From Knowledge Concepts to Process-Based Skills: A Feedback-Driven Ability Induction Framework for Student Modeling

Anonymous ACL submission

Abstract

Ability representations are central to student modeling because they connect item requirements to students' observed responses. Most existing approaches use expert-defined knowledge concepts as ability units, which describe *what* content an item involves but often miss *how* it is solved when solving requires multi-step procedures. We study a process-based alternative by inducing ability units from solution procedures. Specifically, we treat large language model (LLM) reasoning traces as noisy process observations, and propose a closed-loop framework that extracts operation-level signals from structured LLM reasoning and consolidates them into stable, reusable process-based abilities. To reduce noise and keep induced abilities aligned with the target items, we combine semantic consolidation with a global feasibility signal from downstream student modeling, and use a TextGrad-style discrete controller to refine generation constraints and consolidation hyperparameters. Experiments on two benchmark datasets show that replacing expert concepts with induced process-based abilities improves student modeling performance and yields coherent ability structures with informative granularity and coverage.[1]

1 Introduction

A central goal in intelligent education is to infer learners' ability states from observed learning interactions for assessment and personalized instruction (Wang et al., 2020; Abdelrahman et al., 2023; Dong et al., 2025). Most student modeling pipelines, including cognitive diagnosis (CD) (Liu et al., 2023a) and knowledge tracing (KT) (Song et al., 2022; Scarlatos et al., 2025), rely on expert-defined knowledge concepts as ability units. Yet these concept-based representations mainly capture *what* content a problem involves, not *how* it is solved. This limitation is especially pronounced in multi-step problems, where a single concept label may obscure multiple reusable procedures (Figure 1). This observation motivates a complementary representation question:

> *Can we automatically induce reusable, process-based ability units that replace expert-defined concepts while remaining compatible with existing student modeling frameworks?*

Recent advances in Large Language Models (LLMs) make process-based ability induction feasible by explicitly externalizing intermediate reasoning steps in text, for example, through Chain-of-

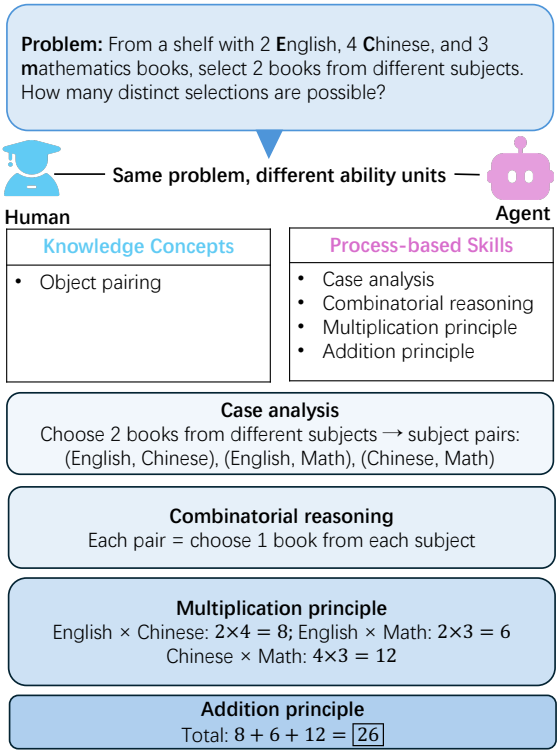

Figure 1: Comparison between knowledge-based and process-based ability representations on a multi-step combinatorial counting problem.

[1]Anonymous code repository: https://anonymous.4open.science/status/ProcessAbilityInduction

Thought prompting (Wei et al., 2022; Liang et al., 2023). By exposing sequences of operations, intermediate states, and reasoning order, such traces provide a rare opportunity to observe otherwise implicit problem-solving processes. However, these traces often exhibit substantial stylistic variability, redundancy, and noise (Turpin et al., 2023; Tutek et al., 2025). As a result, directly treating raw reasoning text as ability representations is unreliable. Nevertheless, when aggregated across many problems, LLM-generated traces exhibit recurring structural regularities, such as repeated operation types and reasoning patterns. We therefore treat reasoning traces as noisy but information-rich process-level observations, which require explicit abstraction and consolidation to yield stable and reusable procedural abilities.

Building on this perspective, we propose a framework for process-based ability induction that explicitly separates process observation and semantic abstraction. Rather than operating on individual items in isolation, our framework performs induction at the dataset level. At each iteration, a frozen LLM generates Structured Chain-of-Thought traces for each item under explicit generation constraints, ensuring that reasoning steps are annotated with normalized operation labels. The operation-level signals extracted from all items are then aggregated and consolidated into canonical procedural abilities through semantic clustering, after which each cluster is assigned a concise and reusable canonical name, yielding a global item–skill mapping that captures recurring procedural patterns shared across the dataset.

Textual coherence alone does not guarantee that induced skills are useful for student modeling. In particular, assigning skills independently to individual items can be misleading, as the effectiveness of a skill representation depends on how skill assignments across all items jointly support student modeling. To address this, we introduce a student modeling-based feasibility evaluation that assesses the quality of the induced item–skill mapping using validation performance on downstream student modeling tasks (Liu et al., 2023a; Scarlatos et al., 2025). This feasibility evaluation produces a single global signal based on validation performance of downstream student models. The signal reflects the quality of the induced ability representation at the dataset level, since ability assignments across different items must be considered jointly rather than evaluated item by item.

Finally, to allow this external feedback to guide induction, we adopt a TextGrad-style (Yuksekgonul et al., 2025) discrete controller. The controller translates feasibility feedback into updates of generation constraints and consolidation hyperparameters, enabling the induced skill space to progressively stabilize over multiple iterations. Throughout this process, the language model remains frozen and all improvements arise from feedback-driven refinement of generation constraints and induction hyperparameters.

In summary, our contributions are:

- We propose a feedback-driven framework for automatically inducing fine-grained, process-based ability units from LLM reasoning traces, providing an alternative to expert-defined knowledge concepts for student modeling.
- We design a structured induction pipeline that converts noisy reasoning traces into reusable procedural skills through explicit operation labeling and semantic consolidation.
- We introduce student modeling-based feasibility feedback and a TextGrad-style discrete controller to iteratively refine the induced ability space.
- We conduct experiments on two benchmark datasets and show consistent improvements over human-defined knowledge concepts in cognitive diagnosis and knowledge tracing, together with more informative skill structures.

2 Related Work

2.1 Ability Representation in Student Modeling

Cognitive Diagnosis (CD) and Knowledge Tracing (KT) are two core paradigms in student modeling. Most existing approaches rely on a predefined set of expert-defined knowledge concepts to link problem requirements with students' observed responses (Gao et al., 2022; Liu et al., 2019). Such concept-based representations are primarily content-oriented, describing *what* knowledge a problem involves rather than *how* it is solved.

Prior work has largely improved student modeling through model-level advances, including more expressive architectures, inference mechanisms, and temporal dynamics (Li et al., 2025; Pandey and Karypis, 2019; Minn et al., 2018; Cui et al., 2023), while assuming the underlying ability units to be

fixed and sufficient. Recent automated knowledge-component generation methods, such as KCGen-KT (Duan et al., 2025), relax manual concept annotation but still abstract knowledge components from generated descriptions and one-shot semantic clustering. In contrast, our work shifts the focus from model optimization to ability representation: we induce process-based ability units from language-model-generated reasoning traces, and use CD/KT predictive performance as evaluation signals for the resulting representations.

