The Model Writes, the Service Checks: Olympiad-Style Maths Problems for Primary-School Children inside Chat Assistants

Aleksandr Riazanov

Independent Researcher
altedtech.info@gmail.com

Abstract. Olympiad-style problems ask children to solve what nobody has taught them to solve, and practice needs a steady supply of problems new to each child. A chat assistant can write them, but with a wrong key, two correct options or none, and a child cannot tell a faulty key from their own mistake. We present MathTrail, an open-source service for grades 1–6 that runs inside chat assistants and splits the work: the chat's own model writes every task, and a small deterministic service that calls no model proposes what the child practises next and admits a task only when none of nine checks finds a fault. The central check runs a program the model wrote with the task, which must single out the keyed option of five, again under rotated labels; the answer stays sealed in the child's profile until they answer. A hierarchical Elo model with a guessing floor places a new child with a trial series and aims practice at 70–85 % expected success. Offline, every injected defect that broke a rule a check states was refused, as correct code must, while a second correct option in words and most defects of meaning passed. In simulation, the service estimated a child who stays put as well as any baseline, but most tasks missed the corridor, most declarations of mastery were false, and Glicko-2 followed a learning child far better: 0.24 logits behind, against 1.12. We release the service and its 603 reference tasks, each with its solver.

Keywords: problem generation · verification · learner models · chat assistants

1 Introduction

School mathematics is best at procedures: the National Research Council found students taught by traditional curricula more fluent in procedures than in understanding, strategy or reasoning [23]. Solving a problem one has never seen is a different skill: understanding the problem, devising a plan, carrying it out and looking back [27]. Olympiad-style problems call for exactly this skill. In our working definition, a problem is olympiad-style when its method is not immediately obvious to the solver, its solution rests on an idea rather than a procedure,

Version of 2026-11-07, commit 534aaea3f1ce.

it needs no knowledge beyond the child's grade, and, once the idea is found, the solution is short. Practising them needs a steady supply of problems, each new to the child, a little harder than the last, and still solvable.

Large language models can supply such problems without end, and a chat assistant puts them one message away. What a model writes, though, cannot go to a child unchecked. When models tutored college algebra, only 56.6 % of 150 tutoring dialogues were entirely correct [15], and teachers find some of the word problems models write unsolvable, wrongly answered or unfit for school, even from the strongest models [9,10]. A wrong answer key turns a child's correct solution into a "mistake", and the child cannot tell which of the two is at fault.

We let different parties write and check. The chat's own model, which already talks to the child, writes each task. A small service that makes no model calls of its own proposes what the child should practise, hands the model what it needs, admits the task only after checking everything that can be checked mechanically, shows it without its answer and records the child's answer. We ask:

RQ1. Which defects in a model-written task do the service's checks catch, and which do they miss?

RQ2. Does the learner model place a new child and keep practice in its target corridor, and how often does it declare a topic mastered when it is not?

We contribute an architecture in which the chat's model writes every task and a stateless service that calls no model checks it and seals its answer (Section 3); nine checks, centred on the model's own solver run twice under rotated option labels (Section 4); a learner model for tasks used once, a hierarchical Elo model with a guessing floor on a designed ladder across grades 1–6, with a trial series for placement, related to maximum likelihood (Section 5); an offline evaluation with its negative results (Section 6); and the code and 603 reference tasks under the MIT license.

2 Related Work

Generating and checking problems. Language models have written word problems for kindergarten to grade 8, each answered by a program the model wrote and judged by teachers [9], problems for grades 3–5 [10], and routine exercises for the elementary grades, where a model agreed poorly with people on whether a problem could be solved [2]. Among the works we read, none writes olympiad-style problems for primary school, and none admits a problem by running a program its author wrote against the problem's options. The nearest are GSM-Symbolic, which computes each key with a program people wrote [21]; Beast Academy, olympiad-style problems for ages 6–13 written by people [3]; and MathForces, which generates olympiad quizzes with a model for an audience it does not state, with no check it describes [20]. A model reviewing its own reasoning without outside feedback rarely finds its mistakes [16]; a program can compute the answer instead [8], and a model can check its own answer with code

it writes, reading the result itself [29]. Elsewhere, model agents judge the problems a model rewrote [17]; we take the program route without a second model: the service runs code the model wrote and compares the result with the model's own key.

