

LLM-As-An-Assessor: Can Open-Weight LLMs Assess Computational Thinking via Student-Designed Embodied Games?

William Lee

University of Massachusetts Amherst
williamlee@cs.umass.edu

Sai Gattupalli

Society & AI Research Group
sai@societyandai.org

Ivon Arroyo

University of Massachusetts Amherst
ivon@cs.umass.edu

Brendan O'Connor

University of Massachusetts Amherst
brenocon@cs.umass.edu

Abstract

We evaluate whether open-weight large language models (LLMs) can assess and infer Computational Thinking (CT) knowledge in middle-school students' finite-state game designs against human-coded labels. Across two game designs, one high-fidelity with rich CT behaviors and one low-fidelity with sparse CT behaviors, models detect some CT behaviors at near-human agreement. However, they also consistently over-credit behaviors that are absent and fail on basic characteristics retrieval, loop/branch detection, and state-level standards mapping. A frontier baseline, Claude Fable 5, matched the human coder perfectly on the rich game but did no better than open-weight models on the sparse one. The largest model is among the least reliable. We trace several failures to fixable task setup and argue for further investigation before such systems inform teacher-facing feedback.

1 Background

Computational Thinking (CT) is now a foundational K-12 competency (Weintrop et al., 2021). Yet teachers lack practical tools to assess it from student work. Using the *WearableLearning* (WL) platform (developed at the Advanced Learning Technologies Lab, UMass Amherst), grades 5-8 students design games as finite state automata (FSA) in a block-based editor in which CT competence is embedded directly in each design's structure (Arroyo et al., 2023). Educators interviewed previously in a pilot study revealed the importance of formative CT assessments aligned to state and national standards, echoing broader evidence that teacher-facing analytics must be grounded in teachers' information needs (Hadi Nezhad et al., 2024). Although LLMs are capable of reading these FSA artifacts, using these models as an assessment instrument demands evidence of validity, reliability, and fairness. We focus on open-weight models because schools need assessment that runs locally, so

they can protect student data and eliminate cloud-based cost. However, their assessment abilities for structured CT artifacts remain largely undocumented. Our 2024 pilot (Lee and Arroyo, 2025) showed frontier and reasoning models can match human counts of basic FSA characteristics ($F1 \approx 1.0$) while advanced analytics (e.g., loop detection) remained fragile.

2 Related Work

Assessment, especially for CT, is an active field including rubrics, artifact analysis, and scoring (Weintrop et al., 2021). Our pilot showed that models can recover basic FSA characteristics (e.g., states) but are sensitive to prompt wording – detecting loops only once the word “loops” was reworded to “cycles” (Lee and Arroyo, 2025). Two literatures inform the present study. First, using LLMs as judges is known to introduce systematic biases and “explanations” could be unfaithful to the model's actual computation (Turpin et al., 2023), which cautions against showing models' rationales to teachers. Second, the operations our task demands have known issues: graph reasoning over structured inputs has been shown to be difficult for LLMs (Wang et al., 2023; Fatemi et al., 2024), and aggressive low-bit quantization, common in local deployment, could further degrade numerical reasoning (Li et al., 2025).

3 Objectives

This research investigates if local open-weight LLMs can assess student-created CT artifacts. **RQ1:** To what extent can open-weight LLMs detect the CT design patterns presented in student FSA games when compared and measured against human-coded labels? **RQ2:** To what extent does CT behavior and standards detection become reliable enough to support valid formative feedback? An additional goal is to identify the evaluation

metrics, prompt structures, and rubric designs that make an *LLM-As-An-Assessor* pipeline trustworthy and informative for educators, researchers, and camp counselors.

4 Data

The corpus consists of two student-designed WL games. Each game is represented in a structured JSON format (states, transitions, connections, branching). One game is a high-fidelity design with complex patterns (Fraction Pong) while the other is a low-fidelity design with simple patterns (Square Roots). For each game, a human-coded standard captures 13 CT behavior judgments and the basic characteristics. Basic characteristics are extracted by parsing the JSON via Python. Hence, we found Fraction Pong has 23 states, 36 transitions, 38 connections, 5 cycles, and 7 branches while Square Roots has 12 states, 10 transitions, 11 connections, 0 cycles, and 0 branches. As for gold-standard label reliability, two researchers independently coded the 13 behaviors for two games. Agreement was reached on 84.6% of judgments with Cohen’s $\kappa = 0.692$ (Cohen, 1960). Square Roots was coded identically (100%) and all disagreement fell on Fraction Pong (69%). Disagreements were resolved by discussion to form the gold labels.

