

Imperial College of Science, Technology and Medicine
Department of Computing

Effects of Machine-Learned Logic Theories on Human Comprehension in Machine-Human Teaching

Lun Ai

Doctor of Philosophy in Computing of the University of London and
the Diploma of Imperial College, April 8, 2024

Abstract

Explainable Artificial Intelligence (XAI) is an area of AI that aims to make AI systems human-understandable by explaining their functions, knowledge and actions. However, it is difficult to define and quantify how well people actually understand machine explanations. This thesis investigates an operational approach to define and analyse the effects of machine explanations on human comprehension. We explore AI-human interactions where people are taught by AI. An AI “teacher” presents machine explanations and human “students” are evaluated on how accurately they can apply the taught knowledge on new materials. Notably, we show that explanations provided by AI can quantifiably boost human task performance and lead to the re-discovery of efficient computational algorithms by non-experts.

We present an assessment framework E_{ex} to empirically quantify 1) both the beneficial and harmful effects of machine explanations and 2) the impacts of machine-human teaching curricula. This framework can evaluate machine explanations regardless of their representations. To demonstrate this framework, we examine explanations generated from logic theories learned via a Machine Learning approach called Inductive Logic Programming (ILP). These logic theories are computational logic statements learned by ILP systems from training examples. We describe a preliminary theoretical work that characterises the properties of logic theories and curricula for enhancing human task performance. We show that learning from a machine-learned logic theory that is informatively compact and less cognitively demanding improves human task performance. We also observe that teaching by building on simpler concepts benefits human task performance. When concepts with increasing complexity are explained, human novices can develop efficient computational algorithms.

Acknowledgements

I would like to thank Prof. Stephen Muggleton for his supervision during my doctoral research. His vision and attitude toward research and science have deeply impacted me in my pursuit of a doctoral degree in Computer Science. His continuous guidance has helped me stay on the course to explore a cross-disciplinary research area. It is my fortune to have Stephen as my PhD supervisor.

Additionally, this endeavour would not have been possible without the expertise of the Cognitive System group at the University of Bamberg. I was blessed to have the chance to learn from and work with Prof. Ute Schmid, Mark Gromowski, Johannes Langer and Carina Walker who were jointly responsible for designing and conducting experiments involving human participants. They brought crucial insights and feedback from Psychology during our collaborative projects.

What motivates me to explore and learn is the support of my beloved family members and friends. I would like to thank my entire family for their belief in me and the financial support of my study in the UK. I also want to thank Bridget Gundry, Alireza Tamaddoni-Nezhad, Wang-Zhou Dai, Celine Hocquette and Stassa Patsantzis for their help, encouragement and friendly company.

I am very grateful for this journey.

Declaration of originality

I declare that the content of this thesis is my own work. All the assistance received in preparing this thesis has been acknowledged. All other related work is appropriately referenced.

Copyright declaration

The copyright of this thesis rests with the author. Its contents are licensed under a Creative Commons Attribution 4.0 International Licence (CC BY-NC 4.0). Under this licence, researchers may copy and redistribute the material in any medium or format. Researchers may create and distribute adapted versions of the work. This is under the terms that appropriate credits are given and researchers do not use it or any derivatives for a commercial purpose. When reusing or sharing this work, the licence terms should be made clear to others by naming the licence and providing a link to the licence text. When a work has been modified, researchers should describe that the work has been changed and indicate those changes. Please request permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Contents

Abstract	i
Acknowledgements	i
Declarations	ii
List of abbreviations	xxi
1 Introduction	1
1.1 Motivation	1
1.2 Thesis statement	6
1.3 Contributions	7
1.4 Publications	8
1.5 Thesis outline	9
1.6 Summary	10
2 Related work	12
2.1 The role of explanations	12
2.2 Explainable artificial intelligence	15

2.2.1	Interpretability	16
2.2.2	Comprehensibility	17
2.3	Assessment of explanations in AI	19
2.4	Human-AI communication	20
2.4.1	Dialogue systems	21
2.4.2	Explanation-based machine learning	22
2.4.3	Argumentation-based machine learning	23
2.4.4	Machine teaching	23
2.4.5	Virtual convention modelling	24
2.4.6	Human-assisted computation	25
2.4.7	Behavioural cloning	26
2.4.8	Explainable AI for game playing	27
2.5	Computational complexity	28
2.6	Computational learning theory	29
2.7	Theory of tractable cognition	31
2.8	The human window	32
2.9	Logic programming background	36
2.9.1	Logic notations	36
2.9.2	Computation of logic programs	39
2.9.3	Inductive logic programming	40
2.9.4	Meta-interpretive learning	43
2.9.5	Metagol	47

2.10 Summary	51
3 Theoretical framework	52
3.1 Explanatory effect E_{ex} framework	52
3.1.1 Beneficial and harmful explanatory effects	53
3.1.2 Explanatory effect of a sequential teaching curriculum	57
3.2 Theory of a cognitive window	60
3.2.1 Cognitive cost of a logic program Cog	61
3.2.2 A cognitive window of human comprehension	65
3.2.3 Sequential teaching improvement	66
3.3 Summary	67
4 Beneficial and harmful explanatory machine learning	69
4.1 Introduction	69
4.2 Meta-interpretive learning of game strategy	71
4.3 Experimental framework	75
4.3.1 Quality of textual answers	75
4.3.2 Experimental hypotheses	76
4.4 Experiments	78
4.4.1 Materials	78
4.4.2 Methods and design	83
4.5 Results	86
4.6 Discussion	90

4.7	Summary	92
5	Explanatory machine learning for sequential human teaching	94
5.1	Introduction	94
5.2	Meta-interpretive learning of sorting algorithm	96
5.3	Experimental framework	100
5.3.1	Evaluating human comprehension of sorting	101
5.3.2	Experimental hypotheses	102
5.4	Experiments	106
5.4.1	MaRs-IB pre-test	106
5.4.2	Materials	108
5.4.3	Method and design	116
5.5	Results	118
5.6	Discussion	124
5.7	Summary	130
6	Conclusions and future work	131
6.1	Summary of contributions	133
6.2	Future Work	134
6.2.1	Evaluation of comprehensibility	135
6.2.2	Optimisation of comprehensibility	140
6.2.3	Applications of sequential teaching curricula	143
6.3	Summary of thesis achievements	144

List of tables

- 4.1 A set of win rules learned by *MIGO*. *MIGO*'s background knowledge contains a general move generator *move/2* and a won classifier *won/1* to encode the minimum rules of the game. The program is dyadic and *win_1/2* can be reduced to $win_1(A, B) : \neg move(A, B), won(B)$ by removing literals after unfolding. A more detailed description of the program learned by *MIGO* was given in [212] . 72
- 4.2 Logic program learned by *MIPlain*. It represents a strategy for the first player to win at different depths of the game. The predicate *win_3_4/1* can be reduced to $win_3_4(A) : \neg win_2(A, B)$. The program learned by *MIPlain* can be described as: a board A is won at depth 1 if there exists a move from A to B such that B is won; a board A is won at depth 2 if there exists a move from A to B such that X has exactly two pairs and O has no pairs in B; a board A is won at depth 3 if there exists a move from A to B such that X has exactly one pair in B and there exists a move from B to C such that X does not have any pair in C and C is won at depth 2 for X. 73
- 4.3 Comparison of program size (unfolded and without redundant literals) and cognitive cost. For winning in two (*win_2*) and three moves (*win_3*), the logic programs learned by *MIPlain* are smaller and have lower cognitive costs. . . . 73
- 4.4 Criteria for evaluating textual answers and examples for category *win_2/2*. . . . 76

4.5	Summary of experiment design. Participants played one mock game against a random computer player for the more difficult Island Game. After selecting a move in training and regardless of its correctness, participants received the labels of the two moves presented; treated participants additionally received explanations generated from <i>MIPlain</i> 's learned program. We introduced the primitive set used by <i>MIPlain</i>	83
4.6	The number and accuracy of HQ and LQ responses for groups MSR , MMR , SSR , SMR and each question category. For <i>win_3</i> , the most mentally challenging category of all three, no HQ response was given.	88
4.7	Hypotheses concerning the quality of textual answers and comprehension. S stands for supported, N denotes not supported, H stands for hypothesis. Test outcomes are presented for <i>win_1</i> , <i>win_2</i> and <i>win_3</i> categories.	91
4.8	Hypotheses concerning the cognitive window and explanatory effects. S stands for supported, H stands for hypothesis, T stands for test outcome.	91
5.1	Logic programs learned by <i>Metagolo</i> . When the robot performs <i>parse_exprs/2</i> , 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. <i>compare_nums/2</i> first selects a number from the left bag and a number from the right bag. Then it compares the two numbers. Afterwards, <i>compare_nums/2</i> 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 <i>drop_bag_remaining/2</i> to append the rest of the numbers to the last expression in the memory. <i>recycle_memory/2</i> takes all expressions in the memory and fills the expression list <i>exprs</i> . <i>single_expr/2</i> checks if there exists only one expression in the expression list <i>exprs</i>	97

5.2	Meta-rules used by <i>Metagol_O</i> . <i>Metagol_O</i> uses two meta-rules. 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.	98
5.3	We consider two independent experimental variables. 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.	103
5.4	Definitions of the rank functions. In curricula MS/WEX and MS/WOEX , humans learn merging before learning sorting. 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 <i>Metagol_O</i>	104
5.5	Design of the experimental groups. Group 1 learned merging before learning sorting and was provided with visual and textual explanations of merging based on machine-learned rules. 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.	116

5.6	Mean strategy frequency in sort training and sort performance test. Each human sorting trace in the sort training and sort performance test was matched to one of the following categories of sorting algorithms, bubble sort (<i>BS</i>), dictionary sort (<i>DS</i>), insertion sort (<i>IS</i>), merge sort (<i>MS</i>), quick sort (<i>QS</i>), a combination of insertion and dictionary sort (<i>Hybrid</i>) and unclassified (<i>Other</i>). The number of human sorting traces in each category was averaged by the total number of traces. We highlight positive differences greater than .05 in grey and in bold font. In each group, the highlighted difference is the largest increase in algorithm application frequency from sort training to sort performance test.	122
5.7	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. In Table 5.6, 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.	123
5.8	H1 to H3 are hypotheses concerning effects of curriculum order and explanations on human sorting comprehension. 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, S is an abbreviation for supported, and N stands for not supported.	125
5.9	Summary of the significance of participant's strategy adaptation and performance improvement of the adapted algorithm compared with other algorithms. A tick denotes a significant analysis result, and a cross indicates an insignificant analysis outcome.	127

List of figures

1.1	Explanations of machine-learned output that can make people more competent are beneficial.	2
1.2	Overview of the comprehensibility evaluation. Given a machine learner that takes training examples from an oracle, explanations of these examples are generated from the machine-learned output. A benchmark score is first established using a short intelligence test or some equivalent test before human subjects are taught to answer questions. Human subjects have access to the same set of training examples from the oracle and are presented with generated explanations according to these examples. Their comprehension of explanations is evaluated in performance tests based on their predictive accuracy of unseen test examples.	4
2.1	The set of functions describing possible cognitive capacities [300]. According to the P-Cognition thesis [8, 278], the set of functions describing possible cognitive capacities is a subset of polynomial-time computable functions.	31

- 2.2 The original illustration of Human Window from Donald Michie is based on an informal analysis of the computation and memory cost of four King and Pawn against King solution representations by Harris [24], Bramer [29], Beal [22] and Clarke [61]. The play of King and Pawn against King is one of the classic end-games of chess [30]. Bramer's representation uses "about 20 concepts (patterns)" and performs "little amount of search". Michie considered it the only human-intelligible and human-executable representation from the four analysed representations, assuming the brain's limitations on calculation and memory are around 20 binary discriminations per second and 10^{10} bits respectively [182]. 33
- 2.3 A plot of space-time compromise between hypothetical computational representations [299]. The x and y axes correspond to time and space complexity respectively. The origin represents the set of empty programs requiring no processing or memory resources. The frontier includes points 1 to 4. 34
- 2.4 Space-time trade-off curves for computing hypothetical functions f , g and h [185]. Underneath the axis of memory space, the left-most arrow (10^4 bits) and right-most arrow (10^{50} bits) are bounds on the length of minimal size programs for computing f , g and h . The rest of the arrows from left to right are the β' -complexity of f , the β' -complexity of g and the β' -complexity of h . Bounds of human evaluation denoted by a cross-hatched inner rectangle are hypothetically placed at 10^4 bits time-wise (β') and 10^{11} bits space-wise. Bounds of possible machine evaluations represented by a hatched outer rectangle are placed at 10^{10} bits time-wise (β) and 10^{15} bits space-wise. 35

4.1	An example of a Noughts and Crosses isomorphism called the Island Game. Every turn, each player occupies a cell with some resources (marked by the castle, fish, tree icons, etc). A player wins if he or she controls three cells on the same island or three pieces of the same resource. In our experiments, human participants, who play Blue, are presented with a game position and have to choose some alternative moves between the white cells. In Figure 4.1b, the cell with a yellow border highlight indicates a “bad” move that leads to a loss if the Orange player plays optimally. More details of the material design are given in Section 4.4.1.	70
4.2	Example of pairs. O has two pairs (green lines) and X has no pairs.	72
4.3	Letters P, Q, R, S, T, U, and V denote existentially quantified second-order variables and A, B, and C are universally quantified first-order variables. . . .	72
4.4	Example of pre- and post-training questions for the Island Game. A board is presented to the participant to select a move that he or she thinks is optimal. . .	79
4.5	Explanations generated based on logic programs learned by <i>MIPlain</i> . Both visual and textual explanations preserve the structure of hypotheses without redundancy and account for the reasons that make a move correct (highlighted in green). Explanations are presented for the possible sequence of wrong moves in participant’s turns (highlighted in red) by comparing against <i>MIPlain</i> ’s learned theory.	80
4.6	Generation of textual and visual explanations. Each Noughts and Crosses canonical state is converted into an Island Game state. A learned logic program is translated by using the Noughts and Crosses state and mapped natural language texts.	81
4.7	Visualisation of experiment design. Design (a) represents the experimental procedure for human self-learning where participants only learn from examples. Machine-aided learning corresponds to design (b) where participants learn from both examples and explanations.	84

- 4.8 Interface featuring a training example with explanations. Human participants, who play Blue, are confronted with a game position and have to choose between two alternative moves that are highlighted in yellow. When Blue owns two cells on the same island or two pieces of the same resource, related cells or resources are highlighted in bold. 85
- 4.9 The number of correct answers in pre- and post-training in different question categories. Bars show standard errors. We compare the post-training performance of learning from training examples (self-learning) and learning from machine explanations (machine-aided learning) via independent two-sample t-tests. In (a), machine-aided learning led to a higher average human performance in the *win_2* case and a worse average human performance in *win_3* case. In (b), self-learning results on the left have zero standard errors due to a sample size of two. There were no performance changes by machine-aided learning. 87
- 4.10 *win_3* reuses *win_2* and uses four *number_of_pairs/3* when unfolded. Figure 4.10a also shows that student groups could not provide a significant number of responses covering more than two of these predicates respectively. In Figure 4.10b, both mixed background groups (**MSR** and **MMR**) had lower proportions of responses covering one predicate than student groups (**SSR** and **SMR**). . . . 90
- 5.1 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. 95
- 5.2 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. 107

- 5.3 Participants are 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. Participants are taken to the next page where feedback and explanations are presented (Figure 5.4). 109
- 5.4 Explanations of merging is shown on this page. 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. . . . 110
- 5.5 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. 111
- 5.6 The balance on the interface (illustrated by Figure 5.3, 5.4 and 5.5) has two white boxes on two sides where fruit labels can be entered. It would tilt to the side of the fruit that is heavier similar to a physical balance. 113

5.7	Illustration of explanation generation. One set of integer sequences was sampled as training examples and for generating explanations. s_2 and s_3 are outcomes associated with a positive example and a negative example respectively. Given one input integer sequence s_1 and two example integer sequences (s_2 and s_3), logic program <i>merger/2</i> is executed on the input <i>Lt_expr</i> expression. Intermediate computation expressions are compared with two examples and mapped to equivalent Fruit Weighing representations as two piles of fruits. The error is recognised as s_3 appears inconsistent with the intermediate result which accounts for incorrect decision-making.	114
5.8	In setup (a), participants learned merging before learning sorting (MS). In setup (b), participants learned sorting before learning merging (SM).	116
5.9	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.	119

List of abbreviations

Argumentation-Based Machine Learning = ABML

Artificial Intelligence = AI

Amazon Mechanical Turk = AMT

Explanation-Based Learning = EBL

Inductive Logic Programming = ILP

Most General Unifier = MGU

Meta-Interpretive Learning = MIL

Machine Learning = ML

Natural Language Processing = NLP

Probably Approximately Correct Learning = PAC Learning

Sequential Teaching Curriculum = STC

Ultra-Strong Machine Learning = USML

Universal Turing Machine = UTM

Explainable Artificial Intelligence = XAI

Explanatory Effect Framework = E_{ex} Framework

Chapter 1

Introduction

1.1 Motivation

Artificial Intelligence (AI) is the science and engineering of intelligent computer programs [175]. AI is relevant to a wide variety of intellectual tasks [259]. In particular, Machine Learning (ML) focuses on developing computer programs whose performance of some tasks can be improved through experience [198]. Exciting results have been achieved by ML in many domains. For instance, machine learning systems have proven themselves capable of outperforming the best human players in highly complicated games [126, 277]. Machine-learned models [35, 304] have been demonstrated to be capable of processing complex natural language texts. Various clinical studies have shown that machine-learned models possess the potential to improve healthcare on various levels [79, 292]. Machine learning approaches have also become increasingly prevalent for supporting discoveries in scientific research and engineering [144, 316].

However, most machine learning systems are too opaque for humans to understand. The inability to transfer learned knowledge to humans has led to concerns about the risks of employing black-box machine learning [1, 111, 257]. For AI technologies to participate in today's society, their communication capabilities must be cautiously inspected [94]. Explainable Artificial Intelligence (XAI) [112] attempts to address these concerns by studying AI systems that enable human understanding and provide explanations of their outputs, structures, decisions and functions. Currently, many core definitions are ambiguous and not quantifiable [160].

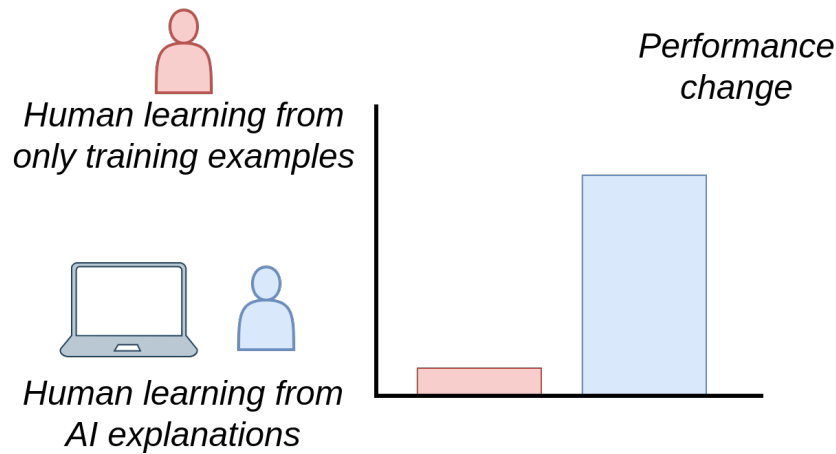


Figure 1.1: Explanations of machine-learned output that can make people more competent are beneficial.

Whereas standard evaluation metrics exist to evaluate the performance of a predictive model, no standardised evaluation strategy for machine explanations exists in XAI [222]. In particular, interpretations of human understanding of machine explanations are often based on subjective views [96, 191]. A critical question that needs to be answered is, *how can one quantitatively evaluate the human understanding of an AI system’s explanations?*

This thesis answers this question based on prior work in machine learning comprehensibility. As a notion key to AI [106], comprehensibility denotes a system’s ability to represent knowledge in a human-understandable format [15]. However, it is often associated with the complexity of machine-learned models due to quantification challenges of human comprehension [111]. *Ultra-strong* Machine Learning (USML) [186] by Donald Michie is a distinct notion that emphasises the machine-learned model to be symbolic and the ability of the system to teach humans to improve their performance. By relating human comprehension of symbolic representation to human out-of-sample predictive accuracy [268], USML was demonstrated, for the first time, using logic programs and Inductive Logic Programming methods [215]. Logic programs are sets of sentences in the language of Symbolic Logic [102]. Logic programs can implement logic mechanisms of human reasoning [121, 130, 152] and can represent knowledge structurally analogous to human concept hierarchy [14, 44, 223]. Inductive Logic Programming (ILP) [205] is a machine learning method that learns hypotheses expressed as logic programs from training examples. These hypotheses, called machine-learned logic theories, are often assumed to be human-understandable.

Importantly, we contribute by demonstrating that explanations generated from automatically induced logic theories can help people become measurably better [3] (Figure 1.1) and more innovative at performing tasks [2]. Domains such as manufacturing [220], education [55], healthcare [79] and science [144, 316] have increasing demands for AI that can help improve human competence in complex tasks. This thesis presents an empirical approach to evaluate machine explanations with the potential to facilitate high-performance human-AI collaborations. We identify and characterise some cases where machine explanations do not improve or even degrade human task performance [3, 2]. This thesis includes parts from two publications [3, 2].