Learner models and difficulty. Our response model is the logistic model with the guessing floor of Birnbaum's three-parameter model [5], its slope fixed as in Rasch's, updated as the hierarchical Elo rating of adaptive practice: a general level corrected by a topic level, with a step that shrinks with the number of answers [26]. Math Garden estimates ability and item difficulty on the fly and aims at an expected success of 0.75 [19]. Our tasks are never reused, so a task's difficulty cannot be learned from other children's answers: it is asked for, and the model's writing must deliver it. The Bayesian trackers that give the step a principled size, Glicko-2 and Urnings among them, have, as published, no guessing floor [13,6].

Tutors inside chat assistants. Unguarded chat help can harm learning: high-school students who had practised with a plain GPT-4 chat did worse on a later exam than those who never had it [4]. Where tutoring with a model helped, correctness came from outside the model, such as instructors' solutions in the prompt [18]. We found no trial of a model that writes tasks for children of primary age. An open-source MCP tutor separates what the model writes from a rule-driven engine of progression, as we do, but describes no check of what the model writes [14].

3 The System

Each mechanism of the service serves a learning goal, and at the pinned commit all of them are built: every problem new, by a near-duplicate check against reference tasks and the child's past ones; "slightly harder", by a corridor of predicted success of 0.70–0.85; diagnosable mistakes, by a trap of a closed catalog behind each wrong option, explained; the answer not given away, by a seal until the child answers; and a checked key, by the model's own solver run twice.

Who does what. Figure 1 follows one task. A deterministic rule reads the child's profile and builds a brief: the goal, the topic, a grade level and difficulty on one ladder, a setting and the child's two most frequent traps in the topic; the model may ask for something else, but must say why. With the brief comes a package — the corridor, the trap catalog, the skills the task may not use, the child's grade, interests and the parent's notes, three reference tasks, solver templates and a guide — which carries no pseudonym and fits 64 KiB. The model hands in the question, five options, the key, a hint, a solution, each wrong option's trap and explanation, a self-check and a solver; a refused task goes back with every problem found, three attempts a request. The card shows the question, the drawing, the options and the hint; once the child answers, the service unseals the task, marks it, explains a wrong option's mistake and updates the ratings.

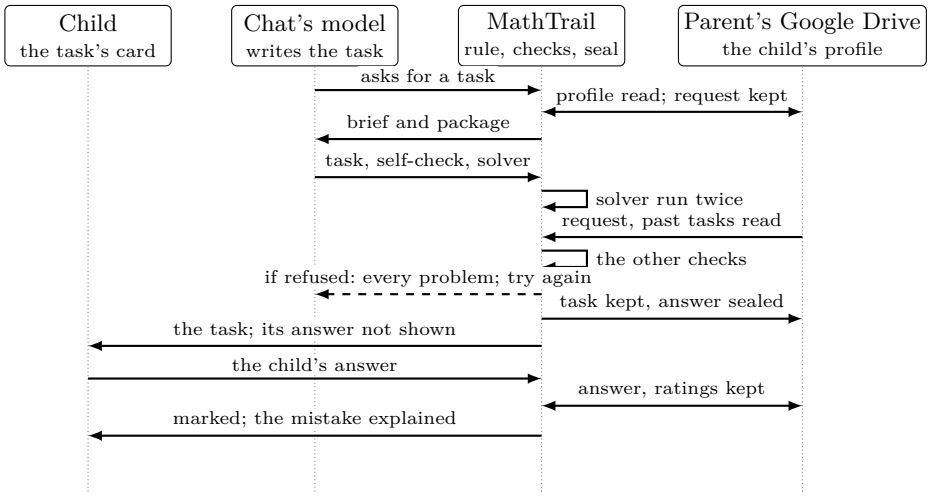


Fig. 1. One accepted task, from request to answer: the model writes it and the service checks it; a refused task goes back with every problem found (dashed). The profile, with the sealed answer, lives in the parent's Google Drive.

Architecture and content. The service is a single Go binary with no database, on Google Cloud Run, that keeps nothing of a child between requests. The profile is a JSON file in the parent's own Google Drive, reached with the parent's token through the service's own OAuth server. It speaks the Model Context Protocol, revision 2026-07-28, without sessions, through seven tools, calls no language model and depends on no model provider's SDK; each card is an MCP Apps page [22] that names no outside domain. The catalogs hold 17 topics, 20 traps — typical mistakes, such as answering a different question — and 25 skills. The 603 reference tasks, 200, 250 and 153 for grades 1–2, 3–4 and 5–6, each have a solver and a trap behind each wrong option. An earlier study of sources chose problem books in the public domain, whose copyright has expired; Claude wrote the tasks from their ideas at our request, in wording of its own, and we checked them. In first runs with a real model, Claude Sonnet 5 wrote 15 tasks over 17 hand-ins, 13 accepted at the first attempt, and a trial series of five tasks went through in Claude's web client. Such runs are too few to measure anything, so our evaluation runs offline.