5 Methods

Each model receives an identical payload – the final-design JSON and the WL behavior rubric. Inference runs locally on the open-weight models (llama.cpp behind a llama-swap proxy; vLLM for larger models) and, for the baseline, the Claude API. We evaluate a set of open-weight LLMs (Table 1) chosen for architectural diversity (dense, sparse mixture-of-experts, and hybrid Mamba-Transformer, about 27B-120B parameters) with a proprietary frontier model, Claude Fable 5, serving as the baseline.

5.1 Pipeline Evolution

Our current evaluation approach has evolved through three phases. In Phase 1, each model received the full game JSON and the rubric together with a single, complex instruction as a prompt through a chat interface (Open WebUI). Results were poor: models frequently regurgitated the prompt back – restating the rubric and instructions rather than analyzing the artifact. The responses produced were fluent and professional but hollow

tables. We attribute this likely to overload, since a large structured JSON payload plus a multi-part instruction in one monolithic prompt probably exceeded what most open-weight models could hold and reason over. In Phase 2, following consultation with our advisors, we decomposed the single instruction into a series of seven sub-prompts and added in-context examples to each model. Hence, the model addressed one CT sub-task at a time with a concrete pattern to follow. This improved output quality. In Phase 3, we automated the pipeline through each model’s OpenAI-compatible API served through custom Python, llama.cpp, and llama-swap orchestration, replacing the manual chat interface. This automation enables reproducible runs, captures reasoning traces, and collects responses programmatically. Appendix A shows the Phase 1 prompt in full and summarizes the seven Phase 2 sub-prompts.

5.2 Analysis

Outputs from the LLMs are scored against the human labels. For the 13 presence/absence CT behavior judgments, we use precision, recall, F1 (Manning et al., 2008), and Cohen’s κ (Cohen, 1960), which specifically measures agreement beyond what chance would produce. We report per-model metrics, an item-by-model agreement heatmap, and per-model confusion matrices.

6 Results

6.1 CT Behavior Detection

Both games are scored on the same 13 behaviors against gold labels determined by the two independent coders. On the high-fidelity Fraction Pong game, 10 of 12 open-weight models gave usable output (Nemotron-3-nano and Phi-4-reasoning-plus unfortunately did not produce responses consistently). Their F1 ranged from 0.31 to 0.95 (mean ≈ 0.80 ; accuracy ≈ 0.74 ; $\kappa \approx 0.45$). Gemma-4-31B was the best open-weight model (F1 = 0.95, $\kappa = 0.75$), and the frontier baseline matched the gold exactly ($\kappa = 1.00$). On the low-fidelity Square Roots game, agreement fell (average F1 ≈ 0.59 , range 0.50 – 0.67; average $\kappa \approx 0.37$). Table 2 illustrates the per-model values.

An important finding is about how many behaviors the models say are present. Fraction Pong has 11 of 13 behaviors present and Square Roots has 4. The models did not track this difference. On average these models detected about 7.9 be-

haviors as present on Fraction Pong and about 5.3 on Square Roots. For the rich game, this means these models seem to miss detecting real CT behaviors, whereas on the sparse game, it means these models credit CT behaviors that are not actually there. The confusion matrices show this directly (Figure 2). With Fraction Pong, every model had precision = 1.00: pooled across models there were zero false positives and 34 misses. The pattern flipped for Square Roots: 31 false positives and 15 misses. We think one reason for this difference is the design of the games (Fraction Pong has only two absent CT items, and Square Roots only four present CT items). The errors also cluster on specific CT items (Figure 1). For Fraction Pong, 7 of 11 models missed both “Encouraging feedback” and “Difficulties/levels,” while all models got “Sequences” correct. For Square Roots, every model credited “Outputs: text,” which the game lacks, and every model missed “Inputs: buttons,” which it has. When most models make the same mistake, the cause is possibly in the rubric or the prompt.

