SLR: Automated Synthesis for Scalable Logical Reasoning

Lukas Helff[1,2], **Ahmad Omar**[1], **Felix Friedrich**[1,2], **Antonia Wüst**[1], **Hikaru Shindo**[1], **Tim Woydt**[1], **Rupert Mitchell**[1,2], **Patrick Schramowski**[1,2,3,4], **Wolfgang Stammer**[1,2,5], **Kristian Kersting**[1,2,3,5,6]

[1]TU Darmstadt [2]hessian.AI [3]DFKI [4]CERTAIN, Germany

[5]Lab1141 [6]Centre for Cognitive Science, Darmstadt

Code: https://github.com/ml-research/ScalableLogicalReasoning

Data: https://huggingface.co/datasets/AIML-TUDA/SLR-Bench

Abstract

We introduce SLR, an end-to-end framework for systematic evaluation and training of Large Language Models (LLMs) via **S**calable **L**ogical **R**easoning. Given a user's task specification, SLR automatically synthesizes (i) an instruction prompt for an inductive reasoning task, (ii) a validation program, executable on model outputs to provide verifiable rewards, and (iii) the latent ground-truth rule. This process is fully automated, scalable, requires no human annotations, and offers precise control over task difficulty. Using SLR, we create SLR-BENCH, a benchmark comprising 19k prompts organized into 20 curriculum levels that progressively increase in relational, arithmetic, and recursive complexity. Large-scale evaluation reveals that contemporary LLMs readily produce syntactically valid rules, yet often fail at correct logical inference. Recent reasoning LLMs demonstrate improved performance but incur very high test-time computation, with costs exceeding $300 for just 1,000 prompts. Finally, curriculum learning via SLR doubles Llama-3-8B accuracy on SLR-BENCH, achieving parity with Gemini-Flash-Thinking at a fraction of computational cost. Moreover, these reasoning capabilities generalize to a wide range of established benchmarks, underscoring the effectiveness of SLR for downstream reasoning.

1 Introduction

Logical reasoning is a fundamental aspect of intelligence, yet state-of-the-art AI systems still struggle with tasks that require robust reasoning and systematic generalization (Delfosse et al., 2025; Kostikova et al., 2025; Woydt et al., 2025; Wüst et al., 2025; Helff et al., 2025; Sinha et al., 2019). Existing benchmarks intended to evaluate reasoning capabilities, however, primarily emphasize *deductive* reasoning, where conclusions necessarily follow from given premises. This includes tasks such as math word problems (Hendrycks et al., 2021) and logic puzzles (Lin et al., 2025; Xie et al., 2025; Liu et al., 2025). *Inductive* reasoning, by contrast, involves inferring general rules or patterns from specific examples, which remains particularly challenging and underexplored in large language models (Luo et al., 2024; Xie et al., 2024) (see also Tab. 1).

Current evaluation frameworks commonly employ constrained formats (e.g., multiple-choice) or rely on other LLMs as judges (Patel et al., 2024a; Lin, 2024; Lin et al., 2024), making it difficult to assess whether models genuinely understand logical structure or are merely exploiting superficial patterns in the data. Moreover, as training sets grow, benchmark items or their paraphrases increasingly overlap with pre-training data, making apparent reasoning abilities potentially just memorization (Shojaee et al., 2025; Xie et al., 2024).

To tackle these challenges, this paper introduces **SLR** (Scalable Logical Reasoning), an open-source framework for evaluating and training models in inductive logical reasoning. Based on a user-defined logic task (Fig. 1, left), the task synthesizer (center) automatically generates novel inductive logic programming (ILP) tasks (Muggleton and de Raedt, 1994; Cropper and Dumančić, 2022) of controllable complexity. Each task comes with (i) a latent ground-truth rule, (ii) an executable validation program, and (iii) an instruction prompt for the reasoning task. The ground-truth rule serves as the reference answer, while the validation program deterministically evaluates any candidate hypothesis. SLR supports both systematic model evaluation (Fig. 1, top right) and downstream model training, via supervised finetuning or reinforcement learning with rewards provided by the integrated symbolic judge (Fig. 1, bottom right). SLR 's fully symbolic and automated pipeline eliminates the need for human annotation and avoids dataset overlap.

Leveraging SLR, we present **SLR-BENCH** (Fig. 3), a 19k task benchmark that forms a twenty-level curriculum of increasing logical complexity. These

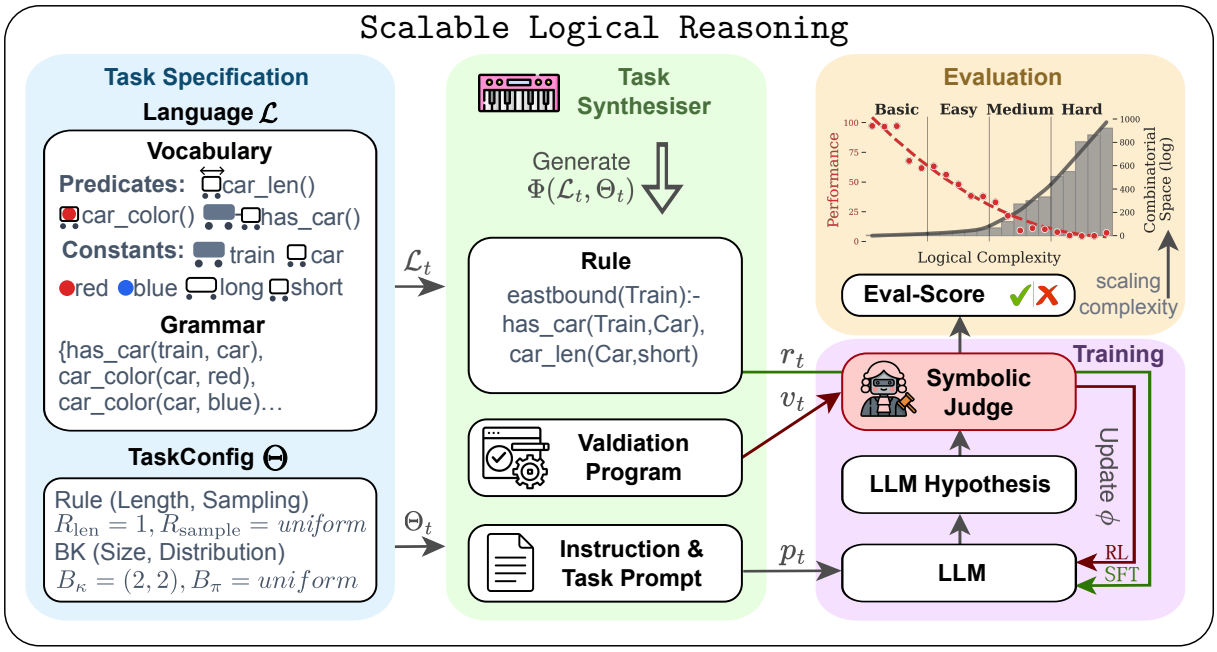

Figure 1: Overview of the **SLR** Framework, including task specification, automated task synthesis, training, and evaluation. **Left (blue):** Language defines vocabulary and grammar, Task Config specifies configuration parameters for the synthesis. **Middle (green):** The task synthesizer automatically generates ground-truth rules, validation programs, and instruction prompts. **Right (purple):** Training LLMs on logic tasks via SFT (cross-entropy) or RL (symbolic judge feedback). **Right (orange):** Evaluates LLMs using feedback provided by the symbolic judge. Arrows denote data and control flow through synthesis, prompting, evaluation, and downstream training loops.

levels are further organized into four curriculum tiers: *basic*, *easy*, *medium*, and *hard*. Each task is unique, with a precise and systematic assessment of inductive logical reasoning skills. In evaluations, we find that while LLMs are generally well-versed in generating syntactically valid rules, robust logical reasoning remains challenging. Performance declines sharply as task complexity increases. Scaling model size brings only marginal improvements, while scaling test-time compute boosts reasoning, but returns diminish as complexity rises.

Beyond evaluation, SLR enables curriculum learning, boosting reasoning both in-domain and across established reasoning benchmarks. SLR-tuned Llama-3-8B not only surpasses all conventional LLMs on SLR-BENCH, but also outperforms recent reasoning LLMs such as Gemini-2.0-flash-thinking, while using fewer inference tokens. Notably, these enhanced reasoning capabilities generalize downstream, improving performance on MMLU, LogicQA, HellaSwag, and GPQA.

In sum, our contributions are: (i) SLR, an open framework for automated synthesis and symbolic evaluation of logical reasoning in LLMs; (ii) SLR-BENCH, a 19k-task benchmark organized as a 20-level curriculum of increasing logical complexity, enabling both training and evaluation across a controlled reasoning spectrum; (iii) a large-scale evaluation of LLMs on SLR-BENCH, revealing key insights and trade-offs in model performance; (iv) curriculum learning with SLR substantially improves both in-domain and downstream reasoning.

