
Curriculum Abductive Learning

Wen-Chao Hu, Qi-Jie Li, Lin-Han Jia, Cunjing Ge,
Yu-Feng Li, Yuan Jiang, Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artificial Intelligence, Nanjing University, China

Abstract

Abductive Learning (ABL) integrates machine learning with logical reasoning in a loop: a learning model predicts symbolic concept labels from raw inputs, which are revised through abduction using domain knowledge and then fed back for retraining. Due to the nondeterminism of abduction, the training process often suffers from instability, especially when the knowledge base is large and complex, resulting in a prohibitively large abduction space. While prior works focus on improving candidate selection within this space, they typically treat the knowledge base as a static black box. In this work, we propose Curriculum Abductive Learning (C-ABL), a method that explicitly leverages the internal structure of the knowledge base to address the ABL training challenges. C-ABL partitions the knowledge base into a sequence of sub-bases, progressively introduced during training. This reduces the abduction space throughout training and enables the model to incorporate logic in a stepwise, smooth way. Experiments across multiple tasks show that C-ABL outperforms previous ABL implementations, significantly improves training stability, convergence speed, and final accuracy, especially under complex knowledge setting.

1 Introduction

Recent advances in data-driven machine learning methods have achieved remarkable success across a wide range of tasks. However, when it comes to leverage structured, formal knowledge—such as logic rules or domain-specific constraints—these methods often fall short, therefore mitigating their interpretability and reliability. This has motivated the emergence of data- and knowledge-driven artificial intelligence, which aims to tightly integrate learning from data with logical reasoning [Hitzler, 2022, Yang et al., 2025].

Abductive Learning (ABL) [Zhou, 2019, Zhou and Huang, 2022] is a representative framework that integrates machine learning with logical reasoning in a balanced loop. In ABL, a machine learning model first predicts symbolic concept labels from input data, which may initially violate domain knowledge when the model is under-trained; abduction then generates a set of concept label candidates compatible with domain knowledge, and from which the best candidate is selected via consistency optimization. The selected candidate is subsequently used to supervise the model’s next update. This loop iterates, enabling the system to gradually align predictions with logical constraints.

While elegant in principle, this loop setup often introduces training inefficiency and instability [Yang et al., 2024, He and Li, 2025]. Due to the nondeterminism of abduction [Magnani et al., 2009], often multiple candidates compatible with the domain knowledge exist. When the knowledge base is complex, the resulting abduction space can become prohibitively large, causing the model to oscillate among many plausible yet incorrect concept labels. This undermines the reliability of the supervision signal and hinders the training process.

Prior work [Huang et al., 2021, Tao et al., 2024, He et al., 2024] has attempted to improve ABL stability by refining consistency optimization, i.e., the selection process within the abduction space. While these methods offer incremental gains, they remain fundamentally limited when the space itself is too large. A key limitation is that they treat the knowledge base as a static black box, ignoring its internal structure [Hu et al., 2025]. We argue that a more principled solution requires **leveraging the structure of knowledge base and actively manage its complexity throughout training**.

Many knowledge bases exhibit inherent staged or hierarchical structural properties [Glanois et al., 2022]. Take the legal domain as an example [Huang et al., 2020], some laws and regulations are relatively simple and foundational, while others involve more complex conditions or exceptions. Such rules can be organized into phases of increasing complexity. Motivated by this commonly observed structure, we propose a logic-level **Curriculum Abductive Learning (C-ABL)** method. Instead of exposing the model to the full knowledge base from the beginning, C-ABL incrementally introduces logic rules over several phases during training. In this way, abduction space is substantially reduced throughout training, enabling faster and more stable optimization. As training progresses, increasingly complex logic is introduced, allowing for a gradual alignment with full-domain constraints.

To realize this idea, we first propose a principled algorithm that leverages the structural properties of knowledge base and partitions it into a sequence of sub-bases with strong local dependencies, increasing complexity, and self-contained reasoning. These are then integrated in ABL via a curriculum training process. Our theoretical analysis shows that this curriculum design achieves efficient and stable training within phases by leveraging prior knowledge learned in earlier phases, and ensures smooth transitions across phases without catastrophic forgetting. Experiments on digit addition, chess attack, and real-world legal judgment tasks demonstrate that C-ABL significantly improves training stability, convergence speed, and final accuracy over strong ABL and neuro-symbolic baselines.

In summary, our contributions are as follows:

- We identify the uncontrolled abduction space as the core bottleneck in ABL training and highlight the limitations of prior implementations.
- We propose the C-ABL method that explicitly structures and stages the logical knowledge base, transforming abduction from a static process into a dynamic and adaptive one.
- We theoretically and empirically demonstrate that our curriculum design leads to a more efficient and stable training across various domains.

To our knowledge, this is the first work to explicitly leverage the internal structure of the knowledge base in ABL—an essential yet previously overlooked component—shifting from black-box invocation to structure-aware, logic-enhanced training. This perspective opens new directions for improving ABL and extends its potential for broader and more reliable applications.

2 Problem Setting and Preliminary

2.1 Knowledge Base Concepts

We begin with a brief introduction to the basic concept of a **knowledge base**. This paper focuses on domain knowledge expressed in **logic programming** [Huth and Ryan, 2004, Ben-Ari, 2012], a formalism rooted in first-order logic that offers expressiveness and human interpretability, while remaining computationally tractable for reasoning. Specifically, the knowledge base \mathcal{KB} is represented as a set of logic programming rules. Each **rule** is written in the Horn clause form [Horn, 1951]:

$$A \leftarrow B_1 \wedge \cdots \wedge B_n,$$

where “ \leftarrow ” denotes logical implication, meaning that A (left-hand side, the **head** of the rule) holds when the conjunction of B_i (right-hand side, the **body** of the rule) holds. Each A and B_i is a logical statement expressed with logical **predicates**.

Given such a knowledge base, both **deduction** and **abduction** [Josephson and Josephson, 1996, Magnani et al., 2009] can be performed. Deduction allows deriving A (head) from the premises $\bigwedge_i^n B_i$ (bodies), while abduction seeks possible explanations $\bigwedge_i^n B_i$ (bodies) that would account for the observation of A (head).

2.2 Problem Setting

The task of this paper is as follows: Given an input data sequence $\mathbf{x} = (x^{(1)}, \dots, x^{(m)})$ of length m , where each element $x^{(i)}$ is sampled from an input space \mathcal{X} and corresponds to a **concept label** $z^{(i)}$ drawn from a symbolic concept label set $\mathcal{Z} = \{z_1, \dots, z_N\}$, the entire sequence \mathbf{x} is associated with a **target label** y that reflects a target reasoning outcome. In addition, we are provided with a logical knowledge base \mathcal{KB} , expressed in first-order logic, that defines how concept labels can entail specific target labels. Specifically, each concept label $z \in \mathcal{Z}$ appears as a predicate in the bodies of rules in \mathcal{KB} , and target labels y appear in the heads.

The goal is to learn a model f that maps each $x^{(i)}$ to a concept label $z^{(i)}$, such that the sequence $\mathbf{z} = (z^{(1)}, \dots, z^{(m)})$, together with \mathcal{KB} , can be used to deduce the target label y , in other words, \mathbf{z} satisfies logical condition $\mathbf{z} \wedge \mathcal{KB} \models y$.

Example 2.1 (Decimal d -digit Addition). *Consider a task of predicting the sum of two decimal d -digit numbers, represented by a sequence of images [Manhaeve et al., 2018]. The input $\mathbf{x} = (x^{(1)}, \dots, x^{(2d)})$ consists of $2d$ images from a visual input space \mathcal{X} , such as MNIST [LeCun et al., 1998] or CIFAR10 [Krizhevsky et al., 2009], (e.g., $\mathbf{x} = (\mathbf{7}, \mathbf{3}, \mathbf{7}, \mathbf{3})$). The concept labels are drawn from the symbol set $\mathcal{Z} = \{\text{zero}, \dots, \text{nine}\}$, and the target label y is their sum. The knowledge base \mathcal{KB} encodes the rules of multi-digit addition. The goal is to map each $x^{(i)}$ to its corresponding concept label $z^{(i)}$, such that the resulted sequence \mathbf{z} (e.g., $[\text{one}, \text{three}, \text{seven}, \text{three}]$) and \mathcal{KB} allows deduction of the correct sum y (e.g., 86).*

2.3 Brief Introduction of Abductive Learning

Abductive Learning (ABL) [Zhou, 2019, Zhou and Huang, 2022] consists of two components: a machine learning model $f : \mathcal{X} \rightarrow \mathcal{Z}$ for concept perception, and a logical reasoning module that utilizes the first-order rules in \mathcal{KB} . During the training process, when the input \mathbf{x} is received, ABL uses f to map it to perceptive concept labels $\hat{\mathbf{z}}$, then the logical reasoning module checks whether $\hat{\mathbf{z}} \wedge \mathcal{KB} \models y$ holds, i.e., $\hat{\mathbf{z}} \wedge \mathcal{KB}$ can correctly deduce y . If the condition is not satisfied, ABL utilizes the reasoning module to get revised concept labels $\bar{\mathbf{z}}$, and then, $\bar{\mathbf{z}}$ is used as the ground-truth concept label for \mathbf{x} to train f . This process is repeated iteratively until the training of f converge.

Specifically, the process by which the reasoning module obtains the revised concept labels $\bar{\mathbf{z}}$ involves the following two sequential steps [Dai et al., 2019, Huang et al., 2024]: (1) **Abduction**: Return the **abduction space**, i.e., the set of all concept label candidates that are compatible with \mathcal{KB} , denoted as $\mathbb{S} = \{\mathbf{z} \mid \mathbf{z} \wedge \mathcal{KB} \models y\}$; and (2) **Selection**: Performing consistency optimization to select the candidate $\bar{\mathbf{z}} \in \mathbb{S}$ that is most consistent with the model’s current prediction $p = f(\mathbf{x})$:

$$\bar{\mathbf{z}} = \arg \min_{\mathbf{z} \in \mathbb{S}} 1 - \prod_{i=1}^m p^{(i)}(z^{(i)}), \quad (1)$$

where $p^{(i)}(z^{(i)})$ denotes the probability assigned by the model to label $z^{(i)}$.

Challenges. Due to the nondeterminism of abduction, for a given y , the size of abduction space $|\mathbb{S}|$ is often large, i.e., there may be multiple \mathbf{z} satisfying $\mathbf{z} \wedge \mathcal{KB} \models y$. However, for each input \mathbf{x} , the correct concept labels should be unique [He et al., 2024]. Therefore, selecting an incorrect \mathbf{z} from \mathbb{S} introduces erroneous supervision to the perception model f , which could then be reinforced through subsequent iterations in the learning loop. This creates a vicious cycle that slows convergence, destabilizes training, and ultimately degrades final model accuracy.