2.2 Chain-of-Thought as Process-Level Signals

Chain-of-Thought (CoT) prompting (Wei et al., 2022) enables large language models to produce intermediate reasoning steps in natural language, making problem-solving processes observable. Although CoT traces are not faithful explanations of internal model computation and can exhibit noise and stylistic variability (Turpin et al., 2023), aggregated traces often reveal recurring structural patterns, such as repeated operations and reasoning motifs.

Most existing work leverages CoT to improve reasoning performance, provide post-hoc explanations, or supervise model training (Plaat et al., 2025). In contrast, we treat CoT as a source of process-level observations rather than a final representation. Rather than analyzing individual CoT traces in isolation, we generate structured CoT traces for all items in the dataset and analyze them collectively at the system level. By aggregating and consolidating CoT-derived operations across multiple items, we aim to induce stable and reusable procedural abstractions that can serve as generalizable skill labels.

3 Methodology

3.1 Problem Setup and Overview

We consider an educational dataset consisting of assessment items (e.g., math word problems or programming exercises), denoted by $\mathcal{I}$. Our goal is to induce a reusable, process-based *skill space* that captures *how* items are solved, rather than only *what* content they assess. Formally, each item $i \in \mathcal{I}$ is associated with an induced skill set $\mathcal{S}_i \subseteq V$, where V is a global vocabulary of process-based skills.

A key challenge is that raw LLM reasoning traces are noisy and stylistically variable: the same procedure may appear under different surface forms, while overly aggressive merging can collapse distinct procedures. To address this, we employ an iterative closed-loop induction framework (Figure 2), in which the skill space is repeatedly refined through structured process observation, semantic consolidation, student modeling-based feasibility evaluation, and TextGrad-style discrete feedback updates.

System state. At induction round r, the system maintains the following components:

- a skill vocabulary $V^{(r)}$, defining the currently available operation labels;
- generation-side constraints $\mathcal{C}^{(r)}$, which comprise prompt instructions and structured schemas that govern all LLM-based generation steps in the induction process, including (i) structured process observation for producing step-wise reasoning traces, and (ii) semantic consolidation for assigning canonical names to operation clusters; these constraints encourage reuse of labels in $V^{(r)}$ while allowing `NEW:` labels only when necessary;
- consolidation hyperparameters $\Theta^{(r)}$, which control how extracted operation phrases are clustered and promoted into canonical skills.

We denote the complete system configuration at round r as $\text{State}^{(r)} = \left(V^{(r)}, \mathcal{C}^{(r)}, \Theta^{(r)}\right)$.

Induction loop. Each induction round proceeds as follows. First, the LLM generates Structured Chain-of-Thought traces under $\mathcal{C}^{(r)}$ and $V^{(r)}$. Second, extracted operation phrases are semantically consolidated into canonical skills, yielding updated item-skill mappings and forming a new skill vocabulary $V^{(r+1)}$. Third, the induced skill space is evaluated as a whole using downstream student modeling performance as a feasibility signal. Finally, a TextGrad-style discrete controller invokes a frozen LLM to interpret the global feasibility feedback and to explicitly generate updated generation constraints $\mathcal{C}^{(r)}$ and consolidation hyperparameters $\Theta^{(r)}$, without any differentiable optimization.

3.2 Process Observation via Structured Chain-of-Thought

For each item $i \in \mathcal{I}$, we generate a *Structured Chain-of-Thought* (Structured CoT) using a frozen

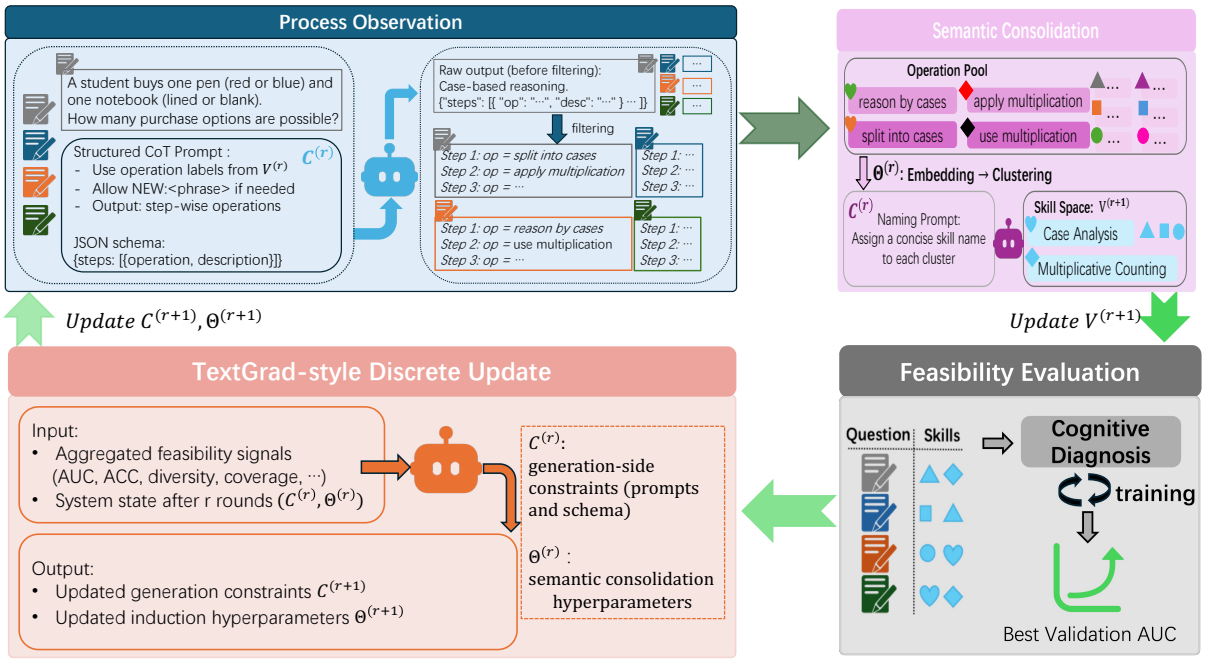

Figure 2: Overview of the proposed feedback-driven closed-loop framework for process-based ability induction.

LLM under generation-side constraints $\mathcal{C}^{(r)}$:

$$\mathbf{c}_i^{(r)} = \mathrm{LLM}\Big(i;\, V^{(r)}, \mathcal{C}^{(r)}\Big)\,, \tag{1}$$

where the initial prompt and JSON schema for structured process observation are shown in Appendix Table 14.

Structured CoT follows a fixed JSON schema that enforces step-wise reasoning with explicit operation annotations. Each step is labeled with an operation selected from the current vocabulary $V^{(r)}$, and a reserved `NEW:`*phrase* prefix is allowed when no suitable label exists. This yields a machine-readable sequence of reasoning steps, enabling a consistent process observation across items.

From each trace, we extract a set of operation labels:

$$\mathcal{O}_i^{(r)} = g\Big(\mathbf{c}_i^{(r)}\Big)\,, \tag{2}$$

where $g(\cdot)$ retains only step-level operation annotations. We treat $\mathcal{O}_i^{(r)}$ as a redundant and non-canonical process observation, rather than a finalized skill representation.