4 Verification

A submitted task is admitted exactly when none of nine checks finds a problem (Table 1). The solver runs before the child's profile is read, the other checks after; a task that lacks an input some check needs is refused by the structure check in any case. Every problem is reported at once, so the model can fix all of

them in one more attempt, and no message quotes an option, because a refusal reaches the child's card.

Table 1. The checks and what each refuses.

Code	Refuses
`bad_structure`	a task out of format, or one that does not match the brief
`distractor_explanations`	an explanation of a wrong option that cannot tell a child what went wrong
`drawing_format`	a text drawing wider than 30 cells or taller than 12 lines, or with characters fonts draw differently
`drawing_mismatch`	a drawing that does not match its declared structure
`readability`	a question too hard to read for the task's level
`solver_error`	a solver that does not run to an answer
`solver_disagrees`	a solver that finds no option, several, or another than the key; a self-check that reaches another answer
`self_check_blocking`	a task whose own self-check reports a blocking issue
`near_duplicate`	a question that copies a reference task of its level or one of the child's past tasks

The solver, run twice. With each task the model writes a short program in Starlark, a deterministic dialect of Python, that computes the answer from the problem's data and returns the letters of the options whose text means that value. The service runs it in a sandbox limited to 25,000,000 steps and two seconds. The run must single out exactly one option, the keyed one. A program could defeat this by returning a letter written out by hand, so the service runs it again with the option labels moved by two, A becoming C and so on: a program that computes follows its text to the new label, and one that writes out a letter names another text and fails. One that writes out the value still passes.

Near-duplicates and readability. A question is a near-duplicate of another when the Jaccard index of their character trigrams is at least 0.7. It is compared with the reference questions of its level and with the child's last 200 tasks, kept as MinHash sketches [7] of 192 positions cut to four bits. The longest sentence may have 20, 25 or 30 words at grade level 1–2, 3–4 or 5–6, and an English question's Flesch–Kincaid grade may exceed the level's youngest grade by at most three.

What the checks cannot see. An admitted task carries one guarantee about its key: the model's own program, under both labellings, singles out the keyed option and no other. The key is right relative to the model's formalisation of the problem, which is what the checks test, not the meaning of its text. If the model misreads its own question, and the solver computes the misreading and the key agrees, every check passes: a program removes arithmetic errors, not misreadings [8]. The self-check asks what benchmarks of ill-posed problems ask

— a missing condition, two readings, a vague word — but it is the same model's second opinion [16]. In the first local run, the solver check refused once, falsely: the program named an option by part of its text. No check saw two hints that gave too much away, or a question shortened for the readability check that dropped the word the answer rested on.

The answer's path. The answer, the explanations, the solution and the solver are sealed in the profile with XChaCha20-Poly1305, bound to this child and this task; until the child answers, no result returned to the card and no log line, trace or metric holds any of them. Two copies stay outside this path: the model that wrote the task knows its answer, and the host hands the card the arguments of the call that submitted the task, the key and the solution among them.

5 Learner Model and Selection Rule

5.1 The Model

The service predicts the chance that a child answers a task correctly as

$$P = c + (1 - c)\,\sigma(\theta + \delta_t - \beta), \qquad c = 0.2, \qquad \beta = s_\ell + (d - 3),$$

where θ is the child's overall level and δ_t the child's offset for the task's topic t. A task is written for a grade level ℓ — grades 1–2, 3–4 or 5–6 — and a difficulty d from 1 to 5 inside it; the shifts $s_\ell = 0$, 2.5 and 5 put all fifteen points on one ladder; the shift of 2.5 between levels is a guess nothing has measured yet. The floor c is the chance of guessing one of five options. After the trial series (Section 5.4), both levels move Elo-style after each answer $S \in \{0, 1\}$:

$$\theta \leftarrow \theta + K_\theta(S - P), \qquad \delta_t \leftarrow \delta_t + K_\delta(S - P), \qquad K = \frac{K_0}{1 + 0.05\,n},$$

with $K_0 = 0.2$ for θ, n counting the child's answers, and $K_0 = 0.4$ for δ_t, n counting the answers in topic t. "I don't know" counts as wrong. The difficulty β is given, never updated: each task is written for one child and never handed out again.

