Machine Learning (2023) 112:3591–3632
https://doi.org/10.1007/s10994-023-06351-8

Explanatory machine learning for sequential human teaching

Lun Ai[1] · **Johannes Langer**[2] · **Stephen H. Muggleton**[1] · **Ute Schmid**[3]

Received: 25 May 2022 / Revised: 14 March 2023 / Accepted: 8 May 2023 /
Published online: 28 June 2023

Abstract
The topic of comprehensibility of machine-learned theories has recently drawn increasing attention. Inductive logic programming uses logic programming to derive logic theories from small data based on abduction and induction techniques. Learned theories are represented in the form of rules as declarative descriptions of obtained knowledge. In earlier work, the authors provided the first evidence of a measurable increase in human comprehension based on machine-learned logic rules for simple classification tasks. In a later study, it was found that the presentation of machine-learned explanations to humans can produce both beneficial and harmful effects in the context of game learning. We continue our investigation of comprehensibility by examining the effects of the ordering of concept presentations on human comprehension. In this work, we examine the explanatory effects of curriculum order and the presence of machine-learned explanations for sequential problem-solving. We show that (1) there exist tasks A and B such that learning A before learning B results in better comprehension for humans in comparison to learning B before learning A and (2) there exist tasks A and B such that the presence of explanations when learning A contributes to improved human comprehension when subsequently learning B. We propose a framework for the effects of sequential teaching on comprehension based on an existing definition of comprehensibility and provide evidence for support from data collected in human trials. Our empirical study involves curricula that teach novices the merge sort algorithm. Our results show that sequential teaching of concepts with increasing complexity (a) has a beneficial effect on human comprehension and (b) leads to human re-discovery of divide-and-conquer problem-solving strategies, and (c) allows adaptations of human problem-solving strategy with better performance when machine-learned explanations are also presented.

Keywords Explainable artificial intelligence · Machine learning comprehensibility · Meta-interpretive learning · Inductive logic programming

Editors: Alireza Tamaddoni-Nezhad, Alan Bundy, Luc De Raedt, Artur d'Avila Garcez, Sebastijan Dumančic, Cèsar Ferri, Pascal Hitzler, Nikos Katzouris, Denis Mareschal.

Extended author information available on the last page of the article

1 Introduction

Human learning can be described as a concept formation progression in which observations of objects and events are summarised and formulated by some inter-dependent hierarchical structure (Gennari et al., 1989). As an example, in the context of teaching algorithms to human students, we may imagine a human teacher using the material in Fig. 1 for teaching the merge sort algorithm. In a bottom-up teaching approach with increasing concept complexity, students learn how to merge first (left) and then move on to study sorting (right). In contrast, via a top-down teaching approach which presents concepts with decreasing complexity, students learn to sort first (right) without prior knowledge of merging and study merging afterwards (left). It might be of interest to the human teacher to question which approach yields higher student sorting performance and what effects the two teaching approaches have on students' sorting strategies.

It was stressed by the Machine Learning pioneer (Michie, 1991) that an intelligent machine not only needs to excel humans in performance but should also possess the capacity to effectively interact with and transfer knowledge to humans. An intelligent machine can in this way fulfil an integrated role that contributes to both the "stock of software" and the "stock of knowledge" (Michie, 1991). Owing to increasing awareness of the importance of explainable AI, reported by comprehensive reviews (Adadi & Berrada, 2018; Anjomshoae et al., 2019; Minh et al., 2021), a great number of studies have emerged with an emphasis on systems that provide support for human understanding. In addition, there is an increasing emphasis on AI that interacts to keep humans "in the loop" (Miller, 2019; Schmid, 2021). However, owing to the absence of quantifiable definitions and evaluation procedures, some issues such as ambiguity and lack of empirical evaluations have been highlighted (Freitas, 2014; Lipton, 2018; Miller, 2019).

At the intersection of these two paradigms is the research area that focuses on the comprehensibility of machine learning. An operational definition of comprehensibility was provided in Schmid et al. (2017) for examining machine-human teaching interaction. This definition relates human comprehension to the out-of-sample prediction accuracy of human learners after studying training examples and a machine-learned logic program. It allowed the first demonstration (Muggleton et al., 2018) of Michie's

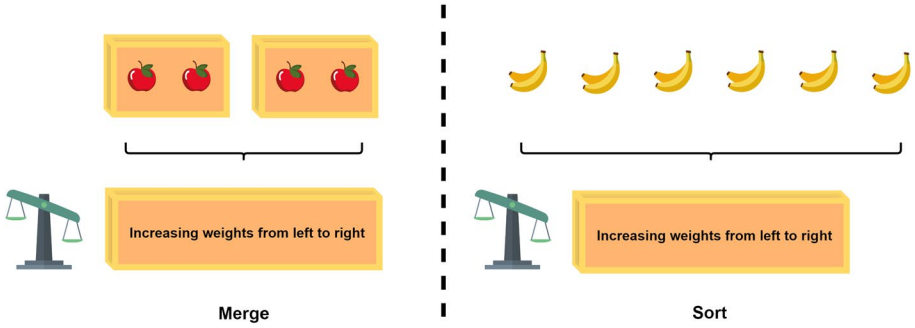


Fig. 1 An illustration of teaching materials of merging and sorting that can be learned in incremental order (left to right). All fruits have different weights which can be pairwise compared using the balance scale at the corner. On the left-hand side, two sets of apples should be merged into a larger collection in increasing weights from left to right. On the right-hand side, a set of bananas needs to be arranged in increasing weights from left to right

Ultra-Strong Machine Learning (USML) which stressed the role of teaching machine-learned knowledge to humans such that their proficiency on a task can be raised to a higher level with respect to learning from randomly selected examples. Later results in Ai et al. (2021) showed that explanations generated from machine-learned logic programs can have both beneficial and harmful effects on human comprehension and suggested, via a cognitive window framework, that explanatory effects are dependent on both descriptive and executional complexity. Despite the classification of explanatory effectiveness in Ai et al. (2021), the effects of the order of concepts in a teaching curriculum on human learning—an essential topic of machine learning and human problem solving that has yet to be explored for understanding the comprehensibility of machine-learned logic programs.

Based on established results and teaching procedures in Ai et al. (2021) and Muggleton et al. (2018), in this work, we extend the frameworks of comprehensibility and explanatory effects to account for the impacts of sequential teaching on comprehension. We refer to the cognitive window framework which attributes the explanatory effects to (1) the descriptive complexity of machine-learned rules and (2) the cost of knowledge application with respect to an execution stack. A human trial of sequential teaching was conducted to examine the effects of concept ordering in curricula and explanations in the context of learning an efficient sorting strategy. We hypothesise the improvement of human comprehension from sequential teaching as the result of the reduction in the size of the hypothesis space associated with the target concept definition. Our proposed framework is consistent with empirical evidence collected from the human trial involving participants with no background in programming.

We summarise the paper's contributions as follows:

- We define a measure to evaluate beneficial/harmful explanatory effects of a sequential teaching curriculum on human comprehension.
- We hypothesise the improvement of human comprehension from sequential teaching curricula based on the Blumer bound (Blumer et al., 1989).
- We demonstrate based on an analysis of empirical results that a sequential teaching curriculum with increasing concept complexity has a beneficial effect on human comprehension.
- We show results that indicate the re-discovery of divide-and-conquer algorithms after human novices learn from concepts with increasing complexity.
- We provide evidence of the optimisation of problem-solving strategy as a result of additionally studying explanations generated from machine-learned logic rules.

This paper is organised as follows. In Sect. 2, we review the literature relevant to the paper. In Sect. 3, we present our theoretical framework of sequential teaching curricula. In Sect. 4, we describe our empirical approach including the experimental hypotheses. In Sect. 5, we show the results of our main Amazon Mechanical Turk experiment that involved teaching human participants to sort based on the training of merging two ordered sequences. In Sect. 5.5, we provide a detailed discussion of our findings on the effects of curriculum order and explanations on human comprehension. In Sect. 6, we provide a review of our work and comment on our contributions. We additionally discuss future extensions to the demonstrated results, behavioural cloning and human knowledge re-discovery.

2 Related work

This section summarises related research on human–machine teaching interactions and highlights our motivations. We give an overview of related results in human learning in Psychology. This is followed by a brief account of learning between humans and machines and objective means for evaluating the comprehensibility of machine explanations.

2.1 Human learning in curricula with explanations

The ability of humans to recognise and extract knowledge useful to a problem-solving context is dependent on prior experience and understanding of related domains (Anzai & Simon, 1979). While a classification decision is dependent only on information concerning the current state, goal-directed decision-making requires consideration of a decision's possible effect on future states (Barto et al., 1989). For sequential decision-making tasks such as the Tower of Hanoi, state transitions of the problem solution can be explicitly expressed by rules (Seger, 1994). "Chunking" reusable sub-goals was demonstrated as an efficient problem-solving approach to encode problem-solution structures and reduce the load on short-term memory (Anzai & Simon, 1979).

The reliance of human problem solving on the problem domain and experience of a person (Dienes & Perner, 1999) can be attributed to implicit (System 1) and explicit (System 2) knowledge (Kahneman, 2011). In contrast to implicit knowledge which is only attainable through practice, declarative knowledge can be transferred explicitly in the form of explanations (Chi & Ohlsson, 2005). For knowledge accumulation, explicit explanations and hints help make sense of the new information and allow integration into existing knowledge (Mayer, 2004). In the context of solving simple logical problems (Craig, 1956; Kittel, 1957) and generating LOGO programs (Lee & Thompson, 1997), guided learning has been demonstrated more efficient than exploration-oriented approaches. However, multiple studies suggest that explanations should not be presented alone. As demonstrated by Aleven and Koedinger (2002), Anderson et al. (1997), Berry and Broadbent (1995), human learners do not always benefit from explanations when the specific problem-solving context is absent. The combination of explicit guidance and worked examples provides a good hands-on problem-solving experience for learners to study and practise for themselves due to the limited capacity of working memory (Sweller et al., 2007). The presentation of worked examples along with explanations makes this specific task context available and was shown to support learning (Alfieri et al., 2011; Reed & Bolstad, 1991).

2.2 Teaching between the machine and humans

Machine teaching is an area of machine learning that focuses on the optimisation of training sets and the design of teaching protocols in an example-based setting (Zhu, 2015). Studies in machine teaching have provided formal methods for teaching situations in which the goal is to train human students via an optimal set of examples to learn a target hypothesis. The machine teacher in these setups (Mansouri et al., 2019; Rafferty et al., 2016; Yeo et al., 2019) is provided information about the quantified cognitive model of the human learner and has access to the learning progress based on the student's responses to examples. The teacher can then monitor the reactions of students which allows the teacher to make adjustments to teaching materials. However, often excessive assumptions about the teacher's knowledge are made for deriving theoretical results. The teacher is usually assumed to have

perfect knowledge about the computational model of students such as their learning rate and their background knowledge (Devidze et al., 2020). Thus, owing to the specificity of these theoretical models, they are difficult to apply in real-world teaching settings.

Communication of machine-learned solutions can be alternatively realised through explanations in the form of rules. For Inductive Logic Programming (ILP) (Muggleton, 1991), systems use logic programs to represent hypotheses as generalisations of examples and background knowledge. ILP systems make use of both induction and abduction, which are critical mechanisms for human knowledge attainment (Hobbs, 2008; Lemke et al., 1967). A hypothesis learned by an ILP algorithm is represented as a set of rules and can be used to communicate the discovered knowledge to humans. For instance, an ILP system (Bain & Muggleton, 1994) learns optimal strategies of chess endgames for depths 0, 1 and 2 and incorporates a complexity constraint on the number of clauses in a predicate definition based on the hypothesised limitation of working memory capacity of 7 ± 2 "chunks" (Miller, 1956). A fully automated discovery system (King et al., 2004) re-discovers the role of genes of known function by combining ILP-based scientific discovery software with a laboratory robot. In Sternberg and Muggleton (2003), the ILP system produces hypotheses as logic rules that depict the chemical connectivity of molecules and predict mutagenicity, which accommodates collaboration with a human chemist. However, in most systems, it remains an ongoing challenge to ensure human understanding due to the high complexity of information encoded via these interfaces. In a recent work (Ai et al., 2021), it was shown that visual and textual explanations for playing a simple two-player game, generated based on rules learned by an ILP system, may cause improvement or degradation in human comprehension after a brief study. The authors demonstrated that the outcome of studying explanations provided is affected by a cognitive window which takes account of the descriptive complexity and execution requirements of the rules.

2.3 Operational measures of human understanding in AI

Explainable Artificial Intelligence (XAI) is an area of AI that studies AI systems that allow human understanding by providing human-readable explanations of decisions based on structures and functions. Researchers have attempted to devise AI systems to present knowledge attained in various formats, such as texts (Hind et al., 2019; McAuley & Leskovec, 2013), visualisations (Poulin et al., 2006; Ribeiro et al., 2018; Tamagnini et al., 2017; Xu et al., 2015) and visualisation-text hybrid demonstrations (Schmid & Finzel, 2020). Simpler explanations of local predictions can be devised to provide a localised understanding of a learned classification model (Ribeiro & Guestrin, 2016). For reinforcement learning applications, learned strategies are implicitly encoded by policy functions with continuous domains which are difficult for humans to understand. This concern about the opacity of learned strategies was addressed by incorporating the use of relational biases (Burke et al., 2019; Džeroski et al., 2001; Zambaldi et al., 2019) to enhance human understandability. In addition, case-based summaries of a policy (Amir et al., 2019) from sets of selected states in a larger state space of an agent allow a limited human understanding of the agent's decisions.

In XAI, there exists a diversity of motivations and technical descriptions of systems which support human understanding. Terms such as transparency, interpretability, explainability and comprehensibility are frequently used terms at the core of XAI (Arrieta et al., 2020). Transparency typically refers to the ability of a model or a system to be human-understandable on its own (Arrieta et al., 2020). Interpretability relates to the degree of

clarity of information revealed to the user by a system (Minh et al., 2021). Explainability stresses the role of a system as an "explainer" interface to the user (Arrieta et al., 2020). Comprehensibility denotes a system's ability to represent machine-learned knowledge in a human-understandable format (Arrieta et al., 2020; Gleicher, 2016). However, owing to the absence of operational definitions and procedures of evaluation, a great body of recent results was established based on subjective views (Freitas, 2014; Miller, 2019) which are discordant (Lipton, 2018) with respect to the sub-problem of XAI that the studies attempted to address. Studies were often found to report limited empirical evidence as support of the proclaimed effect and did not take into consideration that human understanding is just as vital as the computational procedures that generate explanatory information (Miller, 2019). It was stressed in Adadi and Berrada (2018) that a great body of studies did not evaluate the effect of explanations on their target users. Those works that did include explanatory effect evaluations often failed to provide a good account of the context, results and limitations (Anjomshoae et al., 2019). A number of reports also brought to attention the potential misinterpretation and misuse of explanatory information. For example, partially transparent explanatory information may cause more confusion if key decision-making of systems is omitted (Rudin, 2019). The authors in Stumpf et al. (2016) expressed trustworthiness concern and showed that human decision-making over-relied on the intelligent system even when explanatory information provided by the system was inaccurate.

The importance of comprehensibility has long been recognised in machine learning of human-oriented knowledge. Michalski (1983) suggested that comprehensible machine learning should produce outputs that share similar structure and semantics as those produced by human experts and should present learned knowledge in human-understandable "chunks". Owing to the challenges of quantification, comprehensibility has usually been associated with the complexity of machine-learned models (Guidotti et al., 2018). Research on the operational definition of machine learning comprehensibility can be traced back to Michie's definition of *Ultra-Strong* Machine Learning in the 1980s (Michie, 1988). This criterion stresses machine learners' ability to teach hypotheses to humans such that their performance over the task is improved by a substantial margin compared with human learning alone. An operational definition of comprehensibility (Muggleton et al., 2018; Schmid et al., 2017) was proposed which assesses human comprehension based on the human out-of-sample classification accuracy. In recent work Ai et al. (2021), both beneficial and harmful effects of explanatory machine learning were identified in the context of symbolic machine learning. However, the effects of the teaching order of concepts on human comprehension have not been fully explored in the literature on machine learning comprehensibility.