3.3 Semantic Consolidation into Canonical Skills

As illustrated in Figure 2, operation labels extracted from all items are aggregated into a global *operation pool*. Due to linguistic variability (e.g., "reason by cases" vs. "split into cases"), directly reusing these non-canonical phrases would lead to a fragmented skill space.

We perform semantic consolidation through three stages. First, each operation phrase is encoded into a semantic embedding using the pre-trained sentence encoder BGE-M3 (Chen et al., 2024). Second, operation embeddings are grouped via similarity-based clustering, governed by the consolidation hyperparameters

$$\Theta^{(r)} = \{\tau, m, u\}, \tag{3}$$

where τ denotes the similarity threshold, m the minimum cluster size, and u the minimum item-level support required for a cluster to be promoted as a canonical skill. Third, for each stable cluster, we invoke a frozen LLM with a naming prompt to assign a concise and reusable canonical skill name. The initial naming prompt is provided in Appendix Table 15 and is later updated automatically as part of $\mathcal{C}^{(r)}$ through TextGrad-style discrete feedback. This step consolidates linguistically diverse operation phrases into a single canonical skill (e.g., *Case Analysis*).

Based on the resulting canonical skills, each item $i \in \mathcal{I}$ is assigned an induced skill set $\mathcal{S}_i^{(r)}$. The updated skill vocabulary is then constructed by aggregating induced skills across all items:

$$V^{(r+1)} = \bigcup_{i \in \mathcal{I}} \mathcal{S}_i^{(r)}. \tag{4}$$

3.4 Student Modeling-Based Feasibility Evaluation

To assess the dataset-level feasibility of the induced skill space as an ability representation, we introduce a student modeling-based feasibility evaluation, using student response prediction performance to estimate this global feasibility.

Concretely, we instantiate the evaluation using a representative cognitive diagnosis model, KaNCD (Wang et al., 2022), and quantify feasibility via validation AUC. This choice is not tied to KaNCD specifically, but reflects the general principle that a feasible ability representation should enable accurate prediction of student responses under standard student modeling frameworks. Accordingly, we define the global feasibility score at round r as:

$$\text{Score}^{(r)} = \text{AUC}_{\text{val}}. \tag{5}$$

A higher score indicates that the induced skill space, when considered as a whole, better supports modeling student response behavior at the dataset level.

By default, we do not use knowledge tracing performance as a feedback signal during induction, as KT models are sequence-dependent and less sensitive to local item-skill structure. Using CD as the feasibility signal isolates the effect of skill representations from temporal modeling choices. We additionally evaluate a KT-feedback variant in Appendix Table 8.

After completing all induction rounds, we select the round with the highest validation AUC and retain the corresponding skill space $V^{(r+1)}$ together with the item-skill assignments $\mathcal{S}^{(r)}$. The retained item-skill mapping is then fixed and used as the ability representation for downstream cognitive diagnosis and knowledge tracing models.

3.5 TextGrad-Style Discrete Feedback Updates

To incorporate external feasibility feedback without relying on differentiable optimization, we adopt a TextGrad-style (Yuksekgonul et al., 2025) discrete update mechanism driven by a large language model.

Inputs. At iteration r, the update module takes the following as input: (i) feasibility signals aggregated across previous multiple update rounds (three rounds in practice), including validation AUC and auxiliary statistics (e.g., skill coverage and skill diversity), and (ii) the current system configuration $(\mathcal{C}^{(r)}, \Theta^{(r)})$.

Outputs. Given these inputs, the language model produces discrete proposals for: (i) updated generation-side constraints $\mathcal{C}^{(r+1)}$ (e.g., prompt-level instructions and schema rules), and (ii) updated semantic consolidation hyperparameters $\Theta^{(r+1)}$.

Formally, the update step is written as:

$$(\mathcal{C}^{(r+1)}, \Theta^{(r+1)}) = \Phi\left(\mathcal{C}^{(r)}, \Theta^{(r)}, \text{Score}^{(r)}\right), \tag{6}$$

where Φ denotes a language-model-driven discrete update operator. In practice, Φ is implemented by invoking a frozen LLM with TextGrad-style prompts to update generation constraints and induction hyperparameters, rather than performing differentiable optimization (see Appendix Tables 16 and 17).

Table 1: Dataset statistics for the processed splits used in our experiments.

Statistic	XES3G5M	BEPKT
#Students	100	905
#Items	1055	551
#Interactions	5003	25853
#Concepts	285	100

4 Experiments

We evaluate the proposed framework from complementary perspectives, including downstream student-modeling performance, structural properties of the induced skill space, component-level ablations, and expert judgments of representational quality.

4.1 Experimental Setup

Datasets. We evaluate our framework on two educational datasets with expert-defined knowledge concepts: **XES3G5M** (Liu et al., 2023b), an online mathematics learning dataset, and **BEPKT** (Zhu et al., 2022), which contains programming learning trajectories. These datasets differ in domain, scale, and concept granularity; statistics are summarized in Table 1.

Skill representations and baselines. We consider three baseline skill representations: human knowledge concepts, GPT-4o-generated concept labels from problem descriptions, and KCGen, an

Table 2: Downstream performance on cognitive diagnosis and knowledge tracing using different skill representations. "Human Knowledge" denotes expert-provided knowledge concepts.

Model	XES3G5M				BEPKT			
	CD-AUC	CD-ACC	KT-AUC	KT-ACC	CD-AUC	CD-ACC	KT-AUC	KT-ACC
GPT Knowledge	0.6288	0.7360	0.6895	0.7409	0.6659	0.6558	0.6689	0.6335
KCGen (GPT-4o)	0.6432	0.7261	0.6608	0.7296	0.6670	0.6562	0.6703	0.6361
Human Knowledge	0.6502	0.7240	0.6855	0.7490	0.6630	0.6551	0.6779	0.6395
Qwen3-0.6B	0.6768	0.7670	0.7013	0.7521	0.6641	0.6479	0.6617	0.6275
Qwen3-1.7B	0.6877	0.7671	0.6856	0.7382	0.6739	0.6487	0.6598	0.6220
Qwen3-4B	0.7008	0.7600	0.6998	0.7510	0.6723	0.6519	**0.6906**	**0.6462**
Qwen3-8B	0.6838	0.7561	**0.7076**	0.7531	0.6703	0.6602	0.6849	0.6458
Qwen3-14B	0.6883	0.7580	0.7041	**0.7592**	**0.6793**	0.6549	0.6782	0.6406
Qwen3-32B	**0.7010**	0.7622	0.7033	0.7520	0.6773	**0.6623**	0.6866	0.6423
Llama3-3B	0.6545	0.7610	0.6875	0.7452	0.6644	0.6514	0.6769	0.6343
Llama3-8B	0.6818	**0.7690**	0.6943	0.7516	0.6749	0.6591	0.6856	0.6442

automated knowledge-component generation baseline following KCGen-KT (Duan et al., 2025).

We instantiate the method with frozen Qwen3 (Yang et al., 2025) and Llama3 (Dubey et al., 2024) backbones of multiple sizes; implementation details are provided in Appendix Section B.

Evaluation tasks and metrics. We evaluate each item-skill mapping from three complementary perspectives: task-level performance, structural properties, and expert-based qualitative assessment.

Task-level evaluation. We evaluate induced item-skill mappings on two downstream student modeling tasks: (i) Cognitive Diagnosis (CD) using KaNCD, and (ii) Knowledge Tracing (KT) using the classical DKT (Piech et al., 2015) model.

