Decomposed Inductive Procedure Learning: Learning Academic Tasks with Human-Like Data Efficiency

Daniel Weitekamp,[1] **Christopher MacLellan,** [1] **Erik Harpstead,** [2] **Kenneth Koedinger,**[2]
[1] Georgia Institute of Technology, Atlanta, GA 30332
[2] Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

Human learning relies on specialization—distinct cognitive mechanisms working together to enable rapid learning. In contrast, most modern neural networks rely on a single mechanism: gradient descent over an objective function. This raises the question: might human learners' relatively rapid learning from just tens of examples instead of tens of thousands in data-driven deep learning arise from our ability to use multiple specialized mechanisms of learning in combination? We investigate this question through an ablation analysis of inductive human learning simulations in online tutoring environments. Comparing reinforcement learning to a more data-efficient 3-mechanism symbolic rule induction approach, we find that decomposing learning into multiple distinct mechanisms significantly improves data efficiency, bringing it in line with human learning. Furthermore, we show that this decomposition has a greater impact on efficiency than the distinction between symbolic and subsymbolic learning alone. Efforts to align data-driven machine learning with human learning often overlook the stark difference in learning efficiency. Our findings suggest that integrating multiple specialized learning mechanisms may be key to bridging this gap.

A key idea within the learning sciences, popularized by Anderson's ACT-R theory (2013) and expanded upon by others (Koedinger, Corbett, & Perfetti, 2012), is that human performance is enabled by independent knowledge components—individual facts, skills, or principles—that must be understood and retained to exhibit mastery of higher-level capabilities. For instance, addition tables from 1 to 10 comprise $\frac{10*(10+1)}{2} = 55$ facts. More complex procedures, such as adding two large numbers, may require several additional skills like aligning numbers, adding over columns, and carrying the tens-digits of partial sums. More advanced capabilities require mastery of even more interdependent knowledge components which may build upon these.

Intelligent tutoring systems (ITS) are educational technologies that mimic one-on-one tutoring interactions by providing highly adaptive step-by-step instructional support designed to aid the acquisition of unmastered knowledge components. When students practice skills in these controlled learning environments, their rate of learning proves to be remarkably quick and astonishingly consistent between individuals (Koedinger, Carvalho, Liu, & McLaughlin, 2023). Data from ITSs (Koedinger et al., 2010) illustrate that students typically master skills in about a dozen practice opportunities or fewer—orders of magnitude faster than modern data-driven machine learning (ML) approaches, such as reinforcement learning, which relies on gradient-descent and often require tens of thousands to millions of examples.

This work investigates three learning mechanisms that have emerged from efforts to simulate humans' inductive learning of academic tasks. Simulated learner systems like Sierra (VanLehn, 1990), SimStudent (Matsuda, Cohen, & Koedinger, 2015), the Apprentice Learner (AL) architecture (Maclellan, Harpstead, Patel, & Koedinger, 2016), and AI2T (Weitekamp, Harpstead, & Koedinger, 2024) have successfully learned dozens of domains, acquiring skills directly from ITSs and similar environments, and in some cases, directly from human instruction. These systems have not only replicated the remarkable rate of learning observed in humans (Maclellan et al., 2016; Weitekamp, Harpstead, MacLellan, Rachatasumrit, & Koedinger, 2019), but also produce patterns of error that mirror those found in student data (VanLehn, 1990; Weitekamp, Ye, Rachatasumrit, Harpstead, & Koedinger, 2020).

The central question of this work is: why have cognitive systems succeeded at replicating rapid human learning, while neural network approaches lag behind by several orders of magnitude? While symbolic learning mechanisms in simulated learners certainly play a role, we show that the key factor to their efficiency is the integration of multiple functionally distinct learning systems. Through an ablation analysis, we compare a single learning mechanism (either neural reinforcement learning or a symbolic learning approach) with progressively more decomposed learning that uses two, and finally, three mechanisms, as used in prior simulated learners. We coin the name *Decomposed Inductive Procedure Learning (DIPL)* to refer to this common multi-mechanism approach.