Example 2.2. *In Example 2.1, given $y = 86$ and \mathcal{KB} , the resulting set \mathbb{S} would be of size $L = 87$ ($\mathbb{S} = \{[\text{zero}, \text{zero}, \text{eight}, \text{six}], [\text{zero}, \text{one}, \text{eight}, \text{five}], \dots\}$). However, the concept labels that are consistent with $\mathbf{x} = (\mathbf{7}, \mathbf{3}, \mathbf{7}, \mathbf{3})$ should be unique: $\bar{\mathbf{z}} = [\text{one}, \text{three}, \text{seven}, \text{three}]$. In this case, all other elements in \mathbb{S} may introduce false labels that mislead the training of f .*

Several solutions have been proposed to address this training challenge, including methods by Huang et al. [2021], Tao et al. [2024], He et al. [2024], which primarily focus on consistency optimization, i.e., how to **select** the best candidate \mathbf{z} within \mathbb{S} . However, such methods remain limited when \mathbb{S} is large—no selection strategy can fully compensate for the challenges brought by an overwhelming hypothesis space [Yang et al., 2024, Jia et al., 2025]. Addressing the root of training instability requires going beyond improving candidate selection and actively managing the size and structure of the abduction space itself.

3 Our Method

This section proposes **Curriculum Abductive Learning (C-ABL)**, a method that addresses the limitations in previous ABL methods. C-ABL actively managing the complexity of the knowledge base during training: Instead of exposing the model to the full knowledge base all at once, it divides ABL training into multiple curriculum **phases**, with each phase p guided by a sub-base \mathcal{KB}_p , progressively expanding towards \mathcal{KB} .

3.1 Curriculum Design over Knowledge Base

To enable curriculum-style progression, we first introduce a principled algorithm that partitions the full knowledge base \mathcal{KB} into a sequence of structured sub-bases $\mathcal{KB}_1, \dots, \mathcal{KB}_P$ used in each curriculum phase. Our partitioning strategy follows three intuitive principles that together ensure each sub-base builds upon the last in a stable and constructive manner:

- P1 Dependency Cohesion** (*What should be grouped together?*): Rules depending on each other, such as those frequently co-occur or form tight reasoning chains, appear in the same phase.
- P2 Stepwise Complexity** (*What should be introduced first?*): Simpler rules are placed in earlier phases, while more complex rules built on prior predicates are introduced later.
- P3 Self-contained Reasoning** (*Can it reason independently?*): Each phase includes all rules necessary for reasoning over the involved concepts, without relying on unseen rules.

To implement these principles, we introduce the following structure:

Definition 3.1 (Dependency Graph). Given a knowledge base \mathcal{KB} consisting of first-order logic rules, its dependency graph $G = (V, E)$ is a directed graph where each node $r \in V$ corresponds to a rule, and there is an edge $(r_i, r_j) \in E$ if the head predicate of r_i appears in the body of r_j .

This graph captures the structural reasoning flow: if r_j relies on conclusions from r_i , there is an edge from r_i to r_j . Based on this graph, we design a partitioning algorithm shown in [Algorithm 1](#).

Partitioning Algorithm. The algorithm begins by scanning each concept label $z \in \mathcal{Z}$ (as introduced in [Section 2.2](#), each z is a predicate that appears in the body of some rule) and forms an initial cluster C_z containing all rules that directly reference z ([Line 3](#)). This cluster is then recursively expanded by traversing the dependency graph, incorporating additional rules that are required to derive the final target label y , as well as rules that define intermediate predicates used in this process ([Line 4](#)). In cases where the dependency graph contains cycles (e.g., formed by recursive rules), the above process will cluster all rules involved in the cycle together.

Algorithm 1: \mathcal{KB} partitioning

Input: Knowledge base \mathcal{KB} (a set of first-order logic rules), minimum sub-base size τ (optional)

Output: A sequence of sub-bases $\mathcal{KB}_1, \dots, \mathcal{KB}_P$

- 1 Construct the dependency graph $G = (V, E)$ of \mathcal{KB} \triangleright from [Definition 3.1](#);
 - 2 **for each concept label** $z \in \mathcal{Z}$ **do**
 - 3 Initialize cluster C_z with all rules referencing z ;
 - 4 Recursively expand C_z via edges in G ;
 - 5 **for each pair** (C_a, C_b) **do**
 - 6 **if** $C_a = C_b$ **then**
 - 7 Merge clusters;
 - 8 **else if** $\exists (r_i, r_j) \in E$ with $r_i \in C_a, r_j \in C_b$ **then**
 - 9 Set $C_a \prec C_b$;
 - 10 Topologically sort clusters under \prec ;
 - 11 Set phase counter $p \leftarrow 1$;
 - 12 **for each sorted cluster** C_i **do**
 - 13 If $|\mathcal{Z}_i| < \tau$, merge C_i with the next cluster $\triangleright \mathcal{Z}_i$: the set of concept labels C_i involves;
 - 14 Assign sub-base: $\mathcal{KB}_p \leftarrow \bigcup_{j=1}^i C_j$;
 - 15 $p \leftarrow p + 1$;
-

Once the initial clusters are formed, we refine them in two ways: (1) Merge any duplicate clusters with identical rule sets (Line 7); (2) Define a precedence ordering \prec , and perform topological sort (Line 9-10). Specifically, this ordering is based on the interdependencies of clusters: If there is an edge from a rule in C_a to a rule in C_b , i.e., reasoning in C_b relies on conclusions from C_a , we assign $C_a \prec C_b$ (when bidirectional dependencies exist, we let the cluster with fewer rules appear earlier).

Sub-bases are then formed incrementally by aggregating clusters up to a given index (Line 14). This process satisfies **P1** by grouping together rules in the same reasoning chain, keeping each phase logically localized around certain concept predicates. It also satisfies **P2**, as the precedence ordering introduces foundational rules and simpler concepts earlier, and allows subsequent phases to build upon prior ones, forming a progression from easy to complex reasoning throughout training.

To avoid overly fine-grained partitions, small clusters can be optionally merged with their immediate successors (Line 13). This step is governed by an optional threshold τ , which specifies the minimum number of concept label predicates per phase; if not set, no merging is performed.

Illustrative examples of the above process are shown in Appendix B.1. We also provide the computational cost analysis in Appendix B.2, showing that compared to the ABL training pipeline, the partitioning is a one-time, offline preprocessing step with negligible computational overhead.

Guarantee of Self-Contained Reasoning. To ensure **P3**, we define the concept label domain of each phase p , denoted \mathcal{Z}_p , as the set of concept labels \mathcal{KB}_p involves. We require that \mathcal{KB}_p is logically sound and complete for reasoning over \mathcal{Z}_p , without relying on future rules. This is formalized below.

Theorem 3.2. *For any phase p , let φ be any logical statement over concepts in \mathcal{Z}_p . Then:*

1. **Soundness:** $\mathcal{KB}_p \models \varphi \implies \mathcal{KB} \models \varphi$; that is, \mathcal{KB}_p does not derive any conclusions that are invalid under the full knowledge base.
2. **Completeness:** $\mathcal{KB}_p \models \varphi \iff \mathcal{KB} \models \varphi$; that is, \mathcal{KB}_p is sufficient to derive all valid conclusions over its concept label domain.

We then have the following corollary, indicating that the final sub-base \mathcal{KB}_P is logically equivalent to the full knowledge base \mathcal{KB} in terms of reasoning over all concept labels.

Corollary 3.3. *For the final phase P , for any formula φ over \mathcal{Z} , we have: $\mathcal{KB}_P \models \varphi \iff \mathcal{KB} \models \varphi$.*

All proofs are provided in Appendix C.

3.2 Curriculum-Guided Training

Building on the sub-bases from Algorithm 1, we now present C-ABL, which conducts ABL training across P curriculum phases. The pseudocode is provided in Algorithm 2 in Appendix D.

In phase p , training is guided by sub-base \mathcal{KB}_p . To align the model’s prediction space with the reasoning scope of \mathcal{KB}_p , we dynamically schedule training data whose concept labels fall within the domain \mathcal{Z}_p [Bengio et al., 2009, Marconato et al., 2023a]. Then, each phase follows the standard ABL procedure, as stated in Section 2.3.

Training in phase p continues until the prediction accuracy for every label $z \in \mathcal{Z}_p$ exceeds the uniform guessing baseline $1/|\mathcal{Z}|$, and then C-ABL proceeds to the next phase $p + 1$. This is a mild criterion requiring only a weak signal for each concept to facilitate progression. Nonetheless, as shown in Section 4, reaching this level is sufficient for one phase to benefit learning in the next, enabling the model to build upon previously acquired knowledge in a curriculum-style progression. Note that concept labels are evaluated on a held-out validation set and are not used for training or backpropagation, and the only supervision signal during training comes from the target label y .

This process repeats until reaching the final phase P , at which point the full knowledge base \mathcal{KB} is employed and training continues until a termination condition is met, such as reaching a maximum number of training iterations.

4 Theoretical Analysis: Efficient and Smooth Optimization

In this section, we provide a formal analysis of the Curriculum Abductive Learning (C-ABL) method. We demonstrate how C-ABL improves the training process of ABL by addressing two core aspects: (1) Improving the efficiency and stability of training within each phase, and (2) Ensuring smooth transitions across phases. All proofs are provided in [Appendix C](#).

4.1 Efficient Training within Phase

We begin by analyzing each phase individually. In particular, we analyze how our method reduce the size of the abduction space by benefiting from curriculum training paradigm, and therefore contributes to more efficient reasoning and stable optimization.

The key bottleneck in ABL training arises from the size of the abduction space $\mathbb{S} = \{z \mid z \wedge \mathcal{KB} \models y\}$. This space \mathbb{S} includes all possible concept labels compatible with the knowledge base, constituting the set of candidates ABL search and select from.

When using the full knowledge base, the size of \mathbb{S} can grow exponentially:

Lemma 4.1. *If the knowledge base \mathcal{KB} involves N concept labels and the input has m positions, then the size of the abduction space with full knowledge base is bounded by $|\mathbb{S}| \leq N^m$.*

To mitigate this, C-ABL partitions the knowledge base to reduce complexity. At phase p , the subset $\mathcal{Z}_p \subseteq \mathcal{Z}$ is active, and the abduction space is: $\mathbb{S}_p = \{z \in \mathcal{Z}_p^m \mid z \wedge \mathcal{KB}_p \models y\}$. The theorem below shows that, under the curriculum strategy proposed in [Section 3.2](#), the size of \mathbb{S}_p is largely reduced.