3 Theoretical framework

To introduce our theoretical framework, in Sect. 3.1 we begin by referring to the logic notations and background of Meta-Interpretive Learning (MIL) (Muggleton et al., 2015). A MIL system $Metagol_O$ (Cropper & Muggleton, 2015) has been modified in Sect. 3.2 to learn a logic theory of sorting as explanations in our empirical human trial. In Sect. 3.3, the frameworks of comprehensibility (Schmid et al., 2017) and explanatory effects (Ai et al., 2021) have been extended to account for sequential teaching curricula containing "blocks" of teaching materials. In Sect. 3.4, we refer to the cognitive cost of a logic program (Schmid et al., 2017) which was proposed to approximate the complexity of the human execution of

the logic program as humans simulate the knowledge being taught. We describe the cognitive window and sequential teaching improvement conjectures in Sect. 3.5 based on Ai et al. (2021) and our proposed new framework.

3.1 Meta-interpretive learning

An Inductive Logic Programming (ILP) (Muggleton, 1991) algorithm uses background knowledge $\mathcal{B}$ to induce a hypothesis in the form of a logic program which entails all of the positive examples $\mathcal{E}^+$ and none of the negative examples $\mathcal{E}^-$. Meta-Interpretive Learning (MIL) (Muggleton et al., 2015) is a variant of ILP which supports predicate invention (Muggleton et al., 2015), dependent learning (Lin et al., 2014) and learning of recursive and higher-order programs. A MIL algorithm solves the following problem: given a tuple $(\mathcal{B}, \mathcal{M}, \mathcal{E}^+, \mathcal{E}^-, \mathcal{I}, \mathcal{H})$ where the background knowledge $\mathcal{B}$ is a first-order logic program, $\mathcal{M}$ is a set of second-order clauses, positive examples $\mathcal{E}^+$ and negative examples $\mathcal{E}^-$ are ground atoms, learns a logic program H such that

$$\forall e^+ \in \mathcal{E}^+ \quad \mathcal{M} \cup H \cup \mathcal{B} \models e^+$$
$$\forall e^- \in \mathcal{E}^- \quad \mathcal{M} \cup H \cup \mathcal{B} \not\models e^-$$
$$\mathcal{M} \cup \mathcal{B} \models H$$

$\mathcal{I}$ is a set of predicate symbols reserved for predicate invention. Output logic program H is a hypothesis in the hypothesis space $\mathcal{H}$ described by a finite number of predicate symbols and constants. The background knowledge $\mathcal{B}$ contains a set of predicate definitions as primitives. The set of second-order clauses $\mathcal{M}$ is referred to as meta-rules. Each meta-rule contains existentially quantified second-order variables and universally quantified first-order variables. A MIL algorithm employs meta-rules and substitutes second-order variables with predicate symbols to derive first-order theories as logical generalisations.

3.2 MIL for learning sorting algorithm

The MIL system *Metagol* (Cropper & Muggleton, 2016) is an implementation of a meta-interpreter in Prolog. *Metagol* supports predicate invention and dependent learning which reduces the size of the hypothesis space and improves learning performance (Cropper, 2017). $Metagol_O$ (Cropper & Muggleton, 2015) makes use of composite objects and actions defined from primitive objects and actions. $Metagol_O$ addresses the issue that most ILP systems only considered the textual complexity of hypothesis programs. The inability to distinguish two algorithms, for instance, insertion sort and quick sort, would result in the less computationally efficient but textually compact algorithm being learned. Both textual and resource costs of hypothesis programs are minimised by $Metagol_O$ using a technique called iterative descent (Cropper & Muggleton, 2015) which computes the resource cost of hypotheses of increasing sizes according to a pre-defined cost function. For learning robot postman and robot sorting strategies, the background knowledge is defined in composite actions and objects and provided to $Metagol_O$. The authors showed that $Metagol_O$ converges to hypotheses with the optimal resource complexity and learns resource-efficient robot strategies (Cropper & Muggleton, 2015).

We modify the background knowledge of $Metagol_O$ for learning a bottom-up variant of the merge sort algorithm (Goldstine & Neumann, 1963) for sorting positive integers. Merge sort is a recursive sorting algorithm based on a divide-and-conquer approach. The

conventional implementation recursively splits an input sequence into units and performs merging on these units to build up sorted sequences that are shorter. The aforementioned variant takes an input sequence as a list of sub-sequences of length one and iteratively merges two sub-sequences in a bottom-up fashion. For human learners with little or no prior knowledge of recursive sorting algorithms, we consider that iterative algorithms are conceptually easier teaching materials in the case of unfamiliarity with recursive programs. In addition, we assume that human sorting strategies do not usually involve merging. Humans may learn to utilise merging for creative sorting strategies.

A demonstration of the execution of the bottom-up merge sort algorithm is included in Example 1. To learn the bottom-up variant of merge sort,[1] we define the world state that the robot sorter would see. $Metagol_O$ is provided with primitives to manipulate expressions which compactly represent ordered sequences. Let *Lt_expr* represent a non-empty monotonically increasing sequence of integers and be defined as $Lt\_expr := Int \mid Lt\_expr < Lt\_expr$.

Example 1 Let the sequence *[4, 6, 5, 2, 3, 1]* be an input to the bottom-up variant of merge sort. Merging is iteratively performed from the end of the list with lower indices. Merging takes two expressions as inputs, e.g. 4 and 6, and outputs one expression, e.g. $4 < 6$. In *iteration 1*, three merging would be executed on expressions *4* and *6*, *5* and *2*, *3* and *1* which would return $[4 < 6, 2 < 5, 1 < 3]$. In *iteration 2*, one merging would be executed on two leftmost expressions $4 < 6$ and $2 < 5$ which would give $[2 < 4 < 5 < 6, 1 < 3]$. In *iteration 3*, expressions $2 < 4 < 5 < 6$ and $1 < 3$ are merged which would return $[1 < 2 < 3 < 4 < 5 < 6]$. Only one expression remains which represents the sorted sequence and the algorithm terminates with $[1 < 2 < 3 < 4 < 5 < 6]$ as output.

Objects are separated into primitive objects O_{prim} and composite objects O_{comp} where $O = O_{prim} \cup O_{comp}$ denotes the set of all objects in the robot world. Each composite object is defined by primitive objects or other composite objects. S is a set of states and each state is a tuple of objects. In particular, a state is defined as a tuple $(exprs, energy, left\_bag, right\_bag, memory)$ with an arity of 5. The *exprs* is a list of *Lt_expr*. The *energy* records the resource cost. The *left_bag* and *right_bag* are lists of the parsed numbers. The *memory* is a list of newly created *Lt_expr*. An action $a \in A$ is a function such that $a : S \rightarrow S$. Each action is either a primitive action included in A_{prim} or a composite action included in A_{comp}. Let A be an enumerable set of actions where $A = A_{prim} \cup A_{comp}$. Every composite action is constructed based on primitive actions or other composite actions. A resource function $r : A \times S \rightarrow \mathbb{N}$ defines the resources consumed by carrying out an action $a \in A$ in state $s \in S$.

$Metagol_O$'s background knowledge is supplied with composite actions A_{comp} = {*parse_exprs*/2, *compare_nums*/2, *single_expr*/2, *drop_bag_remaining*/2, *recycle_memory*/2} for learning *merger*/2 and *sorter*/2 in Table 1. Meta-rules in Table 2 allow $Metagol_O$ to learn recursive programs. In *merger*/2, two *Lt_expr* from *exprs* are first parsed into *left_bag* and *right_bag*. Then numbers are compared pair-wise from two bags and the smaller number is appended to the last *Lt_expr* in *memory*. *sorter*/2 applies *merger*/2 over all *Lt_expr* in *exprs*. Constructed *Lt_expr* in *memory* are recycled back to *exprs* by *recycle_memory*/2. This process is iterated until there is only one *Lt_expr* left in

[1] Source code and demos are available on https://github.com/LAi1997/sequential-teaching.

Table 1 When the robot performs *parse_exprs*/2, it parses two expressions by removing the "<" symbols, and it puts one sequence of numbers into the left bag and the other sequence of numbers into the right bag

Definition	Rules

merger/2	`merger(A,B):-parse_exprs(A,C),merger_1(C,B).`
	`merger_1(A,B):- compare_nums(A,C),merger_1(C,B)`
	`merger_1(A,B):-compare_nums(A,C),drop_bag_remaining(C,B).`

sorter/2 (after learning merger/2)	`sorter(A,B):-merger(A,C),sorter(C,B).`
	`sorter(A,B):-recycle_memory(A,C), sorter(C,B).`
	`sorter(A,B):-single_expr(A,C), single_expr(C,B).`

sorter/2 (without learning merger/2)	`sorter(A,B):-parse_exprs(A,C),sorter(C,B).`
	`sorter(A,B):-compare_nums(A,C), sorter(C,B).`
	`sorter(A,B):-drop_bag_remaining(A,C), sorter(C,B).`
	`sorter(A,B):-recycle_memory(A,C), sorter(C,B).`
	`sorter(A,B):-single_expr(A,C), single_expr(C,B).`

compare_nums/2 first selects a number from the left bag and a number from the right bag. Then it compares the two numbers. Afterwards, *compare_nums*/2 uses the smaller number to extend the last expression in the memory and puts the larger number back in its original bag. When one of the bags is empty, the robot performs *drop_bag_remaining*/2 to append the rest of the numbers to the last expression in the memory. *recycle_memory*/2 takes all expressions in the memory and fills the expression list *exprs*. *single_expr*/2 checks if there exists only one expression in the expression list *exprs*

Table 2 $Metagol_O$ uses two meta-rules

Name	Meta-rule	Orders
Chain	$P(x,y) \leftarrow Q(x,z), R(z,y)$	$P \succ Q, P \succ R$
Tailrec	$P(x,y) \leftarrow Q(x,z), P(z,y)$	$P \succ Q, x \succ z \succ y$

P, Q, and R are existentially quantified higher-order variables and x, y, and z are universally quantified first-order variables. $\succ$ is a total ordering constraint over the Herbrand base to guarantee the termination of hypotheses

exprs and *memory* is empty. We define a cost function for *compare_nums*/2 such that the *energy* is incremented by one whenever two numbers are successfully compared. We additionally define a constraint over the world states. This constraint based on Spearman's rank correlation coefficient (Spearman, 1904) ensures the expressions in the current state are not less sorted compared with expressions in the immediate next state.

3.3 Explanatory effect of curriculum order

We have employed a quantifiable approach that provides operational definitions for the evaluation of the effect of sequential teaching on human comprehension. This allows us to better account for the context, results and limitations. We describe the basis of our framework by referring to previous work on comprehensibility (Schmid et al., 2017)

Table 3 A summary of the connections between definitions of comprehensibility, explanatory effects and cognitive window in Muggleton et al. (2018), Ai et al. (2021) and the proposed framework to account for effects of sequential teaching curricula

Framework	Definitions and results in previous works	Extended definitions and results for sequential teaching
Comprehensibility	Definitions 1 and 2	Definitions 5 to 7
Explanatory effects	Definitions 3 and 4	Definitions 8 and 9
Cognitive window	Definitions 10 to 14, Remark 1 and Conjecture 1 to 2	Conjecture 3

and explanatory effects (Ai et al., 2021) (Table 3). Then we define a sequential teaching curriculum and its relative effectiveness with respect to another sequential teaching curriculum.

Based on an operable definition of comprehensibility (Schmid et al., 2017), the authors in Ai et al. (2021) explored the explanatory effects of machine-learned logic rules on human comprehension and proposed a framework of a cognitive window that focuses on the symbolic subset of machine learning. The definitions of comprehensibility and explanatory effects are extended in this present work to account for sequential teaching curricula in the context of symbolic machine learning. We use the logic programs learned from $Metagol_O$ to generate explanations for our human trial. For the purpose of experimentation, we only consider curricula with noise-free training examples. This does not mean, however, that the proposed definitions could not be extended outside of the considered scope.

The operational definition of comprehensibility (Schmid et al., 2017) is an objective means of evaluating human comprehension of a concept based on human out-of-sample predictive accuracy. It was extended in Ai et al. (2021) to define the explanatory effectiveness of machine-learned logic programs which measures the extent to which examples with explanations generated from machine-learned logic programs can be simulated by humans.

Definition 1 (*Unaided human comprehension of examples*, $C_h(D, H, E)$) Given that D is a logic program representing the definition of a target predicate, H is a human group and E is a set of examples of the target predicate. The unaided human comprehension of examples E is the mean accuracy with which a human $h \in H$ after a brief study of E and without further sight can classify new material sampled randomly from the domain of D.

Compared with the unaided human comprehension of examples, the machine-explained human comprehension of examples corresponds to the out-of-sample classification accuracy after studying the machine-learned explanation $M(E)$ where M is a machine learning algorithm and E is a set of training examples.

Definition 2 (*Machine-explained human comprehension of examples*, $C_{ex}(D, H, M(E))$): Given that D is a logic program representing the definition of a target predicate, H is a human group, $M(E)$ is a theory learned using machine learning algorithm M and E is a set of examples of the target predicate. The machine-explained human comprehension of examples E is the mean accuracy with which a human $h \in H$ after a brief study of an explanation based on $M(E)$ and without further sight can classify new material sampled randomly from the domain of D.

The explanatory effect of a machine-learned theory on human comprehension of a task is defined as the difference between machine-explain and unaided human comprehension of examples of the task.

Definition 3 (*Explanatory effect of a machine-learned theory*, $E_{ex}(D, H, M(E))$): Given a logic program D representing the definition of a target predicate, a human group H and a machine learning algorithm M, the explanatory effect of the theory $M(E)$ learned from examples E is

$$E_{ex}(D, H, M(E)) = C_{ex}(D, H, M(E)) - C_h(D, H, E)$$

In the case that the explanations provided by the machine lead to a positive improvement, the explanatory effect of the learned theory is beneficial to human comprehension. When this difference is negative, the explanatory effect of the machine-learned theory is harmful to human comprehension.

Definition 4 (*Beneficial/harmful effect of a machine-learned theory*): Given a logic program D representing the definition of a target predicate, a set of training examples E, a group of humans H, and a symbolic machine learning algorithm M:

- $M(E)$ learned from E is *beneficial* to H if $E_{ex}(D, H, M(E)) > 0$
- $M(E)$ learned from E is *harmful* to H if $E_{ex}(D, H, M(E)) < 0$
- Otherwise, $M(E)$ learned from E does not have an effect on H

Whether or not an explanatory effect is significantly beneficial or harmful can be determined empirically by applying an independent two-sample t-test or similar method to the means of unaided- and machine-explained human comprehension. In the case that the mean difference is not significant, we cannot classify the effect of a machine-learned theory as either beneficial or harmful.

The definitions of human comprehension and explanatory effects are extended to account for the ordering of concepts. We define a sequential teaching curriculum to contain a set of concept definitions, sets of examples and explanations. The arrangement of concepts in a curriculum can be decided based on an ordering function which outputs a score to rank concepts.

Definition 5 (*Curriculum rank function*, $\lambda(D, E, M)$) Given a set of predicate definitions Ds, sets of examples Es and a set of machine learning algorithms Ms, $\lambda : Ds \times Es \times Ms \rightarrow \mathbb{N}$ is a rank function that returns the rank value of any predicate definition $D \in Ds$ in a curriculum.

A sequential teaching curriculum contains "chunks" of teaching material and a curriculum has at least one such "chunk". The order of concepts received by humans is determined by a defined rank function which assigns a score to every block of teaching material which contains the concept definitions, examples and explanations.