Across two ITS tasks, we show that each stage of ablation—from a single mechanism learning to DIPL's 3-mechanism learning—yields several orders of magnitude of learning efficiency improvement. We hypothesize that these gains arise from how different mechanisms, each with a well-defined role, simplify error attribution and reduce the total computational complexity of knowledge induction. This work highlights the vast benefits that theory-driven cognitive modeling can offer to the fields of cognitive science and machine learning. In an odd divergence from historical definitions (VanLehn, Ohlsson, & Nason, 1994), large language models (LLMs) have inspired a wave of so-called "simu-

In D. Barner, N.R. Bramley, A. Ruggeri and C.M. Walker (Eds.), *Proceedings of the 47th Annual Conference of the Cognitive Science Society*

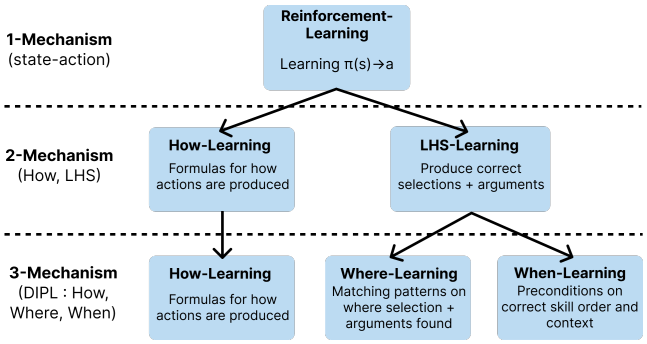

Figure 1: Decomposition from 1-mechanism learning, like RL, that maps states to actions to DIPL's 3-mechanism learning. A 2-mechanism system bridges the difference and uses *how-learning* but combines *where-* and *when-learning*.

lated learners" that generate responses but do not actually learn (Käser & Alexandron, 2024). In response, we highlight the fundamental advantage of a cognitive theory-based approach over simple gradient descent based learning in artificial neural networks. Our direct comparison between reinforcement learning and simulated learners suggests that an essential component of this advantage lies in how several specialized mechanisms of learning cooperate to achieve human-like learning efficiency.

Related Work

Large Language Models

Artificial neural systems have come a long way in performing academic tasks such as mathematics. For instance, LLMs have succeeded on challenging word problems (Bubeck et al., 2023) from datasets like MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021). In 2023, GPT models trained step-by-step have demonstrated 78% accuracy on the MATH dataset (Lightman et al., 2023), and yet higher accuracies have been achieved with several recent models on several benchmarks (Team et al., 2024). However, since these models were trained on a quantity of learning experiences many orders of magnitude greater than a human would experience in a lifetime, their decidedly data-driven developmental process offers little insight into human learning. Other misalignments of deep learning with human learning include the phenomena of catastrophic forgetting (McCloskey & Cohen, 1989), where new training causes models to forget capabilities learned from previous training instances. Additionally, strong evidence has emerged showing that LLM's problem-solving capabilities are largely memorized and not in fact a result of effective generalization from data. For instance, Mirzadeh et. al (2024) show that LLMs' performance drastically declines when questions from the GSM8K dataset have their numerical values replaced with different values or if additional distractor clauses are added.

Speed-up Learning in Cognitive Systems

A shortcoming of many attempts to model learning in symbolic cognitive systems has been the over-reliance on knowledge engineering and "learning" by planning with hard-coded, and often domain specifc, prior knowledge representations. Much work in logic programming (Manhaeve, Dumancic, Kimmig, Demeester, & De Raedt, 2018) and a great deal of early work in cognitive architectures, like SOAR (Laird, 2019) and ACT-R (Ritter, Tehranchi, & Oury, 2019), can be held to this criticism. No doubt, models of learning ought to represent and build upon prior knowledge. Yet, many cognitive systems neglect important aspects of learning by starting with knowledge representations that are hard-coded to plan toward target tasks. Laird, Lebiere, and Rosenbloom (2017) have presented the view that learning is only "a side effect of performance." This claim evokes the principle of learning-by-doing, yet is greatly misaligned with the realities of learning in an academic setting. "Learning" for these systems is typically characterized as a cognitive speed-up achieved by repurposing prior knowledge to shortcut later computational effort. In an academic setting, one would not consider a human student to have learned if they could "perform" without errors in advance of instruction. The limited focus of many cognitive systems on error-free speed-up learning (Neves, 1985) overlooks this important inductive component of human knowledge formation that enables the rapid, yet initially error-prone acquisition of entirely new capabilties through instruction and practice.