We introduce a new framework E_{ex} [3, 2] to quantitatively assess the explanatory effects of machine explanations on human comprehension. Explanations by AI have diverse representations [192, 222]. Prior work on comprehensibility and USML in [215, 268] only account for logic programs. The E_{ex} framework does not have this limitation. To account for all representations, we categorise a machine explanation as the output produced by any machine. E_{ex} is representation-independent and is based on human predictive accuracy so it can be tailored for different tasks. In contrast to the comprehensibility definition in [268], our E_{ex} framework considers the harmfulness of machine explanations. It distinguishes human learning from machine explanations, where humans learn to predict the output of a function from explanations generated by a computer system, and human learning from training examples, where humans learn to predict the function’s output from input and output pairs of the function.

Additionally, we extend prior work [215, 268] by studying the effect of machine explanations on human comprehension in AI-human interactions. Human learning is considered incremental [37, 318, 103]. Humans can learn by building connections from a new context to prior knowledge [14, 87]. The ability to transfer and reshape past knowledge is considered a strength of human intelligence [231, 235], but explanations between humans and AI are often not interactive in XAI [191]. Motivated by the teaching aspect in USML, we model comprehensibility evaluations as *sequential teaching curricula* (STCs). In an STC, blocks of teaching materials such as explanations and domains of training examples are dependent on each other and are sequentially ordered. Research [216] shows humans tend to better reference concepts positioned towards the beginning and the end of a series. This is known as the serial-position effect. We pay particular attention to the order of concepts in a sequential teaching curriculum and consider a curriculum

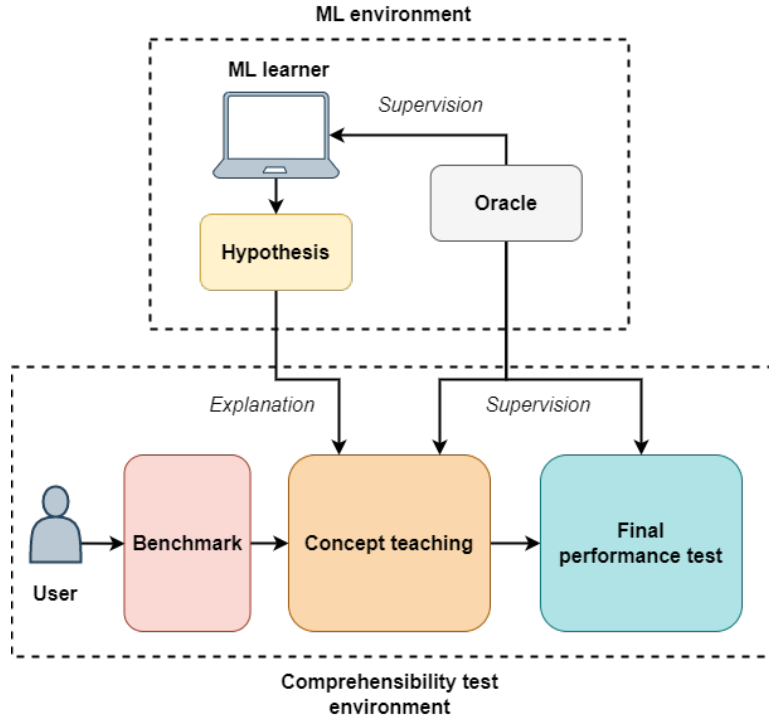


Figure 1.2: Overview of the comprehensibility evaluation. Given a machine learner that takes training examples from an oracle, explanations of these examples are generated from the machine-learned output. A benchmark score is first established using a short intelligence test or some equivalent test before human subjects are taught to answer questions. Human subjects have access to the same set of training examples from the oracle and are presented with generated explanations according to these examples. Their comprehension of explanations is evaluated in performance tests based on their predictive accuracy of unseen test examples.

incremental when concepts to be taught require solutions of increasing time complexity. E_{ex} offers new perspectives that advance prior work on USML and assessment of comprehensibility.

We have developed a preliminary theory on human comprehension of a logic program. We hypothesise two theoretical constraints of comprehending a logic program [3] based on the program’s 1) descriptive complexity and 2) execution stack requirements for knowledge application. Our theory is related to Donald Michie’s Human Window [182]. Michie speculated that computational representations best facilitate human understanding when they have an appropriate balance of descriptive size and executional cost. Representations that are descriptively complex or executional costly are unlikely to accommodate human memory or mental execution. For a range of machine representations, bounds on their descriptive and executional complexity suitable for human memory and computation represent a “window”. We postulate a cognitive window exists for a logic program: explaining a machine-learned logic program improves human comprehension if it can be cognitively executed and has low descriptive complexity.

The maintenance cost of storing sub-goals when executing a logic program is estimated by our proposed cognitive cost Cog . In addition, we hypothesise that a sequential teaching curriculum can enhance human comprehension of a concept if it teaches other suitable concepts that reduce the complexity of searching for a plausible solution. Our theory has been supported by our empirical results [2, 3]. Section 2.8 further explains Michie’s Human Window.

Recursion plays a key role in human understanding of concepts and semantics of language [116]. Efficient human operations often involve grouping goals and actions and reapplying them [14]. A variant of ILP, Meta-Interpretive Learning (MIL) [214], can be used to learn recursive logic programs, invent programs that represent new concepts [208] and reuse previously learned programs [158]. These traits make MIL an ideal machine learning approach to demonstrate the E_{ex} framework. To apply the E_{ex} framework, we have designed curricula to teach textual explanations and visualisations to humans (Figure 1.2). Each curriculum contains “comprehensibility tests” to examine human learners’ predictive accuracy after being exposed to teaching materials such as training examples or machine explanations. Human task performance aided by machine explanations is measured after a brief studying period. It is contrasted with a controlled curriculum where people study only training examples.

We perform empirical trials in the context of simple games and sorting. Both are fundamental topics in AI and Computer Science [65, 259]. Games provide dynamic environments where human inferential abilities can be analysed at different levels [228]. Developing efficient sorting approaches requires deliberate reasoning [311] and the ability of humans to combine previously learned concepts can be investigated in sorting. We purposely design experimental materials with elements from real-world settings to sufficiently challenge human learners. We demonstrate the E_{ex} to assess the effects of explaining logic programs on human comprehension in these complex situations. Logic programs are often assumed comprehensible while computationally complex or memory-demanding computational representations do not necessarily allow human comprehension [182]. Contrary to most XAI results which only highlight the positive effects [257, 287], we use the E_{ex} framework and cognitive window to study both the *beneficial* and *harmful* explanatory effects. In the context of computational algorithm teaching, we demonstrate human knowledge re-discovery - the recognition of known concepts through explorations - as human novices develop efficient sorting algorithms in sequential teaching curricula.

1.2 Thesis statement

We aim to answer how the explanations generated from machine-learned logic programs and the teaching order of concepts affect human task performance. Therefore, we examine the following thesis statement.

Thesis Changes in human comprehension are quantitatively identifiable and affected by both explanations of machine-learned hypotheses and the presentation order of target concepts.

We consider a *non-trivial task* as a specific objective that requires a sequence of actions to perform and would need considerable reasoning and training experience of a normal human novice to achieve high-level performance. Games and sorting are non-trivial tasks [228, 311], so we examine the thesis statement in these contexts. First, we focus on quantitatively measuring the beneficial and harmful effects of machine explanations on human comprehension.

Sub-thesis 1 For a non-trivial task, improvement and degradation in human comprehension can be quantitatively identified and are influenced by explanations of machine-learned hypotheses.

We examine **Sub-thesis 1** for a two-player game and sorting in Chapter 4 and 5. In addition, background knowledge and related experience often facilitate the human acquisition of new information [37, 14, 87]. Considering the impact of experience on human learning and the serial-position effect, we explore how human comprehension is affected by different sequential orders of machine explanations.

Sub-thesis 2 For a non-trivial task, improvement in human comprehension in sequential curricula can be quantitatively identified and is influenced by the presentation order of target concepts.

We assess **Sub-thesis 2** in Chapter 5 in the context of sorting.

1.3 Contributions

Contribution 1: Explanatory effects of machine-learned logic theories on human comprehension (supports Sub-thesis 1)

The definition of comprehensibility [268] is restricted to logic programs only. It makes no distinctions between human learning from training examples and human learning from machine explanations. In addition, to our knowledge, no prior empirical framework of comprehensibility and USML has accounted for the harmfulness of machine explanations. We extend on prior work [215] and create an explanatory effect framework E_{ex} which is flexible and representation-independent. We have developed a preliminary theory to account for the limitations of human reasoning and learning. We present an estimation of the goal maintenance cost Cog and introduce a hypothesis of a cognitive window on a logic program. In the context of learning a two-player game, our empirical analysis supports that a machine-learned theory would not have a beneficial effect on human comprehension when theory complexity is high or execution is cognitively challenging. This supports our hypothesis of a cognitive window. My contributions include:

- (Section 3.1.1) The first operational definition of beneficial/harmful explanatory effects of machine-learned theory on human comprehension as part of the E_{ex} framework.
- (Section 3.2.2) The definition of a cognitive window of a machine-learned theory that affects human comprehension based on the descriptive complexity and an estimated maintenance cost Cog of the explained logic theory.
- (Section 4.5) Empirical evidence of a beneficial effect of a machine-learned theory on human comprehension (**supports Sub-thesis 1**).
- (Section 4.5) Empirical demonstration of a harmful effect of a machine-learned theory on human comprehension (**supports Sub-thesis 1**).

Contribution 2: Effects of a sequential teaching curriculum on human comprehension (supports Sub-thesis 1 and 2)

The order of concepts in learning is important to both machine learning and human learning. Understanding the effects of curriculum order has been essential to machine learning theory and human problem-solving. However, this subject has not been examined in the context of comprehensibility and USML. We have extended the E_{ex} framework in [2] to account for the impacts of sequential teaching on comprehension. In our preliminary theory, we postulate an improvement of human comprehension by incremental sequential teaching. For learning computational algorithms, we demonstrate the improvement of human comprehension from sequential teaching as a result of the reduction in the size of the hypothesis space associated with the target concept definition. This supports our hypothesis on a sequential teaching curriculum improvement effect. My contributions are:

- (Section 3.1.2) The first operational definition of the beneficial/harmful effects of a sequential teaching curriculum on human comprehension as part of the E_{ex} framework.
- (Section 3.2.3) The definition of sequential teaching’s improvement of human comprehension based on a reduction in the size of the hypothesis space of the target concept by an incremental teaching order.
- (Section 5.5) Empirical results that indicate a beneficial effect of learning from sequential teaching curricula on human comprehension and support sequential teaching improvement (**supports Sub-thesis 2**).
- (Section 5.5) Empirical demonstration of the re-discovery of divide-and-conquer algorithms after human novices learn from concepts with increasing complexity.
- (Section 5.5) Empirical evidence of the optimisation of sorting strategy as a result of studying in an incremental curriculum with explanations generated from machine-learned logic rules (**supports Sub-thesis 1 and 2**).

1.4 Publications

Parts of two papers [2, 3] are included in this thesis. Both papers [2, 3] are published in the Machine Learning Journal.

My contributions to the published paper [3] include 1) developing the E_{ex} framework of explanatory effects, 2) defining a cognitive cost Cog of machine-learned logic theories, 3) hypothesising a cognitive window, 4) the modification of an ILP system to learn Noughts and Crosses winning strategy suitable for the empirical trials, 5) the implementation of the experiment interface with explanations generated from machine-learned logic theories, 6) analysis of empirical data collected from human participants and 7) writing of the majority of the paper.

In the published paper [2], my contributions are 1) an extension of the E_{ex} framework to account for sequential teaching curricula, 2) a hypothesis of sequential teaching curriculum improvement on human comprehension, 3) the implementation to learn logic theories for merging and sorting, 4) the implementation of the experiment interface with explanations generated from machine-learned logic theories, 5) analysis of empirical data collected from human participants and 6) writing of the majority of the paper.

1.5 Thesis outline

Chapter 2 discusses relevant research and important background to this thesis. In Chapter 3, we present the theoretical framework from both papers. In Chapter 4, we examine **Sub-thesis 1**. In Chapter 5, we assess **Sub-thesis 1 and 2**. Chapter 6 discusses future work. The remaining chapters of this thesis are summarised:

Chapter 2 Related work We review the role of explanations in Psychology. We summarise research in Explainable AI and the evaluation of explanations in AI. We describe topics in computational complexity and learning theory that are related to this thesis. We discuss human-AI communication including argumentation systems and language models. We also explain the theory of Tractable Cognition and Donald Michie’s work on the Human Window. We introduce the background of logic reasoning, ILP and MIL.

Chapter 3 Theoretical frameworks We describe our framework E_{ex} based on our papers [3, 2]. We define the explanatory effects of a machine-learned logic theory and sequential teaching curricula. We also define a cognitive cost Cog as an estimation

of the cognitive maintenance complexity. We present our hypotheses on a cognitive window and a sequential teaching improvement.

Chapter 4 Beneficial and harmful explanatory machine learning We include parts from our published paper [3]. We describe the empirical framework in terms of the design, materials and preparation of the experiments. In the context of game learning, the results of our experiments support our hypothesis of a cognitive window. We quantitatively demonstrate the beneficial (**supports Sub-thesis 1**) and harmful (**supports Sub-thesis 1**) explanatory effects of machine-learned logic theories on human comprehension using the E_{ex} framework.

Chapter 5 Explanatory machine learning for sequential human teaching Parts from the published paper [2] are included. We present the empirical framework covering the design, materials and preparation of the empirical trials that contrast curricula with different orders and materials. For teaching sorting algorithms, we show quantitative results that support the hypothesis of a sequential teaching improvement on human comprehension (**supports Sub-thesis 2**). Results also demonstrate that incremental learning in the presence of machine-learned explanations leads to adaptation of efficient sorting approaches with better performance (**supports Sub-thesis 1 and 2**).

Chapter 6 Conclusions and future work Remarks are made regarding the impacts of this thesis and limitations of the approach. We reiterate the contributions of this thesis. We discuss future work on applications of sequential teaching curricula for joint learning and discovery as well as extensions of E_{ex} and optimisations of comprehensibility.

1.6 Summary

This thesis targets the lack of empirical definition and understanding of human perspectives in XAI. We operationally define and analyse the comprehensibility of machine explanations. Our E_{ex} framework extends an existing line of work on comprehensibility but has broader ap-

plicability in XAI. We develop a preliminary theory on comprehensibility and it is supported by empirical results. We claim that *changes in human comprehension are quantitatively identifiable and affected by both explanations of machine-learned hypotheses and the presentation order of target concepts*. Notably, we show that learning with machine-learned explanations can improve human task performance but can also degrade human task performance. Then, we demonstrate that learning in incremental order results in higher human task performance and when explanations are additionally presented, newly developed efficient strategies can be effectively applied by human novices.

Chapter 2

Related work

We briefly review the literature on the role of explanations from Psychology (Section 2.1). Then, we summarise work in Explainable AI and distinguish interpretability and comprehensibility (Section 2.2). We review methods that empirically evaluate explanations in AI but focus on those that include humans (Section 2.3). We discuss various modes of human-AI communication including those enabled by argumentation systems and language models (Section 2.4). We touch on the potential of our E_{ex} framework as an objective assessment approach. Concepts in computational complexity theory and computational learning theory relevant to our theoretical framework and ILP are introduced (Section 2.5 and 2.6). We also review the theory of Tractable Cognition in Psychology which refers to computational complexity for modelling human cognition (Section 2.7). We explain Donald Michie’s work and related perspectives on the Human Window and their relations to this thesis (Section 2.8). At the end of this chapter, we introduce the background of Logic Programming, ILP and MIL (Section 2.9).

2.1 The role of explanations

Explanations facilitate learning [163] and allow humans to integrate new information into their existing knowledge [173]. Human learning is incremental [37, 318] and temporally correlated [85]. Human knowledge attainment is analogical to searching in a pool of solutions guided by some preference ordering and biases [37]. The working memory [190] is a hypothetical human

information processing device which has a limited capacity to hold information temporarily. The working memory capacity does not bound implicit cognitive processes that are outside of conscious awareness [228]. This limit of working memory is hypothesised to be around 7 ± 2 items [190]. The cognitive load [288] refers to the amount of working memory resources consumed, which directly affects the performance of human problem-solving [45]. Despite the limitation of working memory, humans can recognise and extract useful knowledge from prior experience [37, 14, 87]. The hierarchisation of relevant experience or goals, a technique called “chunking”, has been demonstrated as an efficient approach to transfer past knowledge to the current task [44, 223] and reduce the load on working memory [14]. Studies [67, 139, 151, 166] demonstrated that guided human learning has more potential for better learning outcomes than without guidance. Explanations should not be presented alone with actual problem-solving contexts. Results [9, 23, 4] show that human learners cannot always make full use of explanations when problem-solving contexts are missing. Human problem-solving ability can be improved by hands-on experience enabled by explicit guidance and worked examples [289]. The presentation of training examples along with explanations makes the specific task contexts available and has been demonstrated to benefit learning [250, 5].

The functionality of explanations is multifaceted and not limited to the transfer of knowledge [163]. Humans can explain actions in higher cognition tasks via verbalising the rules by which they apply to produce a given result [267]. The verbalisation of these rules makes them part of the declarative memory [10, 267] which is the intentional recollection of factual information and knowledge [296]. An explanation can be viewed as a communicative process [163, 191]. Research [310] shows that even engaging in an explanatory endeavour without arriving at a correct explanation can improve human learning. As a social process, an explanation involves two parties, the information transmitter that is responsible for explaining and the receiver of information [118]. In a sequence of explanations, a transmitter might require further information from the recipient to momentarily exchange its role with the recipient [191]. In social environments, people use explanations to manage social interaction, reconcile contradictions, change other’s beliefs and influence their actions [169]. People may believe explanations to be insensible when their beliefs, desires, and intentions are not in agreement. Interaction with the explainee can serve as a means to develop trust [169].

An explanation can also be considered as an inference for causes of an event [163]. An explanation as a product is an answer to a “why”, “how” or “what” question [232]. While causal explanations are not the only type of explanation [105, 146], people still seek associative information between observed events, unobserved events and situational changes. People sometimes do not explain the causes for an event directly, but explain relative to some other event that did not occur [191]. Explaining the cause of an event relative to some other events that did not occur is a contrastive explanation. It was argued that all “why” questions ask for contrastive explanations, even if the reference to the hypothetical event is not made explicit [164]. In the case of a discrepancy between observations and expected events [119], people tend to seek explanations [119]. The process of deriving the best explanation for an observed phenomenon is known as abduction [233]. Abductive reasoning often involves 1) obtaining unexpected observations, 2) generating potential hypotheses about these observations, 3) determining the plausibility of the hypotheses based on existing knowledge and 4) selecting the most plausible hypothesis as the explanation [191, 123]. Abductive reasoning is key to the cognitive process of explanations [114, 164, 251] and scientific discovery [237]. Research [118] also shows that many explanations are claim-backing. These explanations aim to justify why a statement or belief holds and why a decision is good but do not aim to explain the actual decision-making process [191]. Justifications play the role of cause reporting but also serve as an argument rather than just an answer to a question.

This distinction between “scientific” and “everyday” explanations is relevant in the context of AI [191]. Everyday explanations answer why particular facts (events, properties, decisions) occur [170]. Scientific explanations centre on causal relationships [237]. Explanations in AI are not all scientific since they do not account for general relationships or scientific laws [191]. These explanations may provide partial causal relationships, for instance, between decisions, events, trained models or applications [32]. Hence, the characterisation of explanations in AI is broad [32, 191]. A brief review of a research area that focuses exclusively on explanations from AI is given in the next section.

2.2 Explainable artificial intelligence

Attaining human-understandable knowledge representation has always been a central topic in AI [147]. Early studies on expert systems exploited the information obtained to allow human understanding. For example, the knowledge of chess can be represented in the form of decision trees [241] and simplified to accommodate the understanding of human experts [240]. Experts on infectious diseases can be advised by an interface built from decision rules [276]. These systems exploit the information obtained from expert knowledge to allow human understanding. While experts are often the target users of these systems, auxiliary information is often required by those who are not actively involved in maintaining these systems [60].

Explainable Artificial Intelligence (XAI) focuses on AI systems that enable human understanding and provide explanations of their structures, decisions and functions [112]. A diversity of motivations and techniques has emerged to explain the decision-making of machine learning systems [222]. The same key notions in different studies are frequently discordant and studies often vaguely describe core concepts without providing further specifications [160]. Without operational quantifications, results depend on researchers' subjective intuitions about what makes up a "good" explanation and accommodates human understanding [191, 96].

One criticism is that proclaimed results in many studies have little empirical evidence to support and human understanding of generated explanations has been neglected [1, 191]. Those studies that included empirical evaluations often do not provide a good account of the context, results and limitations [13]. Harmful effects of XAI such as potential misinterpretation and misuse of explanations have been discussed in [257, 287]. For example, partially transparent explanations may cause more confusion if key decision-making of systems is omitted [257]. Human decision-making can over-rely on the intelligent system regardless of the accuracy of information provided by the system [287, 137]. The valuable body of studies in Philosophy, Psychology and Cognitive Science has not yet been well exploited by XAI researchers [191]. For AI to play a bigger role in society, there has been increasing emphasis on the social perspective of AI [94, 191].

Owing to the inconsistent references to some of the key terms in XAI, we distinguish between interpretability and comprehensibility according to [15]. While interpretability usually refers to

a passive trait of a model for making itself transparent and sensible, comprehensibility describes a more active ability of a model to present its learned knowledge in human-understandable representations. This notion of comprehensibility is closely related to Donald Michie’s definition of Ultra-Strong Machine learning [186] which is distinguished by its trait to teach humans. In the following subsections, we provide an overview of research that studies these two topics.