Definition 6 (*Sequential teaching curriculum*, $ST(Ds, Es, \lambda, Ms)$): Given that $Ds = \{D_1, D_2, \ldots D_n\}$ is a set of predicate definitions of size n, $Es = \{E_1, E_2, \ldots, E_n\}$ contains sets of examples where E_i are examples of D_i for $1 \leq i \leq n$, Ms denotes a set of machine learning algorithms and $\lambda : Ds \times Es \times Ms \rightarrow \mathbb{N}$ is a rank function, a sequential teaching curriculum $ST(Ds, Es$

λ, Ms) is an enumerable set of tuples $\{(R_1, D_1, E_1, M_1), (R_2, D_2, E_2, M_2), ..., (R_n, D_n, E_n, M_n)\}$ in which $R_j = \lambda(D_j, E_j, M_j)$ is the rank of a concept in the curriculum for $1 \leq j \leq n$. A sequential teaching curriculum enumerates concepts with respect to increasing rank value.

Human comprehension is considered aided by explanations if the machine-learned logic program in a teaching material block is not empty. Otherwise, human comprehension is established based on training examples alone. We proceed to define the effect of a sequential teaching curriculum on the human comprehension of a concept.

Definition 7 (*Human comprehension of examples in a sequential teaching curriculum,* $C_{seq}(T, H)$): Given that $T = ST(Ds, Es, \lambda, Ms)$ is a sequential teaching curriculum where Ds is a set of n predicate definitions, Es are sets of examples, $\lambda : Ds \times Es \times Ms \rightarrow \mathbb{N}$ is a rank function, Ms is a set of machine learning algorithms and H is a human group, the human comprehension of examples in the sequential teaching curriculum is

$$C_{seq}(T, H) = \left\{ (R_i, \tau) \middle| \begin{array}{ll} \tau = C_h(D_i, H, E_i), & M_i(E_i) = \emptyset \\ \tau = C_{ex}(D_i, H, M_i(E_i)), & M_i(E_i) \neq \emptyset \\ \text{for } (R_i, D_i, E_i, M_i) \in S, 1 \leq i \leq n & \end{array} \right\}$$

We provide examples of sequential teaching definition which specify a curriculum that presents training materials of merging before introducing training materials of sorting and a curriculum with the reverse order of teaching.

Example 2 Let $Ds = \{merger, sorter\}$ denote predicate definitions of merging and merge sort, E_{merge} and E_{sort} be the respective example sets and $Es = \{E_{merge}, E_{sort}\}$ and $Ms = \{Metagol_O\}$ which generates explanations. We define $\lambda_{merge/sort}$ ($merger$, E_{merge}, $Metagol_O$) = 0 and $\lambda_{merge/sort}$ ($sorter$, E_{sort}, $Metagol_O$) = 1. Let $\lambda_{sort/merge}$ ($merger$, E_{merge}, $Metagol_O$) = 1 and $\lambda_{sort/mergel}$ ($sorter$, E_{sort}, $Metagol_O$) = 0. The merge-then-sort curriculum is represented by $ST(Ds, Es, \lambda_{merge/sort}, Ms)$ which teaches merging before teaching sorting and the sort-then-merge curriculum is denoted by $ST(Ds, Es, \lambda_{sort/merge}, Ms)$ which teaches sorting before teaching merging. $\lambda_{merge/sort}$ specifies the ordering of concepts in the merge-then-sort curriculum $ST(Ds, Es, \lambda_{merge/sort}, Ms)$ and $\lambda_{sort/merge}$ defines the ordering of concepts in the sort-then-merge curriculum $ST(Ds, Es, \lambda_{sort/merge}, Ms)$.

Definition 8 (*Effect of a sequential teaching curriculum,* $E_{seq}(C_1, C_2, D)$): $C_1 = C_{seq}(ST(Ds, Es, \lambda_1, Ms), H)$ and $C_2 = C_{seq}(ST(Ds, Es, \lambda_2, Ms), H)$ are sequential teaching curricula where Ds is a set of n predicate definitions and $D \in Ds$ is a predicate definition, $Es = \{E, ...\}$ are sets of examples, $\lambda_1, \lambda_2 : Ds \times Es \times Ms \rightarrow \mathbb{N}$ are two rank functions, Ms is a set of machine learning algorithms and H is a human group. $R_1 = \lambda_1(D, E, M_1)$ and $R_2 = \lambda_2(D, E, M_2)$ are the ranks of D in C_1 and C_2 where $M_1, M_2 \in Ms$. The explanatory effect of curriculum C_1 over curriculum C_2 on predicate definition D is

$$E_{seq}(C_1, C_2, D) = \tau_1 - \tau_2$$

where $(R_1, \tau_1) \in C_1, (R_2, \tau_2) \in C_2$.

In Definition 7, comprehension measurements are obtained with respect to the ranking of concepts in a sequential teaching curriculum. The effect of a sequential teaching

curriculum reflects the degree to which human comprehension of a concept is affected by a curriculum. Given two sequential teaching curricula and a concept of interest, the effect of one curriculum over another curriculum corresponds to the difference in comprehension measurements of the concept. A significant difference in human comprehension as a result of learning from one curriculum over another curriculum can be classified as a beneficial or harmful effect of learning from the former curriculum.

Definition 9 (*Beneficial/harmful effect of a sequential teaching curriculum)*: Given that $C_1 = C_{seq}(ST(Ds, Es, \lambda_1, Ms), H)$ and $C_2 = C_{seq}(ST(Ds, Es, \lambda_2, Ms), H)$ are sequential teaching curricula where Ds is a set of n predicate definitions and $D \in Ds$ is a predicate definition, Es are sets of examples, $\lambda_1, \lambda_2 : Ds \times Es \times Ms \rightarrow \mathbb{N}$ are two rank functions, Ms is a set of machine learning algorithms, and H is a human group, the curriculum C_1 in comparison with the curriculum C_2 is

- *beneficial* to H on D if $E_{seq}(C_1, C_2, D) > 0$
- *harmful* to H on D if $E_{seq}(C_1, C_2, D) < 0$
- Otherwise there is no effect on H for D in curriculum C_1 compared with curriculum C_2

The effect on the predicate D is beneficial when the curriculum ordered by λ_1 results in a better comprehension of D than the comprehension of D in the curriculum ordered by λ_2. Conversely, the effect on D is harmful in the case that a degradation of comprehension is observed when comparing the comprehension of D between curricula described by λ_1 and λ_2. As with the effect of a machine-learned theory (Definition 4), whether or not an effect of a sequential teaching curriculum is significantly beneficial or harmful can be determined using an independent two-sample t-test or similar method.

3.4 Cognitive cost of a logic program

A consensus between the literature on cognitive psychology and artificial intelligence (Miller, 1956; Johnson-Laird, 1986; Newell, 1990) is that the information processing of humans in working memory can be modelled by manipulation of symbols in machines which can be formally captured by Turing machines. The limitation on working memory capacity corresponds to a bounded tape length and instruction complexity in Turing Machines. For rule-based concept acquisition (Bruner et al., 1956), human concept attainment is analogical to a search in a collection of hypotheses guided by some preference ordering which is comparable to version space learning in machine learning (Mitchell, 1982). In addition, the organisation of complex action sequences into hierarchical structures in human information processing is an efficient mechanism for general problem-solving. In the contexts of analogical problem-solving (Carbonell, 1985) and production systems (Newell, 1990), top-level goals are decomposed into sub-goals which can be described by a set of rules that guide actions in problem-solving sequences. Although rules generally can be considered as procedural knowledge, in complex domains human verbalisation of rules utilises declarative memory (Anderson et al., 1993; Schmid & Kitzelmann, 2011). The explicit representations of a human problem-solving solution correspond to rule-like descriptions. For the following definitions, we assume that:

- Human learners are version space learners who can make only limited hypothesis space searches

- Human learning is guided by meta-rules which construct the sub-goal structure and uses predicates as background knowledge
- An increase in the execution complexity of the problem solution can have a negative effect on performance
- Rules can be declaratively represented in a verbalisable form whose complexity can be measured by the Kolmogorov complexity

The cognitive cost is a variant of Kolmogorov complexity (Kolmogorov, 1963) proposed in Ai et al. (2021). The cognitive cost estimates the textual complexity of logic terms and atoms declared in the execution stack of working memory. These logic terms and atoms are instances of predicates in a datalog program during the execution of a goal.

A datalog program is a declarative subset of logic programs that represent data structures using predicate arguments but do not use function symbols. Given logic terms and atoms $T_1, T_2, ..., T_{n-1}, T_n$, a tuple of finite arity can be represented as

$$tuple(T_1, tuple(T_2, tuple(..., tuple(T_{n-1}, tuple(T_n, \epsilon))...)))$$

Similarly, a list of finite size can be represented as

$$list(T_1, list(T_2, list(..., list(T_{n-1}, list(T_n, \epsilon))...)))$$

where ϵ denotes an empty character. The cost of logic terms and atoms is computed based on the length of the representation, in particular, the number of symbols involved.

Definition 10 (*Cognitive cost of a logic term and atom, $C(T)$*): Given T a logic term or atom, the cost of $C(T)$ can be computed as follows:

- $C(\top) = C(\bot) = 1$
- A variable V has cost $C(V) = 1$
- A constant c of a finite length has cost $C(c)$ which is the number of digits and characters in c
- An atom $Q(T_1, T_2, ...)$ of a finite arity has cost $C(Q(T_1, T_2, ...)) = 1 + C(T_1) + C(T_2)+...$

Example 3 An atom $list(1, list(0, \epsilon))$ which represents a list [1, 0] has a cognitive cost $C(list(1, list(0, \epsilon))) = 5$.

Example 4 An atom $merger(State1, State2)$ with variables $State1$ and $State2$ has a cognitive cost $C(merger(State1, State2)) = 3$.

In the event of a query q, an execution stack resembles the mental computation and represents the memory of grounded logic terms and atoms instantiated.

Definition 11 (*Execution stack of a datalog program, $S(P, l, q)$*): Given a query q and a non-negative integer l, the execution stack $S(P, l, q)$ of a datalog program P is a finite set of size l of atoms or terms evaluated during the execution of P to compute an answer for q. An evaluation in which an answer to the query is found ends with value $\top$, and an evaluation in which no answer to the query is found ends with $\bot$.

The cognitive cost represents the tax of memorising goals in the human short-term memory. The effort of maintaining goals in working memory correlates to the performance

of human problem-solving (Carpenter et al., 1990). The complexity of executing a set of rules is estimated by the sum of the cognitive cost of logic terms and atoms in the execution stack. The depth bound of an execution stack denotes the limitation of working memory and ensures the computation of the sum of cognitive costs halts.

Definition 12 (*Cognitive cost of a datalog program, Cog*(P, q)): Given a query q, a non-negative integer l, the cognitive cost of a datalog program P is

$$Cog(P, q) = \sum_{t \in S(P,l,q)} C(t)$$

Example 5 The program *merger*/2 in Table 1 takes two input *Lt_expr* expressions in the expression list of a world state and produces an expression *Lt_expr* in the memory of the output world state. The output expression contains integers of increasing magnitude from left to right connected by the "<" symbol. Given $ex_1 = list(1, list(0, \epsilon))$, $lb_1 = \epsilon$, $rb_1 = \epsilon$ and $m_1 = \epsilon$, $s_1 = tuple(ex_1, tuple(0, tuple(lb_1,tuple(rb_1,m_1))))$ denotes the initial world state and $C(s_1) = 13$. Let $ex_2 = list(\epsilon, \epsilon)$, $lb_2 = list(1, \epsilon)$, $rb_2 = list(0, \epsilon)$ and $m_2 = \epsilon$. $s_2 = tuple(ex_2, tuple(0, tuple(lb_2,tuple(rb_2,m_2))))$ denotes the world state after executing $parse\_expr/2$ and $C(s_2) = 15$. $parse\_expr/2$ takes two expressions from the expression list and puts one expression into the left bag and puts the other into the right bag. Let V_1 and V_2 be variables. The length of the execution stack denoted by $S(merger, 4, merger(s1, V_1))$ is 4. The cognitive cost of executing *merger*/2 given the query $merger(s_1, V_1)$ is 60.

$S(merger, 4, merger(s1, V_1))$	$C(T)$
$merger(s_1, V_1)$	15
$parse\_expr(s_1, V_2)$	15
$parse\_expr(s_1, s_2)$	29
$\perp$	1
$Cog(merger, merger(s_1, V_1))$	**60**

In human problem solving, the role of background knowledge is to facilitate the transfer of existing solutions (Anderson & Thompson, 1989; Novick & Holyoak, 1991) to the current context. In the case that auxiliary knowledge is absent, the construction of the solution relies on lower-level sub-goals and performance is limited by human operational error. A primitive problem solution thus denotes a program that involves the minimum amount of background knowledge of the task.

Definition 13 (*Minimum primitive solution program,* $\bar{M}(\phi, E)$): Given a set of primitives ϕ and examples E, a datalog program learned from examples E using a symbolic machine learning algorithm $\bar{M}$ and a set of primitives $\phi' \subseteq \phi$ is a minimum primitive solution program $\bar{M}(\phi, E)$ if and only if for all sets of primitives $\phi'' \subseteq \phi$ where $|\phi''| < |\phi'|$ and for all symbolic machine learning algorithm M' using ϕ'', there exists no machine-learned program $M'(E)$ that is consistent with examples E.

Definition 14 (*Cognitive cost of a problem solution,* $CogP(E, \bar{M}, \phi, q)$): Given examples E, primitive set ϕ, a query q and a symbolic machine learning algorithm $\bar{M}$ that learns a minimum primitive solution, the cognitive cost of a problem solution is

$$CogP(E, \bar{M}, \phi, q) = Cog(\bar{M}(\phi, E), q)$$

where $\bar{M}(\phi, E)$ is a minimum primitive solution program.

The presence of informative knowledge enables efficient problem-solving as projections from background knowledge and allows shortcuts in the construction of the current solution process. In comparison with a primitive problem solution, executional shortcuts in a non-primitive solution could lead to less cognitive load on working memory and fewer performance errors during the execution of the solution.

3.5 Conjectures of human comprehension

Michie (1982) discussed the effect of the representational and executional cost of a program on human understanding. He described the notion of a human window in terms of a class of programs with the right balance of storage and computational complexity that are fit for human understanding. The cognitive window framework (Ai et al., 2021) shares Michie's view of the human window and describes two constraints of machine-learned theory in regard to its effects on human comprehension. The first constraint relates to the size of the hypothesis space associated with the representation of a machine-learned logic theory. This constraint specifies the limitation of human search in the hypothesis space.

Conjecture 1 (*Cognitive bound on the hypothesis space size*, $B(P, H)$): *Consider a symbolic machine-learned datalog program P using p predicate symbols and m meta-rules each having at most j body literals. Given a group of humans H, $B(P, H)$ is a population-dependent bound on the size of hypothesis space such that at most n clauses in P can be comprehended by all humans in H and $B(P, H) = m^n p^{(1+j)n}$.*

The conjecture of the human searchable hypothesis space size $B(P, H)$ relates to the MIL complexity analysis in Cropper (2017); Lin et al. (2014). The authors in Ai et al. (2021) demonstrated empirical evidence which supports Conjecture 1. Humans learned can only learn some of the rules for a two-players simple game when the descriptive size of the rules is high. This implies that the hypothesis space associated with the program class is too large for a complete search and imposes a high cognitive load on working memory. Since only a fraction of the original rules is learned, problem solutions which produce action sequences are not complete and errors become more likely to occur.

The cognitive window conjecture describes that a machine-learned theory has a harmful explanatory effect on comprehension when learning requires a search in a hypothesis space that is too large for working memory to manage. In addition, a machine-learned theory has no beneficial explanatory effect if its cognitive complexity in executing explained knowledge is not lower than the cognitive complexity of a solution that uses less auxiliary knowledge.