5.2 The Update and Maximum Likelihood

The log-likelihood of one answer is $\ell = S \ln P + (1 - S) \ln(1 - P)$. With $x = \theta + \delta_t - \beta$, its gradient is

$$\frac{\partial \ell}{\partial x} = (S - P)\,w(P), \qquad w(P) = \frac{P - c}{P\,(1 - c)}.$$

The service's step $K(S - P)$ is the gradient step divided by $w(P)$. When the model is right — every task's true chance of success is the P the model gives it at the child's true levels — the two have the same expected zero, where each

topic's level $\theta + \delta_t$ is the child's true one: dropping w adds no bias and changes only the size of the step. At the corridor's ends, $P = 0.85$ and $P = 0.70$, the service steps 4.6 % and 12 % further than the gradient, and 2.4 times as far at $P = 0.3$: the level moves further, and more noisily, after tasks harder than the corridor. The step shrinks as $1/(1 + 0.05\,n)$, as Robbins and Monro's recursion needs [28]: for one topic and a child who does not change, with every task at $P = 0.775$ on a continuous scale, it converges in probability to where the child's success is 0.775, provided the response curve never falls as tasks get easier and crosses 0.775 with a positive slope. A child who learns is followed ever more slowly, since nothing floors the step.

5.3 The Corridor and Mastery

The target is a predicted success in $[0.70, 0.85]$. Inverting the model, the corridor is an interval of difficulty,

$$\beta \in [\,\theta + \delta_t - 1.4663,\ \theta + \delta_t - 0.5108\,],$$

0.96 wide, narrower than the step of 1 between two difficulties of a level, so it holds at most one of them and sometimes none; where a topic's levels overlap, their points interleave and it may hold two. The brief therefore asks for the topic's point whose P is closest to 0.775, the easier of two equally close. After a wrong answer the rule keeps that answer's topic while it stays within reach; otherwise it takes the unmastered topic within reach that has waited longest. A topic counts as mastered once it has at least five answers and, since its last wrong answer, at least three qualifying ones: correct, without the hint, on a task whose P was at most 0.775 (an eligible task). Two wrong answers in a row take mastery away, and a mastered topic leaves the rotation until every topic within reach is mastered. A streak is not learning: three correct in a row raised a platform's count of mastery but not what students retained [25], so we report mastery as the rule's state. For a child whose true chance is the same on every eligible task, the run is a Markov chain that bounds the chance of a false "mastered" from above: at a true chance of 0.6, it reaches three within 10 eligible attempts with probability 0.70.

5.4 Placement by a Trial Series

The grade the parent enters sets only the start, $\theta_0 = s_\ell$. A child placed a level too high would fail task after task while the step walked down, so the first five answers, each on a new topic, are a trial series, after each of which θ is set to the most likely level given the start and every trial answer:

$$\hat{\theta} = \arg\max_{\theta} \Big[-\frac{(\theta - \theta_0)^2}{2\sigma_0^2} + \sum_i \ln P_i(S_i) \Big], \qquad \sigma_0 = 2.5,$$

where $P_i(S_i)$ is the model's chance of the i-th trial answer at level θ, under a normal prior one level shift wide, its spread chosen by simulation. The topic

offsets stay put during the series, and an early slip is weighed again at every later answer.

6 Evaluation

Both experiments run on the service's own code at the pinned commit, with no child and no language model, under protocols frozen and timestamped before their first runs; every number is produced by a script over saved data, and the deviations from the protocols are recorded with the artifact.

6.1 RQ1: Fault Injection

Design. We inject one defect at a time into reference tasks with their solvers, in classes fixed by the protocol. Those the checks must catch break a rule some check states, such as a wrong key, two correct options or none, a hard-coded solver, a hollow explanation or a copy; a few operators in them make a defect a child would see that no check is defined by. Those out of scope, the X rows of Table 2, measure the limits of Section 4. Apart from the operators that change them, every case keeps its task's reference solver and a self-check that finds nothing, so a wrong key meets a solver that computes rightly: a model's solver that computes its key's error is beyond this experiment. We report the share of cases refused, with 95 % intervals. False refusals of good new tasks would need tasks the checks were never tuned on, and are not measured.

Results. Of the 603 reference tasks, 447 pass every check as they are: all 153 of grades 5–6, but 126 of 200 at grades 1–2 and 168 of 250 at grades 3–4, where the others read too hard for their level or as near-copies of another, mostly in topics whose tasks share a stock opening, such as knights and liars. On these, 68 operators of 27 classes made 23,453 cases. Every case of the 53 operators that break a rule a check states was refused by the check expected: written from the checks' own rules, they test that the checks do what they state, not how far they reach. Every 95 % interval starts at 96.8 % or higher, or at 85 % for the drawing operators, which only the 23 or 24 tasks with a drawing take. All but two of these operators were refused by one check alone, so the checks seldom back each other up.