2.2.1 Interpretability

Interpretability is a central concept of XAI and relates to the degree of clarity of information revealed to the user by a system [192]. Decision trees and rules are often assumed to be inherently interpretable [111]. A system based on a decision tree constructs a tree-like structure representing the decision-making process. Parts of the decision-making are localised at nodes of the tree. Users may check if certain conditions are satisfied at tree nodes and follow a branch of the tree to a specific outcome. The path from the root node to a leaf node can be linearised as a set of decision rules [241, 95]. A decision rule often consists of a set of if-then conditions. A decision rule maps an observation to a decision outcome via a sequence of condition checks [240]. Quantification of conditions can be explicitly represented as in M-of-N rules [218] where a decision outcome corresponds to the verification of M conditions from a set of N conditions. Rules can be extracted from opaque neural network models to enhance interpretability [12]. Owing to the efficiency of querying neural networks, rules can be learned from both training examples and the query results of a neural network model [68, 69]. For discrete attribute values, a neural network can be converted into a set of rules by first reconstructing the connection of hidden units in the form of a decision tree [270]. Compared with rules, a decision tree provides a graphical understanding of the decision hierarchy of a model. A decision rule is composable as partial knowledge which can be used to build explanations of local or global behaviours [96]. Hence, for an effective understanding of a set of decision rules, the ordering information of the rules is required. While decision trees and decision rules both enable human understanding [111], the conversion of a decision tree into a set of rules improves human understanding when rules are interpreted separately [241].

Another view of interpretability separates the process that generates explanations from the

decision-making process. Post-hoc interpretability originates from the interpretation of human behaviours that explanations can have the additional function of justifying decision-making and intuitions [118, 284]. The processes by which we make decisions and those by which we provide explanations are distinct [160]. Post-hoc explanations are usually processed from a machine learning system into, for example, natural language explanations, visualizations or case-based explanations. Textual and visual explanations provide a qualitative understanding of the relationship between the features of an input and a model’s output [32]. In the medical domain, given the image of a tissue sample, visual demonstration and textual information of the tumour tissue provided by a logic-based system help users to identify the size and the position of the tumour [266]. The authors in [174] have attempted to justify the rating decisions of a recommendation system by designing an additional model to provide a textual explanation of the top words of the topic associated with an item. Visualisation interfaces such as [290] explain to experts the rationale behind predictions of a binary classification model by exploring features relevant to predictions in a high-dimensional space. Local explanations aim to explain a particular prediction of a model by fitting a local and interpretable model around the prediction [252]. Case-based explanations in [47, 136] involve the proximity of instances and the learned representation is used as a distance metric to identify instances similar to the predictions being explained. In exchange for not sacrificing predictive performance, explanations are generated after the output has already been computed by these systems. Since most post-hoc interpretations require additional processing that could be misleading and unfaithful, they should be rigorously evaluated [160, 137].

2.2.2 Comprehensibility

Distinct from the notion of interpretability, comprehensibility emphasises the activeness of the explanations of a system to transfer its learned knowledge. The importance of comprehensibility has long been recognised in machine learning of human-oriented knowledge. Michalski [179] suggested that comprehensible machine learning should produce outputs in human-understandable “chunks” that share similar structure and semantics as those produced by human experts. The comprehensibility in AI has not been extensively studied and has usually been associated with

some complexity measure of the machine-learned model [111]. An increase in a model’s complexity is often thought to increase the computational effort of decision-making and reduce its comprehensibility. For a linear model, its complexity can be identified by the number of adjustable weights [57]. In addition, the number of representable distributions can be used to reflect the complexity of a parameterised model [38]. The descriptive complexity of a decision tree or a single rule can be measured by its depth [312] or width [157]. For decision rules, their complexity can be described by the average number of rule conditions that were evaluated [96]. To our knowledge, none of the existing comprehensibility metrics takes into account how computationally costly it is for humans to simulate a model given some domain instances. In contrast, we provide a novel framework in Chapter 3 that can assess human comprehension in sequential human-AI interactions based on human out-of-sample task performance. Via this framework, we enable operational assessment that can objectively quantify comprehensibility.

Research on the operational definition of comprehensibility can be traced back to the Weak, Strong and Ultra-Strong Machine Learning criteria [186]. Michie’s definitions aim to provide operational criteria for both predictive performance and knowledge comprehensibility of machine-learned models. The Weak Machine Learning criterion focuses on predictive measurements and the machine learner’s ability to improve its performance over increasing data. Today most machine learning systems are only considered weak machine learning according to Michie’s criterion. The Strong Machine Learning definition further requires learners to produce a symbolic representation of the learned output. As an extension of the Strong criterion, Ultra-Strong Machine Learning (USML) [186] emphasises a machine learner’s ability to teach its output to humans such that their performance over the task is improved substantially compared to their performance when learning alone. The notion of operational effectiveness is the core of Ultra-Strong Machine Learning and is closely related to the comprehensibility of machine learning. The process of machine teaching humans can be thought of as a form of knowledge sharing which requires knowledge gained by the machine learner to be human-comprehensible. Hence, the procedure of validating USML is equivalent to the assessment of the comprehensibility of a machine learning system.

2.3 Assessment of explanations in AI

While standard evaluations exist for assessing the performance of machine-learned models, there is no agreed-upon evaluation strategy for XAI [222]. Empirical and definitive tests are important for providing quantifiable insights from the perspective of the audience [160, 191]. Evaluations without human participants often examine to what degree certain explanation quality is satisfied [222]. For example, explanation qualities such as correctness [53, 281, 263] addresses the truthfulness of the explanation compared with the machine-learned model. Consistency [93, 253] checks whether identical inputs have identical explanations. Compactness [145, 110, 56] assesses the size and redundancy of explanations. Evaluating these qualities is helpful for researchers and developers to optimise explanations. In this thesis, we concentrate on empirical evaluations that include humans. We refer readers to [222] for a systematic review of XAI evaluations without human participants.

Quantitative evaluations that involve human participants remain to be a small proportion [222]. Common criteria include user satisfaction with explanations and the goodness of explanations evaluated by checklists or surveys [124]. Some studies have asked participants to provide understandability scores for machine-learned models. Human understanding of classification models was investigated in the context of data mining [6]. Two experiments were conducted involving human participants who were asked to rank machine classification models according to their understandability. An extensive empirical study [128] examined the understandability of decision trees and rule-based predictive models for classification. Yet, these results are not definitive due to the subjectivity of user interpretation.

Other methods involve some form of free-text responses accompanying question-answering tasks to elicit how people understand events, processes and systems [124]. The design of empirical studies often allows either think-a-loud problem-solving or retrospective explanation. Human understanding of decision rules was explored via qualitative and quantitative analyses [145]. In the proposed system, optimisation of rules takes into account example and class coverage, rule length, and the number of rules. Human subjects were asked to answer questions regarding the decision boundaries of rules produced by various rule-based systems. Participants were also requested to write descriptions based on the rules presented to them. The effectiveness of differ-

ent explanations on human understanding was explored in the context of decision trees [156]. Human participants were shown examples of a system’s operation along with various explanations. After a period of studying, participants answered test questions and were requested to predict either the input or output of the system. Participants were also asked to report textually and via Likert-scale questions regarding how well they understood the explanations.

Recent attention has been paid to USML and operational quantification of comprehension [268, 215]. The determinant of demonstrating USML is an operational notion of human comprehension of symbolic machine-learned output. The first operational definition of comprehensibility is given in [268] based on an assessment of out-of-sample human task performance.

Definition 2.1. (*Comprehensibility [268]*) *The comprehensibility of a definition (or program) D with respect to a human population H is the mean accuracy with which a human population H after a brief study and without further sight can use D to classify new material sampled randomly from the definition’s domain.*

The first machine learning system satisfying the USML criterion was demonstrated in [215] based on this definition of comprehensibility. The inspection time of the program, relatedness to known knowledge and the textual complexity of the program have been demonstrated to be highly influential factors in human comprehension. Results show an improvement in human out-of-sample predictive accuracy after learning a logic theory which indicates a beneficial effect on human skill acquisition. Schmid’s definition can be further clarified since it does not differentiate human learning from examples and human learning from machine-generated explanations. This notion of comprehensibility is limited to the presentation of machine-learned output in isolation despite the progressiveness of human learning. The harmful effects of machine explanations are also vital for our understanding of machine-human interactions but are not explicitly classified [257, 287, 137].

2.4 Human-AI communication

One of the earliest accounts of the role of AI in modelling human intelligence was given by Alan Turing in his description of the “child” machine [295]. It was argued that similar to how a child

is raised, an appropriate background and some sufficient “education” process that supports the iterative accumulation of knowledge, are essential for building an intelligent system. This view of AI systems as human-like learning devices was later broadened by others on the social impacts of AI. For example, Van Someren stressed the importance of inter-related problem-solving and communication components of AI [302] while Michie underlined the integrated role of AI to contribute to both the “stock of software” and the “stock of knowledge” [187]. Recently, attention has been paid to evaluation metrics for human-AI communications [124, 137]. However, very little work focuses on objective assessments of how humans perceive AI explanations [191]. We highlight research areas that we believe our E_{ex} framework in Chapter 3 can be relevant.

2.4.1 Dialogue systems

Early work on dialogue systems [48, 49] generates conversations based on a set of predefined decision rules and a corpus of expert explanations. Each decision rule describes a relation between topics and alternative rules represent different discourse strategies. For instance, a rule can represent a compare-contrast strategy of a salesman for describing the capability of a personal computer to a customer, which becomes applicable if the customer knows computers of a similar manufacturer. Structure-based dialogue rules may not adjust flexibly to the course of user dialogues. Changing the domain of dialogue also requires extensive revisions to handcraft new rules. Efforts were made to update user understandings of dialogues [50, 202] as interactive negotiations instead of “one-shot” question-answer pairs. User intentions are interpreted by modelling follow-up questions, which are treated as requests for information specifications. Such feedback allows misunderstandings to be corrected. However, the extent of novel information and application scenarios is still limited.

Recent development focuses on using machine learning techniques to build dialogue systems that can adjust according to new knowledge. This is one of the central topics in Natural Language Processing (NLP). Currently, most approaches utilise neural networks known as language models. These models rely on the quality and availability of textual data [320, 224]. Training schemes involving human subjects were incorporated to tune machine learning models

which helps generalise dialogues. Users can iteratively grade system responses [17] or actively ask questions [154]. While conversation samples and user feedback have been demonstrated to improve the agent’s dialogue ability [113], language models often do not employ feedback to the full extent since only limited response data are used after extensive pre-training [320]. Though large language models can engage in interactive conversations, their dialogues are often hallucinated [20] and unsupported [161]. Their hidden mechanistic structures do not help establish trust in users [168]. Most metrics either evaluate the goodness of dialogues on subjective satisfaction scores from users or “ground truths” input-output data [41]. In contrast, our E_{ex} framework offers to assess comprehensibility objectively and examine real interactions with human participants.

2.4.2 Explanation-based machine learning

Explanation-Based Learning (EBL) [199] encompasses methods in machine learning that exploit strong domain knowledge to construct a general explanation of the training examples. The literature on EBL refers to logic reasoning and human learning. Studies have been motivated by the ability of humans to generalise a concept from a few examples [90]. Based on background knowledge, early EBL techniques [199, 260] often provide a deductive proof of a training example with respect to the target concept. A set of first-order primitives is produced as an explanation of this specific domain instance. These primitives can be included as additional constraints [135] or further processed to produce recursive programs [140]. Though EBL systems have been designed to output explanations, limited empirical evidence has been provided regarding comprehensibility. While some EBL system supports user interactions, human understanding of explanations is often assumed without validation. No measure was provided to evaluate misinterpretation by users. Our E_{ex} framework can address this issue since it objectively quantifies human comprehension.

2.4.3 Argumentation-based machine learning

Compared with the aforementioned dialogue systems and EBL, Argumentation-Based Machine Learning (ABML) [203] focuses on persuasion and claim support [303]. Computational argumentation is supported by diverse theoretical and algorithmic foundations for analysing and implementing argumentation frameworks [81]. One subset of ABML tends to rely on argumentative techniques that directly model the relationship between a set of evidence for and against claims with quantitative descriptions of the strength of these arguments [46, 81]. The semantics of the argumentation framework depends on the quantification of the dialectical strength of arguments [248, 21]. In recommendation systems, ABML is often employed since argumentation frameworks can differentiate between competing recommendations to help resolve conflicting suggestions [34, 255]. Post-hoc ABML focuses on mapping non-argumentative models into an argumentation framework [81]. Argumentative rules can be extracted from sub-symbolic models as an approximated argumentation framework to explain their output [269, 84]. For logic programming, a sound and complete mapping can be achieved between logic programs and argumentation frameworks [256]. Arguments and dialectical relations can be used to describe a variety of data and knowledge formats in machine-learned models from different domains, allowing ABML to play an important role in ensuring the interpretability of AI systems [81]. For example, results [247] show users of recommendation systems have diverse demands and prefer interactive explanations. ABML has an advantage since argumentative explanations can usually be generated based on the user's interactive feedback [80, 247]. For ABML to demonstrate user comprehensibility, operational assessment measures and definitions are prerequisites. Our E_{ex} framework can allow analysis of user comprehension by quantitatively evaluating how accurately users can operate after studying argumentative explanations.

2.4.4 Machine teaching

Distinct to different parties engaging via dialogues and explanations, Machine Teaching [321] focuses on providing an optimal teaching set of training examples to allow maximum learning efficiency of a learner. Machine Teaching extends the theory of computational learning [108] and provides an algorithmic procedure for optimising the sample (teaching) complexity which

refers to the number of training examples needed for a learner to learn a concept [97, 127]. A machine teaching framework consists of a given model that represents the target knowledge of a concept to be taught and a model of the student with learning parameters. The student model is usually a hypothesised computational model of a human. The teaching complexity theory is exclusively based on computational learning theory often without much empirical support. In contrast, the incorporation of cognitive models in a teaching setting has gained increasing traction. In education, machine teaching frameworks have been applied to devise intelligent tutoring systems to select teaching examples [244, 313]. Sequential approaches [171, 308] to machine teaching have been explored because humans are often sensitive to ordered items. Sequential curricula additionally allow the teacher to gain more information over time to allow the teacher to adjust 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 [86]. There is a misalignment between theoretical models and applications which makes machine teaching difficult to apply in real-world teaching settings [86]. We provide a preliminary theory of cognitive window that bridges computational learning theory and empirical evaluations involving human participants based on our E_{ex} framework.

2.4.5 Virtual convention modelling

The aforementioned techniques require a human to adapt his or her behaviour to some communication protocol. This differs from common human-human interactions which are very flexible and instantaneous [194, 195]. In joint problem-solving, recent work in cognitive psychology describes the spontaneous development of implicit communication conventions. The most recognised approach is based on the Nash equilibrium [221], where each agent makes the best decision based on the actions of the other agent but is only concerned with its pay-offs. In contrast, Virtual Bargaining denotes the observation that two parties with the same or different goals can establish interdependent non-verbal protocols by collaborating [193]. It provides a cooperative account of the development of communication conventions alternative to the Nash equilibrium as agents might not only consider their pay-offs [54, 82, 262]. This model depicts

communications as “coordination games”, in which a sender knows the key information but is unable to act on it and a receiver can perform the necessary actions but requires additional information to decide amongst alternatives. Both agents “bargain” based on limited information exchanged via some signals, and they decide their actions by imagining themselves from the other’s perspective. Recent work has been done to simulate this kind of coordination by a logic-based system, in the context of a simple game of item selection and danger avoidance [40]. Such spontaneous “bargaining” is modelled by logical inference representing the joint reasoning of agents about game rules and individual game contexts. In particular, logical inference plays the role of adjusting faults in an established protocol, described as a logic program. Faults are classified as either made incorrect predictions or missed correct predictions. The ABC system [155] can repair these faults via the edition of axioms or conditions in the logic rules [66, 98], or the reformation of the syntax of the rules [39]. To date, implementations of Virtual Bargaining have not yet quantitatively examined how virtual conventions affect human performance. Our E_{ex} framework can be applied to test if explanations provided by a collaborative AI agent can help establish virtual conventions that improve human task performance.

2.4.6 Human-assisted computation

Human-Assisted Computation [272] refers to human-machine interactions where a machine performs certain functions by consulting humans. For example, in software development, when a program produces an unexpected outcome, a human programmer can become the “explainer” and provide information about the program’s functionality for corrections to be made to its computation. There is an increasing emphasis on AI that interacts to keep humans “in the loop” [131, 264, 191]. The objective is to harness the cognitive ability of humans to aid AI [219]. The increasing attention to machine learning and AI systems becoming cooperative has also encouraged studies to include humans to debug machine learning systems [59, 167]. Relevant to the scope of this thesis, Algorithmic Debugging [274] studies this approach in the context of logic programming. It involves an interactive cycle between a debugging algorithm for a program which has produced an erroneous output and a human who knows the design of the program and its functionality [43]. A computation tree that leads to the erroneous output is first con-

structured, followed by the human answers to a series of questions about nodes in the computation tree to locate the source of error. The debugging algorithm then provides a counter-example or candidates of a potential missing statement. This approach has been generalised to functional programming [226] and imperative programming [133]. Since our E_{ex} framework requires explanation to be operable by humans, the collaboration between humans and AI to jointly program complements our framework. In the case that explanations are human-intelligible but not human-executable, human-assisted computation can help construct human knowledge in machine representations. In Chapter 6, we discuss how we can use human-assisted computation to counteract the requirement of the E_{ex} framework on human executability.

2.4.7 Behavioural cloning

Behavioural Cloning [31] is the inverse of Human-Assisted Computation. It refers to the computational process of using machine learning to reconstruct implicit human knowledge that is easy to operate but difficult to describe. Skills required for operating in complex task domains are usually sub-cognitive and it is a challenging task even for expert operators to give a complete description. Piloting [189] and crane operation [297] are examples of such domains where behavioural cloning systems have been devised. Behavioural Cloning is implemented by rule-learning systems [188, 189, 297] due to the analogy in conceptual structure between goal-oriented knowledge and human control strategies [31]. To clone a trace of human behaviour, the system takes a sequence of human actions as training examples and produces rules that map the current state of the process to a set of actions for achieving a target goal. The benefit of cloning is a “clean-up” effect as cloned human experience and knowledge become more consistent and less error-prone due to human perceptual and executional inconsistency being averaged across the entire original behavioural trace [188]. Behavioural Cloning is relevant to our experimental analysis in Chapter 5 where we examine different human problem-solving solutions based on human action traces. While we have not yet learned from human traces, our analysis allows us to approximate the closest matches of human solutions to a predefined set of computational algorithms. In Chapter 6, we consider behavioural cloning of human solutions from empirically observed action traces to be an interesting extension of our work.

2.4.8 Explainable AI for game playing

In Chapter 3, we describe how our E_{ex} framework can be used to characterise the beneficiality and harmfulness of AI explanations on human performance of a two-player game, a Nought-and-Crosses isomorphism. Learning game strategies is an important topic in Machine Learning due to the richness of game environments for testing machine learning algorithms [309, 307]. The reinforcement learning system *MENACE* [180] was designed to learn an optimal strategy for Noughts and Crosses. Q-Learning [306] and Deep Reinforcement Learning [305] allow AI to compete in more challenging games such as Atari 2600 games [200] and the game of Go [277]. While these systems defeated the strongest human players, they cannot explain the encoded knowledge to humans.

Human-understandable game strategies can help humans get better at playing games and analysing the decisions of machine agents [238]. Early approaches to learning game strategies [273, 239] used the decision tree learner ID3 to classify minimax depth-of-win for positions in Chess end games. These approaches used carefully selected board attributes as features. The Chess experts had difficulty understanding the learned decision tree due to its high complexity [184]. Methods for simplifying decision trees without compromising their accuracy have been investigated [240] on the basis that simpler models are more comprehensible to humans. An early ILP system can learn optimal Chess endgame strategies at depth 0 or 1 [18]. An informal complexity constraint was applied which limits the number of clauses used in any predicate definition to 7 ± 2 clauses according to the hypothesised limit of human working memory [190]. Recent approaches such as [314] have aimed to explain the policies of reinforcement learning, but the learned strategy is implicitly encoded into the continuous parameters of the policy function which makes their operation opaque to humans. By augmenting training data with high-level annotated explanations, actions from learned policies can be mapped directly to explanations [120]. Relational Reinforcement Learning [89] and Deep Relational Reinforcement Learning [315] incorporate relational biases and logic reasoning over structured representations to ensure human understandability. In Chapter 4, we have chosen a simple two-player game domain to demonstrate our E_{ex} framework, but it can be generalised to assess human comprehension of more complex games.

2.5 Computational complexity

We introduce computational complexity theory relevant to our theory of cognitive window in Chapter 3 and empirical study on teaching computational algorithms in Chapter 4 and 5. We direct readers to [280] for a complete background on computational complexity. The Turing machine [293] is one of the earliest and most well-known models of computation. The Turing machine operates on infinite memory tape. A Turing machine can use its tape head to perform symbol reading, editing and manoeuvre of head position according to a system of machine instructions. An algorithm is a sequence of operations that manipulates symbols according to machine instructions [117]. In Turing’s model of computation, a function is computable by a finite sequence of operations if and only if it is computable by a Turing machine [294]. This model of computation is hypothetical as physical constraints limit the memory capacity of computers.

One focus of the computation complexity theory is to study the theoretical limitations of this computation mechanism [280]. Two common evaluation criteria of an algorithm are its space and time complexity. The time complexity accounts for the number of computation steps that an algorithm makes as a function of the input size. The space complexity measures the memory required by an algorithm as a function of the input size. Typically, an analysis of computational complexity follows an asymptotic approach [280]. For example, the quadratic function $f(n) = 2n^2 + 5n + 3$ is asymptotically bounded by $g(n) = 5n^2$, and thus $f(n) = O(n^2)$. An algorithm is efficient in time and space when the number of computation steps and the memory requirement do not drastically grow as the input size increases. Algorithms often possess distinguished properties which cause them to behave differently given particular inputs. A complexity analysis often focuses on the maximum time and space resources used in the worst-case scenarios. We implement sorting algorithms of different complexity to compare with human sorting strategies in Chapter 5.