Inductive Simulated Learners

VanLehn's (1990) Sierra is an early inductive simulated learner (SL). Experiments with Sierra demonstrated that inductive learning underlies the acquisition of early mathematical skills, replicating and explaining more mistakes in subtraction problem-solving datasets than prior case-by-case analyses. Later efforts with SimStudent and the Apprentice Learner (AL) architecture have empirically reproduced student learning curves across dozens of ITS domains (MacLellan & Koedinger, 2020). Instead of fitting to student data, these systems learn to solve problems from the same ITS interactions human students experience, generating step-by-step solutions that improve with practice. (Maclellan et al., 2016; Weitekamp et al., 2020). These simulated learners induce production rules from ITS examples and correctness feedback—initially producing errors, but are rapidly restructured towards mastery through supervised practice. AI2T (Weitekamp et al., 2024) extends this principle, letting untrained users teach it interactively. Notably, half of AI2T users successfully trained it to exhibit 100% correct and complete behavior on mathematical tasks with small user-selected training sequences of just 14-21 problems.

Decomposed Inductive Procedure Learning

SimStudent and the Apprentice Learner induce production rules using a combination of three learning mechanisms that

independently determine *how* actions are taken, *where* it is possible to take actions, and *when* (i.e. under what circumstances and in what order) those actions should be applied to execute a target behavior. Sierra and AI2T include a fourth mechanism for inducing the hierarchical *process* by which higher-level tasks are divided into subtasks. This *process-learning* mechanism arranges production rules into hierarchical task networks (Erol, Hendler, & Nau, 1994) that control how high-level tasks are broken down into partially ordered subtasks. Our notion of Decomposed Inductive Procedure Learning (DIPL) encompasses both these 3- and 4-mechanism approaches. However, for our evaluations, we focus on 3-mechanism DIPL, which includes only *how-learning*, *where-learning*, and *when-learning*.

While multi-mechanism learning approaches, such as the actor-critic paradigm (Konda & Tsitsiklis, 1999), are commonplace in AI, DIPL's learning mechanisms cooperate in a uniquely modular, localized fashion. Unlike most actor-critic methods, DIPL's learning mechanisms do not apply globally; instead, each is instantiated separately for every learned skill (i.e., production rule). Within the induction of a single skill, these mechanisms cooperate in such a way that each mechanism simplifies learning for the others. Rather than relying on a single global mechanism like gradient descent, each instance of each individual learning mechanism serves a distinct, well-defined role—acquiring specific generalizations for individual skills. Each skill, in turn, is built up from these induced pieces and is responsible for performing particular kinds of actions. As a model of learning, DIPL's division of capabilities into individual skills aligns with the notion of knowledge components, expressed as production rules that are refined over time within an evolving expert system.

How-Learning

How-learning determines *how* skills apply actions using an abductive process. Prior simulated learners have generally implemented *how-learning* with a search process that composes primitive domain-general prior-knowledge functions (like arithmetic functions and string operations) to reproduce observed actions. This search typically produces multiple candidate compositions that reproduce the worked example, some of which may be incorrect. Among the candidate compositions that reproduce a worked example, the most parsimonious (having the fewest operations and arguments) is chosen. The chosen explanation is generalized by replacing the constants in the grounded composition with variables. For instance, `OnesDigit(7+5)` in Figure 2 may be generalized to depend upon two argument variables `Arg0=Var(TextField)` and `Arg1=Var(TextField)` which match to any TextField type interface elements.

Prior implementations of *how-learning* have composed primitive functions in an iterative-deepening fashion (Matsuda et al., 2015). At each deepening the composition depth is increased by composing primitive functions with prior compositions, for instance, two depth=1 compositions Add(a,b), and Subtract(a,b) could be composed with Divide(a,b) to produce a depth=2 composition Divide(Add(a,b), Subtract(c,d)). This search process explodes combinatorially, so search depths are typically limited to 1 to 3. A method that we have come to call Set Chaining optimizes this search considerably. Set Chaining executes primitive prior knowledge functions in waves, using every combination of unique values from the previous wave as arguments in the next. By keeping a lightweight record of every way each unique value is produced in each wave, compositions can be built by tracing back from the goal value, once it is found. This method cuts down on combinatoric search and is amenable to multithreaded implementations.