6.2 Basic CT Characteristics Retrieval

When counting Fraction Pong’s basic characteristics, models did not perform well. The gold has 23 states, 36 transitions, 38 connections, 5 cycles, and 7 branches. Models undercounted these characteristics heavily. The closest, DeepSeek-R1-distill, reported 21 states, 20 transitions, 23 connections, 1 cycle, and 3 branches while several models reported zero transitions or connections. No model found all 5 cycles, which repeats the “loop-detection” failure from our pilot study. Square Roots has no cycles or branches, so it cannot test loop detection.

6.3 State-Level Standards Mapping

Earlier in the evolution of our evaluation pipeline, we attempted to see if these models can extract CT behaviors and map them to the Massachusetts Digital Literacy and Computer Science (DLCS) standards. Models regurgitated the text description of a DLCS standard instead of making any inference. We traced this to the rubric and possibly the prompt: many of our rubric items mixed the DLCS standard codes and the descriptions in one field and stacked several standards behind line breaks when feeding into the prompt. Two of eight models on Fraction Pong returned output we could not parse (for example, raw Python). The polished look of many generated Markdown tables hid these errors.

6.4 Frontier Baseline

The effectiveness of the frontier model used in this evaluation depended on the richness of a game’s design. Claude Fable 5 matched the human gold labels exactly ($F1 = 1.00$, $\kappa = 1.00$) for Fraction Pong whereas the best open-weight model only reached $\kappa = 0.75$. For Square Roots, Fable scored $F1 = 0.60$ and $\kappa = 0.37$ which is in the middle of the pack, below the three best open-weight models (RNJ-1, Granite-4-tiny, and Qwen3.6, each $\kappa = 0.49$). Interestingly, Fable’s errors on Square Roots were the same ones every other model made: it credited “Outputs: text” and “Inputs: text,” which the game did not have, and missed “Inputs: buttons.”

Hence, it seems that the frontier model was able to find every behavior that was present in a richly designed game. Yet, the frontier model, just like the other models, has difficulty confirming behaviors that were *absent* in a sparsely designed game. Rankings among open-weight models also did not carry over between games. RNJ-1 was the best open-weight model on Square Roots ($\kappa = 0.49$) and the worst on Fraction Pong ($\kappa = 0.06$, missing 9 of 11 present behaviors).

The largest model, GPT-OSS-120B, was the least reliable on Square Roots ($\kappa = 0.25$) and only performed decently on Fraction Pong ($\kappa = 0.35$). Neither being a frontier model nor having a large number of parameters predicts reliable assessment across games.

7 Lessons Learned

Beyond the quantitative agreement metrics, systematic inspection of model outputs revealed some interesting patterns. We group them into five categories. (1) Prompt regurgitation: many models often repeat the rubric or instruction instead of analyzing the artifact. This was especially common during Phase 1’s monolithic setup. (2) Description vs. standard mapping: models return a DLCS standard’s prose description rather than mapping a standard to a CT behavior, which also happened in Phase 1. (3) Count contradictions: counts of basic characteristics disagree with the game’s JSON structures suggesting that these models might have difficulties parsing messy and complex JSON objects. (4) Unparseable output: raw HTML, Python code, or malformed text was produced in place of the explicitly requested judgment, mapping, or inference (two of eight models on Fraction Pong). (5) Non-response: no usable output at all was returned

Item-by-model agreement with human gold on the same 13 WL behaviors (models sorted by Cohen's κ)

