Toward In-Context Teaching: Adapting Examples to Students' Misconceptions

Alexis Ross **Jacob Andreas**
MIT CSAIL
{alexisro,jda}@mit.edu

Abstract

When a teacher provides examples for a student to study, these examples must be *informative*, enabling a student to progress from their current state toward a target concept or skill. Good teachers must therefore simultaneously infer what students already know and adapt their teaching to students' changing state of knowledge. There is increasing interest in using computational models, particularly large language models, as pedagogical tools. As students, language models in particular have shown a remarkable ability to adapt to new tasks given small numbers of examples. But how effectively can these models *adapt as teachers* to students of different types? To study this question, we introduce a suite of models and evaluation methods we call ADAPT. ADAPT has two components: (1) a collection of simulated Bayesian student models that can be used for evaluation of automated teaching methods; (2) a platform for evaluation with human students, to characterize the real-world effectiveness of these methods. We additionally introduce (3) ATOM, a new probabilistic method for adaptive teaching that jointly infers students' past beliefs and optimizes for the correctness of future beliefs. In evaluations of simulated students across three learning domains (fraction arithmetic, English morphology, function learning), ATOM systematically outperforms LLM-based and standard Bayesian teaching methods. In human experiments, both ATOM and LLMs outperform non-adaptive random example selection. Our results highlight both the difficulty of the adaptive teaching task and the potential of learned adaptive methods for solving it.

1 Introduction

Good teaching is *adaptive* to students' specific beliefs and preconceptions (Corbett, 2001). Imagine, for example, that you have been tasked with tutoring students fraction arithmetic. You may start by first probing a student's understanding, asking

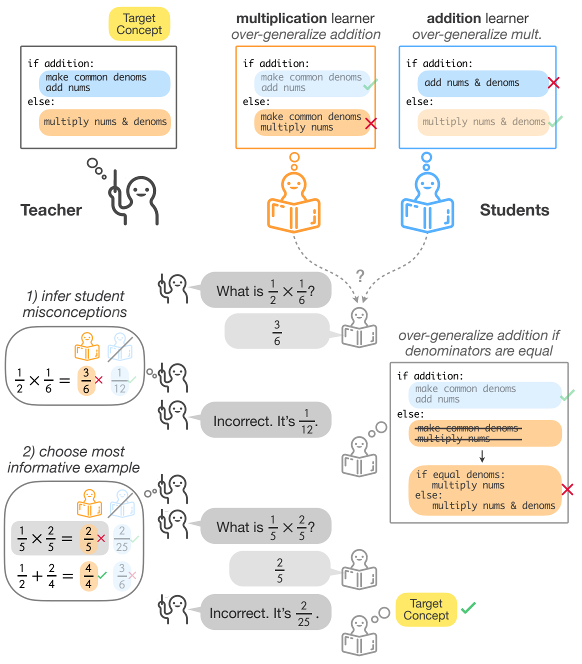


Figure 1: In the **ADAPT (Adaptive Teaching) evaluation framework** (§3), a teacher selects examples to teach a target concept to a student; however, the student has prior misconceptions that are *unknown* to the teacher. In the fraction arithmetic task (§3.1), some students (*multiplication learner*) tend to over-generalize the addition procedure of making common denominators and performing arithmetic only on numerators; others (*addition learner*) tend to over-generalize the multiplication procedure of applying arithmetic on both numerators and denominators. In order to teach effectively, the teacher must jointly 1) *infer* the student's misconceptions online by observing their behavior throughout their interaction (*i.e.,* the teacher infers that the student is a *multiplication learner* after observing the prediction $\frac{1}{2} \times \frac{1}{6} \rightarrow \frac{3}{6}$), and 2) *adapt* to such misconceptions by selecting examples that will most efficiently correct these misconceptions (*i.e.,* the teacher anticipates the student's new incorrect belief that all fractions with equal denominators should be treated as addition problems and selects the example $\frac{1}{5} \times \frac{2}{5} = \frac{2}{25}$ to correct it). We propose **ATOM**, a two-part probabilistic approach that achieves adaptive teaching by maintaining explicit inferences about student priors (§4.1).

them what $\frac{1}{5} \times \frac{2}{5}$ is. Suppose the student answers $\frac{2}{5}$. Immediately, you might develop a hypothesis about this student's misconceptions: they seem to be over-generalizing the rule for *addition*, only applying the operation to the numerator. Now suppose another student *correctly* answers $\frac{1}{5} \times \frac{2}{5} = \frac{2}{25}$, but answers $\frac{1}{2} + \frac{2}{4} = \frac{3}{6}$. This student would appear to be over-generalizing the rule for *multiplication*. These (discrete and systematic) categories of student misconceptions have been found to be widespread among real students learning fraction arithmetic (Braithwaite et al., 2017).

As this example highlights, interactions with students reveal insights about their misconceptions, and these misconceptions in turn influence the course of effective teaching. A good teacher should provide different problems for these students to target their specific misconceptions: The addition-generalizer would benefit from multiplication examples, especially those with common denominators, while the multiplication-generalizer would benefit from addition examples.

What does this mean for NLP? Computational models—particularly language models (LMs)—are increasingly used as pedagogical tools (Kasneci et al., 2023). But it is unclear how effectively any of today's models can tailor instruction to perform "in-context teaching" (§2) for students with differing degrees of skill and prior knowledge. In this work, we draw on a long line of literature on rational models of pedagogy (Shafto et al., 2014) to study this question.

To do so, we introduce **ADAPT (Adaptive Teaching)**, an evaluation suite targeted at teaching students with varied prior misconceptions (§3). In ADAPT, a teacher is tasked with selecting examples to teach a particular target concept. As shown in Figure 1, the teacher selects examples one by one and can observe predictions made by the student. Importantly, the student has prior misconceptions that are unknown to the teacher. ADAPT is designed such that correct inferences about student misconceptions can enable more efficient learning. ADAPT has two components:

1. An **offline, probabilistic framework** for reproducibly evaluating how efficiently teachers can teach **simulated students** with unknown prior misconceptions (§3.2).

2. An evaluation platform for measuring teachers' efficacy with **human students** with prior misconceptions (§3.3).

Simulated and human experiments allow us to characterize the pedagogical capabilities of teachers along several dimensions: their inferences about student beliefs (§5.3, §6.3), adaptivity of chosen examples (§5.4), and differences in teaching mathematical concepts (§5.5). In addition to these evaluations, we introduce:

3. A new probabilistic method, **ATOM (Adaptive Teaching tOwards Misconceptions)**, which performs *online inference of student priors*, then uses these inferences to select informative teaching examples. ATOM provides proof-of-viability for adaptive teaching methods in simulated and human students (§4.1).

Using ADAPT evaluations, we characterize ATOM, GPT-4, and a range of other methods. In simulated students, we find that while GPT-4 exhibits some adaptation to student misconceptions, there is room for improvement with adaptive approaches; it substantially underperforms ATOM, suggesting promise in using adaptive methods (§5). In human experiments, however, both ATOM and GPT-4 outperform random example selection, highlighting the potential of learned models (of various kinds) for adaptive teaching (§6).[1]

2 Preliminaries

We formulate our problem setting as one in which a **teacher** interactively provides feedback to a **student** to communicate a new concept (a mapping from inputs x to outputs y). A teacher observes a sequence of $(x, \hat{y})$ pairs guessed by a student, and must infer what additional training example (x, y) pair will most improve the student's understanding of the concept. We term this problem **in-context teaching** to draw an analogy to the widely studied phenomenon of "in-context learning" (Brown et al., 2020; Min et al., 2022; Akyürek et al., 2023). Formally:

- The teacher begins with a target **concept** h^* drawn from some concept space $\mathcal{H}$. A concept parameterizes a mapping between an input space $\mathcal{X}$ and an output space $\mathcal{Y}$. In Fig. 1, h is the true procedure for adding and multiplying fractions, and $\mathcal{H}$ is the space of other possible fraction manipulation algorithms, so $\mathcal{X}$ contains arithmetic expressions involving fractions, and $\mathcal{Y}$ contains fractions.

[1]Our code is publicly available at https://github.com/alexisjihyeross/adaptive_teaching.

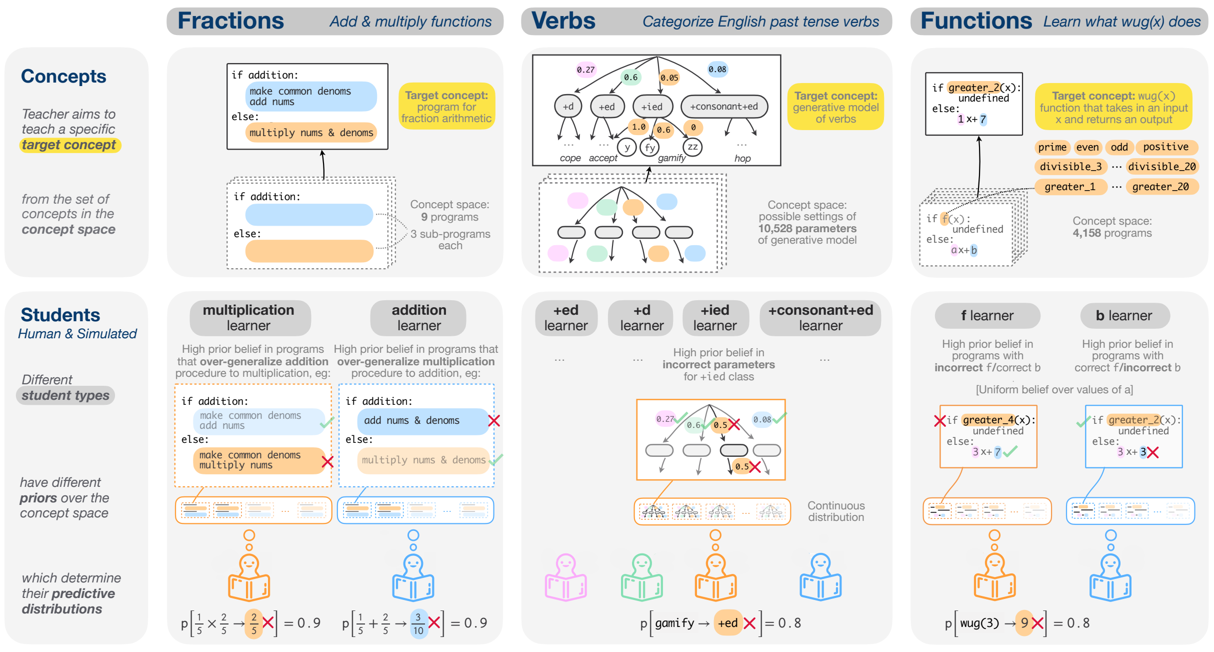

Fractions
Add & multiply functions
Verbs
Categorize English past tense verbs
Functions
Learn what wug(x) does
Concepts
Teacher aims to teach a specific target concept
from the set of concepts in the concept space
Target concept: program for fraction arithmetic
Concept space: 9 programs
3 sub-programs each
Target concept: generative model of verbs
Concept space: possible settings of 10,528 parameters of generative model
Target concept: wug(x) function that takes in an input x and returns an output
Concept space: 4,158 programs
Students
Human & Simulated
Different student types
have different priors over the concept space
which determine their predictive distributions
multiplication learner
addition learner
High prior belief in programs that over-generalize addition procedure to multiplication, eg:
High prior belief in programs that over-generalize multiplication procedure to addition, eg:
+ed learner
+d learner
+ied learner
+consonant+ed learner
High prior belief in incorrect parameters for +ied class
Continuous distribution
f learner
b learner
High prior belief in programs with incorrect f/correct b
High prior belief in programs with correct f/incorrect b
[Uniform belief over values of a]

Figure 2: An overview of the tasks and student types in the ADAPT (Adaptive Teaching) evaluation framework (§3). ADAPT has three tasks: fractions, verbs, and functions. For the fraction and function tasks, a student's concept space consists of *programs*; for the verbs task, a student's concept space is the space of *generative models* corresponding to English past tense verb classes.

- We assume that teachers interact with students by providing **examples** $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. For convenience, we denote a sequence of such examples $(\underline{x}, \underline{y}) = [(x_1, y_1), \ldots, (x_n, y_n)]$. In each round of teaching, the teacher first presents the student with an input x_i, the student produces a guess $\hat{y}_i$, and then the teacher reveals the true y_i.[2]

- Given a collection of examples $(\underline{x}, \underline{y})$, we assume that students compute a *posterior* over concepts $p_S(h \mid \underline{x}, \underline{y})$. For example, a student who has just seen that $\frac{1}{3} \times \frac{2}{3} = \frac{2}{9}$ may be less likely to believe that fraction addition and multiplication follow the same rules. The process by which students infer concepts from examples can in principle be arbitrary; however, for some methods in this paper we will assume that students are Bayesian, with:

$$p_S(h \mid \underline{x}, \underline{y}) \propto p_S(h) \prod_i p_S(y_i \mid x_i, h) \quad (1)$$

under some **prior** belief $p_S(h)$ about the concept space.

[2]This is both a model of real-world educational practice and a standard paradigm for online learning; future work might study richer forms of interaction with explanations and instructions.

Given this setup, a teaching strategy is a **policy** $p_T(x, y \mid \underline{x}, \underline{y})$ that chooses examples to maximize the probability that the student assigns to h^*. In the **optimal teaching (OT)** framework of Shafto et al. (2014), for example, $p_S(h \mid \underline{x}, \underline{y})$ is assumed known, and teachers choose examples:

$$x_{i+1}, y_{i+1} = \arg\max_{x,y}\ p(h^* \mid \underline{x}, \underline{y}, x, y) \quad (2)$$

In the running example, this criterion is more likely to select examples of addition for a student who has already mastered multiplication, and vice-versa. In addition to this greedy approach, it is possible to *plan* sequences of informative examples for students (Rafferty et al., 2016).