Conjecture 2 (*Cognitive window of a machine-learned theory*): *Given a logic program D representing the definition of a target predicate, a machine learning algorithm M, a minimum primitive solution learning algorithm $\bar{M}$ and examples E, $M(E)$ is a machine-learned theory using the primitive set ϕ and belongs to the program class with hypothesis space S. For a human group H, E_{ex} satisfies both*

1. $E_{ex}(D, H, M(E)) < 0$ *if* $|S| > B(M(E), H)$
2. $E_{ex}(D, H, M(E)) \leq 0$ *if* $Cog(M(E), x) \geq CogP(E, \bar{M}, \phi, x)$ *for all unseen query* x *that* $h \in H$ *have to perform after study*

Explanations are only beneficial if they are of appropriate complexity and are not overwhelming nor more cognitively expensive than a primitive solution to the problem. This cognitive window suggests that complex machine-learned models and models which cannot provide abstract descriptions are difficult to be comprehended by humans effectively.

Remark 1 $Metagol_O$ is provided with a set of composite predicate definitions $A_{comp} = \{parse\_exprs/2, compare\_nums/2, single\_expr/2, drop\_bag\_remaining/2, recycle\_memory/2\}$ to learn *merger*/2 in Table 1. A_{comp} is constructed from a set of primitive action definitions A_{prim}. The composite actions of breaking number sequences into individual numbers (*parse_exprs*/2), comparing two numbers (*compare_nums*/2) and appending two sorted sequences (*drop_bag_remaining*/2) are sufficient for merging and the respective predicates appear in the learned logic program *merger*/2. Given action definitions $A = A_{prim} \cup A_{comp}$, since composite actions A_{comp} are constructed from multiple primitive actions from A_{prim}, A_{comp} is the necessary and sufficient subset of A to learn a definition of merging. This implies that *merger*/2 learned by $Metagol_O(E)$ is the minimum primitive solution. Therefore, given test examples X of merging and sufficient training examples E, $Cog(merger, x) = CogP(E, Metagol_O, A, x)$ for all $x \in X$.

The complexity analysis based on the Blumer bound (Blumer et al., 1989) in Cropper (2019) attributes a sample complexity decrease to the introduction of new predicate symbols to the machine learner. Previously learned predicate definitions extend the background. Reusing learned concepts leads to a decrease in the size of the target hypothesis and a reduction in the size of the hypothesis space which improves learning performance (Cropper, 2019). Based on our assumptions in Sect. 3.4, we consider an analogy between the human reusing of previously learned concepts and the introduction of predicate definitions into a new learning context. By referring to Definition 9, we attribute a beneficial effect of a curriculum on human comprehension over another curriculum to an improvement in the sample complexity based on the analysis in Cropper (2019).

Conjecture 3 (*Sequential teaching curriculum improvement*): $C_1 = C_{seq}(ST(Ds, Es, \lambda_1, Ms), H)$ *and* $C_2 = C_{seq}(ST(Ds, Es, \lambda_2, Ms), H)$ *are sequential teaching curricula where* Ds *is a set of* n *predicate definitions and* $D \in Ds$ *is a predicate definition*, Es *are sets of examples*, $\lambda_1, \lambda_2 : Ds \times Es \times Ms \rightarrow \mathbb{N}$ *are rank functions*, Ms *is a set of machine learning algorithms and* H *is a human group*. R_1, R_2 *denote ranks of* D *such that* $(R_1, D) \in C_1$ *and* $(R_2, D) \in C_2$. *For* $1 \leq i \leq n$ *such that* $(R_i, D_i) \in C_1$ *and* $R_i < R_1$, *let* p *denote the sum of the number of predicates in* D_i *and* u *be the minimum number of clauses to express* D *using* D_i. *For* $1 \leq j \leq n$ *such that* $(R_j, D_j) \in C_2$ *and* $R_j < R_2$, *let* $p + c$ *denote the number of predicates in* D_j *and* $u + k$ *be the minimum number of clauses to express* D *using* D_j. $E_{seq}(C_1, C_2, D) > 0$ *when*:

$$u \ln(p) < (u + k) \ln(p + c)$$

Given a target predicate D and a fixed human search bound $B(D, H)$ for a human group H, human learners are more likely to find D if the hypothesis space associated becomes smaller as a result of learning new knowledge prior to learning D. For a fixed set of training

examples in two curricula, a reduction in sample complexity correlates to a decrease in predictive error and an improvement in performance.

4 Experimental framework

In previous works (Ai et al., 2021; Muggleton et al., 2018), human comprehension was measured by the accuracy with which participants answered classification questions. The accuracy of problem-solving performance for classification tasks and sequential decision-making tasks can be different due to the nature of the operations involved. In this section, we present our approach to evaluating human sorting performance based on statistics and describe our experimental hypotheses in relation to the theoretical framework.

4.1 Evaluating human comprehension of sorting

For merging and sorting, human comprehension corresponds to the competence of participants to arrange items with respect to the total ordering of the items. Decisions related to the arrangement of items are reflected by the classification of items into respective positions in the sequence. While the out-of-sample classification accuracy is an objective measure of human comprehension for classification tasks, it does not estimate the error of sequential predictions. Although alternative metrics exist, we use Spearman's rank correlation coefficient (Spearman, 1904) as a means to assess the error between the expected sequence and the actual sequence provided by a participant. Spearman's rank correlation coefficient is a non-parametric rank correlation test and indicates the degree of monotonic ordering between two variables. Given two sequences of values representing samples of two variables, when Spearman's rank correlation coefficient is $+1$ or -1, it means the variables are perfectly monotonic functions of each other. On the other hand, when Spearman's rank correlation coefficient tends to 0, it indicates there is no monotonic relationship between the two variables. We use Spearman's rank correlation coefficient to examine the extent of monotonic alignment between a sequence of integers that has been provided by a human and the correctly arranged sequence produced by a machine algorithm.

In addition, we take into consideration the possibility that participants provide incorrect answers that are inversely sorted with respect to the order specified in the experiment instructions. In this case, a coefficient score tending to -1 would mean that the participant successfully managed to sort the sequence but forgot the specified ordering. In the experiment trials, we found that such answers rarely occurred. However, in order to avoid diluting the significance of results due to the inclusion of negative coefficients, we define the following function to discount negative coefficients and normalise all of them into the interval [0, 1].

Definition 15 (*Normalised performance score, perf*(R_h, R_m)) Given two sets of integers R_h and R_m, a defined discounting constant $0 < c < 1$ and the Spearman's rank correlation coefficient ρ, the normalised performance score is defined as

$$perf(R_h, R_m) = \begin{cases} \rho(R_h, R_m), & \rho(R_h, R_m) \geq 0 \\ |\rho(R_h, R_m)| \cdot c, & \rho(R_h, R_m) < 0 \end{cases}$$

Table 4 We consider two independent experimental variables

Variable name	Variable type	Treatment abbreviations
Curriculum order (**CO**)	Independent variable	**MS**, **SM**
Presence of explanations (**EX**)	Independent variable	**WEX**, **WOEX**
Performance score (**PS**)	Dependent variable	–

The order of concepts in the curriculum (**CO**) is varied between learning merging before learning sorting (**MS**) and learning sorting before learning merging (**SM**). The presence of explanations generated from machine-learned rules (**EX**) alters between learning with explanations (**WEX**) and learning without explanations (**WOEX**). The performance score of human participants (**PS**) is the dependent experimental variable

Example 6 shows the performance evaluation of two unordered sequences by comparing them to a sorted sequence.

Example 6 Let R_{h1} denote the sequence *[4, 6, 5, 2, 3, 1]*, R_{h2} denote the sequence *[1, 2, 6, 3, 4, 5]* and R_m be the perfectly monotonic sequence *[1, 2, 3, 4, 5, 6]* sorted by a machine program. Compared with R_m, R_{h2} has only one number that is out of place. However, R_{h1} is almost a sequence with decreasing magnitude with numbers 2 and 4 misplaced. Given R_m is a monotonically increasing sequence, R_{h2} aligns better with the machine output R_m and $perf(R_{h1}, R_m) = .386 < perf(R_{h2}, R_m) = .657$.

The Spearman's rank correlation coefficient lies in the interval $[-1, 1]$ and the absolute value of the negative coefficient is multiplied by a discounting constant. Based on results from multiple trials, the discounting constant c is set to .5 to moderately penalise responses that involve reversed sorted sequences.

4.2 Experimental hypotheses

We define our experimental hypotheses in this section. Table 4 shows the experimental variables and treatments. We examine the effect of the curriculum order (**CO**) via two teaching arrangements: (a) learning merging before learning sorting (**MS**) and (b) learning sorting before learning merging (**SM**). In addition, we assess the effect of the presence of explanations (**EX**) in two environments: (a) learning with explanations of merging (**WEX**) (b) learning without explanations of merging (**WOEX**).

Explanations are not provided for learning sorting to avoid introducing merge prematurely in the curricula that learn sorting before learning merging. We use the normalised performance score *perf* in Definition 15 to measure both human performances of merging and sorting (**PS**).

Let $Ds = \{merger, sorter\}$ denote target theories of merging and sorting. Let $Es = \{E_{merge}, E_{sort}\}$ where E_{merge} denotes a sufficient set of examples for learning the target theory of merging and E_{sort} is a sufficient set of examples for learning the target theory of sorting. Both human learners and $Metagol_O$ are provided with the same sets of examples Es. Let $Ms = \{Metagol_O, \emptyset\}$ denote the machine learning algorithms used to learn rules for generating explanations and $\emptyset$ is an empty program. H stands for a human group. The rank functions for curricula that learn merging before learning sorting are defined as $\lambda_{MS/WEX}$ and $\lambda_{MS/WOEX}$. The rank functions of curricula that learn merging before learning sorting

Table 5 Definitions of the rank functions. In curricula **MS/WEX** and **MS/WOEX**, humans learn merging before learning sorting

Curricula abbreviation	Rank function definitions
MS/WEX	$\lambda_{MS/WEX}(merger, E_{merge}, Metagol_O) = 0$
	$\lambda_{MS/WEX}(sorter, E_{sort}, \emptyset) = 1$
MS/WOEX	$\lambda_{MS/WOEX}(merger, E_{merge}, \emptyset) = 0$
	$\lambda_{MS/WOEX}(sorter, E_{sort}, \emptyset) = 1$
SM/WEX	$\lambda_{SM/WEX}(merger, E_{merge}, Metagol_O) = 1$
	$\lambda_{SM/WEX}(sorter, E_{sort}, \emptyset) = 0$
SM/WOEX	$\lambda_{SM/WOEX}(merger, E_{merge}, \emptyset) = 1$
	$\lambda_{SM/WOEX}(sorter, E_{sort}, \emptyset) = 0$

In curricula **SM/WEX** and **SM/WOEX**, humans learn sorting before learning merging. The rank functions of **MS** assign a lower rank value to merging. The rank functions of **SM** assign a lower rank value to sorting. In curricula **MS/WEX** and **SM/WEX**, explanations about merging are generated from $Metagol_O$

Table 6 A summary of experimental independent variables, definitions and results related to each experimental hypothesis

Hypotheses	Independent experimental variables	Related definitions
H1	**CO**	
H2	**EX**	Definition 5 to 9, 15 and Conjecture 3
H3	**CO** and **EX**	
H4	**EX**	Definition 1 to 4, 15 and Conjecture 2
H5	**EX**	Definition 15

are defined as $\lambda_{SM/WEX}$ and $\lambda_{SM/WOEX}$. The definitions of the rank functions are summarised in Table 5.

Let $C_{MS/WEX} = C_{seq}(ST(Ds, Es, \lambda_{MS/WEX}, Ms), H)$ denote the human comprehension of the sequential teaching curriculum that learns merging before learning sorting with explanations of merging. Let $C_{MS/WOEX} = C_{seq}(ST(Ds, Es, \lambda_{MS/WOEX}, Ms), H)$ denote the human comprehension of the sequential teaching curriculum that learns merging before learning sorting without explanations of merging. Let $C_{SM/WEX} = C_{seq}(ST(Ds, Es, \lambda_{SM/WEX}, Ms), H)$ denote the human comprehension of the sequential teaching curriculum that learns sorting before learning merging with explanations of merging. Let $C_{SM/WOEX} = C_{seq}(ST(Ds, Es, \lambda_{SM/WOEX}, Ms), H)$ denote the human comprehension of the sequential teaching curriculum that learns sorting before learning merging without explanations of merging. We then introduce hypotheses that focus on how curriculum order and explanations generated from machine-learned rules affect human learning of sorting. In Table 6, we show how the following experimental hypotheses relate to the conjectures and our framework.

H1: *Learning merging before learning sorting leads to a beneficial effect on human comprehension of sorting* ($E_{seq}(C_{MS/WEX}, C_{SM/WEX}, sorter)$ $E_{seq}(C_{MS/WEX}, C_{SM/WEX}, sorter) + E_{seq}(C_{MS/WOEX}, C_{SM/WOEX}, sorter) > 0$) *with respect to learning sorting before learning merging*.

In **H1**, we examine whether human learners achieve better sorting performance from the curriculum with increasing concept complexity (**MS**) than from the curriculum with decreasing concept complexity (**SM**).

H2: *Learning merging with explanations results in a beneficial explanatory effect on human comprehension of sorting* $(E_{seq}(C_{MS/WEX}, C_{MS/WOEX}, sorter) + E_{seq}(C_{SM/WEX}, C_{SM/WOEX}, sorter) > 0)$ *with respect to learning merging without explanations.*

In **H2**, we assess if learning merging with aids in the form of explanations constructed from machine-learned rules (**WEX**) improves sorting performance compared with learning merging without explanations (**WOEX**).

H3: *Learning merging with explanations further increases the beneficial effect of curriculum order on human comprehension of sorting* $(E_{seq}(C_{MS/WEX}, C_{MS/WOEX}, sorter) - E_{seq}(C_{SM/WEX}, C_{SM/WOEX}, sorter) > 0)$ *with respect to learning merging without explanations.*

In **H3**, we inspect if there is an interaction effect between curriculum order (**CO**) and explanations (**EX**) on sorting performance.

We suspect that for human learners who have no previous programming experience, learning merging in the presence of explanations results in adapting new sorting strategies in comparison with learning merging in the absence of explanations. Since merging can be taught in isolation and it does not depend on the knowledge of sorting in the curricula, we refer to Definition 4 and the cognitive window to account for the effect of explanations on merging. However, owing to $Metagol_O$ learning a minimum primitive solution *merger*/2, we anticipate no beneficial effect on human comprehension of merging as a result of learning from explanations. Let A denote the background knowledge of $Metagol_O$ containing definitions of primitive and composite actions and X denote questions that the human group H performs in the merge performance test.

H4: *Learning merging with explanations generated from rules without a low cognitive cost* $(Cog(merger, x) \geq CogP(E_{merge}, Metagol_O, A, x)$ for all $x \in X)$ *does not result in a beneficial explanatory effect on human comprehension of merging* $(E_{ex}(merger, H, merger) \leq 0)$.

In **H4**, we examine if learning with explanations of merging (**WEX**) results in no observable performance difference or worse performance with respect to learning without explanations of merging (**WOEX**).

H5: *Learning merging with explanations before learning sorting leads to adaptation of efficient human sorting strategies with better performance.*

We consider executed comparisons of human participants as problem-solving traces. In Sect. 5.4, we estimate through human trace analysis the correspondence between human sorting strategies and machine sorting algorithms in training and performance tests. An underlying assumption is that learning a new strategy during the performance test is challenging without feedback and strategy adaptation is easier during the training stage. In addition, we assume that the presence of explanations and the incremental curriculum contribute to efficient and effective human decision-making. Therefore, in **H5**, we inspect the performance of specific sorting strategies that increase in usage from sort training to sort performance test.