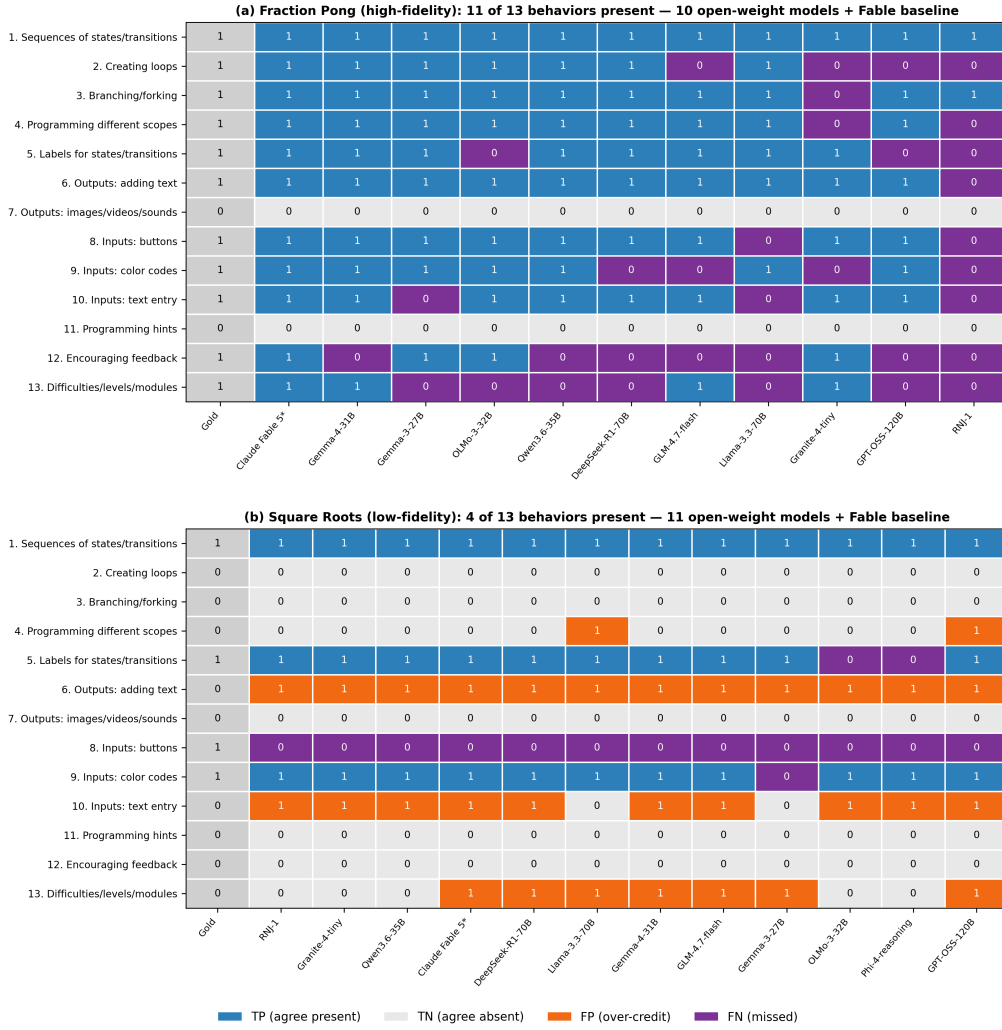


Figure 1: Item-by-model agreement with human gold on the same 13 WL behaviors, models sorted by κ . Top: Fraction Pong (high-fidelity; 11 of 13 behaviors present; 10 open-weight models + Fable baseline). Bottom: Square Roots (low-fidelity; 4 of 13 present; 11 open-weight models + Fable baseline). Orange = over-crediting (FP); purple = missed (FN).

(e.g., Nemotron-3-nano on Square Roots).

A common observation ties these to our instrumentation premise. Very early on, nearly all outputs were rendered as well-formed Markdown tables as requested. They looked authoritative, with consistent columns, plausible language, and a confident tone. Yet, when inspected carefully, the underlying content was frequently wrong, hollow, or just regurgitation from the prompt. We believe a well-structured table of wrong answers would be uninformative, to say the least, for a teacher compared with a broken one because it invites trust. This “rendering–substance” gap was most severe in Phase 1, when a full JSON payload and a single monolithic prompt overwhelmed most of the mod-

els. Reducing the prompt through decomposition and in-context examples led to the quality gains we saw in the later evolution of the pipeline.

8 Implications, Discussion, and Conclusions

Several implications follow. First, several failures are setup- and configuration-related and not likely related to the models’ ability. For example, a rubric schema we created that fuses DLCS codes with descriptions, and a bundled prompt without any in-context examples likely led to task performance degradation and interference. Each is addressable through schema cleanup, task decomposition, and in-context examples. The revised pipeline is being

Pooled confusion matrices vs. human gold (top-right = over-crediting; bottom-left = missed)