As discussed in §1, however, the assumption that teachers have exact knowledge of $p_S(h \mid \underline{x}, \underline{y})$ is often unrealistic—real-world teaching involves students of many different types, whose beliefs and misconceptions may not be known (or easily discerned) *a priori*. Thus, we study teaching when students' priors are themselves unknown. We assume that students are drawn from a distribution over **student types** $p(\alpha)$, each associated with a concept prior $p(h \mid \alpha)$. In the running example, these priors may correspond to different beliefs about the algorithms for fraction addition and multiplication, with "addition generalizers" assigning high probability to a spurious multiplication algorithm that manipulates only numerators.

In this setting, teachers must still implement an effective example selection policy $p_T(x, y \mid \underline{x}, \underline{y})$; however, choosing informative examples now requires *inferring* student priors in order to estimate the effect of these examples on $p_S(h^* \mid \underline{x}, \underline{y})$. In the next section, we describe our proposed framework for evaluating adaptive teaching policies. In §4, we describe a set of candidate adaptive teaching policies (including our new ATOM method), and in §5 and §6 use ADAPT to evaluate these teaching policies with simulated and human students.

3 The ADAPT Evaluation Framework

ADAPT has two parts: an offline evaluation framework with simulated students (§3.2), and a platform for doing experiments with human students (§3.3). We first describe the tasks in ADAPT (§3.1). An overview of ADAPT is shown in Figure 2.

3.1 Tasks

Fractions In the first task, the teacher aims to teach the student how to add and multiply fractions. Here, student types correspond to different prior beliefs (possibly incorrect) about the rules for fraction arithmetic.

Verbs The second task is English past-tense conjugation. In this task, students are presented with lemmas and must choose an appropriate ending (*e.g., play* → *+ed*, fry → *+ied*). Here, student types correspond to different degrees of familiarity with possible English past-tense endings.

Functions In the third task, reminiscent of existing number concept learning tasks (Tenenbaum, 1999), the teacher aims to teach the student a function that takes in numbers and returns either numbers or *undefined*. These functions can be represented as *programs* that take an input x and compute:

```
if f(x): return undefined      (3)
   else: return a*x+b
```

where f is a boolean function and a, b are integers. The teacher chooses input/output pairs (x, wug(x)) to show the student to maximize the student's belief that the concept is the correct program h^*. We create 24 target concepts, which combine 3 unique settings of a/b and 8 settings of f. Student types can be instantiated by selecting preferences for specific primitives (*e.g.,* f, a, b); arbitrary priors over programs can then be derived from these preferences for primitives.

3.2 Simulated Students

The first component of ADAPT evaluates teachers with simulated, Bayesian students. These students maintain belief distributions over the full concept space.[3] As shown in Figure 2, different "student types" are implemented by initializing students with different priors over the concept space. All student types begin with low initial belief in the target concept h^* and assign high probability to other spurious concepts.

Fractions For the fraction task, we represent understanding of fraction arithmetic as programs. Students maintain a belief distribution over the space of possible programs, as shown in Figure 2. We create two student types, **`mult-learner`** (a model of a student who has not yet mastered multiplication and incorrectly applies the procedure for addition to multiplication) and **`add-learner`** (a student who performs addition by incorrectly applying the procedure for multiplication). These correspond to common incorrect strategies that children exhibit when learning fraction arithmetic (Braithwaite et al., 2017) by *over-generalizing* the procedure for one operation to another.

Functions For the function task, students again maintain a belief distribution over the space of possible concepts. We create two types of students for each target concept: a `b-learner` and a `f-learner`. The **`f-learner`** knows the true value of b but has an incorrect, spurious belief about what function `f(x)` is; the **`b-learner`** knows the true `f(x)` in the target program h^* but has an incorrect belief about the value of b. See §B for how we select incorrect beliefs for students.

Verbs For the verb task, we represent understanding of verb conjugation as *generative models* of English past-tense verbs. Students are naïve Bayes models with features for word-final character n-grams, so $p(h \mid \underline{x}, \underline{y})$ is a distribution over model parameters, with Dirichlet/Beta priors over the verb class/feature occurrence parameters, respectively. We fit a naïve Bayes model on the Unimorph

[3]The concept space consists of 9 concepts for the fraction task, 4,158 concepts for the function task, and a continuous space of possible values for 10,528 parameters (corresponding to 329 features and 4 verb classes) for the verb task.

dataset[4] (Batsuren et al., 2022) and use the mode of the resulting posterior as the target concept.[5]

We create four student types by picking one of the classes as the "unknown" class: a **`+d-learner`**, **`+ed-learner`**, **`+consonant+ed-learner`**, and a **`+ied-learner`**. To simulate students who are familiar with all but one class, we initialize the student's priors by using the posterior mode parameters of the model fit on the full data, setting the parameters for the "unknown" class to all 1s (effectively removing any learned information about the class): Figure 2 shows how setting the prior in this way determines the mode of the `+ied-learner`'s prior distribution over generative models.

3.3 Human Students

The second component of the ADAPT evaluation framework is a platform for evaluating adaptive teaching with human students, specifically for the function learning task. Human participants are tasked with learning what a "mystery machine" called `wug` does. They are given 10 minutes to interact with a teacher who presents teaching examples through a chat interface. Their task is to figure out when `wug(x)` is undefined (*i.e.,* guess what `f` is), and when `wug(x)` is defined, what it computes (*i.e.,* what `a` and `b` are in `a*x+b`). They can submit guesses for how `wug(x)` operates whenever and however many times they choose to during the 10 minutes of interaction. `wug` guesses have 3 components corresponding to `f`, `a`, and `b`, and we allow partial guesses. See §6.1 for more details on instructions, bonus compensation, and other parts of the human study.

We create `b-learners` and `f-learners` by priming the human participants with hints from a "Dr. Smith"; `b-learners` receive a hint with the correct `f` but incorrect value for `b`, and the reverse for `f-learners`. An example hint is given in Table 3.

4 Methods

4.1 Teaching Toward Misconceptions

We introduce an approach that makes explicit inferences about the parameters of the student's prior. We call this method **ATOM (Adaptive Teaching tOward Misconceptions)**.

Like the OT method described in (§2), ATOM assumes that students are Bayesian reasoners and chooses examples to maximize the posterior probability that the student assigns to the target concept h^*. Because the student's prior is unknown, however, this process involves two steps:

1. **Maximum *a posteriori* estimation of student priors**. Recall that, during interaction, the teacher provides inputs x_i, then observes student guesses $\hat{y}_i$ before providing ground-truth labels y_i. In ATOM, the teacher selects student prior parameters that best explain the student's sequence of guesses:

$$\alpha_i = \arg\max_{\alpha} \sum_i \log p_S(\hat{y}_i \mid \underline{x}, \underline{y}, x_i, \alpha) \quad (4)$$

where $(\underline{x}, \underline{y}) = [(x_1, y_1), \ldots, (x_{i-1}, y_{i-1})]$. Estimating this $\arg\max$ requires a tractable procedure for computing the posterior predictive distribution, which is available for each simulated student model we evaluate in §5.

2. **Optimal selection of informative examples**. As in OT, once α has been estimated, we choose an example (x_{i+1}, y_{i+1}) to optimize:

$$\arg\max_{x_{i+1}, y_{i+1}} p_S(h^* \mid \underline{x}, \underline{y}, x_i, y_i, x_{i+1}, y_{i+1}, \alpha) \quad (5)$$

We note that many more sophisticated ATOM-type methods are possible—for example, explicit marginalization (rather than MAP estimation) of student priors. More basically, the method described above does not perform any active experimentation to identify the student prior; alternative ATOM implementations could explicitly trade off between exploration (of the student type) and exploitation (of the student posterior).

4.2 Other Methods

Random The **RANDOM** baseline uniformly samples an input to show the student.

Ranking A second baseline ranks the datapoints at the first step according to the objective in Eq 2, then chooses them in this order for the rest of the teaching interaction. The student type is chosen uniformly. We refer to this baseline as **RANKING**.

Non-Adaptive A third baseline selects examples according to the OT objective in Eq 2 but maintains a *fixed* guess about the student type, chosen uniformly at the start of teaching. This baseline can be thought of as an ablation of the adaptive piece of ATOM. We refer to this baseline as **NON-ADAPTIVE** (Shafto et al., 2014).

[4] `https://github.com/unimorph/unimorph`

[5] By Dirichlet–Multinomial conjugacy, the posterior distribution over parameters factorizes, is also a product of Dirichlet and Beta distributions, and can be efficiently computed.

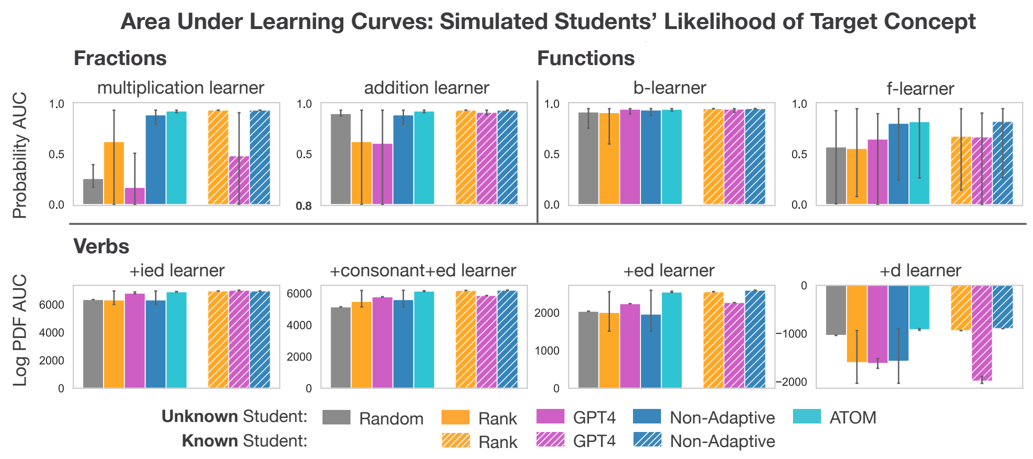


Figure 3: Area under simulated students' learning curves, where curves plot students' posterior beliefs in the target concept by number of datapoints. We report results by task and student type with 3 random seeds per bar. Dashed bars indicate that the true student type is assumed. Note that the y-axis for the `f-learner` for functions starts at 0.8, as these students all learn the concept early on, and so differences in teaching methods are small. Error bars show min/max values across seeds. Full learning curves are shown in Figure 9.

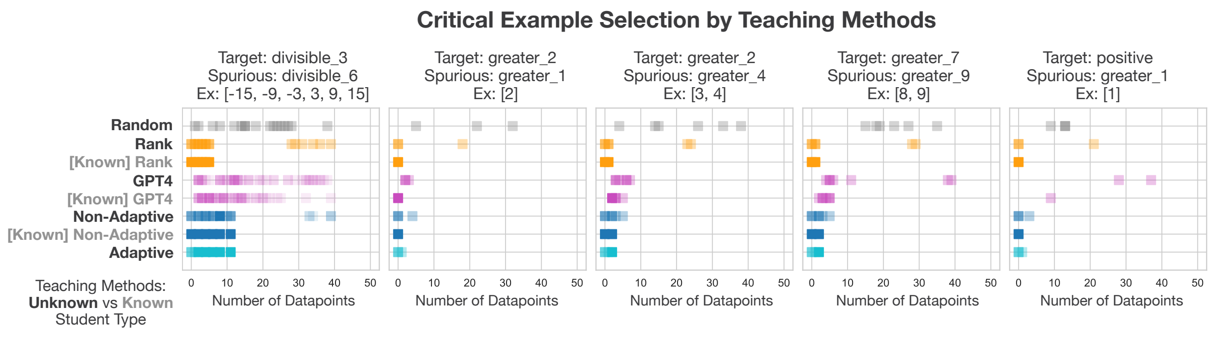


Figure 4: Critical example selection by different teaching methods for the function task. Results are for simulated `f-learners`, who have a spurious belief about `f` that agrees with the target `f` on all but a few examples, as labeled. The opacity of each square corresponds to the mean value of whether the example chosen by the teaching method at that step in learning is a "critical example" (averaged across experimental conditions: seed and concepts). See §5.4 for details. We report a subset of results here; see §11 for full results.

GPT-4 We prompt the `gpt-4-0314` model to select teaching examples (and provide no other explanations); the prompt describes the target concept, the student's hypothesis space, and the student types. The model is instructed to try to infer the student type in order to teach most efficiently. See Appendix E for actual prompts. To control for the fact GPT-4 sometimes generates incorrect outputs for examples, we use **ground truth** outputs for generated inputs.[6] We call this method **GPT-4**.

4.3 Oracle Methods

We also compare against several methods that assume access to the true student. These serve as comparison points for how well methods *could* do if they inferred the correct student type. We run this reference for all methods except RANDOM, which does not make use of a student model. We refer to these methods as **RANKING-KNOWN**, **NON-ADAPTIVE-KNOWN**, and **GPT-4-KNOWN**.

5 Simulated Experiments

5.1 Experimental Set-Up

We run 3 random seeds for all experimental conditions. For all methods except the GPT-4-based methods, we restrict the teaching methods from selecting previously selected examples. Teaching interactions last 40 steps for the fraction/function tasks and 50 steps for the verb task. For non GPT-4 methods, we enumerate over either the full dataset (fractions/functions) or a sampled subset (500 examples for verbs) to choose teaching examples.