5 Experiments

We first describe the implementation of a pre-test which was given to all participants at the beginning of the experiment. In addition, we report the masking and presentation of the sorting task in Sect. 5.2. In Sect. 5.3, we explain the setup of the experiment which was

carried out on a crowdsourcing platform Amazon Mechanical Turk. Results are demonstrated in Sect. 5.4 which is followed by a discussion of our findings.

5.1 MaRs-IB pre-test

We have reason to assume that the participant's performance in the experimental task varies depending on their cognitive abilities. Recording their cognitive ability provides us with two possibilities:

1. It allows statistical control of the participants' cognitive ability.
2. It can be determined whether the mean cognitive ability in all experimental conditions is the same. If that is the case, mean differences in performance between the conditions can be interpreted without having to control for each participant's cognitive ability first.

There have been two major suggestions for recording this control: either confront participants with a version of the experimental task itself or use a short intelligence test in the form of a Raven Matrices short test (Arthur & Day, 1994) or similar material. We argue that using the experimental task will undermine the sequential nature of our experiment, as either group would start with the low-complexity task or with the high-complexity task. A third option is for each group to start with their respective first task, which would introduce task familiarity effects (the high-complexity-first condition would have completed more high-complexity tasks than the low-complexity-first condition). This approach does not pose a good control since each group would complete a different task.

Using an intelligence test avoids interference with the experiment since the test should not influence strategy development in the experimental task in a way that could be considered to work against the sequential order of the experiment. However, the original Raven Matrices test and many comparable tests are expensive and cannot be digitized due to copyright. The matrix reasoning item bank (MaRs-IB) has neither of these two issues (Chierchia et al., 2019), thus making it an appropriate candidate for establishing a measure of intelligence in an online experiment. In addition, the authors of the MaRs-IB test set reported acceptable internal consistency and convergent validity, and reasonable test-retest reliability (Chierchia et al., 2019). Although there is no normalized score which would be comparable to the IQ, the MaRs-IB test can serve as a measure of the relative cognitive ability of participants. Another concern regards the open access to the material (Chierchia et al., 2019) and familiarity with the test from other studies. This risk can be mitigated by randomised arrangement and the selection of items from the multiple test sets provided by the authors.

5.1.1 Application

The authors Chierchia et al. (2019) provided three sets of 80 items (Fig. 2). Each of these sets uses different symbols and is available in three further variations though colour-vision-deficient subjects were informed not to participate. The testing procedure described in the validation paper (Chierchia et al., 2019) will be consistently completed in under 10 min (8 min test and a series of practice items beforehand) and is thus applicable for our use case. Participants will train on 10 practice items until they completed three of them correctly. An item in the full test starts with a 500 ms fixation cross, followed by a 100 ms white screen mask (Chierchia et al., 2019). The matrix will then be displayed for 30 s, or

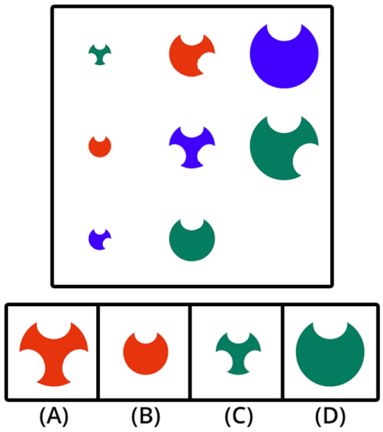


Fig. 2 Each MaRs-IB item consists of an incomplete 3×3 matrix in which 8 cells with shaped icons depict a modification rule to (an) abstract shape(s). The test asks the participant to fill in the missing piece at the right bottom corner of the 3×3 matrix by choosing one of four provided options (A, B, C and D). The correct answer to this example is A

until the participant responded (whichever happens earlier). After 25 s a clock indicates that participants have 5 s left for this specific item (Chierchia et al., 2019). Participants complete these items for a total of 8 min. The accuracy with which the participant chooses the correct missing pattern in the given time frame is considered the measurement of pre-test performance.

5.2 Materials

The arrangement of objects arguably makes up a great proportion of human daily tasks. This implies a high familiarity of human participants with the task of sorting. The preferred sorting approaches of humans with no background knowledge in programming could resemble some sorting algorithms. However, we assume that learning recursive algorithms such as merge sort without a well-defined curriculum demands a great effort and applying such algorithms efficiently is challenging.

5.2.1 Problem masking

In order to prevent participants from relying on the total ordering of natural numbers, we designed an isomorphic problem that represented integer sequences by the weights of fruits in piles. Numbers were masked by visual icons of fruits and the numeric values were hidden from participants. A balance scale mechanism (Fig. 3) was introduced to allow participants to compare the weights between two fruits and acted as an instrument for constructing the total ordering of the numbers. The immediate access to the total ordering of natural numbers was prevented and participants had to establish it themselves. This setup was used for teaching both merging and sorting. A solution of this setup corresponded to a solution of merging and sorting a sequence of numbers without changing the underlying solution structure.

According to the definition of isomorphic problems (Simon & Hayes, 1976), the new fruit weighing problems are isomorphic to the respective problems for merging and sorting numbers. Such isomorphic translations only affect the initial stage of problem-solving while attempting to recognise a useful analogy via *analogical access* but do not hinder problem-solving via *analogical inference* if an analogy for the problem has been

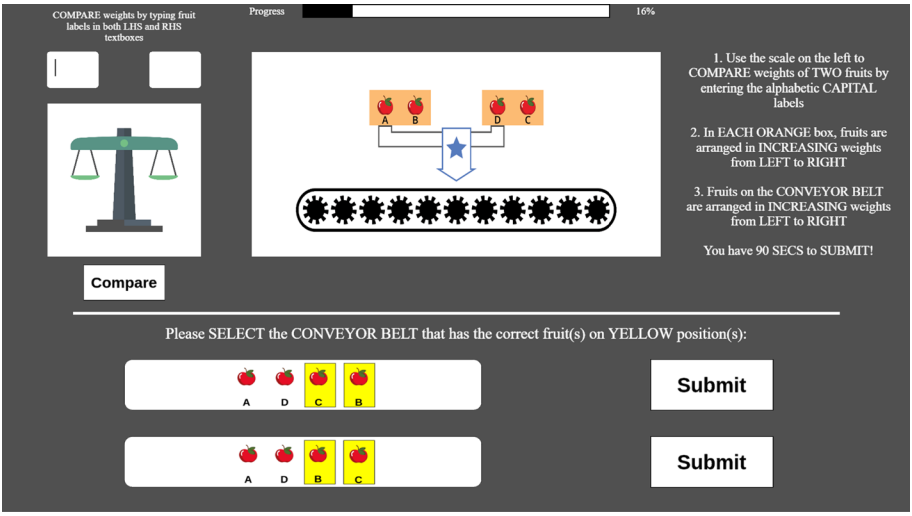


Fig. 3 Participants were asked to distinguish between two alternative choices by using the provided balance scale and submit the correct sequence by pressing a button to the right

consciously recognized (Gentner & Landers, 1982; Holyoak & Koh, 1987; Reed et al., 1990). Owing to the difficulty of immediately recalling past experiences of merging and sorting, we anticipated that participants would initially perform less optimal merging and sorting strategies and there would be more potential for performance improvement later.

For the training and performance test of merging and sorting, we randomly sampled sequences of various lengths to provide a spectrum of problem difficulty. For each of the question sequences, we made sure that it was not trivial and required an appropriate level of computational effort for a machine program to sort. A question in the merge training and performance test involved two sub-sequences of similar sizes as inputs and the length of each sub-sequence ranged from 1 to 4.

In the sort training and performance test, a question consists of a fruit sequence and the length of sequences varied from 6 to 10. In the pilot trials, we identified a higher usage of the insertion sort algorithm. For generating sort training and testing questions, we made an adjustment to include sequences that would lead to an advantage in the number of comparisons made when applying the merge sort algorithm over the insertion sort algorithm. This change would allow room for people to optimise their sorting strategy instead of adhering to what they already knew. The sequences in performance tests are different from those in the training stages but they share similar patterns.

5.2.2 Problem setup

To make the experiment more engaging, we introduced the task background where each participant was asked to help two robots, Alice and Bob, perform tasks in two warehouses. Each robot was responsible for a distinct task and provided information about the associated task to the participant. The robot Alice was designed to teach merging while the other one Bob taught participants how to sort. To evoke an understanding of the connection between merging and sorting, at the beginning of the

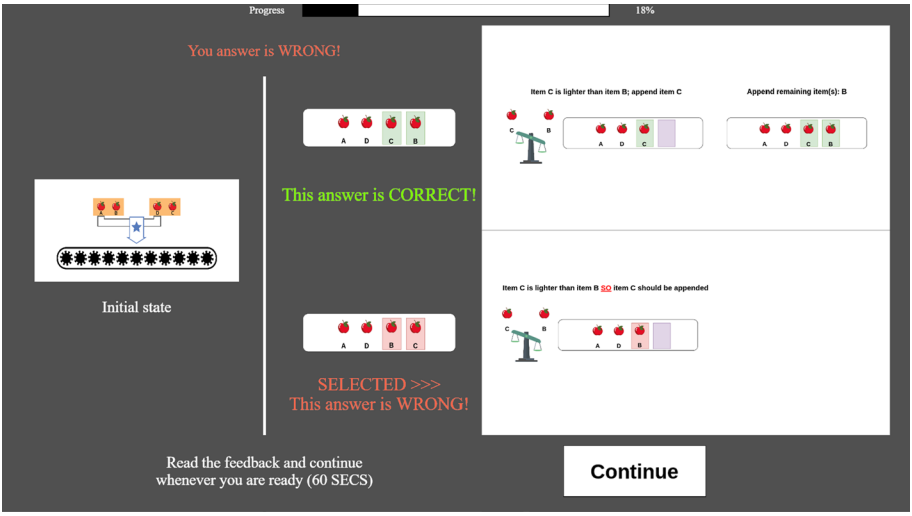


Fig. 4 On the top row, the visualisations show the comparison result (fruits B and C). The textual explanations explicitly describe the objects involved (fruits B and C) and the actions performed (comparison and appending) which resulted in C being appended before B due to C's lesser weight. On the bottom row, the interface highlighted in red that fruits were out of order (B and C) and described the correct ordering

experiment, participants were explicitly reminded with a note of the link between task materials taught by two robots and were encouraged to pay attention to the connection.

To learn merging, participants were asked to learn from the robot Alice and help it operate a special machine to arrange fruits on a conveyor belt. In Fig. 3, fruits labelled with A, B, C and D were inputs to a machine called the "blue star" that represented the merging operation. Each question in the sections related to merging asked the participant to merge two sequences of fruits.

The balance scale to the left of the interface page (Figs. 3, 4 and 5) allowed participants to compare the weights of fruits by entering two fruits' alphabetic labels on both sides of the balance. Participants were presented with instructions to use the balance scale. The balance would tilt to the side of the fruit that is heavier. A piece of text would be shown to further clarify the result of comparing two fruits. Participants were provided with the information that the two input sequences of fruits in the questions had increasing weights from left to right. Two output answer choices differed in the fruits that were highlighted with the yellow colour. Participants were asked to select from one of the two output answer choices that had the fruits in the correct order.

Feedback was presented as shown by Fig. 4 on whether the participant's selection was correct. Based on the same initial problem state, this feedback provides a pair of positive and negative examples which helps contrast the correct and wrong sequence of decision-making. The white blocks on the right-hand-side of Fig. 4 contain textual and graphical explanations that instantiate the logic rules learned by $Metagol_O$ in Table 1. Groups without machine-learned explanations would only see blank white boxes with no textual and graphical explanations during the merge training stage of the experiment.

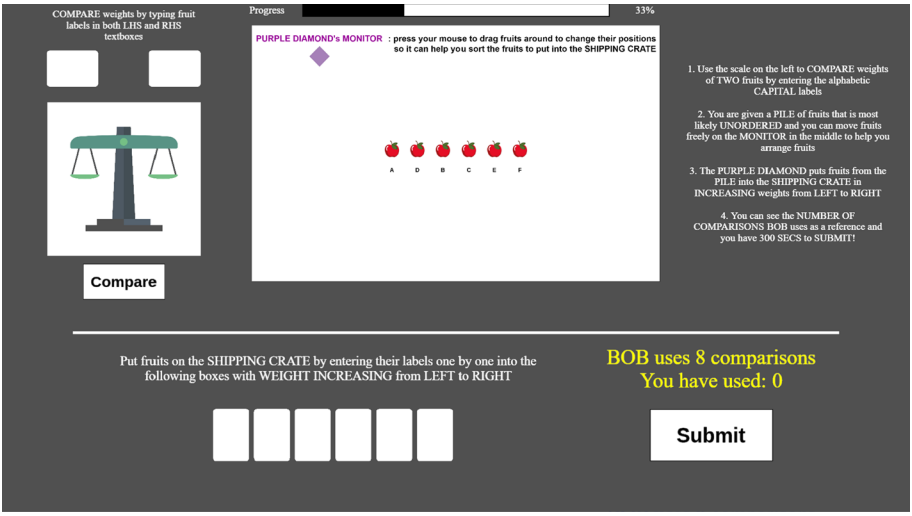


Fig. 5 Sort training interface. In sort training, participants were advised to sort sequences of fruits (in this case fruits A, D, B, C, E and F) in the centre white box called the "whiteboard" and provided instructions to efficiently use it. The "whiteboard" was a device of the interface which allowed a free arrangement of the input fruit sequence by using a mouse to drag and drop fruits. The labels and fruits were intentionally randomised before the experiment so that participants would not receive the wrong impression that the sequence had been already alphabetically ordered

5.3 Method and design

The experiment,[2] was implemented based on a four-group factorial 2×2 design to account for all combinations (Table 7) of the two independent experimental variables, the curriculum order (**CO**) and the presentation of explanations (**EX**). Figure 6 shows two curriculum orders. In the curriculum shown by Fig. 6a, participants received training material on merging first. In the reverse curriculum demonstrated by Fig. 6b, subjects were trained on sorting first.

Participants were not allowed to repeat the experiment. Every participant was assigned to one group and went through training and test with no repeat. All groups received the same material of training examples and test questions. The numbers of questions in the experiment sections merge training, merge testing, sort training and sort testing were 6, 5, 4 and 8 respectively. However, the number sequences used for merging were different from those used for sorting in order to prevent participants from reusing previous answers.

We recorded participants' basic information such as their age, gender and education after acquiring their consent. This allows us to control the distribution of background in experiment groups. To control participants' cognitive ability, we then collected their responses in the MaRs-IB pre-test based on Sect. 5.1. The experiment would start with a participation consent followed by the MaRs-IB pre-test and an introduction to the task background. Participants were advised prior to the beginning of the experiment on the

[2] The experiment interface is available on https://github.com/LAi1997/sequential-teaching. The interface was created using PsychoPy (Peirce et al., 2019) an open-source package for implementing free interfaces with stimulus presentation and control in Python and JavaScript.

Table 7 Group 1 learned merging before learning sorting and was provided with visual and textual explanations of merging based on machine-learned rules

CO	EX	
	WEX	WOEX
MS	Group 1 (MS/WEX)	Group 2 (MS/WOEX)
SM	Group 3 (SM/WEX)	Group 4 (SM/WOEX)

Group 2 received the same curriculum as Group 1 but without explanations. The participants in Group 3 learned sorting prior to learning merging with explanations in the same format as received by Group 1. Group 4 did not get explanations

approximate length of the experiment and the requirements. Participants were informed that if they have colour-vision deficiency, they should not participate in the study.

In order to gain information about participants' experience relating to programming and sorting algorithms, we additionally asked participants to fill out the following questions after the completion of all performance tests: (1) whether participants have a degree in computer science or a certificate in programming, (2) if they have studied or are learning sorting algorithms in school and (3) whether they know or have applied any sorting algorithms while being presented with a list of sorting algorithms (two of which are distracting choices and do not exist).