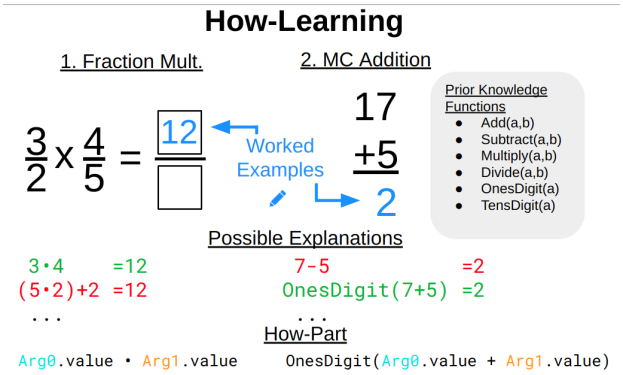

Figure 2: *How-learning* explanations for worked examples in fractions and multi-column addition. Of the several explanations, some may be correct (green) or incorrect (red).

Prior work has also used instructional annotations, like the arguments of the true composition (Matsuda et al., 2015) and even natural language hints (Weitekamp, Rachatasumrit, Wei, Harpstead, & Koedinger, 2023), to guide the explanation process and reduce the amount of combinatorial search.

Where-learning

Where-Learning discovers matching patterns to determine where skills can be applied. While *how-learning* is responsible for producing the operational generalizations within skills, *where-learning* builds spatial generalizations that specify their applicable contexts. In a multi-column addition task (Figure 3), *where-learning* could generalize a skill for computing the one's digit of a partial sum so that it applies across columns. The patterns it learns are expressed using argument variables (e.g., `Arg0` and `Arg1` from the *how-learning* example above), along with a single selection variable (e.g., `Sel=Var(TextField)`) that matches the interface element to act upon.

The learned *where-part* pattern consists of a logical statement, expressing necessary conditions and spatial relationships between variables. The pattern produced from an initial worked example is typically highly constrained and may only bind to a limited set of selections and arguments. Subsequent examples can generalize the *where-part* by generalizing or removing the relations that comprise it. *How-learning* has a supporting role in identifying the sets of arguments that

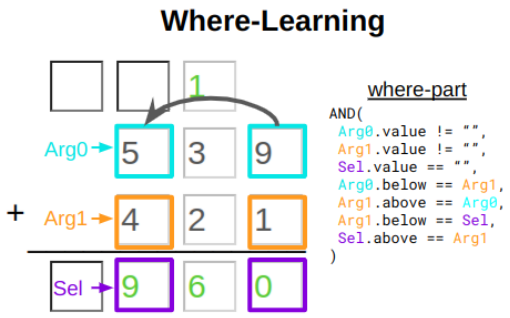

Figure 3: An example of a *where-part* pattern generalized to act across columns in multi-column addition.

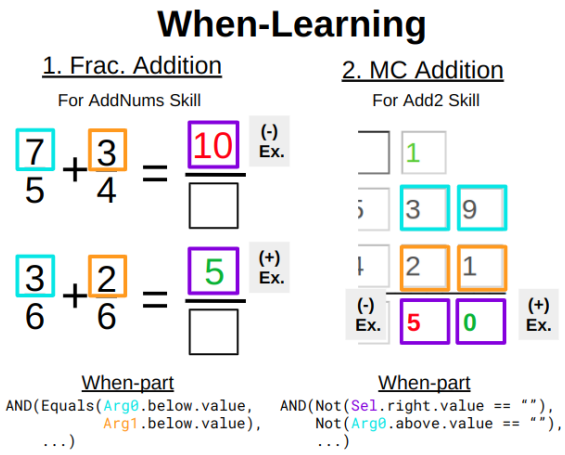

Figure 4: Positive and negative examples of an AddNum skill in fraction addition, and an Add2 skill in Multi-Column addition. Partial *when-parts* that accept the positive examples but reject the negative example are shown expressed with relational features generated by relative featurization.

where-learning generalizes from. *How-learning* attempts to explain each new worked example using existing skills' *how-part* compositions. If there are any candidate explanations, the one with arguments that would make the minimal change to an existing skill (quantified by a score that measures structure similarity) is used for *where-part* generalization. Otherwise, *how-learning* generates a new skill from the example.

Where-part generalizations can also match to neighbors and parents of the selections and arguments, to identify their placement within hierarchical representations. For instance, Li et. al. (Li, Matsuda, Cohen, & Koedinger, 2015) employed representation learning in SimStudent, to learn and match to hierarchies of expressions, terms, coefficients, and variables within algebra equations.