[6]We parse GPT-4 generations to get inputs and create new messages with target outputs. More details can be found in §E.

5.2 Students' Learning Efficiency

We evaluate teacher effectiveness by measuring the student's **probability of the target concept,** h^*. Figure 3 shows the area under simulated students' learning curves, where curves reflect students' beliefs in h^* (full curves are shown in Figure 9). We observe that ATOM performs almost as well as the optimal strategy, NON-ADAPTIVE-KNOWN, and outperforms NON-ADAPTIVE, suggesting adaptation is both possible (*i.e.,* student type is inferrable from interaction) and that it leads to improved teaching efficiency. We also observe that both GPT-4-KNOWN and GPT-4 outperform RANDOM but underperform both ATOM and the non-adaptive probabilistic approaches.

5.3 GPT-4's Inferences about Student Type

We query GPT-4 for the student type at the end of the teaching interaction (*Based on this interaction, which kind of student do you think I was at the start of this teaching session ...*). See Table 11 for an example prompt. The mean accuracies of GPT-4's student type guesses are **100%** for verbs, **66.67%** for fractions, and **53.47%** for functions.

A possible explanation for these discrepancies is that for the fraction and function tasks, students successfully learn the target concept by the end of the teaching interaction and thus make accurate predictions; for the verb task, however, students are still making errors by the end.[7] We analyze how these accuracies change throughout the teaching interactions. For the function task, the student type accuracies are **64.2%**, **60.4%**, **56.3%**, **53.47%** after 10, 20, 30, and 40 steps, respectively: This decrease suggests that GPT-4 exhibits **recency bias** in making inferences about student type.

5.4 Selection of Critical Examples

For function concepts, we evaluate how early teaching methods select "critical examples," or key examples that distinguish the target `f` from the spurious `f`.[8] Consider the case where the target `f` is `greater_2` but the `f-learner` believes it is `greater_4`: The critical examples are **3** and **4** because they are the only examples on which the target `f` and spurious `f` would return different outputs. Observing `wug(x)` on one of these inputs would make clear to the `f-learner` that their belief about `f` is wrong. Therefore, an effective teacher should select such examples early in teaching.

As shown in Figure 4, the probabilistic methods assuming the student type (RANKING-KNOWN, NON-ADAPTIVE-KNOWN) all select critical examples early. GPT-4-KNOWN also shows a concentration of critical examples early, though they are more spread out for some concepts (*i.e.,* for target `f divisible_3/positive`); GPT-4 thus exhibits some pedagogical reasoning, focusing on examples that will target the `f-learner`'s misconceptions when it knows that the student type is a `f-learner`. When GPT-4 does not know the student type, we observe that critical examples are still more concentrated at the start than for RANDOM, suggesting some degree of adaptivity. Finally, we observe that ATOM selects critical examples at comparable points to NON-ADAPTIVE-KNOWN despite having to guess about student type.

5.5 Qualitative Differences in Teaching Math

For function concepts, recall that when `wug(x)` is defined, it computes `a*x+b`. We analyze how different methods teach what `a` and `b` are when `wug(x)` is defined by plotting the inputs they choose. As Figure 5 shows, GPT-4 tends to select inputs in order of increasing magnitude. In contrast, ATOM starts with higher-magnitude examples, then selects examples in increasing order. These qualitative differences suggest that GPT-4 may have encoded information that inputs closer to the origin are easier to learn from than those further from the origin.[9]

[7]See Figure 10 for how the correctness of student predictions on teaching examples changes throughout learning.

[8]We consider a subset of target concepts/spurious concepts where the number of critical examples is less than 10.

[9]In contrast, ATOM scores inputs according to how many incorrect concepts they "rule out," treating all else as equal.

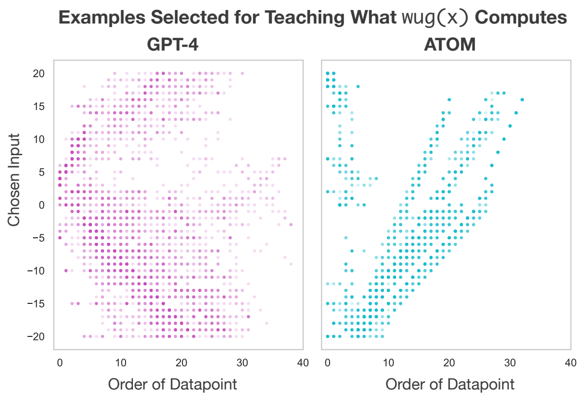


Figure 5: Examples selected by different teaching methods for teaching `a` and `b` in the function learning task (*i.e.,* what `wug(x)` computes when it is defined). The x-axis indicates the order of the chosen example compared to other examples targeting `a` and `b`.

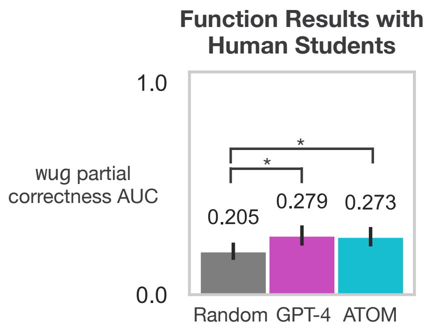


Figure 6: Results with human students showing how efficiently students guessed the correct wug concept (§6.2). Stars indicate statistically significant results under a paired t-test. Error bars show 95% confidence intervals.

6 Human Experiments

6.1 Experimental Set-Up

We recruit Prolific users who are fluent English speakers and who indicated some experience with computer programming. We pay participants a base pay of $4.00 per study ($16/hour) and offer a bonus based on the accuracy of their predictions to the teacher and on how early they guess the correct value for `wug(x)`. See Appendix F for details.

We evaluate RANDOM, GPT-4, and ATOM teachers on the function task. We run 5 experiments per teaching method per experimental condition.[10] These experiments were classified as an exempt Benign Behavioral Intervention by our IRB.

6.2 Students' Learning Efficiency

We evaluate the effectiveness of teachers by measuring the **correctness of wug guesses** made by human participants, computing an AUC-like metric. We consider all timestamps where at least one participant made a guess for wug and compute a partial correctness metric for each guess: (f is correct) + 0.5(a is correct) + 0.5(b is correct).[11] We report the *mean* correctness values across timestamps.

As shown in Figure 6, we find that both GPT-4 and ATOM improve significantly over the RANDOM baseline ($p < 0.05$ using a paired t-test). Interestingly, these differences are entirely explained by model behavior for `f-learners`: after controlling for student type, improvements over the random teacher are significant for `f-learners` but insignificant for `b-learners`. For results by student type, time, and individual participants, see §F.2.

6.3 Inferences about Student Type

We evaluate the accuracy of predictions of student type made by ATOM and GPT-4 after each minute of teaching. ATOM makes more accurate predictions than GPT-4, with respective mean accuracies of **71.33%** and **52.61%** across participants and minutes. Accuracies over time are shown in §F.2 (Figure 19).

7 Discussion

Both our human and simulated results show that ATOM and GPT-4 exhibit pedagogical ability over random example selection. ATOM's performance with human students suggests that the Bayesian assumptions made by ATOM are accurate models of some aspects of human learning. Across human and simulated experiments, we find evidence of some adaptivity in GPT-4, though less than in ATOM, both in the examples selected (§5.4) and inferences about student type (§5.3, §6.3). See §A for additional analyses about how adaptive selected teaching examples are to student beliefs.

We also observe other qualitative limitations of GPT-4 as a teacher: selecting the same teaching examples multiple times or terminating teaching early due to an incorrect belief that all teaching examples have been exhausted. It is important to highlight that because we use ground truth outputs (*e.g.,* ground truth function evaluations) with GPT-4, the GPT-4 results represent an upper bound on GPT-4's performance.

Despite these limitations, however, GPT-4 performs comparably to ATOM with human students, suggesting pedagogical benefits beyond adaptivity. In particular, the analysis in §5.5 suggests that LMs may encode information about human learning that is hard to represent in more structured approaches like ATOM—*e.g.,* that it is easier for humans to learn the weights of a line from inputs closer to the origin.

Together, our results point to complementary advantages of LM teachers like GPT-4 and more structured, probabilistic methods like ATOM. They suggest that there is substantial headroom to improve real-world teaching by augmenting the in-

Because higher-magnitude inputs tend to result in outputs that can be explained by fewer functions, ATOM selects higher-magnitude inputs early in teaching.

[10]There are 22 unique experimental conditions (11 target concepts, 2 student types per concept).

[11]If no new guess was made by a user, we use their last guess.

ferences of structured models with richer information about the priors that humans bring to learning, whether by combining such structured methods with information encoded in LMs or by developing other rich models of student priors—*e.g.,* by learning more complex "student types" from naturally occurring data. We perform an initial experiment with human students in this direction by combining GPT-4 and ATOM but do not find improvements compared to either teacher alone; see §F.3 for details.

Other directions for future work include modeling more complex student phenomena—accounting for students who ask questions, reason pedagogically about teacher intentions, and provide feedback to teachers (Chen et al., 2022)—and creating methods for adaptive teaching with natural language explanations in real-world teaching domains.

8 Related Work

This work builds on a long line of work in **rational models of pedagogy**, including Bayesian models like those described by Shafto et al. (2014) and Sumers et al. (2021), as well as improved planning and inference procedures like the one described by Rafferty et al. (2016). Past work generally assumes students' initial belief states are known. In parallel, Rafferty et al. (2015) use an inverse planning model to infer students' prior beliefs from their actions, and Chen et al. (2022) show that human teachers adapt examples to these prior beliefs.

This work is also closely related to other bodies of work that aim to infer student knowledge. **Item response theory (IRT)** infers a scalar measure of student's skill based on their responses to questions (Hambleton and Swaminathan, 1981; Hambelton and Jodoin, 2003). **Knowledge tracing (KT)** models students' evolving knowledge states over time separately for a fixed set of skills (Corbett and Anderson, 1994); previous work has used both bayesian methods for modeling individual students' prior knowledge (Yudelson et al., 2013) and neural models for modeling (Piech et al., 2015) and targeting (Srivastava and Goodman, 2021) students' evolving learning states. In contrast to IRT and KT, our work aims to infer the student's entire prior and posterior over the concept space; these inferences in turn enable more fine-grained design of individual teaching examples.

Inferring student misconceptions from errors also uses tools from a broader literature on computational models of **theory of mind**. Prominent work includes general-purpose bayesian models of other agents' beliefs and desires (Baker et al., 2011) and models of pragmatic inference grounded in recursive reasoning about speakers and listeners (Frank and Goodman, 2012). More recent work has studied theory of mind capabilities in LMs; they find largely negative results in unaugmented LLMs (Sap et al., 2023) but positive results from LMs augmented with structured belief representations (Sclar et al., 2023). Recent work has also explored LLMs' theory of mind abilities in teaching smaller LMs (Saha et al., 2023).

There is also a large body of work on how to optimally provide and interpret human **supervision for ML models**. General frameworks for this problem include Machine Teaching (Zhu et al., 2018) and Cooperative IRL (Hadfield-Menell et al., 2016); related ideas appear in program synthesis (Vaduguru et al., 2022), robot learning (Milli and Dragan, 2019; Dragan et al., 2013), and natural language processing (Li et al., 2023) as well.

There has been increased interest in using LLMs to assist or supplement human teachers. See Kasneci et al. (2023) for a survey and Wang et al. (2023) for a specific application to math teaching problems. Concurrent work by Chandra et al. (2024) uses program synthesis techniques to infer misconceptions and provide explanations about Javascript. Our work adds to this literature by providing a framework that allows reproducible evaluation of the effectiveness and personalization skills of LLMs as teachers, as well as a new model that empirically improves upon LLM baselines in teaching humans a new task.

9 Conclusion

We introduce ADAPT, an evaluation suite measuring how effectively automated teaching methods can teach students with different prior misconceptions. We also introduce ATOM, a two-part probabilistic approach to adaptive teaching that maintains explicit inferences about student priors. Our evaluations of ATOM, LLMs, and other probabilistic baselines with both simulated and human students highlight the potential of learned adaptive models for solving the adaptive teaching task.

10 Limitations

One limitation of our work is that in our model of teaching, teachers are limited to example selection and students are limited to observation. These restrictions leave out that teachers can provide explanations, and students can ask questions and provide feedback to teachers. An interesting direction for future work would be to both create an evaluation framework for such phenomena and develop models for these richer forms of teaching and learning feedback. Relatedly, students can also engage in pedagogical reasoning about why teachers chose particular examples, which can in turn influence how they learn from these examples; while the simulated students in ADAPT do not model this phenomenon, future work could explore richer models of students.

Another limitation is that we create the student types by instantiating priors in particular ways rather than deriving the priors from real-world data. Future work could explore how to automatically *learn* the types of priors that human students bring to different teaching scenarios.

Acknowledgements

We are grateful to Ekin Akyürek, Leshem Choshen, Gabriel Grand, Robert Hawkins, Andi Peng, Megha Srivastava, Lionel Wong, Zhaofeng Wu, and members of the MIT Language & Intelligence group for helpful discussions and feedback on this work. AR was funded by an MIT Shillman Fellowship and the NSF GRFP 2023357727; additional funding was provided by the National Science Foundation under grants IIS-2238240 and IIS-2212310, and by the Alfred P. Sloan Foundation. We also acknowledge the use of the following icons from Noun Project: “student” and “Teacher” by Crystal Gordon, “Star” by Bluetip Design, “X” by Alice Design, and “Check” by Siti Masriatun.

References

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2023. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*.

Chris Baker, Rebecca Saxe, and Joshua Tenenbaum. 2011. Bayesian theory of mind: Modeling joint Belief-Desire attribution. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 33(33).