During induction, KaNCD is used as a feasibility evaluator, producing a validation score for each candidate item-skill mapping. This score is used solely to select the mapping that best supports student modeling, rather than to tune or compare different student models. After the induction process is completed, the selected item-skill mapping is fixed. We then evaluate its downstream performance on the held-out test set using KaNCD for cognitive diagnosis and DKT for knowledge tracing, and report CD-AUC, CD-ACC, KT-AUC, and KT-ACC.

Structure-level evaluation. To characterize induced ability spaces beyond predictive performance, we report a set of structural statistics, including: (i) the total number of induced skills (#Skills); (ii) skill granularity measured by the average and 90th-percentile number of skills per item (Avg./Item and P90); (iii) coverage concentration measured by the Gini coefficient over skill frequencies (Gini) and the fraction of assignments covered by the 20 most frequent skills (Top-20). Formal definitions of the structure-level metrics are provided in Appendix Section A.

Expert-based evaluation. As a complementary evaluation, we conduct a questionnaire study with domain experts to qualitatively assess the induced skill representations.

Training protocol and reproducibility. All downstream models are trained using fixed architectures and identical data splits across skill representations, with the same optimization and hyperparameter selection procedure to ensure fair comparison. For reproducibility, the full inference settings for LLM generation, as well as the complete training configurations for KaNCD and DKT, are provided in Appendix Section B.

4.2 Main Results on Cognitive Diagnosis and Knowledge Tracing

Cognitive diagnosis performance. As shown in Table 2, process-induced skills consistently outperform expert-defined knowledge concepts on cognitive diagnosis across both datasets. On XES3G5M, CD-AUC improves from 0.6502 using expert concepts to around 0.70 for several Qwen3 variants. Similar improvements are observed on BEPKT, where multiple medium-to-large models achieve the strongest CD performance. These gains are achieved without increasing the complexity of the model, suggesting that the improvement stems from more informative skill representations rather

Table 3: Skill structure statistics on XES3G5M and BEPKT.

Model	XES3G5M					BEPKT				
	#Skills	Avg./Item	P90	Gini	Top-20	#Skills	Avg./Item	P90	Gini	Top-20
GPT Knowledge	291	1.28	2	0.64	0.48	511	2.8	4	0.55	0.32
KCGen	100	1.7	3	0.712	0.767	100	1.83	3	0.67	0.74
Human Knowledge	285	1.10	1	0.54	0.35	100	1.76	3	0.70	0.78
Qwen3-0.6B	72	1.70	3	0.75	0.86	58	1.63	3	0.80	0.92
Qwen3-1.7B	157	2.60	4	0.79	0.75	86	4.56	6	0.82	0.92
Qwen3-4B	175	2.60	4	0.70	0.60	200	3.82	6	0.66	0.54
Qwen3-8B	213	2.70	4	0.72	0.60	144	4.79	7	0.74	0.72
Qwen3-14B	167	3.00	5	0.70	0.60	182	5.15	7	0.70	0.57
Qwen3-32B	223	3.50	5	0.67	0.50	161	5.59	8	0.70	0.64
Llama3-3B	149	2.30	4	0.69	0.59	130	2.82	4	0.69	0.70
Llama3-8B	164	4.30	7	0.71	0.64	204	4.71	7	0.71	0.59

than model capacity. On XES3G5M, our one-round Qwen3-8B setting improves CD-AUC from 0.6432 to 0.6678 and KT-AUC from 0.6608 to 0.7031 over KCGen, indicating that process-based induction provides benefits beyond automatic concept abstraction. The best CD-AUC reaches 0.7010 on XES3G5M with Qwen3-32B and 0.6793 on BEPKT with Qwen3-14B, both exceeding the human-knowledge and KCGen baselines. This pattern suggests that the induced procedural skills provide a more discriminative item–ability interface for cognitive diagnosis than content-level concept labels.

Knowledge tracing performance. As shown in Table 2, process-induced skills achieve performance that is competitive with or better than the knowledge-based baseline on knowledge tracing across both datasets. On XES3G5M, KT-AUC increases from 0.6855 to over 0.70 with skills induced by several Qwen3 models, while performance on BEPKT remains stable or improves slightly. Notably, despite introducing higher-granularity skills, performance does not degrade on knowledge tracing. Importantly, even though the skills are induced solely under a cognitive diagnosis-based feedback signal, they yield consistent gains on knowledge tracing, demonstrating that the induced representations capture transferable procedural structure beyond the specific optimization objective. We also evaluate GPT-4o-mini as a stronger induction backbone in Appendix Table 10. The results show that stronger models can improve process observation and consolidation quality, while the final skill space still benefits from downstream feasibility validation. For example, Qwen3-8B achieves the best KT-AUC on XES3G5M (0.7076), while Qwen3-4B achieves the best KT-AUC on BEPKT (0.6906). These results indicate that the process-induced skills are not only useful for static diagnosis, but also preserve temporal signals needed by sequential knowledge tracing models.

4.3 Skill Structure Analysis

Skill granularity. As shown in Table 3, human-defined knowledge concepts are relatively coarse-grained, with each item linked to few skills on average. In contrast, process-induced skills are more fine-grained, yielding more unique skills and higher per-item skill counts. For example, on BEPKT, the average number of skills per item increases from 1.76 under expert concepts to over 5 in several induced settings. This suggests that the induced representations decompose problems into reusable procedural components rather than single content-level labels.

Coverage concentration and balance. Table 3 also shows how skill usage is distributed across items using the Gini coefficient and Top-20 coverage. High concentration indicates that a small number of generic skills dominate the assignments. For example, Qwen3-0.6B has very high Top-20 coverage, suggesting over-generalized skills. In contrast, medium-capacity models produce more balanced distributions, with lower concentration and more even skill usage.

4.4 Ablation Study

Role of multi-round feedback. As shown in Table 4, multi-round feedback improves over one-

Table 4: Ablation results on XES3G5M. "Multi rounds" denotes the full closed-loop induction, while "One round" disables iterative feedback updates. BEPKT results are reported in Appendix Table 7.

Model	Setting	XES3G5M				
		CD-AUC	CD-ACC	KT-AUC	KT-ACC	Avg./Item
Qwen3-8B	One round	0.6678	0.7403	0.7031	0.7403	2.5
	Multi rounds	**0.6838**	**0.7561**	**0.7076**	**0.7531**	**2.7**
	w/o KaNCD	0.6725	0.7431	0.6882	0.7392	2.6
	w/o BGE-M3	0.6698	0.7483	0.6917	0.7401	2.6
	w/o TextGrad	0.6794	0.7509	0.6875	0.7382	2.3
Llama3-8B	One round	0.6733	0.7620	0.6890	0.7461	3.5
	Multi rounds	**0.6818**	**0.7690**	**0.6943**	**0.7516**	**4.3**
	w/o KaNCD	0.6744	0.7620	0.6292	0.6959	3.3
	w/o BGE-M3	0.6723	0.7610	0.6827	0.7507	3.4
	w/o TextGrad	0.6704	0.7566	0.5602	0.6649	3.8

round induction across both backbones. For Qwen3-8B on XES3G5M, CD-AUC increases from 0.6678 to 0.6838 and KT-AUC increases from 0.7031 to 0.7076; the same trend holds for Llama3-8B and on BEPKT (Appendix Table 7). For Llama3-8B, multi-round feedback also raises Avg./Item from 3.5 to 4.3, suggesting that iterative refinement improves performance while allowing a richer skill decomposition. Appendix Table 9 further shows that sampling $N = 1$ to $N = 5$ complete reasoning traces per item does not consistently improve KT performance, indicating that the dataset-level feedback loop already captures most reusable process variation.

