

From Mastery Profile to Simulated Response: Stochastic Student Knowledge Graphs (SSKG) for Faithful LLM Student Simulation

Yuan An, Emily Wang[§], Benjamin Wang[§], Ruhma Hashmi

School of Computer and Information Sciences, Nick Howley College of Engineering and Computing
Drexel University

Philadelphia, PA 19104, USA

{ya45, rh927}@drexel.edu

Abstract—Large language models (LLMs) are increasingly used to simulate students at different mastery levels. These simulations can generate synthetic training data and stress-test tutoring systems. However, common prompt-based approaches leave the answer decision to the LLM, which tends to perform according to its built-in capabilities even when instructed to simulate a student with low mastery. As a result, these approaches may have difficulty distinguishing students with low and high levels of mastery. We demonstrate this limitation using 379 College Board-calibrated SAT Algebra items and five archetypal mastery profiles. Three LLMs from three vendors (Gemini 3.1 Flash Lite, Claude Haiku 4.5, and GPT-5.4-mini) achieve 96.8-100% accuracy across all profiles. To address this limitation, we introduce a method grounded in a Stochastic Student Knowledge Graph (SSKG). A curriculum knowledge graph (CKG) is extracted from an open algebra textbook, and each SAT solution is decomposed into a chain of required triples. The SSKG assigns a mastery probability to each triple, which is sampled to determine question correctness. An LLM then generates a first-person rationale consistent with the outcome. The simulation reduces accuracy to 44.1-85.2% across profiles and produces a clear monotone mastery gradient.

Index Terms—knowledge graphs, student simulation, large language models, knowledge tracing, synthetic educational data, stochastic ontology

I. INTRODUCTION

Synthetic student response data is valuable when real classroom data is expensive, slow, or difficult to collect at scale due to ethical or practical constraints. Applications include training knowledge-tracing models before a course has run [1], stress-testing an intelligent tutoring system across a spread of mastery levels [2], or piloting new assessment items before they reach real students [3]. LLMs are an attractive engine for generating such data because they can answer a wide range of items and explain their reasoning in natural language [4]. The open question is *control*: given an LLM that can solve a curriculum’s questions essentially perfectly [5], how does one make it behave like a student who cannot?

Example. Given a set of N SAT Algebra questions, we would like to simulate three types of students who have mastered the algebra knowledge required by these questions at a high, a

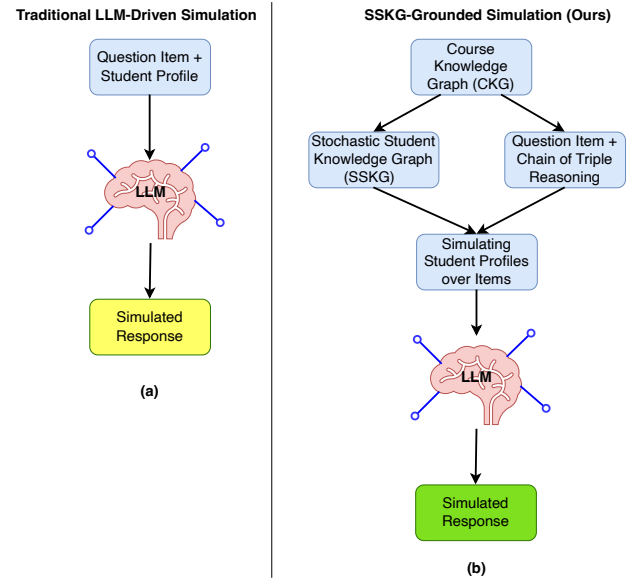


Fig. 1. Traditional LLM-driven student simulation (a) versus SSKG-grounded simulation (b). In the traditional pipeline (a), a question item and a student profile are handed directly to an LLM, which decides both the simulated answer and its correctness. In our pipeline (b), a Curriculum Knowledge Graph (CKG) is built as a set of fine-grained triples extracted from the source curriculum. A student profile is represented as a Stochastic Student Knowledge Graph (SSKG), which assigns a mastery probability to each CKG triple at a time point. Each question is mapped to the chain of triples required to solve it. We simulate a student profile over an item by sampling over the item’s chain against the SSKG’s mastery probabilities to decide whether the response is correct. The resulting item paired with this correctness label is then passed to the LLM, which narrates a first-person rationale consistent with the outcome.

medium, and a low level. A faithful simulation should then answer a high percentage of the N questions correctly for the first student, a medium percentage for the second, and a low percentage for the third. Moreover, if a student has a gap in a specific part of the curriculum, their simulated errors should concentrate on questions that require that knowledge. Each incorrect answer should also be traceable to the specific algebra concept or skill the student failed to apply. □

[§]Drexel summer interns.

A common LLM-based approach in the literature and in practice is to describe the desired student in the prompt, for example, by specifying a skill level [6], an exam grade [7], or an archetypal student profile [8], and let the LLM decide both the answer and the explanation (Figure 1 (a)). We show this approach fails on a concrete, realistic benchmark. Using 379 SAT Algebra multiple-choice items [9] and five archetypal profiles ranging from near-expert to severely gapped, three LLMs spanning three vendors answer at 96.8-100% accuracy *regardless of profile*, including under a profile explicitly labeled “struggling student who has low mastery of most algebra concepts.” Two diagnostic profiles designed to carry structured, chapter-localized gaps are statistically indistinguishable from the high-mastery profile for every LLM tested. This is not a new observation in isolation, it replicates and extends a competency-bias finding first reported for GPT-3.5 on non-math domains [6]. Here, we confirmed at larger scale, on math reasoning items, and across different LLMs.

To address the problem, we present a simulation method that is grounded in the concept of Stochastic Student Knowledge Graph (SSKG) (see Figure 1 (b)). After constructing a curriculum knowledge graph (CKG) containing a set of fine-grained triples, we represent a student profile as an SSKG which assigns a mastery probability to each CKG triple at a time point. Furthermore, we characterize a question’s solution as a chain of triples identified from the CKG. To simulate, we check the required chain of triples of a question against the profile’s mastery vector by explicit sampling. This analytic step decides both whether the simulated student answers correctly and, on failure, which specific knowledge unit broke first. Finally, an LLM is invoked to render a first-person rationale consistent with the already-decided outcome.

We design this SSKG-based method into four cumulative ablation arms (a-d) so that any gain can be attributed to a specific mechanism. *arm-a* is a single-draw sampler. One Bernoulli draw is made per required triple, so every triple in an item’s chain is exactly as consequential as every other. *arm-b* adds a retrieval/execution (knowledge-tracing) decomposition, separating whether a fact is available to the student at all (retrieval) from whether it is applied without error (execution). *arm-c* adds taxonomy-conditioned item weighting so that failing a purely definitional fact (e.g., forgetting what a variable is) is not as consequential as failing a procedural or relational one (e.g., misapplying a multi-step method or picking the wrong solution approach). *arm-d* adds distractor routing. Rather than selecting uniformly a wrong choice, a failure at a specific triple selects the specific wrong answer choice that failure would produce using an item’s distractor map.