Theorem 4.2. *Assuming that all previously introduced concepts $z \in \mathcal{Z}_{p-1}$ are predicted with accuracy exceeding random chance (i.e., $\text{Acc}(z) > \frac{1}{|\mathcal{Z}|}$), and pseudo-labels are selected via [Eq. \(1\)](#). Then the size of the abduction space in phase p is bounded by $|\mathbb{S}_p| \leq |\mathcal{Z}_p \setminus \mathcal{Z}_{p-1}|^m$.*

Remark 4.3. The above theorem relies on a natural assumption that the semantics of concepts remain constant when the knowledge base evolves (e.g., the image [3](#) consistently corresponds to the concept three regardless of training phase) [[Marconato et al., 2023b, 2024](#)]. This theorem suggests that the abduction space in each phase is effectively confined to newly introduced concepts $\mathcal{Z}_p \setminus \mathcal{Z}_{p-1}$, benefiting from the accumulated knowledge of earlier phases: The previously learned concepts provide a foundation with lower risk of noisy supervision [[Van Krieken et al., 2024, Maene et al., 2024](#)], enabling the model to focus on resolving the new concepts introduced in the current phase.

Combining [Lemma 4.1](#) and [Theorem 4.2](#), the relative upper bound of the phase-level abduction space satisfies

$$\frac{|\mathbb{S}_p|}{|\mathbb{S}|} \leq \left(\frac{|\mathcal{Z}_p \setminus \mathcal{Z}_{p-1}|}{|\mathcal{Z}|} \right)^m,$$

which shrinks exponentially in m . The following example illustrates this reduction:

Example 4.4. *In the d -digit addition task, as shown in [Figure 1](#), the size of \mathbb{S} increases rapidly with d when applying the full knowledge base. In contrast, with phase partitioning (using a phase threshold $\tau = 2$ in [Algorithm 1](#)), the abduction space remains consistently small across all values of d .*

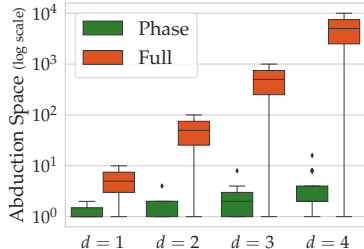


Figure 1: Abduction space size.

This reduction yields two advantages:

- Improved computational efficiency:** A smaller abduction space significantly lowers the reasoning cost per training iteration, especially when the knowledge base is large, where the number of logically compatible candidates explodes exponentially, making abduction prohibitively expensive [[Yu et al., 2016, Hu et al., 2025](#)];
- Faster and stabilized convergence:** A large abduction space introduces many plausible yet incorrect candidates, especially in early training when the model’s predictions are still noisy. These incorrect candidates can mislead training and cause the model to oscillate or reinforce suboptimal hypotheses. Limiting the abduction space will promote faster, more stable convergence, which is captured in the following theorem:

Theorem 4.5 (Iteration Complexity of ABL). *Let $f^{(t)}$ be the model at iteration t in ABL. Assuming that: (1) At each iteration, the model receives a pseudo-label $\bar{z}^{(t)} \in \mathbb{S}$ that satisfies $\bar{z}^{(t)} \wedge \mathcal{KB} \models y$; (2) The model updates are driven by minimizing a loss function $\ell(f(x), \bar{z}^{(t)})$ that is convex and ρ -Lipschitz; (3) The learning rate $\eta > 0$ is fixed. Then, the number of iterations T required to reach an expected consistency error less than ε satisfies: $T = \mathcal{O}\left(\frac{|\mathbb{S}|^2 \cdot \rho^2}{\varepsilon^2}\right)$.*

This theorem formally shows that the number of required iterations grows quadratically with the size of the abduction space. Therefore, reducing \mathbb{S} in each phase not only improves the speed of logical reasoning but also accelerates overall convergence, particularly in large knowledge base scenarios.

4.2 Smooth Phase Transition

Besides improvements within each phase, we now examine transitions across phases. Two properties are key to maintaining stability: **logical consistency**, ensuring conclusions made in earlier phases remain valid, and **topological continuity**, ensuring the optimization landscape evolves with no abrupt changes. Together, these properties help prevent catastrophic forgetting [Bengio et al., 2009], a common challenge in curriculum learning where new updates may destabilize previously acquired knowledge, by preserving previously acquired knowledge both semantically and parametrically as the curriculum progresses.

To ensure logical consistency, we require that conclusions derived in one phase remain valid in all subsequent phases. The following theorem shows sub-bases from Algorithm 1 satisfy this property:

Theorem 4.6. *For any formula φ over \mathcal{Z}_p , we have $\mathcal{KB}_p \models \varphi$ if and only if $\mathcal{KB}_{p+1} \models \varphi$.*

To analyze topological continuity, we adopt the notion of Stone spaces [Johnstone, 1982], which offer a topological view of logical model sets (a brief introduction is provided in Appendix C.7). Assuming each concept label z_i is a unary predicate with a Boolean assignment, the set of formulas over \mathcal{Z}_p then forms a Boolean algebra \mathcal{B}_p . Its corresponding Stone space $S(\mathcal{B}_p)$ —the set of ultrafilters over \mathcal{B}_p —captures the space of logically consistent models. For these model spaces, we have:

Theorem 4.7. *For each p , $S(\mathcal{B}_p) \subseteq S(\mathcal{B}_{p+1})$.*

As a result, the model spaces form a nested sequence of compact topological spaces, ensuring that admissible models evolve smoothly, without causing abrupt shifts in the optimization landscape.

5 Experiments

In this section, we evaluate our method across three tasks. On digit addition and its variants, the most widely used benchmarks in the neuro-symbolic field, we thoroughly validate the effectiveness of our method. On a chess attack task, we test robustness when only Boolean target label is provided, leading to a significantly larger abduction space. Finally, we apply our method to a real-world legal judgment task to demonstrate its practical applicability in complex domains.

5.1 d -digit Addition

This task, as outlined in Example 2.1, the input consists of image sequences representing two d -digit numbers, and the goal is to predict their sum. We aim to assign a concept label to each image, so that the final result is obtained through reasoning. The standard decimal setting includes 10 concept labels, and we use images from CIFAR [Krizhevsky et al., 2009] to represent digits. To extend the challenge, we introduce a hexadecimal variant with 16 concept labels, constructed using the first 16 classes of CIFAR100 [Krizhevsky et al., 2009]. More details and additional experimental results are provided in Appendix F.1.

We compare our method against two ABL baselines: (1) the original **ABL** [Huang et al., 2024], and (2) the improved variant **A³BL** [He et al., 2024]. We also include several representative neuro-symbolic methods: (1) **NeurASP** [Yang et al., 2020], (2) **LTN** [Serafini and Garcez, 2016], (3) **DeepProbLog** [Manhaeve et al., 2018, 2021a], and (4) **DeepStochLog** [Winters et al., 2022]. Introduction of compared methods are provided in Appendix E. All methods use ResNet18 [He et al., 2016] as the perception module, and are trained for a total of 5,000 iterations.

Table 1: Comparison of prediction accuracy (top) and total training time in minutes (bottom) on digit addition tasks. “N/A” indicates runtime exceeding 12 hours. C-ABL consistently achieves higher accuracy and faster training, with its benefits more pronounced as reasoning complexity increases.

Method	Decimal				Hexadecimal		
	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 1$	$d = 2$	$d = 3$
NeurASP	10.00±0.00	9.73±0.46	9.73±0.46	N/A	6.27±0.85	N/A	N/A
DeepProbLog	7.55±0.58	N/A	N/A	N/A	6.56±0.73	N/A	N/A
DeepStochLog	70.56±0.89	73.25±1.40	72.88±1.30	71.80±1.50	49.25±2.34	42.42±1.72	35.10±2.03
LTN	68.25±1.08	71.10±1.52	71.77±1.18	70.95±1.36	52.85±1.88	53.59±2.01	51.14±2.22
ABL	70.05±1.10	74.49±1.82	74.50±1.06	73.05±1.29	60.87±1.22	61.75±1.16	64.14±1.32
A ³ BL	72.06±1.12	75.62±1.59	74.65±1.72	73.28±1.52	22.45±5.25	60.75±1.58	65.81±1.12
C-ABL	71.77±1.03	76.74±1.34	77.65±1.32	76.30±1.22	63.02±1.03	64.25±1.12	66.67±1.11
NeurASP	113.2±7.8	214.7±13.6	376.5±22.9	N/A	108.5±10.3	N/A	N/A
DeepProbLog	269.8±7.7	N/A	N/A	N/A	214.3±14.2	N/A	N/A
DeepStochLog	32.8±2.1	59.2±3.8	101.4±7.9	492.5±22.4	34.5±2.3	97.3±5.9	182.4±9.1
LTN	30.2±2.0	55.6±3.5	96.7±6.8	473.2±25.1	33.1±2.0	93.5±5.7	176.0±8.8
ABL	12.6±1.1	27.4±2.1	47.1±3.2	253.6±14.8	20.8±1.2	49.1±3.1	99.7±5.2
A ³ BL	20.6±1.9	29.1±2.4	60.1±4.0	291.2±18.2	31.2±2.6	68.2±4.3	109.8±6.8
C-ABL	16.2±1.4	21.5±1.9	39.8±2.8	132.6±7.7	16.4±1.9	32.0±2.0	55.5±4.2

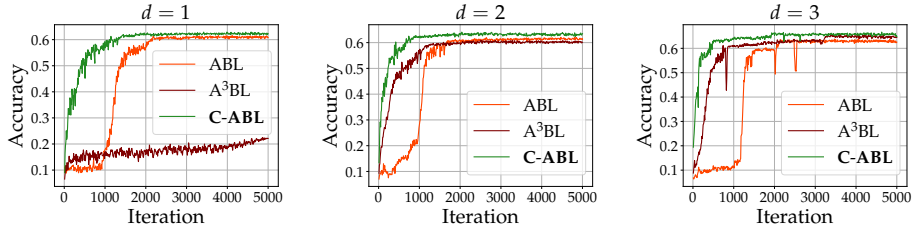


Figure 2: Comparison of training curves on the hexadecimal addition task with varying digit length d .

Experiments were conducted on decimal addition ($d = 1$ to 4) and hexadecimal addition ($d = 1$ to 3). As shown in Table 1, compared to previous ABL and other neuro-symbolic methods, C-ABL consistently achieves higher accuracy and requires significantly less total training time, indicating a more efficient reasoning process within each iteration. This improvement becomes especially evident in more complex settings (e.g., higher-digit or hexadecimal tasks), where the knowledge base is larger and reasoning becomes more expensive, highlighting the benefits of curriculum-style learning.

Analysis of the Training Process. We examine how curriculum learning affects ABL training process. Figure 2 shows the training curves on the hexadecimal setting with varied d . We may see that C-ABL achieves faster and more stable convergence than ABL and A³BL across all settings: While the baselines often exhibit noisy or stagnant training curves, C-ABL rapidly improves within the first few hundred iterations and reaches near-maximum performance by iteration 1,000. This aligns with our theoretical findings: C-ABL mitigates training instability and achieves faster convergence.

Figure 3 further shows the training curves of three randomly selected concept labels in C-ABL (shown in green curves), each introduced in a different phase, with three vertical dashed lines indicating the start of each phase. (Here we present the first 2,000 iterations; the full training curves are provided in Figure 6 in Appendix F.1.) As shown, each concept improves immediately and stably once its corresponding phase begins, while learned concepts from earlier phases remain unaffected. In contrast, ABL introduces all labels at once (shown in red curves), leading to early-stage oscillations and slower convergence for all labels. This supports our claim: C-ABL enables stable, stepwise learning with smooth transitions across curriculum phases.

