WORDBENCH: A BENCHMARK FOR LEARNING AND SEARCH IN ALGEBRAIC WORD PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mathematical discovery systems increasingly combine large language models (LLMs), reinforcement learning (RL), and formal verifiers, yet most mathematical benchmarks remain static collections of problems. We introduce **WordBench**, a benchmark for algebraic word problems that combines procedurally generated datasets with executable search environments and exact verification. A word problem asks whether finite symbolic expressions, or *words*, represent the same mathematical object under specified generators and relations. An open example is the Andrews–Curtis conjecture, where a successful certificate is a sequence of allowed presentation moves transforming a relator tuple to the trivial presentation. WordBench contains 24 families spanning rewrite and witness-construction problems, each with generative datasets, a deterministic symbolic search environment, and exact verification of submitted traces, action sequences, or algebraic objects in the Lean theorem prover. The released corpus, `WordBench-Lit`, contains 13,324 instances, and the underlying data generators can produce fresh instances beyond this set. We benchmark classical breadth-first search, PPO agents, and LLMs under common verifier-defined success criteria. The results reveal strongly solver-dependent empirical difficulty: for example, PPO solves 99.1% of instances on Thompson's group F while some open-source LLMs remain below 10%, whereas frontier LLMs achieve near-perfect performance on Artin–Tits and braid rewriting, where PPO remains below 1%. More broadly, frontier LLMs solve a large fraction of `WordBench-Lit`, but performance remains nonuniform across families, with Andrews–Curtis being substantially less solved. Beyond benchmark performance, WordBench's environments enable controlled experiments on search behavior, learning dynamics, and empirical relationships across mathematical problems. As a case study, we find that cross-family PPO transfer learning can solve new dataset instances not discovered by target-only training, with different source families providing complementary coverage. Together, these results position WordBench as a common verifier-facing benchmark for studying learning and search across diverse mathematical word problems.

1 INTRODUCTION

Mathematical reasoning has become a central testbed for LLMs, RL, and tool-augmented systems. Recent systems solve Olympiad-style geometry problems, search for formal proofs in Lean, discover new mathematical constructions through program search, and combine informal reasoning with machine-checked verification (Trinh et al., 2024; Hubert et al., 2026; Romera-Paredes et al., 2024; Ju et al., 2026). The benchmarks used to measure this progress, however, still leave important gaps. Many are finite and hand-curated, making saturation and contamination hard to avoid. Many are scored by final answer strings or prose explanations, which can reward a correct-looking output without verifying the mathematical object that produced it. Others focus on a single interface or a single problem family, making it difficult to compare how methods behave across different invariants, move systems, search horizons, and certificate formats (Cobbe et al., 2021; Hendrycks et al., 2021; Mirzadeh et al., 2025). These limitations are especially restrictive for RL, where the central object is not a prompt but an environment: a state space, an action space, exact transitions, and a verifier-defined success condition.

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

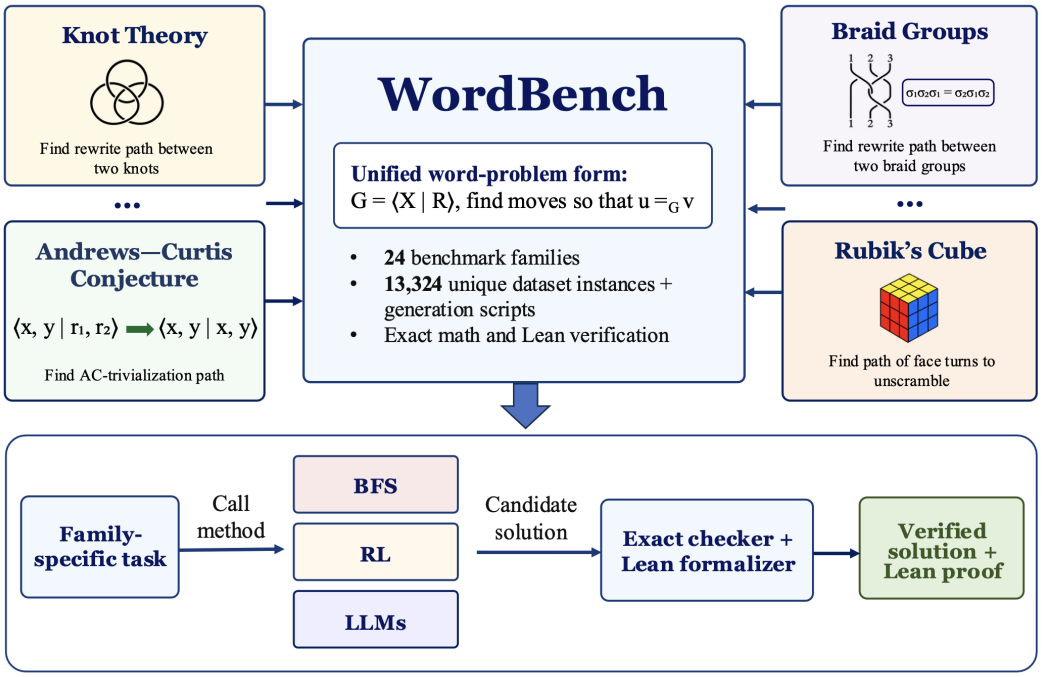

Figure 1: **Overview of WordBench.** The benchmark uses the common word-problem form $G = \langle X \mid R \rangle$ and $u =_G v$ to organize diverse mathematical search tasks. Each family defines a symbolic search problem with family-specific states, actions, transitions, and exact verification. Classical search, learned RL policies, and LLM-based methods can therefore be evaluated against the same verifier-defined success criteria.

We introduce **WordBench**, a unified, generative, and exactly checkable benchmark for algebraic word problems. A presentation $G = \langle X \mid R \rangle$ specifies generators X and relations R; the word problem asks whether two symbolic expressions u and v represent the same mathematical object, written $u =_G v$. A familiar example is Rubik's cube: face turns are generators, a scramble is a word, and a certificate is a move sequence returning the cube to the solved state. An open conjectural example is Andrews–Curtis trivialization, where the solver searches for a sequence of allowed presentation moves that transforms a relator tuple to the trivial presentation. Table 1 gives the full family landscape.

The fully general word problem for finitely presented groups is undecidable (Stillwell, 1982). Therefore, rather than binary decidability, WordBench instead focuses on *mathematical search*: given a concrete instance, a solver must construct an explicit sequence of valid moves or a witness that is verifier-checkable to certify success. Most families in the benchmark are decidable, but differ substantially in the structure and difficulty of the search required to produce such a certificate. For example, free groups reward local cancellation; RAAGs, Artin–Tits systems, and braid groups create neutral move plateaus (Charney, 2007; Birman, 1974); witness families require conjugators, subgroup expressions, automorphism traces, or substitutions (Kapovich & Myasnikov, 2002; Makanin, 1977; Plandowski, 2004; Myasnikov & Haralick, 2006); Burau and Gröbner families require exact algebraic arithmetic; and Rubik-style families become large finite planning problems (Agostinelli et al., 2019). For the open family of the Andrews–Curtis conjecture, positive paths are checkable, but timeouts are not theorems (Bridson, 2015; Shehper et al., 2024). We discuss this further in Section 3.

WordBench contains 24 families with procedural generators and difficulty knobs such as word length, witness length, scramble depth, and search horizon. The released corpus, `WordBench-Lit`, contains 13,324 instances. Each family provides an executable environment with exact transition semantics and a verifier for candidate solutions. These executable checkers are paired with Lean formalizations of the family definitions, admissible outputs, verifier predicates,

soundness statements, and the verified-solve relation (de Moura et al., 2015; The mathlib Community, 2020). This combination gives the benchmark both a scalable evaluation layer and a rigorous machine-checked semantic specification.

Each WordBench family exposes a symbolic search problem through a family-specific state representation, action space, transition function, and exact success condition. We evaluate three methods on these tasks: breadth-first search (BFS) as a classical search baseline, PPO as a learned policy trained through interaction with the symbolic environments, and LLMs as general-purpose reasoning models. Regardless of how a candidate solution is produced, benchmark credit is assigned only when the corresponding family verifier accepts it. This allows classical search, RL agents, and LLMs to be compared under the same exact mathematical acceptance criteria.

The empirical results reveal substantial variation across both problem families and solving methods. PPO solves nearly all instances for several local-rewrite and short witness-construction families, but remains limited under our training budgets on problems involving neutral rewrites, large action spaces, sparse rewards, and long-horizon search. The strongest LLMs achieve high overall coverage on `WordBench-Lit`, solving nearly all instances in many families, but their performance also varies substantially across problem types and model classes. Classical search, PPO, and LLMs therefore exhibit different strengths across symbolic search problems, illustrating why our heterogeneous benchmark is useful. We further study cross-family transfer learning: pretraining on different source problems can enable PPO to solve additional target instances beyond target-only training, increasing overall dataset coverage. These results are empirical properties of the current interfaces and search budgets rather than claims about intrinsic mathematical hardness.

We make the following **contributions.**

- **A unified benchmark for algebraic word-problem search.** WordBench organizes 24 families spanning rewrite and witness-construction problems across diverse algebraic and combinatorial structures. This allows for comparison of methods across families with different invariants, action spaces, horizons, and admissible outputs.
- **A generative corpus with mathematically grounded scaling parameters.** Each family comes with procedural generators and modifiable parameters such as word length, witness size, scramble depth, and search horizon. The released `WordBench-Lit` corpus contains 13,324 instances, and we provide data generation scripts that can produce fresh, exactly checkable instances beyond this set.
- **Common symbolic search environments and exact evaluation across methods.** Each family is implemented with bounded symbolic states, finite family-specific actions, exact transitions, and verifier-defined success. We benchmark breadth-first search, PPO agents, and LLMs across these tasks, revealing solver-dependent performance across problem families under the same acceptance criteria. The shared environments also enable studies beyond static benchmarking, including our cross-family transfer experiments.
- **Lean-grounded semantics and exact verification.** We formalize the benchmark semantics in Lean, including the mathematical objects, allowed transformations, admissible outputs, and correctness conditions for each family. Solver outputs can then be translated into corresponding Lean proofs, providing machine-checked certificates for discovered solutions.

2 Related Work

Symbolic and algebraic search. Classical algorithms for algebraic word problems often exploit problem-specific structure, such as normal forms, completion procedures, or specialized decision algorithms (Stillwell, 1982; Holt, 1994). WordBench instead studies constructive search under a prescribed action space: a solver must produce an executable rewrite trace or witness that is accepted by the family verifier. Family-specific ML/RL precedents and the role of each WordBench task are summarized in Appendix A.

Learning-guided mathematical search. Learning has been used to guide search in combinatorial and symbolic domains, including Rubik's cube (McAleer et al., 2018; Agostinelli et al., 2019), braid and knot problems (Gukov et al., 2021), Gröbner-basis computation (Peifer et al., 2020), and Andrews–Curtis search (Shehper et al., 2024; Zhang et al., 2025). Related work in interactive the-

Table 1: Overview of the 24 WordBench families, including task type, formal task, decidability status, corpus size, action-space size, and padded observation dimension. *The benchmark task is decidable by positive construction but is related to an open mathematical problem.

Family	Task type	Formal task	Status	Unique rows	$\lvert\mathcal{A}\rvert$	$\lvert\text{obs}\rvert$
Andrews–Curtis trivialization	Rewrite	$\exists \tau \in \mathrm{AC}^* : \tau(P) = P_{\mathrm{triv}}$	Open	1190	12	72
Artin–Tits word problem	Rewrite	$w =_A e, \quad A = A(M_{abcd})$	Decidable*	1000	30464	128
Baumslag–Solitar BS(1,2)	Rewrite	$w =_G e$ in $\langle a, t \mid tat^{-1} = a^2 \rangle$	Decidable	500	448	64
Braid word rewriting	Rewrite	$u =_{B_n} v$	Decidable	500	1792	128
Burau B_4 search	Rewrite	$\exists w : \beta(w) = \beta(w_\star)$	Decidable	500	72	91
Burau matrix equality	Rewrite	$\beta(w) \stackrel{?}{=} I$	Decidable	500	66	64
Coxeter type-A rewriting	Rewrite	$u =_{S_n} v$	Decidable	482	647	128
Finite permutation action	Rewrite	$\exists w : w \cdot s = t$	Decidable	59	4	10
Free-group conjugacy	Witness	$\exists g : g^{-1} u g =_F v$	Decidable	500	6	80
Free-group subgroup membership	Witness	$w \in \langle h_1, \ldots, h_k \rangle$	Decidable	500	6	144
Free-group word	Rewrite	$u =_F v$	Decidable	500	31	64
Graph-product word	Rewrite	$u =_G v$	Decidable	500	127	259
Gröbner normal form	Rewrite	$r = \mathrm{NF}_G(f)$ in $K[x_1, \ldots, x_n]/\langle G \rangle$	Decidable	1000	9	614
Heisenberg collection	Rewrite	$u =_{H(\mathbb{Z})} v$	Decidable	500	96	64
Kaplansky group rings	Witness	$\exists \alpha, \beta \neq 0 : \alpha\beta = 0$ in $K[G]$	Decidable*	30	4	64
KBMAG rewriting	Rewrite	$w \Rightarrow^* \mathrm{NF}(w)$	Decidable	500	1568	65
Knot equivalence	Rewrite	$\exists \pi : D \Rightarrow_\pi D'$	Decidable	1000	193	128
Makanin word equations	Witness	$\exists \sigma : \sigma(L) = \sigma(R)$	Decidable	500	9	60
Path-RAAG word	Rewrite	$u =_{A_{P_n}} v$	Decidable	64	127	256
Rubik's cube	Rewrite	$a_T \cdots a_1 \cdot s = s_{\mathrm{solved}}$	Decidable	1000	18	324
Small-cancellation word	Rewrite	$w =_G e$ in a $C'(1/6)$ presentation	Decidable	500	145	120
Small-overlap monoid word	Rewrite	$u =_M v$ in $M = \langle \Sigma \mid R \rangle$	Decidable	500	97	98
Thompson's group F	Rewrite	$\exists w \in \langle x_0, x_1 \rangle : w \cdot g = e$	Decidable	499	4	129
Whitehead free group	Witness	$\exists \varphi \in \mathrm{Aut}(F_n) : \varphi(u) = v$	Decidable	500	24	64

orem proving combines learned search with formal checking (Yang et al., 2023). WordBench complements these efforts by placing 24 heterogeneous mathematical search families behind a common executable interface and exact verification criterion, enabling comparison of classical search, learned policies, and general-purpose models across problem structures.