2 Related Work

Evaluating LLMs' Logical Reasoning. Tab. 1 provides an overview of existing logical reasoning benchmarks in terms of inference types, dataset origins, and evaluation formats. Notable datasets include LogiQA/2.0 (Liu et al., 2020, 2023), FOLIO (Han et al., 2022) (deductive reasoning), AbductionRules (Young et al., 2022) (abductive reasoning), bAbI (Weston et al., 2015), and CLUTRR (Sinha et al., 2019) (synthetic QA with inductive reasoning). Aggregate testbeds such as BIG-Bench (Kazemi et al., 2025; Suzgun et al., 2023), HLE (Phan et al., 2025), FineLogic (Zhou et al., 2025), and LogiGLUE (Luo et al., 2024) span a range of tasks and inference styles. Other benchmarks, such as Proofwriter, PrOntoQA, KOR-Bench, FLD, Multi-LogiEval, SynLogic, ZebraLogic, and the K&K Sandbox (Oyvind et al., 2021; Saparov and He, 2023; Morishita et al., 2024;

Table 1: Comparison of **logic reasoning benchmarks.** **Reasoning Type**: Logical inference type (deduction, induction, abduction). **Creation**: Dataset origin (synthetic, human-annotation, DS collection). **Evaluation**: Output scoring (symbolic execution, multiple choice (MC), LLM, exact match (EM)). **Task Synthesis**: Supports for tasks generation. **Custom Tasks**: User-defined task creation (via language, grammar, or setup). **Curriculum Learning**: Curriculum-based progression of difficulty. **Scalable Complexity**: Supports arbitrarily scaling task complexity. (✓: fully supported, ✗: not supported, (✓): partially/limited)

Dataset	Reasoning Type	Data Creation	Evaluation Methodology	Task Synthesis	Custom Tasks	Curriculum Learning	Scalable Complexity
LogiQA (Liu et al., 2020)	Deduction	Human	MC	✗	✗	✗	✗
LogiQA 2.0 (Liu et al., 2023)	Mix	Human	MCQA/Auto	✗	✗	✗	✗
FOLIO (Han et al., 2022)	Deduction	Human	EM	✗	✗	✗	✗
AbductionRules (Young et al., 2022)	Abduction	Synthetic/Human	EM	✗	✗	✗	✗
FineLogic (Zhou et al., 2025)	Deduction	DS collection	Symbolic/LLM	✗	✗	✗	✗
HLE (Phan et al., 2025)	Mix	DS collection	LLM	✗	✗	✗	✗
Big-Bench (Phan et al., 2025)	Mix	DS collection	Mix	✗	✗	✗	✗
LogiGLUE (Luo et al., 2024)	Mix	DS collection	MCQA/Auto	✗	✗	✗	✗
CLUTRR (Sinha et al., 2019)	Induction	Synthetic	EM	✗	✗	✗	✗
KOR-Bench (Ma et al., 2024)	Mix	Synthetic/Human	EM/Symbolic	(✓)	✗	✗	✗
PrOntoQA (Saparov and He, 2023)	Deduction	Synthetic	EM	✓	✗	✗	✗
SynLogic (Liu et al., 2025)	Deduction	Synthetic	EM	✓	✗	✗	✗
FLD (Morishita et al., 2024)	Deduction	Synthetic	Symbolic	✓	✓	✗	(✓)
K&K (Xie et al., 2025, 2024)	Deduction	Synthetic	EM	✓	✗	(✓)	(✓)
ZebraLogic (Lin et al., 2025)	Deduction	Synthetic	EM	✓	✗	(✓)	(✓)
SLR (ours)	Induction	Synthetic	Symbolic	✓	✓	✓	✓

Patel et al., 2024b; Liu et al., 2025; Lin et al., 2025; Xie et al., 2025, 2024; Mondorf and Plank, 2024) generate tasks using various logics or ontologies, often with fixed configurations or evaluation via exact match. However, most existing benchmarks lack key features such as scalability, extensibility, controlled curricula, and flexible task synthesis, all of which are addressed by SLR.

Limits and Promises of Reasoning LLMs. LLMs like GPT-4 (OpenAI et al., 2024), Llama-3 (Meta et al., 2024), and Qwen (Bai et al., 2023) can handle basic reasoning and coding tasks but often struggle with true abstraction (Shojaee et al., 2025; Xie et al., 2024). Recent *reasoning LLMs* attempt to bridge this gap by scaling *test-time* compute. Systems like OpenAI's *o1/o3* (OpenAI, 2025) or DeepSeek-R1 (DeepSeek-AI et al., 2025) generate and re-rank thousands of reasoning traces per query, achieving state-of-the-art results on, e.g., math or coding (Quan et al., 2025; Hendrycks et al., 2021; Rein et al., 2024; Gao et al., 2024a; Fourrier et al., 2024; Clark et al., 2018). However, these gains come at a steep cost (Fan et al., 2025; Kim et al., 2025). Some studies question whether such models truly learn logical structure or merely exploit surface-level patterns (Fan et al., 2025; Shojaee et al., 2025; Xie et al., 2024). Curriculum learning has been shown to enhance training robustness and generalization (Bengio et al., 2009; Bursztyn et al., 2022), yet no prior framework offers a flexible framework for task synthesis for automatic curriculum generation with symbolic evaluation for reasoning at scale. SLR addresses this gap.

3 SLR: Automatic Benchmark Synthesis

SLR is a scalable methodology for systematically generating, evaluating, and training LLMs on inductive reasoning tasks. Its goal is to automate the creation of diverse, challenging logical reasoning benchmarks, embedded as natural language prompts, with model outputs that can be efficiently verified via symbolic execution of Inductive Logic Programming (ILP) programs (Muggleton and de Raedt, 1994; Cropper and Dumančić, 2022). The overall pipeline (Fig. 1) has three main stages: *task specification*, *synthesis*, and *evaluation/training*.

3.1 Task Specification (Input)

The SLR synthesizer is controlled by the Task Language $\mathcal{L}$, which defines the logical vocabulary and grammar, and the Task Configuration Θ, which controls the generation process (see Fig. 1, left).

Language Specification ($\mathcal{L}$): We define a language $\mathcal{L} = (\mathcal{V}, \mathcal{G})$ that specifies the building blocks for task generation. The Vocabulary $\mathcal{V}$ comprises a set of constant, function, and predicate symbols that form the syntax for generating rules, examples, and background knowledge. The vocabulary induces the Herbrand base $\mathrm{HB}(\mathcal{V})$, which is the set of

Algorithm 1 Task Synthesizer

Require: $\mathcal{L}$, B_π, κ_{pos}, κ_{neg}, $R_{\text{sample}}, R_{\text{len}}$
1: $B \leftarrow \varnothing, E^+ \leftarrow \varnothing, E^- \leftarrow \varnothing$
Rule Synthesis
2: $R^\star \leftarrow \text{RULEGENERATOR}(\mathcal{L}, R_{len}, R_{sample})$
3: **while** $|E^+| < \kappa_{\text{pos}}$ **or** $|E^-| < \kappa_{\text{neg}}$ **do**
Background Synthesis
4: $b \leftarrow \text{BACKGROUNDGENERATOR}(\mathcal{L}, B_\pi)$
5: $(y, q) \leftarrow \text{ASSIGNLABEL}(R^\star, b)$
Stratified Rejection Sampling
6: **if** $y = 1$ **and** $|E^+| < \kappa_{\text{pos}}$ **then** ▷ accept positive
7: $B \leftarrow B \cup \{b\};\ E^+ \leftarrow E^+ \cup \{q\}$
8: **else if** $y = 0$ **and** $|E^-| < \kappa_{\text{neg}}$ **then** ▷ accept negative
9: $B \leftarrow B \cup \{b\};\ E^- \leftarrow E^- \cup \{q\}$
10: **else**
11: **continue** ▷ reject sample
12: **end if**
13: **end while**
Synthesizer Output
14: program $\leftarrow \text{VALIDATIONPROGRAM}(B, E^+, E^-)$
15: prompt $\leftarrow \text{PROMPTGENERATOR}(B, E^+, E^-)$
16: **return** ($R^\star$, program, prompt)

all syntactically valid ground atoms (facts) (Lloyd, 2012). The Grammar $\mathcal{G}$ is formed by a set of semantic rules that filter the Herbrand base to include only meaningful atoms, $\text{HB}_\mathcal{G}(\mathcal{V})$. For instance, a color can be assigned to a car (car_color(car, red)) but not to semantically incompatible objects.

Task Configuration (Θ): The configuration parameters $\Theta = \langle R_{\text{sample}}, R_{\text{len}}, B_\pi, \kappa \rangle$ give control over the synthesis process. The (i) Rule Sampling Policy (R_{sample}) controls the synthesis of the ground truth rule $R^\star$, which can either be sampled randomly (Uniform Sampling) or generated via an LLM (LLM-Guided Generation). To ensure the LLM produces diverse and challenging logic rules, we leverage an exhaustive prompt (see App. E) that covers a wide array of logical structures and Prolog features (containing arithmetics, recursions, variables, cuts, or comparison operators, etc.). The (ii) Rule Length (R_{len}) specifies the number of literals in the body of the ground-truth rule $R^\star$. The (iii) Background Sampling Policy (B_π) defines a probability mass function that assigns a selection probability to each ground atom in the grammar-filtered Herbrand base $\text{HB}_\mathcal{G}(\mathcal{V})$, enabling designers to encode priors on the data distribution (e.g., *uniform*). We also include *mirror* sampling, where backgrounds for (E^+, E^-) are identical except for ground atoms relevant to $R^\star$. The (vi) Problem Size ($\kappa = (\kappa_{\text{pos}}, \kappa_{\text{neg}})$) specifies the target number of positive ($\kappa_{\text{pos}} = |E^+|$) and negative ($\kappa_{\text{neg}} = |E^-|$) examples. This directly controls the size and class balance of each generated task.

3.2 Task Synthesis (Generation)

The task synthesizer (Fig. 1, center) is an automated process detailed in Alg. 1. Given a high-level task specification, the synthesizer generates complete and solvable ILP problems. The process consists of two main phases: rule synthesis and background synthesis.

Rule Synthesis (Alg. 1, line 2). The process begins with the RULEGENERATOR creating a latent, ground-truth rule $R^\star$. This rule represents the underlying logical pattern that a model is expected to induce. The generation is guided by pre-defined parameters (R_{len}, R_{sample}) that control the length and generation policy for the rule. The resulting rule is a syntactically valid definite clause of the form $h :\!- b_1,\ \ldots,\ b_{R_{\text{len}}}$.

Background Synthesis (Alg. 1, lines 3-13). Once $R^\star$ is fixed, the synthesizer enters a loop to construct the background knowledge B and the label sets E^+ and E^-. This loop executes three steps until the desired number of positive and negative examples is generated:

(i) **Sample Background:** The BACKGROUNDGENERATOR samples a set of ground atoms specifying the properties and relationships of the background instance. Ground atoms are drawn from the probability mass function B_π over $\text{HB}_\mathcal{G}(\mathcal{V})$.