Task-related information was presented via the interface throughout the experiment. All groups received a performance test of sorting last. This experiment design ensured all groups experienced the same amount of training and performance tests respecting the curriculum order.

In merge training, participants first answered the displayed question illustrated by Fig. 3 and were taken to the next page where feedback and explanations were presented. In Fig. 4, explanations are presented in two rows which demonstrate the correct and the wrong action sequences. The format in which participants were asked to input their responses is the same in the merge performance test, sort training and sort performance test. When providing a sequence response, participants were required to input labels representing the fruit sequence that they believed to have increasing weights from left to right. Another robot instructor Bob helped participants learn to operate the machine "purple diamond" to sort fruits.

In the sort training and performance test, participants should put fruits into shipping boxes by entering fruit labels into a sequence of text boxes. In the sort training (Fig. 5), participants were informed of the number of comparisons their robot instructor Bob made

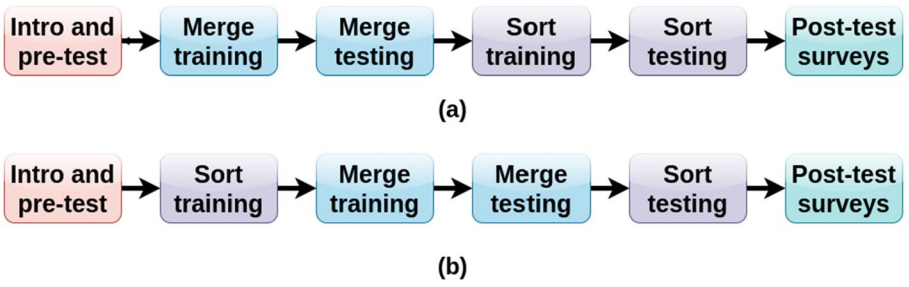


Fig. 6 In setup **a**, participants learned merging before learning sorting (MS). In setup **b**, participants learned sorting before learning merging (SM)

to sort the input sequences. Participants were asked to sort the same input sequences and encouraged to use fewer comparisons. The intention of this competition was to provide an incentive for participants to consciously think about how to revise their strategy in order to improve sorting efficiency. This information was accessible by participants in all groups during the sort training. In the sorting sections, participants were also provided with the balance scale instrument.

Participants were encouraged to take two sessions of one-minute rest, one after the MaRs-IB pre-test and one prior to the final performance test of sorting. The break sessions allowed participants to recharge and regain attention to perform tasks in later stages. The progress bar at the top of the interface showed the percentage of materials that the participants had completed with respect to all materials in the experiment. Participants could use the progress bar to estimate the time that they would need to finish the experiment within the allocated session. For all training and performance test sections, we recorded response time, answers and the comparisons made for analysis. The final section included four reflective questions. We asked participants whether they understood the connection between the two tasks or used what they learned in one task for the other task. We also checked their background in programming and sorting algorithms in the survey at the end of the experiment.

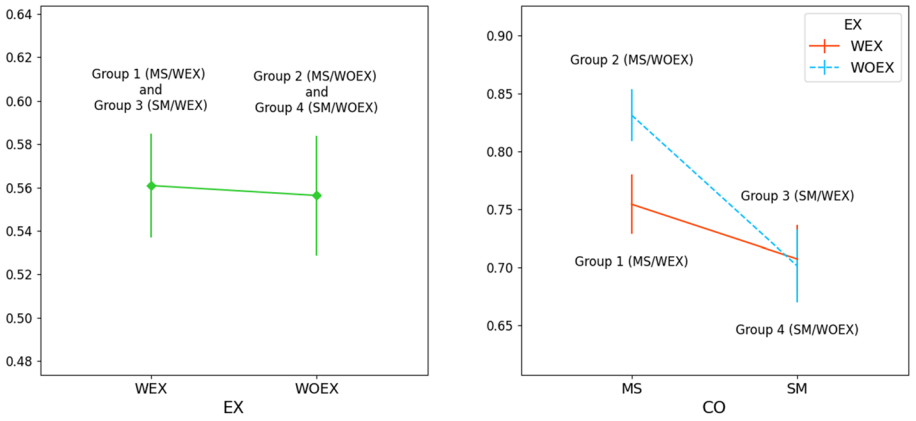
5.4 Results

Prior to the experiment, we conducted two experimental trials with one sample of students from the University of Bamberg and another student sample from Imperial College London. The trial studies provided additional insights into the interface and helped us revise the experimental design. However, due to the limited scale of the trials, we were not able to observe any meaningful outcome. We proceeded to recruit a larger sample of 124 mixed-background participants from Amazon Mechanical Turk (AMT). These 124 participants from AMT were randomised and assigned to the four groups. A partition of 79 participants was created based on the MaRs-IB test and those included had a test accuracy within one standard deviation ($\sigma = .169$) of the mean ($\mu = .654$). Demographically, this partition had a lower male-to-female ratio (36 to 42, one opted out), most participants were between 25 to 54 in age (61 of 79) and almost all had a college degree or higher (67 of 79).

5.4.1 Analysis of task performance

We focused on this AMT sample partition and employed a quantitative approach to test hypotheses **H1** to **H5**. The AMT partition had an even distribution of participants in the four groups, Group 1 (**MS** and **WEX**, $n = 20$), Group 2 (**MS** and **WOEX**, $n = 20$), Group 3 (**SM** and **WEX**, $n = 20$) and Group 4 (**SM** and **WOEX**, $n = 19$). Figure 7a, b present the mean performance score (**PS**) of the groups in the merge performance test and sort performance test. For each performance test, we carried out an ANOVA test to assess the effect of the independent experimental variables followed by a Tukey's HSD test. We used a significance criterion $\alpha = .05$ for both tests.

Figure 7a illustrates almost no change to the mean merge performance score when explanations were introduced. The one-way ANOVA test did not demonstrate a significant effect of the machine-learned theory (**EX**) on the merge performance test score (**PS**). Figure 7b illustrates a pattern that switching from the curriculum order **MS** to **SM** results in a reduction in performance. The two-way ANOVA test indicated a significant effect



(a) Mean merge performance test scores and standard errors.

(b) Mean sort performance test scores and standard errors.

Fig. 7 Performance test scores (PS) of four groups, Group 1 (MS/WEX) Group 2 (MS/WOEX), Group 3 (SM/WEX) and Group 4 (SM/WOEX) with standard errors. Across the x-axis in **a**, data points indicate the change in the mean merge performance score as a result of learning with explanations. In **b**, the red line shows the mean sort performance score when participants learned with explanations and the blue line represents the mean sort performance score when participants learned without explanations. Across the x-axis in **b**, data points show the change in the mean sort performance score as a result of altering the curriculum arrangement

($F = 10.4, p = .001 < .05$) of curriculum order (**CO**) on sort performance test score (**PS**). The post-hoc Tukey's HSD test confirmed a significantly beneficial effect ($p = .001 < .05$) of the sequential teaching curriculum **MS** (Group 1 and Group 2) over **SM** (Group 3 and Group 4). The two-way ANOVA test did not show a significant effect of the machine-learned theory (**EX**) on sort performance test score (**PS**) and there is no evidence of an interaction effect between curriculum order (**CO**) and presence of explanations (**EX**).

5.4.2 Analysis of human sorting strategy

In addition to the performance score, we analyse the correspondence between human sorting strategies and machine sorting algorithms. For sorting, sequential decision-making is revealed through the actions that human sorters make to arrange items according to a specified order. In particular, we consider the comparisons of objects as traces of decision making which indicate the sorting strategy used by human sorters and the sorting algorithm implemented by a machine program.

When there is a consistently high correlation of comparison execution between a human sorter and a machine algorithm, we assume that this human executes a strategy that closely implements the machine algorithm. Considering the target human population has limited programming experience, we additionally assume that a human sorting strategy would implement at best quick sort, merge sort, insertion sort, bubble sort or dictionary sort. Based on the results and preferences of the trial participants, we include hybrid variants of insertion sort and dictionary sort. These hybrid variants initially perform insertion sort until the intermediate result reaches length k and then execute binary searches of the conventional dictionary sort to insert the rest of the numbers into the partially sorted sequence. In total, we implemented twenty-four algorithms including the conventional

implementations of quick sort, merge sort, insertion sort, bubble sort and dictionary sort as well as their variants (different pivoting, merge and insertion implementations).

Example 7 Given $Ns = [4, 6, 5, 2, 3, 1]$ is a sequence of numbers for a human participant h and a set of machine algorithms to sort. Let M_h denote the human sorting algorithm and $ht(Ns, M_h) = [(6, 4), (5, 2), (3, 1), (4, 2), (5, 4), (6, 5), (2, 1), (3, 2), (4, 3)]$. We are able to find a close match of the human trace. The machine trace is produced by the bottom-up variant of the merge sort algorithm denoted as M' and $mt(Ns, M') = [(4, 6), (5, 2), (2, 4), (4, 5), (5, 6), (3, 1), (1, 2), (2, 3), (3, 4)]$. For each of the $N \times N$ combinations of number pairs, we check whether it is a member of the traces. Two symmetric pairs of numbers, e.g. (4, 6) and (6, 4), are considered identical. We create the following 2×2 contingency table.

	Not in human trace	In human trace
Not in machine trace	13	1
In machine trace	1	10

The Chi-squared test[3] in this example has a $X^2 = 14.3$ with $p < .001$ which confirms the correlation of number pairs in $ht(Ns, Mh)$ and $mt(Ns, M')$. For common pairs of numbers in $ht(Ns, Mh)$ and $mt(Ns, M')$, we obtain their ranks and compute Spearman's rank correlation coefficient. The correlation coefficient is.9 and $p < .001$ which confirms the monotonic relationship between the two traces.

Given an input set of integers Ns and a sorting algorithm M, a sorting trace $M(Ns)$ is a list of pairs $< n_i, n_j >$ denoting the comparisons of integers made by M where $n_i, n_j \in Ns$. We employ two statistical tests (Example 7) to estimate the correspondence between a human sorting trace and a machine sorting trace. Let n be an integer and Ns denote the set of integers in the interval $[1, n]$. Let Tr be a cross product defined by $Tr : Ns \times Ns$, a comparison in a trace is a pair $< n_i, n_j > \in Tr$ where $n_i, n_j \in Ns$. Given a human trace with sorting algorithm M_h and a machine trace with algorithm M, for every pair $< n_i, n_j > \in Tr$, we check if $< n_i, n_j >$ is a member of $ht(Ns, M_h)$ and if $< n_i, n_j >$ is a member of $mt(Ns, M)$. We then perform a Chi-squared test of independence with a significance level $\alpha = .025$ using the 2×2 contingency table and check for rejection. The significance criterion of the Chi-squared test is set at.025 to reduce the probability of mismatching a human trace with a wrong machine trace. A rejection of the Chi-squared test's null hypothesis implies an association between comparisons made by the human sorter and comparisons made by the program. Once a test is rejected, we exclude comparison pairs that are not in the intersection of $ht(Ns, M_h)$ and $mt(Ns, M)$ from both traces and compute Spearman's rank correlation coefficient (Spearman, 1904). The intention of computing Spearman's rank correlation coefficient is to examine the ordering alignment of compared integer pairs.

Since performing comparisons in the reverse order of the machine trace does not correspond to a reverse execution of the algorithm, we confirm a match ($\alpha = .05$) between a human strategy and a machine algorithm only when Spearman's rank correlation coefficient is positive.

[3] We add one to all frequency counts to avoid including zeros in the table for the Chi-squared test.

Table 8 Mean strategy frequency in sort training and sort performance test

PS	Categories						
	BS	*DS*	*IS*	*MS*	*QS*	*Hybrid*	*Other*
Group 1 (MS/WEX)	–	–	–	–	–	–	–
Training	.012	.075	.150	.000	.175	.162	.425
Performance test	.056	.094	.162	.025	.238	.175	.250
Differences	.044	.019	.012	.025	**.063**	.013	− .175
Group 2 (MS/WOEX)	–	–	–	–	–	–	–
Training	.000	.062	.162	.025	.162	.225	.362
Performance test	.012	.038	.181	.100	.194	.181	.294
Differences	.012	− .024	.019	**.075**	.032	− .044	− .068
Group 3 (SM/WEX)	–	–	–	–	–	–	–
Training	.012	.050	.088	.038	.225	.175	.412
Performance test	.019	.138	.100	.025	.244	.119	.356
Differences	.007	**.088**	.012	− .013	.019	− .056	− .056
Group 4 (SM/WOEX)	–	–	–	–	–	–	–
Training	.000	.079	.184	.026	.158	.237	.316
Performance test	.013	.099	.243	.053	.158	.237	.197
Differences	.013	.020	**.059**	.027	.000	.000	− .119

Eachhuman sorting trace in the sort training and sort performance test was matched toone of the following categories of sorting algorithms, bubble sort (BS), dictionarysort (DS), insertion sort (IS), merge sort (MS), quick sort (QS), a combinationof insertion and dictionary sort (Hybrid) and unclassified (Other). The number ofhuman sorting traces in each category was averaged by the total number of traces.Wehighlight positive differences greater than .05 in grey and in bold font. In each group,the highlighted difference is the largest increase in algorithm application frequencyfrom sort training to sort performance test

Table 8 demonstrates the change in the distribution of strategy as the difference in mean strategy frequency from the training and performance test. In addition, Table 9 shows the strategy adaptation of participants. Participants were categorised with respect to whether they had applied a particular strategy in the training phase and in the performance test. We then employ McNemar's tests (approximated by binomial distribution for small frequencies) to determine if there is a difference in the number of participants who applied the strategy in the test and who used it in training. For strategies that have been identified to have increased applications from the training to performance test, we additionally examine the performance of these strategies using t-tests.

Group 1 (**MS/WEX**) and Group 3 (**SM/WEX**) both received explanations (**WEX**). According to Table 8, from training to performance test the average use of quick sort in Group 1 increased by .063 and the average application of dictionary sort in Group 3 improved by .088. McNemar's test ($\alpha = .05$) reports a significant change in the application of quick sort in Group 1 ($p = .031 < .05$). There is no significant difference in the application of dictionary sort in Group 3. Based on Table 9, in Group 1, we can observe that more participants used quick sort in the performance test (13) than in the training stage (7).

Further t-tests ($\alpha = .05$) on performance test scores showed that in Group 1 quick sort like approaches ($\mu = .834, \sigma = .274$) achieved a higher mean score ($p = .043 < .05$) than the rest of the strategies ($\mu = .729, \sigma = .337$) and in Group 3 responses corresponding to dictionary sort like approaches ($\mu = .885, \sigma = .288$) had a better performance ($p = .007 < .05$) than the other strategy categories ($\mu = .679, \sigma = .373$).

Table 9 The number of participants in Group 1 to 4 who had applied one of the highlighted algorithms (QS, MS, DS and IS) in the training and the performance test

Group 1 (MS/WEX)		Training		
Quick Sort		Application	No application	Total
Performance test	Application	7	6	13
	No application	0	7	7
	Total	7	13	20
Group 2 (MS/WOEX)		Training		
Merge Sort		Application	No application	Total
Performance test	Application	2	7	9
	No application	0	11	11
	Total	2	18	20
Group 3 (SM/WEX)		Training		
Dictionary Sort		Application	No application	Total
Performance test	Application	3	7	10
	No application	1	9	10
	Total	4	16	20
Group 4 (SM/WOEX)		Training		
Insertion Sort		Application	No application	Total
Performance test	Application	7	11	18
	No application	0	1	1
	Total	7	12	19

In Table 8, algorithms QS, MS, DS and IS have increased application frequency in Group 1, 2, 3 and 4 respectively. Every participant was categorised with respect to whether they had applied the strategy at least once in sort training and performance test. While apart from those four algorithms analyses were performed in each group with other strategies, the results presented were the most distinguished

Group 2 (**MS/WOEX**) and Group 4 (**SM/WOEX**) did not receive explanations (**WOEX**). From Table 8, in Group 2 the frequency of application of merge sort increased by.075 and in Group 4 the average use of insertion sort increased by.059. McNemar's tests show significant changes in the application of merge sort in Group 2 ($p = .016 < .05$) and insertion sort in Group 4 ($p = .001 < .05$). In Table 9, we can observe that in Group 2 more participants were using merge sort in test (9) than in training (2). In Group 4, almost all participants used insertion sort in the performance test (18) compared with in the training (7).