3 WordBench Design

3.1 Problem Families and Datasets

Problem families were selected to provide diversity in mathematical structure, transformation rules, task goals, and search difficulty. Each family also has an exact verifier: a submitted object is accepted only if it can be replayed or checked against the underlying mathematical rules. WordBench contains 24 problem families spanning combinatorial group theory and related problems in algebra and topology, including tasks over free groups, rings, monoids, and braid groups. Each family defines its own state representation, transformation actions, and goal condition, together with a dataset of problem instances. Table 1 summarizes the 24 families and their key characteristics.

The 24 families fall into two broad types of mathematical search: *rewrite* and *witness construction*. In rewrite problems, the agent must discover a sequence of valid transformations that takes a starting state to a specified target state, such as a target word or the identity. In witness construction problems, the agent must construct an explicit mathematical object satisfying the desired property. For example, an agent may build a conjugator by successively appending or deleting words until a valid witness is obtained. In both settings, the solution is an executable sequence of actions, in addition to verifying that the end state satisfies the task goal.

WordBench also spans different levels of mathematical status in terms of open discovery. We include the open Andrews–Curtis (AC) conjecture, where our problem instances are drawn from the Miller–Schupp family of potential counterexamples. Finding an AC move rewrite sequence for an instance eliminates it as a potential counterexample. The remaining 23 families provide constructive tasks with instances known to be solvable. Two of these families, the Artin-Tits word problem

and Kaplansky group rings tasks, are connected to open mathematical questions: decidability of the Artin–Tits word problem for all groups is unknown, and the Kaplansky Zero Divisor Conjecture is open for torsion-free groups. However, the instances included in WordBench are positive, constructive instances for which a valid solution is known or can be verified. These constructive instances are decidable, meaning they can be verified as positive solutions through an algorithm that terminates in finite time. However, decidability of a problem does not imply that the corresponding sequence of actions is easy to find. For example, in the Kaplansky group rings family, algebraic reasoning can establish the existence of zero divisors in the group ring $\mathbb{F}_2[C_n]$, but the WordBench task requires the agent to search for an explicit pair α, β satisfying $\alpha\beta = 0$, constructing the witness through the environment's actions. Thus, knowing that a solution exists, or having an algorithm that decides whether one exists, is distinct from discovering the concrete sequence of actions that constructs it. WordBench addresses this gap by requiring agents to discover and submit executable solution paths, providing a testbed for studying mathematical search directly.

All WordBench datasets are generative and, for most families, can be scaled to increase search difficulty by varying parameters such as word length or rewrite scrambling depth. For our experiments, we instantiate each family using the configurations detailed in Appendix C. These configurations define the datasets used for our benchmark experiments for consistent baselines. Our dataset generation scripts have the ability to generate harder instances in future work.

3.2 Executable Environments

Each WordBench family f is implemented as a deterministic, finite-horizon environment $(\mathcal{S}_f, \mathcal{A}_f, T_f, G_f, R_f, H_f)$, where $\mathcal{S}_f$ and $\mathcal{A}_f$ are the family-specific state and action spaces, T_f is the deterministic transition function, G_f defines the goal condition, R_f is the training reward, and H_f is the episode horizon. All 24 families expose a common reset/step interface, allowing the same learning and search methods to be applied and compared across substantially different mathematical problems.

For an instance with initial state s_0, a candidate solution is a finite sequence of actions $a_0, \ldots, a_{t-1}$. The sequence is valid if applying these actions sequentially through the deterministic transition function T_f produces states according to $s_{k+1} = T_f(s_k, a_k)$ and ultimately reaches a state satisfying the family-specific goal condition G_f. In rewrite tasks, the goal typically requires that the current mathematical object reaches a specified target, such as the identity or a target word. In witness-construction tasks, the goal is an exact algebraic condition on the constructed object, such as a conjugator z satisfying $zuz^{-1} = v$. A proposed solution can be certified by resetting the environment to the initial instance, executing the claimed actions using T_f, and checking the resulting state against G_f. Illegal or inapplicable actions are handled by the environment according to the family-specific transition rules. Shaped rewards R_f may encourage progress during training, but do not impact solution validity.

3.3 Lean Formalization

To provide an additional layer of correctness, we include Lean formalizations corresponding to the WordBench environments. The formalizations mirror the executable environment rules, allowing a solved action trace to be directly passed through our proof-generation scripts to produce a corresponding Lean proof. This creates a corpus of machine-checked mathematical search solutions and a foundation for future capabilities in which AI systems can discover solutions directly as formally verified mathematical proofs.

These Lean formalizations ensure that each environment is anchored to the concrete problem. Each Lean environment mirrors the RL WordBench environment by taking in the starting state and action sequence of an RL agent and outputting a certified Lean theorem that the initial problem has been solved. By performing replays in Lean in addition to the WordBench environment, each problem solve by an agent can be confirmed to correspond to a valid solution instead of agent or environmental errors. Additionally, each Lean environment we provide formalizes a section of the basic definitions and primary theorems for each problem, which we hope can aid future mathematicians in developing the corpus of knowledge surrounding each one.

4 EXPERIMENTAL SETUP

We evaluate learning and search methods across the 24 WordBench families. An instance is counted as solved only when an action sequence, replayed from its initial state using the environment's transition function T_f, reaches a state satisfying the family-specific goal condition G_f.

4.1 CLASSICAL SEARCH

We use breadth-first search (BFS) as the classical baseline. BFS expands successor states by applying the same environment actions and transition function T_f used by the learning agents, and checks success using the same goal condition G_f. Search is bounded by a maximum of 100,000 expanded nodes for all families except the Artin–Tits word problem, whose large action space of 30,464 actions makes exhaustive expansion substantially more costly; we therefore use a maximum of 500 expanded nodes per instance for this family.

4.2 REINFORCEMENT LEARNING

4.2.1 PPO BASELINES

We use proximal policy optimization (PPO) with an MLP architecture. Actor and critic branches are separate, with each consisting of a family-specific input projection layer, a two-layer hidden trunk of width 256, and a task-specific output head. All PPO experiments use a fixed budget of 1M or 10M total environment steps with 256-step rollouts. We use GAE $\lambda = 0.95$, PPO clipping coefficient 0.2, entropy coefficient 0.01, value loss coefficient 0.5, and gradient clipping at 0.5. The remaining family-specific training and environment hyperparameters are specified in Appendix D. All PPO baselines and subsequent transfer learning experiments are conducted across 5 seeds.

For PPO, we measure performance by training-time discovery coverage: the number of unique dataset instances for which a valid solution is found at least once during training. This metric reflects the discovery-oriented setting we study, where finding a verifier-accepted solution to a previously unsolved instance is itself the objective, rather than estimating expected performance of the final policy on an underlying data distribution. Generalization of the final learned policy to newly generated instances is a distinct question that we leave to future work.

4.2.2 TRANSFER LEARNING

We investigate cross-family transfer learning by first training a source PPO agent on one WordBench family and then using its trained weights to initialize the corresponding network on a target family. Because the actor and critic branches are separate, we independently transfer and ablate over the actor and critic networks.

The hidden layers have a fixed width of 256 across all tasks and are transferred directly. Since input dimensions vary across problems, the input projection layer is adapted when transferred. When the target input dimension is larger than the source, the shared portion of the projection weights is copied, and the remaining rows are randomly initialized. When the target input dimension is smaller, the projection weights are truncated. All remaining parameters are randomly initialized. Transferred weights are loaded once at initialization and subsequently optimized using the standard PPO objective with generalized advantage estimation (GAE). All transfer experiments use 1M environment steps of source pretraining followed by 1M steps of target training. We compare against a baseline of target-only training for 2M steps, which matches the total budget of the transfer procedure.

4.3 LARGE LANGUAGE MODELS

We evaluate large language models as zero-shot, open-loop planners over the same WordBench environments used by our other baselines. For each instance, the model receives a description of the mathematical problem, the initial state, and definitions of the available environment actions. The model is then prompted to return a JSON object containing a sequence of integer action indices, for example, {"actions":[1,2]}. The proposed sequence is replayed in the environment and counted as correct only if it reaches the family-specific goal condition, so successful generations correspond to verifier-accepted executable certificates rather than textual answers. We evaluate five

LLMs, including open-weight models: GPT 6 - Astra, Opus 5.5, Qwen3 235B, DeepSeek V3.2, and Kimi K3. We use per-attempt output-token ceilings of 16,384 on most families, increased to 1,000,000 for Andrews–Curtis, Burau B_4 search, and Rubik's cube. We use provider-default sampling, without explicitly setting temperature or a random seed, and report `pass@1`: one scored attempt per unique instance under each reported configuration. Models have no access to tools or intermediate verifier feedback.

5 Results

5.1 Benchmark Baselines

We report family-level solve rates for breadth-first search (BFS), PPO after 1M and 10M environment steps, and zero-shot LLMs. PPO results measure training-time discovery coverage, while LLM results report `pass@1`. All reported solutions must satisfy the same family-specific verifier.

Table 2: Unique instances solved (%). PPO reports mean ± standard deviation over five seeds. LLM results are pass@1.

Family	N	BFS	PPO 1M	PPO 10M	Astra	Opus	Qwen3 235B	DeepSeek V3.2	Kimi K3
Tier I: saturated at 1M PPO steps.									
Baumslag–Solitar BS(1,2)	500	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0	98.8	80.4	97.2	89.2
Burau matrix equality	500	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0	95.4	66.2	79.0	72.6
Finite permutation action	59	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0	100.0	79.7	100.0	96.6
Free-group word	500	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0	100.0	99.2	99.8	96.8
Gröbner normal form	1,000	99.2	100.0 ± 0.0	100.0 ± 0.0	100.0	99.9	94.1	95.0	98.1
Kaplansky group rings	30	20.0	100.0 ± 0.0	100.0 ± 0.0	100.0	100.0	86.7	93.3	90.0
Small-cancellation word	500	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0	95.8	32.0	67.0	94.4
Small-overlap monoid word	500	100.0	100.0 ± 0.0	100.0 ± 0.0	100.0	99.2	97.6	100.0	96.6
Tier II: partially solved at 1M PPO steps.									
Coxeter type-A rewriting	482	88.2	21.3 ± 14.3	95.5 ± 2.6	100.0	99.8	97.9	99.8	97.1
Free-group conjugacy	500	0.0	26.3 ± 1.2	26.8 ± 1.1	100.0	99.8	70.8	97.0	97.0
Free-group subgroup membership	500	100.0	74.2 ± 1.5	78.7 ± 1.1	100.0	99.2	86.4	99.0	95.6
Graph-product word	500	41.2	34.6 ± 2.2	60.1 ± 1.0	100.0	98.6	43.2	92.0	63.8
Heisenberg collection	500	97.8	89.4 ± 0.9	99.0 ± 0.4	100.0	99.4	90.6	94.0	94.8
Makanin word equations	500	100.0	76.6 ± 0.9	85.9 ± 0.4	100.0	97.4	93.6	98.0	97.4
Path-RAAG word	64	15.6	18.4 ± 1.3	25.0 ± 1.1	100.0	100.0	95.3	98.0	96.9
Thompson's group F	499	87.0	44.2 ± 8.0	99.1 ± 0.3	100.0	94.0	2.4	33.0	7.2
Whitehead free group	500	100.0	90.2 ± 1.8	99.0 ± 0.2	100.0	95.0	29.0	54.0	56.8
Tier III: challenging at 1M PPO steps or open discovery.									
Andrews–Curtis trivialization	1,190	10.3	1.4 ± 0.2	6.5 ± 0.8	45.9	15.4	1.2	9.5	19.2
Artin–Tits word problem	1,000	0.0	0.0 ± 0.0	0.0 ± 0.0	100.0	96.5	0.0	3.0	1.2
Braid word rewriting	500	6.8	0.2 ± 0.1	0.4 ± 0.2	100.0	97.2	30.0	82.0	67.6
Burau B_4 search	500	38.6	9.4 ± 1.1	9.7 ± 1.6	94.8	53.2	0.0	1.6	26.6
KBMAG rewriting	500	100.0	0.0 ± 0.0	0.8 ± 0.4	100.0	99.8	83.0	97.0	96.4
Knot equivalence	1,000	84.3	8.8 ± 2.5	11.3 ± 4.9	100.0	99.5	39.8	75.0	83.6
Rubik's cube	1,000	0.0	0.0 ± 0.0	0.0 ± 0.0	97.6	86.1	0.0	0.0	0.0
Overall (weighted by cases)	**13,324**	**62.4**	$\mathbf{45.3 \pm 0.7}$	$\mathbf{52.9 \pm 0.4}$	**94.8**	**88.2**	**48.6**	**63.2**	**63.3**

PPO performance at 1M steps reveals three empirical regimes that reflect different search structures under the current environment and training setup. *Saturated* families, for which PPO solves all instances, generally admit short certificates, local progress signals, or compact witness spaces. *Partially solved* families require more sustained search or planning: for example, Coxeter type-A rewriting requires selecting among many relation–position combinations, while Thompson's group F requires exact planning over finite symbolic representations of an infinite group. *Challenging* families, for which PPO solves at most 10% of instances at 1M steps, tend to involve neutral rewrites, large action spaces or branching factors, sparse terminal rewards, or long solution trajectories. Additional training supports this distinction: Coxeter type-A rises from 21.3% to 95.5% and Thompson's group F from 44.2% to 99.1% at 10M steps, whereas Andrews–Curtis rises only from 1.4% to 6.5%, braid rewriting from 0.2% to 0.4%, and Artin–Tits remains at 0%.

We next compare with BFS. On Kaplansky group rings, BFS solves only 20% of instances, whereas PPO reaches 100% at both training budgets. Conversely, BFS solves 100% of the KBMAG rewriting and free-group subgroup membership instances, compared with 0.8% and 78.7%, respectively, for PPO after 10M environment steps. BFS also reaches a higher solve rate of 10.3% on Andrews–Curtis, compared with 6.5% for PPO at 10M environment steps, which is consistent with prior

work showing that systematic search remains competitive for this problem (Shehper et al., 2024). These differences highlight complementary strengths: BFS can be highly effective when solutions lie within a manageable search depth, but its cost grows rapidly with the size and branching factor of the search space. PPO can learn to prioritize productive actions and avoid exhaustive expansion, but this becomes difficult when successful trajectories are rare, intermediate progress provides little signal, or reward is only obtained after a complete solution is found.