(ii) **Assign Label:** The function determines whether a query atom q (that is the ground atom of the target predicate h) is logically entailed by the sampled background b and the ground-truth rule $R^\star$ (i.e., whether $b \cup R^\star \models q$ holds). This produces a label (positive or negative) for the query. We denote the labeling function as:

$$\text{ASSIGN}(R^\star, b) \;=\; \begin{cases} (1, q), & \text{if } b \cup R^\star \models q \\ (0, q), & \text{otherwise} \end{cases}$$

(iii) **Accept/Reject Sample:** To ensure the desired class balance, a stratified rejection sampling strategy is used to populate the example sets. The generated background b and query q are accepted only if the corresponding example set (E^+ or E^-) is not yet full, as specified by the task size parameter (κ). If accepted, b is added to the task's main background knowledge B, and q is added to the appropriate example set. Otherwise, it is discarded.

Synthesizer Outputs (Alg. 1, lines 14–17). For each task, the synthesizer generates three outputs: (1) the *latent ground-truth rule* $R^\star$; (2) a *validation program*, an executable logic program encoding (B, E^+, E^-) for automatic evaluation; and (3) an

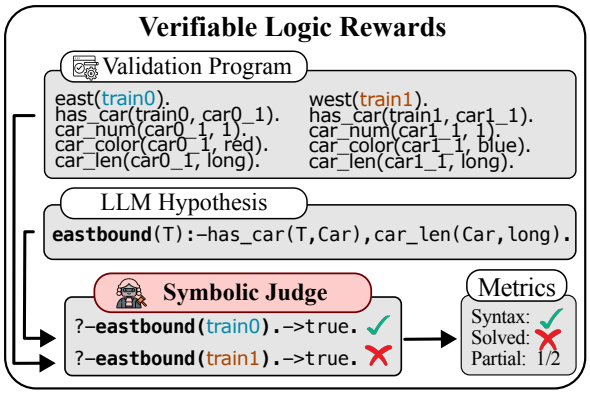

Figure 2: Verifiable Logic Rewards: A candidate hypothesis is evaluated by executing it against the validation program. It outputs three metrics: syntactic validity (binary), perfect task completion (binary), and a partial score for the fraction of correctly classified examples.

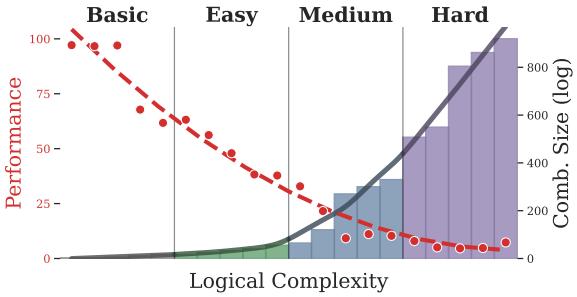

Figure 3: Overview of **SLR-BENCH**: The benchmark curriculum spans from basic to hard tasks with increasing logical and combinatorial complexity (bars, right y-axis). As logical complexity increases, Model performance (red, left y-axis) declines, highlighting current LLMs' limitations on more challenging reasoning tasks.

instruction prompt presenting the task in natural language or Prolog, ready for LLM input. See App. Fig.6 for an example synthesis run.

3.3 Training and Evaluation

The final stage, shown in Fig. 1 (right), uses the synthesized task to evaluate and train models.

VERIFIABLE LOGIC REWARDS. The SymbolicJudge is a core component of SLR, used for both training and evaluation. It deterministically assesses candidate hypotheses for logic tasks by executing them against a validation program and providing verifiable logic rewards (see Fig. 2). Specifically, it checks whether all positive examples (E^+) are entailed and all negative examples (E^-) are not entailed, producing three types of feedback:

SYNTAX SCORE: A binary score (0 or 1) that indicates whether the candidate hypothesis H is a syntactically valid Prolog rule. If H is invalid, subsequent scores are set to 0.

OVERALLSCORE: Indicates perfect task completion. It returns 1 if the hypothesis H, in conjunction with the background knowledge B entails all positive examples (E^+) and refutes all negative examples (E^-). Otherwise, the score is 0.

$$\begin{aligned}\text{OVERALLSCORE}_{B,E^+,E^-}(H) = [\![\\ \forall q \in E^+ : (B \cup H) \models q \quad \wedge \\ \forall q \in E^- : (B \cup H) \not\models q]\!] \in \{0,1\}\end{aligned}$$

Where $[\![\cdot]\!]$ is the Iverson bracket, evaluating to 1 if the condition inside is true, and 0 otherwise.

PARTIALSCORE: The proportion of examples (from $E^+ \cup E^-$) that are correctly classified by H given B, providing a continuous measure of progress even if the task is not perfectly solved.

$$\text{PARTIALSCORE}_{B,E^+,E^-}(H) = \frac{\sum_{q\in E^+}[\![(B\cup H)\models q]\!] + \sum_{q\in E^-}[\![(B\cup H)\not\models q]\!]}{|E^+ \cup E^-|}$$

The numerator counts correctly entailed positives and refuted negatives, and the denominator normalizes the score to $[\![0,1]\!]$.

Model Evaluation. SLR streamlines the creation of logical reasoning benchmark datasets for systematic model evaluation. By specifying various combinations of task language and configuration, users can generate diverse ILP tasks that span a broad range of domains, prompt styles, and reasoning complexities. Each synthesized task comprises a natural language prompt, the corresponding ground-truth rule, and an executable validation program for assessment by the Symbolic Judge.

Model Training. SLR enables automated training loops, with model feedback available in two flavors. For supervised fine-tuning (SFT), the ground-truth rule $R^\star$ serves as the training target, allowing for cross-entropy loss updates based on predicted rules. For reinforcement learning, the symbolic judge provides verifiable rewards (RLVR) to guide the model's policy updates. This cohesive, automated pipeline enables scalable generation, evaluation, and training of LLMs, facilitating systematic advances in logical reasoning capabilities.

Novelty and Systematic Generalization. As SLR is fully synthetic, overlap with existing data is statistically negligible, making it robust against

Instruction & Task Prompt

```
You are a train classifier who is observing trains that are traveling either east- or westbound. Each train is
composed of one or more cars, and each car is characterized by a set of properties, represented as ground atoms over
a fixed set of predicates. The direction (eastbound or westbound) of a train is to be determined from its composition.
To describe the trains we define a set of predicates and grounding domains:

'has_car(Train, Car)': Specifies that 'Car' is part of the train 'Train'.
'car_num(Car, CarNumber)': Specifies the position of the car within its train. 'CarNumber' is a positive integer.
'car_color(Car, Color)': Specifies the color of the car. 'Color' can be 'red', 'blue', 'green', 'yellow', or 'white'.
'car_len(Car, Length)': Specifies the length of the car. 'Length' can be either 'short' or 'long'.
'has_wall(Car, WallType)': Specifies the wall type of a car. 'WallType' can be either 'full' or a 'railing'.

You are provided with positive and negative examples in the form of eastbound(t) or westbound(t) for each train t,
together with background knowledge consisting of ground facts over the above predicates which describe its composition.

eastbound(train0).                      westbound(train1).
has_car(train0, car0_1).                has_car(train1, car1_1).
car_num(car0_1, 1).                     car_num(car1_1, 1).
car_color(car0_1, red).                 car_color(car1_1, red).
car_len(car0_1, long).                  car_len(car1_1, short).
has_wall(car0_1, railing).              has_wall(car1_1, railing).

Your task is to formulate a hypothesis, i.e. a prolog rule of the form 'eastbound(Train) :- Body.' that correctly
distinguishes eastbound from westbound trains. The hypothesis must be true for all positive examples (i.e., eastbound
trains) and false for all negative examples (i.e., westbound trains). Aim to find the shortest correct rule, that is,
one that uses the fewest possible body literals subject to the prior constraints. Your rule must use only  predicates
defined above and must perfectly separate eastbound from westbound trains.
```

Ground-Truth Rule

```
eastbound(Train) :- has_car(Train,Car), car_len(Car,long)
```

Figure 4: Illustrative **prompt** and **ground-truth rule** generated by SLR (Level 1, SLR-BENCH). Language ($\mathcal{L}$): 5 predicates, 1 car variable per train. Task configuration (Θ): $\kappa = (1, 1)$ (one positive and one negative example); B_π = mirror; $R_{\text{len}} = 1$; R_{sample} = uniform. The prompt provides the full ILP instance, including background B, positive/negative examples (E^+, E^-), and natural-language instructions for the learning task.

data leakage and memorization. By default, test set ground-truth rules ($R^\star$) are excluded from the training set, guaranteeing that inputs and outputs at test time are completely novel to the model. This enables assessing whether the model is capable of systematically generalizing to entirely new rules.

4 SLR-BENCH: Instantiating SLR

With SLR-BENCH, we instantiate SLR as a 20-level curriculum of logical reasoning tasks with increasing complexity (see Fig. 3). Each level specifies its own language $\mathcal{L}$ and configuration θ, producing a total of 19k generated reasoning tasks. Each level contains 1k train[1], 10 eval, and 50 test samples. Each task comes with (i) a generated latent ground-truth rule, (ii) the corresponding validation program, and associated instruction prompt for the task. An illustrative example for prompts and ground-truth rules can be found in Fig. 4.

Design rationale. The logic task is inspired by the V-LoL *trains* domain (Helff et al., 2025; Michalski, 1980; Mitchell, 1997), chosen for three main reasons. First, its hierarchical object structure (trains → cars → attributes) naturally gives rise to first-order rules that are far richer than simple lookups, yet more tractable than general theorem proving. Second, every attribute has a small, discrete grounding domain, which allows us to measure and control the complexity of the problem precisely. Third, we ensure expandability and novelty, as SLR-BENCH is fully synthetic and programmatically generated.