Table 2 gives the rest. A copy of a reference task with every number raised by one was not reliably caught, refused in 94.6 % of cases, under the 95 % the protocol asks; a repeat of the child's own task with new numbers was refused in 96.1 %, every miss of either in one topic whose short questions are mostly numbers. A second correct option, the right number in words beside its digits, was never refused: the solver matches an option by its text. Out of scope, copies with their people and objects renamed were mostly refused as near-duplicates: a child would read them as copies, while a plot written anew passes by design. The other out-of-scope defects went through, except where a check tripped over the changed text: a removed condition was refused in 7.1 % of cases, by the readability check or the drawing check.

Table 2. Defects that no check is defined by (D) and those out of scope (X): cases refused by any check, with exact 95 % intervals. All 18,383 cases of the 53 operators that break a rule a check states were refused by the check expected.

	Defect	Cases	Refused, %
D02b	a second correct option: the right number in words beside its digits	213	0.0 (0.0–1.7)
D14b	a copy of a reference task with every number raised by one	332	94.6 (91.6–96.8)
D15b	a repeat of the child's own task with new numbers	357	96.1 (93.5–97.8)
X01a	an exact quantity made vague	366	2.7 (1.3–5.0)
X01b	an exact quantity made a range	357	1.7 (0.6–3.6)
X02	a condition removed	325	7.1 (4.5–10.4)
X03	a word of the question swapped for its opposite	206	1.0 (0.1–3.5)
X04a	two wrong options' traps swapped	426	0.0 (0.0–0.9)
X04b	a trap replaced by another of the catalog	447	0.0 (0.0–0.8)
X05	a hint that gives the answer away	447	0.0 (0.0–0.8)
X06	the pseudonym in the text	447	3.4 (1.9–5.5)
X07	a well-formed task of another topic	447	0.0 (0.0–0.8)
X08	a well-formed task two or more difficulties away	396	0.0 (0.0–0.9)
X09a	a copy with its people and objects renamed	177	95.5 (91.3–98.0)
X09b	the same, with new numbers as well	127	88.2 (81.3–93.2)

6.2 RQ2: Simulated Learners

Design. Simulated children go through the service's own rule, profile and update for 200 answers each. A child has a true level, drawn around the start of a random grade with a spread of 1 logit, and a true offset in every topic; a written task's difficulty misses the one asked for by a normal error of standard deviation 0.5. Nine generators of 1,000 children take this population as it is or change it in one respect: a grade level off, learning and forgetting, a jump, a host writing harder tasks, linked topics, another slope or floor, hints. Fourteen alternatives run beside the service's update: its step with one level per child or per topic; a constant step and the likelihood-gradient step, with one level per child, per topic, or level plus offsets; Glicko-2 [13], Glicko-2 with the floor, and Urnings [6], each with one level per child or per topic. All rules meet the same children with the same random draws and the service's choice of tasks and mastery; the step rules keep the trial's placement, the others estimate from the first answer. A declaration of mastery is false when the child's true chance on the level's middle task was below 0.775. Each number is a mean or pooled share over a cell's children, with a 95 % bootstrap interval; a paired difference is found when its interval excludes zero. Four comparisons were fixed as primary before the run: the service against each baseline on children who stay put and on learners, the trial series on children placed a level off, and floors under the step on a jump; every other number is secondary.

Results. On children who stay put, the service's estimate and the gradient step in the same structure were the most accurate of the fifteen rules after 200 answers, with root mean square errors of 0.495 and 0.493 logits and no difference found between them (paired difference −0.004 to 0.008), against 0.58–0.83 for the others. The service's predicted chances were calibrated on these children, with an expected calibration error of 0.004, and less so on learners, at 0.079. Under every rule most tasks missed the corridor (Fig. 2), which is 0.96 logits wide while the points a topic can be set at stand up to 1 logit apart and the writing and the estimate err besides; the service kept 42 % of its tasks inside.

When the child changed, the service fell behind. With a level that grows by 0.01 logits an answer, its estimate stood 1.12 logits below the truth over answers 101–200, and it kept 26 % of its tasks in the corridor. Six baselines lagged less, 0.24–0.80 logits, and kept 31–43 % in the corridor — Glicko-2, its floored variant, Urnings and the constant step, each with one level per child, the constant step with level plus offsets, and Glicko-2 per topic; the other eight lagged 1.13–1.48. After a jump of one logit, 71 % of children were not caught up within the run, against 5 % under Glicko-2. After 200 answers, Glicko-2 with one level per child also had the lower error under a harder host and another slope, 0.61 and 0.66 against the service's 0.66 and 0.74, and the service the lower one under a level off, linked topics, another floor and hints. The largest floor under the step left 56 % of children not caught up after a jump, against 71 %, and shortened the mean catch-up, a child never caught up counted at the run's end, by 7.1 answers (6.0–8.3), at no cost found on children who stay put; the smaller floors left the catch-up as it was.