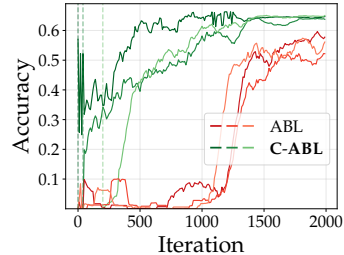


Figure 3: Training curve of three concept labels (hex., $d = 1$), each introduced in a different phase.

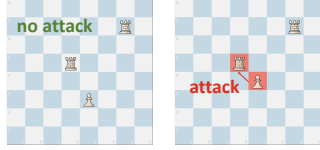


Figure 4: Examples of chess attack task.

Table 2: Results on chess attack task.

Method	Accuracy \uparrow	Iterations to Converge \downarrow
ABL	73.75 \pm 0.84	4560 \pm 96
A ³ BL	74.96 \pm 0.78	3389 \pm 105
C-ABL	86.79\pm0.66	2553\pm88

Table 3: Results on judicial sentencing task. C-ABL achieves improved F1, MAE and MSE scores, and takes fewer tokens to converge (defined as the point where F1 reaches 99% of its final value).

Method	Labeled Data %	F1 \uparrow	MAE \downarrow	MSE \downarrow	Tokens to Converge \downarrow
ABL	10%	0.895 \pm 0.024	0.762\pm0.018	0.888 \pm 0.069	4.100M
C-ABL		0.904\pm0.011	0.762\pm0.036	0.865\pm0.094	3.199M
ABL	50%	0.910 \pm 0.016	0.773 \pm 0.010	0.874 \pm 0.022	3.101M
C-ABL		0.920\pm0.006	0.722\pm0.014	0.773\pm0.048	2.106M

5.2 Chess Attack

We consider a chess attack task [Dai et al., 2019, Huang et al., 2023]. The input is a randomly generated chessboard populated with chess pieces, each represented by an image, and the goal is to determine whether any pair of pieces are in an attacking relationship, see examples in Figure 4. We aim to assign a concept label to each piece from $\mathcal{Z} = \{\text{rook, pawn, bishop, king, knight, queen}\}$, such that the assignments can logically determine the boolean target label (attack). The knowledge base defines the attack behavior of each piece. In our setup, chess pieces are represented using randomly sampled MNIST digits, and we use LeNet-5 [LeCun et al., 1998] as the perception model. More details are provided in Appendix F.2.

We compare C-ABL with ABL and A³BL, with results shown in Table 2. C-ABL achieves higher accuracy with faster convergence. Notably, since the target label in this task is binary, the abduction space is particularly large, highlighting the advantage of C-ABL in complex reasoning scenarios.

5.3 Judicial Sentencing

In this section, we apply C-ABL to a real-world task: judicial sentencing. This task aims to predict the final sentence length $y \in \mathbb{R}^+$ based on the input criminal judgment records. The model first predicts concept labels z representing sentencing factors (e.g., “voluntary surrender”, “repeat offense”), and then reasons with a legal knowledge base \mathcal{KB} to deduce y . We use the dataset from Huang et al. [2020], which includes 687 records. We use the pretrained google-bert/bert-base-chinese [Devlin et al., 2019] as the learning model. More details are provided in Appendix F.3.

We compare C-ABL with prior ABL implementations under two semi-supervised training settings, where 10% and 50% of the data include ground-truth concept labels available for training supervision. As shown in Table 3, C-ABL achieves higher F1 scores, lower MAE and MSE, and requires fewer tokens to converge, showing superior accuracy and training efficiency in real-world setting.

6 Related Work

Recently, there has been notable progress in combining data-driven machine learning with knowledge-driven logical reasoning. Some methods embed logical structure into neural networks [Serafini and Garcez, 2016, Xu et al., 2018, Badreddine et al., 2022, Yang et al., 2022, Ahmed et al., 2022], often by relaxing logical constraints, which can undermine reliability. Probabilistic neuro-symbolic systems preserve classical reasoning by treating neural outputs as distributions over symbols [Manhaeve et al., 2018, Yang et al., 2020, Kikaj et al., 2025]. Their inference typically involves weighted model counting [Chavira and Darwiche, 2008], whose exact computation can be intractable for large or complex knowledge bases. Recent efforts have introduced approximate inference methods to alleviate this issue [Manhaeve et al., 2021b, Winters et al., 2022, van Krieken et al., 2023, Li et al., 2023]. Abductive Learning (ABL) [Zhou, 2019, Zhou and Huang, 2022] integrates learning and reasoning in a balanced loop, where abduction and consistency optimization provide a bridge between the two

components. Supported by an open-source toolkit [Huang et al., 2024], ABL has shown success in diverse applications [Huang et al., 2020, Cai et al., 2021, Wang et al., 2021a, Gao et al., 2024, Hu et al., 2025, Wang et al., 2025]. However, current implementations treat the knowledge base as a black box and perform reasoning over the full logical space, often resulting in unstable supervision and inefficient convergence when the logical knowledge base is large or complex.

Curriculum learning improves generalization by organizing the learning process from easier to harder [Bengio et al., 2009, Wang et al., 2021b]. In the neuro-symbolic field, prior work has incorporated curriculum strategies by gradually increasing the complexity of input data or task structure [Mao et al., 2019, Chen et al., 2020, Le-Phuoc et al., 2021, Marconato et al., 2023a, Abbe et al., 2023, Lorello et al., 2024, Wei et al., 2025]. While they mainly focus on the data side, C-ABL introduces a curriculum over the knowledge base itself—leveraging its symbolic structure to guide reasoning progression. This allows the model to engage with increasingly complex logic in a controlled, interpretable way, and better align with the goal of exploiting structured knowledge.

Another line of research investigates ambiguity and shortcut behavior in neuro-symbolic systems and ABL framework [Marconato et al., 2023a, 2024, Li et al., 2024, Yang et al., 2024, He et al., 2024], which shares an underlying connection with our analysis: Large abduction spaces in ABL introduce ambiguity in concept label selection, which can in turn result in unstable or shortcut-driven supervision [He et al., 2024]. C-ABL tackles this structurally by actively managing the reasoning space of knowledge base, thereby enhancing the reliability of reasoning supervision.

Partitioning knowledge bases into structured components has long been studied in the logic and knowledge representation communities [Amir and McIlraith, 2000, 2005, Gocht and Balyo, 2017, Simiński, 2017]. While originally developed for formal reasoning, these ideas motivate our logic-guided curriculum for improving learning dynamics in ABL.

7 Conclusion

This paper presents Curriculum Abductive Learning (C-ABL), a method designed to improve the training stability and efficiency of Abductive Learning (ABL). Unlike prior methods that treat the knowledge base as a fixed black box, C-ABL leverages structural properties of knowledge base, partitions the logic into sub-bases and introduces them progressively. This curriculum design reduces the abduction space, stabilizes supervision, and enables the model to incorporate domain knowledge from simple to complex. We provide theoretical guarantees for phase-level improvements and smooth transitions, and demonstrate strong empirical results on synthetic and real-world tasks. By structurally organizing domain knowledge, C-ABL offers a principled path toward scalable and efficient ABL.

Acknowledgements

This research was supported by the Jiangsu Science Foundation Leading-edge Technology (BK20232003) and the National Natural Science Foundation of China (62176117). Cunjing Ge was supported by the National Natural Science Foundation of China (62202218). The authors would like to thank Reviewer #dk43 of OpenReview for the thorough review and insightful suggestions.

References

- E. Abbe, S. Bengio, A. Lotfi, and K. Rizk. Generalization on the unseen, logic reasoning and degree curriculum. In *International Conference on Machine Learning*, pages 31–60. PMLR, 2023.
- K. Ahmed, E. Wang, K.-W. Chang, and G. Van den Broeck. Neuro-symbolic entropy regularization. In *Uncertainty in Artificial Intelligence*, pages 43–53. PMLR, 2022.
- E. Amir and S. McIlraith. Improving the efficiency of reasoning through structure-based reformulation. In *International Symposium on Abstraction, Reformulation, and Approximation*, pages 247–259. Springer, 2000.
- E. Amir and S. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artificial intelligence*, 162(1-2):49–88, 2005.
- S. Badreddine, A. d. Garcez, L. Serafini, and M. Spranger. Logic tensor networks. *Artificial Intelligence*, 303: 103649, 2022.

- M. Ben-Ari. *Mathematical logic for computer science*. Springer Science & Business Media, 2012.
- Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.
- G. Brewka, T. Eiter, and M. Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.
- L.-W. Cai, W.-Z. Dai, Y.-X. Huang, Y.-F. Li, S. H. Muggleton, and Y. Jiang. Abductive learning with ground knowledge base. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI’21)*, pages 1815–1821, 2021.
- M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799, 2008.
- X. Chen, C. Liang, A. W. Yu, D. Song, and D. Zhou. Compositional generalization via neural-symbolic stack machines. *Advances in Neural Information Processing Systems*, 33:1690–1701, 2020.
- G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.
- K. Conrad. Zorn’s lemma and some applications. *Expository papers*, 2016.
- W.-Z. Dai, Q. Xu, Y. Yu, and Z.-H. Zhou. Bridging machine learning and logical reasoning by abductive learning. *Advances in Neural Information Processing Systems*, 32, 2019.
- L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI 2007, Proceedings of the 20th international joint conference on artificial intelligence*, pages 2462–2467, 2007.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.
- E.-H. Gao, Y.-X. Huang, W.-C. Hu, X.-H. Zhu, and W.-Z. Dai. Knowledge-enhanced historical document segmentation and recognition. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI’24)*, pages 8409–8416, 2024.
- C. Glanois, Z. Jiang, X. Feng, P. Weng, M. Zimmer, D. Li, W. Liu, and J. Hao. Neuro-symbolic hierarchical rule induction. In *International Conference on Machine Learning*, pages 7583–7615. PMLR, 2022.
- S. Gocht and T. Balyo. Accelerating sat based planning with incremental sat solving. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, pages 135–139, 2017.
- H.-Y. He and M. Li. A learnability analysis on neuro-symbolic learning. *arXiv preprint arXiv:2503.16797*, 2025.
- H.-Y. He, H. Sun, Z. Xie, and M. Li. Ambiguity-aware abductive learning. In *Proceedings of The 41st International Conference on Machine Learning. Vienna, Austria*, volume 235, pages 18019–18042, Vienna, Austria, 2024. PMLR.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- P. Hitzler. *Neuro-symbolic artificial intelligence: The state of the art*. IOS Press, 2022.
- A. Horn. On sentences which are true of direct unions of algebras¹. *The Journal of Symbolic Logic*, 16(1): 14–21, 1951.
- W.-C. Hu, W.-Z. Dai, Y. Jiang, and Z.-H. Zhou. Efficient rectification of neuro-symbolic reasoning inconsistencies by abductive reflection. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI’25)*, pages 17333–17341, 2025.
- Y.-X. Huang, W.-Z. Dai, J. Yang, L.-W. Cai, S. Cheng, R. Huang, Y.-F. Li, and Z.-H. Zhou. Semi-supervised abductive learning and its application to theft judicial sentencing. In *Proceedings of the 20th IEEE International Conference on Data Mining (ICDM’20)*, pages 1070–1075, 2020.
- Y.-X. Huang, W.-Z. Dai, L.-W. Cai, S. H. Muggleton, and Y. Jiang. Fast abductive learning by similarity-based consistency optimization. *Advances in Neural Information Processing Systems*, 34:26574–26584, 2021.