Languages. Each curriculum level is parameterized by level-specific language $\mathcal{L}$, detailed in App. A.1. The vocabulary includes mutually exclusive class labels `eastbound` and `westbound`, which serve as the targets for classification tasks. Additionally, the vocabulary includes five predicates (`has_car`, `car_num`, `car_color`, `car_len`, and `has_wall`) with their respective grounding domains specified in App. A.1. As curriculum levels increase in complexity, the vocabulary expands monotonically by introducing new predicates and grounding domains selected from a predefined set. These include categorical predicates such as `has_roof`, `has_payload`, `has_window`, `car_type`, and numerical predicates `load_num`, `has_wheel`, `passenger_num`. Semantic coherence is ensured by constraining predicate groundings to valid combinations (e.g., only colors as arguments for `car_color`), and by enforcing mutually exclusive constraints across predicates (e.g., passenger cars cannot carry payloads).

[1]The train sets of levels 1-3 are smaller (26, 234, and 793), due to the limited number of different tasks available

Table 2: **SLR-BENCH Learning Curriculum**: The table details how both language and task configuration are systematically increased throughout the curriculum levels. Notably, higher levels involve richer problems with more constants and predicates, larger problems, longer rules, and a transition from mirror to uniform and LLM-guided sampling. The final column reports the approximate combinatorial size of unique tasks available at each level.

Stage	Basic					Easy					Medium					Hard				
Levels	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Language																				
#Const.	1	1	1	2	2	2	2	2-3	2-3	2-3	2-4	2-4	4-6	4-6	4-6	5-6	5-6	5-6	5-6	5-6
#Pred.	5	5	5	5	5	5	6	6	6	7	7	9	9	9	9	10	10	12	12	12
Task Config.																				
κ	2	2	4	4	6	6	6	8	10	12	14	16	18	20	22	24	26	28	30	32
B_π	$\mathcal{M}$	$\mathcal{M}$	$\mathcal{M}$	$\mathcal{M}$	$\mathcal{M}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$
R_{len}	1	1-2	1-2	1-2	1-2	1-2	1-2	1-2	2-3	2-3	2-3	3-4	3-4	4-5	4-5	4-5	4-5	4-5	5	5
R_{sample}	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$	$\mathcal{U}/L$
Comb. Size	10^3	10^3	10^5	10^{10}	10^{16}	10^{16}	10^{24}	10^{32}	10^{40}	10^{55}	10^{65}	10^{120}	10^{271}	10^{300}	10^{330}	10^{507}	10^{549}	10^{805}	10^{861}	10^{919}

$\mathcal{M}$: mirror sampling $\mathcal{U}$: uniform sampling L: LLM-guided generation

Task configs. Each curriculum level is parameterized by level-specific settings of θ, summarized in App. Tab.2 and supplied directly to the synthesizer (Alg.1). Problem size (κ) increases steadily across levels, maintaining an equal balance of positive and negative samples. Levels 1–5 use a *mirror* sampling policy for background knowledge, generating simple, nearly identical east- and westbound trains that differ only in ground atoms relevant to $R^\star$. From level 6 onward, the background is sampled uniformly from the filtered Herbrand base, increasing diversity. Rule generation is uniform for the basic levels; from level 6, 30% of rules are LLM-guided, introducing greater variety in variables, arithmetic, recursion, and more.

Curriculum. SLR-BENCH comprises 20 levels across four tiers: *basic*, *easy*, *medium*, and *hard*. Each level systematically increases complexity by expanding task size (κ), adding new car constants and predicates, lengthening rules, and varying both the background knowledge and rule sampling policy; see App.Sec.A, Tab. 2. As a result, the combinatorial space of possible tasks grows exponentially, and later levels become progressively harder and require deeper reasoning beyond surface cues.

Intended Use. SLR-BENCH is designed for two complementary purposes. (1) As a *static* benchmark, it enables fine-grained evaluation of an LLM's reasoning abilities across tasks of increasing logical complexity. It is also easily extensible to accommodate future improvements in model capabilities. (2) As a *dynamic* curriculum, it serves as a training backbone, supplying structured reasoning tasks and feedback to enhance reasoning in both conventional and reasoning LLMs.

5 LLMs Can't Do Induction at Scale

We evaluate and train LLMs on SLR-BENCH, assessing reasoning, syntactic correctness, and computational efficiency across four difficulty levels: *basic*, *easy*, *medium*, and *hard*. Our analysis highlights key trends, common failure modes, and the effectiveness of curriculum-based logic-tuning.

Training Setup. We investigate how LLMs benefit from curriculum training on SLR-BENCH with SFT (for more details see App. Sec. G). To prevent data leakage, we ensure that no prompts or rules from the test set are included in the training set.

Evaluation Setup. All models are evaluated in a zero-shot setting using SLR-BENCH prompts, with a single attempt per task (pass@1). We report the following metrics:

(i) *Logical Reasoning Level (LRL):* The cumulative model score over all curriculum levels L, where $\#\text{solved}_\ell$ and $\#\text{tasks}_\ell$ denote the number of solved and total tasks at level ℓ, respectively:

$$\text{LRL} = \sum_{\ell=1}^{L} \frac{\#\text{solved}_\ell}{\#\text{tasks}_\ell}$$

(ii) *Syntax Score:* The proportion of predicted logic rules that are syntactically valid.

(iii) *Logical-Reasoning Accuracy:* The fraction of correct solutions per complexity tier.

(iv) *Compute:* The aggregate completion tokens and computational cost for each the models.

For further details on downstream evaluations, see App. Sec. G, and for pricing, refer to Tab. 5.

5.1 Analysis and Key Findings

In the following, we highlight downstream gains from training via SLR curriculum learning (Tab. 3).

Table 3: **Curriculum Learning and Generalization.** Benchmark scores (↑%) for base and SLR-tuned models (Llama3.1-8b-it) on SLR-BENCH and downstream benchmarks; LRL measuring cumulative curriculum progress. The tuned model surpasses the baseline across all curriculum stages, while generalizing to other reasoning tasks.

*Curriculum Learning (*SLR-BENCH*)*

	LRL (↑0-20)	Syntax(↑%)	Basic(↑%)	Easy(↑%)	Medium(↑%)	Hard(↑%)
Llama-8b-it	3.6	99	61	10	1	0
Llama-8b-it-SLR	**8.8** (+5.2)	**100** (+1)	**96** (+35)	**56** (+46)	**20** (+19)	**5** (+5)

Downstream Reasoning Performance

	MMLU (↑%)	MMLU-Stats (↑%)	MMLU-CS (↑%)	MMLU-ML (↑%)	LogiQA (↑%)	LogiQA2 (↑%)
Llama-8b-it	63.3	42.6	61.0	49.1	30.1	34.3
Llama-8b-it-SLR	**66.1** (+2.8)	**59.7** (+17)	**75.0** (+14)	**52.7** (+3.6)	**31.0** (+0.9)	**39.4** (+5.2)
	GPQA (↑%)	GPQA-Ext. (↑%)	GPQA-Dia. (↑%)	ARC-Easy (↑%)	ARC (↑%)	HellaSwag (↑%)
Llama-8b-it-	31.7	26.9	21.7	81.4	52.7	57.4
Llama-8b-it-SLR	**32.8** (+1.1)	**33.0** (+6.1)	**28.3** (+6.6)	**82.8** (+1.4)	**54.6** (+1.9)	**58.9** (+1.5)

Next, we benchmark SOTA conventional and reasoning LLMs on SLR-BENCH (Tab. 4), showcasing the trade-offs of test-time compute (Fig. 5).

SLR Boosts Downstream Reasoning. Curriculum learning on SLR-BENCH yields substantial improvements in both in-domain and downstream reasoning (see Tab 3). SLR-tuned models outperform all conventional LLMs on SLR-BENCH (*cf.* Tab.4) and surpass reasoning LLMs such as Gemini-2.0-flash-thinking, while using far fewer inference tokens and compute resources. On popular reasoning benchmarks, SLR delivers gains on logic-intensive tasks, e.g., on MMLU High School Statistics (+17), Computer science (+14), and Machine learning (+3.6) (Hendrycks et al., 2021), as well as notable improvements on LogicQA (Liu et al., 2020), LogicQA2 (Liu et al., 2023), ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), GPQA, and GPQA-Extended, GPQA-Diamond (Rein et al., 2024). These consistent gains across curriculum levels and downstream benchmarks demonstrate that curriculum learning with SLR not only enhances in-domain reasoning but also generalizes effectively to diverse reasoning tasks.

Curriculum Levels Modulate Task Complexity: LLMs Break Down as Complexity Increases. SLR-BENCH creates a controlled gradient in logical complexity as model performance steadily declines throughout the curriculum levels (*cf.* Fig. 3, Tab. 4). Most models readily solve the *basic* levels. However, base LLMs already struggle on the *easy* tasks, solving less than half. Reasoning LLMs provide improved performance, though they incur substantial drops at the *medium* levels. Only a few reasoning LLMs manage to solve more than half the *medium* tasks, yet none on *hard*. This pattern is also reflected in the LRL score, empirically indicating how far each model can progress before performance collapses.

Reasoning Remains Challenging; Syntax Not. Base LLMs reliably generate syntactically valid rules, reflected in their high syntax scores (see Tab. 4). Reasoning models exhibit slightly lower scores, particularly on more complex reasoning tasks, where longer outputs can lead to invalid or missing responses. Nonetheless, the primary barrier to higher performance is semantic, as reflected by the gap between syntax and LRL in Tab. 4.

Scaling Test-time Compute Improves Reasoning, but Returns Diminish and Costs Escalate. Reasoning LLMs clearly outperform the base models; not even the best base model is able to match any of the reasoning LLMs (*cf.* Tab. 4). This, however, comes at a steep cost as moving from GPT-4o to *o3* doubles accuracy, but considerably increases the number of completion tokens (1777%) and thus the computational costs (1034%). Moreover, scaling test-time compute on the same model also boosts overall performance, but does not guarantee higher accuracy across all tasks. For example, while *o4-mini-high* typically outperforms *o4-mini* (LRL: 12.8 vs. 12.3), it underperforms on the medium complexity tier (40% vs. 52%). This plateau effect demonstrates that, beyond a certain threshold, additional compute may yield diminishing or even negative returns.