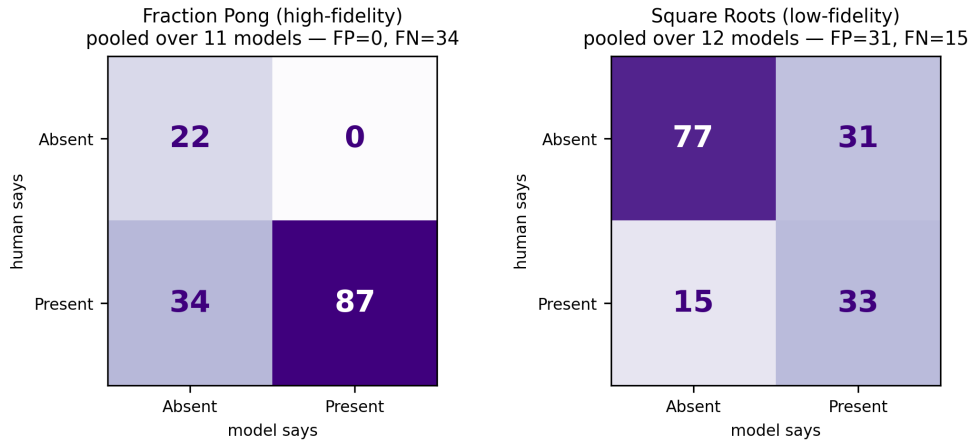


Figure 2: Pooled confusion matrices against human gold. Fraction Pong shows only misses (0 FP, 34 FN); Square Roots is dominated by over-crediting (31 FP, 15 FN). Top-right = over-crediting; bottom-left = missed. Pooled counts are descriptive only.

re-run with additional game designs.

Second, the over-crediting pattern is a problematic teacher-facing risk. Under a sparse design, some of our models credit behaviors the student did not demonstrate. So a “what’s missing” feedback, which depends on correctly detecting absence, would tell a teacher a student did things they actually did not do. On the other hand, for a rich design, the model misses behaviors the student did show, so it fails to recognize real skill and learning. Because detection reliability varies by behavior (Figure 1) and by design fidelity, we think absence-based feedback should be surfaced only for behaviors a model detects reliably. The absence problem fits what is known about LLMs unfortunately. These models tend to affirm what is in front of them and handle negation and absence poorly (Truong et al., 2023). That is why even the frontier model, which found every present behavior in the rich game, credited absent ones in the sparse game. Therefore, we believe the frontier result does not indicate whether open or closed models are better. Instead, it shows that the advantage depends on the design and the raw content. Open-weight models that run locally could still be a good fit for schools once the whole assessment setup, not just the model, has been validated.

Third, our results raise the question of whether different levels of game design call for different models, and whether teachers should be able to choose which model assesses which design. Our data caution against a teacher-facing model picker.

Rankings did not transfer between the two games, so a choice made on one design would not hold on another, and with only two games, the rankings themselves rest on a small number of samples. A model picker would also move a technical burden and decision onto teachers and could invite choosing whichever model gives the preferred answer. Two future alternatives might worth considering. One is to run several models on a game design and treat their agreement as a confidence signal: behaviors that most models agree on are surfaced, and disagreements are flagged for the teacher. The other is to adjust for the design’s density automatically. A design’s basic characteristics (states, transitions, connections) can be counted directly from the JSON before any model runs, so instead of asking a teacher to guess how complex a design is and choose a model to match, let the pipeline measure the complexity itself from the JSON and adjust its own skepticism accordingly.

Fourth, everything reported here assesses a final product. We attempted to assess progression across the two-day WL curriculum but found that the two-day log data need validation before they can serve as indicators of progression. Monitoring progress in a classroom is a natural goal for an LLM-based tool, but our results suggest something else. Progress is a difference between snapshots, so it inherits the error of each snapshots. Yet the most direct measures of progress, changes in states, transitions, connections, and cycles, might not need a model at all. They can be computed determin-

istically from each saved JSON mention previously. A future design would then track structural progress by parsing for these basic characteristics then apply the LLM for interpreting those changes in CT terms and providing teachers with feedback.

Our results so far cover two games with two coders whose disagreements were resolved by discussion. Hence, the number of game designs and CT behaviors is relatively small, so per-model rankings and confusion matrices should be considered as descriptive statistics. Rankings among models are also unstable. They do not carry over between the two games. For example, the RNJ-1 model ranked first on one game design and last on the other. Loop and branch detection can only be tested on the high-fidelity game. Finally, because our two games sit at opposite ends, one very rich and one very sparse, we also cannot generalize our findings yet. More game designs with a balanced number of present and absent behaviors are planned to be tested in the future.