Additional t-tests ($\alpha = .05$) showed no performance difference in Group 2 between merge sort like approaches ($\mu = .872, \sigma = .209$) and the other strategies ($\mu = .827, \sigma = .288$). In contrast, insertion like approaches ($\mu = .877, \sigma = .285$) in Group 4 achieved a higher mean score ($p = .001 < .05$) than the other strategies ($\mu = .644, \sigma = .399$).

Table 10 **H1** to **H3** are hypotheses concerning the effects of curriculum order and explanations on human sorting comprehension

H		T
H1	Learning merging before learning sorting leads to a beneficial effect on human comprehension of sorting with respect to learning sorting before learning merging	C
H2	Learning merging with explanations results in a beneficial explanatory effect on human comprehension of sorting with respect to learning merging without explanations	N
H3	Learning merging with explanations further increases the beneficial effect of curriculum order on human comprehension of sorting with respect to learning merging without explanations	N
H4	Learning merging with explanations generated from rules without a low cognitive cost does not result in a beneficial explanatory effect on human comprehension of merging	C
H5	Learning merging with explanations before learning sorting leads to adaptation of efficient human sorting strategies with better performance	C

Hypotheses **H4** and **H5** relate to explanatory effects on human comprehension of merging and sorting strategy. At the top of table, H stands for hypothesis, T denotes test outcome. In the rightmost column, C is an abbreviation for confirmed, and N stands for not confirmed

5.5 Discussion

In Table 10, we present experimental hypotheses **H1** to **H5** and summarise their test outcomes. Since this work investigates the effect of curriculum order, we have acknowledged the possibility of recency and primacy effects on human sorting performance. Recency and primacy effects Murdock (1962) are cognitive biases which lead to better recall of items and concepts that are positioned towards the beginning and the end of a sequence. As it is part of our experimental plan to explore how human comprehension is affected by different sequential orders of concepts, it is difficult to completely eliminate the impact of position effects. In addition, Table 9 shows that various degrees of adaptations to efficient sorting algorithms happened in multiple groups. This indicates the adjustment of participants to different decision-making processes beyond making better or worse recall as a result of learning in sequential curricula.

Based on the definition of *sorter*/2 learned by $Metagol_O$ after learning *merger*/2 in Table 1, the program size is $u = 3$ and the number of predicates including those used in *merger*/2 is $p = 8$. For the definition of *sorter*/2 learned by $Metagol_O$ without learning *merger*/2 in the same table, the size of *sorter*/2 is $u + k = 5$ and the number of predicates used is $p + c = 6$. According to the Conjecture 3, $3 \cdot \ln(8) < 5 \cdot \ln(6)$ and learning merging before learning sorting results in a reduction in the size of the hypothesis space associated with the target hypothesis of sorting. The ANOVA test and Tukey's HSD test on sort performance test scores (Fig. 7b) demonstrated a beneficial effect from curriculum order. A higher sorting performance has been observed only in groups with the incremental curriculum setup. Since all four groups received the same material in each stage of the experiment so we can eliminate the possibility of learning merging or sorting being a major effect which led to the observed difference in sorting performance. Hence, we attribute the improvement in the performance score of sorting to the incremental curriculum order. This evidence confirms hypothesis **H1** and $E_{seq}(C_{MS/WEX}, C_{SM/WEX}, sorter) + E_{seq}(C_{MS/WOEX}, C_{SM/WOEX}, sorter) > 0$ which supports Conjecture 3.

Results on merge and sort performance tests did not show explanatory effects of the *merger*/2 rules learned by $Metagol_O$. The ANOVA test did not show a significant effect

Table 11 Summary of the significance of participant's strategy adaptation and performance improvement of the adapted algorithm compared with other algorithms

Group	Algorithm adapted	Is adaptation significant	Is performance improvement significant
Group 1 (**MS/WEX**)	QS	✓	✓
Group 2 (**MS/WOEX**)	MS	✓	*X*
Group 3 (**SM/WEX**)	DS	*X*	✓
Group 4 (**SM/WOEX**)	IS	✓	✓

A tick denotes a significant analysis result, and a cross indicates an insignificant analysis outcome

of explanations on human comprehension and there was no significant interaction effect on human comprehension. Therefore, we reject hypotheses **H2** and **H3** due to the lack of evidence. Learning merging with explanations did not improve or degrade human comprehension of sorting in regard to learning merging without explanations. In addition, the ANOVA test on the merge performance test scores did not demonstrate an effect of explanations on human comprehension. This result confirms **H4** and supports Conjecture 2.

Since merging was taught in isolation independent from other secondary or tertiary concepts, we refer to the cognitive window (Remark 1) for a plausible account of this lack of explanatory effect on human comprehension of merging. The cognitive cost of executing explained rules of merging is not sufficiently lower than the cognitive cost of operating a merging solution without the auxiliary information. Owing to limited cognitive capacities, a reduction in computational cost usually corresponds to an improvement in performance. However, solutions of merging after receiving explanations were not sufficiently optimised to yield an observably higher performance score compared with human primitive solutions in the absence of explanations. Given the tasks and the sequential teaching setup, an incremental curriculum order had a significant effect and improved human comprehension. The results provided a demonstration of the beneficial effect of sequential teaching on human comprehension.

Furthermore, we examined human comparisons and estimated sorting algorithms that best resembled human sorting traces. This analysis (Table 8) led to the recognition of four possible sorting strategy adaptations. From Table 9 and McNemar's tests, we show that there is a significant difference in the algorithm application of quick sort in Group 1, merge sort in Group 2 and insertion sort in Group 4. We argue that these results support a higher proportion of strategy adaptations to these algorithms. The additional t-tests compared the sorting performance of adapted strategies with other strategies. Quick sort in Group 1, dictionary sort in Group 3 and insertion sort in Group 4 had higher performance in comparison with other algorithms. Table 11 summarises the participants' adaptations to algorithms and the performance of responses with these adaptations.

Strategy adaptations in Group 1 (**MS/WEX**) and Group 4 (**SM/WOEX**) were significant and resulted in higher performance scores of the associated responses compared with the other strategies. After sort training, Group 1 used a higher volume of quick sort like strategies. The quick sort algorithm is computationally efficient with an average linearithmic run-time and a worst-case quadratic run-time. Utilising the efficiency benefit of concentrating on parts of the original problem one at a time, quick sort creates a pivot to construct two sorted sub-sequences. Based on the follow-up t-tests, adaptation to quick sort led to better comprehension compared with responses that used the other sorting strategies. Since in Group 1 participants learned merging with explanations before learning sorting,

this refutes the null hypothesis of **H5**. This positive difference in sorting performance from using quick sort in Group 1 allows us to confirm hypothesis **H5**.

The t-test results of Group 1 and 3 (summarised in Table 11) suggest that curriculum order does not affect the performance of adapted strategies since Group 1 experienced the incremental curriculum and Group 3 received the decremental curriculum. This observation contrasts with the overall performance analysis which shows curriculum order affects the performance of sorting. It implies that the decremental curriculum in Group 3 reduces the performance of strategies that have a lower change in application rate. We postulate that without the benefit of the incremental curriculum and a compact hypothesis space (Conjecture 3), it is difficult to develop rules that consistently represent the other algorithms (quick sort and merge sort). In addition, we observed in Group 1 that learning incrementally with explanations led to a significant adaptation of an efficient sorting strategy (quick sort) which also has enhanced performance. This shows the association of a higher task competency with the development of novel problem solutions in novices.

5.5.1 Impacts of explanations on performance

As summarised in Table 11, in Group 1 and 3 where explanations were presented, the adaptations to efficient sorting approaches correspond to improved performance. A significant number of participants adapted to quick sort like approaches and some participants adjusted to dictionary sort like strategies in Group 3. Although the dictionary sort algorithm shares many similar traits to insertion sort, dictionary sort is considered a divide-and-conquer algorithm which iteratively makes binary searches for the correct position of an object in a sequence.

We attribute this phenomenon to a better understanding of efficiently merging: two input sequences are sorted and redundant comparisons can be avoided by interleaving comparisons of fruits from input sequences. Provided explanations emphasised the optimal problem-solution structure and illustrated the action sequence by walking through examples with participants. The contexts provided by explanations and examples are effective for the human learning of abstract concepts (Aleven & Koedinger, 2002; Anderson et al., 1997; Berry & Broadbent, 1995; Reed & Bolstad, 1991). We suggest that in the presented teaching setup, explanations of merging involving examples grounded the concept of binary selection and contextualised the construction of problem solutions that utilised this information. As a result, problem solutions that incorporated this idea had less potential for errors and achieved higher performance. This proposition can be partially supported by the increase in the usage of insertion sort like strategies in Group 4 (**SM/WOEX**) where explanations were absent. While insertion sort like strategies correlated to better scores in sort performance test responses of Group 4, the insertion sort algorithm does not involve a divide-and-conquer or binary selection component to sorting and does not share these features with quick sort or dictionary sort.

5.5.2 Impacts of curriculum order on strategy adaptation

Another observation from Tables 9 and 11 is that a significant number of responses in Group 2 (**MS/WOEX**) adapted to merge sort. While this strategy adaptation of Group 2 to merge sort did not correlate with a higher performance score compared with the other strategies, divide-and-conquer algorithms such as quick sort and merge sort that require an

advanced understanding of computational algorithms could be rather difficult for novices to comprehend in a limited time frame. Based on Tables 9 and 11, it is noteworthy that a significant number of participants in Group 1 and 2 appeared to have developed sorting approaches similar to these well-known divide-and-conquer algorithms when they were presented with concepts with increasing complexity. A tentative account of this phenomenon can be referred to Conjecture 3. In Conjecture 3, two curricula are compared in terms of the size of the composite hypothesis spaces of learning some target predicates. Learning concepts in an incremental fashion reduces the size of the hypothesis space. Since humans have limited working memory capacity, it becomes easier for humans to find a consistent hypothesis that incorporates new information in the associated hypothesis space.

Despite performance improvements of adapted strategies in both Group 1 and 3, the adaptation of an efficient strategy was significant and led to better performance in only Group 1, as summarised in Table 11. Since Group 1 exclusively experienced incremental curriculum and machine-learned explanations, this result highlights the impact of combining both components on human learning for promoting task performance and developing novel problem solutions. Although it cannot be assessed by the present framework whether the participants mentally formulated quick sort or merge sort in an explicit way, further investigations of sequential teaching could devise machine learning to derive human strategy from behavioural traces.

6 Conclusions and further work

Previous publications (Ai et al., 2021; Muggleton et al., 2018) on the topic of comprehensibility have investigated the classification of explanatory machine learning into beneficial and harmful categories. The current work proposes an extension of frameworks of comprehensibility (Muggleton et al., 2018) and explanatory effects (Ai et al., 2021) to account for the effects of sequence teaching. Owing to the reduction in the size of the hypothesis space (Cropper, 2019), we hypothesise that learning concepts with increasing complexity enables humans to learn the target hypotheses given limited working memory capacity. This conjecture is supported by our empirical results.

In the limited scope of our experiment which focuses on human learning of sorting algorithms, we have identified an instance of sequential teaching curricula in which learning merging before learning sorting results in a better human comprehension of sorting in contrast to learning sorting before learning merging. This result demonstrates an improvement in human comprehension as a result of learning concepts with increasing complexity. We refer to the cognitive window (Ai et al., 2021) to account for the lack of explanatory effects on human comprehension of merging. It is difficult to improve human comprehension from explanations generated from machine-learned rules when the cognitive cost of the task is already low. In this case, explanations do not cause a sufficiently high improvement in the cognitive cost of performing the task. However, we have taken a rather conservative approach which has led to interesting preliminary results of sequential teaching in a specific domain. While this work is an extended investigation of machine learning comprehensibility based on translated logic programs, we also acknowledge the potential of developing comprehensibility definitions beyond symbolic machine learning and our noise-free framework for future work.

Instruction-based teaching approaches have the advantage of clarity and directness for human learning (Sweller et al., 2007) and allow previous knowledge to be integrated with

new problem-solving contexts (Mayer, 2004). However, such teaching approaches may impose an over-restriction on the links between actions and outcomes and therefore fail to trigger an individualised understanding generalised from the material (Bruner, 1961). In the human trial, we employed a mixed approach to teaching that combined instruction-based learning with discovery learning. We explored a machine-human teaching interaction where curricula allow a higher degree of freedom for learners to re-discover computationally efficient algorithms. We observed adaptations of human sorting strategies to utilise advanced techniques of divide-and-conquer. Such strategy adaptations were observed in humans who learned with explanations and in humans who learned concepts with incremental complexity. When both conditions were fulfilled, humans adapted to an efficient divide-and-conquer strategy with higher performance. We attribute these results to the contextualisation of abstract concepts by explanations and the accessibility of learned knowledge from the reduction in complexity in searching the hypothesis space. In these cases, human learners were able to adapt sorting strategies reminiscent of efficient machine sorting algorithms. Although the analysed participant sample had no background in programming, they adapted sorting strategies by employing divide-and-conquer techniques commonly found in efficient computational sorting algorithms. While not all inspected human learners managed to consistently perform these sorting algorithms after studying from short learning sessions, being able to develop "near-miss" versions of the merge sort algorithm is a remarkable achievement for those with no background in programming.

An exciting prospect of sequential teaching curricula is that machines and humans could take up more active roles to enable two-way cooperative learning (Sheridan, 2016). Sequential teaching curricula can be extended to represent the coordination between humans and machines in a back-and-forth fashion. Human implicit decision-making (System 1) could be made explicit (System 2) (Kahneman, 2011) to benefit human problem-solving (Schmid & Kitzelmann, 2011) by machine-learned theories via a process known as behavioural cloning (Bratko et al., 1995). Traces of human problem solutions can be provided as inputs to an ILP system, which might produce explicit algorithms from the traces to present to the human as explanations. A "clean up" effect on a human's behaviours can be achieved based on the error estimation of the machine-learned human algorithm in training. The debugging of behaviours guided by the prediction of error of a human strategy can be beneficial in intelligent tutoring (Zeller & Schmid, 2016) and for cooperative programming in algorithmic debugging (Shapiro, 1982).

Future work may also explore the phenomenon of human re-discovery of well-established domain knowledge. An investigation of partially guided development of efficient problem-solving strategies would require exposing the implicit cognitive processes. External tools such as eye-tracking devices (Bera et al., 2019) can be used to capture subtle motions of the human body to help us infer implicit human decision-making. Our observations have shown the creativeness of human learning in exploiting the analogical transfer of partial solution structure. For machine learners, insights obtained in other domains can be utilised to learn flexible and efficient problem solutions (Bratko, 2010; Leban et al., 2008). For human learners, the division of information into smaller "chunks" and the presentation of rule-based explanations facilitate the transfer of prior knowledge into a new problem-solving context (Anzai & Simon, 1979). The inclusion of both teaching techniques has the potential to trigger sudden realisations of previously incomprehensible concepts that might lead to innovative learning outcomes. In contrast, the absence of these teaching mechanisms might hinder human learning of complex tasks and lead to more laborious problem-solving strategies. However, further investigations are needed to gain a better

understanding of the degree to which sequential teaching curricula with machine-learned explanations facilitate novel human comprehension.

Author contributions LA wrote all paper sections except Sect. 5.1. JL wrote Sect. 5.1. JL, SHM, US provided feedback and corrections on all paper sections.