When-Learning

When-learning identifies the contexts and order in which skills should be applied. The *when-part* of a skill consists of pre-conditions that define its applicability. *When-learning* is typically implemented using binary classification methods that output symbolic relational expressions. Prior work has used inductive logic programming, decision trees, and incremental concept learning approaches (Matsuda et al., 2015; Maclellan et al., 2016). In any given state, the learned *when-part* preconditions distinguish whether a particular candidate application of a skill matched by the *where-part* pattern should be applied.

When-learning uses positive and negative examples of candidate skill applications in particular problem states. *Where-part* processing assists *when-learning* by associating each example triple of (state, action, reward) with a particular skill, selection, and set of arguments. This allows *when-learning* to construct *when-parts* as variablized concepts consisting of relations that express features in the state as they relate to the selection (e.g. Sel) and arguments (e.g. Arg0, Arg1) instead of as they relate to particular interface elements. This enables generalization across spatially distinct instances of the same skill. For instance, the *when-part* in the fraction example in Figure 4 includes a literal relation `Equals(Arg0.below.value, Arg1.below.value)` which references the values of the elements below `Arg0` and `Arg1`.

Prior work has used FOIL (Quinlan & Cameron-Jones, 1995), an inductive logic programming method, to learn concepts with relational constraints like those above. FOIL searches for and combines new literals like `Below(Arg0, A)` expressed in terms of source variables like `Arg0`, or invented variables like `A`. Searching for such statements can be computationally demanding. In this work we introduce a streamlined means of restating features in the problem state relative to the selection and argument variables. The shortest path through adjacency relationships is found between interface elements using the Bellman-Ford algorithm (Bellman, 1958). Then each feature in the state is relabeled with the shortest path from the selection or arguments (in the dot-notation of Figure 4). This method of *relative featurization* also allows us to keep the *when-learning* classifier independent of relational feature generation.

Decomposing from RL to DIPL

Reinforcement Learning (RL) learns policies $\pi(s) \rightarrow a$, directly or indirectly, that map states to actions to maximize reward over task episodes. RL can maximize overall reward in environments where feedback signals are delayed over states, or even when there are non-deterministic actions. As RL algorithms have evolved and come to rely largely on deep learning, they have shown successes at challenging games (Mnih et al., 2013) and robotics tasks (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017). They have even become a go-to choice in deterministic procedural domains that require symbolic manipulation, such as theorem proving (Kaliszyk, Urban, Michalewski, & Olšák, 2018), and for learning mathematical tasks such as geometry (Xiao & Zhang, 2023), and early K-12 mathematics like fractions and long arithmetic (Poesia, Dong, & Goodman, 2021). However, these deep learning approaches typically require at least thousands of examples and generally learn by attempting tasks in specially

formatted environments with pre-specified action spaces.

By comparison, DIPL-based induction is about as data-efficient as human learning (Maclellan et al., 2016) and does not require a pre-specified action or state space (MacLellan & Gupta, 2021). New "actions" are learned through the induction and generalization of skills with *how-* and *where-learning*. *When-learning* then determines in what order and contexts those skills should be applied. By analogy to RL's choice of actions via a global policy $\pi(s) \rightarrow a$ DIPL generates actions with multiple symbolic pieces $When(s, Where(s)) \rightarrow How(Where(s)) = a$.

Mathematical and algorithmic complexities notwithstanding, deep RL generally relies on gradient descent to tune its behaviors. In our ablation analysis we will decompose from RL's 1-mechanism learning to DIPL's 3-mechanism learning by systematically introducing additional learning mechanisms. Between RL and DIPL a 2-mechanism system employs *how-learning*, but only a single Left-hand-side (LHS) learning mechanism is used to generate correct sets of selections and arguments for producing actions.

Task Domains

We build on the RL gym environments used by MacLellan and Gupta (2021) for two different ITS domains: (1) A fraction arithmetic tutor (Figure 5) that randomly selects among: adding same denominator fractions, different denominator fractions, and multiplying fractions and (2) a multi-column arithmetic tutor that teaches 3-digit addition (Figures 3,4). In both domains, agents can request worked examples (demos), and receive immediate reward signals (1 for correct, -1 for incorrect) on attempted actions. Since action spaces consist only of, checking boxes, placing numbers in fields, or pressing the 'done' button, there are finite primitive actions for an RL system to select from.