One dimension to evaluate the complexity of input information is its descriptive cost [52]. The Kolmogorov complexity [142] of a string is defined as the length of the shortest program that outputs the string. For example, using natural language as a descriptive language, the string of characters “write XAI twice” has a shorter description “XAIXAI” whereas the string

“ARTIFICIAL” does not have a shorter description. However, no program can compute the Kolmogorov complexity for infinitely many strings [51]. This is addressed via time-bounded variants of the Kolmogorov complexity [7, 115] by finding the shortest description in a finite number of steps. It is important to note that compact representations are not always easy for humans to understand. From the example, we might find “write XAI twice” to be more sensible to us. This phenomenon has been accounted for in our cognitive window theory in Chapter 3 and experiment results in Chapter 4.

2.6 Computational learning theory

We summarise key concepts in computational learning theory relevant to our theoretical framework in Chapter 3 and our machine learning approaches in Chapter 4 and 5. Computational learning theory focuses on designing efficient machine learning algorithms and identifying the computational complexity of learning [134]. A (machine-learned) hypothesis is the output of a machine learning algorithm expressed in a language consisting of a set of symbols called the hypothesis language [26]. The hypothesis space contains all hypotheses that are constructible from the hypothesis language. The set of hypotheses consistent with training examples is called the version space [198]. A concept is a boolean function over a set of training examples and a set of concepts is a concept class. One way to perceive machine learning is to consider it as a search in the hypothesis space by the machine learner for a hypothesis that best describes the training examples [198].

The sample complexity of a learning algorithm is a criterion of learning complexity [198]. The sample complexity corresponds to the number of training examples required by a machine learner to converge with a high probability to a successful hypothesis [198]. Formally, given $\epsilon, \delta > 0$, the sample complexity n_{ex} of a machine learning system L for a concept class C is the minimum number of training samples needed to converge with high probability $1 - \delta$ to a hypothesis with error at most ϵ over concepts drawn from C . Gold’s work [107] was one of the earliest attempts to study sample complexity. In Gold’s model, a learning algorithm receives an infinite sequence of training examples, one example at a time. A class of languages is identifiable in the limit if the learner correctly identifies the target language in a finite amount of time after

receiving a certain number of examples. Gold’s model only considers the identification of the target language without errors, and it does not account for the learning resources required for convergence to the target language.

The theory of Probably Approximately Correct learning (PAC) [298] considers learning a partially correct hypothesis. The PAC learning model specifies that a learning algorithm PAC-learns a concept if and only if it learns a hypothesis with probability at least $(1 - \delta)$ and error at most ϵ where $\delta > 0$ and $\epsilon > 0$. PAC learning model differs from Gold’s model in that it provides a lower bound on the learning performance. However, PAC learning does not specify the number of examples needed for learning a hypothesis. Based on the PAC learning model, Blumer [28] demonstrated that an increase in the size of the hypothesis space corresponds to an increase in the number of training examples required by the learning algorithm to PAC-learn a hypothesis. Formally, for a concept associated with a hypothesis space $\mathcal{H}$, the number of examples n_{ex} required to PAC-learn a concept with error at most ϵ and with probability at least $1 - \delta$ is $n_{ex} \geq \frac{1}{\epsilon}(\ln|\mathcal{H}| + \ln\frac{1}{\delta} + \ln(c))$ (c is a constant). Blumer’s theory highlights the association between sample complexity and the size of the concerned hypothesis space. For two hypothesis spaces of different sizes that contain the target hypothesis, searching the smaller hypothesis space will lead to lower sample complexity and higher predictive accuracy with respect to searching the larger hypothesis space.

Additional information on hypotheses, such as domain-specific background knowledge or priors, is usually described as a bias as part of the hypothesis language [196]. Defining hypothesis biases can constrain the hypothesis space to include more correct hypotheses or exclude wrong hypotheses. Hypotheses that are structurally complex or involve many sub-concepts can be difficult to learn. This can be derived from the Blumer bound since hypotheses constructible from a compact language require fewer examples to learn since the associated hypothesis space is smaller [28]. These results are relevant for understanding the complexity of ILP and MIL algorithms. We also refer to the Blumer bound in our hypotheses of a cognitive window and sequential teaching curriculum in Chapter 3. In Chapter 4 and 5, we devise MIL systems to learn logic programs for game playing and sorting to examine these hypotheses.

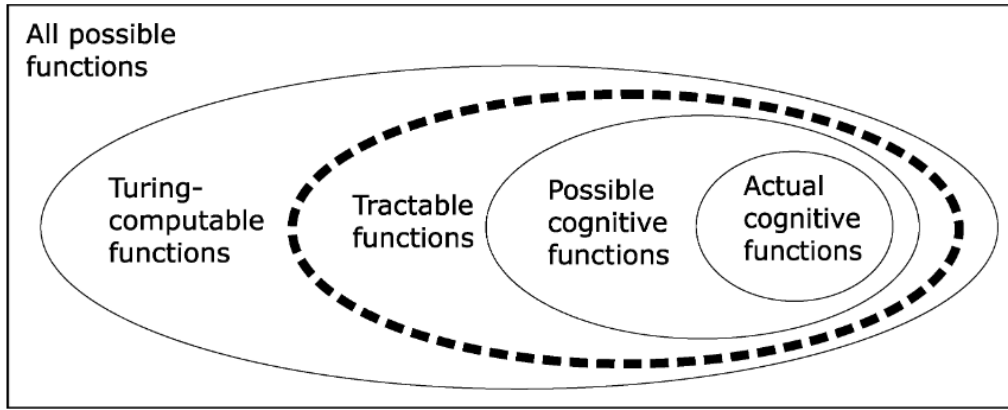


Figure 2.1: The set of functions describing possible cognitive capacities [300]. According to the P-Cognition thesis [8, 278], the set of functions describing possible cognitive capacities is a subset of polynomial-time computable functions.

2.7 Theory of tractable cognition

The idea that the information processing of humans can be modelled by the manipulation of symbols has a long history in both Psychology and Artificial Intelligence [129, 223]. We discuss the Tractable Cognition thesis as a reference from Psychology to the computational complexity theory for modelling human cognition. Many psychologists consider the theory of computational complexity and tractability constraints as a tool for refining models of human cognitive capacities [300, 78]. A common perspective is that cognitive capacity is the ability of the human mind to compute functions of some input and output states [8, 78, 172]. This view allows psychologists to characterise functional mappings between the hypothesised inputs and outputs of the human mind [300]. The P-Cognition thesis [8, 278, 153] describes that cognitive capacities are limited to computable functions in polynomial time ($O(n^\alpha)$ where α is a constant and n is the size of the input to the functions). The class of computable functions in polynomial time from computational complexity theory [99] provides a hypothesised formalisation of the set of functions that are human tractable (Figure 2.1).

A major objection of the Tractable Cognition thesis is that many human-operable tasks are not computable in polynomial time. A variant of the P-Cognition thesis considers Fixed-Parameter Tractability [300]. A function ψ_T is fixed-parameter tractable for a parameter k if there exists at least one algorithm that computes ψ_T in time $O(f(k)n^\alpha)$, where f is a function depending only on the parameter k , α is a constant and n is the size of the input to ψ_T .

Relaxing the P-Cognition thesis helps characterise computational theories of cognition that are not computable in polynomial time [301]. Psychologists [153, 228, 229] have recognised the exponential-time complexity of logic inferences and suggested revisions to computational theories of cognition to obtain realistic models of human reasoning. It is challenging for any computation-level theory to provide a tractable account of everyday human reasoning based on small-scale and artificial experimentation [228]. Boundaries of inference abilities need to be identified in real-world settings to reflect how humans deal with incomplete information and how they incorporate relevant background knowledge [42, 77]. Dynamic environments such as games that involve engaging subjects and real-world reasoning situations like discovery in science can provide the needed complication of empirical analysis [228]. To understand the limits of human understanding and examine our preliminary theory of a cognitive window, we have performed empirical trials in a two-player game (Chapter 4) and computational algorithms (Chapter 5). Both these domains are non-trivial and allow us to stretch human reasoning ability.

2.8 The human window

In Chapter 3, we present a preliminary theory of cognitive window on human comprehension of machine-learned logic programs. This theory is related to Donald Michie's Human Window [182, 187]. For every descriptive concept, there exists a spectrum of complete and correct computational representations which differ in computation and memory cost. Every space-time compromise by computational representations results in relative inefficiency in time, space or both [185]. The Human Window [182] (Figure 2.2) represents such a compromise between human executability and human intelligibility by trading off computation and memory cost for human understanding. Human-intelligible representations refer to the end of the spectrum of computational representation that requires less memory and the other end of the spectrum that demands less computation is human-executable [182]. Examples have been discussed in the domain of chess playing [299]. For instance, a chess move look-up table that allows a human to locate the optimal move with simple operations is human-executable. A program that traverses any possible chess game tree what is good for one player and what is bad for the opponent is human-intelligible. This representation is easy for a human to check for correctness but requires

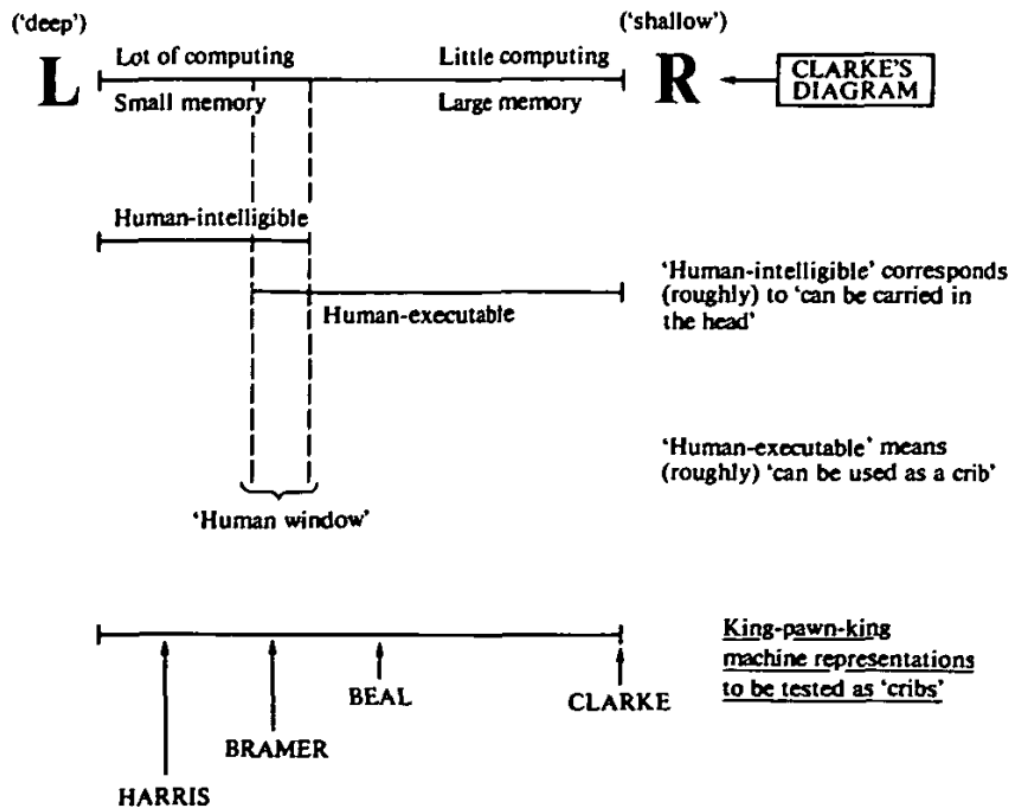


Figure 2.2: The original illustration of Human Window from Donald Michie is based on an informal analysis of the computation and memory cost of four King and Pawn against King solution representations by Harris [24], Bramer [29], Beal [22] and Clarke [61]. The play of King and Pawn against King is one of the classic end-games of chess [30]. Bramer’s representation uses “about 20 concepts (patterns)” and performs “little amount of search”. Michie considered it the only human-intelligible and human-executable representation from the four analysed representations, assuming the brain’s limitations on calculation and memory are around 20 binary discriminations per second and 10^{10} bits respectively [182].

examining moves and counter moves of many layers. Michie envisioned one primary task of AI is to determine whether a general Human Window exists and whether representations given by AI can be engineered to fit inside the Human Window [299].

An alternative account of the space-time trade-off was provided in [299] by extending to two coordinates, the number of processor computation cycles and the number of bytes required for storage. Hypothetically, mapping all computational representations of a concept onto the plot (Figure 2.3) should show a frontier that curves toward the origin [299]. The compact implementation can be made faster if more space is made available, for example, by storing the results of computations and returning the stored results when needing to make the same computations [181]. The data compression process can also make the computational representations more

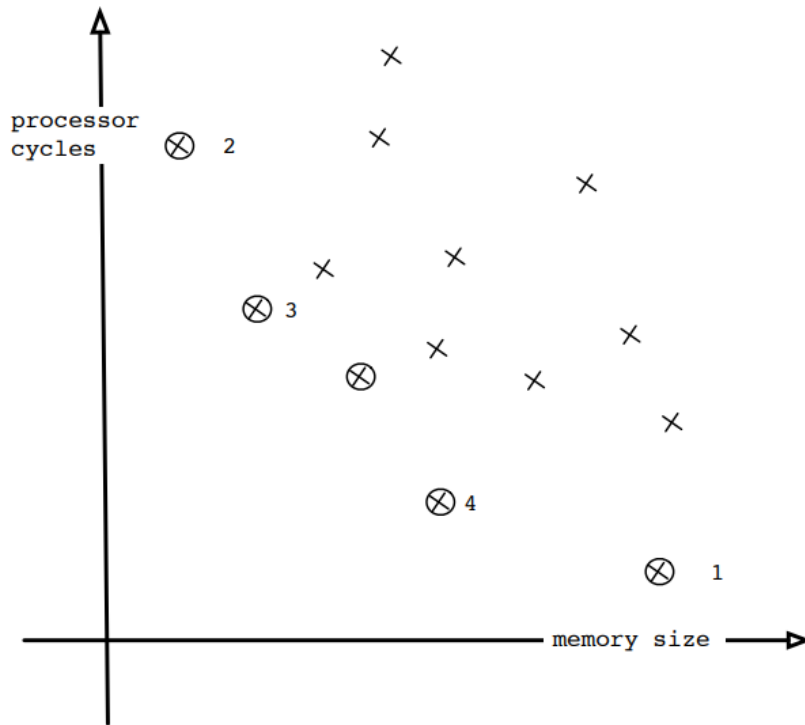


Figure 2.3: A plot of space-time compromise between hypothetical computational representations [299]. The x and y axes correspond to time and space complexity respectively. The origin represents the set of empty programs requiring no processing or memory resources. The frontier includes points 1 to 4.

compact at the expense of processor cycles for decoding. These optimisations are subject to diminishing returns, causing the frontier to curve toward the origin [299]. For a non-trivial concept, the points cannot be at the origin since computational representations of the concept cannot be evaluated with zero processor cycles and zero memory. The points also cannot get arbitrarily far away from the origin because of the limit on the amount of computation and memory that can be utilised [299]. The frontier highlights those representations that have relatively lower costs of these two resources.

A similar perspective of the Human Window was discussed by Michie to account for machine programs that are operational by humans as explanations [185]. When explanations are in the form of programs, the length in binary digits of a minimal program for computing a given function is a compact encoding but may not be desirable if the goal is for humans to simulate its computation. This gives rise to the β -complexity [183, 185], namely the length of an “operationally minimal program”. The β -complexity is the length of the shortest program able in the worst case to evaluate a given function within a β number of computation steps. For

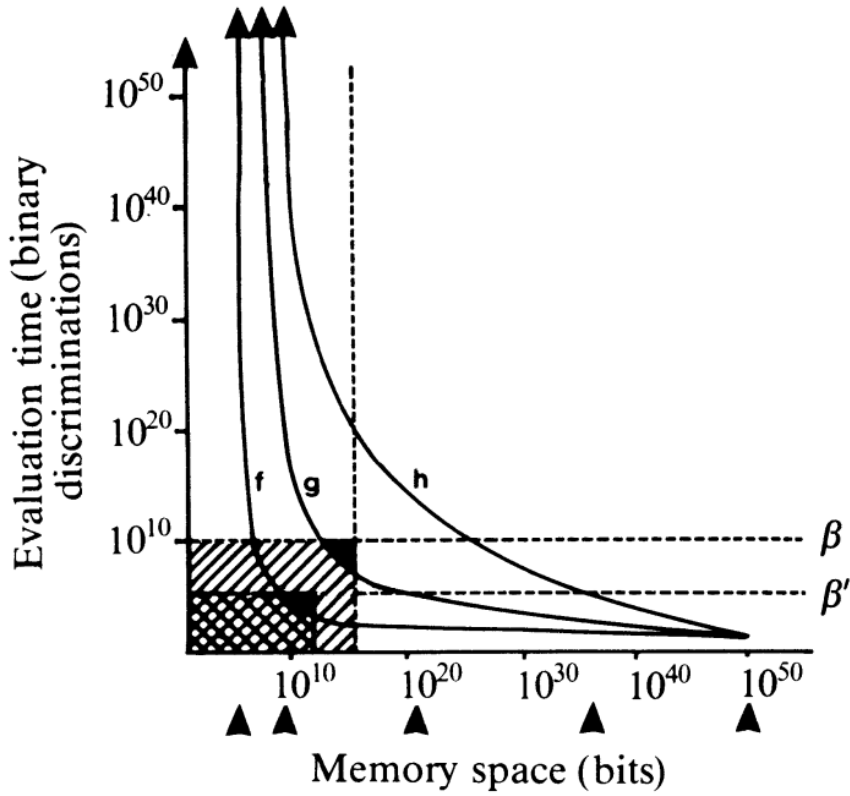


Figure 2.4: Space-time trade-off curves for computing hypothetical functions f , g and h [185]. Underneath the axis of memory space, the left-most arrow (10^4 bits) and right-most arrow (10^{50} bits) are bounds on the length of minimal size programs for computing f , g and h . The rest of the arrows from left to right are the β' -complexity of f , the β' -complexity of g and the β' -complexity of h . Bounds of human evaluation denoted by a cross-hatched inner rectangle are hypothetically placed at 10^4 bits time-wise (β') and 10^{11} bits space-wise. Bounds of possible machine evaluations represented by a hatched outer rectangle are placed at 10^{10} bits time-wise (β) and 10^{15} bits space-wise.

human execution, β can be replaced by a parameter specifying the rate of calculation possible by the human brain [185]. In Figure 2.4, the hatched rectangular zone passed by curves f and g are programs feasible for machines to compute time- and space-wise. Only f passes the inner cross-hatched area. This region depicts a hypothetical window consisting of human computation and memory bounds. In this hypothetical setting, programs in this region are considered feasible for humans to execute by Michie [185].

Both accounts of the space-time trade-off were only studied hypothetically. Empirical assessment of the Human Window is closely related to USML. For accurate communication between

humans and a machine, two questions are crucial based on [187]:

Can the user understand the representation provided by the machine?

Can the user simulate or validate the representation provided by the machine?

The first question asks whether learning the machine representation is possible for a specific person. An operational assessment needs to be performed to examine the second question. To demonstrate USML, human task performance over a task after learning from a machine learner is compared against task performance after learning alone. An improvement in task performance after learning from the machine learner indicates the effectiveness of machine representation for both human understanding and simulation. Descriptive and executional constraints have been hypothesised in our theory of a cognitive window in Chapter 3. We define the notion of explanatory effectiveness in Chapter 3 based on the definition of comprehensibility [268] that was previously used to demonstrate USML in [215]. In Chapter 4 and 5, we present empirical evidence supporting descriptive and executional constraints from our theory of a cognitive window.

2.9 Logic programming background

Logic reasoning has long been considered central to AI [295, 176] and human information processing [130, 152, 121]. We describe logic notations and the background of Logic Programming, ILP and MIL. For more details on Logic Programming and ILP, we refer readers to [162, 225].

2.9.1 Logic notations

A variable is a string made up of characters or digits, which starts with an uppercase letter. A variable is first-order if it can be substituted by a constant symbol or a first-order variable. If a variable can be substituted by a predicate symbol, it is a higher-order variable. The set of all variables is denoted by $\mathcal{V}$. A function symbol or a predicate symbol denotes a string of characters or digits, which starts with a lowercase letter. The arity of a function or predicate is the number of arguments it takes and these arguments form a tuple. A function or predicate

p with arity a can be represented by p/a . Monadic definitions of function and predicate have arity one, and dyadic definitions of function and predicate have arity two. A constant symbol is a function or a predicate symbol with arity zero. The set of all constants is denoted as the constant signature $\mathcal{C}$ and the set of all predicate symbols is represented by the predicate signature $\mathcal{P}$.

A first-order term is either a variable, a constant symbol, or a function symbol associated with a tuple of n terms. A ground term contains no variables. A term is higher-order if it is a predicate symbol or a higher-order variable. An atom is a predicate symbol applied to a tuple of n terms. An atom is ground when it contains only ground terms. An atom is first-order if it contains first-order terms and is higher-order if the terms contained are higher-order. A literal is an atom or the negation of an atom. A literal is positive if it contains an atom, and a literal is negative when it includes a negated atom. A literal is first-order if it contains a first-order atom and is higher-order when it includes a higher-order atom. The Herbrand Universe $\mathcal{U}$ denotes the set of all ground terms created from function and constant symbols. The Herbrand Base $\mathcal{B}$ represents the set of all ground atoms constructed from predicate signature $\mathcal{P}$ and the Herbrand Universe $\mathcal{U}$.