- Y.-X. Huang, W.-Z. Dai, Y. Jiang, and Z.-H. Zhou. Enabling knowledge refinement upon new concepts in abductive learning. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI'23)*, pages 7928–7935, 2023.
- Y.-X. Huang, W.-C. Hu, E.-H. Gao, and Y. Jiang. Ablkit: a python toolkit for abductive learning. *Frontiers of Computer Science*, 18(6):186354, 2024.
- M. Huth and M. Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge university press, 2004.
- L.-H. Jia, S.-Y. Han, L.-Z. Guo, Z. Zhou, Z.-L. Li, Y.-F. Li, and Z.-H. Zhou. A smooth transition between induction and deduction: Fast abductive learning based on probabilistic symbol perception. *arXiv preprint arXiv:2502.12919*, 2025.
- P. T. Johnstone. *Stone spaces*, volume 3. Cambridge university press, 1982.
- J. R. Josephson and S. G. Josephson. *Abductive inference: Computation, philosophy, technology*. Cambridge University Press, 1996.
- A. Kikaj, G. Marra, F. Geerts, R. Manhaeve, and L. De Raedt. Deepgraphlog for layered neurosymbolic ai. *arXiv preprint arXiv:2509.07665*, 2025.
- A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- D. Le-Phuoc, T. Eiter, and A. Le-Tuan. A scalable reasoning and learning approach for neural-symbolic stream fusion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):4996–5005, May 2021. doi: 10.1609/aaai.v35i6.16633. URL <https://ojs.aaai.org/index.php/AAAI/article/view/16633>.
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Z. Li, J. Huang, and M. Naik. Scallop: A language for neurosymbolic programming. *Proceedings of the ACM on Programming Languages*, 7(PLDI):1463–1487, 2023.
- Z. Li, Z. Liu, Y. Yao, J. Xu, T. Chen, X. Ma, and J. Lü. Learning with logical constraints but without shortcut satisfaction. *arXiv preprint arXiv:2403.00329*, 2024.
- L. S. Lorello, M. Lippi, and S. Melacci. The kandy benchmark: Incremental neuro-symbolic learning and reasoning with kandinsky patterns. *arXiv preprint arXiv:2402.17431*, 2024.
- J. Maene, V. Derkinderen, and L. De Raedt. On the hardness of probabilistic neurosymbolic learning. *arXiv preprint arXiv:2406.04472*, 2024.
- L. Magnani et al. *Abductive cognition: The epistemological and eco-cognitive dimensions of hypothetical reasoning*, volume 3. Springer, 2009.
- R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt. Deepproblog: Neural probabilistic logic programming. *advances in neural information processing systems*, 31, 2018.
- R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. De Raedt. Neural probabilistic logic programming in deepproblog. *Artificial Intelligence*, 298:103504, 2021a.
- R. Manhaeve, G. Marra, and L. De Raedt. Approximate inference for neural probabilistic logic programming. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, pages 475–486. IJCAI Organization, 2021b.
- J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=rJgMlhRctm>.
- E. Marconato, G. Bontempo, E. Ficarra, S. Calderara, A. Passerini, and S. Teso. Neuro Symbolic Continual Learning: Knowledge, Reasoning Shortcuts and Concept Rehearsal. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023a.
- E. Marconato, S. Teso, A. Vergari, and A. Passerini. Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts. *Advances in Neural Information Processing Systems*, 36:72507–72539, 2023b.
- E. Marconato, S. Bortolotti, E. van Krieken, A. Vergari, A. Passerini, and S. Teso. Bears make neuro-symbolic models aware of their reasoning shortcuts. *arXiv preprint arXiv:2402.12240*, 2024.

- L. Serafini and A. d. Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*, 2016.
- S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
- R. Simiński. The experimental evaluation of rules partitioning conception for knowledge base systems. In *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology–ISAT 2016–Part I*, pages 79–89. Springer, 2017.
- M. H. Stone. The theory of representation for boolean algebras. *Transactions of the American Mathematical Society*, 40(1):37–111, 1936.
- L. Tao, Y.-X. Huang, W.-Z. Dai, and Y. Jiang. Deciphering raw data in neuro-symbolic learning with provable guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15310–15318, 2024.
- E. van Krieken, T. Thanapalasingam, J. Tomczak, F. Van Harmelen, and A. Ten Teije. A-nesi: A scalable approximate method for probabilistic neurosymbolic inference. *Advances in Neural Information Processing Systems*, 36:24586–24609, 2023.
- E. Van Krieken, P. Minervini, E. M. Ponti, and A. Vergari. On the independence assumption in neurosymbolic learning. *arXiv preprint arXiv:2404.08458*, 2024.
- J. Wang, D. Deng, X. Xie, X. Shu, Y.-X. Huang, L.-W. Cai, H. Zhang, M.-L. Zhang, Z.-H. Zhou, and Y. Wu. Tac-valuer: Knowledge-based stroke evaluation in table tennis. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD’21)*, pages 3688–3696, 2021a.
- X. Wang, Y. Chen, and W. Zhu. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576, 2021b.
- Z. Wang, J. Wang, X. Chen, M. Wang, M. Ma, Z. Wang, Z. Zhou, T. Yang, and W.-Z. Dai. From end-to-end to step-by-step: Learning to abstract via abductive reinforcement learning. In J. Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 6515–6523. International Joint Conferences on Artificial Intelligence Organization, 2025.
- W.-D. Wei, X.-W. Yang, J.-J. Shao, and L.-Z. Guo. Curriculum abductive learning for mitigating reasoning shortcuts. In J. Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 6533–6541, 2025.
- T. Winters, G. Marra, R. Manhaeve, and L. De Raedt. Deepstochlog: Neural stochastic logic programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10090–10100, 2022.
- J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International conference on machine learning*, pages 5502–5511. PMLR, 2018.
- X.-W. Yang, W.-D. Wei, J.-J. Shao, Y.-F. Li, and Z.-H. Zhou. Analysis for abductive learning and neural-symbolic reasoning shortcuts. In *Proceedings of The 41st International Conference on Machine Learning, Vienna, Austria*, Vienna, Austria, 2024. PMLR.
- X.-W. Yang, J.-J. Shao, L.-Z. Guo, B.-W. Zhang, Z. Zhou, L.-H. Jia, W.-Z. Dai, and Y.-F. Li. Neuro-symbolic artificial intelligence: Towards improving the reasoning abilities of large language models. In J. Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 10770–10778, 2025. Survey Track.
- Z. Yang, A. Ishay, and J. Lee. Neurasp: Embracing neural networks into answer set programming. In *29th International Joint Conference on Artificial Intelligence*, 2020.
- Z. Yang, J. Lee, and C. Park. Injecting logical constraints into neural networks via straight-through estimators. In *International Conference on Machine Learning*, pages 25096–25122. PMLR, 2022.
- Y. Yu, H. Qian, and Y.-Q. Hu. Derivative-free optimization via classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.
- Z.-H. Zhou. Abductive learning: towards bridging machine learning and logical reasoning. *Science China Information Sciences*, 62:1–3, 2019.
- Z.-H. Zhou and Y.-X. Huang. Abductive learning. In P. Hitzler and M. K. Sarker, editors, *Neuro-Symbolic Artificial Intelligence: The State of the Art*, pages 353–369. IOS Press, Amsterdam, 2022.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: See [Section 2](#), [Section 3](#), [Section 4](#).

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We have provided limitation discussion in [Appendix A](#).

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: See [Section 3.1](#), [Section 4](#), [Appendix C](#).

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: See [Section 5](#), [Appendix F](#).

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have provided code in the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: See [Section 5, Appendix F](#).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: See [Section 5, Appendix F](#).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: See [Section 5](#), [Appendix B.2](#), [Appendix F](#).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics [https://neurips.cc/public/EthicsGuidelines?](https://neurips.cc/public/EthicsGuidelines)

Answer: [Yes]

Justification: We have followed the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Related contents can be found in [Section 5](#), [Section 6](#), [Section 7](#), [Appendix A](#).

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: See [Section 5](#), [Appendix F](#), and the supplementary material.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their [licensing guide](#) can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A Limitation

While Curriculum Abductive Learning (C-ABL) demonstrates substantial improvements in training stability and reasoning performance, it also has several limitations. First, our partitioning strategy assumes that the knowledge base exhibits certain structural regularity—such as modularity or dependency chains—that supports meaningful curriculum design. This assumption holds for many real-world domains, including law, mathematics, medicine, structured game, etc. However, in some cases where the rule set is densely entangled or lacks clear structure, such partitioning may be less effective. Second, the current design of phase transitions is based on fixed structural principles and a simple accuracy threshold; more adaptive or learnable curriculum schedules could further improve performance but remain unexplored. Third, like standard ABL, C-ABL assumes access to a pre-defined, human-curated knowledge base. While our focus is on leveraging such structured knowledge, future work may explore methods for acquiring or refining the knowledge base, potentially by leveraging information from data.

B Additional Analysis of Algorithm 1

B.1 Examples of Knowledge Base Partitioning

This section illustrates examples of partitioning the knowledge base into sub-bases based on Algorithm 1 in Section 2.1.

Chess Attack. We use the chess attack task as the first example. As stated in Section 5.2, in this task, the input is a chessboard configuration with several pieces, and the concept label set is $\mathcal{Z} = \{\text{rook, pawn, bishop, king, knight, queen}\}$. \mathcal{KB} contains rules of the attack behavior for each piece, with the target label $y = \text{attack}$, indicating whether any pair of pieces in the chessboard are in an attacking relation.

Figure 5 illustrates a portion the aforementioned \mathcal{KB} , and provides a visual example of how the dependency graph is constructed as well as how sub-base clusters are formed based on Algorithm 1.

Each node in the dependency graph corresponds to a rule in the knowledge base, and a directed edge is drawn from rule r_i to r_j if the head predicate of r_i appears in the body of r_j . For example, in the top portion of the figure, the rule for `lshape` depends on both `left` and `fwd`, and in turn supports the rule for `attack` when the piece is a knight. This forms a dependency path: `left, fwd` \rightarrow `lshape` \rightarrow `attack`.