In summary, we think open-weight LLMs can extract and infer some CT behaviors at near-human agreement but are not yet reliable for the counting, loop/branch detection, and standards mapping that trustworthy formative assessment requires. Especially for simple designs, they seem to over-credit and sometimes fabricate evidence. And the tidy format of their output and tables hides errors and regurgitation. Our contributions are (1) a measurement-grounded test harness (F1 and κ for behavior detection, error and element-level analysis for basic characteristics); (2) practical prompt- and rubric-engineering guidance; and (3) evidence, including a frontier baseline, that locally deployable open models can be competitive. This research essentially illustrates asking the question “under what conditions are these open-weight LLMs good enough for assessment?” is better than merely “are open models capable?”

Model	Params	Architecture	Reasoning
DeepSeek-R1-distill-70B	70B	Dense (Llama-3.3 base, R1 distill.)	Yes
Llama-3.3-70B-Instruct	70B	Dense	No
Gemma-3-27B	27B	Dense (vision-capable)	No
Gemma-4-31B	~31B	Dense	No
RNJ-1 (Essential AI)	—	—	—
Nemotron-3-nano (NVIDIA)	Nano	Hybrid Mamba–Transformer	Yes
OLMo-3-32B-think (AllenAI)	32B	Dense	Yes
Phi-4-reasoning-plus	~14B	Dense	Yes
Granite-4-h-tiny (IBM)	Tiny	Hybrid Mamba–Transformer	No
Qwen3.6-35B-A3B	35B / ~3B	Sparse MoE	Yes
GLM-4.7-flash	—	MoE —	—
GPT-OSS-120B (OpenAI)	~120B / ~5B	Sparse MoE	Yes
Claude Fable 5 (baseline)	—	Proprietary	Yes

Table 1: Model roster. Some specifications for recent releases require verification against official model cards.

Model	Fraction Pong (high-fid.)		Square Roots (low-fid.)	
	F1	κ	F1	κ
Gemma-4-31B	0.95	0.75	0.60	0.37
Gemma-3-27B	0.90	0.58	0.50	0.28
OLMo-3-32B	0.90	0.58	0.50	0.28
Qwen3.6-35B	0.90	0.58	0.67	0.49
DeepSeek-R1-70B	0.84	0.45	0.60	0.37
GLM-4.7-flash	0.84	0.45	0.60	0.37
Llama-3.3-70B	0.78	0.35	0.60	0.37
Granite-4-tiny	0.78	0.35	0.67	0.49
GPT-OSS-120B	0.78	0.35	0.55	0.25
RNJ-1	0.31	0.06	0.67	0.49
Nemotron-3-nano	n/r	n/r	n/r	n/r
Phi-4-reasoning	n/r	n/r	0.50	0.28
Claude Fable 5 (baseline)	1.00	1.00	0.60	0.37
Mean	0.80	0.45	0.59	0.37

Table 2: Behavior-detection agreement with human gold (F1 and Cohen’s κ) on both games, scored on the same 13 behaviors against two-coder resolved gold. Open-weight rows sorted by Fraction Pong κ . n/r = non-response.

#	Task	Behaviors covered	Words	Worked example
1	Extract and count basic characteristics from the JSON only	states, transitions, connections, cycles, branches	830	Yes (full extraction output of another game)
2	Mark presence/absence of structural behaviors	sequences; loops; branching/forking	197	Yes
3	Scopes and labels	programming different scopes; labels for states/transitions	208	Yes (state ID + description)
4	Output behaviors	outputs: adding text; outputs: images/videos/sounds	257	Yes (text and media examples)
5	Input behaviors	inputs: buttons; color codes; text entry	297	Yes (transition IDs)
6	Support behaviors	programming hints; encouraging feedback	317	Yes (from other games)
7	Systems-level organization	difficulties/levels/modules	208	Yes (Level 1–3 example)

Table 3: The seven Phase 2 sub-prompts. Each targets a few behaviors and includes a worked example; all but the first also ask for a rendered presence/absence table with justifications from the JSON.