LLMs, which possess reasoning and planning capabilities, reveal a substantially different profile. Astra achieves 94.8% overall and solves all `WordBench-Lit` instances for 21 of the 24 families, including several that remain difficult for task-specific RL, such as Artin–Tits rewriting, braid rewriting, KBMAG rewriting, and knot equivalence. At the same time, performance is highly family- and model-dependent. Thompson's group F and Whitehead reduction, for example, reach 99.1% and 99.0%, respectively, with PPO after 10M environment steps, while several LLMs perform substantially worse: on Thompson's group F, Qwen3 235B, DeepSeek V3.2, and Kimi K3 achieve 2.4%, 33.0%, and 7.2%, respectively; on Whitehead reduction, the same models achieve 29.0%, 54.0%, and 56.8%. LLM performance across models also varies sharply on families such as Burau B_4 search and Rubik's cube. Andrews–Curtis remains the clearest shared challenge, with the strongest LLM solving 45.9% of instances.

Together, these results show that empirical difficulty in WordBench depends strongly on both the mathematical structure of a family and the solver used to search it. The PPO regimes capture meaningful differences under one learning setup, while BFS and LLMs produce substantially different performance orderings across the same problems. This variation is precisely what the heterogeneous WordBench environments and families are designed to expose.

5.2 Transfer Learning Results

Pretraining on different source families can enable PPO to solve new instances that are not reached by target-only training, thereby increasing overall dataset coverage, as shown in Figure 2. Across targets, the main effect of transfer is not necessarily a large increase in aggregate solve count compared to the baseline, but a change in *which* instances are solved. To quantify this complementarity, Table 3 reports the exclusivity percentage of newly solved instances, defined as the fraction of newly solved instances that are solved by exactly one of the source-pretrained agents. A high exclusivity percentage therefore indicates that different source families lead PPO to discover distinct solutions, rather than repeatedly recovering the same ones. Understanding why particular source families unlock particular target instances, and whether these relationships can be predicted from problem structure or learned representations, is an important direction for future work.

Table 3: Complementary target-instance coverage from cross-family transfer. *Base* is the 2M target-only union of solved rows. *New* is the number of additional rows solved by the 5-seed transfer union. *# src.* counts source families contributing at least one new row, and *Excl.* is the fraction of new rows solved by only one source. *Best src.* gives the source(s) contributing the most new rows, with the corresponding count in brackets.

		Actor transfer				Critic transfer			
Target	**Base**	**New**	**# src.**	**Excl.**	**Best src.**	**New**	**# src.**	**Excl.**	**Best src.**
Knot	174	210	23	47.6%	Artin–Tits [33]	213	23	53.1%	Gröbner, BS(1,2) [28]
AC	65	81	23	35.8%	Whitehead [20]	66	23	37.9%	Coxeter A [15]
Burau search	74	55	23	30.9%	Small overlap, Small canc., Permutation [15]	70	23	37.1%	Free word [23]
Coxeter A	381	41	20	39.0%	Conjugacy [15]	41	23	34.1%	RAAG [13]
Thompson F	467	6	7	66.7%	Rubik, Conjugacy [2]	14	19	35.7%	Rubik, KBMAG [6]
RAAG	13	5	21	20.0%	Kaplansky, Artin–Tits [4]	5	18	60.0%	Thompson F, KBMAG, Graph prod., Burau eq., Burau search [2]
Braid	3	5	6	60.0%	Small overlap, Heisenberg [2]	3	10	33.3%	Burau eq. [2]
Heisenberg	478	4	13	25.0%	Small canc., Graph prod., Free word, Braid [2]	7	18	42.9%	Thompson F, Small canc., Rubik, Makanin, Permutation, BS(1,2) [2]
KBMAG	1	1	1	100%	Braid [1]	0	0	–	–

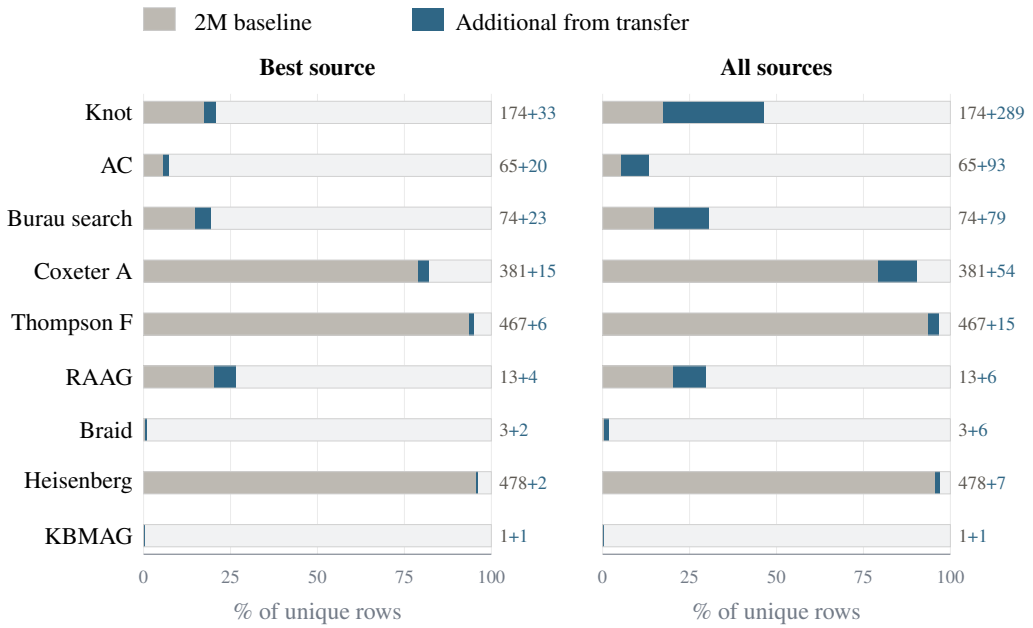

Figure 2: Unique-row dataset coverage on unsaturated target problems. Gray bars are the 5-seed union after 2M same-problem continued training. Left: additional unique rows from the single best source-to-target transfer (5-seed union), taking the better of actor and critic. Right: additional unique rows from the union over all actor and critic source-to-target runs. Numbers at right of the bars are unique-row counts (2M baseline + additional from transfer). Targets with no additional transfer coverage are omitted.

6 Conclusion

We present WordBench, a unified benchmark for constructive mathematical search across diverse algebraic, group-theoretic, and topological problems. WordBench places heterogeneous problem families behind executable search environments with rigorous mathematical verification in Lean. This provides a common setting in which classical search, learned policies, and general-purpose reasoning models can be studied across different mathematical structures. Our experiments show that empirical difficulty is strongly dependent on both the problem structure and the solver used to search it. Families that are challenging for PPO can be solved nearly completely by advanced LLMs, while other families show the opposite pattern or remain difficult across methods.

Several future directions remain. First, frontier LLMs approach saturation on much of the released `WordBench-Lit` corpus. Because WordBench families are procedurally generated, future work can generate larger and more difficult instance sets specifically targeted for these models, and measure performance trends as problem parameters scale. Second, the cross-family transfer results open a range of interpretability questions about whether learned policies and representations capture meaningful mathematical relationships between families, and whether such relationships can be predicted in advance. Overall, WordBench provides a controlled setting for studying these kinds of questions and broader investigations about learning and search across algebraic problems under exact verification.

AI USE STATEMENT

In this work, we used generative AI tools for editing and improving the readability of the paper, reference formatting, and reviewing and editing software code. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results. AI tools for providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and assisting with translation are not applicable to this work.

All AI-assisted text, analysis, and code were reviewed by the authors, and AI-assisted code was tested for correctness. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work develops an evaluation benchmark for exact symbolic reasoning and does not involve human subjects or personal data. We distinguish verifier-accepted certificates from unknown or timed-out cases and aim to report experimental results and limitations accurately. Data provenance and responsible reporting are discussed further in Appendix H, and limitations and future directions are discussed in Section 6. We are not aware of any ethical concerns raised by this work under the Code of Ethics.

REPRODUCIBILITY STATEMENT

Reproducibility details are provided in Appendix E. We also submit supplementary materials at `https://anonymous.4open.science/r/WordBench-4555`, including the executable environments for all 24 families, the complete `WordBench-Lit` corpus, data-generation scripts for producing new instances under different parameter settings, and the corresponding Lean formalizations.

REFERENCES

P. A. Abdulla, M. F. Atig, J. Cailler, C. Liang, and P. Rümmer. Guiding word equation solving using graph neural networks, 2024. arXiv:2411.15194, 2024.

F. Agostinelli, S. McAleer, A. Shmakov, and P. Baldi. Solving the Rubik's Cube with deep reinforcement learning and search, 2019. *Nature Machine Intelligence*, 1:356–363, 2019. doi:10.1038/s42256-019-0070-z.

T. Applebaum, S. Blackwell, A. Davies, T. Edlich, A. Juhász, M. Lackenby, N. Tomašev, and D. Zheng. The unknotting number, hard unknot diagrams, and reinforcement learning, 2025. *Experimental Mathematics*, published online, 2025. doi:10.1080/10586458.2025.2542174. arXiv:2409.09032.

E. Artin. Theory of braids, 1947. *Annals of Mathematics*, 48(1):101–126, 1947.

Olaf Bachmann and Hans-Gert Gräbe. The SymbolicData Project, 2000. Available at `https://symbolicdata.github.io/Papers/issac2000.pdf`.

G. Baumslag and D. Solitar. Some two-generator one-relator non-Hopfian groups, 1962. *Bulletin of the American Mathematical Society*, 68:199–201, 1962.

J. S. Birman. Braids, Links, and Mapping Class Groups, 1974. Princeton University Press, 1974.

M. R. Bridson. The complexity of balanced presentations and the Andrews–Curtis conjecture, 2015.

B. A. Burton. Introducing Regina, the 3-manifold topology software, 2004. *Experimental Mathematics*, 13(3):267–272, 2004.

J. W. Cannon, W. J. Floyd, and W. R. Parry. Introductory notes on Richard Thompson's groups, 1996. *L'Enseignement Mathématique*, 42:215–256, 1996.

R. Charney. An introduction to right-angled Artin groups, 2007. *Geom. Dedicata*, 125:141–158, 2007.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems, 2021. arXiv:2110.14168, 2021.

D. Cox, J. Little, and D. O'Shea. Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 2015. Springer, 4th edition, 2015.

L. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer. The Lean theorem prover (system description), 2015. In *Automated Deduction – CADE-25*, pages 378–388, 2015.

D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, and W. P. Thurston. Word Processing in Groups, 1992. Jones and Bartlett, 1992.

G. Gardam. A counterexample to the unit conjecture for group rings, 2021. *Annals of Mathematics*, 194(3):967–979, 2021.

G. Gardam. Group ring units in SAT, 2022. In *SAT Competition 2022: Solver and Benchmark Descriptions*, 2022. Available at `https://satcompetition.github.io/2022/downloads/sc2022-proceedings-DRAFT.pdf`.

F. A. Garside. The braid group and other groups, 1969. *Quarterly Journal of Mathematics*, 20(1):235–254, 1969.

Joel Gibson, Geordie Williamson, and Oded Yacobi. 4-Strand Burau is Unfaithful Modulo 5. *Journal of Experimental Mathematics*, 1(1):36–50, 2025. doi: 10.56994/JXM.001.001.003.

T. Gowers. Group and semigroup puzzles and a possible Polymath project, 2026. *Gowers's Weblog*, March 20, 2026. Available at `https://gowers.wordpress.com/2026/03/20/group-and-semigroup-puzzles-and-a-possible-polymath-project/`.

E. R. Green. Graph products of groups, 1990. PhD thesis, University of Leeds, 1990.

J. Gryak, R. M. Haralick, and D. Kahrobaei. Solving the conjugacy decision problem via machine learning, 2020. *Experimental Mathematics*, 29(1):66–78, 2020. doi:10.1080/10586458.2018.1434704. arXiv:1705.10417.

S. Gukov, J. Halverson, F. Ruehle, and P. Sułkowski. Learning to unknot, 2021. *Machine Learning: Science and Technology*, 2(2):025035, 2021. doi:10.1088/2632-2153/abe91f. arXiv:2010.16263.

R. M. Haralick, A. D. Miasnikov, and A. G. Myasnikov. Pattern recognition approaches to solving combinatorial problems in free groups, 2004. In *Computational and Experimental Group Theory*, Contemporary Mathematics, vol. 349, pp. 197–213. American Mathematical Society, 2004. doi:10.1090/conm/349/06361. arXiv:math/0604205.

J. Hass and J. C. Lagarias. The number of Reidemeister moves needed for unknotting, 2001. *Journal of the American Mathematical Society*, 14(2):399–428, 2001.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset, 2021. In *NeurIPS Datasets and Benchmarks*, 2021.

Borys Holikov. Computing kernel of Burau representation B_4 modulo p using reinforcement learning. Presentation at the Nineteenth International Symposium on Artificial Intelligence and Mathematics (ISAIM 2026), 2026. URL `https://isaim2026.cs.ou.edu/program.html`. Fort Lauderdale, Florida, January 9, 2026.

D. F. Holt. KBMAG: Knuth–Bendix in Monoids and Automatic Groups, 1994. GAP package / public software documentation, 1994–present.

T. Hubert, R. Mehta, L. Sartran, M. Z. Horváth, G. Žužić, E. Wieser, A. Huang, J. Schrittwieser, et al. Olympiad-level formal mathematical reasoning with reinforcement learning, 2026. *Nature*, 651:607–613, 2026. doi:10.1038/s41586-025-09833-y.

J. E. Humphreys. Reflection Groups and Coxeter Groups, 1990. Cambridge University Press, 1990.

A. Jeż. Recompression: A simple and powerful technique for word equations, 2016. *Journal of the ACM*, 63(1):1–51, 2016.

H. Ju, G. Gao, J. Jiang, B. Wu, Z. Sun, L. Chen, Y. Wang, Y. Wang, Z. Wang, W. He, P. Wu, L. Xiao, R. Liu, B. Dai, and B. Dong. Automated conjecture resolution with formal verification, 2026. arXiv:2604.03789, 2026.