Our algorithm begins by constructing an initial cluster C_z for each concept label $z \in \mathcal{Z}$ by collecting all rules that directly reference z in their bodies. In the figure, these clusters are color-coded by background:

- The **green-shaded region** corresponds to C_{knight} , which includes the `attack` rule with concept knight, the `lshape` rule, and the supporting geometric rules `left` and `fwd`.
- The **red-shaded region** represents C_{rook} , which includes the `attack` rule with concept rook, and the supporting `line` predicate rules that it depends on.
- The **yellow-shaded region** represents C_{bishop} , which includes the `attack` rule with concept bishop, and the supporting `diag` predicate rules that it depends on.
- The **blue-shaded region** shows C_{queen} , which includes the `attack` rule involving queen and the `line_or_diag` predicate, along with its dependencies on both `line` and `diag`.

Edges in the figure indicate predicate-level dependencies between rules, which guide the recursive expansion of clusters and the establishment of inter-cluster precedence. For example, C_{queen} depends on both C_{rook} and C_{bishop} due to its reliance on `line` and `diag` via `line_or_diag`, so both C_{rook} and C_{bishop} must precede C_{queen} in the curriculum.

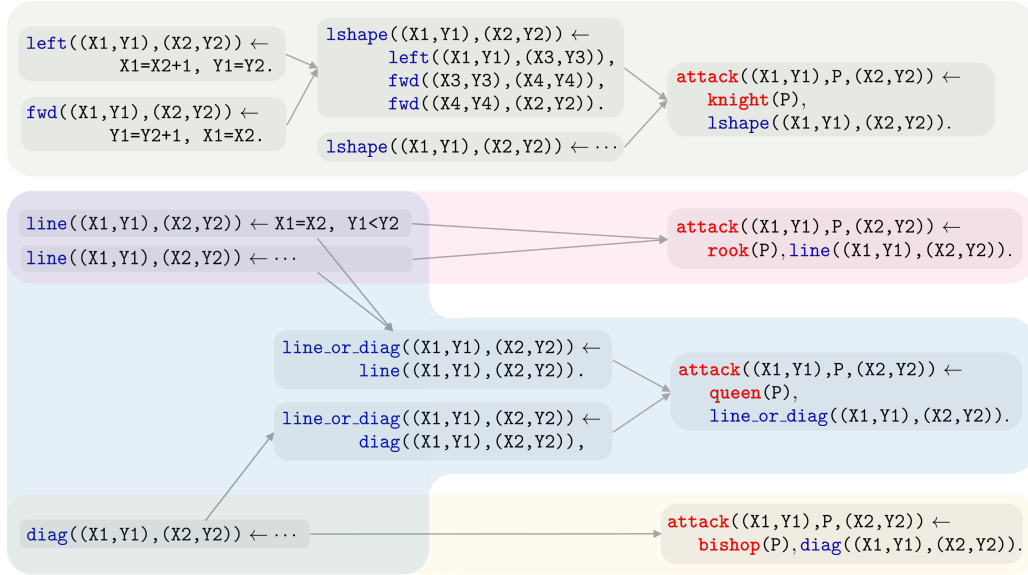


Figure 5: A portion of knowledge base in chess attack task.

The resulting sub-bases, for example, follow an incremental structure such as:

$$\begin{aligned}
 \mathcal{KB}_1 &= C_{\text{knight}} \\
 \mathcal{KB}_2 &= C_{\text{knight}} \cup C_{\text{rook}} \\
 \mathcal{KB}_3 &= C_{\text{knight}} \cup C_{\text{rook}} \cup C_{\text{bishop}} \\
 \mathcal{KB}_4 &= C_{\text{knight}} \cup C_{\text{rook}} \cup C_{\text{bishop}} \cup C_{\text{queen}}
 \end{aligned} \tag{2}$$

***d*-digit Addition.** In the digit addition task (outlined in [Example 2.1](#) and [Section 5.1](#)), the knowledge base \mathcal{KB} encodes multi-digit addition through symbolic predicates for digits and arithmetic operations. It includes logic for digit-wise computation and mappings from concept labels (e.g., zero, one) to numeric values, as illustrated below:

```

addition(Num1, Num2, Y) ← number(Num1, Res1),
                             number(Num2, Res2),
                             Y is Res1 + Res2.

number([], Res, Res) ←.
number([H|T], Acc, Res) ← digit(H, D),
                           Acc1 is D + 10 * Acc,
                           number(T, Acc1, Res).

number(X, N) ← number(X, 0, N).
digit(Pos, 0) ← zero(Pos).
digit(Pos, 1) ← one(Pos).
digit(Pos, 2) ← two(Pos).
digit(Pos, 3) ← three(Pos).
digit(Pos, 4) ← four(Pos).
digit(Pos, 5) ← five(Pos).
digit(Pos, 6) ← six(Pos).
digit(Pos, 7) ← seven(Pos).
digit(Pos, 8) ← eight(Pos).
digit(Pos, 9) ← nine(Pos).

```

When applying [Algorithm 1](#), rules are partitioned based on concept label dependencies. For example, \mathcal{KB}_1 may include rules for $\mathcal{Z}_1 = \{\text{zero}, \text{one}\}$, then expand to \mathcal{KB}_2 with $\mathcal{Z}_2 = \{\text{zero}, \text{one}, \text{two}, \text{three}\}$, and so on. Each \mathcal{KB}_p includes (1) digit-mapping rules relevant to \mathcal{Z}_p , and (2) shared arithmetic rules for addition (rules with predicate `addition` as head) and number construction (rules with predicate `number` as head). While the core arithmetic logic remains fixed across phases, digit-related rules are introduced gradually.

The above two visualizations on chess attack and digit addition make clear how our algorithm traces dependencies to group semantically related rules into coherent sub-bases, and defines a curriculum order based on the reasoning hierarchy encoded in the knowledge base structure.

B.2 Computational Cost Analysis

We now analyze the computational overhead introduced by the knowledge base partitioning algorithm, i.e., [Algorithm 1](#) proposed in [Section 3.1](#).

Time Complexity. Let n and e be the number of nodes (also, number of rules in \mathcal{KB}) and edges in the dependency graph of \mathcal{KB} . The partitioning process involves the following major steps:

1. **Dependency Graph Construction:** For each rule, we inspect its head and compare it against all other rule bodies. This results in $\mathcal{O}(n^2)$ comparisons in the worst case. Also, since predicates tend to be sparse in practice, a hash-based indexing scheme can reduce this to $\mathcal{O}(n + e)$.
2. **Cluster Initialization and Expansion:** For each concept label $z \in \mathcal{Z}$ (with $|\mathcal{Z}| = N$), we initiate a cluster and expand it by traversing the dependency graph, the overall time complexity of this process is $\mathcal{O}(N(n + e))$.
3. **Precedence Ordering and Topological Sort:** After forming clusters, we identify precedence relations based on cross-cluster dependencies. This forms a directed acyclic graph over clusters, with topological sorting in $\mathcal{O}(N + e_c)$ time, where e_c is the number of inter-cluster edges (typically $e_c \ll e$).

Putting all steps together, the overall complexity is approximately $\mathcal{O}(N(n + e))$ under practical sparsity assumptions.

Empirical Partition Time. In practice, we evaluated the partitioning time on the two examples described in [Appendix B.1](#). For the digit addition task, the knowledge base contains 14 rules, and partitioning takes 0.05 seconds. For the chess attack task, which includes 58 rules, partitioning completes in 0.9 seconds. All measurements were conducted on a standard CPU (Intel Xeon Gold 6226R, single thread), with no GPU required.

To put this in context, a full ABL training run (e.g., under 1,000 iterations) typically takes several minutes to hours, depending on the task and model. Therefore, the knowledge base partitioning accounts for a minimal portion of the total runtime. More importantly, it is a **one-time, offline preprocessing** step, independent of data size or iteration count, and its cost amortizes over training. In summary, [Algorithm 1](#) incurs negligible computational overhead while providing benefits in reasoning efficiency, supervision quality, and convergence speed.

C Theorem Proof

C.1 Proof of [Theorem 3.2](#)

Proof. We prove the two directions separately.

Soundness. This follows from the fact that $\mathcal{KB}_p \subseteq \mathcal{KB}$. Any logical derivation that holds under \mathcal{KB}_p also holds under the full knowledge base \mathcal{KB} , since \mathcal{KB} contains all rules in \mathcal{KB}_p . Therefore, we have:

$$\mathcal{KB}_p \models \varphi \implies \mathcal{KB} \models \varphi. \quad (3)$$

Completeness. We assume that $\mathcal{KB} \models \varphi$, i.e., there exists a proof (derivation) of φ using rules from \mathcal{KB} . Let $\mathcal{R} \subseteq \mathcal{KB}$ denote the minimal set of rules used in this derivation. We now argue that all rules in \mathcal{R} are also contained in \mathcal{KB}_p .

As stated in [Algorithm 1](#), for any predicate $z \in \mathcal{Z}_p$, sub-base involving it is constructed by: (1) Create a cluster C_z by collecting all rules whose bodies mention z . (2) Recursively expand C_z by traversing the dependency graph G : If a rule depends on a predicate a , and a is concluded by another rule, then that rule is also included. (3) Continue until all rules needed to derive any conclusions involving z are included. Therefore, for each $z \in \mathcal{Z}_p$, all reasoning chains involving z and the predicates it depends on are included in \mathcal{KB}_p by construction. Since by assumption, φ only involves predicates in \mathcal{Z}_p , and the derivation of φ in \mathcal{KB} uses only predicates and rules in these chains, we conclude that $\mathcal{R} \subseteq \mathcal{KB}_p$.

Hence, the same inference of φ can be carried out within \mathcal{KB}_p . Therefore, we have:

$$\mathcal{KB} \models \varphi \implies \mathcal{KB}_p \models \varphi. \quad (4)$$

□

C.2 Proof of [Corollary 3.3](#)

Proof. Since the final phase P includes all clusters and thus covers all predicates in \mathcal{Z} , we have $\mathcal{Z}_P = \mathcal{Z}$. Therefore, for any formula φ over \mathcal{Z} , we trivially have:

$$\mathcal{KB}_P \models \varphi \iff \mathcal{KB} \models \varphi. \quad (5)$$

□

C.3 Proof of [Lemma 4.1](#)

Proof. Each of the m positions independently selects a label from \mathcal{Z} , giving N^m possible assignments in total. Logical constraints may eliminate some assignments, but the maximum remains N^m . □

C.4 Proof of [Theorem 4.2](#)

Proof. We first denote $\mathcal{Z}_p = \mathcal{Z}_{p-1} \cup \Delta\mathcal{Z}_p$, where $\Delta\mathcal{Z}_p = \mathcal{Z}_p \setminus \mathcal{Z}_{p-1}$ are the new concept labels introduced in phase p , and for any $z \in \mathbb{S}_p$, we write $z = (z_{\text{old}}, z_{\text{new}})$ accordingly.