Role of feasibility evaluation (KaNCD). Removing KaNCD disables feasibility-based feedback during induction. As shown in Table 4, this weakens XES3G5M performance, and Appendix Table 7 shows a similar pattern on BEPKT. The effect is most visible for KT transfer: removing KaNCD reduces KT-AUC from 0.7076 to 0.6882 for Qwen3-8B and from 0.6943 to 0.6292 for Llama3-8B. We further test KT-AUC as an alternative feasibility signal in Appendix Table 8; KT feedback can also guide refinement, while CD feedback remains the default because it provides a stable item-skill selection signal.

Role of semantic consolidation. Removing BGE-M3 disables semantic consolidation of similar operation phrases into canonical skills. As shown in Table 4, this hurts performance; for Qwen3-8B on XES3G5M, CD-AUC drops from 0.6838 to 0.6698. The KT-AUC drop from 0.7076 to 0.6917 further indicates that consolidation helps convert diverse operation phrases into reusable skills that generalize beyond the CD evaluator. These results show the importance of reducing surface variation for skill reuse.

Role of discrete feedback control. Removing TextGrad disables discrete feedback updates. This leads to weaker performance and less stable skill structures. For Llama3-8B, disabling TextGrad causes the largest drop in Table 4, with KT-AUC decreasing from 0.6943 to 0.5602. The controller operates within a predefined structural search space over generation constraints and consolidation parameters (τ, m, u), so it reshapes the induced ability space rather than tuning the downstream student model. Appendix Table 11 shows rapid early improvement followed by a bounded plateau, supporting the stability of this feedback mechanism.

4.5 Expert Evaluation of Ability Representations

Ten education experts evaluated ten questions each across three evaluation aspects, resulting in 300 judgments comparing LLM-generated and expert-defined ability representations. Experts preferred the LLM-generated skills in 82.3% of cases, with 10.7% favoring expert-defined skills and 7% indicating comparable quality. Detailed analyses are provided in Appendix Tables 5 and 6.

5 Conclusion

In this paper, we propose a feedback-driven framework for inducing process-based ability representations from problem-solving traces, with the goal of moving beyond expert-defined knowledge concepts to discover reusable procedural skills that reflect how problems are solved. Specifically, we collect reasoning traces generated by large language models for all items in a dataset and treat them as process-level observations that exhibit substantial surface-level redundancy, where multiple expressions may correspond to the same underlying procedural operation. These traces are progressively refined through structured generation, semantic consolidation, and external feasibility feedback, resulting in stable and behaviorally meaningful dataset-level ability spaces. Experiments on two real-world educational datasets demonstrate consistent improvements in student modeling and reveal informative ability structures in terms of granularity and coverage, suggesting that process-based ability induction offers a practical and scalable alternative to traditional knowledge-based representations.

Limitations

A current limitation of this work is that our empirical evaluation is restricted to mathematics and programming domains, using datasets that primarily reflect learning behavior within a single grade level. While these settings allow us to validate the effectiveness of process-based ability induction in controlled educational scenarios, student abilities often transfer across different knowledge areas and evolve over multiple grade levels. As a result, the extent to which the induced process-based skills generalize across subjects and support longitudinal modeling of student learning remains an open question. In future work, we plan to extend our framework to broader educational contexts by collecting and incorporating datasets spanning multiple domains and grade levels, in order to better assess its applicability to long-term and cross-curricular student modeling.

Ethical Considerations

This work studies automatic induction of process-based ability representations from educational problem-solving data. All experiments are conducted on publicly available datasets that have been anonymized and do not contain personally identifiable information. We do not attempt to infer sensitive attributes or re-identify individual students, and our analysis is performed at the aggregate level. The induced ability representations are intended for research purposes, such as analyzing problem-solving processes and supporting student modeling, rather than for high-stakes educational decision making. As with other data-driven educational models, the induced skills may reflect biases present in the underlying datasets, and their interpretations should therefore be treated with caution, particularly when applied to new populations or contexts. Large language models are used only to generate intermediate reasoning traces from problem descriptions. These traces are treated as process-level observations and are not assumed to be factually correct or pedagogically optimal. The language models are not used for grading, decision making, or generating the manuscript text itself. Finally, while process-based ability induction has the potential to support more fine-grained analysis of learning behavior, we emphasize that any real-world deployment should involve appropriate human oversight and consideration of educational, social, and ethical implications.

References

Ghodai Abdelrahman, Qing Wang, and Bernardo Nunes. 2023. Knowledge tracing: A survey. *ACM Computing Surveys*, 55(11):1–37.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Jiajun Cui, Zeyuan Chen, Aimin Zhou, Jianyong Wang, and Wei Zhang. 2023. Fine-grained interaction modeling with multi-relational transformer for knowledge tracing. *ACM Transactions on Information Systems*, 41(4):1–26.

Zhiang Dong, Jingyuan Chen, and Fei Wu. 2025. Knowledge is power: Harnessing large language models for enhanced cognitive diagnosis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 164–172.

Zhangqi Duan, Nigel Fernandez, Arun Balajiee Lekshmi Narayanan, Mohammad Hassany, Rafaella Sampaio de Alencar, Peter Brusilovsky, Bita Akram, and Andrew Lan. 2025. Automated knowledge component generation for interpretable knowledge tracing in coding problems. *arXiv preprint arXiv:2502.18632*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Lina Gao, Zhongying Zhao, Chao Li, Jianli Zhao, and Qingtian Zeng. 2022. Deep cognitive diagnosis model for predicting students' performance. *Future Generation Computer Systems*, 126:252–262.

Xiaoyu Li, Shaoyang Guo, Jin Wu, and Chanjin Zheng. 2025. An interpretable polytomous cognitive diagnosis framework for predicting examinee performance. *Information Processing & Management*, 62(1):103913.

Yuanyuan Liang, Jianing Wang, Hanlun Zhu, Lei Wang, Weining Qian, and Yunshi Lan. 2023. Prompting large language models with chain-of-thought for few-shot knowledge base question generation. *arXiv preprint arXiv:2310.08395*.

Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Hui Xiong, Yu Su, and Guoping Hu. 2019. Ekt: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering*, 33(1):100–115.

Yingjie Liu, Tiancheng Zhang, Xuecen Wang, Ge Yu, and Tao Li. 2023a. New development of cognitive diagnosis models. *Frontiers of Computer Science*, 17(1):171604.

Zitao Liu, Qiongqiong Liu, Teng Guo, Jiahao Chen, Shuyan Huang, Xiangyu Zhao, Jiliang Tang, Weiqi Luo, and Jian Weng. 2023b. Xes3g5m: A knowledge tracing benchmark dataset with auxiliary information. *Advances in Neural Information Processing Systems*, 36:32958–32970.

Sein Minn, Yi Yu, Michel C Desmarais, Feida Zhu, and Jill-Jenn Vie. 2018. Deep knowledge tracing and dynamic student classification for knowledge tracing. In *2018 IEEE International conference on data mining (ICDM)*, pages 1182–1187. IEEE.