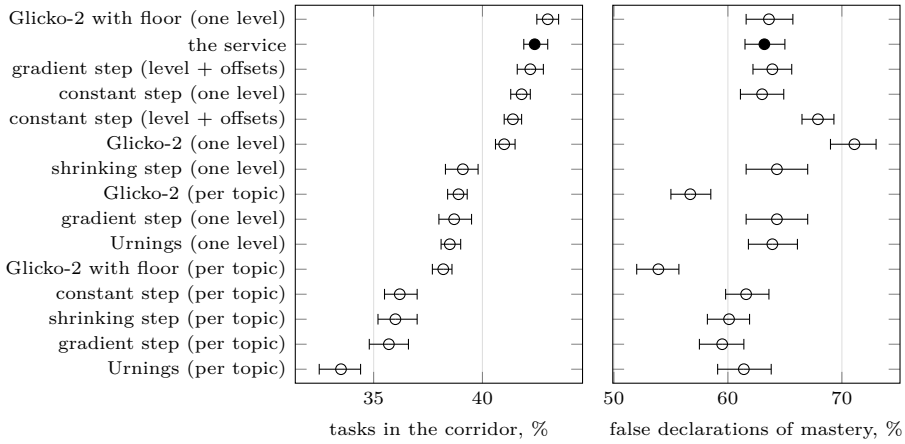


Fig. 2. The fifteen rules on children who stay put: tasks whose true chance fell in the corridor (left), and declarations of mastery made while the child's true chance on the level's middle task was below 0.775 (right), with 95 % intervals; the service is filled.

Mastery was the weak part. Of the service's declarations of mastery, 63 % were false (53 % against the corridor's floor rather than its middle), and under the other rules, which share its mastery rule, 54–71 %: no other estimate made most declarations true. Of the children and topics not truly mastered, 44 % were declared mastered within their first 5 eligible attempts, by a Kaplan–Meier estimate, while 22 % of the topics truly mastered were never declared. For children placed a grade level off, the kind its spread was chosen on, the trial series stood 0.94 logits from the child's level after 5 answers, against 2.24 for the update from the first answer and 1.07 for Glicko-2, yet left the longest run of wrong answers as it was (2.02 against 2.03). Children whose grade placed them well paid for the series, 0.87 against 0.72 without it, and still 0.495 against 0.470 in root mean square error after 200 answers.

6.3 Cost and Threats to Validity

The service makes no model calls, so a task costs it only computation: on a development machine, a review of a reference task took a median of 3.45 ms (26.8 ms at the 99th percentile), and the package the model writes from weighed 17.2–25.2 KiB. A task's time is the model's writing.

Internal: our injected defects may be easier to catch than those models make, the harness ran outside any conversation, and the simulated children answer by the very curve the service assumes, which favours its update. *External:* the reference tasks come from one model family and are in English, on which the readability limits were calibrated; simulated children are not children; the cost was measured on one machine. *Construct:* catching an injected defect is not catching a real one, an operator can make text that is not the defect it is named after, and no check asks whether a task is olympiad-style or at the difficulty asked for. *Conclusion:* each of the 80 paired differences in the four primary comparisons was judged by its own interval, with no correction for their number, and 65 found a difference, a few perhaps by chance. *Search:* our claims of novelty rest on searches in English.

7 Limitations

Children have used the service in real lessons, but its monthly public totals are still to come, and nothing here measures learning; in the trials of model tutoring we read, gains came with correctness from outside the model [18]; whether automated checks can take that place for a child is untested. The seal hides the answer from the child, not from the model, an adult reading the host's tool log, or whoever inspects the card's memory, and what the model writes in the chat is not checked. Repeats are textual and remembered for the last 200 tasks only, and the reference tasks are in English, so a translated copy is not caught. Difficulty is asked for, not measured. A false declaration of mastery is undone by two wrong answers in a row, but a mastered topic leaves the rotation, so the correction waits until the topic returns. Most of the card's 22 languages are machine translations nobody has read.