Recall that the consistency score used for pseudo-label selection is:

$$\text{Con}(z, f) = \prod_{i=1}^m p^{(i)}(z^{(i)}). \quad (6)$$

where $p^{(i)}(z^{(i)})$ denotes the model’s predicted probability for concept $z^{(i)}$ at position i .

Now fix two candidate concept sequences $z^{(a)}$ and $z^{(b)}$ in \mathbb{S}_p , such that: $z^{(a)}$ and $z^{(b)}$ agree on all positions i where $z^{(i)} \in \mathcal{Z}_{p-1}$, but disagree at some positions i where $z^{(i)} \in \Delta\mathcal{Z}_p$. Then their consistency scores differ only in terms involving $\Delta\mathcal{Z}_p$. By the semantic stability assumption, previously learned concepts retain their meaning across phases. Combined with the assumption that for any previously introduced concept $z \in \mathcal{Z}_{p-1}$, the model achieves accuracy exceeding random chance, so that for positions i with $z^{(i)} \in \mathcal{Z}_{p-1}$, the predicted probabilities $p^{(i)}(z^{(i)})$ tend to assign higher mass to correct concepts learned in prior phases. Therefore we have $\text{Con}(z^{(a)}, f) > \text{Con}(z^{(b)}, f)$, meaning the selection strategy will typically prefer $z^{(a)}$ over $z^{(b)}$.

This shows that, under consistency-based selection, the model tends to preserve earlier learned concepts (those in \mathcal{Z}_{p-1}), and variations in \mathbb{S}_p primarily arise from the newly introduced concepts $\Delta\mathcal{Z}_p$. Therefore, the number of valid configurations in \mathbb{S}_p is bounded by the number of possible combinations of z_{new} :

$$|\mathbb{S}_p| \leq |\Delta\mathcal{Z}_p|^m = |\mathcal{Z}_p \setminus \mathcal{Z}_{p-1}|^m. \quad (7)$$

□

C.5 Proof of Theorem 4.5

Proof. Let the loss function at iteration t be defined as:

$$L^{(t)} = \ell(f^{(t)}(x), \bar{z}^{(t)}), \quad (8)$$

where $\bar{z}^{(t)} \in \mathbb{S}$ is selected via abduction given y and \mathcal{KB} , we aim to bound the average suboptimality of the iterates:

$$\frac{1}{T} \sum_{t=1}^T \mathbb{E}[L^{(t)} - L^*] \leq \varepsilon, \quad (9)$$

where $L^* = \min_f \ell(f(x), z^*)$ is the best possible loss under the true concept label z^* .

By chapter 14 in [Shalev-Shwartz and Ben-David \[2014\]](#), under the assumptions that: (1) loss function $L^{(t)}$ is convex, (2) the gradient norm is bounded by the Lipschitz constant ρ , i.e., $\|\nabla L^{(t)}\| \leq \rho$, and (3) the model lies in a bounded domain of diameter D , then the regret over T steps is bounded by:

$$\sum_{t=1}^T L^{(t)} - L^* \leq \frac{D^2}{2\eta} + \frac{\eta T \rho^2}{2}. \quad (10)$$

Choosing the optimal learning rate $\eta = \frac{D}{\rho\sqrt{T}}$, we obtain:

$$\frac{1}{T} \sum_{t=1}^T \mathbb{E}[L^{(t)} - L^*] \leq \frac{D\rho}{\sqrt{T}}. \quad (11)$$

To ensure this average suboptimality is below ε , it suffices that:

$$\frac{D\rho}{\sqrt{T}} \leq \varepsilon \Rightarrow T \geq \left(\frac{D\rho}{\varepsilon}\right)^2. \quad (12)$$

Now, suppose that due to the abductive process, at each iteration t , the pseudo-label $\bar{z}^{(t)}$ is selected uniformly from a candidate set of size $|\mathbb{S}|$, then the effective variance or inconsistency in supervision may grow linearly with $|\mathbb{S}|$. To conservatively account for this, we assume the domain diameter $D = \mathcal{O}(|\mathbb{S}|)$, which reflects the number of pseudo-label variants over which the model needs to generalize. Therefore, we have

$$T = \mathcal{O}\left(\frac{|\mathbb{S}|^2 \cdot \rho^2}{\varepsilon^2}\right). \quad (13)$$

which completes the proof. \square

C.6 Proof of Theorem 4.6

Proof. Recall that for each phase p , we have $\mathcal{KB}_p \subseteq \mathcal{KB}_{p+1}$ and that $\mathcal{Z}_p \subseteq \mathcal{Z}_{p+1}$, therefore, by Theorem 3.2, both \mathcal{KB}_p and \mathcal{KB}_{p+1} are sound and complete with respect to all formulas over \mathcal{Z}_p . Then, for any formula φ over \mathcal{Z}_p , we have:

$$\mathcal{KB}_p \models \varphi \iff \mathcal{KB} \models \varphi \iff \mathcal{KB}_{p+1} \models \varphi \quad (14)$$

where the first equivalence comes from applying Theorem 3.2 to \mathcal{KB}_p and the second from applying it to \mathcal{KB}_{p+1} , since both sub-bases fully capture all reasoning over the same concept domain \mathcal{Z}_p .

Hence, $\mathcal{KB}_p \models \varphi$ if and only if $\mathcal{KB}_{p+1} \models \varphi$, completing the proof. \square

C.7 Background on Stone Spaces and Proof of Theorem 4.7

We first provide backgrounds on stone spaces [[Johnstone, 1982](#)]: Stone spaces provide a topological perspective on logic by characterizing the space of all models (truth assignments) that satisfy a given set of logical formulas. Formally, given a Boolean algebra \mathcal{B} , its Stone space $S(\mathcal{B})$ is defined as the

Algorithm 2: Curriculum Abductive Learning (C-ABL) Training

Input: Sub-bases $\mathcal{KB}_1, \dots, \mathcal{KB}_P$, training dataset \mathcal{D} **Output:** Trained model f

```
1 Initialize  $f$  randomly;
2 for  $p = 1$  to  $P$  do
3   Schedule training stream  $\mathcal{D}_p \subseteq \mathcal{D}$  with concept labels in  $\mathcal{Z}_p$ ;
4   repeat
5     for each  $x \in \mathcal{D}_p$  do
6        $\hat{z} \leftarrow f(x)$ ;
7       if  $\hat{z} \wedge \mathcal{KB}_p \models y$  then
8          $\bar{z} \leftarrow \hat{z}$ ;
9       else
10         $\mathbb{S} \leftarrow \{z \mid z \wedge \mathcal{KB}_p \models y\}$ ;
11         $\bar{z} \leftarrow \arg \min_{z \in \mathbb{S}} 1 - \prod_i p^{(i)}(z^{(i)})$   $\triangleright$  from Eq. (1);
12        Update  $f$  using  $\bar{z}$ ;
13 until Phase transition condition met ( $p < P$ ) or termination condition met ( $p = P$ );
```

set of all ultrafilters over \mathcal{B} , where an ultrafilter is a maximal consistent set of formulas—intuitively, it corresponds to a complete and consistent truth assignment over the logic.

Stone’s representation theorem [Stone, 1936] states that every Boolean algebra is isomorphic to the algebra of clopen (simultaneously closed and open) subsets of its Stone space. This construction enables logical entailment to be interpreted as topological containment: if $\varphi \in \mathcal{B}$ holds in all ultrafilters in $S(\mathcal{B})$, then φ is a logical consequence of \mathcal{B} .

In our setting, for each phase p , we define a Boolean algebra \mathcal{B}_p as the closure of all logical formulas over the predicate set \mathcal{Z}_p . The Stone space $S(\mathcal{B}_p)$ thus encodes all logically consistent concept-level assignments admissible under \mathcal{KB}_p .

Below is the proof of Theorem 4.7.

Proof. Recall that for each phase p , we construct a sub-base \mathcal{KB}_p such that $\mathcal{KB}_p \subseteq \mathcal{KB}_{p+1}$ and that $\mathcal{Z}_p \subseteq \mathcal{Z}_{p+1}$. Let \mathcal{B}_p denote the Boolean algebra of formulas over \mathcal{Z}_p , and \mathcal{B}_{p+1} denote that over \mathcal{Z}_{p+1} . Since $\mathcal{Z}_p \subseteq \mathcal{Z}_{p+1}$, every formula over \mathcal{Z}_p is also a formula over \mathcal{Z}_{p+1} , that is, the Boolean algebras satisfies $\mathcal{B}_p \subseteq \mathcal{B}_{p+1}$.

We now argue that any ultrafilter over \mathcal{B}_p can be extended to an ultrafilter over \mathcal{B}_{p+1} : Following Zorn’s Lemma [Conrad, 2016], any consistent set of formulas over a Boolean algebra can be extended to an ultrafilter over a larger Boolean algebra containing it, therefore, any model over \mathcal{Z}_p can be extended to a model over \mathcal{Z}_{p+1} that preserves the truth values of formulas in \mathcal{B}_p . Then, for every ultrafilter $U_p \in S(\mathcal{B}_p)$, there exists at least one ultrafilter $U_{p+1} \in S(\mathcal{B}_{p+1})$ such that $U_{p+1} \cap \mathcal{B}_p = U_p$. This directly implies:

$$S(\mathcal{B}_p) \subseteq S(\mathcal{B}_{p+1}), \tag{15}$$

since every model in phase p remains valid (or extendable) in phase $p + 1$.

□

D Pseudocode for Curriculum Abductive Learning

This section shows the pseudocode (see Algorithm 2) for the Curriculum Abductive Learning (C-ABL) training process, as stated in Section 3.2.

E Comparison Methods

In this section, we provide a brief supplementary introduction to the compared baseline methods used in experiments.

ABL and its Variants.

- (1) **ABL** [Dai et al., 2019], the original implementation of Abductive Learning, supported by an efficient open-source toolkit [Huang et al., 2024]. For consistency evaluation, we adopt the confidence-based selection strategy as defined in Section 2.3 (see Eq. (1));
- (2) **A³BL** [He et al., 2024], one of the most effective ABL variants, which improves concept label selection by evaluating all candidates in the abduction space.

Other variants such as Huang et al. [2021], Tao et al. [2024] also focus on refining label selection but perform less robustly than A³BL. We do not compare with methods like ABL-Refl [Hu et al., 2025] or ABL-PSP [Jia et al., 2025], as they process the entire input sequence jointly and hence require different perception modules, making direct comparison less meaningful.

Neuro-Symbolic Methods.

- (1) **NeurASP** [Yang et al., 2020], an extension of answer set programs [Brewka et al., 2011] by treating the neural network output as the probability distribution over atomic facts;
- (2) **LTN** [Serafini and Garcez, 2016], a neural-symbolic framework that uses differentiable first-order logic language. This is a representative work of relaxing logical rules as soft constraints in neural networks;
- (3) **DeepProbLog** [Manhaeve et al., 2018], an extension of ProbLog [De Raedt et al., 2007] by introducing neural predicates;
- (4) **DeepStochLog** [Winters et al., 2022], a related strategy to DeepProbLog that improves inference efficiency through stochastic logic program.