M. Kambites. Small overlap monoids I: The word problem, 2009. *Journal of Algebra*, 321(8):2187–2205, 2009.

I. Kapovich and A. Myasnikov. Stallings foldings and subgroups of free groups, 2002. *Journal of Algebra*, 248(2):608–668, 2002.

H. Kera, Y. Ishihara, Y. Kambe, T. Vaccon, and K. Yokoyama. Learning to compute Gröbner bases, 2024. In *Advances in Neural Information Processing Systems*, vol. 37, 2024. doi:10.52202/079017-1044.

D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras, 1970. In *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon, 1970.

A. Lisitsa, M. Salles, and A. Vernitski. An application of neural networks to a problem in knot theory and group theory: untangling braids, 2022. arXiv:2206.05373, 2022.

R. C. Lyndon and P. E. Schupp. Combinatorial Group Theory, 1977. Ergebnisse der Mathematik und ihrer Grenzgebiete 89. Springer, 1977.

G. S. Makanin. The problem of solvability of equations in a free semigroup, 1977. *Mathematics of the USSR-Sbornik*, 32(2):129–198, 1977.

M. Mastin. Links and planar diagram codes, 2013. arXiv:1309.3288, 2013.

S. McAleer, F. Agostinelli, A. Shmakov, and P. Baldi. Solving the rubik's cube without human knowledge, 2018. URL `https://arxiv.org/abs/1805.07470`.

I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, and M. Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models, 2025. In *International Conference on Learning Representations*, 2025. arXiv:2410.05229.

A. D. Myasnikov and R. M. Haralick. A hybrid search algorithm for the Whitehead minimization problem, 2006. arXiv:math/0604206, 2006.

D. Peifer, M. Stillman, and D. Halpern-Leistner. Learning selection strategies in Buchberger's algorithm, 2020. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, vol. 119, pp. 7575–7585. PMLR, 2020.

W. Plandowski. Satisfiability of word equations with constants is in PSPACE, 2004. *Journal of the ACM*, 51(3):483–496, 2004.

B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. R. Ruiz, J. S. Ellenberg, P. Kohli, et al. Mathematical discoveries from program search with large language models, 2024. *Nature*, 625:468–475, 2024.

D. Segal. Polycyclic Groups, 1983. Cambridge Tracts in Mathematics 82. Cambridge University Press, 1983.

A. Shehper, A. M. Medina-Mardones, L. Fagan, B. Lewandowski, A. Gruen, Y. Qiu, P. Kucharski, Z. Wang, and S. Gukov. What makes math problems hard for reinforcement learning: A case study, 2024. arXiv:2408.15332, 2024.

J. R. Stallings. Topology of finite graphs, 1983. *Inventiones Mathematicae*, 71:551–565, 1983.

J. Stillwell. The word problem and the isomorphism problem for groups, 1982. *Bulletin of the American Mathematical Society*, 6(1):33–56, 1982.

The mathlib Community. The Lean mathematical library, 2020. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 367–381, 2020.

T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong. Solving olympiad geometry without human demonstrations, 2024. *Nature*, 625:476–482, 2024.

J. H. C. Whitehead. On equivalent sets of elements in a free group, 1936. *Annals of Mathematics*, 37(4):782–800, 1936.

C. Wrathall. The word problem for free partially commutative groups, 1988. *Journal of Symbolic Computation*, 6(1):99–104, 1988.

K. Yang, A. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. Prenger, and A. Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4441469427094f8873d0fecb0c4e1cee-Abstract-Datasets_and_Benchmarks.html.

C. Zhang, A. Zhou, R. J. George, S. Gukov, and A. Anandkumar. AI-driven mathematical discovery for the Andrews–Curtis conjecture, 2025. OpenReview preprint / MATH-AI 2025 poster, 2025.

A Related work and evaluation roles

Table 4 summarizes the ML/RL literature most closely related to each WordBench family and clarifies the role of the family within our evaluation. Prior work ranges from learning methods applied directly to the same algebraic problem to methods developed for closely related objects or search settings. The evaluation role column distinguishes how each family is defined as a benchmark task in WordBench.

Table 4: Prior ML/RL work most closely related to each of the 24 WordBench families and the role of each family in our evaluation.

Family	Related ML/RL work	Evaluation role
Andrews–Curtis trivialization	Specialized RL and LLM/RL/Lean search has been studied directly (Shehper et al., 2024; Zhang et al., 2025).	Positive traces are checkable; timeouts are not negative theorems (Bridson, 2015; Shehper et al., 2024).
Artin–Tits word problem	No direct ML/RL baseline found for this slice.	All-YES rewrite-to-empty slice with exact replay. Decidable by construction; a restricted puzzle related to the open Artin–Tits word problem (Gowers, 2026).
Baumslag–Solitar BS(1,2)	Adjacent ML work studies conjugacy in non-free groups (Gryak et al., 2020).	Noncommutative rewrites with exact exponent effects (Baumslag & Solitar, 1962).
Braid word rewriting	Related neural and RL methods have been applied to braid and knot problems (Lisitsa et al., 2022; Gukov et al., 2021).	Equality-preserving Artin rewrites with exact noncommutative replay (Birman, 1974).
Burau B_4 search	Closely related learning-based search for Burau B_4 has been studied using NNs (Gibson et al., 2025) and RL (Holikov, 2026).	Constructive search over braid words whose Burau matrix matches a hidden target matrix. Candidate solutions are checked by exact Laurent-polynomial matrix equality.
Burau matrix equality	Closely related learning-based search for Burau B_4 has been studied using NNs (Gibson et al., 2025) and RL (Holikov, 2026).	Hidden-label identity classification. The agent sees the braid word and may apply local Artin rewrites before submitting yes or no for identity classification. Success is determined by agreement with the stored ground-truth label.
Coxeter type-A rewriting	Adjacent finite-group sequence learning.	Finite analogue with involutions, braid moves, and commutations (Humphreys, 1990).
Finite permutation action	Adjacent finite-group learning.	Clean finite-state bridge between word problems and RL planning.
Free-group conjugacy	Adjacent ML for conjugacy decision (Gryak et al., 2020).	Requires an explicit conjugator witness, not just a YES label.
Free-group subgroup membership	No direct ML/RL baseline found.	Synthesizes subgroup-generator expressions checked by exact expansion (Kapovich & Myasnikov, 2002).
Free-group word	Adjacent ML for free-group recognition (Haralick et al., 2004).	Stack-like cancellation and exact normal-form comparison.
Graph-product word	No direct ML/RL baseline found.	Graph-controlled syllable commutation and vertex-group reduction.
Gröbner normal form	Yes: RL and Transformers for Gröbner computation (Peifer et al., 2020; Kera et al., 2024).	Exact polynomial arithmetic and certified normal-form reduction (Cox et al., 2015; Bachmann & Gräbe, 2000).
Heisenberg collection	No direct ML/RL baseline found.	Local collection while tracking central correction terms (Segal, 1983).
Kaplansky group rings	No direct RL baseline found; SAT/search is adjacent.	Constructive zero-divisor witness synthesis: write a coefficient pair whose product is zero, checked by exact group-ring multiplication. Decidable by construction; related to Kaplansky's conjecture (Gardam, 2021; 2022).
KBMAG rewriting	No direct ML/RL baseline found.	Rule-position search in convergent rewriting systems (Holt, 1994).
Knot equivalence	Yes: RL for unknotting and simplification (Gukov et al., 2021; Applebaum et al., 2025).	Replayed Reidemeister/Markov-style traces between diagrams (Hass & Lagarias, 2001; Mastin, 2013; Burton, 2004).
Makanin word equations	Yes: GNN-guided word-equation solving (Abdulla et al., 2024).	Produces exact substitutions for both sides (Makanin, 1977; Plandowski, 2004).
Path-RAAG word	No direct ML/RL baseline found.	Neutral commutations can hide useful cancellations (Wrathall, 1988; Charney, 2007).
Rubik's cube	Yes: specialized deep RL plus search solves it (Agostinelli et al., 2019).	Verifier-facing planning stress test; not an RL-unsolved claim.
Small-cancellation word	No direct ML/RL baseline found.	Find useful length-reducing Dehn pieces (Lyndon & Schupp, 1977).
Small-overlap monoid word	No direct ML/RL baseline found.	Exact monoid derivations without implicit inverses or cancellation.
Thompson's group F	No direct ML/RL baseline found.	Exact planning in an infinite group via finite tree-pair states (Cannon et al., 1996).
Whitehead free group	Adjacent ML for Whitehead-minimal words (Haralick et al., 2004).	Searches over automorphism traces rather than local edits (Myasnikov & Haralick, 2006).

B FAMILY INTERFACES AND VERIFIER CONTRACTS

Table 5 gives details on the solver-facing interface for each family, specifying what a solver must return and how acceptance is checked.

Table 5: Family-level output and verification contracts.

Family	Admissible solver output	Verifier acceptance condition	Regime
Andrews–Curtis trivialization	AC move trace	replay relator inversions, multiplications, conjugations, and reductions to the trivial presentation	open
Artin–Tits word problem	Artin-tile rewrite trace	replay Artin tile moves and cancellations to the empty word	decidable*
Baumslag–Solitar BS(1,2)	group-preserving rewrite trace	replay free cancellations and legal instances of $tat^{-1} = a^2$	decidable
Braid word rewriting	Artin-relation rewrite trace	replay braid, commutation, and cancellation moves exactly	decidable
Burau B_4 search	Artin-relation rewrite trace ending in a submitted braid word	replay braid rewrites, generator appends, and deletions, then accept if the reduced Burau matrix equals that of the hidden target word	decidable
Burau matrix equality	Artin-relation rewrite trace ending in a YES/NO decision	replay local Artin rewrites, then accept if the submitted answer agrees with whether the reduced Burau image is I	decidable
Burau matrix equality	YES/NO decision	submitted answer agrees with whether the reduced Burau image is I	decidable
Coxeter type-A rewriting	Coxeter rewrite trace	replay involution, braid, and commutation relations or compare permutations	decidable
Finite permutation action	generator action sequence	replay actions and check that the final state is the target	decidable
Free-group conjugacy	conjugator g	verify $g^{-1}ug =_F v$ after free reduction	decidable
Free-group subgroup membership	signed subgroup-generator expression	expand the expression and compare with the target word	decidable
Free-group word	local cancellation trace	replay deletions and compare freely reduced words	decidable
Graph-product word	normal-form or commutation trace	check within-vertex reductions and edge-licensed syllable swaps	decidable
Gröbner normal form	reduction trace or claimed remainder	perform exact polynomial division and compare the final remainder	decidable
Heisenberg collection	collection trace or normal-form triple	check local collection rules or exact integer triple equality	decidable
Kaplansky group rings	explicit group-ring witness pair	multiply exactly and check nonzero witnesses with zero product	decidable*
KBMAG rewriting	convergent rewrite trace	check rule id, position, substring match, and target normal form	decidable
Knot equivalence	Reidemeister/Markov-style move trace	replay moves on the diagram encoding and compare the final diagram	decidable
Makanin word equations	variable assignment	substitute all variables and compare the two instantiated strings	decidable
Path-RAAG word	commutation/cancellation trace	replay swaps licensed by the path graph and inverse cancellations	decidable
Rubik's cube	cube move sequence	replay face turns and check the solved state	decidable
Small-cancellation word	Dehn-reduction trace	check each long-piece replacement against the declared relator set	decidable
Small-overlap monoid word	presentation-relation rewrite trace	check literal relation applications with no implicit inverses or cancellation	decidable
Thompson's group F	generator-word trace	multiply in exact tree-pair representation and check identity	decidable

continued on next page

Family	Admissible solver output	Verifier acceptance condition	Regime
Whitehead free group	automorphism trace	apply each Whitehead automorphism and compare with the target	decidable

C DATASET GENERATION CONFIGURATIONS

Table 6 summarizes the configurations used to construct the released WordBench datasets. We report the number of unique released instances and the generation parameters that can be varied to produce additional or differently scaled instances.

Table 6: Generation configurations and details for the released WordBench datasets. Adjustable parameters indicate how additional instances can be generated.