Khuyagbaatar Batsuren, Omer Goldman, Salam Khalifa, Nizar Habash, Witold Kieraś, Gábor Bella, Brian Leonard, Garrett Nicolai, Kyle Gorman, Yustinus Ghanggo Ate, Maria Ryskina, Sabrina Mielke, Elena Budianskaya, Charbel El-Khaissi, Tiago Pimentel, Michael Gasser, William Abbott Lane, Mohit Raj, Matt Coler, Jaime Rafael Montoya Samame, Delio Siticonatzi Camaiteri, Esaú Zumaeta Rojas, Didier López Francis, Arturo Oncevay, Juan López Bautista, Gema Celeste Silva Villegas, Lucas Torroba Hennigen, Adam Ek, David Guriel, Peter Dirix, Jean-Philippe Bernardy, Andrey Scherbakov, Aziyana Bayyr-ool, Antonios Anastasopoulos, Roberto Zariquiey, Karina Sheifer, Sofya Ganieva, Hilaria Cruz, Ritván Karahóǧa, Stella Markantonatou, George Pavlidis, Matvey Plugaryov, Elena Klyachko, Ali Salehi, Candy Angulo, Jatayu Baxi, Andrew Krizhanovsky, Natalia Krizhanovskaya, Elizabeth Salesky, Clara Vania, Sardana Ivanova, Jennifer White, Rowan Hall Maudslay, Josef Valvoda, Ran Zmigrod, Paula Czarnowska, Irene Nikkarinen, Aelita Salchak, Brijesh Bhatt, Christopher Straughn, Zoey Liu, Jonathan North Washington, Yuval Pinter, Duygu Ataman, Marcin Wolinski, Totok Suhardijanto, Anna Yablonskaya, Niklas Stoehr, Hossep Dolatian, Zahroh Nuriah, Shyam Ratan, Francis M. Tyers, Edoardo M. Ponti, Grant Aiton, Aryaman Arora, Richard J. Hatcher, Ritesh Kumar, Jeremiah Young, Daria Rodionova, Anastasia Yemelina, Taras Andrushko, Igor Marchenko, Polina Mashkovtseva, Alexandra Serova, Emily Prud’hommeaux, Maria Nepomniashchaya, Fausto Giunchiglia, Eleanor Chodroff, Mans Hulden, Miikka Silfverberg, Arya D. McCarthy, David Yarowsky, Ryan Cotterell, Reut Tsarfaty, and Ekaterina Vylomova. 2022. UniMorph 4.0: Universal Morphology. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 840–855, Marseille, France. European Language Resources Association.

David W Braithwaite, Aryn A Pyke, and Robert S Siegler. 2017. A computational model of fraction arithmetic. *Psychol. Rev.*, 124(5):603–625.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Kartik Chandra, Tzu-Mao Li, Rachit Nigam, Joshua Tenenbaum, and Jonathan Ragan-Kelley. 2024. From ‘why?’ to ‘wat!’: Explaining perplexing programs by debugging mental models. In *PLATEAU Workshop*.

Alicia M Chen, Andrew Palacci, Natalia Vélez, Robert Hawkins, and Samuel J Gershman. 2022. Learning to teach, teaching to learn.

Albert Corbett. 2001. Cognitive computer tutors: Solving the Two-Sigma problem. In *User Modeling 2001*, pages 137–147. Springer Berlin Heidelberg.

Albert T Corbett and John R Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Model. User-adapt Interact.*, 4(4):253–278.

Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. 2013. Legibility and predictability of robot motion. In *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction*, HRI '13, page 301–308. IEEE Press.

Michael C. Frank and Noah D. Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. 2016. Cooperative inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

RK Hambelton and M Jodoin. 2003. Item response theory: models and features.

Ronald K. Hambleton and Hariharan Swaminathan. 1981. *Journal of Educational Measurement*, 18(3):178–180.

Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274.

Belinda Z Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. 2023. Eliciting human preferences with language models.

Smitha Milli and Anca D Dragan. 2019. Literal or pedagogic human? analyzing human model misspecification in objective learning.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chris Piech, Jonathan Spencer, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J. Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. *CoRR*, abs/1506.05908.

Anna N Rafferty, Emma Brunskill, Thomas L Griffiths, and Patrick Shafto. 2016. Faster teaching via POMDP planning. *Cogn. Sci.*, 40(6):1290–1332.

Anna N Rafferty, Michelle M LaMar, and Thomas L Griffiths. 2015. Inferring learners' knowledge from their actions. *Cogn. Sci.*, 39(3):584–618.

Swarnadeep Saha, Peter Hase, and Mohit Bansal. 2023. Can language models teach weaker agents? teacher explanations improve students via personalization.

Maarten Sap, Ronan LeBras, Daniel Fried, and Yejin Choi. 2023. Neural theory-of-mind? on the limits of social intelligence in large lms.

Melanie Sclar, Sachin Kumar, Peter West, Alane Suhr, Yejin Choi, and Yulia Tsvetkov. 2023. Minding language models' (lack of) theory of mind: A plug-and-play multi-character belief tracker.

Patrick Shafto, Noah D Goodman, and Thomas L Griffiths. 2014. A rational account of pedagogical reasoning: teaching by, and learning from, examples. *Cogn. Psychol.*, 71:55–89.

Megha Srivastava and Noah Goodman. 2021. Question generation for adaptive education. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 692–701, Online. Association for Computational Linguistics.

Theodore R Sumers, Robert D Hawkins, Mark K Ho, and Thomas L Griffiths. 2021. Extending rational models of communication from beliefs to actions.

Joshua Tenenbaum. 1999. Rules and similarity in concept learning. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Saujas Vaduguru, Kevin Ellis, and Yewen Pu. 2022. Efficient pragmatic program synthesis with informative specifications.

Rose E. Wang, Qingyang Zhang, Carly Robinson, Susanna Loeb, and Dorottya Demszky. 2023. Bridging the novice-expert gap via models of decision-making: A case study on remediating math mistakes.

Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. 2013. Individualized bayesian knowledge tracing models. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 171–180. Springer Berlin Heidelberg, Berlin, Heidelberg.

Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N Rafferty. 2018. An overview of machine teaching.

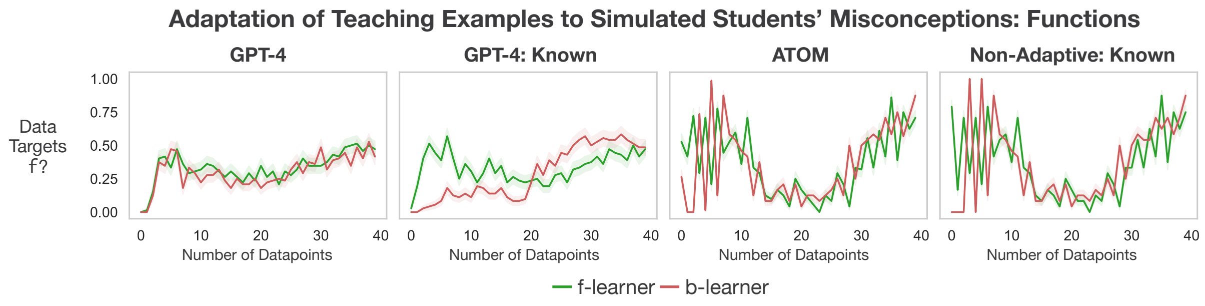


Figure 7: Plot showing whether chosen teaching examples target learning `f` (*i.e.,* when `wug(x)` is undefined) or `a*x+b` (*i.e.,* what `wug(x)` computes when defined) for the function task. A y-value of 1.0 indicates that the teaching example targets `f`; 0.0 indicates that it does not. An input x for which `wug(x)` is *undefined* targets `f`, and an input for which `wug(x)` is *defined* targets `a*x+b`. Intuitively, `f-learners` benefit more from seeing examples targeting `f` early on in teaching, and `b-learners` benefit more from seeing examples that target `a*x+b`. Error regions indicate standard errors of the mean.

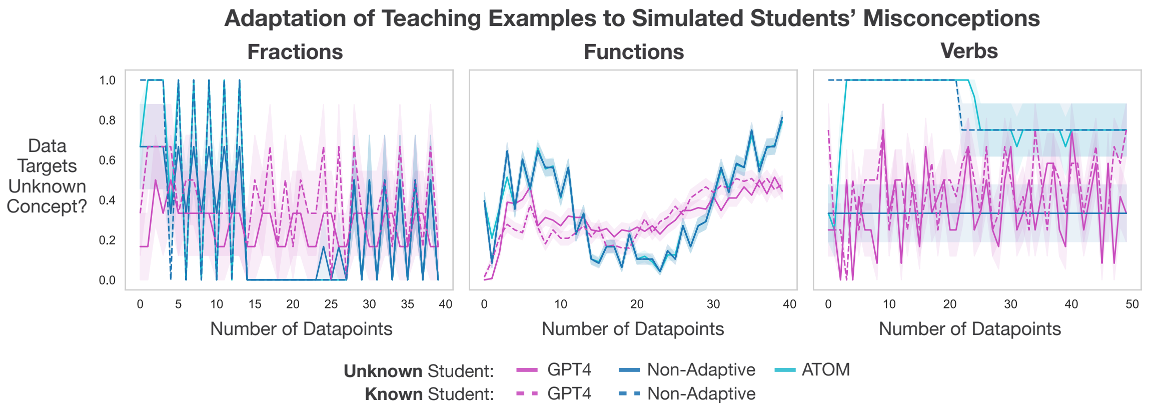


Figure 8: Plot showing whether chosen teaching examples across experiments target student misconceptions. A y-value of 1.0 indicates that the teaching example targets misconceptions; 0.0 indicates that it does not. For fractions, multiplication problems with common denominators target `mult-learners`' misconceptions, and addition problems with different denominators target `add-learners`' misconceptions. For functions, inputs x for which `f(x)` returns True target `f-learners`' misconceptions, and inputs for which `f(x)` returns False target `b-learners`' misconceptions. For verbs, inputs with class "unknown" by the student (*e.g.,* a +ied verb for a `+ied-learner`) target the student's misconception. Error regions indicate standard errors of the mean.

A Do teaching examples target student misconceptions?

For the simulated evaluations, we analyze whether the examples selected by different teaching methods target students' specific misconceptions.

Functions Figure 7 shows whether the selected teaching examples target learning `f` or `a*x+b` in the target `wug` concept, split by student type. We expect an adaptive teacher to select examples targeting `f` for `f-learners` at the start of learning, and similarly for examples targeting `a*x+b` for `b-learners`.

In both plots, we see that the optimal teacher NON-ADAPTIVE-KNOWN, which assumes access to the ground truth student type, exhibits the expected behavior: It selects more examples targeting `f` for the `f-learner` than for the `b-learner` at the beginning of the teaching interaction. ATOM, despite needing to maintain guesses for the student type, shows similar adaptivity to students' priors early on, selecting more examples targeting `f` for the `f-learner` than for the `b-learner` at the start of teaching.

GPT-4-KNOWN also shows this adaptivity when assuming the true student type. However, when it does not have access to the true student type, it does not show this adaptation; the data selection patterns of GPT-4 are highly similar for the `b-learners` and `f-learners`, suggesting that GPT-4 struggles with doing implicit adaptation online.

All Tasks Figure 8 shows adaptation of teaching examples across all tasks and student types. We observe similar trends: Across tasks, ATOM shows similar levels of adaptivity to NON-ADAPTIVE-KNOWN, despite not knowing student type, and outperforms both GPT-4 methods. We also observe GPT-4-KNOWN selecting more examples targeting unknown concepts than GPT-4.

B Creating Function Concepts and Student Types

b has possible values $[1, 2, \cdots, 9]$ and a has possible values $[-5, -4, \cdots, 4, 5]$. For each concept, to create the `b-learner`, we uniformly sample the incorrect b from the range of possible values of b, excluding the target b; to create the `f-learner`, we uniformly sample the incorrect f from a set of spuriously associated functions: These possible values are shown in Table 5. The full list of concepts and incorrect student beliefs are shown in Table 4.

C Creating the Verbs Dataset

For the verbs task, we create verb classes with reg exp matching on their past tense forms. For the GPT-4 method, we create ground truth outputs by first checking if a lemma exists in the Unimorph dataset; if not, we use a Python verb inflection package `pyinflect`[12] to first get the past tense form of the verb, then categorize it. Table 1 shows verb classes and corresponding counts in the resulting dataset.

The model that we fit on the full dataset (to derive the parameters of the target concept) obtains a predictive accuracy of 95.47%, and the mean probability of the ground truth outputs across the full dataset is 0.945.

D Simulated Experiments

For all tasks, we obtain predictions from simulated students by sampling from their predictive distributions.

Program Tasks For the programmatic tasks (functions, fractions), each simulated student's prior belief in a program h is proportional to the number of "special primitives" that appear in the program. We derive the prior over programs by multiplying a value c by the number of special primitives that appear in a program to get values for all programs; we then normalize these values to get a distribution over programs.

The simulated students for the function and fraction tasks also maintain a **noise parameter** that governs how noisy the labels are in the examples they observe; this noise parameter governs their posterior updates. For fractions, this noise parameter is 0.8, and for functions, it is 0.05. We use the same noise values for the teacher's models of the students.

Fractions Table 2 shows the multiplication and addition sub-programs that are used to create the concept space for simulated students for the fraction task. For the `mult-learner` who over-generalizes the procedure for addition, the "special primitives" are (1), (4), and (6). For the `add-learner` who over-generalizes the procedure for multiplication, the "special primitives" are (2), (3), and (5). We use a value of $c = 1e5$.