In the fractions tutor, agents must perform the correct fraction arithmetic procedure step-by-step (multiplying, adding, or converting then adding) based on the two starting fractions and the operator. In the RL-Gym wrapper for this environment, the agent is able to fill in each of 6 number fields with the numbers 1-450, fill in the 'check_convert' field with an "x" or press the done button, for a total of 2,702 unique actions. The multi-column addition domain (see Figures 3, 4) has 7 fields that can be filled with the digits 0-9, plus a done button for a total of 71 unique actions. In this domain, the agent must compute the sum of two 3-digit numbers by computing each partial sum in right-to-left order by placing the ones digit and then carrying the tens-digit when necessary. In both domains, the state is encoded into a vector with 0.0 or 1.0 representing whether each element is present using one-hot encoding (size 2,000 in fractions and 240 in multi-column addition). The one-hot encoding maps each unique interface element-attribute pair to a slot in the state vector.

The non-RL-based agents instead experience the state in its original object-based representation, where each object has a unique identifier, type, position, shape, and value. Additionally, no predefined action space is provided to these agents. Instead, they learn how to produce actions by applying *how-learning* to the on-demand worked example demos. These agents are instantiated with the primitive domain-general prior knowledge functions necessary to compose *how-parts* for each task: `Add(a,b)` and `Multiply(a,b)` for fractions and `OnesDigit(a)`, `TensDigit(a)`, `Add3(a,b,c)`, `Add(a,b)` for multi-column addition. In multi-column addition, extraneous *how-learning* explanations are common, so we aid *how-learning* by annotating each demo with its arguments. We do not provide these annotations for fractions. In the fractions domain, we provided a single feature function `Equals(a,b)`, enabling the agent to identify equal values, which is necessary for learning to check for equal denominators.

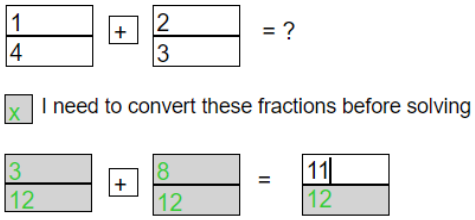

Figure 5: Fraction arithmetic tutoring system for teaching multiplying and adding fractions. Learner insert an 'x' to the 'check_convert' field if the fraction must first be converted.

Ablation Analysis

We apply two RL approaches: an off-policy Deep-Q-Network (DQN) model (Mnih et al., 2015) and an on-policy Proximal Policy Optimization (PPO) model (Schulman et al., 2017). PPO has become a popular RL approach for its relative stability, data-efficiency, and consistent convergence without hyperparameter tuning. We additionally train agents with or without automatically provided worked examples. In the latter case (indicated by "+Demos"), each incorrect action is followed by training on the current step's demo worked example. This mimics the capability of DIPL-based simulated learners to request demos when no next action can be produced. Unfortunately, this method only works with the DQN models, as there is no simple method for training on-policy methods like PPO with actions not produced by its current policy. All models were implemented using OpenAI's stable baselines library and were trained for 500,000 timesteps. For a symbolic comparison, we additionally train a decision tree using the "+Demos" training modality.

Our 2-mechanism model uses a Set Chaining *how-learning* mechanism to learn individual skills, but only a single Left-Hand-Side (LHS) learning mechanism that predicts where and when those skills should be applied. The LHS-learning uses a decision tree as a multi-class classifier that predicts the selection and arguments for the correct next action from features of the problem state.

Finally, we utilize a DIPL-base agent via our re-implementation of the Apprentice Learner (AL) Architecture. Set Chaining is used for *how-learning*. *Where-learning* uses

		Fractions	MC Addition
1-mech	PPO	Not Converge	30,642
	DQN+Demos	11,315	9,496
	DT+Demos	1,944	7,816
2-mech	How+LHS	17	270
3-mech	DIPL (no rel. feat.)	33	38
	DIPL	20	19
	Human Data	∼9-14	N/A

Table 1: Number of problems before $< 10\%$ average error.

a simple implementation that only recalls sets of selections and arguments from past examples. Finally, *when-learning* is achieved with a decision tree. We train both with and without relative featurization to illustrate the effects of utilizing *where-part* processing in *when-learning*.