We make the following contributions.

- 1) We describe a human-guided extraction method to extract and construct a curriculum knowledge graph (CKG) from given sources.
- 2) We develop an SSKG-grounded method to simulate students with stratified profiles. The method makes profile-accuracy monotonicity and skill-specific error patterns.

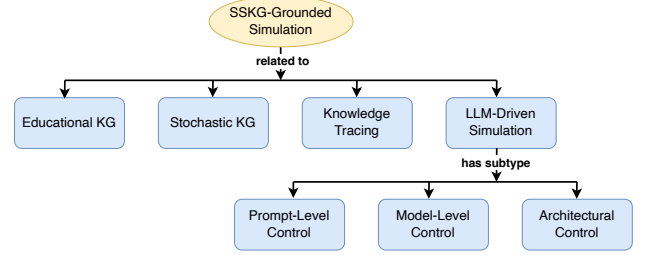


Fig. 2. Related Work Hierarchy

- 3) We present both the theoretical foundation and practical implementation of the simulation method.
- 4) We conduct a four-arm cumulative ablation study that attributes the method’s gain to specific mechanism.

The rest of the paper is organized as follows. Section II discusses the related work. Section III introduces a running illustrative SAT example. Section IV formalizes several key concepts and the problem. Section V describes how the CKG was extracted. Section VI describes how each item’s chain of required triples was identified. Section VII discusses how the distractor map is constructed. Section VIII describes how the five student profile SSKG are represented. Section IX presents the SSKG simulation method’s theoretical foundations and its four cumulative ablation arms. Section X reports the full experimental evaluation. Sections XI–XIII discuss the findings, their limitations, and future work.

II. RELATED WORK

In this section, we describe the related work as characterized in Figure 2.

A. Educational Knowledge Graphs

MOOCCube [10] builds a large-scale knowledge graph of course concepts, videos, and exercises from over 700 real MOOCs specifically to support downstream NLP and learning-analytics tasks. Pan et al. [11] catalog LLM-for-KG-construction patterns at length. Because an LLM-proposed triple is not automatically a correct one, a human verification is needed. Tsaneva et al. [12] show that standalone LLM validation is weak, while hybrid human–LLM validation provides the best overall results.

B. Stochastic and Probabilistic Knowledge Graphs

Attaching a probability to a fact rather than treating it as universally true has been studied at several levels of representation. Probabilistic ontology languages such as PR-OWL [13] extend OWL (Web Ontology Language) with Bayesian semantics so that class membership and relations can be asserted with a degree of belief rather than as certain axioms. Probabilistic knowledge graphs built by open information extraction, such as NELL [14], attach a confidence score to each automatically-extracted belief as it is read from text and promote only beliefs whose confidence clears a threshold. Uncertain knowledge graph embedding [15] attaches a confidence score to each

triple and learns embeddings that predict this confidence for unseen triples. At a general level, a stochastic process [16] is a collection of random variables indexed by a parameter. Here, the parameter is the step in the chain being resolved, and our per-triple sampling provides a concrete instance of this process.

C. Knowledge Tracing and Cognitive Diagnosis

Modeling what a student does and does not know is a decades-old problem independent of LLMs, originally motivated by intelligent tutoring systems (ITS) that adapt instruction to an inferred mastery state. VanLehn [17] ran a meta-analysis of tutoring studies and found that step-based ITS tutoring already approaches the effectiveness of human tutors. Two families of models dominate how that mastery state is inferred. Corbett and Anderson [18] introduced Bayesian Knowledge Tracing (BKT). Piech et al. [19] instead introduced Deep Knowledge Tracing (DKT), replacing BKT’s per-skill Bayesian update with a recurrent network trained end-to-end on interaction sequences. On the cognitive-diagnosis side, Junker and Sijtsma [20] formalized the Deterministic-Input, Noisy-“And”-gate (DINA) model over a binary attribute (here, triple) mastery vector. De la Torre [21] generalized DINA into G-DINA (Generalized DINA). Underlying both families is classical Item Response Theory (IRT) [22], which models a student’s probability of a correct response as a function of a latent ability parameter and item-level difficulty/discrimination parameters. Harwell et al. [23] showed stochastically sampling a simulated examinee’s response to a test item. Their Monte Carlo IRT studies draw a latent ability for each of many simulated examinees and stochastically generate each one’s response to every item from the fitted item parameters.

D. LLM-Driven Student Simulation

Prior LLM student simulation methods control the simulated mastery level at one of following loci: the prompt, the model’s weights or logits, or an external symbolic architecture.

Prompt-level control. Benedetto et al. [6] prompt GPT-3.5 with a numeric skill level $\ell \in \{1, \dots, 5\}$ and ask it to pick the answer choice that student would give. They report that the unconstrained model and their highest simulated level differ by only 0.92 versus 0.90 accuracy, i.e., prompting compresses behavior toward the ceiling, and that the prompt is fragile to inconsequential rewording. Lu and Wang [8] instead describe the profile as a list of knowledge components (KCs) the student has mastered, is confused about, or has no evidence on. They ask an LLM to *predict* a student’s answer rather than role-play the student directly. Evaluated against 100 real students on 20 items, the frame showed profile-aligned predictions reach 85.2% accuracy on mastered KCs versus 11.0% on confused ones. Acquaye et al. [24] estimate item difficulty by aggregating LLM-simulated response accuracy across many simulated test-takers and comparing the result to IRT-calibrated ground truth. Srivatsa et al. [7] provide independent, real-world evidence that LLM-based student simulations fail to preserve differences in student mastery levels.

Prompting LLMs with a National Assessment of Educational Progress (NAEP) grade level (4, 8, or 12) and placing the resulting simulated responses on the same IRT ability scale as real student populations, they find that strong general-purpose models consistently outperform the average real student at every grade with no further guidance. A grade-level prompt alone does not pull a capable model down to the target ability.

Model-level control. PS² [25] interpolates, at the logits level, between a strong upper-bound LLM and a weak lower-bound LLM fine-tuned to commit conceptual and procedural errors. They demonstrate that this preserves monotone accuracy ordering at a finer granularity than prompt baselines, which invert orderings at five to seven levels. PS²’s proficiency dial is nonetheless still a single scalar, the interpolation ratio between the two models. Our SSKG grounding replaces that single dial with a per-triple mastery vector.

TABLE I
ILLUSTRATIVE EXAMPLE: AN SAT ALGEBRA ITEM.