Functions For `b-learners`, the "special primitives" are the target f and spurious b (and so programs with either of these primitives would have higher prior beliefs; programs having *both* the target f and spurious b would have the highest prior belief). Similarly, for `f-learners`, the incorrect f and target b are the special primitives. We use a value of $c = 1e4$.

E GPT-4: Details

Prompts for GPT-4 are shown in Tables 6, 7, and 8 for fractions, Tables 9, 10, and 11 for functions, and Tables 12, 13, and 14 for verbs. An example conversation between GPT-4 and a simulated student is shown in Table 15.

Processing/Filtering GPT-4 Outputs In order to control for the fact that GPT-4 may generate incorrect outputs for examples, we use ground truth outputs for generated inputs. We parse GPT-4-generated messages to obtain inputs, then compute ground truth labels for those inputs and append them to the message history, starting with "That's correct/incorrect." If a message cannot be parsed, we append a canned response, *i.e., "Sorry, I could not learn from that example. I can only learn from examples that are formatted as..."* (if no output can be parsed from the message) or *"I would like to keep learning. Can I have another example?"* (if no input can be parsed from the message); these messages do not count as an "interaction" in comparing against other teaching methods. For human experiments, we do not display these canned responses to the students and instead only display messages asking for predictions on examples and providing ground truth answers.

[12] `https://github.com/bjascob/pyInflect`

Verb Class	Description	Example	Counts
+ed	add *'ed'* to the lemma	*clasp*	6,130
+d	add *'d'* to the lemma	*smile*	13,463
+ied	replace last *'y'* with *'ied'*	*cry*	1,056
+consonant+ed	double last consonant, add *'ed'*	*stop*	1,878

Table 1: Verb classes and corresponding dataset counts for the verb conjugation task (§3.1).

Addition Sub-Programs		
(1)	(2)	(3)

`make common denominators`
`add numerators`
	`add numerators & denominators`	`if denominators are equal:`
		`  add numerators`
		`else:`
		`  add numerators & denominators`

Multiplication Sub-Programs		
(4)	**(5)**	(6)

`make common denominators`
`multiply numerators`
	`multiply numerators & denoms`	`if denominators are equal:`
		`  multiply numerators`
		`else:`
		`  multiply numerators & denominators`

Table 2: The addition and multiplication sub-programs in the concept space for the fraction task in ADAPT (§3). The sub-programs in the target concept (*i.e.,* correct sub-programs for adding/multiplying fractions) are **bolded**.

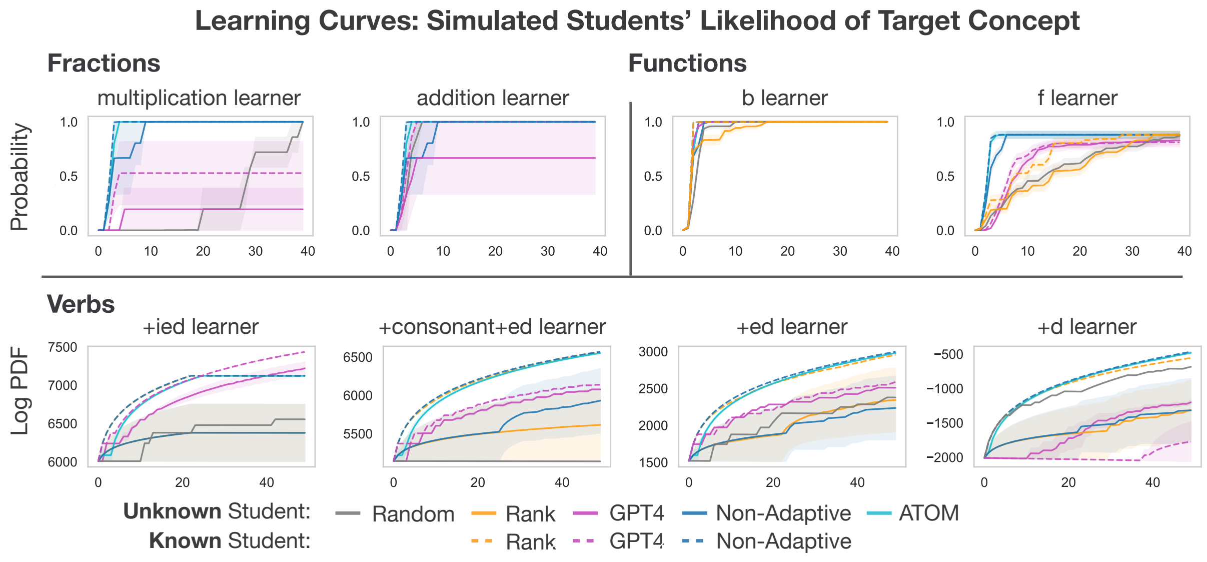


Figure 9: Learning curves for simulated students. Top row: results for function learning and fraction arithmetic, with y-axis showing the probability of the target concept. Bottom row: results for verb conjugation, with y-axis showing the log PDF of the target concept. Each subplot corresponds to a different student type. Color indicates teaching method. Linestyles indicate whether the true student type is assumed (*dashed*=unknown, *solid*=known). Confidence intervals indicate standard error of the mean.

Decoding For all experiments with GPT-4, we use a temperature of 0.5 and maximum tokens of 100.

F Human Experiments

F.1 Set-Up

Post-Processing We filter and rerun any experiments where the chat messages were logged out of order or sent twice.

Hyperparameters ATOM uses a noise parameter value of 0.02 for modeling simulated students.

Instructions An example of hints given to the human participants is shown in Table 3. The full set of instructions shown to human participants, along with the interface, are shown in Figures 12/13 (instructions), 14 (chat), and 15 (end).

Bonus Compensation Participants are told that their bonus depends on two things:

```
Dr. Smith spent a bunch of time studying this machine. She figured out that when wug is defined, it
computes a function of the form a*x+b, where a and b are constant numbers, so you only need to figure
out what a and b are.

She also left a note with some thoughts:
I'm pretty sure, but not totally confident, that:
1) wug is undefined when inputs are greater than 2
2) When wug is defined, b = 3
-Dr. Smith

Dr. Smith is quite familiar with wug, so her note should give you a good place to start! But keep in
mind that it is possible that she is wrong.
```

Table 3: An example of a hint given to a human learner. (1) is correct, while (2) is not, thus creating a `b-learner`.

1. Accuracy of wug guesses: Participants are told they will receive 0.05 for every 10 seconds of the teaching interaction that their guess is correct, with partial compensation if only f or only a/b is correct.

2. Accuracy of predictions on teaching examples: Participants are told they will receive up to an additional 1.00 based on the accuracy of their predictions.

As shown in Figure 13, participants are prompted to indicate their understanding of what their bonus depends on.

F.2 Additional Results

Figure 16 shows the learning efficiency of human students by student type. Figure 17 shows the correctness of wug guesses by component as a function of time. Figure 18 shows the AUCs of wug correctness by component for individual participants. Figure 19 shows the accuracy of student type predictions over time.

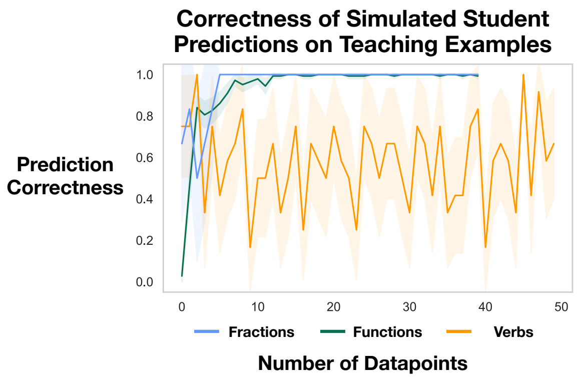


Figure 10: Correctness of simulated student predictions on teaching examples by task. Error regions indicate 95% confidence intervals.

F.3 Combining GPT-4 and ATOM

We run an experiment combining GPT-4 and ATOM in the following way: We use ATOM to make inferences about student type, then prompt GPT-4 with ATOM's inference by updating the system prompt to GPT-4 after each prediction made by the student. Before any predictions are given, GPT-4 is prompted with both student types (*i.e.,* with the prompt given to the teacher that does not know student type). We call this method GPT-4+ATOM.

Results are shown in Figure 20. We find that GPT-4+ATOM outperforms RANDOM ($p < 0.05$ using a paired t-test) but does not outperform GPT-4 or ATOM.

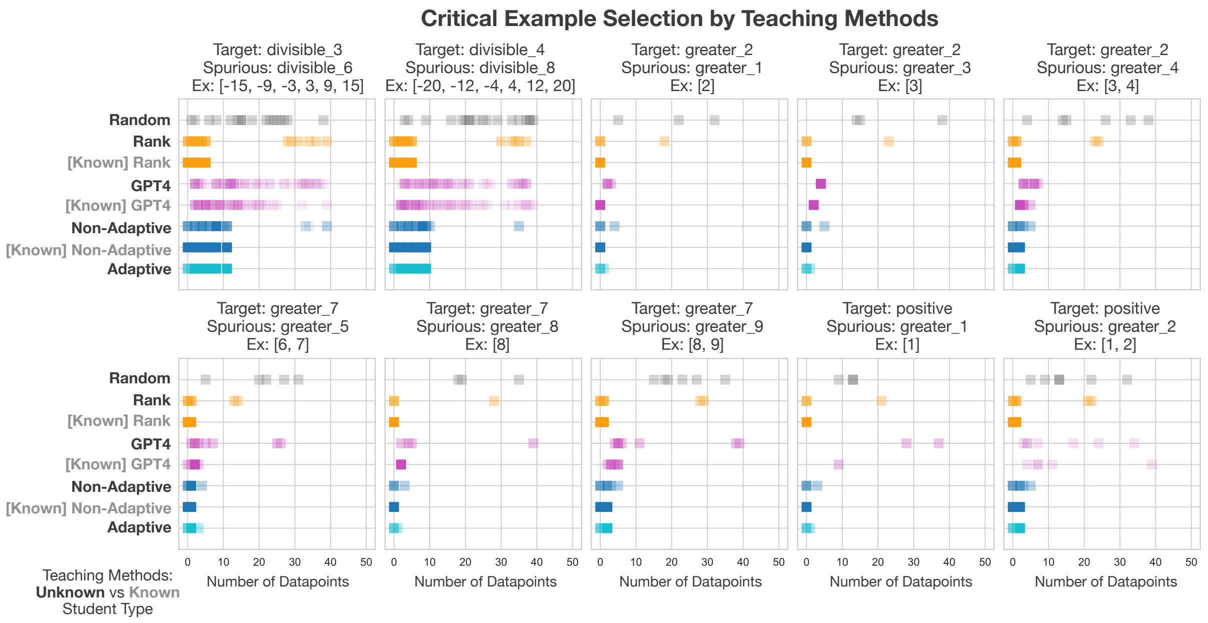
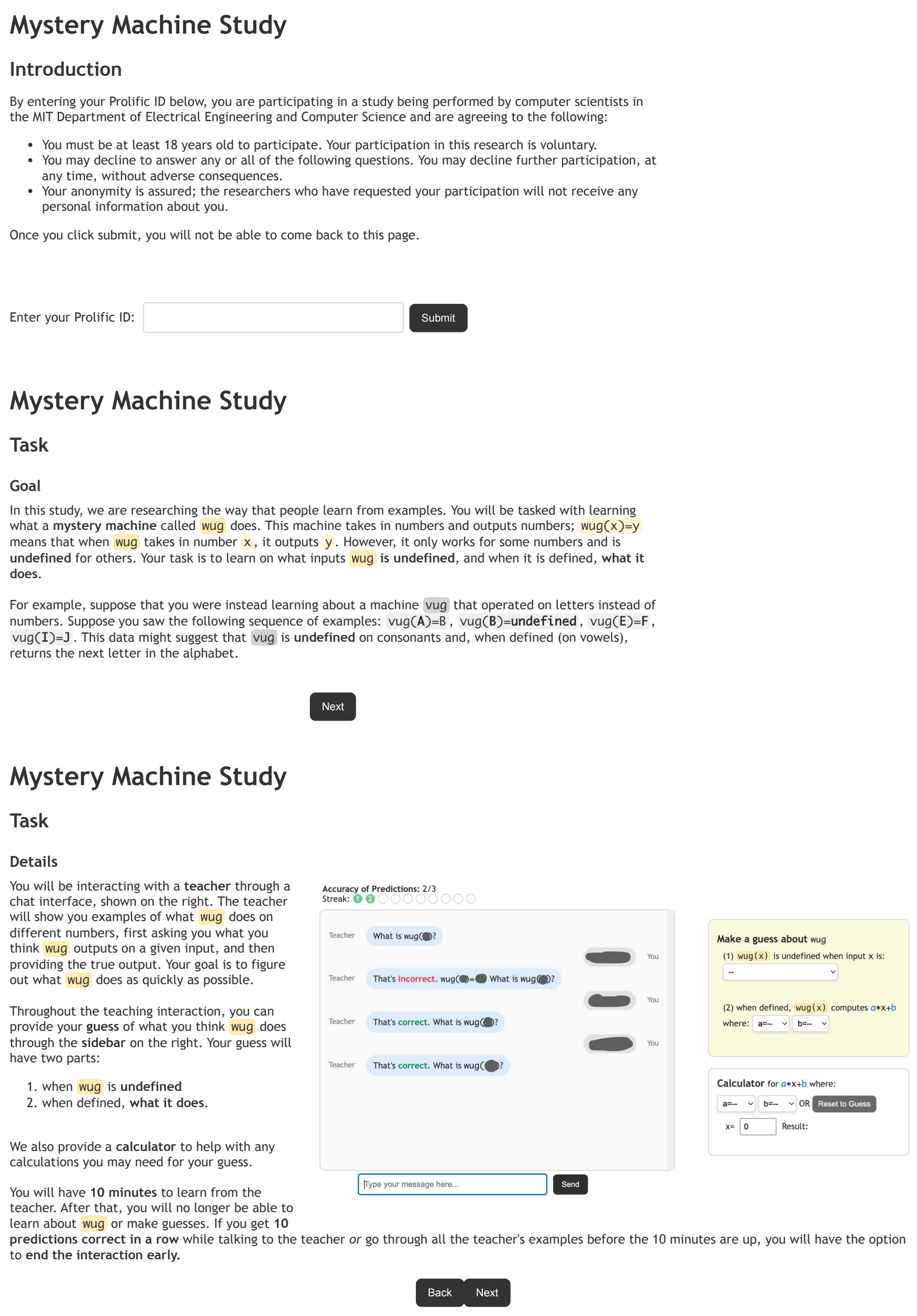


Figure 11: Full results for critical example selection by different teaching methods for the function task. Results are for simulated `f-learners`, who have a spurious belief about `f` that agrees with the target `f` on all but a few examples, as labeled. The opacity of each square corresponds to the mean value of whether the example chosen by the teaching method at that step in learning is a "critical example" (averaged across experimental conditions: seed and concepts). See §5.4 for details.