A clause or a rule is a finite disjunction of literals. A clause is ground when it contains only ground literals. A higher-order clause contains at least one higher-order literal. A Horn clause is a clause with at most one positive literal. Existential and universal quantifiers of a clause can be omitted for brevity. A Horn clause with a finite number of literals $A_0, A_1, \dots, A_n$ has the form:

$$\begin{aligned}
 &A_0 \vee \neg A_1 \vee \dots \vee \neg A_n \\
 &\quad \text{or} \\
 &\neg A_0 \vee \neg A_1 \vee \dots \vee \neg A_n
 \end{aligned}$$

where $n \geq 0$ and each A_i is an atom of the form $p(t_1, \dots, t_n)$ such that predicate symbol $p \in \mathcal{P} \cup \mathcal{V}$ and terms $t_i \in \mathcal{C} \cup \mathcal{P} \cup \mathcal{V}$. A goal is a Horn clause without any positive literal. A ground fact is a ground clause with only one literal. A definite clause is a Horn clause which contains exactly

one positive literal. A definite clause with a head literal A_0 and body literals $A_1, \dots, A_n$ has the form:

$$A_0 \leftarrow A_1, \dots, A_n$$

where $n \geq 0$ and each A_i is an atom of the form $p(t_1, \dots, t_n)$ such that predicate symbol $p \in \mathcal{P} \cup \mathcal{V}$ and terms $t_i \in \mathcal{C} \cup \mathcal{P} \cup \mathcal{V}$.

The substitution of variables $v_1, \dots, v_n$ in a logic formula by terms $t_1, \dots, t_n$ is defined as $\theta = \{v_1/t_1, \dots, v_n/t_n\}$. A unification $A\theta = B\theta$ for two formulae A and B uses θ as a unifier. A most general unifier (MGU) θ for two formulae A and B is a unifier for A and B such that for any unifier α for A and B , there exists a unifier β such that $\alpha = \theta\beta$. A clause C is said to θ -subsume a clause D if there exists a substitution θ such that $C\theta \subseteq D$ [236]. A clausal theory is a conjunction of clauses. A clausal theory containing only Horn clauses is called a logic program. If all predicates in a clausal theory have an arity at most one, the theory is monadic. When all predicates in a clausal theory have an arity at most two, the theory is dyadic. A logic program in which all clauses are definite is a definite (logic) program. The Datalog language is a subset of Prolog. A datalog program is a logic program with only Horn clauses. A datalog program contains no function symbols and disallows some of Prolog's operators such as the cut operator. Conventional datalog programs do not contain complex terms. A complex term is a predicate symbol applied to a set of complex terms [74]. We consider two MIL frameworks [212, 74] which use complex terms to represent static objects in a finite space. For instance, given logic terms $T_1, T_2, \dots, T_{n-1}, T_n$, a tuple of finite arity can be represented as $tuple(T_1, tuple(T_2, tuple(\dots, tuple(T_{n-1}, tuple(T_n, \epsilon))\dots)))$. $list(T_1, list(T_2, list(\dots, list(T_{n-1}, list(T_n, \epsilon))\dots)))$ represents a list of finite size where ϵ here denotes an empty character. These examples show mappings from Prolog lists and tuples of a finite length to terms with a finite number of arguments. This has been shown to reduce the textual complexity of programs learnable by MIL [74]. We use logic theory and logic program interchangeably in this thesis to refer to datalog programs.

2.9.2 Computation of logic programs

In this thesis, we apply our E_{ex} framework to examine human comprehension of machine explanations generated from machine-learned logic programs. We refer to the computation of Prolog and its implementation of logical resolution. Prolog's meta-interpreter generates all variable bindings as a logical consequence of the program. Every computation results in either a success or a failure. In the case that the execution outcome is a success, the result of a query is the substitution of some variables in the goal. The execution of a goal can be non-deterministic and may result in several substitutions. Non-deterministic results can be avoided in Prolog by restricting “looking back” at alternative substitutions and terminating computation when an execution path is found successful.

The computation of Prolog programs is implemented based on the SLD-resolution technique [143]. We describe SLD-resolution, SLD-Derivation, SLD-Refutation and some related concepts, based on [225].

Definition 2.2. (SLD-resolution): *Given a goal G_0 defined as $\leftarrow A_0, \dots, A_m, \dots, A_n$ and a Horn clause C defined as $B_0 \leftarrow B_1, \dots, B_k$ such that the literal A_m is unifiable with B_0 with a unifier θ , SLD-resolution of G_0 and C with substitution θ derives goal G_1 defined as $\leftarrow A_1, \dots, A_{m-1}, B_1, \dots, B_n, A_{m+1}, \dots, A_k$ where G_1 is called the SLD-resolvent.*

In Prolog, the leftmost unifiable literal in a goal is always chosen for derivation. When a goal is computed, the execution steps can be described by SLD-derivation which is a chain of SLD-resolution.

Definition 2.3. (SLD-derivation): *Given a definite program P and an initial goal G_0 , an SLD-derivation of a goal G_n is a possibly infinite sequence of goals $\{G_0, \dots, G_n\}$ such that every G_{i+1} is the SLD-resolvent of G_i and some clause in P .*

SLD-derivations of an empty clause as an end goal determine the set of substitutions needed to successfully resolve the initial goal and its sub-goals based on a given program.

Definition 2.4. (SLD-refutation): *For a definite program P and an initial goal G_0 , an SLD-refutation of G_0 is an SLD-derivation of the empty clause using P .*

Deducing a goal based on SLD-resolution for Horn clauses is refutation complete [225]. As mentioned previously, there could be several definite clauses in P that can be resolved with a goal. This might result in alternative substitutions. Therefore, the different paths of SLD-derivation from a goal to multiple sub-goals are defined as a tree structure.

Definition 2.5. (SLD-tree): *Given a definite program P and an initial goal G_0 , the SLD-tree of $P \cup \{G_0\}$ is a possibly infinite tree with root node G_0 , where each node is a goal and an edge denotes the SLD-derivation between two nodes.*

A successful branch of an SLD-tree corresponds to the derivation from the initial goal as the root node to the empty clause as the leaf node. Formally, for a definite program P , an initial goal G_0 , and the SLD-tree of $P \cup \{G_0\}$, a successful branch is a path between the root node G_0 and a leaf node with the empty clause. Given a sequence of MGU defined as $\theta_1, \dots, \theta_n$ for in computing a successful branch of $P \cup \{G_0\}$, a computed result θ for $P \cup \{G_0\}$ is the composition of $\theta_1, \dots, \theta_n$ for variables in G_0 . When the derivation does not end with the empty clause, there are no appropriate substitutions of variables to resolve the initial goal and sub-goals. Formally, for a definite program P , an initial goal G_0 , and the SLD-tree for $P \cup \{G_0\}$, a failure branch is a path between the root G_0 and a leaf containing a non-empty clause. In a derivation, individual goals must be resolved to compute the composition of substitutions. Hence, the complexity of a derivation is related to the resources needed in its resolution steps and the size of the SLD-tree. We refer to these definitions of SLD-resolution in Section 3.2.1.

2.9.3 Inductive logic programming

We give an overview of Inductive Logic Programming (ILP) and its advantages. This is followed by a summary of challenges faced by most ILP approaches which have led to our consideration for using MIL. An Inductive Logic Programming (ILP) [205] algorithm uses background knowledge B to induce a hypothesis in the form of a logic program which entails all of the positive examples E^+ and none of the negative examples E^- . ILP systems can be data-efficient and can derive theories from as few as a single example [158, 209, 207]. LP learns using BK represented as a logic program. Using logic programs to represent data allows ILP to learn with complex relational information. Compared with other machine learning approaches, ILP

systems can learn relational logic theories which are symbolic and can be understood by humans. This advantage makes ILP systems appropriate candidates for passing Michie's criterion of Ultra-Strong Machine Learning [186].

Definition 2.6. (*Correct hypothesis* [225]): *Given a background knowledge B , a set of positive examples E^+ and a set of negative examples E^- . A hypothesis H in the form of a logic program is correct if and only if H entails every example in E^+ (completeness) and none of the examples in E^- (consistency).*

Definition 2.7. (*ILP problem* [225]): *Given input pair (B, E) where B is a logic program representing the background knowledge, and positive and negative examples $E = E^+ \cup E^-$. The ILP problem is to induce a correct logic program H as a hypothesis which, together with the background knowledge B , entails all the positive examples in E^+ and none of the negative examples in E^- :*

$$\forall e^+ \in E^+ \quad B \cup H \models e^+$$

$$\forall e^- \in E^- \quad B \cup H \not\models e^-$$

Owing to different perspectives on the role of examples, several ILP settings have been defined. The learning from interpretations setting [83] considers examples as logical interpretations, and a hypothesis covers an example when the example is a model of the hypothesis. The learning from proof setting [230] views an example as a proof process, and a hypothesis covers an example if the example is a proof of the hypothesis. In this thesis, we adopt the setting of learning from entailment [210]. In the learning from entailment setting, training examples are clauses and an ILP algorithm should aim to derive a hypothesis that entails all positive examples and no negative examples. Since we place more emphasis on the errors of human learning, we use ILP in the learning from entailment setting and focus on the logic programs that are both complete and consistent with respect to the training examples. In the learning from entailment setting, given some background knowledge B , a hypothesis covers an example if the hypothesis entails the example. However, this specification of ILP can be overly strict in the presence of noise. When training examples become noisy, an ILP algorithm finds a hypothesis that covers a maximum number of positive examples and a minimum number of negative examples.

One challenge in ILP is choosing appropriate background knowledge. Background knowledge is traditionally hand-crafted by domain experts. This convention has been criticised due to the difficulty and cost of constructing appropriate background knowledge [92]. Predicate invention is the automatic introduction of new, possibly useful predicates during the process of learning from examples [204].

Definition 2.8. (*Invention* [70]): Given a background knowledge B , examples E and a hypothesis program H given by an ILP system, a predicate p is called an invention if and only if it is in the predicate signature of H but not in the predicate signature of $B \cup E$.

By inventing new predicates, the goal is to build relevant background knowledge without much human intervention. Invented predicates can reduce the size of a logic theory learned by an ILP system and improve its learning performance [72]. This process is analogous to the re-factoring of a program by a programmer for later re-use. Predicate invention has long been recognised as a key challenge in ILP [72, 208]. However, it is not supported in popular ILP systems such as *FOIL* [242], *Progol* [206] and *ALEPH* [285].

Another challenging problem in ILP is learning recursions [72]. A recursive rule in logic programming has the same predicate appearing both in a head and a body literal. Recursions allow for compact representations with high generalisation ability and high expressivity [72]. The learning of recursive programs is a difficult task within ILP [72] and is not fully supported in the older generation of ILP systems such as *Progol* [206], *Golem* [211], *FOIL* [242] and *TILDE* [27]. The interest in recursion has motivated Meta-Interpretive Learning (MIL)[158] which uses higher-order clauses called meta-rules. These meta-rules can be considered program templates that constrain the hypothesis format and the hypothesis space. Higher-order bias is a form of hypothesis space bias that encodes knowledge via higher-order logic. When used for learning logic programs from examples, higher-order clauses serve as both declarative bias and syntactical bias [62], carrying meanings as logic statements. Higher-order representations provide an abstract and flexible encoding of background knowledge, which has been shown to help reduce sample complexity and improve predictive performance [72]. We discuss in detail MIL and systems that implement MIL in the following section.

The introduction of MIL led to recent approaches of meta-level ILP which addresses the men-

tioned issues by generating meta-level logic program declarations [72]. Meta-level ILP techniques [149, 75, 132, 91, 73] are distinguished by their ability to utilise off-the-shelf systems and can easily learn recursive programs. By delegation of search to systems other than Prolog such as Answer Set Programming systems, some methods [64, 149, 132, 91, 73] demonstrated improvement in learning time and sample efficiency. Answer Set Programming (ASP) [33] is a logic programming paradigm based on the stable model semantics [101]. Answer set programs can be evaluated using dedicated ASP solvers such as Clingo [100]. Yet, encoding large meta-level programs usually results in large ASP specifications that are difficult to scale up [72]. Recent attempts [148, 73] focus on iterative revision of meta-level programs to reduce their complexity by introducing additional constraints to restrain subsequent searches.

The ability of ILP systems to represent learned knowledge in a rule-based format has been explored for communicating with humans. In [138], a fully automated system re-discovers the role of known function genes by integrating an ILP-based system into a laboratory robot. In [286], the ILP system assists a human chemist by producing hypotheses which describe the chemical connectivity of molecules and predict mutagenicity. However, owing to the complexity of encoded information, it remains a challenge in most systems to ensure human understanding. In [215], the textual complexity of the program is shown as an important factor affecting human comprehension. Thus, we consider predicate invention and recursion learning to be prerequisites of a machine learning system in our demonstration of E_{ex} . While we acknowledge that a range of ILP systems could be employed, optimisation for learning efficiency is not the primary goal of this thesis. We choose systems that implement MIL because they meet the requirements for learning recursion and performing prediction inventions. Being able to achieve both brings additional benefits in sample complexity.

2.9.4 Meta-interpretive learning

Meta-Interpretive Learning (MIL) [214, 213] is a variant of ILP which aims to address predicate invention [214], dependent learning [158] and learning of recursive and higher-order programs. Predicate invention aims to construct new predicate definitions. The output of a MIL system is logic programs containing a set of rules. A MIL learner typically implements the meta-

interpreter in Prolog. A MIL learner differs from Prolog's meta-interpreter in that Prolog's meta-interpreter proves by fetching first-order clauses that unify with the goals, whereas a MIL learner also fetches higher-order meta-rules while proving goals. A MIL learner uses examples as goals to initiate proofs and produces new predicate definitions based on meta-rules. These meta-rules are supplied to a MIL learner together with some first-order background knowledge.

Definition 2.9. (Meta-rule [214]). *Given a set of constant symbols $\mathcal{C}$ and a set of predicate symbols $\mathcal{P}$, a meta-rule is a higher-order formula of the form:*

$$\exists \pi \forall \mu A_0 \leftarrow A_1, \dots, A_m.$$

where $m \geq 0$ and π, μ are disjoint sets of variables and for $0 \leq i \leq m$, A_i is an atom $p_i(t_1^i, \dots, t_n^i)$ with some $n \geq 0$ such that $p_i \in \pi \cup \mu \cup \mathcal{P}$ and $t_k^i \in \pi \cup \mu \cup \mathcal{C}$.

Alternatively, a meta-rule definition may have the form $A_0 \leftarrow A_1, \dots, A_m$ by omitting the existential and universal quantifiers. In this thesis, we only consider learning without noise and a MIL setting with respect to learning from entailment, where the objective is to learn a correct hypothesis (Definition 2.6) which entails all the positive examples but no negative examples.

Definition 2.10. (MIL problem [122]). *Given an input tuple (B, E) where $B = B_c \cup M$ based on the background knowledge represented by a definite first-order logic program B_c and a set of second-order meta-rules M and the examples $E = E^+ \cup E^-$ is a set of ground atoms including positive examples E^+ and negative examples E^- . The MIL problem aims to induce a correct logic theory H which when combined with B , entails all the positive examples in E^+ and none of the negative examples in E^- :*

$$\forall e^+ \in E^+ \quad B \cup H \models e^+$$

$$\forall e^- \in E^- \quad B \cup H \not\models e^-$$

$$B \models H$$

where H is entailed by second-order meta-rules M and background knowledge B_c .

The background knowledge B contains a set of primitives which are predicate definitions. The

derived first-order theory $H \models M$ is a logical generalisation of the second-order meta-rules M due to the existentially quantified second-order variable in M . MIL systems mostly use a specific class of higher-order programs as meta-rules and learn hypotheses in this program class.

Definition 2.11. (*H_j^i program class [214]*): Given i, j are natural, the class H_j^i contains all higher-order datalog programs constructed from predicate signatures $\mathcal{P}$ and constant signature $\mathcal{C}$ with predicates of arity at most i and at most j atoms in the body of each clause.

H_j^i describes a general specification, while a MIL learner uses meta-rules of the dyadic fragment of H_j^i due to its decidable computation and universality. The programs learned by the MIL learner belong to the same dyadic fragment.

Definition 2.12. (*H_2^2 program class [214]*): Programs in H_2^2 the class are datalog programs constructed from predicate signatures $\mathcal{P}$ and constant signature $\mathcal{C}$ with predicates of arity at most 2 and at most 2 atoms in the body of each clause.

The dyadic logic program class supplemented with one function symbol is shown to have UTM expressivity [291]. There exists an encoding of a UTM that belongs to the H_2^2 fragment. The H_2^2 fragment also has UTM expressivity and is decidable given finite signatures of predicate symbols and constants[214]. Meta-rules in this fragment are used by a MIL learner to learn programs in the same class via a process called meta-substitution. A MIL learner, when attempting to find an appropriate meta-rule to prove goals, substitutes the higher-order variables in the meta-rule.

Definition 2.13. (*Meta-substitution [70]*): Given a meta-rule M , a meta-substitution in M is a higher-order substitution of the existentially quantified variables in M .

The binding of higher-order variables in a meta-rule to new predicate symbols results in an invented predicate definition. For a finite number of predicate symbols and meta-rules, up to a certain number of clauses, the associated Herbrand Base is finite. We refer to the decidability and sample complexity results of MIL under this condition [70, 158, 214].

Proposition 2.1. (*MIL decidability [214]*). The MIL problem is decidable if meta-rules M , background knowledge B_c , examples E^+ and E^- are datalog and the predicate signature $\mathcal{P}$ and the constant signature $\mathcal{C}$ are finite.

Proposition 2.1 [214] shows a query is always decidable since there is a finite number of programs in the hypothesis space. This corresponds to a bound on the size of the hypothesis space which can be generalised to the program class H_j^i (Definition 2.11).

Proposition 2.2. (*Size of the search space in H_j^i [70, 158]*): Given n_p predicate symbols and m meta-rules, and $i, j, n, n_p \in \mathbb{N}$, the number of H_j^i programs expressible with at most n clauses is $O((mn_p^{(j+1)})^n)$.

Based on the Blumer bound [28], we can compute the resource of training examples needed by a MIL learner to achieve a certain predictive accuracy.

Proposition 2.3. (*Sample Complexity in H_j^i [70]*): Given n_p predicate symbols, m meta-rules in H_j^i , and a clause bound n , the number of training examples s required to PAC-learn with error at most ϵ and probability at least $1 - \delta$ a hypothesis in H_j^i can be specified as:

$$s \geq \frac{1}{\epsilon} (n \ln(m) + (j+1)n \ln(n_p) + \ln(\frac{1}{\delta}) + \ln(c))$$

where c is a constant.

Proposition 2.3 [70] identifies the number of clauses, predicate symbols and meta-rules as key factors that affect the sample complexity of a MIL learner. To reduce the sample complexity, MIL uses dependent learning [214] which automatically constructs small programs that can be reused. This approach builds all predicate definitions which can be expressed up to a number of clauses given some bound. Compact predicates definition can be constructed by reusing previously invented predicates. This search continues until reaching the maximum clause size and the set of learned definitions is returned. This method has been extended in [71] to demonstrate that the sample complexity of a MIL learner can be improved by learning small programs without supervision based on instances sampled from the instances space. These small programs can be reused in inventing predicates analogical to a dependent learning approach.

Theorem 2.1. (*Sample complexity of dependent learning [71]*): Given n_p predicate symbols and m meta-rules in H_j^i , let n be the minimum number of clauses needed to express a target theory with standard MIL. Let $n - k$ be the minimum number of clauses needed to express

a target theory with dependent learning MIL using additional c predicate symbols. Let s and s' be the bounds on the number of training examples required to achieve an error less than ϵ with probability at least $1 - \delta$ with standard MIL and dependent learning MIL respectively. Then $s > s'$ when: $n \ln(n_p) > (n - k) \ln(n_p + c)$.

Theorem 2.1 implies that given two MIL learners supplied with the same set of training examples, the difference in sample complexity is counterbalanced by the error in prediction [71]. A more compact target hypothesis and an appropriate selection of background knowledge and meta-rules result in lower sample complexity. We refer to this result in our hypothesis of the effect of sequential teaching curricula in Chapter 3.

2.9.5 Metagol

We use the state-of-the-art MIL system *Metagol* [214, 75]¹ to learn datalog programs. *Metagol* is an implementation of the meta-interpreter in Prolog. Metagol supports predicate invention and dependent learning which reduces the size of the hypothesis space and improves learning performance [76]. It uses a technique called iterative deepening to traverse the hypothesis space by the number of clauses and increases it over iterations. This ensures the correct hypothesis has a minimal number of clauses and complex predicate definitions can be built from small

¹*Metagol* is still supported by the submission time of this thesis as the state-of-the-art MIL system.

programs. We refer to the code of Metagol [70] in Prolog to illustrate how it works.

learn(Pos, Neg, Prog) : –

prove(Pos, [], Prog),

not(prove(Neg, Prog, Prog)).

prove([], Prog, Prog).

prove([Atom|Atoms], Prog1, Prog2) : –

prove_aux(Atom, Prog1, Prog3),

prove(Atom, Prog3, Prog2).

prove_aux(Atom, Prog, Prog) : –

prim(Atom), !,

call(Atom).

prove_aux(Atom, Prog1, Prog2) : –

member(sub(Name, Subs), Prog1),

metarule(Name, Subs, (Atom : –Body)),

prove(Body, Prog1, Prog2).

prove_aux(Atom, Prog1, Prog2) : –

metarule(Name, Subs, (Atom : –Body)),

prove(Body, [sub(Name, Subs)|Prog1], Prog2).