Question Stem	<i>Alan drives an average of 100 miles each week. His car can travel an average of 25 miles per gallon of gasoline. Alan would like to reduce his weekly expenditure on gasoline by \$5. Assuming gasoline costs \$4 per gallon, which equation can Alan use to determine how many fewer average miles, m, he should drive each week?</i>	
Choices	(A) $\frac{25}{4}m = 95$ (C) $\frac{4}{25}m = 95$	(B) $\frac{25}{4}m = 5$ (D) $\frac{4}{25}m = 5$
Correct answer	D	
Skill	Linear equations in one variable	
Difficulty	Hard (College-Board-calibrated)	
Rationale	<i>Choice D is correct. Since gasoline costs \$4 per gallon, and since Alan’s car travels an average of 25 miles per gallon, the expression $4/25$ gives the cost, in dollars per mile, to drive the car. Multiplying by m gives the cost for Alan to drive m miles in his car. Alan wants to reduce his weekly spending by \$5, so setting $4/25 m$ equal to 5 gives the number of miles, m, by which he must reduce his driving.</i>	

Architectural control. BEAGLE [26] is the closest precedent to our design commitment that architecture, not prompting, should enforce fidelity. It applies a semi-Markov controller over metacognitive states, Bayesian Knowledge Tracing (BKT) with explicit flaw injection, and a decoupled strategist/executor. On open-ended Python problem solving, BEAGLE’s epistemic-fidelity error-recurrence rate (86.2%) sits well above nine prompting baselines (as low as 7.8% for vanilla prompting). BEAGLE’s domain, longitudinal open-ended coding trajectories, is disjoint from ours, single-shot multiple-choice items.

III. ILLUSTRATIVE EXAMPLE

Table I presents an SAT Algebra item, including question stem, four answer choices, correct answer, College-Board-assigned skill and difficulty, and its official rationale.

Two features make this item a useful running example. First, the 100 miles/week figure in the stem is itself a distractor at the reading-comprehension level. It plays no role in the correct equation. Second, its three wrong choices are produced by two distinct, independently identifiable single-step errors, an inverted unit rate (A, B) and a mistranslated right-hand side (A, C). We can trace exactly which curriculum fact corresponds to a simulated failure.

IV. THE FORMALISM AND SIMULATION SKETCH

A. Formal Definitions

A faithful simulator requires a body of curriculum knowledge, the knowledge used to solve the item question, a mapping from triple-level mistakes to likely incorrect answers, and a student’s mastery of the curriculum knowledge. We now formalize these three objects as: curriculum knowledge graph (CKG), triple chain, distractor map, and student profile represented as a stochastic student knowledge graph (SSKG).

Definition 1 (Curriculum Knowledge Graph). A CKG is a pair (T, pos) where $T = \{t_1, \dots, t_n\}$ is a set of triples $t_i = (\text{subj}_i, \text{pred}_i, \text{obj}_i)$ and $\text{pos} : T \rightarrow \mathbb{N} \times \mathbb{R}$ maps each triple to a (chapter, section) position. This induces a curriculum-order preorder $\preceq$ on T by lexicographic comparison,

$$t_i \preceq t_j \iff \text{pos}(t_i) \leq_{\text{lex}} \text{pos}(t_j), \quad (1)$$

with $t_i \sim t_j$ (an unordered layer) when $\text{pos}(t_i) = \text{pos}(t_j)$.

Example 1 (CKG). Figure 3 shows an excerpt of the CKG extracted from an algebra textbook. Each triple is annotated with its (chapter, section) position, for example, (ch1, sec1.8).

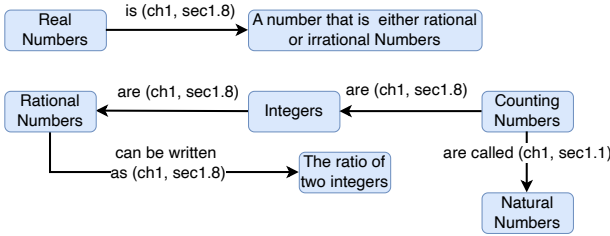


Fig. 3. An excerpt of a CKG’s number-system connections

Definition 2 (Triple chain). For an item q , the chain $\text{chain}(q) = (t_1, \dots, t_k)$ is an ordered sequence of distinct triples drawn from T . Since T may miss some triples in solving q , we denote $\text{gap}(q)$ as the set of steps required by q but not backed by any triple in T . Together, $\text{chain}(q)$ and $\text{gap}(q)$ account for every step of q ’s solution derivation.

Example 2 (Triple Chain). Table II contains the triple chain of the illustrative item in Section III. The triples are identified from the CKG as introduced in Example 1. Each triple has a unique id in the CKG and its predicate is classified into different types (details in Section V.)

TABLE II
EXCERPT OF THE ILLUSTRATIVE ITEM’S TRIPLE CHAIN.

Id	Abbreviated triple (subject, predicate, object)	Type
T19	(Variable, is, A letter representing a changeable number)	def.
T149	(Converting units, uses, The identity property of mult.)	rel.
T150	(Making unit conversions, has_step_1, Multiply by a unit fraction)	proc.
T170	(Translating a sentence, has_step_1, Locate the “equals” word)	proc.
T171	(Translating a sentence, has_step_2, Translate the left side)	proc.
T172	(Translating a sentence, has_step_3, Translate the right side)	proc.

Definition 3 (Distractor Map). For an item q with answer choices $\text{choices}(q)$ and correct choice $y^*(q) \in \text{choices}(q)$, a distractor map is a partial function

$$\text{dmap}(q) : \text{chain}(q) \rightarrow \text{choices}(q) \setminus \{y^*(q)\}, \quad (2)$$

assigning to each mapped triple $t_i \in \text{chain}(q)$ the specific wrong choice produced when t_i ’s step, and only that step, fails. A triple outside the domain of $\text{dmap}(q)$ is unmapped; a failure localized there falls back to a uniform draw over $\text{choices}(q) \setminus \{y^*(q)\}$.

Example 3 (Distractor Map). Figure 4 traces the illustrative item’s chain against its distractor map: a failure at T149/T150 (the unit-conversion step) routes to choice B, and a failure at T170–T172 (the sentence-to-equation translation) routes to choice C. T19 is unmapped, since failing at this definitional step alone does not reproduce any distractor’s exact value. Choice A is likewise outside $\text{dmap}(q)$ ’s range, since it compounds both errors simultaneously and so cannot be attributed to a single triple’s failure.

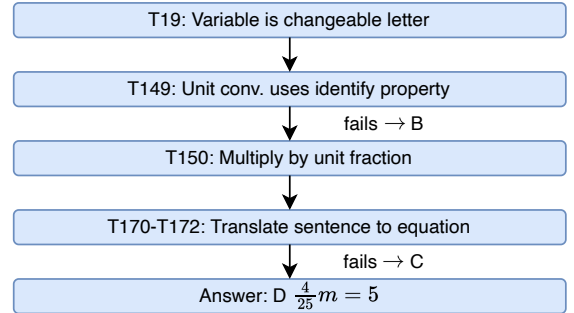


Fig. 4. The illustrative item’s triple chain and distractor map: failures localized at T149/T150 and T170–T172 route to choices B and C, respectively.