Family	Released data configuration	Unique instances	Adjustable parameters
Andrews–Curtis trivialization	Miller–Schupp presentations $\mathrm{MS}(n, w)$ with $n \leq 7$ and $\lvert w\rvert \leq 7$	1190	presentation bounds, relator length
Artin–Tits word problem	Fixed 4-generator presentation; positive instances generated by 48 relation-preserving rewrite steps from the empty word	1000	presentation, word length, scramble depth
Baumslag–Solitar BS(1,2)	$\mathrm{BS}(1, 2)$ identity words generated using defining-relation rewrites and free cancellations	500	word length, rewrite depth
Braid word rewriting	B_4 targets of length 3–9, transformed by 64 braid-relation, commutation, and cancellation moves	500	strand count, word length, scramble depth
Burau B_4 search	B_4 start words of length at most 32; targets generated by extending the start word by 1–32 generators	500	start length, target-extension length
Burau matrix equality	Balanced B_4 equality dataset; positive words generated from conjugated Artin relations and negative words matched by length	500	word length, relation/conjugator complexity, class balance
Coxeter type-A rewriting	Type A_7 targets of length 3–9, transformed by 8 Coxeter-relation rewrites	482	rank, word length, scramble depth
Finite permutation action	Reachable nonidentity targets in S_5 under a fixed generating set	59	permutation degree, generator set
Free group conjugacy	F_2 words of length 1–15 conjugated by hidden words of length 1–7	500	rank, word length, conjugator length
Free group subgroup membership	F_2 with two subgroup generators of length 1–15; witnesses use 1–7 subgroup-generator factors	500	rank, subgroup size, generator length, witness length
Free group word	F_2 targets of length 0–8 with 1–6 inserted cancellable pairs	500	rank, word length, number of insertions
Graph product word	Four-generator graph products over three commutation graphs; targets of length 2–62 with 1–32 inverse mutations	500	graph structure, word length, mutation depth
Gröbner normal form	Polynomial systems with up to 4 variables and 10 terms, spanning multiple basis structures, term orders, and difficulty levels	1000	variables, degree, polynomial size, term order
Heisenberg collection	$H(\mathbb{Z})$ words of length 2–7 transformed by 8 relation-preserving scramble steps	500	word length, scramble depth
Kaplansky group rings	Zero-divisor search in $\mathbb{F}_2[C_n]$ for $2 \leq n \leq 31$	30	group order n
KBMAG rewriting	Finite convergent rewriting systems for D_8 and Q_8 with start words of length 6–15	500	rewriting system, word length
Knot equivalence	Four-strand braid words of length 24 transformed by 12 knot-preserving moves	1000	strand count, braid length, transformation depth
Makanin word equations	Equations over variables X, Y and alphabet $\{a, b\}$, with assignments of length 1–5 and patterns of length 2–8	500	variable length, pattern length, equation size
RAAG path word	Path RAAG with 65 generators; one canonical instance for each even word length from 2 to 128	64	path size, word length
Rubik's cube	1000 distance-20 states from the Cube20 benchmark	1000	scramble depth for synthetically generated instances
Small cancellation word	Three generators and two length-24 relators satisfying $C'(1/6)$; instances built from products of two conjugated relators	500	generator count, relator count/length, conjugator length
Small overlap monoid word	Six-letter alphabet with two fixed relations; equivalent targets generated by bounded relation rewrites	500	relation set, word length, rewrite depth
Thompson F search	Thompson's group F words scrambled to length 10–20 with identity target	499	scramble length
Whitehead free group	F_2 words transformed by 24 random Whitehead automorphisms	500	rank, word length, scramble depth

D PPO ENVIRONMENT SPECIFICATIONS

Table 7 lists the state, action, reward, and horizon interfaces used for the PPO sweeps. Rewrite tasks expose finite algebraic or combinatorial transformations of the current state, while witness-construction tasks require the agent to synthesize an explicit certificate accepted by the verifier. Observations are bounded symbolic encodings, actions are finite family-specific move templates, and rewards are used only for training. Reported benchmark credit always comes from exact verifier acceptance.

Table 7: Implemented PPO environments grouped by task type.

Family	Observation encoding	Action space	Reward / horizon
Rewrite			
Andrews–Curtis trivialization	padded relator array	finite AC move templates: invert, multiply, conjugate, reduce	terminal $+1000$; reduced-length shaping; horizon 64
Artin–Tits word problem	pad(current)‖pad(target)	tile-rule-id $\times$ position over cyclic Artin tiles + cancellations	solve $+5$; step -0.01; invalid -0.05; horizon 128
Baumslag–Solitar BS(1,2)	pad(word)	$(\mathrm{op}, \mathrm{pos})$ over local relation/cancel moves	solve $+5$; step -0.01; invalid -0.05; horizon 64
Braid word rewriting	pad(current)‖pad(target)	rule-id $\times$ position over Artin/cancel rules	solve $+5$; step -0.01; invalid -0.05; horizon 128
Burau B_4 search	pad(current)‖ target Burau features (reduced 3×3 matrix at $t \in \{2, 3, -1\}$ mod $2^{31} - 1$)	rewrite $\times$ position over Artin/cancel rules $+$ 6 generator appends $+$ backspace $+$ submit	solve $+5$; fail -1; step -0.01; invalid -0.05; horizon 64
Burau matrix equality	pad(word)	rewrite $\times$ position over Artin/cancel rules $+$ SUBMIT_YES $+$ SUBMIT_NO	solve $+5$; fail -1; step -0.01; invalid -0.05; horizon 64
Coxeter type-A rewriting	pad(current)‖pad(target)	cancel/commute/braid/insert-pair moves	solve $+5$; step -0.01; invalid -0.05; horizon 64
Finite permutation action	current‖goal states	apply generators and inverses	solve $+10$; step -0.01; horizon 24
Free-group word	pad(current)‖pad(target) (2×32)	choose adjacent position; cancel inverse pair	solve $+5$; step -0.01; invalid -0.05; horizon 64
Graph-product word	pad(current)‖pad(target)‖graph-id	adjacent swap/cancel at position	solve $+5$; step -0.01; invalid -0.05; horizon 64
Gröbner normal form	system-id ‖ poly(current) ‖ poly(target)	reducer-id $\times$ term-slot or submit	solve $+5$; step -0.01; invalid -0.05; horizon 96
Heisenberg collection	pad(current)‖pad(target)	rule-id $\times$ position over collection rules	solve $+5$; step -0.01; invalid -0.05; horizon 64
KBMAG rewriting	pad(current)‖pad(target)‖system-id	rule-id $\times$ position in a convergent system	solve $+5$; step -0.01; invalid -0.05; horizon 64
Knot equivalence	pad(current)‖pad(target)	braid/commute/rotate at position; submit	solve $+5$; step -0.01; invalid -0.05; horizon 128
Path-RAAG word	pad(current)‖pad(target)	adjacent swap/cancel at position	solve $+5$; step -0.01; horizon 64
Rubik's cube	one-hot facelets for a $3 \times 3 \times 3$ cube	18 outer face turns	cost -1/move; horizon 200
Small-cancellation word	pad(relators)‖pad(word)	Dehn-reduce(relator-id,pos) or submit	solve $+5$; step -0.01; invalid -0.05; horizon 128
Small-overlap monoid word	pad(current)‖pad(target)‖relset-id	apply relation at (position,rule) or submit	solve $+5$; step -0.01; invalid -0.05; horizon 64
Thompson's group F	leaf-depth / tree-pair features	right-multiply by $x_0^{\pm 1}, x_1^{\pm 1}$	solve $+5$; step -0.01; invalid -0.05; horizon 20
Witness construction			
Free-group conjugacy	pad(u)‖pad(v)‖pad(conjugator)	append letters, backspace, or submit	solve $+10$; step -0.01; horizon 64
Free-group subgroup membership	pad($h_1, \ldots, h_k$)‖pad(target)‖pad(expr)	append subgroup generators, backspace, submit	solve $+10$; step -0.01; horizon 64
Kaplansky group rings	n, cursor ‖ coeff buffers	write-bit, backspace, submit	solve $+10$; step -0.01; invalid -0.05; horizon 64
Makanin word equations	template-id + variable buffers	edit variables or submit assignment	solve $+10$; step -0.01; invalid -0.05; horizon 32
Whitehead free group	pad(current)‖pad(target)	choose one of finitely many Whitehead moves	solve $+5$; step -0.01; invalid -0.05; horizon 32

E REPRODUCIBILITY AND COMPUTE DETAILS

The released codebase includes scripts and README instructions for regenerating the PPO runs, collecting verifier-accepted solutions, and rebuilding the result tables. The reproduction package contains the benchmark manifests, `WordBench-Lit` corpus, family configuration files, run mani-

fests, collected JSON summaries, and verified solution records. We submit the codebase as supplementary materials here: `https://anonymous.4open.science/r/WordBench-4555`

E.1 PPO

The PPO sweep uses the 24-family WordBench manifest. Each family is trained for 10^6 environment steps at seeds $1, 2, 3, 4, 5$ with an MLP encoder. Jobs are scheduled across available CUDA devices, with one worker per GPU. Our reference run used four NVIDIA A100-SXM4-80GB GPUs on a Linux workstation. Wall-clock time varies by family because verifier and transition costs differ: local rewrite environments are fast, while exact polynomial arithmetic, Burau evaluation, and other algebra-heavy environments are slower. All reported PPO entries are reconstructed from saved run directories and replay-checked solution files.

E.2 LLMs

Output format and verification. Models were asked to return a JSON object containing integer action indices:

```
{"actions":[1,2]}
```

This was a prompted format rather than schema-constrained decoding. The parser extracted the last decodable object containing an `actions` field and checked integer types, action ranges, and the action horizon. Separate reasoning blocks were excluded from answer parsing.

The evaluator reconstructed the frozen Python environment from the saved initial state and replayed the actions. Success required the exact family-specific goal and any required submission action. Invalid outputs and provider failures counted as unsuccessful attempts. These scores measure Python verifier acceptance; individual submissions were not compiled as Lean proofs.

Recorded usage and estimated cost. We estimate inference cost from saved provider token counts and public base rates checked on 25 September 2026. Input counts include repeated conversation input during continuation and are normalized to a no-cache equivalent. For each configuration, we compute

$$\widehat{C} = \frac{T_{\text{input}} p_{\text{input}} + T_{\text{output}} p_{\text{output}}}{10^6},$$

where prices are in USD per million tokens.

Table 8: Recorded token usage and estimated public-rate cost. Token counts are in millions.

Model	Input M	Output M	$/M input	$/M output	Est. USD
Astra	25.54	73.48	11.00	55.00	4,322
Opus 5.5	26.35	149.28	4.00	20.00	3,091
Qwen3 235B	84.42	80.26	0.22	0.88	89
DeepSeek V3.2	211.00	178.12	0.62	1.85	460
Kimi K3	22.69	178.13	3.00	15.00	2,740
Total					10,703

The estimate uses Bedrock's US geographic base rate for Astra, global rates for Opus 5.5 and Kimi, and US West (Oregon) standard rates for Qwen and DeepSeek.

Examples of accepted outputs. For Andrews–Curtis instance `01168`, Astra received the initial relators

$$(R_1, R_2) = (a^{-1}b^7ab^{-8}, a^{-2}b^{-1}ab^4)$$

and returned the following 64-action solution:

```
{"actions":[8,8,10,9,0,5,5,0,8,11,9,7,3,8,1,5,
6,6,1,7,7,4,1,4,7,7,7,7,4,1,7,7,4,1,7,7,
4,1,7,7,4,1,7,4,7,7,7,4,3,5,6,11,2,0,
0,0,0,0,0,0,0,0,2,2]}
```

The final portion of the verified execution reveals a sequence of cancellations. Write $c = a^{-1}b^2$, so that $c^{-1} = b^{-2}a$. After the first 52 actions, the relators are

$$(R_1, R_2) = (bc^9, c^{-10}b^{-1}).$$

Action 2 applies $R_2 \leftarrow R_2R_1$. Free reduction gives

$$R_2 = c^{-10}b^{-1}bc^9 = c^{-1}.$$

The model then applies action 0 nine times. Each application multiplies R_1 by $R_2 = c^{-1}$, cancelling one suffix:

$$R_1 = bc^k \quad \longmapsto \quad bc^kc^{-1} = bc^{k-1}.$$

Every intermediate state from step 52 to termination is shown below; action IDs refer to the supplied environment interface.

Step	Action ID	R_1	R_2
52	–	bc^9	$c^{-10}b^{-1}$
53	2	bc^9	c^{-1}
54	0	bc^8	c^{-1}
55	0	bc^7	c^{-1}
56	0	bc^6	c^{-1}
57	0	bc^5	c^{-1}
58	0	bc^4	c^{-1}
59	0	bc^3	c^{-1}
60	0	bc^2	c^{-1}
61	0	bc	c^{-1}
62	0	b	$b^{-2}a$
63	2	b	$b^{-1}a$
64	2	b	a

Once $R_1 = b$, two further applications of action 2 complete the solution. Each multiplies R_2 by b, followed by the environment's cyclic reduction:

$$b^{-2}a \longmapsto b^{-2}ab \longmapsto b^{-1}a \longmapsto b^{-1}ab \longmapsto a.$$

The final relators (b, a) are distinct singleton generators, satisfying the goal. An independent checker reproduced every intermediate state. The attempt used 34,323 reported output tokens in one response. The explanations above describe verified execution.

F LEAN VERIFICATION AND PROOF ARTIFACTS

WordBench uses Lean to specify the benchmark at the level of mathematical semantics. For each family, the formalization defines the relevant objects—words, presentations, rewrite systems, normal forms, transition systems, moves, and admissible outputs—and then states the verifier predicate that connects a concrete solver output to the intended mathematical claim.

Example: Andrews–Curtis semantics. The formalization is mathematical rather than parser-level. The following excerpt illustrates the theory layer for Andrews–Curtis. Presentations are represented as tuples of free-group relators; AC moves are constructors of a relation on presentations; and `ACEquivalent` is the equivalence closure generated by those moves. A concrete solver trace therefore proves equivalence by composing move constructors rather than by appealing to an external oracle.

```
import Mathlib
import Problems.Core.Presentation
import Problems.AndrewsCurtis.PresentationGroup

def x : FreeGroup (Fin 2) := FreeGroup.mk [(0, true)]
def y : FreeGroup (Fin 2) := FreeGroup.mk [(1, true)]
```

```
inductive ACMove : Presentation (Fin 2) -> Presentation (Fin 2) -> Prop
  | h1 (p : Presentation (Fin 2)) :
    ACMove p
      (Presentation.mk
        (p.relations.set 1
          ((p.relations.getD 1 1) * (p.relations.getD 0 1))))
  | h2 (p : Presentation (Fin 2)) :
    ACMove p
      (Presentation.mk
        (p.relations.set 0
          ((p.relations.getD 0 1) * (p.relations.getD 1 1)^-1)))
  | h5 (p : Presentation (Fin 2)) :
    ACMove p
      (Presentation.mk
        (p.relations.set 1
          (x^-1 * (p.relations.getD 1 1) * x)))
  -- The full file includes the remaining AC generator moves.

def ACEquivalent : Presentation (Fin 2) -> Presentation (Fin 2) -> Prop
    :=
  Relation.EqvGen ACMove
```

The theory also records the target notion of triviality and separates the open Andrews–Curtis conjecture from concrete certificate proofs. In the benchmark, we do not assume the conjecture; we check finite move sequences and, when available, independent obstructions. The theorem statement below is intentionally shown as the open conjectural statement rather than as a used benchmark lemma; the following concrete proof artifact is what the verifier-generated proof scripts use.

```
def trivial_presentation : Presentation (Fin 2) :=
  Presentation.mk [x, y]

def example_presentation : Presentation (Fin 2) :=
  Presentation.mk [x * y, x * y * x^-1]

def totalLength {alpha : Type*} [DecidableEq alpha]
    (p : Presentation alpha) : Nat :=
  (p.relations.map (fun r => FreeGroup.norm r)).sum

def isTrivial (p : Presentation (Fin 2)) : Bool :=
  totalLength p = 2

def ACTrivial (p : Presentation (Fin 2)) : Prop :=
  exists t : Presentation (Fin 2), isTrivial t = true /\ ACEquivalent p
    t

theorem AC_conjecture_statement (p : Presentation (Fin 2)) :
    balanced p ->
    AndrewCurtisConjecture.presentsTrivialGroupAny p ->
    ACEquivalent p trivial_presentation := by
  sorry
```

Concrete solver outputs become Lean proofs by chaining these move constructors. The example below is a small AC proof artifact: it composes two checked moves to obtain a proof term in the formal theory of presentations and AC equivalence.

```
theorem example_is_AC_trivial :
    ACEquivalent example_presentation trivial_presentation := by
  let rels1 : List (FreeGroup (Fin 2)) := [x * y, y]

  have step1 :
      ACMove example_presentation (Presentation.mk rels1) := by
    convert ACMove.h5 example_presentation
    decide
```

```
  have step2 :
      ACMove (Presentation.mk rels1) trivial_presentation := by
    convert ACMove.h2 (Presentation.mk rels1)
    decide

  have eq1 :
      ACEquivalent example_presentation (Presentation.mk rels1) :=
    Relation.EqvGen.rel _ _ step1
  have eq2 :
      ACEquivalent (Presentation.mk rels1) trivial_presentation :=
    Relation.EqvGen.rel _ _ step2

  exact Relation.EqvGen.trans
    example_presentation (Presentation.mk rels1) trivial_presentation
    eq1 eq2
```

G PROBLEM FAMILY CARDS

These cards describe the 24 WordBench tasks implemented by the current environments. A mathematical decision problem and a bounded search task are different: reaching a verified goal establishes a positive result, whereas exhausting an episode does not establish a mathematical negative. Unless explicitly stated otherwise, every example below is a small illustration, not an identified row of the released dataset. Signed generator indices use zero only for padding; ε denotes the empty word. The output of a rewrite or planning task is its action sequence; witness tasks return an assignment or word, and classification returns a Boolean answer. Invalid actions usually leave the state unchanged and incur a penalty; they are not successful rewrite steps. A trace may contain such attempts and still eventually succeed.