Target f	a	b	Incorrect f	b	Used in Human Experiments?
even	1	7	divisible_6	5	
even	-5	5	divisible_6	7	Y
even	3	8	divisible_4	3	Y
greater_2	1	7	greater_4	3	
greater_2	-5	5	greater_1	6	
greater_2	3	8	greater_3	5	Y
prime	1	7	odd	2	Y
prime	-5	5	odd	9	Y
prime	3	8	odd	6	
divisible_3	1	7	divisible_6	6	
divisible_3	-5	5	divisible_6	9	
divisible_3	3	8	divisible_6	5	
divisible_4	1	7	divisible_8	1	
divisible_4	-5	5	divisible_8	8	Y
divisible_4	3	8	divisible_8	9	Y
positive	1	7	greater_2	4	
positive	-5	5	greater_2	2	
positive	3	8	greater_1	4	
odd	1	7	divisible_5	3	Y
odd	-5	5	prime	2	Y
odd	3	8	divisible_3	6	Y
greater_7	1	7	greater_9	2	
greater_7	-5	5	greater_8	6	Y
greater_7	3	8	greater_5	6	

Table 4: The target concepts and incorrect beliefs used in the function learning experiments. The **target** `f`, `a`, `b` define the concepts being taught. The **incorrect** `f` is the belief that the `f-learners` have about `f` at the start of learning, and the incorrect `b` is the value that the `b-learners` believes `b` to be.



Mystery Machine Study

Introduction

By entering your Prolific ID below, you are participating in a study being performed by computer scientists in the MIT Department of Electrical Engineering and Computer Science and are agreeing to the following:

- You must be at least 18 years old to participate. Your participation in this research is voluntary.
- You may decline to answer any or all of the following questions. You may decline further participation, at any time, without adverse consequences.
- Your anonymity is assured; the researchers who have requested your participation will not receive any personal information about you.

Once you click submit, you will not be able to come back to this page.

Enter your Prolific ID:

Mystery Machine Study

Task

Goal

In this study, we are researching the way that people learn from examples. You will be tasked with learning what a **mystery machine** called `wug` does. This machine takes in numbers and outputs numbers; `wug(x)=y` means that when `wug` takes in number `x`, it outputs `y`. However, it only works for some numbers and is **undefined** for others. Your task is to learn on what inputs `wug` **is undefined**, and when it is defined, **what it does**.

For example, suppose that you were instead learning about a machine `vug` that operated on letters instead of numbers. Suppose you saw the following sequence of examples: `vug(A)=B`, `vug(B)=undefined`, `vug(E)=F`, `vug(I)=J`. This data might suggest that `vug` is **undefined** on consonants and, when defined (on vowels), returns the next letter in the alphabet.

Mystery Machine Study

Task

Details

You will be interacting with a **teacher** through a chat interface, shown on the right. The teacher will show you examples of what `wug` does on different numbers, first asking you what you think `wug` outputs on a given input, and then providing the true output. Your goal is to figure out what `wug` does as quickly as possible.

Throughout the teaching interaction, you can provide your **guess** of what you think `wug` does through the **sidebar** on the right. Your guess will have two parts:

1. when `wug` is **undefined**
2. when defined, **what it does**.

We also provide a **calculator** to help with any calculations you may need for your guess.

You will have **10 minutes** to learn from the teacher. After that, you will no longer be able to learn about `wug` or make guesses. If you get **10 predictions correct in a row** while talking to the teacher *or* go through all the teacher's examples before the 10 minutes are up, you will have the option to **end the interaction early.**

Accuracy of Predictions: 2/3
Streak: 1 2

Teacher: What is wug([redacted])?

You: [redacted]

Teacher: That's incorrect. wug([redacted])=[redacted] What is wug([redacted])?

You: [redacted]

Teacher: That's correct. What is wug([redacted])?

You: [redacted]

Teacher: That's correct. What is wug([redacted])?

Make a guess about wug

(1) `wug(x)` is undefined when input x is: --

(2) when defined, `wug(x)` computes a∗x+b where: a=-- b=--

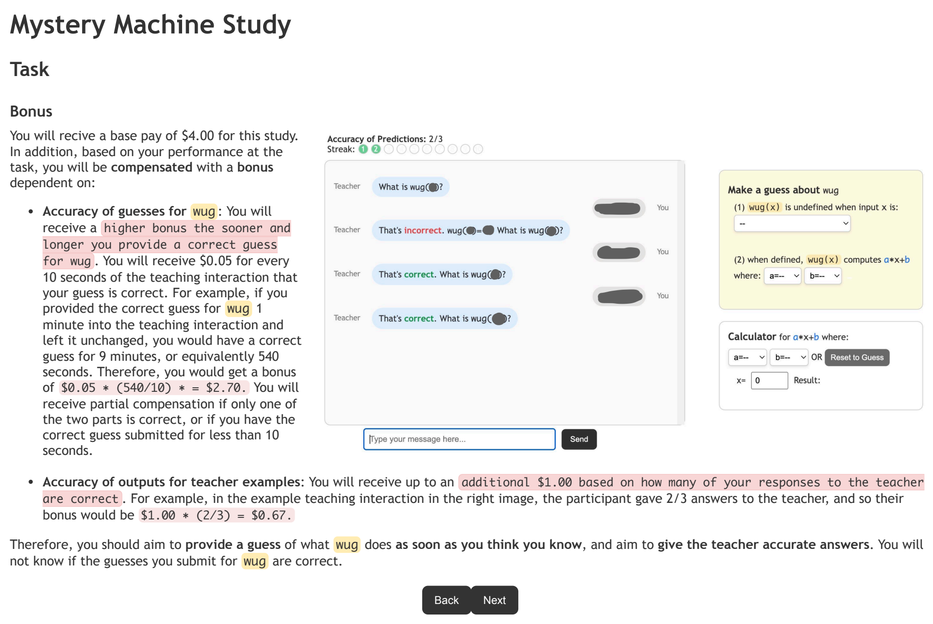
Calculator for a∗x+b where: a=-- b=-- OR

x= 0 Result:

Figure 12: Screens 1-3 (instructions) for the study with human participants.

f	Incorrect f Options
prime	odd
positive	greater_n for $n \in [-2, -1, 1, 2]$
even	divis_4, divis_6
odd	prime, divis_3, divis_5, divis_7
divis_n	divis_m where m is the smallest multiple of n or the largest factor of n (if $m = 2$, this is even)
greater_n	greater_m where $\|m - n\| <= 2$ (if $m = 0$, this is positive)

Table 5: Descriptions of how the options for incorrect f beliefs of f-learners are generated for each target f in the function task. We uniformly sample the incorrect f from the set of options to determine the actual incorrect f belief for the f-learners.



Mystery Machine Study

Task

Bonus

You will recive a base pay of $4.00 for this study. In addition, based on your performance at the task, you will be **compensated** with a **bonus** dependent on:

- **Accuracy of guesses for `wug`**: You will receive a `higher bonus the sooner and longer you provide a correct guess for wug`. You will receive $0.05 for every 10 seconds of the teaching interaction that your guess is correct. For example, if you provided the correct guess for `wug` 1 minute into the teaching interaction and left it unchanged, you would have a correct guess for 9 minutes, or equivalently 540 seconds. Therefore, you would get a bonus of `$0.05 * (540/10) * = $2.70.` You will receive partial compensation if only one of the two parts is correct, or if you have the correct guess submitted for less than 10 seconds.

Accuracy of Predictions: 2/3
Streak: 1 2

Teacher: What is wug([redacted])?
You: [redacted]
Teacher: That's incorrect. wug([redacted])=[redacted] What is wug([redacted])?
You: [redacted]
Teacher: That's correct. What is wug([redacted])?
You: [redacted]
Teacher: That's correct. What is wug([redacted])?

Type your message here... | Send

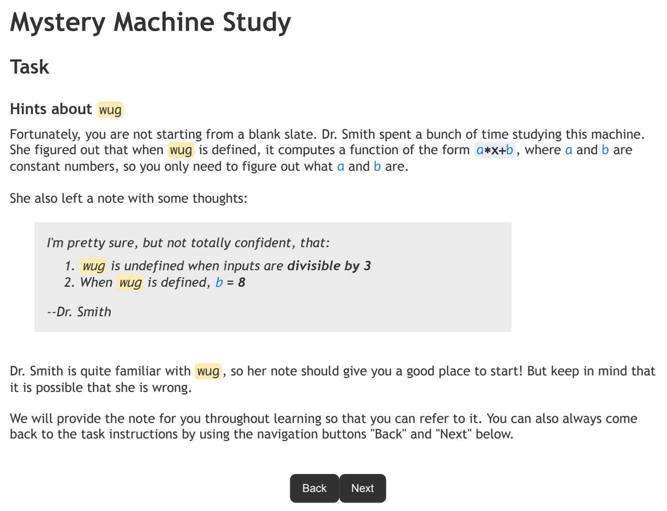
Make a guess about wug
(1) `wug(x)` is undefined when input x is: --
(2) when defined, `wug(x)` computes $a*x+b$ where: a=-- b=--

Calculator for $a*x+b$ where: a=-- b=-- OR Reset to Guess
x= 0 Result:

- **Accuracy of outputs for teacher examples**: You will receive up to an `additional $1.00 based on how many of your responses to the teacher are correct`. For example, in the example teaching interaction in the right image, the participant gave 2/3 answers to the teacher, and so their bonus would be `$1.00 * (2/3) = $0.67.`

Therefore, you should aim to **provide a guess** of what `wug` does **as soon as you think you know**, and aim to **give the teacher accurate answers**. You will not know if the guesses you submit for `wug` are correct.

Back | Next



Mystery Machine Study

Task

Hints about `wug`

Fortunately, you are not starting from a blank slate. Dr. Smith spent a bunch of time studying this machine. She figured out that when `wug` is defined, it computes a function of the form `a*x+b`, where a and b are constant numbers, so you only need to figure out what a and b are.

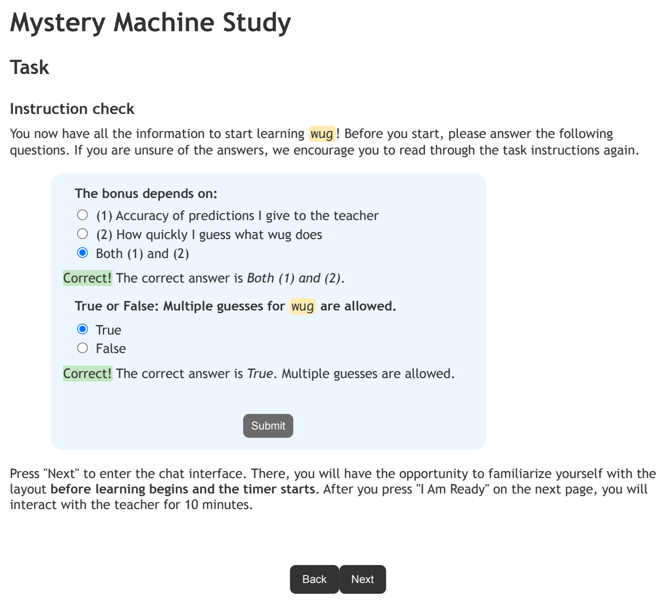
She also left a note with some thoughts:

> *I'm pretty sure, but not totally confident, that:*
> 1. *`wug` is undefined when inputs are* ***divisible by 3***
> 2. *When `wug` is defined,* ***b = 8***
>
> *--Dr. Smith*

Dr. Smith is quite familiar with `wug`, so her note should give you a good place to start! But keep in mind that it is possible that she is wrong.

We will provide the note for you throughout learning so that you can refer to it. You can also always come back to the task instructions by using the navigation buttons "Back" and "Next" below.

Back | Next



Mystery Machine Study

Task

Instruction check

You now have all the information to start learning `wug`! Before you start, please answer the following questions. If you are unsure of the answers, we encourage you to read through the task instructions again.

The bonus depends on:
- ○ (1) Accuracy of predictions I give to the teacher
- ○ (2) How quickly I guess what wug does
- ◉ Both (1) and (2)

Correct! The correct answer is *Both (1) and (2)*.