Definition 4 (Student Profile as a Stochastic Student Knowledge Graph). A stochastic student knowledge graph (SSKG) represents a student’s profile as mastery probabilities over the

CKG’s triples: at any time point θ , each triple $t \in T$ carries a mastery probability $p(t, \theta) \in [0, 1]$, and the vector $(p(t, \theta))_{t \in T}$ changes from one time point to the next as the student gains or forgets a piece of knowledge. This study considers the single time point at which a student answers an SAT item; fixing θ and suppressing it from the notation, a profile π at simulation time is a pair $(p_\pi, p_{\text{gap}, \pi})$ where $p_\pi : T \rightarrow [0, 1]$ assigns each triple a mastery probability and $p_{\text{gap}, \pi} \in [0, 1]$ is a single fixed execution rate applied to every known-gap step under π .

Example 4 (Student Profile as an SSKG). The curriculum knowledge graph (CKG) for algebra contains 864 triples. A student’s profile at the moment of answering is then an SSKG assigning a mastery probability to each of these 864 triples, that is, an 864-dimensional vector p_π . For instance, an average student is represented by drawing each of the 864 values from a uniform distribution $U(0.50, 0.70)$.

Problem of Faithful Simulation. Given a profile π and an item q , let $\text{correct}(q, \pi) := \mathbb{1}[y(q, \pi) = y^*(q)]$ denote whether the produced response $y(q, \pi)$ matches item q ’s correct choice $y^*(q)$. The task is to produce a predicted response $y(q, \pi) \in \text{choices}(q)$ and a rationale $r(q, \pi)$, satisfying the following properties:

- (Prop1) *Monotonicity.* For profiles π, π' with $p_\pi(t) \geq p_{\pi'}(t)$ for all $t \in T$, $\mathbb{E}[\text{correct}(q, \pi)] \geq \mathbb{E}[\text{correct}(q, \pi')]$ for every q .
- (Prop2) *Skill specificity.* For a profile whose mastery vector partitions T into a strong region and a weak region, per-item accuracy differs systematically between items whose chain draws mostly on the strong region versus the weak region.
- (Prop3) *Diagnosability.* An incorrect $y(q, \pi)$ can be attributed to a specific $t^* \in \text{chain}(q)$ whose low mastery under π produced the failure.

B. Simulation sketch

The SSKG-based method solves the **Problem of Faithful Simulation** by separating *deciding* from *narrating*. Given an item’s required triple chain $\text{chain}(q)$ and a profile’s mastery vector p_π , an analytic sampling procedure first decides the outcome, which chain steps the profile succeeds or fails at, and, on failure, which specific wrong choice results, entirely without invoking an LLM. Only afterward is an LLM invoked to narrate a first-person rationale consistent with the outcome. In next sections, we detail the construction of each component and the simulation process.

V. CONSTRUCTING CKG

The curriculum knowledge graph (CKG) was extracted from an open algebra textbook through a chapter-to-section-to-triple pipeline: the text was split into chapters and then sections, each section was cleaned to retain only its expository content, and (subject, predicate, object) triples were extracted using an LLM-assisted, human-verified process seeded by a manually-extracted convention set. Finally, the resulting CKG was checked to achieve the right rate-distortion balance, retaining

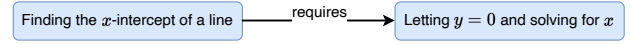


Fig. 5. A procedure node: a precondition triple for the Finding the x -intercept of a line procedure.

the right amount of information from the source [27]. The resulting CKG contains 864 triples spanning ten chapters, with each triple tagged by the chapter and section it came from.

Each triple’s predicate is further classified into one of three types by a taxonomy built over the 163 distinct predicates in T : *definitional* (e.g. *is*, *is_called*, vocabulary that even weak students usually retain, 461 triples), *procedural* (e.g. *has_step_N*, executable steps whose failure changes a computed result, 302 triples), and *relational* (e.g. *uses*, *needs*, method-selection facts whose failure sends the student down the wrong approach entirely, 101 triples).

A. Concept and Procedure nodes

There are two types of nodes in the CKG. Some triples’ subjects/objects denote persisting mathematical objects or categories, for example, *Real Numbers*, *Rational Numbers*, *Integers* in Figure 3. These nodes are connected through *definitional* predicates like (*is*, *are*, *can_be_written_as*). Other triples’ subjects/objects denote a procedure itself, expressed as a gerund or infinitive phrase (e.g. *evaluating an expression*). Figure 5 shows a single such triple, a precondition fact rather than a persisting object. We treat this as a property of the predicate rather than a separate node-type field in the CKG’s schema. Procedure-denoting concepts are overwhelmingly paired with *procedural* predicates like *has_step_N*.

VI. IDENTIFYING TRIPLE CHAIN

For each assessment item, we identified a chain of required triples through the human-AI collaboration workflow described below.

A. AI Annotation

For a given item, an annotator (LLM-assisted) jointly reads the question stem, the answer choices, and the item’s official rationale, then walks through the rationale’s derivation step by step, matching each step against the CKG to locate the triple whose subject, predicate, and object support it. Applying this chain of triples in sequence derives the item’s correct answer.

For example, Table II lists the triples identified for the illustrative item. A few representative steps show the matching process: the rationale’s “4/25 gives the cost... in dollars per mile” step is supported by T149/T150 (unit conversion via the identity property of multiplication), while its implicit “set up the equation from the English sentence” step is supported by T170–T172 (the three steps of translating a sentence into an equation).

The illustrative item also contains one *known-gap* step, which composes a \$/gallon rate and a mi/gallon rate into a combined \$/mile rate; the CKG approximates this composition

through its unit-conversion triples but does not state it as a general principle.

B. Human verification

Every candidate chain is then human-reviewed against three criteria: (i) each triple genuinely justifies the derivation step it is attached to; (ii) the chain is complete, every step in the official rationale is either covered by a triple or a known gap; and (iii) the chain is not padded, no triple is included that the derivation does not actually depend on.

VII. CONSTRUCTING THE DISTRACTOR MAP

To identify a curriculum triple to the specific wrong-answer choice its failure would produce, we build such a distractor map also through human-AI collaboration.

A. Annotation

For each item, given the correct triple chain, an LLM-assisted annotator reverse-engineers every wrong choice from two directions: forward from the item’s official rationale, which often explains why a specific distractor is wrong and therefore what mistake it encodes, and backward from the distractor’s value itself, comparing it against the correct answer to hypothesize which step was misapplied (an inverted fraction points to a unit-conversion step, a sign flip to a negation step, and so on). For the illustrative item, the following shows some example steps landing on distractors: choice B ($25/4m = 5$) inverts the unit rate at T149/T150, and choice C ($4/25m = 95$) mistranslates the right-hand side at T170–T172; choice A compounds both errors simultaneously and so is not attributable to any single triple.