G.1 FREE-GROUP WORD PROBLEM

Mathematical problem. For a free group $F(X)$, a word is a string over $X \cup X^{-1}$. The only reductions are adjacent inverse cancellations $xx^{-1} \to \varepsilon$ and $x^{-1}x \to \varepsilon$. The cancellation system is terminating and confluent, so every word has a unique freely reduced normal form $\rho(w)$. The equality question is therefore

$$u =_F v \quad \Longleftrightarrow \quad \rho(u) = \rho(v) \quad \Longleftrightarrow \quad \rho(uv^{-1}) = \varepsilon.$$

This reduction is also used by the conjugacy and subgroup tasks (Lyndon & Schupp, 1977; Stillwell, 1982).

Implemented task. The observation concatenates padded current and target words. An action selects an adjacent position and deletes the pair only if the letters are inverses. The target is freely reduced at decoding. The task is to reach that target using deletions.

Acceptance. Executable success is literal equality of the current word and the decoded target after a step. Replaying the deletion trace gives a free-group equality certificate. This environment does not accept a NO answer; differing reduced words provide a separate mathematical negative.

Accepted illustration. In $F(a, b)$, let $u = aa^{-1}ba$ and $v = ba$. Deleting the first two letters gives $aa^{-1}ba \to ba$, so the one-step deletion trace proves $u =_F v$.

Negative / rejected illustration. Let $u = ab$ and $v = ba$ in $F(a, b)$. Both words are freely reduced and their normal forms are different strings. Since the free group has no commutation relation between a and b, the instance is NO.

G.2 FREE-GROUP CONJUGACY

Mathematical problem. The conjugacy problem asks whether two words are the same up to a change of basepoint in the Cayley graph:

$$\exists g \in F(X) \quad gug^{-1} =_F v.$$

In free groups, the classical decision procedure freely reduces and cyclically reduces u and v and then compares cyclic rotations of the reduced cores. WordBench uses the constructive form of the problem: a system must produce the conjugator g, not merely say that two cyclic words look compatible (Lyndon & Schupp, 1977).

Implemented task. Observation includes u, v, and a bounded candidate buffer for g. Actions append a generator or inverse to the buffer, delete the last symbol, or submit the current buffer. The output is the submitted conjugator.

Acceptance. The submitted buffer g is accepted exactly when $\rho(gug^{-1}) = \rho(v)$. This is the implementation convention; the inverse conjugator would be needed for the convention $g^{-1}ug = v$. The environment has an explicit submit action, although its diagnostic `is_solved` flag also reports a zero residual before submission.

Accepted illustration. Let $u = a$ and $v = b^{-1}ab$ in $F(a, b)$. The certificate $g = b^{-1}$ is accepted because $gug^{-1} = b^{-1}ab = v$.

Negative / rejected illustration. Let $u = a$ and $v = ab$. The cyclically reduced lengths are 1 and 2. Conjugacy preserves cyclically reduced length in a free group, so no conjugator can exist.

G.3 SUBGROUP MEMBERSHIP IN A FREE GROUP

Mathematical problem. Given $H = \langle h_1, \ldots, h_k \rangle \leq F(X)$ and w, subgroup membership asks whether $w \in H$. Stallings folding decides membership; WordBench instead searches for a signed expression in the supplied generators whose free reduction is w (Stallings, 1983; Kapovich & Myasnikov, 2002). This uses the same free reduction as the word and conjugacy tasks.

Implemented task. The state contains the subgroup generators, the target word, and a candidate expression buffer. An action appends h_i or h_i^{-1}, backspaces, or submits. Thus the policy is not directly editing the ambient word over X; it is writing a program over the supplied generator list whose expansion should reduce to w.

Acceptance. On submission, the environment expands the signed generator-index buffer to $h_{i_1}^{\epsilon_1} \cdots h_{i_m}^{\epsilon_m}$ and compares its free reduction with that of the target. A failed witness or timeout is not a nonmembership certificate; the parity obstruction below is a separate mathematical illustration.

Accepted illustration. In $F(a, b)$, let $H = \langle a^2, b \rangle$ and let $w = a^2b$. The expression $(a^2) \cdot b$ expands and freely reduces to a^2b, so it is a valid membership certificate.

Negative / rejected illustration. With the same subgroup $H = \langle a^2, b \rangle$, the target $w = a$ is not in H. In the abelianization map $F(a, b) \to \mathbb{Z}^2$, every element of H has even a-coordinate, whereas a has odd a-coordinate.

G.4 BAUMSLAG–SOLITAR $\mathrm{BS}(1,2)$ WORD PROBLEM

Mathematical problem. The group

$$\mathrm{BS}(1,2) = \langle a, t \mid tat^{-1} = a^2 \rangle$$

is a compact one-relator example where equality is decidable but local reasoning is less transparent than in a free group. The defining relation says that conjugating a by t doubles its exponent. A useful exact model sends a to $(1,0)$ and t to $(0,1)$ in the semidirect product $\mathbb{Z}[1/2] \rtimes \mathbb{Z}$ with multiplication $(q,n)(r,m) = (q + 2^n r, n + m)$ (Baumslag & Solitar, 1962).

Implemented task. The observation is a padded word over $a^{\pm 1}, t^{\pm 1}$; the goal is the empty word. Actions select an operation and position among seven templates: cancellation, conjugation contraction/expansion, and four signed t-pushing rules. Valid results are freely reduced, subject to the word-length cap.

Acceptance. The action sequence succeeds when the current word becomes empty. The separate exact evaluator computes the dyadic-rational/integer pair (q,n) and can independently validate identity. The environment offers neither arbitrary target submission nor a submitted pair (q,n).

Accepted illustration. The word $tat^{-1}a^{-2}$ is trivial. One legal rewrite gives $tat^{-1}a^{-2} \to a^2a^{-2}$, and the remaining letters freely cancel.

Negative / rejected illustration. The word a is not trivial. Under the exact semidirect-product model it evaluates to $(1,0)$, not $(0,0)$, so no valid rewrite trace can reduce it to ε.

G.5 BRAID WORD REWRITING

Mathematical problem. The braid group B_n has generators $\sigma_1, \ldots, \sigma_{n-1}$ and Artin relations

$$\sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1}, \qquad \sigma_i\sigma_j = \sigma_j\sigma_i \quad (|i-j| > 1),$$

together with inverse cancellations. Equality in B_n is decidable, but local rewrite search has many flat directions: a relation can preserve the braid while increasing, decreasing, or barely changing string similarity to the target (Artin, 1947; Birman, 1974; Garside, 1969; Epstein et al., 1992).

Implemented task. The observation is the padded current word followed by the target. Actions select a rule id and position: inverse cancellation or insertion, far commutation with signed letters, or a same-sign three-letter braid relation in either direction. Insertions must fit the length cap.

Acceptance. The environment applies a rule only at a matching substring and succeeds on literal current–target word equality. It does not replace this endpoint test by a canonical braid comparison. The trace certifies equality, but bounded failure does not decide braid inequality. The same braid relations appear in the Burau and knot tasks.

Accepted illustration. In B_3, $\sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2$ by one Artin braid move. Thus the one-step trace from the left word to the right word is accepted.

Negative / rejected illustration. The words σ_1 and σ_2 are not equal in B_3. Their images in the symmetric group are the distinct transpositions $(1\ 2)$ and $(2\ 3)$, so no braid-rewrite trace can identify them.

G.6 ARTIN–TITS WORD PROBLEM

Mathematical problem. Artin–Tits groups impose alternating relations of length m_{ij}. The fixed WordBench presentation has generators a, b, c, d, $m(c, d) = 2$, and all other off-diagonal entries 3. It generalizes braid rewriting. The selected generated corpus consists of positive rewrite-to-empty instances; its name "mixed is not a YES/NO balance claim. No general word-problem algorithm is asserted for this slice.

Implemented task. The observation concatenates the current and target words (the generated corpus uses an empty target); there is no system-id feature. Actions select a rule and position. Rules are derived from cyclic cuts of relator tiles and include length-changing replacements, full-tile insertion/deletion, and inverse-pair insertion/deletion, within the length cap.

Acceptance. The executable endpoint is literal equality with the target. Each applied rule is from the fixed tile catalog. A successful trace proves equality in the presentation; an invalid attempted commutation does not change the state. Lean separately proves that the tile certificate preserves evaluation in the presented group.

Accepted illustration. If c and d commute in the chosen presentation, then $cdc^{-1}d^{-1}$ is trivial. A valid trace swaps cd to dc and then freely cancels $dcc^{-1}d^{-1}$ to the empty word.

Rejected illustration. For a pair with $m(a, b) = 3$, the valid Artin relation is $aba \leftrightarrow bab$, not $ab \leftrightarrow ba$. An attempted step treating a and b as commuting is invalid and supplies no equality certificate.

Budget-limited search. When no rewrite-to-empty trace is found within the search budget, the row is reported as unsolved under that budget. This is not a proof that no trace exists in the broader Artin–Tits setting.

G.7 COXETER TYPE-A REWRITING

Mathematical problem. The type-A Coxeter group is the symmetric group S_n with adjacent transpositions $s_i = (i\ i+1)$ and relations

$$s_i^2 = 1, \qquad s_i s_j = s_j s_i\ (|i-j| > 1), \qquad s_i s_{i+1} s_i = s_{i+1} s_i s_{i+1}.$$

It looks similar to braid rewriting, but the involution relation $s_i^2 = 1$ collapses the group to a finite one. Exact permutation evaluation therefore gives a simple independent correctness check (Humphreys, 1990).

Implemented task. The observation contains padded current and target words in adjacent transpositions. Actions delete $s_i s_i$, commute distant letters, perform a three-letter braid move, or insert $s_i s_i$ at a position when it fits. Inverse signs are immaterial because each generator is an involution.

Acceptance. Executable success requires literal target-word equality after a move. Permutation evaluation gives an independent mathematical equality test, but is not an alternate submission mode. As in braid rewriting, the finite move budget can prevent reaching an equal target.

Accepted illustration. The word $s_1 s_1$ is equal to the identity because $s_1^2 = 1$. A single deletion certifies $(u, v) = (s_1 s_1, \varepsilon)$.

Negative / rejected illustration. The word s_1 is not equal to the identity: as a permutation it swaps positions 1 and 2. The verifier therefore rejects any trace claiming $s_1 = \varepsilon$.

G.8 HEISENBERG COLLECTION

Mathematical problem. The discrete Heisenberg group is a basic nilpotent group. We use generators x, y, z with z central and the collection convention $xy = yxz$, equivalently

$$yx = xyz^{-1}.$$

Every element has a unique collected normal form $x^a y^b z^c$. Equality is therefore exact integer-triple equality, but producing the normal form requires remembering the central correction caused by swapping x and y (Segal, 1983).

Implemented task. The observation contains current and target words over $x^{\pm 1}, y^{\pm 1}, z^{\pm 1}$. Actions select a rule and position: inverse cancellation, moving a central z letter right, or moving x left across y with the signed central correction. This extends commutation tasks with an extra central exponent.

Acceptance. The output is a collection trace; success requires literal equality with the supplied target. The exact integer-triple evaluator is an independent check, not a submitted-output format. In particular, $yx \mapsto xy$ without z^{-1} is not an available collection rule.

Accepted illustration. The pair $(u, v) = (yx, xyz^{-1})$ is YES under the stated convention: one collection step rewrites yx to xyz^{-1}.

Negative / rejected illustration. The pair $(u, v) = (x, y)$ is NO. The normal forms have triples $(1, 0, 0)$ and $(0, 1, 0)$, which are different.

G.9 KBMAG-STYLE CONVERGENT REWRITING

Mathematical problem. A finite convergent rewrite system gives a unique normal form, as in Knuth–Bendix workflows (Knuth & Bendix, 1970; Holt, 1994). The implemented catalog is much narrower: multiplication-table systems for the order-eight dihedral and quaternion groups. Tokens are nonidentity group elements; each adjacent pair is replaced by its product token or by ε. Associativity and length decrease give convergence. The task does not run KBMAG or completion.

Implemented task. The observation includes a system identifier, the current word, and the target normal form. An action chooses a rule id and a position. If the left-hand side of that rule occurs at the selected position, the environment replaces it by the right-hand side. Otherwise the action is invalid.