Shalini Pandey and George Karypis. 2019. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. *Advances in neural information processing systems*, 28.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, and Niki Van Stein. 2025. Multi-step reasoning with large language models, a survey. *ACM Computing Surveys*.

Alexander Scarlatos, Ryan S Baker, and Andrew Lan. 2025. Exploring knowledge tracing in tutor-student dialogues using llms. In *Proceedings of the 15th International Learning Analytics and Knowledge Conference*, pages 249–259.

Xiangyu Song, Jianxin Li, Taotao Cai, Shuiqiao Yang, Tingting Yang, and Chengfei Liu. 2022. A survey on deep learning based knowledge tracing. *Knowledge-Based Systems*, 258:110036.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965.

Martin Tutek, Fateme Hashemi Chaleshtori, Ana Marasović, and Yonatan Belinkov. 2025. Measuring chain of thought faithfulness by unlearning reasoning steps. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9946–9971.

Fei Wang, Qi Liu, Enhong Chen, Zhenya Huang, Yuying Chen, Yu Yin, Zai Huang, and Shijin Wang. 2020. Neural cognitive diagnosis for intelligent education systems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6153–6161.

Fei Wang, Qi Liu, Enhong Chen, Zhenya Huang, Yu Yin, Shijin Wang, and Yu Su. 2022. Neuralcd: a general framework for cognitive diagnosis. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):8312–8327.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616.

Renyu Zhu, Dongxiang Zhang, Chengcheng Han, Ming Gaol, Xuesong Lu, Weining Qian, and Aoying Zhou. 2022. Programming knowledge tracing: A comprehensive dataset and a new model. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 298–307. IEEE.

A Additional Details of Structure-Level Metrics

This appendix provides formal definitions for the structure-level metrics reported in Table 3. Let $\mathcal{I}$ denote the item set with $|\mathcal{I}| = N$. For each item $i \in \mathcal{I}$, let $\mathcal{S}_i$ be its induced skill set. Let V be the induced skill vocabulary and $|V| = K$.

(i) Number of unique skills. We report the vocabulary size:

$$\#\text{Skills} = |V|. \tag{7}$$

(ii) Granularity (Avg./Item and P90). Define the per-item skill count:

$$c_i = |\mathcal{S}_i|. \tag{8}$$

Then the average number of skills per item is:

$$\text{Avg./Item} = \frac{1}{N}\sum_{i\in\mathcal{I}} c_i. \tag{9}$$

Let $\{c_i\}_{i\in\mathcal{I}}$ sorted in non-decreasing order be $c_{(1)} \le \cdots \le c_{(N)}$. The 90th percentile is:

$$\text{P90} = c_{(\lceil 0.9N \rceil)}. \tag{10}$$

(iii) Coverage concentration (Gini and Top-20 share). We define skill assignment frequency over the entire item set. For each skill $s \in V$, let

$$f_s = \sum_{i\in\mathcal{I}} \mathbf{1}\{s \in \mathcal{S}_i\}, \tag{11}$$

and let the total number of item-skill assignments be

$$F = \sum_{s\in V} f_s = \sum_{i\in\mathcal{I}} |\mathcal{S}_i|. \tag{12}$$

Gini coefficient. Let $\{f_s\}_{s\in V}$ sorted be $f_{(1)} \le \cdots \le f_{(K)}$, and let $\mu = \frac{1}{K}\sum_{k=1}^{K} f_{(k)} = \frac{F}{K}$. We compute the Gini coefficient over skill frequencies as:

$$\text{Gini} = \frac{\sum_{k=1}^{K}\sum_{\ell=1}^{K} |f_{(k)} - f_{(\ell)}|}{2K^2\mu}. \tag{13}$$

A larger Gini indicates that a small number of skills dominate the assignments.

Top-20 share. Let $f^{\downarrow}_{(1)} \ge \cdots \ge f^{\downarrow}_{(K)}$ denote frequencies sorted in descending order. We define:

$$\text{Top-20} = \frac{\sum_{k=1}^{\min(20,K)} f^{\downarrow}_{(k)}}{F}. \tag{14}$$

A larger Top-20 implies more concentrated coverage on a small set of frequent skills.

B Implementation Details and Experimental Configurations

LLM backbones and decoding settings. We instantiate the induction framework with instruction-tuned models from the Qwen3 and Llama3 families, covering a wide range of model scales. For Qwen3, we use models ranging from 0.6B to 32B parameters with a decoding temperature of 0.7, while for Llama3 we use 3B and 8B instruction-tuned variants with a decoding temperature of 0.3.

Datasets and splits. For each dataset, we split the interaction logs into training/validation/test sets with a ratio of 7:1:2.

Downstream student models. We select KaNCD as the representative cognitive diagnosis model and DKT as the representative knowledge tracing model. For KaNCD, we perform a grid search over the learning rate and embedding dimension, with the learning rate selected from 0.001, 0.002, 0.01, 0.02 and the embedding dimension from 8, 16, 32, 64, 128, 256, 512. For DKT, we conduct a grid search over the dropout rate and learning rate, where dropout is chosen from 0.0, 0.1, 0.2, 0.3 and the learning rate from 1e-4, 3e-4, 1e-3, 3e-3. All experiments are repeated with five different random seeds, and the reported results are averaged across runs.

Computational resources. All experiments are conducted on a server equipped with 30 CPU cores, 400GB system memory, and two NVIDIA H100 GPUs (80GB each). The Qwen3 and Llama3 family models are directly deployed on the server for inference during the induction process. The downstream student models, including KaNCD and DKT, are lightweight and run with minimal GPU usage. For ChatGPT-4o, we access the model via the official API.

C Expert Evaluation of Ability Representations

C.1 Evaluation Protocol

Ten education experts participated in the evaluation. Each expert independently evaluated ten questions.

For each question, experts were presented with the problem text, a set of process-based skills induced from structured LLM reasoning traces, and a set of expert-defined knowledge-based abilities derived from curriculum-aligned routes. Experts

were asked to provide judgments along four dimensions:

- **Q1 (Representational Adequacy):** Which ability representation more appropriately captures the key abilities required to solve the question?
- **Q2 (Process Fidelity):** Which representation better reflects learners' actual problem-solving processes, rather than merely the involved content?
- **Q3 (Diagnostic Utility):** Which representation would be more informative for analyzing learner errors or ability differences?
- **Q4 (Issue Identification):** Does the LLM-generated ability representation exhibit any noticeable issues?

No predefined gold standard was assumed; all evaluations were based on expert judgment.

C.2 Quantitative Summary of Expert Judgments

For the preference-based questions (Q1–Q3), expert responses were aggregated as *LLM* (preference for LLM-generated abilities), *EXP* (preference for expert-defined abilities), or *SAME* (comparable quality or no clear preference).

In total, we collected 300 expert judgments (10 experts $\times$ 10 questions $\times$ 3 evaluation aspects). Table 5 summarizes the aggregated results.

Overall, experts preferred the LLM-generated process-based abilities in 82.3% of the judgments, compared to 10.7% favoring expert-defined representations, with 7.0% indicating comparable quality. This preference is consistent across representational adequacy, process fidelity, and diagnostic utility, and is strongest for diagnostic utility.