B. Verification

Every proposed single-triple mapping is then verified by re-derivation: the chain is re-executed with only that one step deliberately wrong, and the mapping is kept only if this reproduces the distractor’s exact value. This verification is deliberately conservative: a speculative mapping that cannot be independently re-derived is discarded rather than kept, and the corresponding distractor is left unmapped, which is a safe outcome since routing then falls back to a uniform-random wrong choice for that failure. Any distractor left unmapped, or any mapping that felt weak even after being kept, is logged to a running issues file rather than silently dropped, so a human reviewer can revisit it later.

VIII. REPRESENTING PROFILES AS SSKGS

Aiming to simulate students who are representative in terms of mastery levels, we define five archetypal profiles P1-P5 (Table III), each of which is one concrete SSKG.

Each profile is specified in two coupled forms, the natural-language archetype (second column of Table III) and the SSKG itself. To instantiate a profile’s SSKG, the mastery probability $p_\pi(t)$ of each of the 864 CKG triples is drawn independently from the band distribution in the third column, and the known-gap execution rate $p_{\text{gap},\pi}$ is drawn once from the profile’s own band (the later-chapter band for P4 and P5).

The five archetypes divide into two groups by the structure of their mastery vectors. P1-P3 are uniform-mastery archetypes spanning near-expert to struggling. P4 and P5 are diagnostic: both sit at an intermediate overall mastery level but concentrate their gap in complementary regions, early-chapter foundations (chapters 1-2, 214 of the 864 triples) versus later-chapter advanced material (chapters 3-10). The chapter threshold is chosen so that the 379 items split into a balanced 140 early-only, 143 mixed, and 96 later-only partition by their chains.

TABLE III
STUDENT PROFILES AS SSKG SNAPSHOTS: NATURAL-LANGUAGE ARCHETYPE AND PER-TRIPLE MASTERY GENERATION.

Profile	Archetype	Mastery Probability Generation
P1	near-expert	$U(0.85, 0.99)$ all triples
P2	average, inconsistent	$U(0.50, 0.70)$ all triples
P3	struggling	$U(0.10, 0.30)$ all triples
P4	prerequisite gap	$U(0.10, 0.25)$ early, $U(0.60, 0.80)$ later
P5	strong basics, weak advanced	$U(0.80, 0.95)$ early, $U(0.15, 0.35)$ later

IX. THE SIMULATION

To simulate the students with different levels of mastery, we propose a method grounded in SSKG with two stages.

A. Two-stage architecture

The SSKG-grounded method consists of two separate stages,

$$\text{Stage}_1 : y(q, \pi) = \mathcal{S}_1(\text{chain}(q), \pi, \sigma), \quad (3)$$

$$\text{Stage}_2 : r(q, \pi) = \mathcal{S}_2(y(q, \pi), \tau(q, \pi, \sigma)), \quad (4)$$

where σ is a recorded seed and $\tau(q, \pi, \sigma)$ is the Stage_1 trace (which chain steps succeeded, which failed and how). $\mathcal{S}_1$ is deterministic given σ and requires no LLM call: it samples whether each triple in the chain is answered correctly and combines these draws into an overall outcome and, on failure, an answer choice. $\mathcal{S}_2$ invokes an LLM, supplying it the serialized SSKG scoped to q ’s chain, the trace $\tau(q, \pi, \sigma)$, and the outcome $y(q, \pi)$. The LLM’s task is to narrate a first-person rationale consistent with that trace.

B. Theoretical foundations

Stage_1’s outcome model rests on two long-standing ideas from student modeling.

The first idea, from Corbett and Anderson’s Bayesian Knowledge Tracing [18], is that a student can fail a step for two different reasons: the required piece of knowledge may not be available at all (*retrieval*), or it may be available and still be misapplied through a careless error (*execution*; in that literature, a *slip*). Stage_1 models each chain step i accordingly, with two Bernoulli draws,

$$\begin{aligned} \text{knows}_i &\sim \text{Bernoulli}(p_i), \\ \text{slips}_i &\sim \text{Bernoulli}(s) \text{ if } \text{knows}_i = 1, \end{aligned} \quad (5)$$

where $p_i = p_\pi(t_i)$ is the profile’s mastery probability for the step’s triple, and s is a single small, profile-independent slip probability: the chance that a student who does know a triple nevertheless misapplies it (we set $s = 0.05$, about one careless error per twenty known steps). Step i succeeds iff $\text{succ}_i = \text{knows}_i \wedge \neg \text{slips}_i$. The payoff of the two-draw decomposition is that every failure carries a *kind*, that is, a recorded cause: a retrieval failure ($\text{knows}_i = 0$, the knowledge was missing) or an execution failure ($\text{knows}_i = 1 \wedge \text{slips}_i = 1$, the knowledge was there but slipped in use). A single-draw model (arm-a below) can record only an *undifferentiated miss*: the step failed, with no indication of why. The kind is what Stage_2 narrates and what the diagnosability property (Prop3) needs.

The second idea comes from cognitive diagnosis, a psychometric tradition that explains a student’s item responses by which discrete skills (*attributes*) the student has mastered, rather than by a single overall ability score. Its baseline model is DINA, the Deterministic Input, Noisy “And” gate model [20]: each item requires a specific set of attributes, and the student answers correctly only if *every* required attribute is mastered, an “and” gate with slip and guess noise on top. The rule is conjunctive rather than compensatory: one missing attribute sinks the item, and strength elsewhere cannot make up for it. Stage_1 adopts exactly this rule, with CKG triples playing the role of attributes and $\text{chain}(q)$ the role of the item’s required set,

$$\text{correct}(q, \pi) = \prod_{i: t_i \in C(q)} \text{succ}_i, \quad (6)$$

where $C(q) \subseteq \text{chain}(q)$ is the *active set* of steps that count toward correctness. This product is the construction of the method-agnostic $\text{correct}(q, \pi)$: on success Stage_1 outputs the correct choice $y(q, \pi) = y^*(q)$ directly, and on failure it selects a choice from $\text{choices}(q) \setminus \{y^*(q)\}$ (uniformly, or via distractor routing under arm-d), so $\text{correct}(q, \pi) = 1[y(q, \pi) = y^*(q)]$ holds by construction without Stage_1 ever comparing its answer to the key.

Plain DINA treats every required attribute as equally necessary, $C(q) = \text{chain}(q)$. That is too blunt for a curriculum graph whose triples differ in type: forgetting a vocabulary definition should not sink an item the way failing a procedural step does. G-DINA, de la Torre’s generalized DINA framework [21], relaxes exactly this assumption by letting different attributes contribute to correctness with different weights instead of through an all-or-nothing gate. We modify Eq. (6) with a type-weighted rule. Each definitional triple has its mastery compressed toward the top,

$$p_{\text{def}} = 1 - \kappa_{\text{def}}(1 - p_i), \quad \kappa_{\text{def}} \in (0, 1), \quad (7)$$

and is dropped from the active set, $C(q) = \text{chain}(q) \setminus \{\text{definitional } t_i\}$, so a definitional failure can no longer change the outcome and surfaces only in the narration, while procedural and relational triples retain their full discriminating power

TABLE IV
THE FOUR-ARM ABLATION LADDER. EACH ARM ADDS ONE MECHANISM ON TOP OF THE PREVIOUS.