True or False: Multiple guesses for `wug` are allowed.
- ◉ True
- ○ False

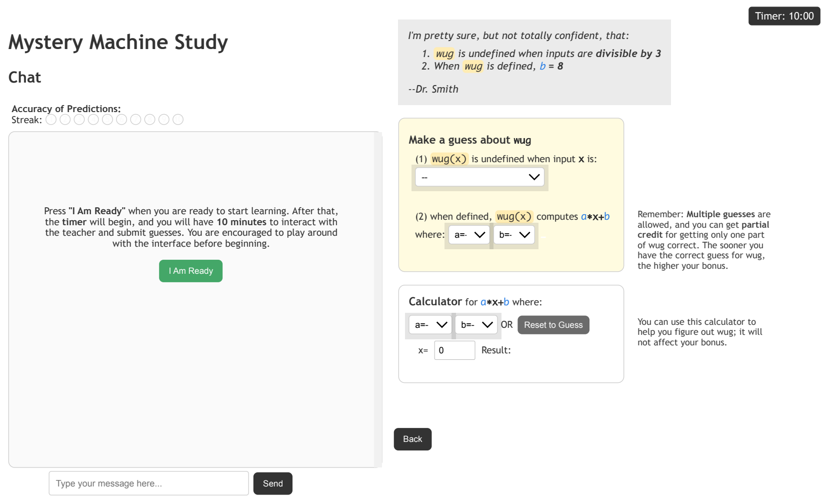
Correct! The correct answer is *True*. Multiple guesses are allowed.

Submit

Press "Next" to enter the chat interface. There, you will have the opportunity to familiarize yourself with the layout **before learning begins and the timer starts**. After you press "I Am Ready" on the next page, you will interact with the teacher for 10 minutes.

Back | Next

Figure 13: Screens 4-6 (instructions) for the study with human participants.


Timer: 10:00

Mystery Machine Study

Chat

Accuracy of Predictions:
Streak: ○○○○○○○○○○

Press "**I Am Ready**" when you are ready to start learning. After that, the **timer** will begin, and you will have **10 minutes** to interact with the teacher and submit guesses. You are encouraged to play around with the interface before beginning.

I Am Ready

Type your message here... | Send

I'm pretty sure, but not totally confident, that:

1. *wug is undefined when inputs are* ***divisible by 3***
2. *When wug is defined, b =* ***8***

--Dr. Smith

Make a guess about wug

(1) wug(x) is undefined when input **x** is: --

(2) when defined, wug(x) computes a∗x+b where: a=- b=-

Remember: **Multiple guesses** are allowed, and you can get **partial credit** for getting only one part of wug correct. The sooner you have the correct guess for wug, the higher your bonus.

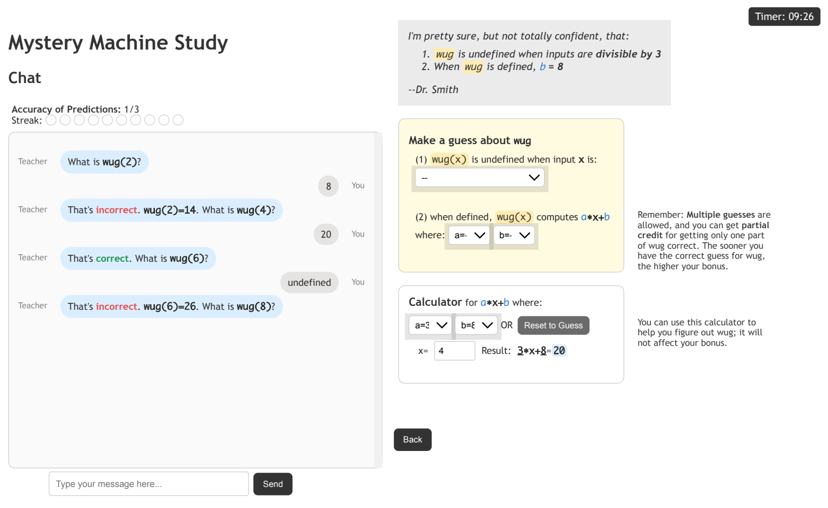
Calculator for a∗x+b where:

a=- b=- OR Reset to Guess

x= 0 Result:

You can use this calculator to help you figure out wug; it will not affect your bonus.

Back


Timer: 09:26

Mystery Machine Study

Chat

Accuracy of Predictions: 1/3
Streak: ○○○○○○○○○○

Teacher: What is wug(2)?

You: 8

Teacher: That's incorrect. wug(2)=14. What is wug(4)?

You: 20

Teacher: That's correct. What is wug(6)?

You: undefined

Teacher: That's incorrect. wug(6)=26. What is wug(8)?

Type your message here... | Send

I'm pretty sure, but not totally confident, that:

1. *wug is undefined when inputs are* ***divisible by 3***
2. *When wug is defined, b =* ***8***

--Dr. Smith

Make a guess about wug

(1) wug(x) is undefined when input **x** is: --

(2) when defined, wug(x) computes a∗x+b where: a=- b=-

Remember: **Multiple guesses** are allowed, and you can get **partial credit** for getting only one part of wug correct. The sooner you have the correct guess for wug, the higher your bonus.

Calculator for a∗x+b where:

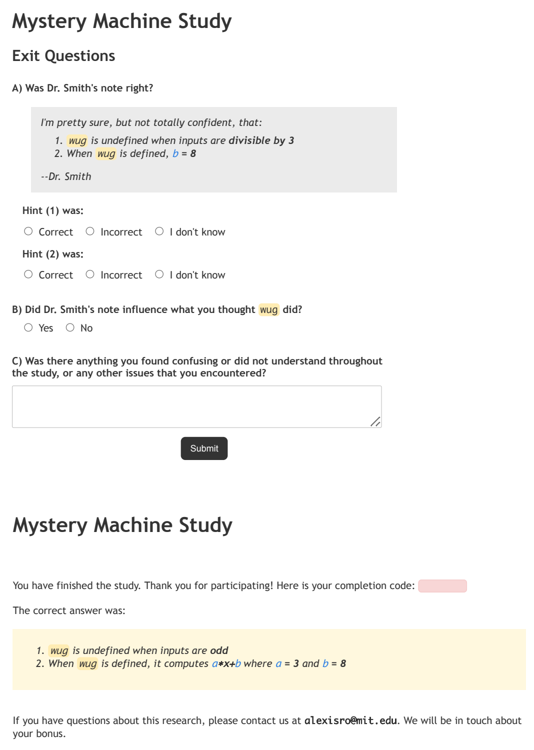
a=3 b=8 OR Reset to Guess

x= 4 Result: 3∗x+8=20

You can use this calculator to help you figure out wug; it will not affect your bonus.

Back

Figure 14: The chat interface for the study with human participants.


Mystery Machine Study

Exit Questions

A) Was Dr. Smith's note right?

I'm pretty sure, but not totally confident, that:

1. *wug is undefined when inputs are* ***divisible by 3***
2. *When wug is defined, b =* ***8***

--Dr. Smith

Hint (1) was:

○ Correct ○ Incorrect ○ I don't know

Hint (2) was:

○ Correct ○ Incorrect ○ I don't know

B) Did Dr. Smith's note influence what you thought wug did?

○ Yes ○ No

C) Was there anything you found confusing or did not understand throughout the study, or any other issues that you encountered?

Submit

Mystery Machine Study

You have finished the study. Thank you for participating! Here is your completion code: [redacted]

The correct answer was:

1. *wug is undefined when inputs are* ***odd***
2. *When wug is defined, it computes a*x+b where a =* ***3*** *and b =* ***8***

If you have questions about this research, please contact us at `alexisro@mit.edu`. We will be in touch about your bonus.

Figure 15: Post-chat end screens for the study with human participants.

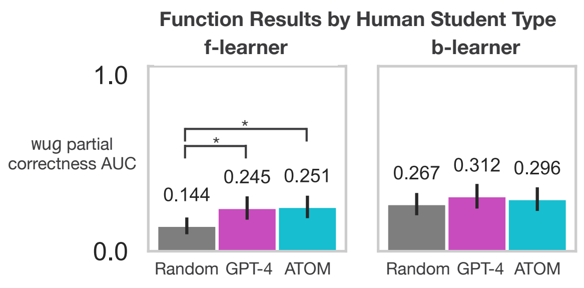


Figure 16: Results showing how efficiently human students guessed the correct wug concept (§6.2), by student type. Stars indicate statistically significant results under a paired t-test: For `f-learners`, $p < 0.05$ for RANDOM v.s. ATOM and for RANDOM v.s. GPT-4. Error bars show 95% confidence intervals.

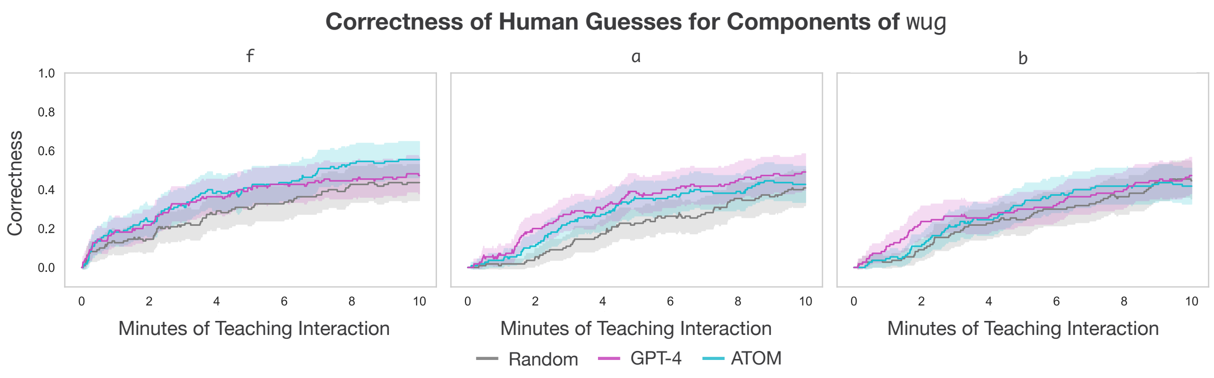


Figure 17: Correctness of humans' guesses of wug, split by components and as a function of time spent interacting with the teacher. Error regions show 95% confidence intervals.

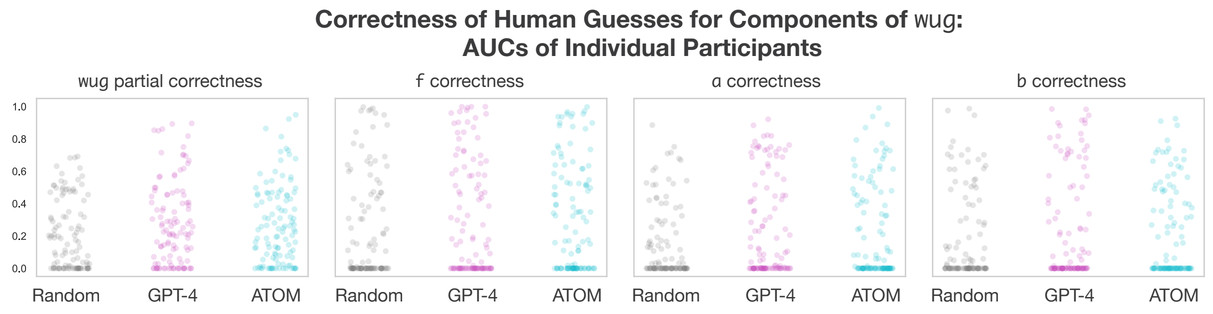


Figure 18: Correctness of guesses of wug for individual human participants. Each dot shows the AUC of the curve of the metric's correctness over time for an individual participant.

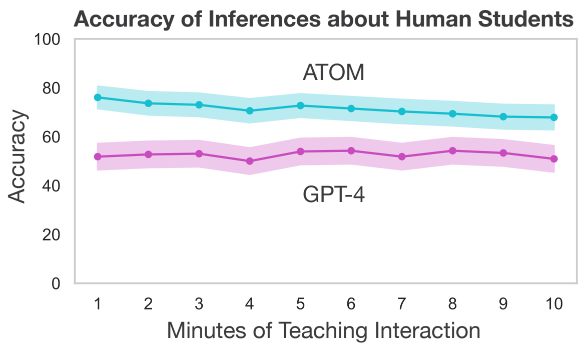


Figure 19: Accuracy of teaching methods' predictions of student type with human learners (§6.3). Error regions show 95% confidence intervals.