Acceptance. The verifier checks literal rule membership and literal substring matching. It then confirms that the final word is the declared target normal form. Legal intermediate rewrites are not enough: a trace that ends at the wrong normal form is still rejected.

Accepted illustration. In the dihedral catalog, $rr^{-1} \to \varepsilon$ is one multiplication-table rule. In its token encoding this is $[1, 3] \to []$.

Negative / rejected illustration. In the same catalog, r is a nonidentity normal form. A claimed reduction $r \to \varepsilon$ has no matching pair rule and cannot solve the empty-target task.

G.10 SMALL-CANCELLATION WORD PROBLEM

Mathematical problem. In a $C'(1/6)$ small-cancellation presentation, Dehn's algorithm is sound and complete for the word problem. If a nonempty freely reduced word represents the identity, it contains a long subword of some cyclic conjugate of a relator; replacing that long piece by the shorter complementary piece reduces length. This gives a clean exact-search task: find the correct length-reducing Dehn moves (Lyndon & Schupp, 1977).

Implemented task. The observation contains padded relators and the current word. An action selects a relator and position, or submits. The deterministic reducer searches cyclic forms of that relator and its inverse for a matching piece longer than half the relator, replaces it by the inverse complementary piece, then freely reduces. Orientation and cyclic shift are not separate actions.

Acceptance. The output is a Dehn-reduction trace. Success is an empty word, reached by reduction or checked on submission. A complete word-problem interpretation needs the $C'(1/6)$ hypothesis; the row decoder does not itself certify that hypothesis. Legal relator rewrites remain sound without a completeness claim.

Accepted illustration. For the illustrative presentation $\langle a, b, c, d, e, f, g \mid abcdefg\rangle$, the deterministic Dehn reducer matches the whole relator $abcdefg$ and replaces it by ε. The symmetrized relator has no nonempty repeated piece, so the small-cancellation hypothesis holds.

Negative / rejected illustration. In that presentation, a is nontrivial: eliminating g leaves the free group on $a, \ldots, f$. It is also nonempty and Dehn-reduced. This is a mathematical negative, not a timeout-based label.

G.11 PATH-RAAG WORD PROBLEM

Mathematical problem. A right-angled Artin group (RAAG) A_Γ has one generator for each vertex of a graph Γ and one commutation relation $[v_i, v_j] = 1$ for each edge. WordBench uses the path graph P_n, so neighboring generators commute but non-neighboring ones do not. The word problem is decidable by normal forms, yet search can be difficult because many legal swaps do not shorten the word (Wrathall, 1988; Charney, 2007).

Implemented task. The observation contains padded current and target words. A position action cancels an inverse pair, otherwise swaps commuting letters. Distinct generators commute only across path edges (equal generator types also commute). Current code uses the action as the literal position, even if the legacy active-span-remapping option is set.

Acceptance. Success requires literal equality with the decoded target. The output is the action trace. Commutations and cancellations preserve the RAAG element; equality of arbitrary group elements is not a substitute for this endpoint test. This is the path-graph case of the next task.

Accepted illustration. In $A(P_3) = \langle a_1, a_2, a_3 \mid [a_1, a_2] = [a_2, a_3] = 1\rangle$, the words $a_1a_2a_3$ and $a_2a_1a_3$ are equal because a_1 commutes with a_2.

Negative / rejected illustration. The words a_1a_3 and a_3a_1 are not equal in $A(P_3)$, because there is no edge between a_1 and a_3. The verifier therefore rejects a trace that swaps them.

G.12 GRAPH-PRODUCT WORD PROBLEM

Mathematical problem. A graph product assigns a group to each vertex and imposes commutation across edges (Green, 1990). WordBench implements only infinite-cyclic vertex groups, hence a RAAG, with a configurable graph catalog. It does not implement arbitrary internal syllable groups. The path-RAAG task is a special case.

Implemented task. The observation is the padded current word, padded target, and one-hot graph id. Tokens are signed generator indices. A position action cancels adjacent inverse letters, otherwise swaps adjacent commuting letters. The row contains two padded words and optionally a graph id; its target is processed by the implementation's canonicalization routine.

Acceptance. The action trace succeeds on literal equality with the processed target. The target routine computes a piling-based representative. A group-equivalent current word must still reach that particular representative within the available moves and bounds; success is not an immediate group-equality test. The Lean decoder currently uses a different target routine.

Accepted illustration. With edge $\{a, b\}$ and processed target ba, start ab reaches ba by swapping its two letters.

Rejected illustration. With no edge $\{a, c\}$, attempting to swap ac to ca is invalid. The words are also mathematically unequal in the free subgroup generated by a, c.

G.13 SMALL-OVERLAP MONOID WORD PROBLEM

Mathematical problem. Monoid word equality uses defining relations without formal inverse cancellation. Small-overlap hypotheses give efficient algorithms for certain presentations (Kambites, 2009); unrestricted finitely presented monoid word problems need not be decidable. WordBench implements bounded derivation search in a small relation catalog, not a general small-overlap decision algorithm.

Implemented task. The observation is the current word, target, and one-hot relation-set id. An action selects an oriented relation and position, or submits. The configured catalog contains $abca \leftrightarrow defd$ and, in a separate system, $bcab \leftrightarrow efde$. Replacements must fit the length cap.

Acceptance. The output is the relation-action trace. The environment succeeds on literal equality with the target, either after a matching replacement or on submission. It performs no extra inverse cancellation or unstated commutation. The catalog name alone is not a check of a small-overlap condition.

Accepted illustration. In the first configured system, $abca \to defd$ is a one-step accepted derivation.

Negative / rejected illustration. In the presentation with the sole relation $abca = defd$, a and b are unequal: neither length-one word admits any step of the length-four relation. An attempted substitution $a \mapsto b$ is invalid.

G.14 WORD EQUATIONS

Mathematical problem. A word equation asks whether variables can be replaced by concrete strings so that two sides become identical. If $L(X_1, \ldots, X_k)$ and $R(X_1, \ldots, X_k)$ are words over constants and variables, the decision question is

$$\exists \sigma \quad \sigma(L) = \sigma(R),$$

where σ maps variables to words over the alphabet. This is the same mathematical core that appears in string-constraint solving, but WordBench scores explicit assignments rather than solver-internal proofs (Makanin, 1977; Plandowski, 2004; Jeż, 2016).

Implemented task. The registered task `makanin_word_equations` exposes both tokenized equation sides and buffers for X, Y over $\{a, b\}$. Actions append to, backspace, or clear one buffer, or submit. Each buffer has length at most five in the selected configuration; each occurring variable must receive a nonempty word. This is a bounded, nonerasing version of ordinary word equations.

Acceptance. The output is a concrete substitution. Exact concatenation of the substituted sides must agree, and every occurring variable must be nonempty. Success terminates immediately when these conditions hold, or is tested on submission. Variables absent from both sides may remain empty. Failure does not imply unsatisfiability of the unbounded equation.

Accepted illustration. The equation $Xa = aX$ over alphabet $\{a, b\}$ is satisfiable. The assignment $X = a$ gives $aa = aa$, so it is accepted.

Negative / rejected illustration. For $Xa = aX$, the empty assignment satisfies the ordinary monoid equation but is rejected by this environment because X occurs. Separately, $Xa = b$ with $a \neq b$ is mathematically unsatisfiable even when empty assignments are allowed.

G.15 Finite permutation action

Mathematical problem. This family is the finite-state planning abstraction behind many algebraic puzzles. An instance gives a finite set S, generators $\Sigma \leq \mathrm{Sym}(S)$, a start state s, and a target state t. The question is whether there is a generator word $w = \sigma_m \cdots \sigma_1$ such that $w \cdot s = t$.

Implemented task. The implementation permutes positions of a finite vector. Observation concatenates the current and goal vectors; actions apply configured permutations and, when enabled, their inverses. The selected configuration has five positions and two generators, $(1\ 2\ 3)$ and $(3\ 4\ 5)$, with inverse actions. This is a smaller planning analogue of Rubik's cube.

Acceptance. The output is an action sequence, replayed as exact coordinate permutations. Success is literal current–goal vector equality. The environment has no NO submission; orbit nonmembership is a separate mathematical obstruction.

Accepted illustration. In an illustrative three-position configuration with only the swap of positions 1 and 2, $[0, 1, 2] \to [1, 0, 2]$ is accepted in one action.

Negative / rejected illustration. In that three-position configuration, $[0, 2, 1]$ is unreachable from $[0, 1, 2]$: the third coordinate never changes. This is an orbit obstruction.

G.16 Thompson's group F solve-from-scramble

Mathematical problem. Thompson's group F is an infinite finitely presented group whose elements can be represented exactly by reduced pairs of finite binary trees. WordBench uses a constructive solve-from-scramble version: the instance is an element generated by a short word in $x_0^{\pm 1}, x_1^{\pm 1}$, and the goal is to produce a word that returns it to the identity (Cannon et al., 1996).

Implemented task. The observation contains the padded leaf-depth sequences of the two trees and normalized remaining steps. Four actions right-multiply by $x_0^{\pm 1}, x_1^{\pm 1}$ and perform exact tree-pair reduction. The word and leaf caps constrain this solve-from-scramble search; it is a planning counterpart of normal-form equality.

Acceptance. The output is a generator-action trace. The environment multiplies the start element by that word using exact tree-pair operations, reduces the resulting pair, and accepts only if the reduced pair is the identity pair. The released slice is all-YES by construction, so red examples below are rejected certificates rather than dataset NO rows.

Accepted illustration. If the scramble is $g = x_0 x_1^{-1}$, then the trace $x_1 x_0^{-1}$ solves it because $g\, x_1 x_0^{-1} = e$.

Rejected illustration. An empty trace does not solve a nonidentity scramble. Even if the two trees share many leaves, the verifier accepts only the exact identity tree pair.

G.17 Rubik's cube planning

Mathematical problem. The standard $3 \times 3 \times 3$ Rubik's cube is a large finite group action generated by outer face turns. WordBench treats it as exact planning: given a legal scrambled cube state, produce a move sequence that returns the state to the solved cube. The family follows the exact-verification philosophy used in learning-and-search work such as DeepCubeA (Agostinelli et al., 2019).

Implemented task. The selected Cube20 configuration observes one-hot facelet colors for a $3 \times 3 \times 3$ cube and offers the 18 outer face turns (quarter, inverse-quarter, and half turns). Exact transitions update the cube; the episode succeeds on the solved state. The output is the sequence of turns, as in finite permutation planning.

Acceptance. The certificate is a move sequence. The checker replays the face turns using an exact cube transition table and accepts only if the final state is the solved cube. Heuristic distance, number of matching stickers, or visual similarity is not correctness evidence. The Cube20-style slice is all-YES; therefore red examples are rejected certificates rather than dataset NO rows.

Accepted illustration. A cube scrambled by one move R is solved by the one-move certificate R^{-1}. Exact replay gives the solved cube.

Rejected illustration. For the same scramble, submitting U is rejected: after replaying U, the cube is still not the solved state, even if many stickers happen to be in the correct locations.

G.18 Whitehead orbit search

Mathematical problem. Whitehead's algorithm decides automorphism orbits of free-group words (Whitehead, 1936; Myasnikov & Haralick, 2006). The registered witness task acts on cyclic words: it seeks automorphisms taking the source to a conjugate of the target. This combines automorphism search with the conjugacy normalization used elsewhere in WordBench.

Implemented task. The observation contains canonical conjugacy representatives of the current and target words. Actions choose a member of the finite Whitehead automorphism library. The transition applies it globally, freely/cyclically reduces, and canonicalizes; moves overflowing the word cap are rejected.

Acceptance. The output is an automorphism-action sequence. Executable success is equality of canonical conjugacy representatives, not literal equality of the original words. Invariants such as primitivity can separately certify that no such orbit relation exists.

Accepted illustration. In $F(a, b)$, the automorphism swapping a and b maps ab to ba. Thus $u = ab$ and $v = ba$ are in the same Whitehead orbit.

Negative / rejected illustration. The words a and a^2 are not in the same automorphism orbit of $F(a, b)$. Automorphisms send primitive elements to primitive elements, and a^2 is not primitive.

G.19 Burau matrix-identity classification

Mathematical problem. For a braid word $w \in B_4$, predict whether its reduced Burau matrix $\rho(w) \in \mathrm{GL}_3(\mathbb{Z}[t, t^{-1}])$ equals I. Exact matrix identity is decidable by Laurent-polynomial arithmetic. This is a representation-level question, related to braid rewriting and to the target-matrix search in Section G.21 (Birman, 1974).

Implemented task. The task `burau_matrix_equality` observes only the padded current word. Position actions attempt local far commutations or same-sign braid rewrites with free reduction. Two additional actions explicitly submit YES or NO. There are no append/backspace actions. Rows store a binary label followed by a nonempty padded input word.

Acceptance. The environment accepts a submitted Boolean exactly when it matches the stored label; it does not recompute the matrix at submission. Reaching the horizon without submission fails. Independent validation can evaluate $\rho(w)$ exactly, but that is a distinct check and assumes no correctness of stored labels.

Accepted illustration. For the illustrative row $w = \sigma_1\sigma_1^{-1}$ with label YES, $\rho(w) = I$ by the exact evaluator. Submitting YES succeeds even though the raw input must be nonempty.

Rejected illustration. For $w = \sigma_1$ with label NO, exact evaluation gives $\rho(w) \neq I$: NO succeeds and YES fails. A mislabeled row would still be scored against its stored label.

G.20 Andrews–Curtis trivialization

Mathematical problem. Andrews–Curtis trivialization searches for presentation moves taking a balanced relator tuple to a trivial tuple. Classical AC moves invert, multiply, and conjugate relators. WordBench studies bounded constructive searches on presentation datasets (Shehper et al., 2024; Zhang et al., 2025); a successful certificate concerns the particular presentation and does not settle the general conjecture. Free reduction connects this task to the free-group families.

Implemented task. The selected rank-two environment observes two padded relators. Its 12 actions multiply a relator on the right by the other relator or its inverse, or conjugate either relator by a signed generator. Free and configured cyclic reduction follow moves. There is no standalone inversion or submit action in this menu.

Acceptance. Direct success detects two distinct signed singleton generators. The selected PPO configuration also enables a solved-presentation cache; a cache hit may terminate before a direct trivial tuple is reached. Such an episode is not by itself a full trivializing trace: a certifying handoff needs the corresponding verified continuation. General theorem-checking is disabled in this configuration. Timeout is not a negative certificate.