Results

Our results are summarized in Table 1 and select learning curves are shown in Figure 6. The DQN models converged only in the "+Demos" condition. PPO successfully converged only in multi-column addition. Among the RL methods, the "DQN + demos" approach achieved the best data efficiency up to the $<10\%$ error mastery point requiring about 10,000 problems in each task. Decision trees were more data-efficient than the RL methods, but still required 1,944 problems in fractions and 7,816 problems in multi-column addition. The 2-mechanism model showed a dramatic improvement in data efficiency, taking just 17 problems to master fractions but 270 for multi-column addition. The 3-mechanism DIPL model by contrast mastered both in 20 problems or less. Turning off relative featurization resulted in a 13-19 problem deterioration of data efficiency. We additionally include human data collected by Patel, Liu, and Koedinger (2016) using the fractions ITS. We shift this data by 6 problems to align the initial 30% error rate exhibited by the most efficient SL, leading to an adjusted mastery intercept of around 9-14 problems. The DIPL agents and humans show similar initial learning rates, but the DIPL agents improve more rapidly beyond the mastery threshold up to less than 1% error after about 130 problems.

Discussion

Data-Efficiency Our 1-mechanism models cover only a small number of RL training approaches yet are fairly representative of the data-efficiency of RL. One contribution to inefficiency is that the RL action models consist of picking numbers instead of computing them (similar to how LLMs approach math). However, in similar experiments in which RL agents were given domain-specific primitive actions equivalent to what an SL would induce through *how-learning*, training still required thousands of episodes (MacLellan & Gupta, 2021).

The decision tree's relatively better data efficiency demonstrates an advantage of symbolic versus sub-symbolic learning. However, learning decomposition proved to be more essential to achieving human-like data efficiency. In the 2-mechanism model, *how-learning* helped produce near human-like efficiency for fractions, but in multi-column tutor a further decomposition into *where-* and *when-learning* was essential. *Where-learning* lets *when-learning* spatially generalize across multiple uses of the same skill (e.g., across columns).

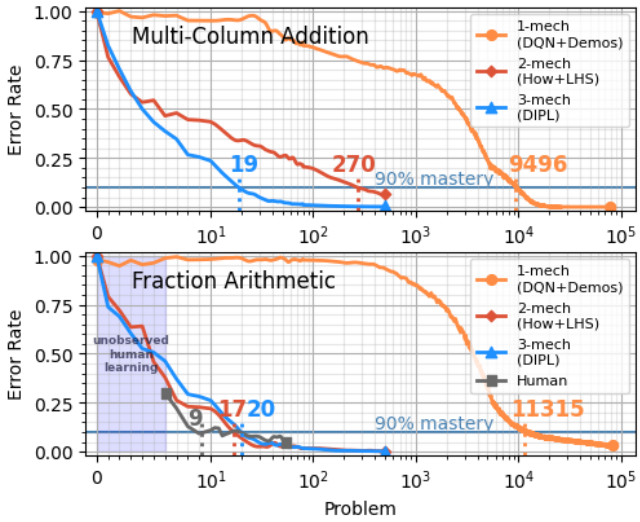

Figure 6: Log-scale x-axis learning curves, for DQN-Demos, How+LHS, DIPL annotated with 10% error intercept. Fraction domain includes human curves (grey) offset to account for unobserved learning opportunities.

Further Decomposition We may consider if yet more decomposition could produce greater data efficiency. For instance, HTN induction in VanLehn's (1990) Sierra, and the *process-learning* mechanism in AI2T may simplify the role of *when-learning*. When situated within an HTN, skills can be ordered explicitly within methods that dictate the steps for accomplishing higher-level tasks. This may reduce the role of *when-learning* so that it only needs to learn simple preconditions that gate the applicability of methods instead of also controlling the order in which primitive actions are applied. Generalizing this approach to work beyond the limited domains on which it has been applied may be a path toward yet greater data efficiency than we achieved here.

Conclusion

The rise of data-driven machine learning and LLMs has inspired a fixation on replicating human performance while overlooking the vast gap in data efficiency between data-driven machine learning and human learning, which is orders of magnitude more data efficient. This work demonstrates how cooperation between specialized learning mechanisms is a potential path to bridging that gap—enabling learning from tens of examples instead of tens of thousands.

References

Anderson, J. R., & Schunn, C. D. (2013). Implications of the act-r learning theory: No magic bullets. In *Advances in instructional psychology, volume 5* (pp. 1–33). Routledge.

Bellman, R. (1958). On a routing problem. *Quarterly of applied mathematics*, *16*(1), 87–90.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., ... others (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., ... others (2021). Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*.

Erol, K., Hendler, J. A., & Nau, D. S. (1994). *Semantics for hierarchical task-network planning*. Citeseer.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., ... Steinhardt, J. (2021). Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Kaliszyk, C., Urban, J., Michalewski, H., & Olšák, M. (2018). Reinforcement learning of theorem proving. *Advances in Neural Information Processing Systems*, *31*.