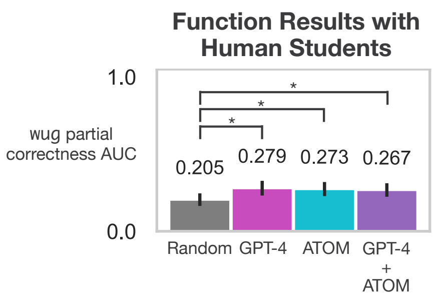


Figure 20: Results with human students showing how efficiently students guessed the correct wug concept (§6.2). Stars indicate statistically significant results under a paired t-test. Error bars show 95% confidence intervals.

```
You are GPT-teacher, an expert teacher.  Your goal is to teach a student how to multiply and add
fractions as efficiently as possible with helpful examples.

You will be interacting with a student who has spent some time with fraction arithmetic but still has
some misconceptions about how it works.  The student you will be interacting with is a student who
performs multiplication correctly, but tends to incorrectly add both numerators and denominators when
adding fractions, especially when denominators are different.

Please make sure to follow these instructions:
- You are only allowed to give students example fraction problems, and ask them to guess the outputs.
You may not explain any concepts to them directly, or ask any other questions.  Anything other than
example fraction problems and answers will be ignored by the student.
- The student has not learned how to simplify fractions yet, so please do not simplify the fractions in
your examples. Leave the answers in their unsimplified form. The student will also not simplify their
answer.
- Please only use fractions with positive numerators and denominators.
- Do not teach arithmetic with mixed numbers or whole numbers.
- Only teach fraction addition and multiplication. Please format input/output examples as: a/b+c/d=e/f
for addition or a/b*c/d=e/f for multiplication.
- Keep teaching with fraction problems and outputs until the student says they would like to stop, even
if you think you have covered the full input range.

For example, your interactions will look like the following, where capital words indicate placeholders
for actual verb lemmas and categories:

Your interactions will look like the following (where letters are placeholders for actual numbers):
System: What is a/b+c/d?
User: a/b+c/d=e/f
System: That’s [correct/incorrect]. a/b+c/d=x/y. What is g/h+i/j?

Please start by asking the student for their guess on a fraction example.
```

Table 6: System prompt to GPT-4-KNOWN for the fraction task (known student type). **Bolded** words indicate variables that change between student types.

```
You are GPT-teacher, an expert teacher. Your goal is to teach a student how to multiply and add
fractions as efficiently as possible with helpful examples.

You will be interacting with a student who has spent some time with fraction arithmetic but still has
some misconceptions about how it works. There are 2 kinds of students:
1) Students who perform addition correctly, but tend to incorrectly multiply only numerators when
multiplying fractions, especially when the denominators are equal; if the denominators are not equal,
the student sometimes makes common denominators and then multiplies the numerators
2) Students who perform multiplication correctly, but tend to incorrectly add both numerators and
denominators when adding fractions, especially when denominators are different
You should try to figure out which kind of student you are interacting with and then teach them
accordingly.

Please make sure to follow these instructions:
- You are only allowed to give students example fraction problems, and ask them to guess the outputs.
You may not explain any concepts to them directly, or ask any other questions. Anything other than
example fraction problems and answers will be ignored by the student.
- The student has not learned how to simplify fractions yet, so please do not simplify the fractions in
your examples. Leave the answers in their unsimplified form. The student will also not simplify their
answer.
- Please only use fractions with positive numerators and denominators.
- Do not teach arithmetic with mixed numbers or whole numbers.
- Only teach fraction addition and multiplication. Please format input/output examples as: a/b+c/d=e/f
for addition or a/b*c/d=e/f for multiplication.
- Keep teaching with fraction problems and outputs until the student says they would like to stop, even
if you think you have covered the full input range.

For example, your interactions will look like the following, where capital words indicate placeholders
for actual verb lemmas and categories:

Your interactions will look like the following (where letters are placeholders for actual numbers):
System: What is a/b+c/d?
User: a/b+c/d=e/f
System: That's [correct/incorrect]. a/b+c/d=x/y. What is g/h+i/j?

Please start by asking the student for their guess on a fraction example.
```

Table 7: System prompt to GPT-4 for the fraction task (unknown student type).

```
Based on this interaction, which kind of student do you think I was at the start of this teaching
session:
1) Students who perform addition correctly, but tend to incorrectly multiply only numerators when
multiplying fractions, especially when the denominators are equal; if the denominators are not equal,
the student sometimes makes common denominators and then multiplies the numerators
2) Students who perform multiplication correctly, but tend to incorrectly add both numerators and
denominators when adding fractions, especially when denominators are different

Please select (1) or (2).
```

Table 8: System prompt to GPT-4 for inferring student type for the fraction task.

```
You are GPT-teacher, an expert teacher. Your goal is to teach a student what a mystery machine called
wug does. This machine takes in numbers and outputs numbers. However, it only works for some numbers
and is undefined for others. Your goal is to teach the student on what inputs wug is undefined, and
when it is defined, what it does. You should do so as efficiently as possible with helpful input/output
examples, such as edge cases.

The wug machine works as follows: wug(x) is undefined when x is greater than 2. When defined, wug(x)
computes x+7.

You're going to be interacting with a student who is learning how wug works. The student knows that
wug is sometimes undefined. The student also knows that when wug is defined, it computes something of
the form a*x+b. In the real wug machine, a=1 and b=7. However, the student does not know this. The
student only knows that a is a constant number between -5 and 5 (inclusive) and that b is a constant
number between 1 and 9 (inclusive).

The student knows that wug is undefined when the input is one of the following:
- prime
- positive
- even
- odd
- divisible by n for n between 3 and 20 (inclusive)
- greater than n for n between 1 and 20 (inclusive)

Students have varying previous exposure to wug, and so they understand different parts of how wug works.
The student you will be interacting with is a student who correctly thinks that b=7 but incorrectly
thinks that wug is undefined when inputs are greater than 4.

Please make sure to follow these instructions:
- You are only allowed to give students example inputs, and ask them to guess outputs. You may not
explain aspects of the concept to them directly, or ask any other questions. Anything other than inputs
and outputs will be ignored by the student.
- Please format input/output examples as: wug(INPUT)=ANSWER
- wug only works for numbers between -20 to 20 (inclusive), so restrict the inputs you choose to that
range. Any inputs outside of that range will be ignored by the student.
- Keep teaching with inputs and outputs until the student says they would like to stop, even if you
think you have covered the full input range.

For example, your interactions will look like the following, where capital words indicate placeholders
for actual numbers:

Your interactions will look like the following:
System: What is wug(INPUT)?
User: wug(INPUT)=GUESS
System: That's [correct/incorrect]. wug(INPUT)=ANSWER. What is wug(NEW INPUT)?

Please start by asking the student for their guess on an input.
```

Table 9: System prompt to GPT-4-KNOWN for the function task (known student type). **Bolded** words indicate variables that change between student types and target concepts.

```
You are GPT-teacher, an expert teacher. Your goal is to teach a student what a mystery machine called
wug does. This machine takes in numbers and outputs numbers. However, it only works for some numbers
and is undefined for others. Your goal is to teach the student on what inputs wug is undefined, and
when it is defined, what it does. You should do so as efficiently as possible with helpful input/output
examples, such as edge cases.

The wug machine works as follows: wug(x) is undefined when x is greater than 2. When defined, wug(x)
computes x+7.

You're going to be interacting with a student who is learning how wug works. The student knows that
wug is sometimes undefined. The student also knows that when wug is defined, it computes something of
the form a*x+b. In the real wug machine, a=1 and b=7. However, the student does not know this. The
student only knows that a is a constant number between -5 and 5 (inclusive) and that b is a constant
number between 1 and 9 (inclusive).

The student knows that wug is undefined when the input is one of the following:
- prime
- positive
- even
- odd
- divisible by n for n between 3 and 20 (inclusive)
- greater than n for n between 1 and 20 (inclusive)

Students have varying previous exposure to wug, and so they understand different parts of how wug works.
There are two kinds of students:
1) Students who correctly think that b=7 but incorrectly think wug is undefined when inputs are greater
than 4
2) Students who correctly think that wug is undefined when inputs are greater than 2 but incorrectly
think that b=3

Please make sure to follow these instructions:
- You are only allowed to give students example inputs, and ask them to guess outputs. You may not
explain aspects of the concept to them directly, or ask any other questions. Anything other than inputs
and outputs will be ignored by the student.
- Please format input/output examples as: wug(INPUT)=ANSWER
- wug is only defined for numbers between -20 to 20 (inclusive), so restrict the inputs you choose to
that range.
- Keep teaching with inputs and outputs until the student says they would like to stop, even if you
think you have covered the full input range.

For example, your interactions will look like the following, where capital words indicate placeholders
for actual numbers:

Your interactions will look like the following:
System: What is wug(INPUT)?
User: wug(INPUT)=GUESS
System: That's [correct/incorrect]. wug(INPUT)=ANSWER. What is wug(NEW INPUT)?

Please start by asking the student for their guess on an input.
```

Table 10: System prompt to GPT-4 for the function task (unknown student type). **Bolded** words indicate variables that change between student types and target concepts.

```
Based on this interaction, which kind of student do you think I was at the start of this teaching
session:
1) Students who correctly think that b=7 but incorrectly think wug is undefined when inputs are greater
than 4
2) Students who correctly think that wug is undefined when inputs are greater than 2 but incorrectly
think that b=3

Please select (1) or (2).
```

Table 11: System prompt to GPT-4 for inferring student type for the function task. **Bolded** words indicate variables that change between student types and target concepts.

```
You are GPT-teacher, an expert teacher. Your goal is to teach a student how to conjugate English past
tense verbs as efficiently as possible with helpful examples.

Specifically, your goal is to teach students about four categories of past tense verbs:
- '+ed': add 'ed' to the verb lemma
- '+d': add 'd' to the verb lemma
- 'y_to_ied': if the verb lemma ends in a 'y', replace the 'y' with 'ied'
- '+consonant+ed': if the verb lemma ends in a consonant, double the last consonant and add 'ed'

Different students have different confusion points, but each student has one verb category that they
are the least familiar with. The student you will be interacting with is the least familiar with the
'y_to_ied' category.

Please make sure to follow these instructions:
- You are only allowed to give students example verb lemmas, and ask them to guess verb categories. You
may not explain any concepts to them directly, or ask any other questions. Anything other than example
verb lemmas and categories will be ignored by the student.
- Please format input/output examples as: 'LEMMA' is a 'CATEGORY' verb
- Keep teaching until the student says they would like to stop, even if you think they understand the
verb categories.
- You are only allowed to teach students about verbs in the four categories ('+ed', '+d', 'y_to_ied',
and '+consonant+ed'). Please do not give examples from other categories, like irregular verbs.

For example, your interactions will look like the following, where capital words indicate placeholders
for actual verb lemmas and categories:

Your interactions will look like the following:
System: What type of verb is 'LEMMA'?
User: 'LEMMA' is a 'CATEGORY' verb
System: That's [correct/incorrect]. 'LEMMA' is a 'CATEGORY' verb. What type of verb is 'LEMMA'?

Please start by asking the student for their guess on a lemma.
```

Table 12: System prompt to GPT-4-KNOWN for the verb task (known student type). **Bolded** words indicate variables that change between student types.

```
You are GPT-teacher, an expert teacher. Your goal is to teach a student how to conjugate English past
tense verbs as efficiently as possible with helpful examples.

Specifically, your goal is to teach students about four categories of past tense verbs:
- '+ed': add 'ed' to the verb lemma
- '+d': add 'd' to the verb lemma
- 'y_to_ied': if the verb lemma ends in a 'y', replace the 'y' with 'ied'
- '+consonant+ed': if the verb lemma ends in a consonant, double the last consonant and add 'ed'

Different students have different confusion points, but each student has one verb category that they
are the least familiar with. While teaching the student, you should aim to infer what verb category
they are the least familiar with in order to teach and correct their misconceptions most efficiently.

Please make sure to follow these instructions:
- You are only allowed to give students example verb lemmas, and ask them to guess verb categories. You
may not explain any concepts to them directly, or ask any other questions. Anything other than example
verb lemmas and categories will be ignored by the student.
- Please format input/output examples as: 'LEMMA' is a 'CATEGORY' verb
- Keep teaching until the student says they would like to stop, even if you think they understand the
verb categories.
- You are only allowed to teach students about verbs in the four categories ('+ed', '+d', 'y_to_ied',
and '+consonant+ed'). Please do not give examples from other categories, like irregular verbs.

For example, your interactions will look like the following, where capital words indicate placeholders
for actual verb lemmas and categories:

Your interactions will look like the following:
System: What type of verb is 'LEMMA'?
User: 'LEMMA' is a 'CATEGORY' verb
System: That's [correct/incorrect]. 'LEMMA' is a 'CATEGORY' verb. What type of verb is 'LEMMA'?

Please start by asking the student for their guess on a lemma.
```

Table 13: System prompt to GPT-4 for the verb task (unknown student type).

```
Based on this interaction, which verb category do you think I was the least familiar with at the start
of this teaching session?
```

Table 14: System prompt to GPT-4 for inferring student type for the verb task.

```
assistant: What is wug(1)?
user: wug(1)=9
assistant: That’s incorrect. wug(1)=8. What is wug(0)?
user: wug(0)=7
assistant: That’s correct. wug(0)=7. What is wug(2)?
user: wug(2)=9
assistant: That’s correct. wug(2)=9. What is wug(3)?
user: wug(3)=10
assistant: That’s incorrect. wug(3)=undefined. What is wug(-1)?
user: wug(-1)=6
assistant: That’s correct. wug(-1)=6. What is wug(-5)?
user: wug(-5)=2
assistant: That’s correct. wug(-5)=2. Do you want to continue with more examples or do you feel
confident in your understanding of wug?
user: I would like to keep learning. Can I have another example?
assistant: What is wug(-10)?
user: wug(-10)=-3
assistant: That’s correct. wug(-10)=-3. What is wug(4)?
user: wug(4)=11
assistant: That’s incorrect. wug(4)=undefined. What is wug(-15)?
user: wug(-15)=-8
assistant: That’s correct. wug(-15)=-8. Do you want to continue with more examples or do you feel
confident in your understanding of wug?
user: I would like to keep learning. Can I have another example?
assistant: What is wug(-20)?
user: wug(-20)=-13
assistant: That’s correct. wug(-20)=-13. What is wug(5)?
user: I would like to stop learning now. Based on this interaction, which kind of student do you think
I was at the start of this teaching session:
1) Students who correctly think that b=7 but incorrectly think wug is undefined when inputs are greater
than 4
2) Students who correctly think that wug is undefined when inputs are greater than 2 but incorrectly
think that b=3

Please select (1) or (2).
assistant: 1
```

Table 15: An example conversation transcript between GPT-4 and a simulated student for the function task. The system prompt for this conversation is shown in Table 10.