Scaling Model Parameters Brings Limited Gains in Logical Reasoning. Increasing model size yields only marginal gains in logical reasoning.

Table 4: **SLR-BENCH Leaderboard**. We report the models' Logical Reasoning Level (LRL), syntax score, stage-specific logical reasoning accuracy (basic, easy, medium, hard), total completion tokens, and inference cost. Higher LRL and accuracy indicate superior logical reasoning; lower compute, greater efficiency. Performance drops as complexity increases, while Reasoning LLMs (orange) consistently outperform conventional LLMs (blue).

	LRL	Syntax	Logical-Reasoning Acc. (%)↑				Total Compute	
Model	(↑0-20)	Score (↑%)	Basic	Easy	Medium	Hard	Tokens (↓M)	Costs (↓$)
o3	**15.5**	80	**99**	93	**74**	**45**	4.30	207.24
o4-mini-high	12.8	88	98	**96**	40	21	4.62	24.24
o4-mini	12.3	86	93	88	52	13	3.98	21.43
o1	11.9	68	92	89	41	15	5.19	364.72
o3-mini	11.6	75	97	90	37	7	4.73	24.71
o4-mini-low	10.3	91	91	81	25	9	0.77	7.26
o1-mini	10.1	95	97	82	20	3	3.65	19.98
R1-Llama-70B[2]	8.8	75	98	67	8	4	11.61	5.33
Gemini-thinking[1]	8.6	83	93	65	13	1	—	—
gpt-4.5-prev	7.3	**100**	94	47	5	1	0.37	576.40
gpt-4o	6.2	**100**	93	29	2	0	**0.26**	20.03
Llama-3.3-70B	5.9	**100**	94	24	0	0	0.48	0.81
gpt-4-turbo	5.4	**100**	89	18	2	0	0.41	81.30
Llama 3.1-8B	3.6	99	61	10	1	0	1.96	0.20
Llama 3.2-3B	0.7	70	13	1	0	0	2.10	**0.16**
Llama 3.2-1B	0.0	34	0	0	0	0	5.30	0.23

[1]Gemini-2.0-flash-thinking-exp-01-21 [2]DeepSeek-R1-Distill-Llama-70B — information not available

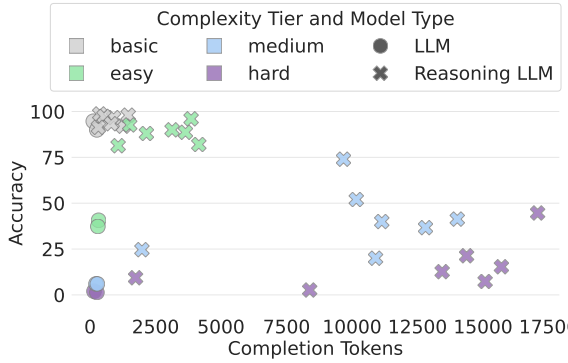

Figure 5: **Compute–Performance Trade-Off**. Reasoning LLMs achieve higher accuracy than base LLMs but require more compute. While more complex tasks typically demand more completion tokens, increased compute does not always translate to higher accuracy.

While larger models like Llama-3-70B and GPT-4.5-prev generally outperform their smaller counterparts, returns diminish as improvements are increasingly modest (see Tab. 4). As even the largest base models still fall short of the reasoning models, a capability gap remains that suggests that scaling model parameters alone does not guarantee substantial advances in logical reasoning capabilities.

Compute Increases with Task Complexity. As reasoning tasks become more complex, reasoning LLMs require more tokens to solve the tasks, leading to increased financial costs (see Fig. 5). These increased demands impose limits on scaling reasoning through increased test-time compute.

6 Conclusion and Future Direction

In this work, we introduced SLR, a fully automated and scalable framework for synthesizing logical reasoning benchmarks with verifiable rewards provided by logic programs. Our instantiation, SLR-BENCH, offers a 20-level curriculum spanning 19k tasks with increasing logical complexity.

Our evaluations reveal that while current LLMs readily produce syntactically valid logic rules, robust logical reasoning remains elusive for conventional LLMs, especially as task complexity scales. Scaling model parameters yields only limited gains. Reasoning LLMs, aided by increased test-time compute, close part of this gap, albeit at significant computational costs.

Notably, curriculum learning on SLR-BENCH significantly boosts both in-domain and downstream reasoning. Our SLR-tuned Llama3-8B not

only outperforms all conventional LLMs on SLR-BENCH, but also surpasses several SOTA reasoning LLMs at a fraction of their inference costs. Furthermore, we observe improved reasoning capabilities across a wide range of established benchmarks, underscoring the effectiveness of curriculum-based logic-tuning for downstream reasoning tasks.

Looking ahead, SLR paves the way for several promising research directions, including the integration of reinforcement learning for reasoning LLMs using SLR, expanding into richer logical domains, benchmarking neuro-symbolic and interactive reasoning systems, and moving toward higher-order logic tasks like causal inference. Ultimately, SLR provides a flexible and extensible resource for probing and advancing the frontiers of logical reasoning in the next generation of LLMs.

7 Limitations

While SLR and SLR-BENCH provide a scalable testbed for logical reasoning, there remain many opportunities for further enrichment. Although SLR-BENCH currently applies SLR to the train domain with a single rule, the framework is readily extensible to multiple more complex, multi-rule reasoning scenarios and to entirely different domains. Our current focus on first-order, function-free Horn clauses enables systematic benchmark creation and evaluation; future instantiations could expand towards higher-order logic or probabilistic reasoning. While synthetic task generation comes with many benefits, such as ensuring novelty and precise control, it makes it difficult to incorporate real-world diversity and ambiguity. Our symbolic judge provides deterministic, discrete scoring and could potentially be enhanced to also recognize partial solutions, syntactically invalid rules, or natural language formulations. Overall, these points highlight the flexibility of our framework and outline promising directions for broadening its reach and impact.

Acknowledgements

We acknowledge support of the hessian.AI Innovation Lab (funded by the Hessian Ministry for Digital Strategy and Innovation), the hessian.AISC Service Center (funded by the Federal Ministry of Education and Research, BMBF, grant No 01IS22091), and the Centre for European Research in Trusted AI (CERTAIN). Further, this work benefited from the ICT-48 Network of AI Research Excellence Center "TAILOR" (EU Horizon 2020, GA No 952215), the Hessian research priority program LOEWE within the project WhiteBox [GA No LOEWE/2/13/519/03/06.001(0010)/77], the HMWK cluster projects "Adaptive Mind" and "Third Wave of AI", and from the NHR4CES. This work was supported by the Priority Program (SPP) 2422 in the subproject "Optimization of active surface design of high-speed progressive tools using machine and deep learning algorithms" funded by the German Research Foundation (DFG). Further, this work was funded by the European Union (Grant Agreement no. 101120763 - TANGO) as well as the AlephAlpha Collaboration lab 1141. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HaDEA). Neither the European Union nor the granting authority can be held responsible for them.

Broader Impact

SLR and SLR-BENCH provide a scalable, reproducible foundation for evaluating and advancing logical reasoning in AI without relying on human annotation. By enabling robust measurement and targeted training, our framework supports progress in areas such as scientific discovery, program synthesis, and trustworthy AI. However, as LLMs acquire deeper logical competence, the risk of dual-use knowledge increases, enabling beneficial applications but also the potential for misuse, such as generating deceptive arguments or bypassing safety mechanisms. We urge responsible use and active consideration of ethical risks as these capabilities advance.

References

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Victor Bursztyn, David Demeter, Doug Downey, and Larry Birnbaum. 2022. Learning to perform complex

tasks through compositional fine-tuning of language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1676–1686, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Andrew Cropper and Sebastijan Dumančić. 2022. Inductive logic programming at 30: A new introduction. *J. Artif. Int. Res.*, 74.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Quentin Delfosse, Jannis Blüml, Fabian Tatai, Théo Vincent, Bjarne Gregori, Elisabeth Dillies, Jan Peters, Constantin Rothkopf, and Kristian Kersting. 2025. Deep reinforcement learning agents are not even close to human intelligence. *Preprint*, arXiv:2505.21731.

Siqi Fan, Peng Han, Shuo Shang, Yequan Wang, and Aixin Sun. 2025. Cothink: Token-efficient reasoning via instruct models guiding reasoning models. *Preprint*, arXiv:2505.22017.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. 2024. Open llm leaderboard v2. `https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard`.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024a. The language model evaluation harness.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024b. The language model evaluation harness.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, and 7 others. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Lukas Helff, Wolfgang Stammer, Hikaru Shindo, Devendra Singh Dhami, and Kristian Kersting. 2025. V-lol: A diagnostic dataset for visual logical learning. *Journal of Data-centric Machine Learning Research*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K. Jain, Virginia Aglietti, Disha Jindal, Peter Chen, Nishanth Dikkala, Gladys Tyen, Xin Liu, Uri Shalit, Silvia Chiappa, Kate Olszewska, Yi Tay, Vinh Q. Tran, Quoc V. Le, and Orhan Firat. 2025. Big-bench extra hard. *Preprint*, arXiv:2502.19187.

Jiin Kim, Byeongjun Shin, Jinha Chung, and Minsoo Rhu. 2025. The cost of dynamic reasoning: Demystifying ai agents and test-time scaling from an ai infrastructure perspective. *Preprint*, arXiv:2506.04301.

Aida Kostikova, Zhipin Wang, Deidamea Bajri, Ole Putz, Benjamin Paassen, and Steffen Eger. 2025. Lllms: A data-driven survey of evolving research on limitations of large language models.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Bill Yuchen Lin. 2024. Zeroeval: A unified framework for evaluating language models.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. 2025. Zebralogic: On the scaling limits of llms for logical reasoning.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *Preprint*, arXiv:2406.04770.

Hanmeng Liu, Jian Liu, Leyang Cui, Zhiyang Teng, Nan Duan, Ming Zhou, and Yue Zhang. 2023. Logiqa 2.0—an improved dataset for logical reasoning in natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2947–2962.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3622–3628. International Joint Conferences on Artificial Intelligence Organization.