Funding This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No. 952215. The third author acknowledges support from the UK's EPSRC Human-Like Computing Network.

Data availibility Experimental material and data have been made available at https://github.com/LAi1997/sequential-teaching.

Code availability Source code has been made available at https://github.com/LAi1997/sequential-teaching..

Declarations

Conflict of interest The authors have no conflict of interest to disclose.

Ethics approval Ethics approval was not required. Included data and experiments were conducted by Imperial College London. Prior to carrying out the experiment, the authors had referred to the ethics approval guideline provided by Imperial College London (https://www.imperial.ac.uk/research-ethics-committee/ethics-approval-overview/). The authors had assessed that none of the five criteria applied to the planned experiment. Hence, the judgement was "No, we do not need to apply". The authors confirmed with the Research Governance and Integrity team at Imperial College London who accepted the authors' judgement.

Consent to participate Informed consent for participation in the experiment was obtained from all individual participants included in the study.

Consent for publication Informed consent for publication of data was obtained from all individual participants included in the study.

References

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access, 6*, 52138–52160.

Ai, L., Muggleton, S., Hocquette, C., Gromowski, M., & Schmid, U. (2021). Beneficial and harmful explanatory machine learning. *Machine Learning, 110*, 695–721.

Aleven, V., & Koedinger, K. R. (2002). An effective metacognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive science, 26*(2), 147–179.

Alfieri, L., Brooks, P., Aldrich, N., & Tenenbaum, H. (2011). Does discovery-based instruction enhance learning? *Journal of Educational Psychology, 103*, 1–18.

Amir, O., Doshi-Velez, F., & Sarne, D. (2019). Summarizing agent strategies. *Autonomous Agent Multi-Agent System, 33*, 628–644.

Anderson, J. R., Fincham, J. M., & Douglass, S. (1997). The role of examples and rules in the acquisition of a cognitive skill. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 23*(4), 932.

Anderson, J. R., Kushmerick, N., & Lebiere, C. (1993). *Rules of the Mind*, chapter The Tower of Hanoi and goal structures (pp. 121–142). L. Erlbaum.

Anderson, J. R., & Thompson, R. (1989). *Use of analogy in a production system architecture* (pp. 267–297). Cambridge University Press.

Anjomshoae, S., Najjar, A., Calvaresi, D. & Främling, K. (2019). Explainable agents and robots: Results from a systematic literature review. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems* (pp. 1078–1088).

Anzai, Y., & Simon, H. (1979). The theory of learning by doing. *Psychological Review, 86*(2), 124–140.

Arrieta, A., Díaz-Rodríguez, N., Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion, 58*, 82–115.

Arthur, W., Jr., & Day, D. V. (1994). Development of a short form for the raven advanced progressive matrices test. *Educational and Psychological measurement, 54*(2), 394–403.

Bain, M., & Muggleton, S. H. (1994). *Machine intelligence 13: Machine intelligence and inductive learning, chapter learning optimal chess strategies* (pp. 291–309). Oxford University Press Inc.

Barto, A. G., Sutton, R. S., & Watkins, C. (1989). *Learning and sequential decision making*. University of Massachusetts Amherst.

Bera, P., Soffer, P., & Parsons, J. (2019). Using eye tracking to expose cognitive processes in understanding conceptual models. *Management Information Systems Quarterly, 43*(4), 1105–1126.

Berry, D. C., & Broadbent, D. E. (1995). *Complex problem solving: The European perspective*, chapter Implicit learning in the control of complex systems (pp. 131–150). Lawrence Erlbaum Associates Inc.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1989). Learnability and the Vapnik–Chervonenkis dimension. *Journal of the ACM, 36*(4), 929–965.

Bratko, I. (2010). *Discovery Science. DS 2010. Lecture Notes in Computer Science*, volume 6332, chapter Discovery of Abstract Concepts by a Robot. Springer.

Bratko, I., Urbančič, T., & Sammut, C. (1995). Behavioural cloning: Phenomena, results and problems. *IFAC Proceedings Volumes, 28*(21), 143–149.

Bruner, J. S. (1961). The art of discovery. *Harvard Educational Review, 31*, 21–32.

Bruner, J. S., Goodnow, J. J., & Austin, G. A. (1956). *A study of thinking*. Wiley.

Burke, M., Penkov, S., & Ramamoorthy, S. (2019). From explanation to synthesis: Compositional program induction for learning from demonstration. In A. Bicchi, H. Kress-Gazit, and S. Hutchinson (Eds.), *Robotics: Science and system XV*.

Carbonell, J. (1985). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. *Machine Learning, 11*, 26.

Carpenter, P., Just, M., & Shell, P. (1990). What one intelligence test measures: A theoretical account of the processing in the raven progressive matrices test. *Psychological Review, 97*, 404–431.

Chi, M., & Ohlsson, S. (2005). *Complex declarative learning*. Cambridge University Press.

Chierchia, G., Fuhrmann, D., Knoll, L. J., Pi-Sunyer, B. P., Sakhardande, A. L., & Blakemore, S. (2019). The matrix reasoning item bank (mars-ib): Novel, open-access abstract reasoning items for adolescents and adults. *Royal Society Open Science, 6*, 190232.

Craig, R. C. (1956). Directed versus independent discovery of established relations. *Journal of Educational Psychology, 47*, 223–234.

Cropper, A. (2017). *Efficiently learning efficient programs*. PhD thesis, Imperial College London.

Cropper, A. (2019). Playgol: Learning programs through play. In *Proceedings of the twenty-eighth international joint conference on artificial intelligence* (pp. 6074–6080).

Cropper, A., & Muggleton, S. H. (2015). Learning efficient logical robot strategies involving composable objects. In *Proceedings of the 24th international conference on artificial intelligence* (pp. 3423–3429).

Cropper, A., & Muggleton, S. H. (2016). Metagol system. https://github.com/metagol/metagol.

Devidze, R., Mansouri, F., Haug, L., Chen, Y., & Singla, A. (2020). Understanding the power and limitations of teaching with imperfect knowledge. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence* (pp. 2647–2654).

Dienes, Z., & Perner, J. (1999). A theory of implicit and explicit knowledge. *Behavioral and Brain Sciences, 22*(5), 735–808.

Džeroski, S., De Raedt, L., & Driessens, K. (2001). Relational reinforcement learning. *Machine Learning, 43*, 7–52.

Freitas, A. A. (2014). Comprehensible classification models: A position paper. *SIGKDD Explorations Newsletter, 15*, 1–10.

Gennari, J. H., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence, 40*(1), 11–61.

Gentner, D., & Landers, R. (1982). Analogical reminding: A good match is hard to find. In *Proceedings of the international conference on systems, man and cybernetics*.

Gleicher, M. (2016). A framework for considering comprehensibility in modeling. *Big Data, 4*(2), 75–88.

Goldstine, H., & Neumann, J. (1963). Planning and coding of problems for an electronic computing instrument, part ii, volume 2. Volume V: Design of computers, theory of automata and numerical analysis. *John von Neumann collected works* (pp. 152–214). Pergamon Press.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Computing Surveys, 51*(5), 1–42.

Hind, M., Wei, D., Campbell, M., Codella, N., Dhurandhar, A., & Mojsilovic, A., et al. (2019). Ted: Teaching ai to explain its decisions. In *Proceedings of the 2019 AAAI/ACM conference on AI, ethics, and society*.

Hobbs, J. R. (2008). *The handbook of pragmatics, chapter Abduction in natural language understanding*. Peter Peregrinus.

Holyoak, K. J., & Koh, K. (1987). Surface and structural similarity in analogical transfer. *Memory & Cognition, 15*(4), 332–340.

Johnson-Laird, P. N. (1986). *Mental models: Towards a cognitive science of language, inference, and consciousness*. Harvard University Press.

Kahneman, D. (2011). *Thinking, fast and slow*. Macmillan.

King, R., Whelan, K., Jones, F., Reiser, P., Bryant, C., Muggleton, S., Kell, D., & Oliver, S. (2004). Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature, 427*, 247–252.

Kittel, J. E. (1957). An experimental study of the effect of external direction during learning on transfer and retention of principles. *Journal of Educational Psychology, 48*, 391–405.

Kolmogorov, A. N. (1963). On tables of random numbers. *Sankhya: The Indian Journal of Statistics, Series A, 207*(25), 369–375.

Leban, G., vZabkar, J., & Bratko, I. (2008). An experiment in robot discovery with ILP. In *Proceedings of the 18th international conference on inductive logic programming* (pp. 77–90). Springer.

Lee, M., & Thompson, A. (1997). Guided instruction in logo programming and the development of cognitive monitoring strategies among college students. *Journal of Educational Computing Research, 16*, 125–144.

Lemke, E., Klausmeier, H., & Harris, C. (1967). Relationship of selected cognitive abilities to concept attainment and information processing. *Journal of Educational Psychology, 58*, 27–35.

Lin, D., Dechter, E., Ellis, K., Tenenbaum, J., & Muggleton, S. H. (2014). Bias reformulation for one-shot function induction. In *Proceedings of the 23rd European conference on artificial intelligence* (pp. 525–530).

Lipton, Z. (2018). The mythos of model interpretability. *Communications of the ACM, 61*, 36–43.

Mansouri, F., Chen, Y., Vartanian, A., Zhu, X., & Singla, A. (2019). Preference-based batch and sequential teaching: Towards a unified view of models. In *Advances in neural information processing systems 32: Annual conference on neural information processing systems 2019* (pp. 9195–9205).

McAuley, J., & Leskovec, J. (2013). Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on recommender systems* (pp. 165–172). Association for Computing Machinery.

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? The case for guided methods of instruction. *The American Psychologist, 59*(1), 14–19.

Michalski, R. (1983). A theory and methodology of inductive learning. In *Machine learning*. Symbolic computation.

Michie, D. (1982). Experiments on the mechanization of game-learning: 2-rule-based learning and the human window. *The Computer Journal, 25*, 105–113.

Michie, D. (1988). Machine learning in the next five years. In *Proceedings of the third European working session on learning* (pp. 107–122). Pitman.

Michie, D. (1991). Machine intelligence and the human window. *Applied Artificial Intelligence, 5*(1), 1–10.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review, 63*, 81–97.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence, 267*, 1–38.

Minh, D., Wang, H., Li, Y., & Nguyen, T. N. (2021). Explainable artificial intelligence: A comprehensive review. *Artificial Intelligence Review*.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence, 18*, 203–226.

Muggleton, S., Lin, D., & Tamaddoni-Nezhad, A. (2015). Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. *Machine Learning, 100*, 49–73.

Muggleton, S., Schmid, U., Zeller, C., Tamaddoni-Nezhad, A., & Besold, T. (2018). Ultra-strong machine learning: Comprehensibility of programs learned with ILP. *Machine Learning, 107*, 1119–1140.

Muggleton, S. H. (1991). Inductive logic programming. *New Generation Computing, 8*, 295–318.

Murdock, B. B. (1962). The serial position effect of free recall. *Journal of Experimental Psychology, 64*(5), 482–488.

Newell, A. (1990). *Unified theories of cognition*. Harvard University Press.

Novick, L., & Holyoak, K. (1991). Mathematical problem solving by analogy. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 17*, 398–415.

Peirce, J. W., Gray, J. R., Simpson, S., MacAskill, M. R., Höchenberger, R., Sogo, H., Kastman, E., & Lindeløv, J. (2019). Psychopy2: Experiments in behavior made easy. *Behavior Research Methods*.

Poulin, B., Eisner, R., Szafron, D., Lu, P., Greiner, R., Wishart, D., Fyshe, A., Pearcy, B., Macdonell, C., & Anvik, J. (2006). Visual explanation of evidence with additive classifiers. In *Proceedings of the national conference on artificial intelligence* (Vol. 2).

Rafferty, A. N., Brunskill, E., Griffiths, T. L., & Shafto, P. (2016). Faster teaching via pomdp planning. *Cognitive Science, 40*, 1290–1332.

Reed, S. K., Ackinclose, C. C., & Voss, A. A. (1990). Selecting analogous problems: Similarity versus inclusiveness. *Memory & Cognition, 18*(1), 83–98.

Reed, S. K., & Bolstad, C. A. (1991). Use of examples and procedures in problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 17*(4), 753.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135–1144). Association for Computing Machinery.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence, 1*, 206–215.

Schmid, U. (2021). *Human-like machine intelligence*, chapter Interactive learning with mutual explanations in relational domains. Oxford University Press.

Schmid, U., & Finzel, B. (2020). Mutual explanations for cooperative decision making in medicine. *KI-Künstliche Intelligenz, 34*(2), 227–233.

Schmid, U., & Kitzelmann, E. (2011). Inductive rule learning on the knowledge level. *Cognitive Systems Research, 12*, 237–248.

Schmid, U., Zeller, C., Besold, T., Tamaddoni-Nezhad, A., & Muggleton, S. (2017). How does predicate invention affect human comprehensibility? In *Proceedings of the 26th international conference on inductive logic programming* (pp. 52–67).

Seger, C. A. (1994). Implicit learning. *Psychological Bulletin, 115*(2), 163.

Shapiro, E. Y. (1982). Algorithmic program debugging. ACM distinguished dissertation.

Sheridan, T. (2016). Human–robot interaction: Status and challenges. *Human Factors, 58*(4), 525–532.

Simon, H. A., & Hayes, J. R. (1976). The understanding process: Problem isomorphs. *Cognitive Psychology, 8*, 165–190.

Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, 15(1).

Sternberg, M. J. E., & Muggleton, S. H. (2003). Structure activity relationships (SAR) and pharmacophore discovery using inductive logic programming (ILP). *QSAR and Combinatorial Science, 22*(5), 527–532.

Stumpf, S., Bussone, A., & O'sullivan, D. (2016). Explanations considered harmful? User interactions with machine learning systems. In *Proceedings of the ACM SIGCHI conference on human factors in computing systems (CHI)*.

Sweller, J., Kirschner, P. A., & Clark, R. E. (2007). Why minimally guided teaching techniques do not work: A reply to commentaries. *Educational Psychologist, 42*(2), 115–121.

Tamagnini, P., Krause, J., Dasgupta, A., & Bertini, E. (2017). Interpreting black-box classifiers using instance-level visual explanations. In *Proceedings of the 2nd workshop on human-in-the-loop data analytics*. Association for Computing Machinery.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd international conference on machine learning* (pp. 2048–2057).

Yeo, T., Kamalaruban, P., Singla, A., Merchant, A., Asselborn, T., Faucon, L., Dillenbourg, P., & Cevher, V. (2019). Iterative classroom teaching. In *Proceedings of the thirty-third AAAI conference on artificial*

intelligence and thirty-first innovative applications of artificial intelligence conference and ninth AAAI symposium on educational advances in artificial intelligence.
Zambaldi, V. F., Raposo, D. C., Santoro, A., Bapst, V., Li, Y., & Babuschkin, I., et al. (2019). Deep reinforcement learning with relational inductive biases. In *ICLR*.
Zeller, C., & Schmid, U. (2016). Automatic generation of analogous problems to help resolving misconceptions in an intelligent tutor system for written subtraction. In *Workshops proceedings for the twenty-fourth international conference on case-based reasoning* (Vol. 1815, pp. 108–117).
Zhu, X. (2015). Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 29).

Authors and Affiliations

Lun Ai[1] · Johannes Langer[2] · Stephen H. Muggleton[1] · Ute Schmid[3]

✉ Lun Ai
lun.ai15@imperial.ac.uk

Johannes Langer
johannes-miran.langer@stud.uni-bamberg.de

Stephen H. Muggleton
s.muggleton@imperial.ac.uk

Ute Schmid
ute.schmid@uni-bamberg.de

[1] Department of Computing, Imperial College London, London, UK

[2] University of Bamberg, Bamberg, Germany

[3] Cognitive Systems Group, University of Bamberg, Bamberg, Germany