Metagol takes sets of positive examples and attempts to prove these facts as goals. Metagol first proves by deduction using background knowledge by calling Prolog *call/1*. If this fails, Metagol tries to unify the goal with the *metarule/3* predicate which represents some meta-rule. It tries to bind variables based on the predicate and constant signature. Metagol then saves the grounded higher-order variables in a meta-rule as a meta-substitution and proceeds to prove the body literals of the meta-rule. If it creates a new predicate definition, a predicate symbol will be created. Metagol constructs a hypothesis by mapping the meta-substitutions onto their respective meta-rules when all proofs are completed. The resulting hypothesis is tested against negative examples. If the learned program covers any negative examples, the hypothesis is inconsistent. In this case, Metagol backtracks and tries alternative derivation paths. Iterative deepening ensures that it returns a hypothesis with a minimal number of clauses. Metagol gradually increases the size of the hypotheses constructed which expands the hypothesis space that it searches. Owing to the completeness of SLD-resolution (Definition 2.2), all hypotheses up to a certain size that cover the positive examples can be constructed. We illustrate how Metagol learns using the following example [70].

Example 2.1. (Metagol): Let the following set of facts be the background knowledge:

$mother(ann, amy). \quad mother(ann, andy).$
 $mother(amy, amelia). \quad mother(amy, bob).$
 $mother(linda, gavin). \quad father(steve, amy).$
 $father(steve, andy). \quad father(gavin, amelia).$

Let Metagol use two meta-rules, *ident* and *chain*, where P , Q , and R represent existentially quantified higher-order variables and A , B and C denote universally quantified first-order variables.

$metarule(ident, [P, Q], ([P, A, B] : -[Q, A, B])).$
 $metarule(chain, [P, Q, R], ([P, A, B] : -[Q, A, C], [R, C, B])).$

Given some ground facts, divided into a set of positive examples Pos and a set of negative examples Neg :

$Pos = [grandparent(ann, amelia), grandparent(steve, amelia),$
 $grandparent(steve, bob), grandparent(linda, amelia)],$
 $Neg = [grandparent(amy, amelia)].$

Metagol creates the following meta-substitutions and saves them in *Prog*. Each meta-substitution is identified by the bindings and the name of the meta-rule. A new predicate symbol is introduced which represents the invented clauses.

$sub(ident, grandparent_1, 2, [grandparent_1, father]),$
 $sub(ident, grandparent_1, 2, [grandparent_1, mother]),$
 $sub(chain, grandparent, 2, [grandparent, grandparent_1, grandparent_1])$

By mapping the higher-order bindings onto the meta-rule, *Metagol* outputs the program:

$$\text{grandparent}(A, B) : -\text{grandparent_1}(A, C), \text{grandparent_1}(C, B).$$

$$\text{grandparent_1}(A, B) : -\text{father}(A, B).$$

$$\text{grandparent_1}(A, B) : -\text{mother}(A, B).$$

MIL is capable of learning strategies of sequential tasks from solving the tasks themselves and transferring them for learning other tasks. The MIL learning framework *MIGO* [212] learns logic programs for playing Noughts and Crosses and Hexapawn. *MIGO* generates positive examples by playing against the optimal opponent and learning from them to revise its current strategy. It uses minimal background knowledge which contains only the game terminal conditions and the valid move generator. With slight changes to the background knowledge, *MIGO* can transfer a learned strategy to another game. Results demonstrate that transferring the learned strategy of one game to another game speeds up convergence [212]. In particular, *MIGO* was demonstrated to converge to high-performance strategies using less training data than Deep and classical Q-Learning for Noughts and Crosses and Hexapawn. In Chapter 4, we discuss how we modify *MIGO* for learning strategies of an isomorphic game of Noughts and Crosses that we invented.

A variant of *Metagol*, *Metagol_O* [74] makes use of composite objects and actions defined from primitive objects and actions. *Metagol_O* addresses the issue that most ILP systems only take into account the textual complexity of hypotheses. *Metagol_O* uses an extended iterative deepening technique to minimise both the textual and resource costs of hypothesis programs [74]. It computes the resource cost of hypotheses of increasing sizes according to a pre-defined cost function. *Metagol_O*, in comparison with other ILP systems which focus on classification problems, exploits higher-order bias for learning recursive programs as sorting strategies. In Chapter 5, we explain how we adapt *Metagol_O* to learn a variant of the merge sort algorithm.

2.10 Summary

The role of explanations is dynamic and multifaceted. This has motivated us to explore human-AI interactions via explanations. Representations of explanations in AI are also diverse. Without unambiguous definitions and operational empirical evaluations, it is challenging to account for human comprehension. We focus on a notion of comprehensibility that accounts for knowledge transfer through teaching explanations. This notion is closely connected to Ultra-Strong machine learning [186]. Most machine learning systems do not yet satisfy its criterion. In Chapter 3, a novel framework for assessing comprehensibility (E_{ex}) is presented in contrast to ambiguous core notions and subjective empirical interpretations in XAI. Results in areas of AI such as ABML and language models bring exciting perspectives to how humans can interact with AI. Various research areas on human-AI communications can potentially benefit from the operational assessment in the E_{ex} framework. We have briefly discussed this in this chapter. In Chapter 6, we will further explain how the E_{ex} framework can potentially be applied in these contexts.

Logic programs provide distinctive traits in both the representation of knowledge and the computational of logic processes. Specifically, MIL allows predicate invention and recursion learning to efficiently induce complex programs representing game strategies and computational algorithms. Hence, logic programming and MIL are suitable candidates to demonstrate the E_{ex} framework in non-trivial human reasoning tasks such as game playing and algorithm learning. We explain empirical studies in those domains in Chapter 4 and 5. Limitations of human information processing have been studied by both Psychologists and AI researchers. In particular, Donald Michie’s work on the Human Window [182] has inspired our preliminary theory of a cognitive window to model the descriptive and executional constraints on human comprehension of machine-learned logic programs. We present this theory in Chapter 3.

Chapter 3

Theoretical framework

In this chapter, we present our representation-independent E_{ex} framework [3, 2] for measuring the effects of machine explanations on human comprehension. E_{ex} accounts for human learning from only examples, human learning from machine explanations and sequential human learning which are all defined in this chapter. E_{ex} has wider applicability in XAI since there is no restriction to logic programs. Inspired by Michie’s Human Window [182], our preliminary theory of a cognitive window explores the limitations of human learning and execution. Since we focus on logic programs, only a cognitive window of logic programs has been explored. We postulate the existence of a cognitive window: explaining a machine-learned logic program improves human comprehension when learning the program does not require searching a large hypothesis space and the cognitive complexity of computing it is low. To estimate the cognitive complexity, we introduce a cognitive cost for approximating the load on working memory during the human execution of logic programs. We formally define a sequential teaching curriculum and hypothesise that a sequential teaching curriculum improves human comprehension if it introduces concepts that reduce the size of the hypothesis space for learning the concepts.

3.1 Explanatory effect E_{ex} framework

The E_{ex} framework defines and assesses the effect of machine explanations on human comprehension which we call the explanatory effect. It measures how well humans can understand and

apply the knowledge they learn from machine explanations. In the only operational definition of comprehensibility before our work (Definition 2.1), no clear distinction was made between human learning from examples and human learning from machine explanations. While examples contribute to the contextualisation of problem-solving, they cannot explicitly represent the goal hierarchy. This distinction is made clear in E_{ex} since learning from explanations in contrast to learning from examples is precisely the target of our investigation. Also, Definition 2.1 focuses on human comprehension of logic programs. Human out-of-sample predictive accuracy as a metric of human comprehension is not restricted to any computational representation. We define human comprehension for evaluating machine explanations in prediction tasks, allowing a broader applicability of this assessment in XAI. Contrary to the XAI literature which focuses primarily on the positive effect of explanations, we classify and examine their harmfulness. Both sides of the explanatory effect spectrum are important for a complete understanding of machine-human interactions.

We define explanatory effects based on empirical evaluation of the difference in human task performance following human learning from examples and human learning from machine explanations. We additionally define the explanatory effects of sequential teaching curricula which have not been inspected by existing definitions of comprehensibility. A sequential teaching curriculum models human learning via a sequence of teaching materials. Empirically, we consider curricula with noise-free training examples in this thesis. The source of human comprehension differences is restricted to only the human learning error when the machine-learned logic programs are correct. However, E_{ex} does not pose a limitation on the predictive accuracy of the machine-learned model from which the explanations are generated.

3.1.1 Beneficial and harmful explanatory effects

Definition 2.1 of comprehensibility [268] is a performance-based evaluation that indicates how well humans can simulate a machine-learned logic program by learning from it. Domain materials and explanations are given in a teaching curriculum which involves a machine “teacher” and human “students”. Human “students” are asked to classify the unseen test examples after having spent time learning from a predicate definition. As aforementioned, Definition 2.1 does

not distinguish human comprehension established on logic programs and that obtained from only ground facts. In contrast, our framework defines explanatory effects by differentiating learning from explanations and learning from examples.

Definition 3.1. (*Unaided human comprehension of examples, $\tau_h(D, H, E)$*): Given that D is a function, H is a human group and E is a set of input-output pairs of D , the unaided human comprehension of E is the mean accuracy of H after a brief study of E and without further sight can produce outputs of D from new random samples of its domain.

Example 3.1. (*Unaided human comprehension of string transformation examples*): Let D be a function that reverses and capitalises a string sequence. E consists of pairs of strings and reverse-upper-cased strings. According to Definition 3.1, unaided human comprehension is the mean accuracy of humans in predicting the correct reverse-upper-cased strings given new randomly sampled strings after learning from E .

Machine-explained human comprehension differs in that humans learn from an explanation produced by a machine. In XAI, explanations have diverse representations [192, 222]. To account for all representations generated by diverse procedures, the machine-explained human comprehension is representation-independent as we categorise a machine explanation as the output produced by any machine. The outputs from explanation generation procedures and machine learning algorithms are differentiated solely by their explanatory effects on human comprehension. In other words, we judge a machine's explanation not by its purpose or format but by its ability to facilitate high-accuracy human evaluations. We consider $M(E)$ an explanation produced by some Turing machine M where E is a set of input-output pairs of D . It makes sense to restrict $M(E)$ to an explanation that is always producible in a finite number of computation steps. However, evaluations of explanations that could take a great amount of time to generate are not always feasible in reality.

Definition 3.2. (*Machine-explained human comprehension of examples, $\tau_{ex}(D, H, M(E))$*): Given that D is a function, H is a human group, $M(E)$ is the output from a machine M and E is a set of input-output pairs of D , the machine-explained human comprehension of E is the mean accuracy of H after a brief study of $M(E)$ and without further sight can produce outputs of D from new random samples of its domain.

In the context of XAI, D can be a function describing input-output pairs of a prediction task, E is a set of training examples and $M(E)$ are explanations generated by a machine (e.g. some machine-learned output). For an ILP system, humans can inspect the machine-learned logic programs so M is the ILP learner. We can generalise to any machine learning system that produces symbolic output where M is the machine learner. “Black box” systems often use post-hoc explanations and require multiple post-processing steps [160, 137]. Then, M denotes a sequence of conversions from an uninterpretable machine-learned output to some human-readable representation. In a very extreme case when $M(E)$ is a binary sequence, our notion of machine-explained comprehension is still applicable but we would expect a lower human accuracy. In both Definition 3.1 and 3.2, a sensible accuracy metric can be determined based on the nature of the target function. For instance, when the output of D contains a continuous value, the accuracy can be computed using the mean squared error between human predictions and D ’s output.

Given a sufficient set of training examples E for learning the target concept, there are at least three possible types of human comprehension depending on the involvement of examples and explanations: 1) humans learn from only examples E , 2) humans learn from $M(E)$ and 3) humans learn from $M(E')$ where M uses a different set of examples E' . Definition 3.1 is related to the case (1) and Definition 3.2 is relevant to (2) and (3) respectively. However, we only examine (1) and (2) in the scope of this thesis. The machine producing the explanations and human subjects who learn from only examples have access to the same training set. Using the same training examples limits the variable to the presence of explanations. In the case of (3), the examples utilised by the machine introduce an additional variable. The effect of this extra variable is not the focus of this thesis.

We define explanatory effect [3] to measure the extent to which machine explanations can be correctly simulated by humans. The explanatory effect of machine explanations (for a classification task) is defined as the task performance difference between machine-explained (τ_{ex}) and unaided human comprehension (τ_h).

Definition 3.3. (*Explanatory effect of a machine explanation*, $E_{ex}(D, H, M(E))$):
Given that D is a function, H_1 and H_2 are sufficiently large and disjoint human groups randomly partitioned from H , $M(E)$ is the output from a machine M and E is a set of input-output pairs

of D , the explanatory effect of $M(E)$ is

$$E_{ex}(D, H, M(E)) = \tau_{ex}(D, H_1, M(E)) - \tau_h(D, H_2, E)$$

Randomly partitioned human groups need to be sufficiently large to ensure they have a similar mean and variance of cognitive ability. No human learner should receive both machine explanations and examples so the two partitions should be disjoint. In the case that machine explanations lead to an improvement in task performance, the explanatory effect is classified as beneficial. When this difference in task performance is negative, the explanatory effect is harmful. Since human understanding varies between tasks and materials, there is no guarantee that machine explanations produced by a procedure are always beneficial. Given a task, we can examine if a machine explanation is beneficial or not from the explanatory effect. This can be done objectively using a statistical test where interpreting the outcome is unambiguous.

Definition 3.4. (*Beneficial/harmful effect of a machine explanation*): Given that D is a function, H is a human group, a machine M and E is a set of input-output pairs of D ,

- $M(E)$ is beneficial to H if the explanatory effect $E_{ex}(D, H, M(E)) > 0$
- $M(E)$ is harmful to H if the explanatory effect $E_{ex}(D, H, M(E)) < 0$
- Otherwise, $M(E)$ 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. We also account for the case when the difference between machine-explained and unaided human comprehension is negligible. In the case that the mean difference is not significant, we cannot classify the effect of a machine explanation as either beneficial or harmful. Though only beneficial and harmful effects are highlighted in Definition 3.4, all three effects are of great interest as potential empirical outcomes. In particular, given that the presence of explanations leads to a harmful effect or does not have an effect, insights into the design and the complexity of the machine explanations can lead to improved human task performance.

3.1.2 Explanatory effect of a sequential teaching curriculum

Human learning has been demonstrated to depend on experience [87]. Humans utilise the strategy of reusing related experience to solve complex problems efficiently [14]. Complex concepts that are difficult to learn might result in low comprehension scores. To improve human comprehension, concepts can be broken down into smaller “chunks” with a lower cognitive load [45]. We consider human learning of these interrelated “chunks” as a curriculum. In such a curriculum, a machine acts as a teacher who presents blocks of teaching material to humans one after another. Informally, a sequential teaching curriculum (STC) describes a curriculum where teaching materials are sequentially ordered and interdependent. These humans who play the role of students are iteratively tested and their comprehension scores are recorded for each block of teaching material. An STC contains a set of interrelated concepts, examples and explanations. According to the literature on Psychology [37, 14, 87], the order of learning affects human learning performance. The purpose of defining STCs is to study the interactive effect between the order of human learning and machine explanations. The arrangement of concepts in a curriculum is specified using an ordering function which ranks and differentiates concepts.

Definition 3.5. (*Curriculum rank function*, $\lambda(D, E, M)$) *Given a function D , a set of examples E and a machine learning algorithm M , λ returns a non-negative integer. For any two tuples of inputs $\lambda(D_1, E_1, M_1) \neq \lambda(D_2, E_2, M_2)$ if $(D_1, E_1, M_1) \neq (D_2, E_2, M_2)$.*

Every concept is expressed as a tuple of teaching material. The order of concepts received by human learners is determined by a rank function which is designated to assign a score to every tuple of teaching material based on the concept definitions, examples and explanations. Each STC contains a finite number of concepts and at least one concept is described by a tuple of teaching material.

Definition 3.6. (*Sequential teaching curriculum*, $STC(DS, ES, MS, \lambda)$): *Given that $DS = \{D_1, D_2, \dots, D_n\}$ is a set of functions, $ES = \{E_1, E_2, \dots, E_n\}$ contains sets of input-output pairs, $MS = \{M_1, M_2, \dots, M_n\}$ is a set of machines and λ is a rank function, a sequential teaching curriculum $STC(DS, ES, MS, \lambda)$ is a finite set of tuples $\{(R_1, D_1, E_1, M_1), (R_2, D_2, E_2, M_2), \dots, (R_n, D_n, E_n, M_n)\}$ where R_i is the rank $\lambda(D_i, E_i, M_i)$ for $1 \leq i \leq n$.*

Concepts in an STC can be ordered with respect to increasing rank value. We consider a curriculum incremental when functions requiring greater (worse-case) time complexity to compute have higher ranks.

Example 3.2. (*An incremental curriculum for learning merge sort*): *The time complexity of merging a sequence of items of size n is $O(n)$ while merge sort has a time complexity of $O(n \log n)$. An incremental curriculum for learning merge sort is by first learning how to merge and then learning how to perform merge sort.*

The definitions of human comprehension and explanatory effects in Section 3.1.1 are extended to account for sequential curricula and the ordering of teaching materials. Unaided human comprehension and machine-explained human comprehension can be distinguished by the cardinality of machine explanations. Human comprehension is aided by explanations (τ_{ex}) if a tuple of teaching material contains a non-empty set of machine explanations. Otherwise, human comprehension is based on training examples (τ_h).

Definition 3.7. (*Human comprehension of examples in a sequential teaching curriculum*, $\tau_{seq}(C, H)$): *Given that C is a sequential teaching curriculum and H is a human group, the human comprehension of examples in the sequential teaching curriculum is*

$$\tau_{seq}(C, H) = \left\{ (R_i, \tau) \left| \begin{array}{ll} \tau = \tau_{ex}(D_i, H, M_i(E_i)), & \text{if } M_i(E_i) \neq \emptyset \\ \tau = \tau_h(D_i, H, E_i), & \text{otherwise} \end{array} \right. \right\} \\ \text{for } 1 \leq i \leq n, (R_i, D_i, E_i, M_i) \in C$$

In Definition 3.7, comprehension measurements are obtained with respect to the ranking of concepts in a sequential teaching curriculum. Based on the definition of the rank function, each comprehension score can be uniquely identified by the rank of the corresponding concept in a curriculum via an assignment of distinct rank values to concepts. We define the effect of an STC on human comprehension of a concept over another STC by measuring human comprehension scores of the concept in two curricula and comparing them.

Definition 3.8. (*Effect of a sequential teaching curriculum*, $E_{seq}(C_1, C_2, D, H)$): *Given C_1 and C_2 are sequential teaching curricula and H_1 and H_2 are sufficiently large and disjoint*

human groups randomly partitioned from H , the explanatory effect of curriculum C_1 over curriculum C_2 on a function D is

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

where R_1 and R_2 are the ranks of D in C_1 and C_2 respectively and (R_1, τ_1) and (R_2, τ_2) are measurements of human comprehension in $\tau_{seq}(C_1, H_1)$ and $\tau_{seq}(C_2, H_2)$ respectively.

Definition 3.8 provides a general test of the degree to which human comprehension of a concept is affected by a curriculum C_1 over some other curriculum C_2 . When C_1 and C_2 only differ in their rank functions, we can concentrate on the effect of the sequential arrangement of concepts on human comprehension. A difference in human comprehension as a result of learning from one STC over another STC is classified as either beneficial, harmful or no effect.

Definition 3.9. (*Beneficial/harmful effect of a sequential teaching curriculum*):

Given that C_1 and C_2 are sequential teaching curricula, D is a function and H is a human group, the curriculum C_1 in comparison with the curriculum C_2 is

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

In Definition 3.9, the effect on human predictions of the function D is beneficial when the curriculum C_1 results in better predictive accuracy of D than the curriculum C_2 . Conversely, the effect by C_1 is harmful in the case that C_1 causes a lower human predictive accuracy than C_2 . Whether or not an effect of a sequential teaching curriculum is significantly beneficial or harmful can be determined similarly to analysing the effect of a machine explanation in Definition 3.4. One can use an independent two-sample t-test or similar statistical test methods such as ANOVA tests.

3.2 Theory of a cognitive window

Donald Michie’s Human Window [182, 187] describes a compromise between human executability and human intelligibility for human understanding of computational representations of knowledge. To understand this trade-off in non-trivial tasks, our goal is to explore the relationship between the properties of machine-learned logic programs and their explanatory effects. This requires means to model human execution and learning of logic programs.

Limitation in human execution is formulated by a constraint on the cognitive cost. This constraint describes that a machine-learned logic program has no beneficial explanatory effect if its cognitive cost in executing the program is not lower than the cognitive cost of a program that uses less background knowledge. We define a cognitive cost of evaluating a logic program. The computation of Prolog is used as a model of human execution of declarative knowledge when humans study explanations from logic programs. Human execution is initiated with a query specifying a goal by the human. The limit of working memory capacity corresponds to restricted tape length and instruction complexity in Turing Machines. Human actions and decision-making in problem-solving are guided by the hierarchy of goals [44]. The decomposition of goals into sub-goals can be described by a set of rules. Rules as a knowledge representation can be either seen as a procedure or a declaration. In complex problem-solving domains, human declarative memory is associated with explicit rule verbalisation [10, 267]. An increase in the complexity of goal declarations can lead to a larger load on working memory which negatively affects human problem-solving performance [45]. To define a cognitive cost of a logic program, we assume that:

- An increase in the execution complexity of the problem solution can have a negative effect on performance
- The execution of goals utilises declarative memory and its complexity can be measured by the Kolmogorov complexity

The bound on human learning is accounted for by a constraint on hypothesis space size. This bound says that a machine-learned logic program has a harmful explanatory effect if the associated hypothesis space is larger than the hypothesis space searchable by a group of humans.