Käser, T., & Alexandron, G. (2024). Simulated learners in educational technology: A systematic literature review and a turing-like test. *International Journal of Artificial Intelligence in Education*, *34*(2), 545–585.

Koedinger, K. R., Baker, R. S., Cunningham, K., Skogsholm, A., Leber, B., & Stamper, J. (2010). A data repository for the edm community: The pslc datashop. *Handbook of educational data mining*, *43*, 43–56.

Koedinger, K. R., Carvalho, P. F., Liu, R., & McLaughlin, E. A. (2023). An astonishing regularity in student learning rate. *Proceedings of the National Academy of Sciences*, *120*(13), e2221311120.

Koedinger, K. R., Corbett, A. T., & Perfetti, C. (2012). The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science*, *36*(5), 757–798.

Konda, V., & Tsitsiklis, J. (1999). Actor-critic algorithms. *Advances in neural information processing systems*, *12*.

Laird, J. E. (2019). *The soar cognitive architecture*. MIT press.

Laird, J. E., Lebiere, C., & Rosenbloom, P. S. (2017). A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics. *Ai Magazine*, *38*(4), 13–26.

Li, N., Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2015). Integrating representation learning and skill learning in a human-like intelligent agent. *Artificial Intelligence*, *219*, 67–91.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., ... Cobbe, K. (2023). Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

MacLellan, C. J., & Gupta, A. (2021). Learning expert models for educationally relevant tasks using reinforcement learning. *International Educational Data Mining Society*.

Maclellan, C. J., Harpstead, E., Patel, R., & Koedinger, K. R. (2016). The apprentice learner architecture: Closing the loop between learning theory and educational data. *International Educational Data Mining Society*.

MacLellan, C. J., & Koedinger, K. R. (2020). Domain-general tutor authoring with apprentice learner models. *International Journal of Artificial Intelligence in Education*, 1–42.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., & De Raedt, L. (2018). Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, *31*.

Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2015). Teaching the teacher: Tutoring simstudent leads to more effective cognitive tutor authoring. *International Journal of Artificial Intelligence in Education*, *25*(1), 1–34.

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation* (Vol. 24, pp. 109–165). Elsevier.

Mirzadeh, I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., & Farajtabar, M. (2024). Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Neves, D. M. (1985). Learning procedures from examples and by doing. In *Ijcai* (pp. 624–630).

Patel, R., Liu, R., & Koedinger, K. R. (2016). When to block versus interleave practice? evidence against teaching fraction addition before fraction multiplication. In *Cogsci.*

Poesia, G., Dong, W., & Goodman, N. (2021). Contrastive reinforcement learning of symbolic reasoning domains. *Advances in neural information processing systems*, *34*, 15946–15956.

Quinlan, J. R., & Cameron-Jones, R. M. (1995). Induction of logic programs: Foil and related systems. *New Generation Computing*, *13*, 287–312.

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2019). Act-r: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, *10*(3), e1488.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati,

A., ... others (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

VanLehn, K. (1990). *Mind bugs: The origins of procedural misconceptions*. MIT press.

VanLehn, K., Ohlsson, S., & Nason, R. (1994). Applications of simulated students: An exploration. *Journal of artificial intelligence in education*, *5*, 135–135.

Weitekamp, D., Harpstead, E., & Koedinger, K. (2024). Ai2t: Building trustable ai tutors by interactively teaching a self-aware learning agent. *arXiv preprint arXiv:2411.17924*.

Weitekamp, D., Harpstead, E., MacLellan, C. J., Rachatasumrit, N., & Koedinger, K. R. (2019). Toward near zero-parameter prediction using a computational model of student learning. *International Educational Data Mining Society*.

Weitekamp, D., Rachatasumrit, N., Wei, R., Harpstead, E., & Koedinger, K. (2023). Simulating learning from language and examples. In *International conference on artificial intelligence in education* (pp. 580–586).

Weitekamp, D., Ye, Z., Rachatasumrit, N., Harpstead, E., & Koedinger, K. (2020). Investigating differential error types between human and simulated learners. In *International conference on artificial intelligence in education* (pp. 586–597).

Xiao, Z., & Zhang, D. (2023). A deep reinforcement learning agent for geometry online tutoring. *Knowledge and Information Systems*, *65*(4), 1611–1625.