Question	LLM	EXP	SAME	Total
Q1 (Adequacy)	78	13	9	100
Q2 (Process)	83	10	7	100
Q3 (Diagnostic)	86	9	5	100
All (Q1–Q3)	247	32	21	300

Table 5: Expert preferences between LLM-generated process-based skills and expert-defined knowledge-based representations across three evaluation aspects (Q1–Q3).

C.3 Issue Analysis of LLM-Generated skills

Experts further assessed whether the LLM-generated skills exhibited noticeable issues (Q4). The distribution of identified issue types is reported in Table 6.

As shown in Table 6, in 70% of the evaluations, experts judged the LLM-generated skills to be reasonable and free of obvious issues. When issues were reported, they mainly concerned granularity mismatches, including skills perceived as overly coarse (9%) or overly fine-grained (9%). Other issues, such as the presence of irrelevant skills (8%) or semantic redundancy (4%), were relatively infrequent. Severe semantic errors were rarely observed, suggesting that most induced skills are meaningful and usable in practice.

Issue Type	Count
Reasonable	70
Too coarse	9
Too fine	9
Irrelevant	8
Redundant	4

Table 6: Distribution of expert-identified issue types for LLM-generated skills (Q4).

D Additional Rebuttal Experiments

This appendix reports the additional analyses introduced during the rebuttal period. These experiments address reviewer questions about comparison with recent knowledge-component generation work, alternative feasibility signals, multi-path reasoning, stronger LLM backbones, and controller stability.

D.1 Additional Ablation Results on BEPKT

Table 7: BEPKT ablation results.

Model	Setting	BEPKT				
		CD-AUC	CD-ACC	KT-AUC	KT-ACC	Avg./Item
Qwen3-8B	One round	0.6632	0.6581	0.6811	0.6387	3.1
	Multi rounds	**0.6703**	**0.6602**	**0.6849**	**0.6458**	**4.79**
	w/o KaNCD	0.6667	0.6541	0.6814	0.6425	3.6
	w/o BGE-M3	0.6690	0.6409	0.6813	0.6398	3.3
	w/o TextGrad	0.6641	0.6498	0.6809	0.6440	3.9
Llama3-8B	One round	0.6677	0.6495	0.6743	0.6293	3.4
	Multi rounds	**0.6749**	**0.6591**	**0.6856**	**0.6382**	**4.71**
	w/o KaNCD	0.6736	0.6453	0.6786	0.6332	3.6
	w/o BGE-M3	0.6722	0.6445	0.6753	0.6316	3.5
	w/o TextGrad	0.6748	0.6492	0.6747	0.6259	4.0

D.2 KT Feedback as an Alternative Feasibility Signal

Table 8 replaces validation CD-AUC with validation KT-AUC as the controller's global feasibility signal. These results indicate that KT feedback can also guide process-based ability induction, although we use CD feedback as the default because it is less sensitive to sequence-model variance. For non-feedback baselines, the table reports their downstream evaluation results under the same metrics for reference.

Table 8: Downstream performance when validation KT-AUC is used as the feasibility signal during induction. Non-feedback baselines, including KCGen (GPT-4o), are reported for reference under the same downstream metrics.

Model	XES3G5M				BEPKT			
	CD-AUC	CD-ACC	KT-AUC	KT-ACC	CD-AUC	CD-ACC	KT-AUC	KT-ACC
GPT Knowledge	0.6288	0.7360	0.6895	0.7409	0.6659	0.6558	0.6689	0.6335
KCGen (GPT-4o)	0.6544	0.7431	0.6842	0.7431	0.6672	0.6510	0.6802	0.6384
Human Knowledge	0.6502	0.7240	0.6588	0.7490	0.6630	0.6551	0.6779	0.6395
Qwen3-0.6B	0.6721	0.7637	0.7001	0.7515	0.6683	0.6491	0.6845	0.6402
Qwen3-1.7B	0.6738	0.7641	0.6915	0.7464	0.6789	0.6553	0.6961	0.6453
Qwen3-4B	0.6839	0.7673	0.7098	0.7538	0.6733	0.6587	0.6895	0.6433
Qwen3-8B	0.7014	0.7633	0.6993	0.7532	0.6701	0.6597	0.6883	0.6421
Qwen3-14B	0.6878	0.7572	0.6976	0.7497	0.6801	0.6621	0.6838	0.6409
Qwen3-32B	0.6993	0.7611	0.7013	0.7511	0.6786	0.6591	0.6852	0.6415
Llama3-3B	0.6553	0.7569	0.6813	0.7429	0.6631	0.6503	0.6881	0.6438
Llama3-8B	0.6998	0.7683	0.7123	0.7612	0.6771	0.6578	0.6902	0.6446

D.3 Multi-Path Reasoning Samples

Table 9 studies whether sampling multiple complete reasoning traces for each item improves the final skill space. Increasing the number of paths from $N = 1$ to $N = 5$ does not yield consistent improvements, suggesting that the multi-round dataset-level induction loop already captures most reusable process signals.

Table 9: KT performance when sampling N complete reasoning traces per item before consolidation.

Model	Metric	XES3G5M					BEPKT				
		N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5
Qwen3-8B	KT-AUC	0.7076	0.7037	0.6998	0.6933	0.6941	0.6849	0.6830	0.6843	0.6839	0.6821
	KT-ACC	0.7531	0.7516	0.7493	0.7486	0.7500	0.6458	0.6388	0.6456	0.6373	0.6451
Llama3-8B	KT-AUC	0.6943	0.6851	0.6839	0.6841	0.6902	0.6856	0.6819	0.6829	0.6843	0.6730
	KT-ACC	0.7516	0.7488	0.7501	0.7416	0.7493	0.6442	0.6367	0.6391	0.6412	0.6367

D.4 Stronger LLM Backbone

Table 10 reports GPT-4o-mini as an induction backbone. Stronger language models improve process observation and semantic consolidation, but the results still pass through the same feasibility-guided induction framework.

Table 10: Performance of GPT-4o-mini as the base model for process-based ability induction.

Model	XES3G5M				BEPKT			
	CD-AUC	CD-ACC	KT-AUC	KT-ACC	CD-AUC	CD-ACC	KT-AUC	KT-ACC
GPT Knowledge	0.6288	0.7360	0.6895	0.7409	0.6659	0.6558	0.6689	0.6335
Human Knowledge	0.6502	0.7240	0.6588	0.7490	0.6630	0.6551	0.6779	0.6395
Qwen3-Best	0.7010	0.7671	0.7076	0.7592	0.6793	0.6623	0.6906	0.6462
Llama3-Best	0.6818	0.7690	0.6943	0.7516	0.6749	0.6591	0.6856	0.6442
GPT-4o-mini	0.7092	0.7726	0.7151	0.7650	0.6785	0.6631	0.7021	0.6517

D.5 Controller Stability

Table 11 summarizes representative round-level dynamics. The controller improves rapidly in early rounds and then remains in a bounded plateau, providing empirical evidence against uncontrolled oscillation or collapse into degenerate skill spaces.

Table 11: Representative controller dynamics over induction rounds.