F Experiment Details and Additional Results

All experiments are performed on a server with Intel Xeon Gold 6226R CPU and Tesla A100 GPU, and each experiment is repeated 5 times.

F.1 d -digit Addition Tasks

Curriculum Design. The curriculum partitioning of \mathcal{KB} for this task is detailed in Appendix B.1. We set the minimum phase size to $\tau = 2$.

Additionally, Table 4 presents a sensitivity analysis on different values of τ , conducted under the hexadecimal addition setting with $d = 3$. In this configuration, there are a total of 16 concept labels, and we experimented with τ values ranging from 1 to 4. Results show that our method consistently outperforms both ABL and A³BL across different τ settings. In general, a smaller τ leads to finer-grained sub-bases and a smaller abduction space, which often improve reasoning accuracy and training efficiency. Indeed, when the phases are too fine-grained, the model will go through more curriculum, which slightly brings implementation complexity.

Cross-Dataset Analysis. We further validate our method on alternative datasets. In the decimal setting, we now use digit images from MNIST dataset [Krizhevsky et al., 2009], and in the hexadecimal setting, we use digits 0–9 and letters A–F from EMNIST dataset [Cohen et al., 2017]. All methods use LeNet-5 [LeCun et al., 1998] as the perception model, and are trained for a total of 1,000 iterations. Results in Table 5 show that C-ABL continues to yield superior reasoning accuracy and training efficiency compared to baselines.

Full Training Curve (Complementing Figure 3). Previously in Figure 3, we illustrated a segment of the training curves for three randomly selected concept labels, each introduced in a different phase (with three vertical dashed lines indicating the start of their respective phases). Here in Figure 6, we present the full training curve of these concept labels (left). We also provide a zoomed-in view (right) focusing on the early training period of C-ABL, which highlights that in C-ABL, each concept label improves immediately and stably upon the start of its corresponding curriculum phase, while the performance of previously learned concepts remains unaffected—demonstrating the smoothness and stability of phase transitions.

Table 4: Sensitivity analysis on different τ values across different tasks. C-ABL outperforms ABL and A³BL across different τ settings.

Method	Digit Addition		Chess Attack		
	Test Accuracy	Training Time (min)	Test Accuracy	Training Time (min)	
ABL	64.14 \pm 1.32	99.7 \pm 5.2	73.75 \pm 0.84	36.7 \pm 2.8	
A ³ BL	65.81 \pm 1.12	109.8 \pm 6.8	74.96 \pm 0.78	45.6 \pm 3.2	
C-ABL	$\tau = 1$	66.78 \pm 1.05	53.7 \pm 2.7	85.86 \pm 0.74	20.9 \pm 1.6
	$\tau = 2$	66.67 \pm 1.11	55.5 \pm 4.2	86.79 \pm 0.66	23.1 \pm 1.8
	$\tau = 3$	65.76 \pm 1.08	64.8 \pm 4.9	84.69 \pm 0.71	26.3 \pm 2.1
	$\tau = 4$	66.01 \pm 1.16	70.1 \pm 5.1	–	–

Table 5: Comparison of prediction accuracy (top) and total training time in minutes (bottom) on digit addition tasks with MNIST / EMNIST dataset. “N/A” indicates runtime exceeding 3 hours. C-ABL consistently achieves higher accuracy and faster training.

Method	Decimal				Hexadecimal		
	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 1$	$d = 2$	$d = 3$
NeurASP	97.01 \pm 0.24	10.21 \pm 0.39	N/A	N/A	77.65 \pm 6.29	N/A	N/A
DeepProbLog	97.42 \pm 0.29	N/A	N/A	N/A	96.59 \pm 0.48	N/A	N/A
DeepStochLog	99.00 \pm 0.13	98.72 \pm 0.14	98.59 \pm 0.12	98.46 \pm 0.11	97.56 \pm 0.12	97.96 \pm 0.11	97.13 \pm 0.15
LTN	98.64 \pm 0.18	98.53 \pm 0.21	98.48 \pm 0.17	98.34 \pm 0.19	97.18 \pm 0.14	96.53 \pm 0.17	97.11 \pm 0.13
ABL	98.79 \pm 0.10	98.99 \pm 0.08	98.59 \pm 0.09	98.29 \pm 0.11	97.17 \pm 0.10	97.49 \pm 0.08	97.50 \pm 0.06
A ³ BL	98.94 \pm 0.12	99.03 \pm 0.09	98.68 \pm 0.10	98.42 \pm 0.12	96.76 \pm 0.12	97.62 \pm 0.09	97.65 \pm 0.07
C-ABL	99.04\pm0.06	99.17\pm0.05	99.09\pm0.04	99.17\pm0.03	98.48\pm0.03	98.41\pm0.04	98.83\pm0.02
NeurASP	33.1 \pm 1.6	87.8 \pm 2.2	N/A	N/A	48.3 \pm 4.6	N/A	N/A
DeepProbLog	44.8 \pm 1.5	N/A	N/A	N/A	69.8 \pm 7.7	N/A	N/A
DeepStochLog	3.9 \pm 0.5	7.6 \pm 0.9	38.5 \pm 1.4	166.8 \pm 6.3	6.8 \pm 0.5	24.0 \pm 2.0	139.4 \pm 9.1
LTN	4.2 \pm 0.4	8.0 \pm 1.0	29.1 \pm 1.6	140.2 \pm 5.9	7.1 \pm 0.6	24.5 \pm 2.2	155.0 \pm 13.3
ABL	1.9 \pm 0.2	2.5 \pm 0.3	11.9 \pm 1.2	80.3 \pm 7.8	1.4 \pm 0.2	4.8 \pm 0.7	30.8 \pm 2.4
A ³ BL	2.9 \pm 0.3	4.6 \pm 0.5	15.5 \pm 1.7	105.8 \pm 6.5	3.2 \pm 0.1	6.5 \pm 0.6	54.2 \pm 3.7
C-ABL	0.9\pm0.1	1.8\pm0.2	6.9\pm0.7	33.2\pm2.3	1.0\pm0.1	3.4\pm0.4	20.6\pm2.1

F.2 Chess Attack Task

Curriculum Design. The partitioning strategy for this task is also described in [Appendix B.1](#). We again set the minimum phase size to $\tau = 2$. A sensitivity analysis of this hyperparameter is provided also in [Table 4](#), where we evaluate τ values ranging from 1 to 3 under the setting with 6 concept labels. Results show performance advantage of C-ABL over ABL and A³BL in almost all tested τ values. As in the digit addition task, a smaller τ introduces more fine-grained sub-bases and smaller abduction spaces, which may lead to improved accuracy or faster convergence.

F.3 Judicial Sentencing Tasks

Dataset Description. The dataset consists of 687 criminal judgment records for theft cases authored by judges in Guizhou, China, between 2017 and 2018 [[Huang et al., 2020](#)]. Each record includes detailed descriptions of the defendant’s background and criminal record, as well as some facts and legal basis of the ruling.

Knowledge Base. The legal domain offers rich structured prior knowledge that can be encoded as first-order logic rules, forming the knowledge base \mathcal{KB} . These include statutory law, attribute-matching rules, and general common sense constraints. Below is an example adapted from [Huang et al. \[2020\]](#), which outlines how various attributes and the stolen amount of money determine the final sentence:

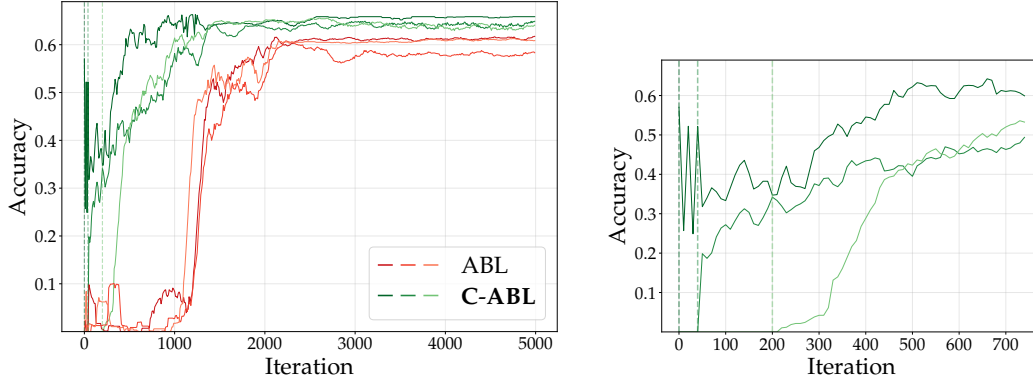


Figure 6: Full training curves of three concept labels introduced in different phases (left) and a zoomed-in early-stage view for C-ABL (right).

```

penalty(X, Y) ← base_penalty(X, Z1) ∧ weight(X, Z2) ∧ Y = Z1(1 + Z2).
base_penalty(X, Y) ← money(X, m) ∧ Y = 0.7m + 5.7.
weight([], 0) ← .
weight([X|Xs], Y) ← element_weight(X, Z1) ∧ weight(Xs, Z2) ∧ Y = Z1 + Z2.

```

However, such symbolic knowledge is often incomplete—for instance, the rule “voluntary surrender reduces sentence” may exist, but the precise reduction is unspecified. To incorporate these partially defined patterns into a learnable system, we parametrize the knowledge base as:

$$y = (1 + w^T z)(am + b), \quad (16)$$

where z is a binary vector representing predicted case attributes (i.e., concept labels), $m \in \mathbb{N}$ is the amount of money involved (a ground-truth numeric input), and $y \in \mathbb{R}^+$ is the predicted sentence length. This formulation mirrors the logical structure of first-order rules while allowing the unknown parameters w , a , and b to be learned via a linear regression model. Like the original rule-based system, Eq. (16) supports abduction: given a known sentence length y and monetary amount m , the model can infer plausible case attributes z .

In summary, this task requires both the prediction of concept labels and the joint optimization of symbolic parameters to produce sentence predictions directly from judgment documents. The integration of symbolic reasoning and statistical learning makes this possible in a unified framework.

Curriculum Design. In C-ABL, the knowledge base is partitioned according to the number of case attributes involved: earlier phases include rules with fewer attributes, while later phases expand to incorporate more. For example, the initial sub-knowledge bases contain rules for basic case attributes, such as whether the defendant is a repeat offender. In contrast, later phases include rules incorporating additional factors, such as the defendant’s cooperation with law enforcement, whether the crime involves mitigating circumstances, etc. Specifically, we define the sub-knowledge base for phase p as:

$$\mathcal{KB}_p = \{y = (1 + w^T z)(am + b) \mid \mathbf{1}^T z \leq p\}, \quad (17)$$

ensuring that earlier phases focus on simpler cases, thereby facilitating a smoother and more stable training process.