It has been shown that human concept attainment based on rules can be characterised as a search in a space of hypotheses [37]. This notion of search-driven learning, guided by some preference order, is analogical to machine learning of consistent hypotheses in the version space [197]. The connections between human concept hierarchisation [44, 223] and hypothesis biases are also explored. ILP approaches such as MIL use background knowledge, higher-order rules and invented predicates to learn logic programs that can describe the knowledge hierarchy of concepts. We make the following two assumptions:

- Human learners are version space learners who can search a hypothesis space with a limited size
- Human learning is guided by some form of abstract rules such as meta-rules which construct the sub-goal structure, and it uses predicates as background knowledge

We hypothesise the existence of a cognitive window including these two constraints. In Chapter 4 and 5, we provide empirical support for this hypothesis in the contexts of teaching a simple two-play game and teaching an advanced computational algorithm. In addition, we explore human comprehension in sequential teaching curricula. We postulate that learning while being able to refer to obtained knowledge and concepts benefits human task performance. This is captured by our hypothesis of the improvement of sequential teaching curriculum on human comprehension based on the Blumer bound [28]. This hypothesis describes that a curriculum has a beneficial effect when it introduces the appropriate reusable concepts to lower the overall sample complexity. This hypothesis is supported by experimental evidence in Chapter 5.

3.2.1 Cognitive cost of a logic program Cog

Specific to logic programs, we define a cognitive cost Cog [3, 2] to estimate the complexity of maintaining concepts described by logic terms and atoms in the working memory. A cognitive cost is defined based on the bit length of logic terms and atoms that appear in the computation stack of a logic program during the execution of a goal. We assume that every finite length term or atom can be minimally represented in datalog, its complexity equals the number of symbols used to express it according to the time-bounded Kolmogorov complexity [115]. Constants

and special symbols such as $\top$ and $\perp$ have a cost corresponding to their lengths. A variable refers to a location in memory that can be identified by a fixed-size pointer. A predicate is a composition of symbols. In memory, a predicate is identified by a predicate symbol of a fixed size which is applied to a set of arguments. For example, if all symbols are ASCII characters, then each symbol is 8 bits in length. For simplicity, we omit the unit of cognitive cost.

Definition 3.10. (*Cognitive cost of a logic term and atom*, $\psi(T)$): Given T a logic term or atom, the cost $\psi(T)$ can be computed as follows:

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

Example 3.3. (*Cognitive cost of an atom*): An atom $p(V1, V2)$ with variables $V1$ and $V2$ has a cost $\psi(p(V1, V2)) = 3$.

Since we augment conventional datalog with complex predicates to represent static objects, we show an example of computing the cognitive cost of an atom that contains another predicate as an argument.

Example 3.4. (*Cognitive cost of an atom with a complex argument*): An atom $list(1, list(0, \epsilon))$ which represents a list $[1, 0]$ has a cognitive cost $\psi(list(1, list(0, \epsilon))) = 5$.

When a human is given a query q to compute, an execution stack represents the trace of the computation according to SLD-resolution in Prolog. and represents the memory of grounded logic terms and atoms instantiated.

Definition 3.11. (*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 the shortest 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 $\perp$.

The execution stack of a goal corresponds to the derivation of the empty clause and the shortest successful branch of the SLD-tree. The shortest successful refutation involves no backtracking. Otherwise, a shorter refutation exists. The shortest derivation path corresponds to a lower bound of execution length and executional complexity.

Every pair of open and resolved goals is declared in the stack. When the size of the stack is not sufficiently large such that a derivation does not lead to an empty clause, the stack ends with $\perp$ which denotes this partial derivation. For a successful refutation, the execution stack ends with $\top$. The execution stack does not contain derivation of the sub-goals in primitives because they are parts of the background knowledge for reusing. We assume the unification of literals in a goal takes place in sequential order from left to right.

Example 3.5. (*Execution stack*): Let $\text{sum}/3$ be a primitive that computes the sum of two integer inputs. The query is $q = \text{sum}(5, 3, X)$. Given the depth of the stack is 3, execution of the goal evaluates to $\top$.

$S(\text{sum}, 3, \text{sum}(5, 3, X))$	$\psi(T)$
$\text{sum}(5, 3, X)$	4
$\text{sum}(5, 3, 8)$	4
$\top$	1

Definition 3.11 models the human working memory that maintains a limited number of goals. The complexity of executing a set of rules is the sum of the cognitive cost of logic terms and atoms in the execution stack. The depth bound of an execution stack ensures that the computation of the sum of cognitive costs always halts given sufficient resources.

Definition 3.12. (*Cognitive cost of a datalog program*, $\text{Cog}(P, q)$): Given a query q , a non-negative integer l , the cognitive cost of a datalog program P is

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

Example 3.6. (*Cognitive cost of a datalog program*): Based on Example 3.5, given primitive $\text{sum}/3$ which computes the sum of two integer inputs and query $q = \text{sum}(5, 3, X)$, the cognitive cost of $\text{sum}/3$ is $\text{Cog}(\text{sum}, \text{sum}(5, 3, X)) = 9$ when the depth of the execution stack is 3.

In human problem-solving, background knowledge facilitates the transfer of existing solutions [11, 227] to the current problem context. In the case that auxiliary knowledge is absent, the construction of the solution relies on low-level goals and performance is limited by human operational error. We define this type of problem-solving solution as the primitive problem solution which utilises the minimum amount of task background knowledge.

Definition 3.13. (*Minimum primitive solution program*: Given a set of primitives ϕ and examples E , a datalog program $\bar{M}(\phi, E)$ learned from examples E using a machine learning algorithm $\bar{M}$ and a set of primitives $\phi' \subseteq \phi$ is called a minimum primitive solution program if and only if for all $\phi'' \subseteq \phi$ where $|\phi''| < |\phi'|$ and for all machine learning algorithm M' using ϕ'' , there exists no machine-learned program $M'(E)$ that is correct with respect to examples E .

Given a set of primitives and training examples, a minimum primitive solution program is a correct hypothesis program that uses a necessary and sufficient subset of the primitives. Since a minimum primitive solution program is also a datalog, its cognitive cost can be similarly computed as that of a datalog program.

Definition 3.14. (*Cognitive cost of a minimum primitive problem solution*, $CogP(E, \bar{M}, \phi, q)$): Given examples E , primitive set ϕ , a query q and a 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.

Efficient problem-solving is enabled by mapping from background knowledge which allows “shortcuts” to be integrated into the current solution. In comparison with a primitive problem solution, executional shortcuts in a non-primitive solution could lead to less cognitive load on working memory and better performance during the execution of the solution. In Chapter 4 and 5, the cognitive costs of machine-learned solutions are compared with the cognitive costs of some primitive problem solutions learned by MIL learners where we examine the relationship between cognitive cost and explanatory effects.

3.2.2 A cognitive window of human comprehension

Our theory of a cognitive window of human comprehension mainly consists of two constraints. 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 boundary of human search in the hypothesis space, given that every human learner is considered a version space learner with a limited search capability. We refer to Proposition 2.2: given p predicate symbols of arity at most i and m meta-rules, and the number of hypothesis programs with at most j body literals and n clauses is $O(m^n p^{(1+j)n})$.

Definition 3.15. (*Cognitive bound on the hypothesis space size, $B(P, H)$*): Let P be a machine-learned datalog program with 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 definition of the human searchable hypothesis space size $B(P, H)$ relates to the MIL complexity analysis in [158, 70]. This bound can be empirically demonstrated. In Section 4.5, we demonstrate that humans can only learn a subset of the rules for winning a two-player game when a logic program contains five clauses. 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 parts of the original rules are learned, problem solutions which produce action sequences are not complete and errors become more likely to occur.

Definition 3.16. (*Cognitive window of a machine-learned theory [3]*): Given a function D , primitives ϕ , a machine learning algorithm M , a machine learning algorithm $\bar{M}$ that learns a primitive solution program and examples E , $M(E)$ is a machine-learned theory using ϕ that is associated with hypothesis space S . For a human group H ,

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

This hypothesis says 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 background knowledge. Explanations cannot be beneficial when one of the two conditions holds. That is when explanations have inappropriate informativeness or are more cognitively taxing than a primitive solution to a problem. We argue that machine-learned logic programs which cannot provide abstract descriptions are difficult to comprehend by humans. Executional shortcuts should be incorporated for humans to effectively evaluate the learned logic programs. We support this hypothesis based on empirical evidence in Chapter 4 and 5.

3.2.3 Sequential teaching improvement

The complexity analysis in Theorem 2.1 [71] demonstrates a sample complexity decrease with the invention of appropriate predicate definition by a machine learner. Previously learned predicate definitions extend the background and reusing learned predicates can reduce the size of the target hypothesis. This results in a smaller hypothesis space. From our assumption that human learners are version space learners with limited capacity, we consider an analogy between human learning by using previously learned concepts and reusing obtained predicate definitions. Recall in Definition 3.9 in Section 3.1.2, we compare two curricula to obtain the

relative human task performance improvement or degradation. We attribute a beneficial effect of a curriculum on human comprehension over another curriculum to a reduction in hypothesis space size. We refer to the sample complexity result in Theorem 2.1 [71]. It says that given two MIL learners with the same set of training examples, the increase in sample complexity is associated with an increased error in prediction. An appropriate curriculum design can lower the sample complexity by allowing the machine learner to build reusable smaller programs. We hypothesise that human predictive performance can be improved by developing reusable knowledge.

Definition 3.17. (*Sequential teaching curriculum improvement [2]*): Let C_1 and C_2 be STCs that differ only in rank functions, H be a human group and D be a function in C_1 and C_2 . In C_1 , let u denote the minimum number of clauses in a logic program that computes D with p predicates. In C_2 , let $u + k$ be the minimum number of clauses in a logic program that computes D with $p + c$ predicates. Then, $E_{seq}(C_1, C_2, D, H) > 0$ when $u \ln(p) < (u + k) \ln(p + c)$ (Theorem 2.1).

Hypothesis 3.17 implies that for a fixed set of training examples in two STCs, a reduction in sample complexity correlates to a decrease in predictive error and an improvement in predictive performance. We hypothesise that given a function D and a fixed human search bound $B(D, H)$ for a human group H , human learners are more likely to predict D if the hypothesis space associated becomes smaller as a result of developing reusable knowledge. We provide empirical support for this hypothesis in Chapter 5.

3.3 Summary

We contribute to XAI by proposing our E_{ex} framework. It include operable definitions of explanatory effects and effects of sequential teaching curricula (STCs) based on human out-of-sample predictive accuracy. These definitions enable the quantification of the beneficiality and harmfulness of machine explanations on human comprehension. Our framework assesses general machine explanations so it is representation-independent and has broad applicability. With appropriate human predictive accuracy measures, it can also be tailored for evaluating

human comprehension of specific prediction tasks. We expand on this in Chapter 6. Since we aim to demonstrate the E_{ex} framework in the context of logic programming, we attempt to formulate the limitations of human learning and human evaluation of logic programs. These limitations relate to human intelligibility and executability in Michie’s analysis of the Human Window. Our theory of a cognitive window bridges human comprehension and the properties of machine-learned logic programs. A cognitive cost Cog is defined to estimate the complexity of evaluating logic programs in human working memory. We also hypothesise a bound on the size of searchable hypothesis space by a group of human learners. For a machine-learned logic program to have a beneficial effect on human comprehension, we postulate that it should have a low cognitive cost and can be learned within the human searchable hypothesis space size. This means that such a logic program needs to contain executional shortcuts and have low descriptive complexity. This theory contains two novel hypotheses: 1) a cognitive window of a machine-learned theory and 2) the human comprehension improvement by a sequential teaching curriculum. In Chapter 4 and 5, we carry out case studies in two non-trivial human reasoning domains and provide empirical evidence to support these two hypotheses.

Chapter 4

Beneficial and harmful explanatory machine learning

We explain experiments and empirical results supporting **Sub-thesis 1** from Chapter 1 and Hypothesis 3.16 of a cognitive window. We include parts from our published Machine Learning Journal paper [3].

4.1 Introduction

Prior to the work described in this chapter, machine learning comprehensibility has been explored in [215] to demonstrate the first instance of Ultra-Strong Machine Learning (USML) based on this operational definition. This indicates a beneficial effect of comprehensible machine-learned output on human knowledge attainment. As the complexity of explanations increases, the load on human information processing in working memory also increases. However, the negative effects of machine explanation on human comprehension have not been explored.

To address this, we examine the explanatory effects on human comprehension (beneficial or harmful) using the E_{ex} frameworks described in Chapter 3. For a simple two-player game domain, we present explanations of machine-learned logic programs of game strategies to human learners in human trials. Notably, we show that explaining a machine-learned logic program can have a measurable beneficial effect on human comprehension. In addition, we have observed

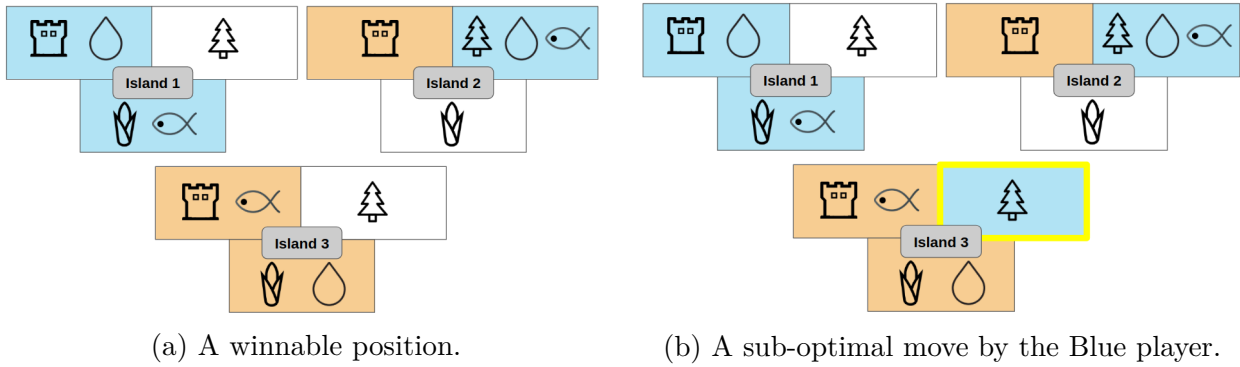


Figure 4.1: An example of a Noughts and Crosses isomorphism called the Island Game. Every turn, each player occupies a cell with some resources (marked by the castle, fish, tree icons, etc). A player wins if he or she controls three cells on the same island or three pieces of the same resource. In our experiments, human participants, who play Blue, are presented with a game position and have to choose some alternative moves between the white cells. In Figure 4.1b, the cell with a yellow border highlight indicates a “bad” move that leads to a loss if the Orange player plays optimally. More details of the material design are given in Section 4.4.1.

that a machine-learned logic program has a quantifiable harmful effect when it is explained to humans. Both support **Sub-thesis 1**. In our hypothesis of a cognitive window (Hypothesis 3.16 of Chapter 3), machine-learned theories do not have a beneficial effect if they are either 1) informatively overwhelming or 2) more cognitively expensive than problem solutions that involve no auxiliary information. Results show that when human learners are presented with explanations based on a machine-learned theory that has low descriptive and executional complexity, human task performance is higher compared to learning from examples alone. This supports Hypothesis 3.16.

In Chapter 2, we review research on machine learning of game strategies. USML of game strategies can help humans achieve higher performance in games and help them interpret the decisions of machine agents. Games also represent a class of problems that generally require explicit human reasoning. Empirical studies of games can help us understand better the limitations of human inference. We use a two-player game that we designed as the material for experimentation. This game called the Island Game is isomorphic to Noughts and Crosses (Figure 4.1). The selection of Noughts and Crosses as the base ensures an appropriate level of task difficulty. We use a small but sufficient set of examples to machine-learn correct strategies for winning. For different depths of this game, the amount of information carried by the machine-learned strategies may vary. This gives to a range of complexity levels to test human learning and reasoning. The complexity of machine-learned game strategies allows us to correlate the

properties of these logic programs with human comprehension.

We arrange this chapter as follows. In Section 4.2, we describe a game learning system **Meta-Interpretive exPLAINable** game learner *MIPlain* based on modified *MIGO* system [212] (explained in Chapter 2) to learn logic programs with reduced size. The MIL learning framework *MIGO* [212] learns logic programs that represent strategies of Noughts and Crosses and Hexapawn. In Section 4.3, we introduce our analysis approach and experimental hypotheses. In Section 4.4, we elaborate on the invention of the Island Game domain to reduce the ceiling effect and the two-group experimental setup for comparing the effects of explanations. Textual and visual explanations are generated from programs learned by *MIPlain* and shown to treated participants along with training examples for winning the Island Game. In Section 4.5, we present results from our experiments. In Section 4.6, we discuss empirical evidence that supports **Sub-thesis 1** and Hypothesis 3.16 of a cognitive window.

4.2 Meta-interpretive learning of game strategy

The MIL game learning framework *MIGO* [212] is based on the adapted Prolog meta-interpreter Metagol [75]. *MIGO* learns exclusively from positive examples by playing against the optimal opponent. For Noughts and Crosses and Hexapawn, *MIGO* produces a rule-like game strategy that supports human understanding and was demonstrated to converge using less training data compared to Deep and classical Q-Learning.

When *MIGO* learns to play Noughts and Crosses, it uses three primitives, *move*/2, *won*/1, and *drawn*/1. These primitives correspond to a move generator, a won and a drawn classifier respectively. These primitives represent the minimal information that a human would know before playing Noughts and Crosses and Hexapawn. For successive values of k , *MIGO* learns a series of interrelated definitions for predicates *win_k*/2 for playing as either X or O. These predicates define the minimax winning strategy in k -ply (winning in k moves).

In this work, we use a modified version of *MIGO*, called **Meta-Interpretive exPLAINable** game learner (*MIPlain*)¹. This variant of *MIGO* focuses on learning the task of winning the game

¹MIPlain source is available at <https://github.com/LAi1997/MIPlain>.

Table 4.1: A set of win rules learned by *MIGO*. *MIGO*'s background knowledge contains a general move generator *move/2* and a won classifier *won/1* to encode the minimum rules of the game. The program is dyadic and *win_1/2* can be reduced to *win_1(A, B) :- move(A, B), won(B)* by removing literals after unfolding. A more detailed description of the program learned by *MIGO* was given in [212]

Depth	Rules
1	$\text{win_1}(A,B) :- \text{win_1_1_1}(A,B), \text{won}(B).$ $\text{win_1_1_1}(A,B) :- \text{move}(A,B), \text{won}(B).$
2	$\text{win_2}(A,B) :- \text{win_2_1_1}(A,B), \text{not}(\text{win_2_1_1}(B,C)).$ $\text{win_2_1_1}(A,B) :- \text{move}(A,B), \text{not}(\text{win_1}(B,C)).$
3	$\text{win_3}(A,B) :- \text{win_3_1_1}(A,B), \text{not}(\text{win_3_1_1}(B,C)).$ $\text{win_3_1_1}(A,B) :- \text{win_2_1_1}(A,B), \text{not}(\text{win_2}(B,C)).$

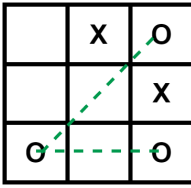


Figure 4.2: Example of pairs. O has two pairs (green lines) and X has no pairs.

Name	Meta-rule
postcon1	$P(A, B) \leftarrow Q(A, B), R(B).$
postcon2	$P(A) \leftarrow Q(A, B), R(B).$
conjunc1	$P(A) \leftarrow Q(A, S, T), R(A).$
conjunc2	$P(A) \leftarrow Q(A, S, T), R(A, U, V).$

Figure 4.3: Letters P, Q, R, S, T, U, and V denote existentially quantified second-order variables and A, B, and C are universally quantified first-order variables.

of Noughts and Crosses with some adaptations. In addition to learning from positive examples, *MIPlain* learns strategies from negative examples for the task of winning in order to reduce the generality of the hypothesis learned. Based on supplied training examples, *MIPlain* performs a selection of hypotheses using a program cost function as a hypothesis space constraint [76]. The tree cost iteratively checks the number of SLD-resolutions performed when calling primitives *move/2* and *number_of_pairs/3* as programs are constructed from meta-substitutions. A learned program with the lowest cost implies the most efficient and correct applications of the primitives.

An additional primitive *number_of_pairs/3* is provided to *MIPlain* which describes the number of pairs for a player (X or O) on a given board. A pair is the alignment of two marks of one player and the third square of this line is empty. Examples of some pairs are shown in Figure 4.2. The primitive *number_of_pairs/3* is an executional shortcut that reduces the

Table 4.2: Logic program learned by *MIPlain*. It represents a strategy for the first player to win at different depths of the game. The predicate $win_3_4/1$ can be reduced to $win_3_4(A) : -win_2(A, B)$. The program learned by *MIPlain* can be described as: a board A is won at depth 1 if there exists a move from A to B such that B is won; a board A is won at depth 2 if there exists a move from A to B such that X has exactly two pairs and O has no pairs in B; a board A is won at depth 3 if there exists a move from A to B such that X has exactly one pair in B and there exists a move from B to C such that X does not have any pair in C and C is won at depth 2 for X.