Junteng Liu, Yuanxiang Fan, Zhuo Jiang, Han Ding, Yongyi Hu, Chi Zhang, Yiqi Shi, Shitong Weng, Aili Chen, Shiqi Chen, Yunan Huang, Mozhi Zhang, Pengyu Zhao, Junjie Yan, and Junxian He. 2025. Synlogic: Synthesizing verifiable reasoning data at scale for learning logical reasoning and beyond. *Preprint*, arXiv:2505.19641.

John W Lloyd. 2012. *Foundations of logic programming*. Springer Berlin, Heidelberg.

Man Luo, Shrinidhi Kumbhar, Ming shen, Mihir Parmar, Neeraj Varshney, Pratyay Banerjee, Somak Aditya, and Chitta Baral. 2024. Towards logiglue: A brief survey and a benchmark for analyzing logical reasoning capabilities of language models. *Preprint*, arXiv:2310.00836.

Kaijing Ma, Xinrun Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, Wenhao Huang, and Ge Zhang. 2024. Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. *Preprint*, arXiv:2410.06526.

Meta, Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, and 543 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Ryszard S. Michalski. 1980. Pattern recognition as rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(4):349–361.

Tom Michael Mitchell. 1997. Machine learning, international edition. In *McGraw-Hill Series in Computer Science*.

Philipp Mondorf and Barbara Plank. 2024. Liar, liar, logical mire: A benchmark for suppositional reasoning in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7114–7137, Miami, Florida, USA. Association for Computational Linguistics.

Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2024. Enhancing reasoning capabilities of llms via principled synthetic logic corpus. In *Annual Conference on Neural Information Processing Systems*.

Stephen Muggleton and Luc de Raedt. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629–679.

OpenAI. 2025. Openai o3 and o4-mini system card. https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Tafjord Oyvind, Dalvi Bhavana, and Clark Peter. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634. Association for Computational Linguistics.

Bhrij Patel, Souradip Chakraborty, Wesley A. Suttle, Mengdi Wang, Amrit Singh Bedi, and Dinesh Manocha. 2024a. Aime: Ai system optimization via multiple llm evaluators. *Preprint*, arXiv:2410.03131.

Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. 2024b. Multi-logieval: Towards evaluating multi-step logical reasoning ability of large language models. *Preprint*, arXiv:2406.17169.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, Adam Khoja, Ryan Kim, Richard Ren, Jason Hausenloy, Oliver Zhang, Mantas Mazeika, and 1090 others. 2025. Humanity's last exam. *Preprint*, arXiv:2501.14249.

Shanghaoran Quan, Jiaxi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, Yibo Miao, Yunlong Feng, and 1 others. 2025. Codeelo: Benchmarking competition-level code generation of llms with human-comparable elo ratings. *arXiv preprint arXiv:2501.01257*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China. Association for Computational Linguistics.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *ACL (Findings)*, pages 13003–13051.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv: Artificial Intelligence*.

Tim Woydt, Moritz Willig, Antonia Wüst, Lukas Helff, Wolfgang Stammer, Constantin A. Rothkopf, and Kristian Kersting. 2025. Fodor and pylyshyn's legacy – still no human-like systematic compositionality in neural networks.

Antonia Wüst, Tim Tobiasch, Lukas Helff, Inga Ibs, Wolfgang Stammer, Devendra S Dhami, Constantin A Rothkopf, and Kristian Kersting. 2025. Bongard in wonderland: Visual puzzles that still make ai go mad? In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2024. On memorization of large language models in logical reasoning. *Preprint*, arXiv:2410.23123.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *Preprint*, arXiv:2502.14768.

Nathan Young, Qiming Bao, Joshua Bensemann, and Michael Witbrock. 2022. AbductionRules: Training transformers to explain unexpected inputs. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 218–227, Dublin, Ireland. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Yujun Zhou, Jiayi Ye, Zipeng Ling, Yufei Han, Yue Huang, Haomin Zhuang, Zhenwen Liang, Kehan Guo, Taicheng Guo, Xiangqi Wang, and Xiangliang Zhang. 2025. Dissecting logical reasoning in llms: A fine-grained evaluation and supervision study. *Preprint*, arXiv:2506.04810.

A Task Specification

Each task in SLR-BENCH is precisely governed by a combination of language features and task configuration parameters, enabling fine-grained control over complexity and diversity. A task specification comprises two main components: (i) the logical language, which determines the set of predicates and argument types available, and (ii) the task configuration, which defines structural aspects of the task such as problem size, background knowledge sampling, rule length, sampling strategy, and the combinatorial space of realizable tasks. Tab. 2 details the curriculum's level-wise specifications, showing how both language elements and the task config to create the individual levels.

A.1 Language

Predicates and Types. SLR defines a flexible, extensible vocabulary to support the systematic generation and evaluation of logical reasoning tasks. The primary predicate signatures and their argument types used in SLR-BENCH are:

$$\texttt{eastbound}(\mathit{TRAIN})$$
$$\texttt{westbound}(\mathit{TRAIN})$$
$$\texttt{has\_car}(\mathit{TRAIN}, \mathit{CAR})$$
$$\texttt{car\_num}(\mathit{CAR}, \mathit{NUM})$$
$$\texttt{car\_color}(\mathit{CAR}, \mathit{COLOR})$$
$$\texttt{car\_len}(\mathit{CAR}, \mathit{LEN})$$
$$\texttt{has\_wall}(\mathit{CAR}, \mathit{WALL})$$
$$\texttt{has\_roof}(\mathit{CAR}, \mathit{ROOF})$$
$$\texttt{has\_payload}(\mathit{CAR}, \mathit{LOADS})$$
$$\texttt{load\_num}(\mathit{CAR}, \mathit{NPAY})$$
$$\texttt{has\_wheel}(\mathit{CAR}, \mathit{WHEELS})$$
$$\texttt{has\_window}(\mathit{CAR}, \mathit{WINDOW})$$
$$\texttt{car\_type}(\mathit{CAR}, \mathit{CTYPE})$$

Grounding Domains. Each argument type is grounded in a finite set of discrete constants:

$$\begin{aligned}
\mathit{NUM} &::= [\![0\text{–}9]\!]+ \\
\mathit{CAR} &::= [\![0\text{–}9]\!]+ \\
\mathit{COLOR} &::= \texttt{red} \mid \texttt{blue} \mid \texttt{green} \mid \\
&\quad\ \texttt{yellow} \mid \texttt{white} \\
\mathit{LEN} &::= \texttt{short} \mid \texttt{long} \\
\mathit{WALL} &::= \texttt{full} \mid \texttt{railing} \\
\mathit{ROOF} &::= \texttt{roof_foundation} \mid \texttt{solid_roof} \mid \\
&\quad\ \texttt{braced_roof} \mid \texttt{peaked_roof} \mid \texttt{none} \\
\mathit{WHEELS} &::= \texttt{2} \mid \texttt{3} \\
\mathit{LOADS} &::= \texttt{blue_box} \mid \texttt{golden_vase} \mid \\
&\quad\ \texttt{barrel} \mid \texttt{diamond} \mid \texttt{metal_pot} \mid \\
&\quad\ \texttt{oval_vase} \mid \texttt{none} \\
\mathit{NPAY} &::= \texttt{0} \mid \texttt{1} \mid \texttt{2} \mid \texttt{3} \\
\mathit{WINDOW} &::= \texttt{full} \mid \texttt{half} \mid \texttt{none} \\
\mathit{CTYPE} &::= \texttt{passenger} \mid \texttt{freight} \mid \texttt{mixed} \\
\mathit{NPAX} &::= [\![0\text{–}9]\!]
\end{aligned}$$

Grammar Constraints. Predicates are only instantiated with semantically compatible constant types. For example, $\texttt{car\_color}(\cdot,\cdot)$ only takes car objects and color constants as arguments; ill-typed facts are excluded during synthesis.

B Logic Rewards Provided by the Symbolic Judge

The SYMBOLICJUDGE computes verifiable logic rewards for a candidate hypothesis H against a given background knowledge base B and sets of positive (E^+) and negative (E^-) examples. These rewards are used for both evaluation and model training, and they include three distinct metrics:

Syntax Validity Score: This binary score (0 or 1) indicates whether the candidate hypothesis H is a syntactically and semantically valid Prolog rule. This serves as a prerequisite check for further evaluation; if H is invalid, other scores are typically assigned 0.

OVERALLSCORE: This metric provides a binary indication of perfect task completion. It is 1 if and only if the hypothesis H, in conjunction with the background knowledge B, correctly entails all positive examples (E^+) and correctly refutes all

negative examples (E^-). Otherwise, the score is 0.

$$\begin{aligned}\textsc{OverallScore}_{B,E^+,E^-}(H) = [\![\\ \forall q \in E^+ : (B \cup H) \models q \quad \wedge \\ \forall q \in E^- : (B \cup H) \not\models q]\!] \in \{0, 1\}\end{aligned}$$

Where $[\![\cdot]\!]$ is the Iverson bracket, evaluating to 1 if the condition inside is true, and 0 otherwise.

PARTIALSCORE: This metric reflects the fraction of examples (from both E^+ and E^-) that are correctly classified by the candidate hypothesis H when combined with the background knowledge B. This provides a continuous signal of progress, even when the overall task is not perfectly completed.

$$\textsc{PartialScore}_{B,E^+,E^-}(H) = \frac{\sum_{q \in E^+} [\![(B \cup H) \models q]\!] + \sum_{q \in E^-} [\![(B \cup H) \not\models q]\!]}{|E^+ \cup E^-|}$$

Here, the numerator sums the count of correctly entailed positive examples and correctly refuted negative examples. The denominator is the total number of examples, ensuring the score is normalized between 0 and 1.

These metrics provide rich feedback for both discrete evaluation (e.g., for filtering valid rules) and continuous optimization (e.g., for guiding reinforcement learning agents), allowing for robust assessment of learned logical hypotheses.