8 Discussion, Ethics and Conclusion

The design gives each party what it does well: the model writes a fresh problem for this child, in the child's language and a setting the child likes; the service decides what to practise and checks the key and the repeats. The self-check is the model's opinion of its own work, and the solver its own formalisation, which the service runs and judges. Every defect that breaks a rule a check states was caught, while a question turned round under its unchanged solver, as a misreading the solver shares would leave it, and the same answer in words went through, and copies with new numbers were not reliably caught. The learner model needs the most change: it estimates a child who stays put as well as the best baseline, but its mastery rule declares mastery the child does not have, and its shrinking step leaves a child who learns or jumps behind, where Glicko-2 keeps up and a floor helps only in part; a step that does not shrink away and a stricter rule are future work.

Ethics. The children who use the service took part in no study. The service learns no name, birth date or school: it knows a child by a pseudonym and a random identifier, and a parent in its logs by a keyed hash of their account, beside the IP addresses the platform logs; as a pseudonym does not make data anonymous [11], the design keeps as little as it can: the profile lives in the parent's own Google Drive and the service keeps no copy, task texts, answers and the pseudonym stay out of logs and metrics, the service trains no model, and its policy promises no advertising. The child's grade and interests and the parent's notes do reach the chat's model with every task. We leave open whether COPPA or Article 8 of the GDPR applies [12,11]. Neither host admits a child of primary age as a user: Claude's users must be at least 18, ChatGPT's at least 13 [1,24]. The design assumes an adult beside the child who runs the lesson, as the service's terms say; on the card the child answers, and what the child types into the chat goes to a model whose replies no check covers. A study with children would need an ethics board's review, the parent's permission and the child's assent.

Cost and openness. The service calls no model, so every task is written within the family's own use of the chat and the operator pays for computation alone, and it speaks only the open protocol, tied to no host's model. The code and the reference tasks are open under the MIT license at `https://github.com/MathTrail/mathtrail-standalone`, and the experiments' code, data and seeds and the evidence ledger are archived at [**TBD: K11: the artifact's DOI**].

Conclusion. Letting a chat model write for a child is safe only as far as what it writes is checked. We built a service that admits a model's task only after a program the model wrote has agreed with its key under both labellings and keeps the answer sealed in the child's profile until the child answers, and we measured what its checks catch and what they cannot. Acceptance in real hosts, generation by other model families, a stricter mastery rule and a study with children come next.

Acknowledgments. This work received no external funding.

Disclosure of Interests. The author has no competing interests to declare that are relevant to the content of this article.

References

1. Anthropic: Consumer terms of service. Effective October 8, 2025 (2025), https://www.anthropic.com/legal/consumer-terms
2. Ariyarathne, N., Bandara, H., Heshan, Y., Gamage, O., Ranathunga, S., Nayanajith, D., Sivapalan, Y., Lihinikaduarachchi, G., Vihidun, T., Chandirakumar, M., Premakumar, S., Gathsara, S.: Elementary math word problem generation using large language models (2025). https://doi.org/10.48550/arxiv.2506.05950
3. Art of Problem Solving: Beast academy. Product pages, undated (2026), https://beastacademy.com/
4. Bastani, H., Bastani, O., Sungu, A., Ge, H., Kabakcı, Ö., Mariman, R.: Generative AI without guardrails can harm learning: Evidence from high school mathematics. Proceedings of the National Academy of Sciences **122**(26) (2025). https://doi.org/10.1073/pnas.2422633122
5. Birnbaum, A.: Some latent trait models and their use in inferring an examinee's ability. In: Lord, F.M., Novick, M.R. (eds.) Statistical Theories of Mental Test Scores, chap. 17, pp. 397–424. Addison-Wesley, Reading, MA (1968), https://faculty.ucmerced.edu/jvevea/classes/290_21/readings/week%209/Birnbaum.pdf
6. Bolsinova, M., Maris, G., Hofman, A.D., van der Maas, H.L.J., Brinkhuis, M.J.S.: Urnings: A new method for tracking dynamically changing parameters in paired comparison systems. Journal of the Royal Statistical Society Series C: Applied Statistics **71**(1), 91–118 (2022). https://doi.org/10.1111/rssc.12523
7. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations. Journal of Computer and System Sciences **60**(3), 630–659 (2000). https://doi.org/10.1006/jcss.1999.1690
8. Chen, W., Ma, X., Wang, X., Cohen, W.W.: Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks (2022). https://doi.org/10.48550/arxiv.2211.12588
9. Christ, B.R., Kropko, J., Hartvigsen, T.: MATHWELL: generating educational math word problems using teacher annotations. In: Findings of the Association for Computational Linguistics: EMNLP 2024. pp. 11914–11938. Association for Computational Linguistics (2024). https://doi.org/10.18653/v1/2024.findings-emnlp.696
10. Christ, B.R., Molitz, P., LeBlond, B., Gottesman, Z., Kropko, J., Hartvigsen, T.: EDUMATH: generating standards-aligned educational math word problems. In: Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 42611–42643. Association for Computational Linguistics (2026). https://doi.org/10.18653/v1/2026.acl-long.1973
11. European Parliament and Council of the European Union: Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). Official Journal of the European Union, L 119, 4 May 2016, pp. 1–88 (2016), https://eur-lex.europa.eu/eli/reg/2016/679/oj