Arm	Adds
a	Single draw $\text{ok}_i \sim \text{Bernoulli}(p_i)$ per triple, raw taxonomy-blind p_i , guess floor 0.25 on failure.
b	Retrieval/execution decomposition: $\text{knows}_i \sim \text{Bernoulli}(p_i)$, then $\text{slips}_i \sim \text{Bernoulli}(s=0.05)$; still taxonomy-blind p_i .
c	Taxonomy awareness: definitional $p_i \rightarrow p_{\text{def}}$ per Eq. (7) ($\kappa_{\text{def}}=0.3$); active set $C(q)$ in Eq. (6) drops to procedural/relational/gap steps (definitional failures are narration-only).
d	Distractor routing: an execution failure at a mapped triple selects that triple’s specific wrong-answer choice instead of a uniform-random one.

and stay in $C(q)$. In G-DINA terms, definitional triples are demoted from necessary attributes to low-discrimination ones.

C. The four-arm ablation ladder

Table IV summarizes the four experimental arms. Each arm adds one new mechanism to the previous arm.

For each (item, profile) pair, we generate random draws for every step in the solution chain using a single recorded seed σ . One draw determines whether the student knows the required triple, and another determines whether the student makes a slip. The same random draws are reused across all four arms using common random numbers (σ). This allows us to compare the arms on the same item and profile under identical random conditions.

D. Illustrative Example, Continued

Consider the illustrative example item from Section III under profile P3, which has a mean per-triple mastery of 0.201. Suppose the seeded draw shows that the student fails triple T150, a procedural triple involving a step in unit conversion.

In arm-a and arm-b, any failure in the solution chain makes the answer incorrect, without distinguishing the type of failure. In arm-c, the failure still makes the answer incorrect because T150 is procedural rather than definitional. However, the model now records the failure as an *execution* error.

In arm-d, the failure is also linked to a specific distractor. For this item, an execution error at T149 or T150 leads to choice B, $\frac{25}{4}m = 5$. This represents a realistic unit-conversion mistake: dividing miles by dollars instead of multiplying by a unit fraction that cancels gallons. The model therefore selects choice B rather than choosing randomly among the incorrect answers.

The LLM is then asked to generate a rationale that reflects this specific mistake and carries it through to choice B. For example, it should describe the unit-conversion error rather than simply saying, “I made an arithmetic error.”

By contrast, a failure at T19, which is a definitional triple, would not make the answer incorrect in arm-c or arm-d. Instead, it would appear only as uncertainty or hesitation in the student’s explanation.

X. EVALUATION

We evaluate the simulation under two methods:

- Method 1 (direct prompting) lets an LLM jointly decide y and r from a natural-language description of π alone.
- Method 2 (SSKG-based) decides y analytically from $(\text{chain}(q), \pi)$ followed by LLM-generated narration.

A. Experimental design

The evaluation asks four questions, the first three tracking properties Prop1–Prop3 of the **Problem of Faithful Simulation** and the fourth attributing any gain within Method 2:

- (Q1) *Monotonicity (Prop1)*. Does accuracy fall as mastery falls, from profile P1 down to P3?
- (Q2) *Skill specificity (Prop2)*. Do the structured-gap profiles P4 and P5 fail where their gaps are, that is, on items whose chains route through their weak chapter band?
- (Q3) *Diagnosability (Prop3)*. Can each wrong answer be traced to a specific low-mastery triple, and does low mastery on a triple in fact predict errors on the items that need it?
- (Q4) *Attribution*. Which rung of the ablation ladder (a→b, b→c, c→d) is responsible for the fidelity gain?

B. Datasets

The datasets contain: (1) 379 SAT Algebra multiple-choice items, (2) an 864-triple CKG, and (3) five profiles (Table III). Method 1 was run on all $379 \times 5 = 1,895$ (item, profile) pairs for each of three LLMs (Gemini 3.1 Flash Lite, Claude Haiku 4.5, GPT-5.4-mini). Each of arm-a to arm-d was run at 1,895 records. We use Gemini as narrator. Method 2 accuracy is LLM-invariant by construction since the sampled trace, not the narrator, decides correctness.

C. Metrics

The primary metric is accuracy by profile, $\text{Accuracy}(\pi) = |Q|^{-1} \sum_{q \in Q} \mathbb{1}[y(q, \pi) = y^*(q)]$, computed per method and arm, and further stratified by the early/mixed/late chain-footprint partition (Q2) and by the SAT difficulty label. For Q3 we use Spearman rank correlations between mastery and observed errors, plus arm-d’s by-construction diagnostic labels. Because the two methods answer the *same* 379 items under each profile, method contrasts are made item-wise. We count the items on which the two methods disagree and in which direction.

D. Evaluation Results

1) *Monotonicity (Q1): direct prompting collapses; SSKG grounding recovers the gradient*: Method 1 is flat at 96.8–100% for every profile (100.0%, 99.7%, 96.8%, 100.0%, and 100.0% for P1–P5), indistinguishable from the 100% unconstrained ceiling, including under P3, whose prompt explicitly describes a “struggling student who has low mastery of most algebra concepts.” The collapse is not specific to one vendor or model family: Claude Haiku 4.5 and GPT-5.4-mini are flatter still, 99.2–99.7% on every profile, with GPT’s P3 actually

exceeding its P2. Since no LLM is meaningfully more profile-sensitive than the others under Method 1, we report Gemini numbers in the remainder of this section.

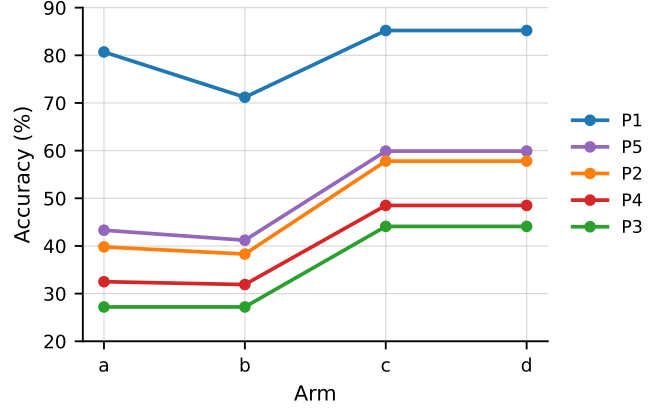


Fig. 6. Accuracy by profile across the four Method 2 arms (LLM-invariant). Every arm spreads the five profiles far below Method 1’s flat 96.8–100% band; the b→c rung produces the largest change, and arm-c and arm-d coincide exactly.