C Compute Costs and Model Pricing

Compute Costs. Compute costs are reported as the total USD cost to run all prompts, based on publicly listed API prices as of 01.05.2025. Pricing ignores server-side token caching, as actual cache hit counts are unavailable. Table 5 summarizes per-model cost rates and API sources.

D Example: Task Synthesis Process

E LLM Guided Rule Generation

This section provides the prompt used for LLM-guided rule generation. The prompt was carefully designed to be both diverse and comprehensive, including a wide range of logical structures and Prolog features such as conjunction, disjunction, negation, recursion, aggregation, and pattern matching. By presenting the model with these varied and complex examples, we encourage the generation of challenging and realistic logic rules. This diversity is crucial for robust model generation of new rules, as it ensures that the LLM is exposed to representative samples of possible rule types encountered in real-world logic programming tasks.

1. Conjunction with Existential Quantification: There exists a red short car

There is at least one car that is both short and red.

```
eastbound(Train) :-
    has_car(Train, Car),
    car_color(Car, red),
    car_len(Car, short).
```

2. Disjunction: Some car is white or yellow.

At least one car is either white or yellow.

```
eastbound(Train) :-
    has_car(Train, Car),
    (car_color(Car, white) ; car_color(
    Car, yellow)).
```

3. Negation: The train does not contain any red cars

No car on the train is red.

```
eastbound(Train) :-
    \+ (has_car(Train, Car), car_color(
    Car, red)).
```

4. Inequality/Distinctness: Two cars must have different colors

There are at least two cars on the train with different colors.

```
eastbound(Train) :-
    has_car(Train, CarA),
    has_car(Train, CarB),
    CarA \= CarB,
    car_color(CarA, Color1),
    car_color(CarB, Color2),
    Color1 \= Color2.
```

5. Aggregation/Counting: There are more green cars than yellow cars

The train contains more green cars than yellow cars.

```
eastbound(Train) :-
    findall(Car, (has_car(Train, Car),
    car_color(Car, green)), Greens),
    findall(Car, (has_car(Train, Car),
    car_color(Car, yellow)), Yellows),
    length(Greens, G),
    length(Yellows, Y),
    G > Y.
```

Table 5: Model Pricing ($ per 1M tokens). API rates as of 01.05.2025.

Model	Model Tag	Input	Input(Cached)	Output	API
gpt-4.1	gpt-4.1-2025-04-14	2.00	0.50	8.00	OpenAI
gpt-4.1-mini	gpt-4.1-mini-2025-04-14	0.40	0.10	1.60	OpenAI
gpt-4.1-nano	gpt-4.1-nano-2025-04-14	0.10	0.025	0.40	OpenAI
gpt-4.5-preview	gpt-4.5-preview-2025-02-27	75.00	37.50	150.00	OpenAI
gpt-4o	gpt-4o-2024-08-06	2.50	1.25	10.00	OpenAI
gpt-4o-mini	gpt-4o-mini-2024-07-18	0.15	0.075	0.60	OpenAI
o1	o1-2024-12-17	15.00	7.50	60.00	OpenAI
o1-pro	o1-pro-2025-03-19	150.00		600.00	OpenAI
o3	o3-2025-04-16	10.00	2.50	40.00	OpenAI
o4-mini-low	o4-mini-2025-04-16	1.10	0.275	4.40	OpenAI
o4-mini	o4-mini-2025-04-16	1.10	0.275	4.40	OpenAI
o4-mini-high	o4-mini-2025-04-16	1.10	0.275	4.40	OpenAI
o3-mini	o3-mini-2025-01-31	1.10	0.55	4.40	OpenAI
o1-mini	o1-mini-2024-09-12	1.10	0.55	4.40	OpenAI
DeepSeek-R1	DeepSeek-R1-Distill-Llama-70B	0.10		0.40	OpenRouter
Internlm2-20b	Internlm2-20b	0.15		0.20	Estimated
Llama-3.2-3B	Llama-3.2-3B-Instruct	0.015		0.025	OpenRouter
Llama-3.1-8B	Llama-3.1-8B-Instruct	0.02		0.03	OpenRouter
Llama-3.3-70B	Llama-3.3-70B-Instruct	0.10		0.25	OpenRouter
Mixtral-8x7B	Mixtral-8x7B-Instruct-v0.1	0.24		0.24	OpenRouter
Qwen2-57B-A14B	Qwen2-57B-A14B-Instruct	0.70		0.70	Estimated
QwQ-32B	QwQ-32B-Preview	0.15		0.20	OpenRouter
CodeLlama-34b	CodeLlama-34b-Instruct-hf	0.776		0.776	TogetherAI
MetaTuned Llama	Llama-3.1-8B-Tuned-FFT	0.02		0.03	OpenRouter
MetaTuned Llama LoRA	Llama-3.1-8B-Tuned-LoRA	0.02		0.03	OpenRouter

6. Mutual Exclusion: Only one car is yellow; all others are not yellow

There is exactly one yellow car; all others are not yellow.

```
eastbound(Train) :-
    findall(Car, (has_car(Train, Car),
    car_color(Car, yellow)), [YellowCar
    ]),
    forall(
        (has_car(Train, Car), Car \=
    YellowCar),
        (car_color(Car, NotYellow),
        NotYellow \= yellow)
    ).
```

7. Uniqueness: No two cars have the same color

All cars have unique colors.

```
eastbound(Train) :-
    findall(Color, (has_car(Train, Car),
     car_color(Car, Color)), Colors),
    sort(Colors, UniqueColors),
    length(Colors, N),
    length(UniqueColors, N).
```

8. No-Other/Uniqueness: Only two cars in the train

Only two cars are present in the train.

```
eastbound(Train) :-
    findall(Car, has_car(Train, Car),
    Cars),
    length(Cars, 2).
```

9. Universal Quantification: Every full-wall car is long

All cars with a full wall must be long.

```
eastbound(Train) :-
    forall(
        (has_car(Train, Car), has_wall(
    Car, full)),
        car_len(Car, long)
    ).
```

10. Conditional Implication: All long cars are either red or blue

Every long car is either red or blue.

```
eastbound(Train) :-
    forall(
```

```
        (has_car(Train, Car), car_len(Car, long)),
        (car_color(Car, Color), (Color = red ; Color = blue))
    ).
```

11. Conditional Aggregation: All long cars are either red or blue

Every long car is either red or blue.

```
eastbound(Train) :-
    forall(
        (has_car(Train, Car), car_len(Car, long)),
        (car_color(Car, Color), (Color = red ; Color = blue))
    ).
```

12. Pattern Matching: All full-wall cars are white

Every full-wall car is white.

```
eastbound(Train) :-
    forall(
      (has_car(Train, Car), has_wall(Car, full)),
      car_color(Car, white)
    ).
```

13. Symmetry: Two cars are neighbors with same color

CarA and CarB are neighbors on the train and have the same color.

```
eastbound(Train) :-
    has_car(Train, CarA),
    has_car(Train, CarB),
    CarA \= CarB,
    car_num(CarA, N1),
    car_num(CarB, N2),
    (N2 =:= N1 + 1 ; N2 =:= N1 - 1),
    car_color(CarA, Color),
    car_color(CarB, Color).
```

14. Combinatorial Group: Exactly two short yellow cars

There are exactly two yellow cars, and both are short.

```
eastbound(Train) :-
    findall(Car, (has_car(Train, Car), car_color(Car, yellow), car_len(Car, short)), L),
    length(L, 2).
```

15. Recursion: At least one long car in the train

The train has at least one long car.

```
eastbound([Car|Cars]) :-
    car_len(Car, long)
    ;
    eastbound(Cars).
```

16. Existence of a Structure (Sublist Pattern Matching)

Exists three cars in sequence: Num, Num+1, Num+2, matching pattern.

```
eastbound(Train) :-
    has_car(Train, Car1), car_num(Car1, N),
    car_len(Car1, short),
    N2 is N+1, N3 is N+2,
    has_car(Train, Car2), car_num(Car2, N2), car_len(Car2, long),
    has_car(Train, Car3), car_num(Car3, N3), car_len(Car3, short).
```

17. Min/Max and Extremal Values

A short car followed by a long car followed by a short car, anywhere in the train.

```
eastbound(Train) :-
    findall(N, (has_car(Train, Car), car_num(Car, N)), Numbers),
    max_list(Numbers, Max),
    has_car(Train, LastCar),
    car_num(LastCar, Max),
    car_color(LastCar, white).
```

18. Subset/Superset Constraints

All full-wall cars are among the first three cars.

```
eastbound(Train) :-
    forall(
      (has_car(Train, Car), has_wall(Car, full)),
      (car_num(Car, N), N =< 3)
    ).
```

19. Projection/Aggregation Over Multiple Properties

All pairs of cars have different (color, length) tuples.

```
eastbound(Train) :-
    has_car(Train, CarA), has_car(Train, CarB), CarA \= CarB,
    car_color(CarA, ColA), car_len(CarA, LenA),
    car_color(CarB, ColB), car_len(CarB, LenB),
    (ColA \= ColB ; LenA \= LenB).
```

20. All-Different on Multiple Attributes

Enforce all car colors are different, AND all car numbers are different (car numbers are unique by assumption, but see structure).

```
eastbound(Train) :-
    findall(Color, (has_car(Train, Car), car_color(Car, Color)), Colors),
    sort(Colors, UniqueColors),
    length(Colors, N), length(UniqueColors, N).
```