Metric	Value
Initial AUC (r1)	0.6595
Max AUC (first 10 rounds)	0.6788
Global max AUC	0.6836
Mean AUC (last 50 rounds)	0.6769
AUC range	[0.6653, 0.6836]
Mean AUC (rounds 1–10)	0.6720
Mean AUC (rounds 41–50)	0.6767
Mean AUC (rounds 51–60)	0.6773
Mean AUC (rounds 91–100)	0.6766

E Case Study: Process-Induced skills vs. Human Concepts

We present two illustrative examples to qualitatively compare process-induced skills with expert-defined knowledge concepts.

	Example 1: Outfit Combination	Example 2: Factoring with a Common Factor
Problem	Xiaomei bought two tops and two skirts at a supermarket. How many different outfits can she make if she wears exactly one top and one skirt?	Compute: $178 \times 15 + 15 \times 22$.
Process observations	`reason_by_cases`: decompose the selection into choosing a top and choosing a skirt. `enumeration`: count the available tops and skirts independently. `multiplication_rule`: combine independent choices by multiplying their counts.	`pattern_detection`: identify a shared multiplicative factor across terms. `factor_extraction`: rewrite the expression by factoring out the common term. `addition`: simplify the expression inside the parentheses. `multiplication`: compute the final result after simplification.
Induced skills	Case-based decomposition; enumeration of possibilities; multiplication-based composition.	Detection of shared factors; algebraic transformation via factor extraction; basic arithmetic operations.
Expert concept	Item matching / pairing.	Integer construction and extraction.

Table 12: Representative examples comparing process-induced skills with expert-defined knowledge concepts.

F Prompt Templates for Process Observation and Feedback

To support reproducibility and clarify the interfaces used in our framework, we provide representative prompt templates for the GPT Knowledge baseline, process observation, and feedback-driven updates. These prompts define structured, discrete interaction protocols with the language model, rather than trainable components. Throughout the induction process, all language model parameters remain fixed, and the prompts serve as the primary mechanism for controlling generation, abstraction, and feedback incorporation.

Table 13 gives the prompt used to construct the GPT Knowledge baseline. Specifically, Table 14 shows the Structured Chain-of-Thought prompt used for process observation, which elicits step-wise reasoning with explicit operation annotations. This prompt may be further refined through feedback-driven updates (see Table 17), while its core output interface remains unchanged. Table 15 presents the prompt for symbolic naming of consolidated operation clusters, which affects only the canonical identifiers of induced skills and does not influence clustering or promotion decisions; this naming prompt may likewise undergo minor refinements during feedback-driven updates. Table 16 illustrates a representative TextGrad-style controller prompt used to update semantic consolidation hyperparameters based on global feasibility signals. In later update rounds, this controller may lead to refinements of generation-side constraints, such as modifying the Structured Chain-of-Thought prompt in Table 14 to remove rules that allow the introduction of new operation labels (e.g., `NEW:<short phrase>`), thereby encouraging stricter reuse of an existing operation vocabulary. Finally, Table 17 shows the prompt used to refine reasoning analysis and semantic consolidation templates in a feedback-driven manner. Through this mechanism, semantic consolidation hyperparameters may be adjusted across iterations; for example, the similarity threshold τ in $\Theta^{(r+1)}$ can change from an initial value of 0.72 to 0.68 in later stages to promote more stable consolidation under downstream feasibility feedback.

Together, these prompts specify the complete discrete control surface of the proposed framework, enabling reproduction of the induction process without modifying or fine-tuning the underlying language models and without manual rewriting of the prompt templates.

F.1 GPT Knowledge Baseline Prompt

For the GPT Knowledge baseline, GPT-4o is prompted as a curriculum designer to generate knowledge components from the problem statement. The output is constrained to a JSON object containing a list of knowledge concepts.

System role. You are an expert curriculum designer and educational assessment specialist.

Task. Identify the underlying knowledge components required to solve the given problem.

Input. `Problem: {problem_text}`

Output format.

```
{
  "knowledge_concepts": [
    {
      "name": "...",
      "description": "..."
    }
  ]
}
```

Table 13: Prompt used to construct the GPT Knowledge baseline.

System role. You are an expert problem-solving analyst.

Task. Given a question, generate a structured reasoning trace in JSON format. The goal is to explicitly expose the problem-solving process using reusable operations.

Output format requirements.

- Output must be valid JSON.
- The JSON object must contain:
  - `"steps"`: a list of reasoning steps
  - `"reasoning_types"`: a list of operation labels used
  - `"steps_count"`: the total number of steps

Step schema. Each element in `"steps"` must include:

- `"id"`: step index (starting from 1)
- `"operation"`: operation used in this step
- `"description"`: brief description of the step
- `"evidence"`: supporting justification or intermediate result (optional)

Operation constraints.

- `"operation"` must be selected from a predefined vocabulary.
- If no suitable label exists, a new label may be introduced as `NEW:<short phrase>`.
- New labels should be avoided unless necessary; reuse is preferred.

Additional constraints.

- Reasoning should be correct, concise, and logically coherent.
- The output must contain JSON only, with no extra text.

Instruction. Now solve the given question accordingly.

Table 14: Structured Chain-of-Thought prompt used for process observation.

Role. You are given a set of operation phrases that share similar procedural meaning.

Task. Assign a concise canonical ability name that captures the shared procedure.

Input.

- A list of operation phrases representing a semantic cluster

Requirements.

- Provide a short, general name suitable as a reusable ability identifier.
- Do not consider frequency, clustering decisions, or downstream performance.

Output format. The output must be a JSON object with the following fields:

- `"name"`: canonical ability name
- `"description"`: brief explanation of the ability

Table 15: Prompt used for symbolic naming of induced skills. This step affects only identifiers and does not influence clustering or promotion decisions.

Role. You are a controller responsible for adjusting discrete hyperparameters in the ability induction process based on external evaluation feedback.

Inputs (aggregated from recent rounds).

- Validation performance history (AUC, ACC)
- Induced ability count history
- Coverage statistics history
- Diversity statistics history

Task. Analyze the current state of the induced ability space and determine whether it is:

- over-fragmented (too many overly fine-grained skills),
- over-concentrated (too few overly general skills), or
- reasonably balanced.

Based on this diagnosis, propose updated values for the predefined hyperparameters:

- `tau`: clustering similarity threshold
- `m`: minimum cluster size
- `u`: minimum item-level support

Constraints.

- Output only a JSON object with numeric values for `tau`, `m`, and `u`.
- All values must remain within predefined bounds.
- Do not introduce new parameters or modify the schema.

Output format.

- `"tau"`: updated similarity threshold
- `"m"`: updated minimum cluster size
- `"u"`: updated minimum item-level support

Table 16: A representative TextGrad-style controller prompt for discrete hyperparameter updates based on global feasibility signals.

Role. You are a controller responsible for refining prompt templates used in the ability induction pipeline based on evaluation feedback.

Inputs.

- Current analyst prompt (system and user prefix)
- Current taxonomist prompt for cluster naming
- Aggregated evaluation feedback from recent rounds

Task. Propose small, local refinements to the existing prompt templates to improve reasoning analysis and semantic consolidation. If no modification is necessary, return the original templates unchanged.

Constraints.

- Output only a JSON object.
- Preserve the original prompt structure.
- Do not introduce new prompt components.

Output format. The output JSON must contain the following fields:

- `"analyst.system"`: updated system prompt
- `"taxonomist.name_cluster_prompt"`: updated cluster naming prompt

Table 17: A representative TextGrad-style controller prompt for feedback-driven refinement of reasoning analysis and semantic consolidation prompts.