Every Method 2 arm, by contrast, spreads the five profiles across a 30–55% range (Figure 6); arm-d spans 44.1–85.2%. On the mastery-ordered profiles that Q1 directly concerns, arm-d descends 85.2% (P1) > 57.8% (P2) > 44.1% (P3), and each adjacent gap is decisive, whereas Method 1’s only nonzero gap (P2 vs. P3, 2.9%) is the entire mastery signal it produces. Method 1 is not “more accurate” in any useful sense; it is insensitive to the profile. Note also that the SSKG accuracy order (P5 above P4 despite P5’s lower mean mastery) reflects *where* each profile’s mastery sits relative to the item chains, the structure a single proficiency dial cannot express (Section VIII).

2) *Skill specificity (Q2)*: Partitioning the 379 items by chain footprint (140 early-only, 143 mixed, 96 later-only), Table V tests whether P4 (weak early) and P5 (weak later) fail on the items that route through their weak band. Method 1 is 100.0% in every one of the six cells: direct prompting cannot express a skill-localized gap at all. Under arm-d, P4 shows the clean predicted pattern, weak on early and mixed items and 35.7% stronger on later-only items. P5 is strongest on early items (70.0%), as designed, but its weakest cell is mixed (48.3%) rather than later (62.5%). Mixed chains are longer on average, and under a conjunctive rule every extra step multiplies in another survival probability, so a mixed item forces P5 through its weak-later steps *plus* a length penalty. The mixed column should be read as “both bands required, more total steps,” not as an interpolation.

A label-blind difficulty check. The SAT difficulty label is never an input to Stage 1, so any difficulty pattern in the output is emergent. Under arm-d, P3 shows a clean monotone Easy > Medium > Hard slope (48.3% > 40.8% > 39.2%), while the other profiles are flat or non-monotone, consistent with the fact that the label correlates only weakly with chain length and

TABLE V
ACCURACY BY CHAIN FOOTPRINT (140 EARLY-ONLY, 143 MIXED, 96 LATER-ONLY ITEMS) FOR THE STRUCTURED-GAP PROFILES.

Profile	Method	Early	Mixed	Later
P4 (weak early)	Method 1	100.0%	100.0%	100.0%
	arm-d	39.3%	39.9%	75.0%
P5 (weak later)	Method 1	100.0%	100.0%	100.0%
	arm-d	70.0%	48.3%	62.5%

composition (Spearman 0.209), which are what actually drive the sampler. Method 1 is flat at or near 100% across difficulty for every profile except a slight P3 decline.

3) *Diagnosability (Q3)*: Under arm-d, every wrong answer is assigned a `first_failed_triple_id` and a `failure_kind` (retrieval or execution) by construction. We therefore evaluate diagnosability empirically, independent of this labeling mechanism. Pooling the 525 (triple, profile) pairs in which each triple appears in at least 3 item chains, triple mastery under a profile correlates with the profile’s error rate on those items at Spearman $\rho = -0.459$ ($p \approx 10^{-28}$). Thus, lower mastery of a specific triple is associated with more errors on the items that require it. At the single-item level, the chain’s bottleneck triple (the one with minimum mastery) correlates with the item outcome at $\rho = 0.318$. This weaker correlation is expected because a single bottleneck value cannot fully capture a conjunctive chain. The diagnostic labels are also faithfully reflected in the generated responses. Across all 1,895 arm-d narrations, the narrated final answer matches the sampler-decided answer in every record (0 violations), including all 19 records in which distractor routing was triggered and the narration had to derive the specific mapped wrong choice.

4) *Attribution (Q4)*: Figure 6 shows how each arm contributes to the overall gain in simulation fidelity.

a→b (execution noise). Accuracy decreases by 9.5% for P1 and by at most 2.1% for any other profile. The effect is concentrated where expected. All 36 items that P1 newly fails under arm-b are classified as *execution* failures, with none classified as *retrieval* failures. A near-expert who never slips is not a realistic top student. The slip mechanism is therefore what moves P1 below its initial ceiling, while low-mastery profiles already fail mainly because of retrieval errors that the slip rate does not affect.

b→c (taxonomy awareness). This is the largest gain. Accuracy increases by 14.0 – 19.5% across all profiles as definitional triples are no longer treated as chain-fatal failures. The gain is slightly larger for lower-mastery profiles, with P2–P5 improving by 16.6 – 19.5% compared with 14.0% for P1. This is because definitional triples are common (461 of 864), so treating them as non-fatal helps any chain that contains one. As a mechanism check, the 51 items whose chains are entirely definitional achieve 100.0% accuracy for all five profiles from arm-c onward, exactly as guaranteed by the construction.

c→d (distractor routing). Accuracy remains unchanged at 0.0% for every profile. The item-wise seed invariants also hold with zero violations across all 1,895 matched records: arm-b

TABLE VI
ABLATION RUNGS: MAGNITUDE AND FORMAL STATUS.

Rung	Δ range	Status	
a→b	−9.5 +0.0pp	to	empirical (execution-noise effect)
b→c	+14.0 +19.5pp	to	empirical (taxonomy de- caricaturing, mild, Section X)
c→d	0.0pp, profile	every	exact, by construction

$\leq$ arm-a, $\text{arm-c} \geq \text{arm-b}$, and $\text{arm-d} = \text{arm-c}$. This rung therefore contributes only to diagnosis. In 19 of 1,895 records (1.0%), an execution failure is routed to its mapped wrong choice rather than selected uniformly. In each case, the narration describes the specific single-step error rather than a generic mistake.

XI. DISCUSSION

The results show that direct prompting tends toward a high-accuracy ceiling across all three vendors, regardless of the target mastery level. In contrast, explicit stochastic sampling over the curriculum graph produces the monotone accuracy gradient and skill-specific error patterns required by **the Problem of Faithful Simulation**.

The ablation study shows that these gains come from different mechanisms. The knowledge-tracing decomposition (a→b) mainly moves high-mastery profiles away from an unrealistic near-ceiling while having little effect on already-low profiles. This is the expected direction. A top student who never makes an execution slip is unrealistic. However, this mechanism has a relatively small effect because it changes only one parameter. Taxonomy awareness (b→c) produces the largest gain and has a somewhat stronger effect on lower-mastery profiles. Distractor routing (c→d) does not change accuracy, as expected, but improves the diagnostic value of wrong answers. The results for skill specificity and emergent difficulty are more mixed. P4 shows a clear early/late split, P5’s pattern is affected by chain length, and only P3 shows a clear difficulty gradient. Together, these results suggest that the method captures meaningful aspects of student behavior rather than simply reproducing patterns built into the simulation.

LLM Costs. Because Stage_1 requires no LLM calls, the system can generate an arbitrarily large and exactly reproducible response corpus while using LLM calls only for narration. Every wrong answer is assigned a `first_failed_triple_id` and a `failure_kind` by construction. These labels provide the kind of diagnostic information that knowledge-tracing models need but that is usually unavailable in real classroom data. This creates a direct path to downstream knowledge-tracing and curriculum-graph research, such as training an SSKG-based diagnostic tracer or modeling how mastery changes over time on the same graph.