F First-Order Logic Details

We revisit essential definitions of first-order logic that we follow in this paper. An FOL *Language* $\mathcal{L}$ is a tuple $(\mathcal{P}, \mathcal{A}, \mathcal{F}, \mathcal{V})$, where $\mathcal{P}$ is a set of predicates, $\mathcal{A}$ is a set o constants, $\mathcal{F}$ is a set of function symbols (functors), and $\mathcal{V}$ is a set of variables. A *term* is a constant, a variable, or a term that consists of a functor. A *ground term* is a term with no variables. We denote n-ary predicate $\mathtt{p}$ by $\mathtt{p}/n$. An *atom* is a formula $\mathtt{p}(\mathtt{t_1}, \ldots, \mathtt{t_n})$, where $\mathtt{p}$ is an n-ary predicate symbol and $\mathtt{t_1}, \ldots, \mathtt{t_n}$ are terms. A *ground atom* or simply a *fact* is an atom with no variables. A *literal* is an atom or its negation. A *positive literal* is just an atom. A *negative literal* is the negation of an atom. A *clause* is a finite disjunction ($\vee$) of literals. A *definite clause* is a clause with exactly one positive literal. If $A, B_1, \ldots, B_n$ are atoms, then $A \vee \neg B_1 \vee \ldots \vee \neg B_n$ is a definite clause. We write definite clauses in the form of A :- $B_1, \ldots, B_n$. Atom A is called the *head*, and set of negative atoms $\{B_1, \ldots, B_n\}$ is called the *body*. We call definite clauses by *rules* for simplicity in this paper. An atom is an atomic *formula*. For formula F and G, $\neg F$, $F \wedge G$, and $F \vee G$ are also formulas. *Interpretation* of language $\mathcal{L}$ is a tuple $(\mathcal{D}, \mathcal{I}_\mathcal{A}, \mathcal{I}_\mathcal{F}, \mathcal{I}_\mathcal{P})$, where $\mathcal{D}$ is the domain, $\mathcal{I}_\mathcal{A}$ is the assignments of an element in $\mathcal{D}$ for each constant $\mathtt{a} \in \mathcal{A}$, $\mathcal{I}_\mathcal{F}$ is the assignments of a function from $\mathcal{D}^n$ to $\mathcal{D}$ for each n-ary function symbol $\mathtt{f} \in \mathcal{F}$, and $\mathcal{I}_\mathcal{P}$ is the assignments of a function from $\mathcal{D}^n$ to $\{\top, \bot\}$ for each n-ary predicate $\mathtt{p} \in \mathcal{P}$. For language $\mathcal{L}$ and formula X, an interpretation $\mathcal{I}$ is a *model* if the truth value of X w.r.t $\mathcal{I}$ is true. Formula X is a *logical consequence* or *logical entailment* of a set of formulas $\mathcal{H}$, denoted $\mathcal{H} \models X$, if, $\mathcal{I}$ is a model for $\mathcal{H}$ implies that $\mathcal{I}$ is a model for X for every interpretation $\mathcal{I}$ of $\mathcal{L}$.

G Training and Evaluation Details

G.1 Training Setup

For curriculum learning experiments on SLR-BENCH, we fine-tune the Llama-3.1-8B-Instruct model using supervised fine-tuning (SFT) with LoRA adapters using LLaMA-Factory (Zheng et al., 2024). Training is performed over two epochs on approximately 17k examples, which are presented sequentially, without shuffling, reflecting a curriculum of increasing logical complexity. Training is distributed across 8 GPUs using DeepSpeed with ZeRO Stage 3 optimization, taking 4 hours. Both optimizer states and model parameters are offloaded to CPU with pinned memory to maximize GPU memory efficiency. The AdamW optimizer is used in conjunction with a Warmup Cosine learning rate scheduler. Mixed-precision training is employed, with both bfloat16 and fp16 enabled in automatic mode. Communication overlap and contiguous gradients are activated to improve throughput, and model weights are saved in 16-bit precision at each checkpoint. Due to memory limitations, input sequences are truncated to a maximum length of 6k tokens using the Llama3 template, restricting training to `slr_basic_train`, `slr_easy_train`, and `slr_medium_train` splits. Optimization is performed using cross-entropy loss over the ground truth rule $R^\star$, with a per-device batch size of 5 and gradient accumulation over 2 steps, resulting in an effective batch size of 80 samples per step across 8 GPUs. The learning rate is set to 2×10^{-4}, scheduled with a cosine scheduler and a warmup ratio of 0.03. All relevant hyperparameters and training scripts are included in the codebase for full reproducibility.

G.2 Evaluation Setup

For downstream evaluation, we use the Language Model Evaluation Harness (Gao et al., 2024b) with default settings for each benchmark, enabling few-shot as multiturn prompting to support multi-turn contexts where applicable, including the official pass rate (typically pass@1) and whether evaluation is performed in zero-shot or few-shot mode. All evaluations are conducted on 8 GPUs using vLLM (Kwon et al., 2023) for efficient batch inference. Reported scores reflect accuracy for each model and benchmark, and all results are based on the official evaluation splits and standardized prompt formatting consistent with the SLR curriculum.

H Code and Licenses

This work introduces and publicly releases several scientific artifacts, including the SLR framework for scalable logical reasoning with large language models, the SLR-BENCH dataset comprising 19,000 tasks across 20 curriculum levels, and associated training, evaluation, and logic validation scripts. All code and data with the logic reward interface will be made publicly available after publication.

All original software developed as part of this re-

search is distributed under the MIT License, while the datasets are released under the Creative Commons Attribution 4.0 International License (CC BY 4.0), unless specified otherwise in the respective repositories. These licenses permit broad academic and research use, as well as modification and redistribution, provided appropriate credit is given to the original authors.

In addition to the artifacts created in this project, several external resources were utilized, including pretrained language models (e.g., Llama, OpenAI, DeepSeek, Gemini) and open-source Python libraries such as HuggingFace Transformers and PyTorch. All third-party resources were used strictly in accordance with their respective licenses and intended research purposes, and are appropriately cited in this paper and in the code repositories.

We further note that AI-based tools were used during the preparation of this work. Specifically, AI-guided writing assistants (such as ChatGPT) were employed to refine scientific text, and GitHub Copilot was used to support code development and debugging. The use of these tools was limited to improving clarity and efficiency; all research design, results interpretation, and final manuscript decisions were made by the authors.

The intended use of all released code and data is for research, academic, and educational purposes. Commercial use or deployment in production environments is not permitted without explicit permission or legal review. Any derivatives or extensions of the dataset must comply with the original license terms and the conditions of any incorporated sources. Users are encouraged to consult the individual license files provided in each repository for further details.

I Potential Risks

While this work is primarily intended to advance research in logical reasoning with language models, we recognize several potential risks associated with its development and open release. Enhanced reasoning capabilities in LLMs may be misused, for example, in generating persuasive but misleading arguments, automating manipulation, or circumventing safety mechanisms. The resources and benchmarks we provide, although synthetic and research-focused, could be repurposed for unintended or dual-use applications.

Additionally, while our work does not directly contribute to artificial general intelligence (AGI), we acknowledge broader discussions in the AI community regarding the long-term risks of increasingly capable AI systems. We believe the immediate risks of our work relate to dual-use and misuse as described above, and we encourage responsible use and ongoing monitoring of downstream applications as AI capabilities continue to evolve.

Synthesis Process and Outputs

Task Specification:

(i) **Language** $\mathcal{L} = (\mathcal{V}, \mathcal{G})$:

- Vocabulary $\mathcal{V}$: Predicates $\mathcal{P}$ = {`is_red_train/1, has_car/2, car_color/2, car_len/2`}; Constants $\mathcal{C}$ = {`t1, t2, c1, c2, red, blue, short, long`}
- Grammar $\mathcal{G}$: Restricts predicates to apply to compatible constant types.

(ii) **Configuration** Θ: Rule length $R_{\text{len}} = 2$; Problem size $\kappa = (\kappa_{\text{pos}} = 1,\ \kappa_{\text{neg}} = 1)$

Synthesis Steps:

1. Rule Synthesis: The RULEGENERATOR produces a latent ground-truth rule $R^\star$:

```
is_red_train(T) :- has_car(T, C), car_color(C, red).
```

2. Background Synthesis (Loop):

Iteration 1 (finds a positive example):

- Sample Background (b_1): 'has_car(t1, c1). car_color(c1, red).'
- Assign Label: Query q_1 = *is_red_train(t1)*. Entailment $b_1 \cup R^\star \models q_1$ holds. Result: $(1, q_1)$.
- Accept/Reject: $|E^+| < \kappa_{\text{pos}}$, sample is **accepted**. $B \leftarrow b_1$, $E^+ \leftarrow \{q_1\}$.

Iteration 2 (finds a negative example):

- Sample Background (b_2): 'has_car(t2, c2). car_color(c2, blue).'
- Assign Label: Query q_2 = *is_red_train(t2)*. Entailment $b_2 \cup R^\star \models q_2$ fails. Result: $(0, q_2)$.
- Accept/Reject: $|E^-| < \kappa_{\text{neg}}$, sample is **accepted**. $B \leftarrow B \cup b_2$, $E^- \leftarrow \{q_2\}$.

The loop terminates as both target sizes are met. The final task is $\mathcal{I} = (B, E^+, E^-)$.

Final Synthesizer Outputs:

1. Latent Ground-Truth Rule ($R^\star$):

```
is_red_train(T) :- has_car(T, C), car_color(C, red).
```

2. Validation Program (B, E^+, E^-):

```
has_car(t1, c1).
car_color(c1, red).
has_car(t2, c2).
car_color(c2, blue).
is_red_train(t1).
```

3. Instruction Prompt (example formats):

(a) Prolog-style Prompt:

```
% Given the following background knowledge:
has_car(t1, c1).
car_color(c1, red).
has_car(t2, c2).
car_color(c2, blue).
is_red_train(t1).
% Your task is to find a rule "is_red_train(T) :-" that solves the bk.
```

(b) Natural Language Prompt:

```
% Given the following background knowledge:
Train t1 has a car c1. The car c1 is red.
Train t2 has a car c2. The car c2 is blue.
% Your task is to find a rule "is_red_train(T) :-" that solves the bk.
```

Figure 6: Step-by-step example of the automatic ILP task synthesis process in SLR. Given a task specification, comprising a language and a task config, the synthesizer generates a ground-truth rule, samples background knowledge, assigns positive and negative example labels, and produces symbolic (Prolog-style) or natural-language prompts. The figure illustrates all intermediate steps and the final output of the synthesizer.