Accepted illustration. The tuple (x, y) satisfies the direct terminal predicate. The tuple (xy, y) reaches (x, y) by right-multiplying its first relator by y^{-1} and reducing, giving a concrete one-action solution.

Negative / rejected illustration. The presentation $\langle x, y \mid x^2, y\rangle$ cannot be AC-trivialized to $\langle x, y \mid x, y\rangle$: its abelianization contains a $\mathbb{Z}/2\mathbb{Z}$ factor, while the trivial presentation has trivial abelianization. This is a certified algebraic obstruction.

Budget-limited search. For a Miller–Schupp presentation under study, if no trivializing trace and no obstruction is found within budget, the outcome is *unknown*. WordBench does not convert that budget failure into a NO label.

G.21 Burau target-matrix search

Mathematical problem. The task `burau_b4_search` asks for a word w such that $\rho(w) = M$, where M is supplied indirectly as features of the matrix of a hidden target word. It is constructive matrix matching, complementing identity classification in Section G.19. No faithfulness theorem, distinctness condition, or nontriviality condition is used.

Implemented task. The observation concatenates the padded candidate word and 27 numerical features: the 3×3 target matrix evaluated at $t = 2, 3, -1$ modulo $2^{31} - 1$, scaled and stored as floats. The target word and exact current matrix are hidden from the policy observation. Actions attempt a local rewrite, append one of six signed braid generators, backspace, or submit. Rows contain padded start and target words; the freely reduced target must be nonempty.

Acceptance. On explicit submission, acceptance compares the candidate and target matrices exactly over $\mathbb{Z}[t, t^{-1}]$. Numerical-feature agreement alone does not suffice. Appending and backspacing can change the braid element, so this is not merely a rewrite proof of equality to the starting word. Lean has a target-reachability predicate, but current automatic proof generation is not aligned with this interface.

Accepted illustration. With start $\sigma_1\sigma_2\sigma_1$ and hidden target $\sigma_2\sigma_1\sigma_2$, one braid rewrite followed by submit succeeds. The exact evaluator confirms matrix equality; submitting the unchanged start also succeeds because only the matrix matters.

Rejected illustration. With hidden target σ_1, submitting the empty candidate fails since $I \neq \rho(\sigma_1)$. Appending σ_1 and submitting succeeds; the target word need not be distinct from the candidate.

G.22 Group-ring zero-divisor witness search

Mathematical problem. A group ring multiplies finite sums using bilinearity and group multiplication. The selected WordBench configuration searches in $\mathbb{F}_2[C_n]$, where C_n is finite cyclic and has torsion for $n \geq 2$. This is an exact finite coefficient task, not a test of the torsion-free zero-divisor conjecture. The unit conjecture was disproved by Gardam (Gardam, 2021); unit search is a different predicate from zero-divisor search.

Implemented task. The configured mode is `cyclic_zero_divisor`. A dataset row is $[n]$; observation gives n, a cursor, and two padded coefficient buffers. Actions write 0 or 1, backspace, or submit. The selected file is `train_n31.txt`, with maximum $n = 31$. Other code modes implement unit products or SAT assignments and must be described separately if used.

Acceptance. The submitted output is two fully written length-n binary coefficient vectors for α, β. Acceptance requires both nonzero and cyclic convolution zero modulo two, i.e. $\alpha\beta = 0$ in $\mathbb{F}_2[C_n]$. No claim about torsion-free groups follows. Unit modes check product one and do not enforce nontrivial units.

Accepted illustration. For $C_2 = \langle g \mid g^2 = 1 \rangle$, choose $\alpha = \beta = 1 + g$. Then $(1 + g)^2 = 1 + g^2 = 0$ in characteristic two. Writing $[1, 1, 1, 1]$ and submitting succeeds.

Rejected illustration. For $n = 2$, the fully written pair $\alpha = 0$, $\beta = 1$ is rejected despite having product zero, since both factors must be nonzero.

G.23 KNOT EQUIVALENCE BY RESTRICTED BRAID MOVES

Mathematical problem. Closures of braids describe links. Braid relations and conjugation preserve the closure; general Markov equivalence also uses stabilization. WordBench searches only a fixed-strand braid-move subset, related to braid rewriting, rather than arbitrary Reidemeister or Markov moves (Birman, 1974).

Implemented task. The observation contains padded current and target braid words. Actions attempt a cancellation/three-letter braid rewrite at a position, a far commutation, a cyclic rotation at a cut, or submit. Words are freely reduced. No planar-diagram encoding, Reidemeister menu, or stabilization action is exposed.

Acceptance. The trace succeeds only on explicit submit with equal freely reduced current and target words. An unsuccessful restricted search does not establish inequivalence of closures. Invalid moves leave the state unchanged; they do not supply a mathematical step. Lean replay support for this proxy must be distinguished from a theorem identifying it with topological link equivalence.

Accepted illustration. For start $\sigma_1\sigma_2$ and target $\sigma_2\sigma_1$, rotate at the cut after the first letter and submit. The target is reached and the closures are equivalent by conjugation.

Rejected illustration. For the same start and target, immediate submit fails because the reduced strings differ, even though their closures are equivalent. This is a rejected submission, not a negative link-equivalence result.

G.24 GRÖBNER NORMAL FORM (IDEAL REDUCTION)

Mathematical problem. For a Gröbner basis G over $\mathbb{Q}$ and a fixed monomial order, polynomial reduction yields a unique normal form modulo $\langle G \rangle$ (Cox et al., 2015). WordBench searches for a reduction sequence to a supplied remainder, paralleling word-normalization tasks. The current task does not use $\mathbb{F}_2$ arithmetic.

Implemented task. A self-contained row and observation encode the monomial order (lex or grevlex), number of variables, basis, current polynomial, and target. Rational coefficients are represented by integer numerators and denominators. An action chooses one of eight basis slots, or submits. The reducer automatically selects the greatest term divisible by the chosen leading monomial; the policy does not choose a term slot.

Acceptance. The output is the basis-action trace followed by submit. Each applied reduction uses exact rational arithmetic; submission accepts literal polynomial equality with the stored target. Calling that target a normal form relies on basis/target validation during dataset construction, not on a new Buchberger check at submission.

Accepted illustration. Over $\mathbb{Q}[x]$, take $G = \{x^2 + x\}$, $f = x^2$, and target $-x$. Subtracting the basis polynomial gives $x^2 \to -x$; selecting that reducer and submitting succeeds.

Rejected illustration. With the same instance, immediate submission fails. A claimed target 0 is mathematically wrong: the remainder is $-x$, not 0.

H ASSET PROVENANCE AND RELEASE POLICY

WordBench distinguishes mathematical provenance from asset reuse. Most instances are newly generated from standard definitions, presentations, transition systems, normal forms, and verifier contracts. In those cases, the bibliography records the mathematical source, but no external dataset is copied into the release.

A smaller number of families use existing software or benchmark ecosystems. Word-equation rows may use SMT-LIB or StringFuzz-style resources when the license metadata is explicit and compatible; Andrews–Curtis rows are tied to the AC-Solver and Miller–Schupp experimental line; Rubik planning follows DeepCubeA/Cube20 conventions; and KBMAG/GAP may be invoked as external exact backends. Lean and mathlib are used for the formalization layer. Tables 9 and 10 record the release obligations for these cases.

Generated WordBench data are released under the benchmark data license, and code/verifiers are released under a permissive open-source license. Third-party assets retain their original notices and licenses. Rows or assets without a compatible release path are excluded from the public split.

Table 9: Family-level data provenance and release constraints for the 24 WordBench families.

Family	How the instances enter WordBench	Asset/license status	Status
Andrews–Curtis trivialization	Uses the Miller–Schupp / AC environment line from the AC-Solver literature and/or generated trivializable traces.	AC-Solver is MIT-licensed. If `MS-1190` rows are redistributed, include the AC-Solver citation and MIT license notice in the release package.	vetted
Artin–Tits word problem	Generated by scrambling the empty word using a finite Artin–Tits tile move set (plus free cancellations) for the 4-generator presentation discussed by Gowers.	New WordBench data; cite Gowers for mathematical provenance of the motivating formulation; no copied external dataset is required.	vetted
Baumslag–Solitar BS(1,2)	Generated from the fixed presentation $\langle a, t \mid tat^{-1} = a^2 \rangle$ and verifier-checked rewrite traces.	New WordBench data from a standard mathematical presentation; no external dataset used.	vetted
Braid word rewriting	Generated from Artin braid relations and exact rewrite-to-target traces.	New WordBench data; no copied external dataset.	vetted
Burau B_4 search	Generated from freely reduced B_4 words where targets are formed by appending braid generators and freely reducing.	New WordBench data; no copied external dataset.	vetted
Burau matrix equality	Balanced YES/NO B_4 instances. YES words are constructed from conjugated Artin relators and verified to have identity Burau matrices, while NO words have nonidentity Burau matrices.	New WordBench data; no copied external dataset.	vetted
Coxeter type-A rewriting	Generated from type-A Coxeter relations $s_i^2 = 1$, braid relations, and commuting relations.	New WordBench data; no copied external dataset.	vetted
Finite permutation action	Generated from finite permutation generators and exact start/goal action traces.	New WordBench data; Rubik/DeepCubeA references are conceptual planning precedents, not copied data.	vetted
Free-group conjugacy	Generated by sampling words and explicit conjugators, with NO rows checked by free-group conjugacy criteria.	New WordBench data; no copied external dataset.	vetted
Free-group subgroup membership	Generated from sampled free-group subgroups and witness expressions; Stallings-folding literature is cited for the algorithmic basis.	New WordBench data; no copied external dataset.	vetted
Free-group word	Generated from free reductions, cancellation traces, and exact normal-form comparisons.	New WordBench data; no copied external dataset. Release generated instances under the WordBench data license.	vetted
Graph-product word	Generated from graph-controlled commutation and normal-form semantics.	New WordBench data; no copied external dataset.	vetted
Gröbner normal form	Generated from small Gröbner bases and random polynomials with targets computed by exact reduction.	New WordBench data; standard benchmark families (cyclic/katsura) are mathematical provenance, not copied data.	vetted
Heisenberg collection	Generated from the discrete Heisenberg normal-form collection rules.	New WordBench data; no copied external dataset.	vetted
Kaplansky group rings	Generated from small group-ring instances and verifier-checked explicit zero-divisor witnesses; conjectural torsion-free regimes remain open discovery settings.	New WordBench data; cite Gardam for mathematical provenance; no copied external dataset.	vetted
KBMAG rewriting	Generated from small convergent rewriting systems; KBMAG is cited/optionally invoked as a reference rewriting package.	If KBMAG is invoked, record GAP package `kbmag` under GPL-2.0-or-later and do not vendor it unless GPL obligations are followed. The benchmark rows themselves are newly generated.	vetted
Knot equivalence	Generated by applying valid local move sequences to seed diagram encodings and verifying equivalence by exact replay.	New WordBench data; do not redistribute third-party knot tables unless their licenses are included.	vetted
Makanin word equations	Constructed either from WordBench templates/StringFuzz-generated formulas or from SMT-LIB rows with explicit license metadata.	StringFuzz is MIT-licensed; SMT-LIB 2025 is CC BY 4.0 at the release level and carries per-benchmark contributor/license metadata. Include only rows with explicit compatible license metadata.	vetted
Path-RAAG word	Generated from path right-angled Artin group commutation graphs and exact commutation/cancellation traces.	New WordBench data; no copied external dataset.	vetted
Rubik's cube	Generated by applying legal outer face turns to the solved cube and verifying returned move traces exactly.	Do not scrape or redistribute cube20.org files. If DeepCubeA code conventions are reused, include the MIT notice; otherwise cite DeepCubeA/Cube20 only as planning precedent.	vetted
Small-cancellation word	Generated from small-cancellation-style presentations and Dehn-reduction traces.	New WordBench data; no copied external dataset.	vetted
Small-overlap monoid word	Generated from compact small-overlap presentations following the algorithmic setup of the cited literature.	New WordBench data; no copied external dataset.	vetted
Thompson's group F	Generated from sampled scrambles and exact inverse traces in a tree-pair / normal-form implementation.	New WordBench data from standard mathematical definitions; no copied external dataset.	vetted
Whitehead free group	Generated from a finite library of Whitehead automorphisms and exact orbit witnesses.	New WordBench data; cited Whitehead-search work is mathematical/algorithmic provenance, not copied data.	vetted

Table 10: External assets and release obligations. The table records the role of each asset, the applicable license or terms, and the action required in the release package.

External asset	Role in WordBench	License / terms	Required attribution and release action
SMT-LIB benchmark library	Optional source of word-equation rows, restricted to benchmarks with explicit compatible metadata.	2025 Zenodo release is CC BY 4.0; individual benchmarks also carry contributor and license metadata.	Cite SMT-LIB; preserve per-file source/license metadata; exclude any row without compatible license metadata.
StringFuzz	Optional generator/transformer for word-equation instances in SMT-LIB format.	MIT License, copyright Dmitry Blotsky.	Cite StringFuzz; include MIT notice if code or generated metadata identifies the generator.
AC-Solver / Andrews–Curtis environment	Source line for the AC move environment and Miller–Schupp benchmark usage.	MIT License, copyright Ali Shehper.	Cite Shehper et al.; include MIT notice if rows/code are redistributed or adapted.
DeepCubeA	Planning precedent and optional reference for Rubik environment conventions.	MIT License, copyright Forest Agostinelli.	Cite Agostinelli et al.; include MIT notice only if code is reused.
KBMAG / GAP package `kbmag`	Optional external exact rewriting backend or comparison tool.	GPL-2.0-or-later.	Do not bundle unless GPL obligations are satisfied; when invoked externally, cite KBMAG and record version.
GAP system	Optional host for group-theoretic packages.	GPL-2.0-or-later for the core GAP system.	Record version and license if used in scripts or reproducibility package.
Lean 4	Proof assistant for formalized task semantics.	Apache-2.0.	Cite Lean; include Apache-2.0 notice if Lean-derived files or toolchain manifests are distributed.
mathlib4	Lean mathematical library used by the formalization.	Apache-2.0.	Cite mathlib; preserve license/notice files in the formalization release.