12. Federal Trade Commission: Children's online privacy protection rule (COPPA rule), 16 CFR part 312. Electronic Code of Federal Regulations, as amended at 90 FR 16977, April 22, 2025; the text in force on 2026-09-28 (2025), https://www.ecfr.gov/current/title-16/chapter-I/subchapter-C/part-312
13. Glickman, M.E.: Example of the Glicko-2 system. The author's description of the rating system, dated March 22, 2022 (2022), https://www.glicko.net/glicko/glicko2.pdf
14. Guiovanna, A.: Tutor MCP. GitHub repository README, MIT licence, alpha; repository created 2026-03-30 (2026), https://github.com/ArnaudGuiovanna/tutor-mcp
15. Gupta, A., Reddig, J., Calò, T., Weitekamp, D., MacLellan, C.J.: Beyond final answers: Evaluating large language models for math tutoring. In: Artificial Intelligence in Education, pp. 323–337. Lecture Notes in Computer Science, Springer Nature Switzerland (2025). https://doi.org/10.1007/978-3-031-98414-3_23
16. Huang, J., Chen, X., Mishra, S., Zheng, H.S., Yu, A.W., Song, X., Zhou, D.: Large language models cannot self-correct reasoning yet (2023). https://doi.org/10.48550/arxiv.2310.01798
17. Ikram, F., Kumar, N.A., Lu, J., McNichols, H., Walkington, C., Heffernan, N., Lan, A.S.: A multi-agent approach to validate and refine LLM-Generated personalized math problems. In: Artificial Intelligence in Education, pp. 233–247. Lecture Notes in Computer Science, Springer Nature Switzerland (2026). https://doi.org/10.1007/978-3-032-29744-0_16
18. Kestin, G., Miller, K., Klales, A., Milbourne, T., Ponti, G.: AI tutoring outperforms in-class active learning: an RCT introducing a novel research-based design in an authentic educational setting. Scientific Reports **15**(1) (2025). https://doi.org/10.1038/s41598-025-97652-6
19. Klinkenberg, S., Straatemeier, M., van der Maas, H.: Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation. Computers & Education **57**(2), 1813–1824 (2011). https://doi.org/10.1016/j.compedu.2011.02.003
20. MathForces: Mathforces: olympiads. Product page, undated (2026), https://mathforces.com/olympiads/
21. Mirzadeh, I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., Farajtabar, M.: GSM-Symbolic: understanding the limitations of mathematical reasoning in large language models (2024). https://doi.org/10.48550/arxiv.2410.05229
22. Model Context Protocol: MCP apps: Interactive user interfaces for MCP (SEP-1865), stable release 2026-01-26. Extension specification (2026), https://github.com/modelcontextprotocol/ext-apps/blob/298e884/specification/2026-01-26/apps.mdx
23. National Research Council: Adding It Up: Helping Children Learn Mathematics. National Academies Press (2001). https://doi.org/10.17226/9822
24. OpenAI: Terms of use. Effective January 1, 2026; read from the Internet Archive's capture of 2026-09-29 (2026), https://openai.com/policies/terms-of-use/
25. Oreopoulos, P., Liut, M., Sungu, A., Low, N.: Making AI tutoring productive: Evidence from a mastery-based math practice experiment (2026). https://doi.org/10.26300/01qv-6c22
26. Pelánek, R., Papoušek, J., Řihák, J., Stanislav, V., Nižnan, J.: Elo-based learner modeling for the adaptive practice of facts. User Modeling and User-Adapted Interaction **27**(1), 89–118 (2017). https://doi.org/10.1007/s11257-016-9185-7
27. Pólya, G.: How to Solve It: A New Aspect of Mathematical Method. Princeton University Press (2014). https://doi.org/10.2307/j.ctvc773pk

28. Robbins, H., Monro, S.: A stochastic approximation method. The Annals of Mathematical Statistics **22**(3), 400–407 (1951). https://doi.org/10.1214/aoms/1177729586
29. Zhou, A., Wang, K., Lu, Z., Shi, W., Luo, S., Qin, Z., Lu, S., Jia, A., Song, L., Zhan, M., Li, H.: Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification (2023). https://doi.org/10.48550/arxiv.2308.07921