XII. LIMITATIONS

Several key limitations should be noted. First, building the curriculum knowledge graph (CKG), solution triple chains,

and distractor maps requires substantial human effort. Reducing this manual workload through more automated methods is an important work. Second, our evaluation shows that the simulation produces a monotone performance gradient across mastery profiles, but it does not yet establish how closely the simulated accuracy matches real student performance. Grounding and validating the simulation against real student response data is therefore an important next step. Finally, we have not yet conducted a direct comparison between our approach and existing methods in the literature.

XIII. CONCLUSION

Our results show that direct natural-language prompting does not reliably simulate students at different mastery levels. We address this limitation by moving the mastery model outside the prompt and into a Stochastic Student Knowledge Graph (SSKG). The simulation samples mastery over a curriculum-derived chain of required triples to determine the outcome before the LLM generates a response. This approach restores a monotone mastery gradient, produces skill-specific error patterns, and assigns diagnostic labels to every synthetic response by construction. The computational cost also scales mainly with the amount of generated narration rather than the number of decisions.

Our four-arm ablation further shows how each component contributes to simulation fidelity. Taxonomy-conditioned item weighting provides the largest improvement, while distractor routing adds a smaller but meaningful improvement by producing more realistic and diagnostically useful incorrect responses.

Several directions remain for future work. First, the proposed simulation should be compared directly with existing prompt-level, model-level, and architecture-level approaches to student simulation. Second, simulated performance and item difficulty should be validated against external Item Response Theory statistics and real student data, rather than relying only on the label-blind evaluation used here. Finally, the corpus and simulation framework can be extended to support downstream knowledge-tracing models trained on the by-construction diagnostic labels. These steps would connect the present work to a broader research program on stochastic knowledge graphs and support the development of a benchmark for evaluating faithful student simulation.

REFERENCES

- [1] H. Jung, J. Yoo, Y. Yoon, and Y. Jang, "CLST: Cold-start mitigation in knowledge tracing by aligning a generative language model as a students' knowledge tracer," *Journal of Educational Data Mining*, vol. 17, no. 2, 2025.
- [2] H. Noh, A. Chowdhary, J. Ooge, V. Aleven, and C. Borchers, "Simulating learners' task-selection strategies and system constraints in mastery learning," in *Proceedings of the 19th International Conference on Educational Data Mining*, 2026.
- [3] A. Säuberli, D. Frassinelli, and B. Plank, "Do LLMs give psychometrically plausible responses in educational assessments?" in *Proceedings of the 20th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, 2025.
- [4] OpenAI, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.
- [5] J. de Winter, D. Dodou, and Y. B. Eisma, "System 2 thinking in OpenAI's o1-preview model: Near-perfect performance on a mathematics exam," *Computers*, vol. 13, no. 11, p. 278, 2024.
- [6] L. Benedetto, G. Aradelli, A. Donvito, A. Lucchetti, A. Cappelli, and P. Buttery, "Using LLMs to simulate students' responses to exam questions," in *Findings of the Association for Computational Linguistics: EMNLP 2024*. Miami, Florida, USA: Association for Computational Linguistics, 2024, pp. 11 351–11 368.
- [7] K. V. A. Srivatsa, K. K. Maurya, and E. Kochmar, "Can LLMs reliably simulate real students' abilities in mathematics and reading comprehension?" in *Proceedings of the 20th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, 2025.
- [8] X. Lu and X. Wang, "Using LLM-simulated student profiles to support question item evaluation," in *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, ser. L@S '24. ACM, 2024.
- [9] College Board, "SAT Suite Educator Question Bank," 2026, accessed: 2026-02-13. [Online]. Available: <https://satsuiteeducatorquestionbank.collegeboard.org/>
- [10] J. Yu, G. Luo, T. Xiao, Q. Zhong, Y. Wang, W. Feng, J. Luo, C. Wang, L. Hou, J. Li, Z. Liu, and J. Tang, "MOOCube: A large-scale data repository for NLP applications in MOOCs," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 3135–3142.
- [11] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE TKDE*, vol. 36, no. 7, pp. 3580–3599, 2024.
- [12] S. Tsaneva, D. Dessì, F. Osborne, and M. Sabou, "Knowledge graph validation by integrating LLMs and human-in-the-loop," *Information Processing & Management*, vol. 62, no. 5, p. 104145, 2025.
- [13] P. C. G. da Costa and K. B. Laskey, "PR-OWL: A framework for probabilistic ontologies," in *Proceedings of the 4th International Conference on Formal Ontology in Information Systems (FOIS)*, ser. Frontiers in Artificial Intelligence and Applications, vol. 150, 2006, pp. 237–249.
- [14] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2010.
- [15] X. Chen, M. Chen, W. Shi, Y. Sun, and C. Zaniolo, "Embedding uncertain knowledge graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, 2019, pp. 3363–3370.
- [16] S. Karlin and H. M. Taylor, *A First Course in Stochastic Processes*, 2nd ed. New York: Academic Press, 1975.
- [17] K. VanLehn, "The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems," *Educational Psychologist*, vol. 46, no. 4, pp. 197–221, 2011.
- [18] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Modeling and User-Adapted Interaction*, vol. 4, no. 4, pp. 253–278, 1995.
- [19] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in *Advances in Neural Information Processing Systems*, vol. 28, 2015, pp. 505–513.
- [20] B. W. Junker and K. Sijtsma, "Cognitive assessment models with few assumptions, and connections with nonparametric item response theory," *Applied Psychological Measurement*, vol. 25, no. 3, pp. 258–272, 2001.
- [21] J. de la Torre, "The generalized DINA model framework," *Psychometrika*, vol. 76, no. 2, pp. 179–199, 2011.
- [22] F. M. Lord, *Applications of Item Response Theory to Practical Testing Problems*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1980.
- [23] M. Harwell, C. A. Stone, T.-C. Hsu, and L. Kirisci, "Monte carlo studies in item response theory," *Applied Psychological Measurement*, vol. 20, no. 2, pp. 101–125, 1996.
- [24] C. Acquaye, Y. T. Huang, M. Carpuat, and R. Rudinger, "Take out your calculators: Estimating the real difficulty of question items with LLM student simulations," *arXiv preprint arXiv:2601.09953*, 2026.
- [25] R. Liu, Z. Wen, H. Yan, J. Yin, S. Wang, and J. Cao, "PS²: Parameterized control for fine-grained student proficiency simulation," *arXiv preprint arXiv:2602.00850*, 2026.
- [26] H. Wang, C. Cohn, Z. Xu, S. Guo, G. Biswas, and M. Ma, "BEAGLE: Behavior-enforced agent for grounded learner emulation," *arXiv preprint arXiv:2602.13280*, 2026.
- [27] Y. An, R. Hashmi, M. Rogers, J. Greenberg, and B. K. Smith, "Rate-distortion guided knowledge graph construction from lecture notes using gromov-wasserstein optimal transport," in *2025 IEEE International Conference on Big Data (BigData)*, 2025, pp. 5573–5582.