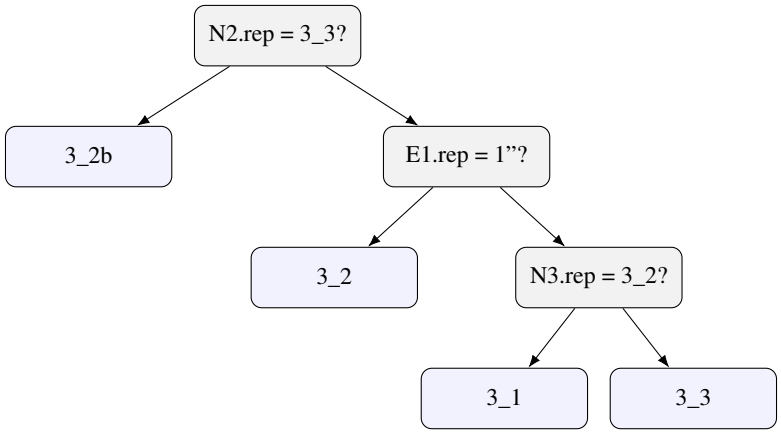
Hd.rep Train: 0.821 Test: 0.875



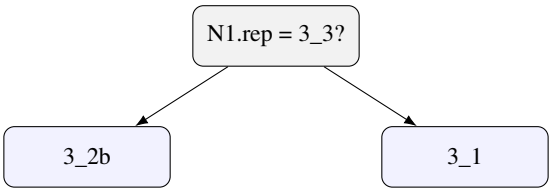
phi1.rep Train: 0.859 Test: 0.941



phi2.rep Train: 0.750 Test: 0.625



phi3.rep Train: 0.741 Test: 0.714



phi5.rep Train: 0.857 Test: 0.750