References

- Ivon Arroyo, Avery H. Closser, Francisco Castro, Hannah Smith, Erin Ottmar, and Matthew Micciolo. 2023. [The WearableLearning platform: A computational thinking tool supporting game design and active play](#). *Technology, Knowledge and Learning*, 28(4):1815–1824.
- Jacob Cohen. 1960. [A coefficient of agreement for nominal scales](#). *Educational and Psychological Measurement*, 20(1):37–46.
- Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.
- Mohammad Hadi Nezhad, Francisco Castro, Beverly Woolf, and Ivon Arroyo. 2024. [Math teachers’ in-class information needs and usage for effective design of classroom orchestration tools](#). In *Technology Enhanced Learning for Inclusive and Equitable Quality Education (EC-TEL 2024)*, volume 15159 of *Lecture Notes in Computer Science*, pages 299–314. Springer.
- William Lee and Ivon Arroyo. 2025. [Toward extracting computational thinking evidence with large language models: Empowering K–12 educators](#). In *Artificial Intelligence in Education (AIED 2025)*, volume 2590 of *Communications in Computer and Information Science*. Springer.
- Zhen Li, Yupeng Su, Runming Yang, Congkai Xie, Zheng Wang, Zhongwei Xie, Ngai Wong, and Hongxia Yang. 2025. [Quantization meets reasoning: Exploring LLM low-bit quantization degradation for mathematical reasoning](#). arXiv:2501.03035. Preprint, arXiv:2501.03035.
- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.
- Tinh Hung Truong, Timothy Baldwin, Karin Verspoor, and Trevor Cohn. 2023. [Language models are not naysayers: An analysis of language models on negation benchmarks](#). In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 101–114. Association for Computational Linguistics.
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don’t always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36.
- Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems*, volume 36.
- David Weintrop, Daisy Wise Rutstein, Marie Binkowski, and Steven McGee. 2021. [Assessing computational thinking: An overview of the field](#). *Computer Science Education*, 31(2):113–116.

A Prompts

This appendix shows the single Phase 1 prompt in full and summarizes the seven Phase 2 sub-prompts (also used, automated, in Phase 3). The contrast illustrates the change in task design: one short instruction pointing at an entire rubric spreadsheet, the game JSON, and a two-day log, versus seven focused prompts of 200–830 words, each targeting a few behaviors and each containing a worked example drawn from a real game (Table 3). Student-identifying content has been removed.

A.1 Phase 1: single monolithic prompt (141 words)

You are a 6th grade computer science teacher in Massachusetts. Your job is to use the “CT breakdown in WL -12_02_2025.xlsx” as a rubric to assess the computational thinking knowledge and skills based on a student designed CT game. Column A of the spreadsheet represents various behaviors during the design of a CT game. Columns E and G are Massachusetts CT standards mapped to those behaviors.

The CT game is designed as a finite state automata. The Fraction_Pong_final.json is the finalized version of the CT game. The Fraction_Pong_2_days.csv is a breakdown of the progress of the design of the CT game during a 2-day period.

Using the rubric, assess the CT game complexity with a score between 1 to 5 and provide explanation. Also assess the student progression using the Fraction_Pong_2_days.csv. Identify if the student was making progress and provide rationale.

A.2 Phase 2: seven decomposed sub-prompts

As an example of the decomposed style, sub-prompt 2 reads as follows (example and the rendering instruction included).

Second, with the basic characteristics of the game’s design identified and counted, your next task is to identify the presence or absence of these basic characteristics as behaviors. For example, if there are any occurrences of states, connections, and transitions identified in the design, a behavior called “Creating sequences of states and transitions” would be marked as 1

to indicate the presence of this behavior. Otherwise mark as empty to indicate absence. If there are any occurrences of cycles or loops identified in the design, a behavior called "Creating Loops" would be marked as 1 to indicate the presence of this second behavior. Otherwise mark as empty to indicate absence. If there are any occurrences of branching/forking identified in the design, a behavior called "Branching/Forking" would be marked as 1 to indicate the presence of this third behavior. Otherwise mark as empty to indicate absence.

Also, create and render a table in markdown or HTML format to show the presence or absence of these three behaviors. I must be able to preview the table in a web browser and I do not want it in any raw code. In the table, provide examples and justifications from the "Fraction_Pong_final.json".