Depth	Rules
1	$win_1(A, B) : -move(A, B), won(B).$
2	$win_2(A, B) : -move(A, B), win_2_1(B).$ $win_2_1(A) : -number_of_pairs(A, x, 2),$ $number_of_pairs(A, o, 0).$
3	$win_3(A, B) : -move(A, B), win_3_1(B).$ $win_3_1(A) : -number_of_pairs(A, x, 1), win_3_2(A).$ $win_3_2(A) : -move(A, B), win_3_3(B).$ $win_3_3(A) : -number_of_pairs(A, x, 0), win_3_4(A).$ $win_3_4(A) : -move(A, B), win_2_1(B).$

Table 4.3: Comparison of program size (unfolded and without redundant literals) and cognitive cost. For winning in two (win_2) and three moves (win_3), the logic programs learned by *MIPlain* are smaller and have lower cognitive costs.

Predicate definition	Smaller program size	Lower cognitive cost
win_1	Both are the same	Both are the same
win_2	<i>MIPlain</i>	<i>MIPlain</i>
win_3	<i>MIPlain</i>	<i>MIPlain</i>

depth of the minimax search.

Remark 4.1. *Given that the training examples of Noughts and Crosses are winnable and MIPlain uses the set of primitives $\phi = \{move/2, won/1, number_of_pairs/3\}$, a minimum primitive solution program is produced by MIGO. This is because MIGO uses primitives $\{move/2, won/1\}$ which is a strict subset of ϕ for making a move and deciding a win when the input is winnable. Primitives $move/2$ and $won/1$ are also necessary and sufficient primitives to win Noughts and Crosses and no theory can be learned using a subset of ϕ with the cardinality of one.*

The meta-rules used by *MIPlain* are presented in Figure 4.3. There are two meta-rules that

describe post-conditions with either a monadic or dyadic head (*postcond1* and *postcond2*). Two other meta-rules specify the conjunctive conditions of the input argument. Variables in first or both body literals are existentially quantified and can be bounded to constants during meta-substitution. The learned strategies presented in Table 4.2 describe patterns in game positions that the player's optimal move has to satisfy. Owing to the setup of the game, *MIPlain* learns a program for playing as X assuming X starts the game.

Remark 4.2. *The program P learned by *MIPlain* has less cognitive cost than the one learned by *MIGO* except for queries concerning win_1/2 . From Remark 4.1 given sufficient examples E , *MIGO* learns primitive solution programs from the background knowledge $\phi = \{\text{move/2}, \text{won/1}, \text{number_of_pairs/3}\}$. ϕ is also used by *MIPlain* as background knowledge. Let $\bar{M}$ denote *MIGO* and P_i ($1 \leq i \leq 3$) denote $\text{win_}i$ predicate learned by *MIPlain*. Let $x_i = \text{win_}i(s_i, V)$ where s_i represents a position winnable in i moves and V is a variable that bounds to an optimal winning move. We examine the cognitive costs in all three cases of i . When $i = 1$, according to Table 4.1 and 4.2, we have $\text{Cog}(P_1, x_1) = \text{CogP}(E, \bar{M}, \phi, x_1)$ since both *MIGO* and *MIPlain* learn the same program. When $i = 2$ and $i = 3$, the cognitive cost of computing x_2 and x_3 based on P_2 and P_3 are much more efficient, thus $\text{Cog}(P_2, x_2) < \text{CogP}(E, \bar{M}, \phi, x_2)$ and $\text{Cog}(P_3, x_3) < \text{CogP}(E, \bar{M}, \phi, x_3)$.*

For successive values of k , $\text{win_}k/2$ definitions are inter-dependent predicates which specify the status of the game in terms of the number of pairs owned by player X or O. Since the predicate *number_of_pairs/3* identifies how close a player is to have three pieces in a line, win_2/2 and win_3/2 definitions specify the advantage of player X with respect to player O for winning the game.

The key differences of programs learned by *MIGO* and *MIPlain* are summarised in Table 4.3. From Table 4.1 and 4.2, the win_2/2 and win_3/2 definitions learned by *MIPlain* are smaller in size compared with those learned by *MIGO* when they are unfolded. According to Remark 4.2, *MIPlain* learns less cognitive costly win_2/2 and win_3/2 .

4.3 Experimental framework

We use a two-group design: 1) human learning from training examples (self-learning) and 2) human learning from machine explanations and training examples (machine-aided learning). The self-learning group is the control group and the machine-aided learning group is the treatment group. Though we acknowledge the possibility of other control group designs, our main focus is to examine the effects of machine explanations on human task performance where this two-group approach is sufficient.

In addition, we assess the correlation between the cognitive window and human comprehension. This framework focuses on 1) a set of criteria for evaluating the participants' learning quality from their own textual descriptions of learned strategies and 2) an outline of experimental hypotheses. For game playing, we assume humans are able to explain actions by verbalising procedural rules of strategy. We expect textual answers to provide insights into human decision-making and knowledge acquisition.

4.3.1 Quality of textual answers

The quality of textual answers can be affected by multiple factors such as motivation, familiarity with the introduced concepts and understanding of the game rules. We take into account these factors in the evaluation criteria.

Definition 4.1. (*Primitive coverage of a textual answer*): A textual answer correctly describes a primitive if the semantic meaning of the primitive is unambiguously stated in the response. The primitive coverage is the number of primitives in a machine-learned theory that is described correctly in a textual answer.

Definition 4.2. (*Quality of a textual answer, $Q(r)$*): A textual answer r is checked against the specifications from Table 4.4 in increasing order from criteria level 1 to level 4. $Q(r)$ is the highest level i that r can satisfy. When a response does not satisfy any of the higher levels, the quality of this response is the lowest level 0.

To illustrate, we consider the predicate `win_2/2` learned by *MIPlain* (Table 4.2). Primitive

Table 4.4: Criteria for evaluating textual answers and examples for category *win_2/2*.

$Q(r)$	Criteria	Exemplary r
Level 0	r does not fit into any of the categories below	“Follow the instructions.”
Level 1	One or more primitives in the machine-learned theory, directly or by synonyms, are described correctly in r	“This move gives me a pair.”
Level 2	All primitives in the machine-learned theory, directly or by synonyms, are described correctly in r	“I should have picked this move to prevent the opponent and get two attacks.”
Level 3	r is unambiguous and all primitives are described correctly, directly or by synonyms, in the same order as in the executional stack of the machine-learned theory	“This move gives me two attacks and prevents the opponent from getting a pair.”
Level 4	r explains one or more primitives in the machine-learned theory in correct causal relations, directly or by synonyms	“This is a good move because by making two pairs and blocking the opponent, the opponent cannot win in one turn and can only block one of my pairs.”

predicates are *move/2* and *number_of_pairs/3*. We present in Table 4.4 a number of examples of textual answers and based on this table, we specify our classification of high and low-quality textual responses.

Definition 4.3. (*High (HQ) / low (LQ) quality textual answer*): A HQ response rh has $Q(rh) \geq 3$ and a LQ response rl has $Q(rl) < 3$.

A high-quality response reflects a high motivation and a good understanding of game concepts and strategy. On the other hand, a poor-quality response demonstrates a lack of motivation or poor understanding.

4.3.2 Experimental hypotheses

In this section, we define our experimental hypotheses and describe our intentions. We refer to the explanatory effect E_{ex} framework for definitions of comprehensibility and cognitive cost.

Evaluations of these hypotheses are discussed in Section 4.6. Let M denote a machine learning algorithm. E stands for examples, D is a logic program representing the definition of a target predicate, and H is a group of participants sampled from a human population. $M(E)$ denotes a machine-learned theory which belongs to a definite clause program class with hypothesis space S . $\bar{M}$ denotes a minimum primitive solution learning algorithm.

We are interested in demonstrating whether 1) the textual answer quality of learned knowledge reflects comprehension, 2) there exist cognitive bounds for humans to provide textual answers of higher quality and 3) the machine-learned theory helps improve the quality of textual answers.

H1: *Unaided human comprehension $\tau_h(D, H, E)$ and machine-explained human comprehension $\tau_{ex}(D, H, M(E))$ manifest in textual answer quality $Q(r)$.* We examine if high post-training accuracy correlates with high response quality and high primitive coverage of each question category.

H2: *Difficulty for human participants to provide textual answers increases with quality $Q(r)$.* We examine if the proportion of textual answers reduces with respect to high response quality and high primitive coverage of each question category.

H3: *Machine-learned theory $M(E)$ improves textual answer quality $Q(r)$.* We examine if machine-aided learning results in more HQ responses.

The impact of a cognitive window on explanatory effects is examined via the following hypotheses. ϕ is a set of primitives introduced to H . Let x denote the set of questions that human $h \in H$ answers after learning.

H4: *Learning a complex theory ($|S| > B(M(E), H)$) exceeding the cognitive bound leads to a harmful explanatory effect ($E_{ex}(D, H, M(E)) < 0$).* We examine if the post-training accuracy, after studying a machine-learned theory that participants cannot recall fully, is worse than the accuracy following self-learning.

H5: *Applying a theory without a low cognitive cost ($\text{Cog}(M(E), x) \geq \text{CogP}(E, \bar{M}, \phi, x)$) does*

not lead to a beneficial explanatory effect ($E_{ex}(D, H, M(E)) \leq 0$). We examine if the post-training accuracy, after studying a machine-learned theory that is cognitively costly, is equal to or worse than the accuracy following self-learning.

4.4 Experiments

This section introduces the materials and experimental procedure ² which we designed to examine the explanatory effects of a machine-learned logic theory. The design of the experimental work has been assisted by our collaborators from the University of Bamberg. I was responsible for the implementation of the experimental interface, the collection and analysis of results. We invented the Island Game based on Noughts and Crosses to prevent a ceiling effect on task performance. We also describe the experiment interface and present experimental results which indicate 1) a beneficial effect, 2) a harmful effect and 3) a bound on human learning capacity.

4.4.1 Materials

Noughts and Crosses is assumed to be a well-known game that the majority of participants are familiar with. This implies many participants already play optimally before training with explanations. There can be a ceiling effect and little room for potential performance increase. In order to address this issue, the *Island Game* was designed as a problem isomorphic to Noughts and Crosses.

Problem masking

In Psychology, isomorphic problems are defined as “problems whose solutions and moves can be placed in a one-to-one relation with the solutions and moves of the given problem” [279]. This changes the superficial presentation of a problem without modifying the underlying structure [104, 125, 249]. This arrangement does not impede solving the problem if the original problem is consciously recognized as an analogy. Given that the Island Game is a major re-design of the Noughts and Crosses representation, it was expected that participants would not easily recall

²Experimental materials, interface and results are available at <https://github.com/LAi1997/MIPlain>

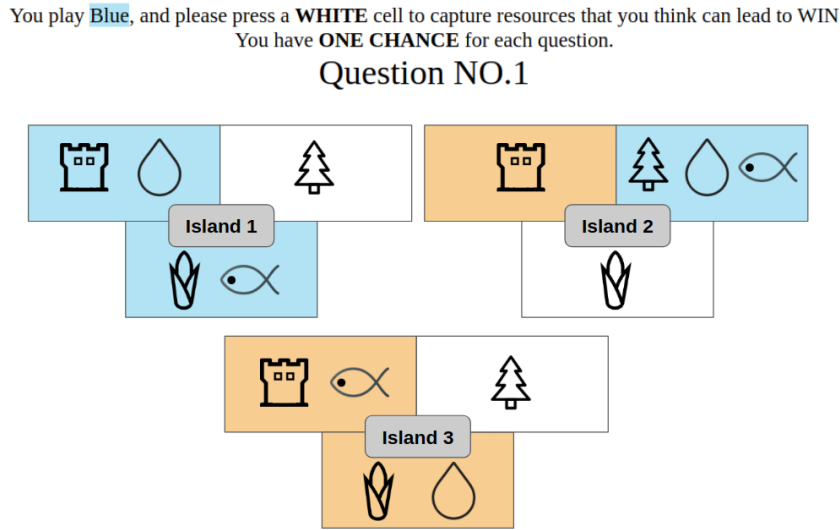


Figure 4.4: Example of pre- and post-training questions for the Island Game. A board is presented to the participant to select a move that he or she thinks is optimal.

their prior experience of Noughts and Crosses which would facilitate problem-solving, which means less optimal play initially and avoidance of a ceiling effect.

Problem setup

The Island Game (Figure 4.4) contains three islands, each with three territories. Each territory or cell has one or more resources. The winning condition is met when a player controls either all territories on one island or three pieces of the same resource. The nine territories resemble the 3x3 cells in Noughts and Crosses and the structure of the original game is maintained in regard to players' turns, possible moves, board states and win conditions. This isomorphism masks a number of spatial relations that represent the membership of a cell to a win condition. In this way, the cells can be rearranged in an arbitrary order without changing the structure of the game.

Textual and visual explanations

An example of the explanations is shown in Figure 4.5 where *MIPlain* provides a set of comments based on the move chosen by the Blue player. Studies [4, 9, 23, 250] suggest explanations are most effective for human learning when they are presented with examples. Therefore, we combine textual explanations which verbalise machine-learned logic programs for a sequence of

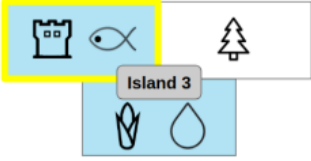
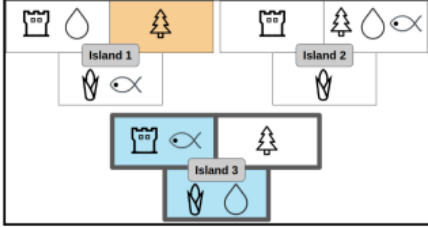
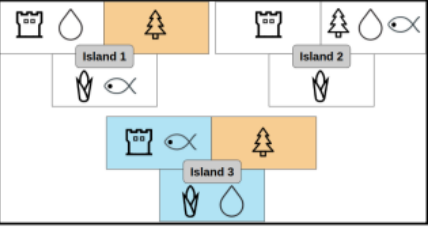
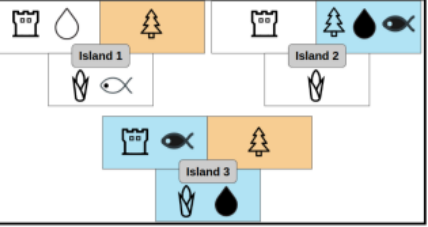
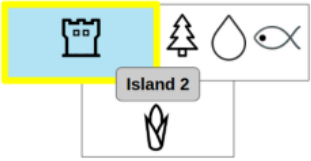
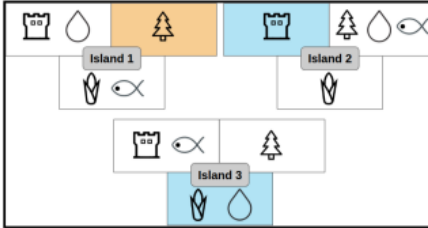
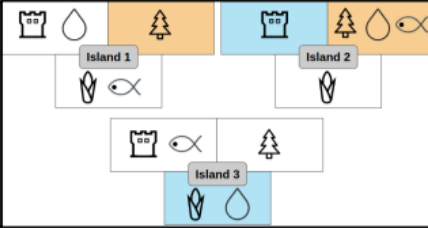
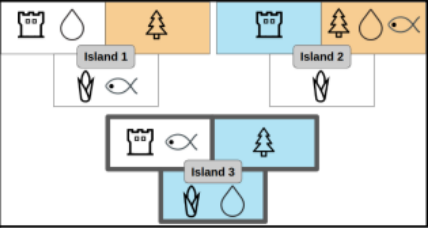
Moves	MIPlain's comments		
 This is a right move	 You select this territory and obtain 1 pair (Island 3)	 Opponent conquers and prevent you from getting a triplet (Island 3)	 You obtain 2 pairs (Water, Fish) and opponent has no pair
 This is a wrong move	 Contrast: Not enough pair(s)	 Contrast: Not enough pair(s)	 Contrast: Not enough pair(s)

Figure 4.5: Explanations generated based on logic programs learned by MIPlain. Both visual and textual explanations preserve the structure of hypotheses without redundancy and account for the reasons that make a move correct (highlighted in green). Explanations are presented for the possible sequence of wrong moves in participant's turns (highlighted in red) by comparing against *MIPlain*'s learned theory.

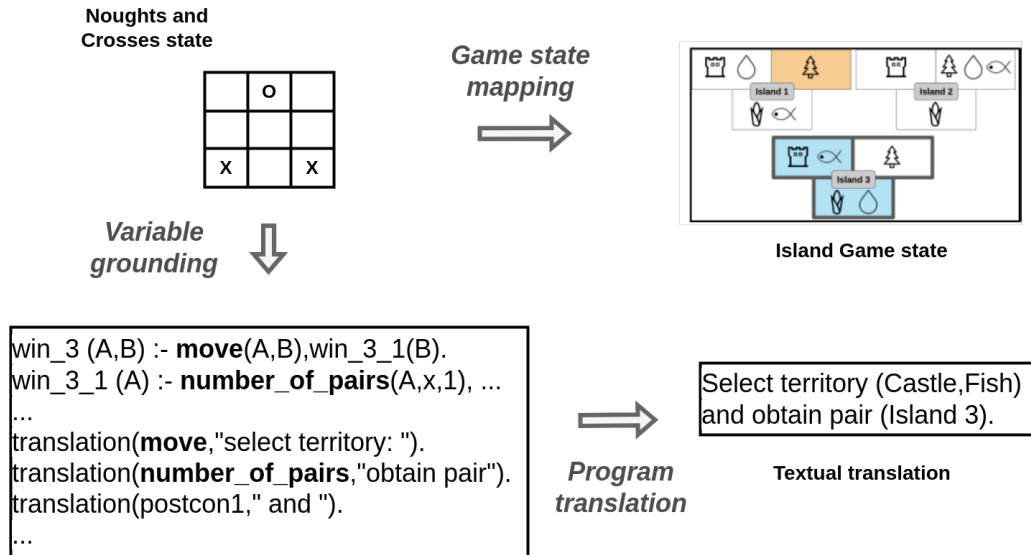


Figure 4.6: Generation of textual and visual explanations. Each Noughts and Crosses canonical state is converted into an Island Game state. A learned logic program is translated by using the Noughts and Crosses state and mapped natural language texts.

game states with visualisations of game boards. These textual explanations and illustrations are generated from grounded programs based on a selection of game states.

Textual and visual explanations are created based on the first-order programs *win_1*, *win_2* and *win_3* learned by *MIPlain* (Table 4.2). The generation³ of textual and visual is illustrated by Figure 4.6. Both textual and visual explanations consist of two main components. Two series of game states compare the consequences between an optimal move and a sub-optimal move by the Blue player (which is played by human participants). For the optimal move, we defined a mapping from each predicate to a short natural language text. The free variables in *MIPlain*'s learned programs are grounded with game states that have been selected as training examples. Term unifications are recorded during the execution of the logic programs. Since the Island Game is isomorphic to Noughts and Crosses, we can generate Island Game states from instantiated terms based on a mapping of states from one game to the other. The textual translation of a clause is the concatenation of natural language templates connected by conjunctions mapped from meta-rules. A logic program consisting of multiple clauses is translated by combining a sequence of clausal translations with connectives. All visual explanations are created from the intermediate terms and their equivalent Island Game states. Visual explanations are organised in accordance with the textual translations by highlighting players' moves and the satisfaction

³A demonstration of an interactive generation of textual explanation is provided at <https://github.com/LAi1997/MIPlain>

of the primitive *number_of_pairs/3* on respective Island Game boards.

The game state sequence with red text colour corresponds to the Blue player’s sub-optimal move. We are able to expand the sub-optimal state sequence based on the assumption that the opponent always plays optimal moves. Consequences of the sub-optimal decision-making have been highlighted according to predicates from the machine-learned programs that have not been satisfied. The opposing Orange player plays optimally from the position, and the best option Blue is presented following Orange’s move. Game states where Blue has to move are represented in Noughts and Crosses and substituted into *MIPlain*’s learned programs. Unsatisfied predicates are also mapped into natural language texts and the respective game state illustrations.

For the selected set of training examples, we generated textual and visual explanations prior to initiating the experimental procedure. Textual explanations were tuned with minimal manual adjustments for correctness and readability. A brief introduction with illustrative examples was given at the beginning of the experiment to explain concepts used in visual and textual explanations. Therefore, with the assumption that participants should not be familiar with the designed Island Game domain, they have been given appropriate information to understand explanations.

4.4.2 Methods and design

Table 4.5: Summary of experiment design. Participants played one mock game against a random computer player for the more difficult Island Game. After selecting a move in training and regardless of its correctness, participants received the labels of the two moves presented; treated participants additionally received explanations generated from *MIPlain*’s learned program. We introduced the primitive set used by *MIPlain*.

Stage	Participant’s assignment	No.	Question format
Intro	Understand rules to move and win	1	practice
Pre-training	Choose the optimal move	15	five canonical positions for each of win_1, win_2 & win_3
Training	Understand the concept of pairs; choose the optimal move and reflect on the choice	9	two choices for each of win_1, win_2 & win_3; presentation of the labels
Post-training tests	Choose the optimal move	15	five canonical positions for each of win_1, win_2 & win_3; rotated and flipped from pre-training questions.
Open questions	Describe the strategy of a previously made move	6	Questions requiring a textual answer
Survey	Provide gender, age group & education level	3	multiple choices

In our experiment⁴ for the Island Game, a human player always played as player one (X for Noughts and Crosses and Blue for the Island Game) who started the game. We adopted a two-group pre-training post-training design (Table 4.5 and Figure 4.7). We first introduced participants to the rules of the Island Game. In the pre-training stage, the performance of participants in both self-learning and machine-aided learning groups is measured identically. In the training stage, we introduced the concept of pairs and asked participants to try to classify pairs with some working examples. Participants could see correct answers. The difference between self-learning and machine-aided human task performance was assessed using an independent two-sample t-test. In the post-training, the performance of both self-learning and

⁴No ethics approval was required by the Machine Learning Journal at the time when the paper was submitted.

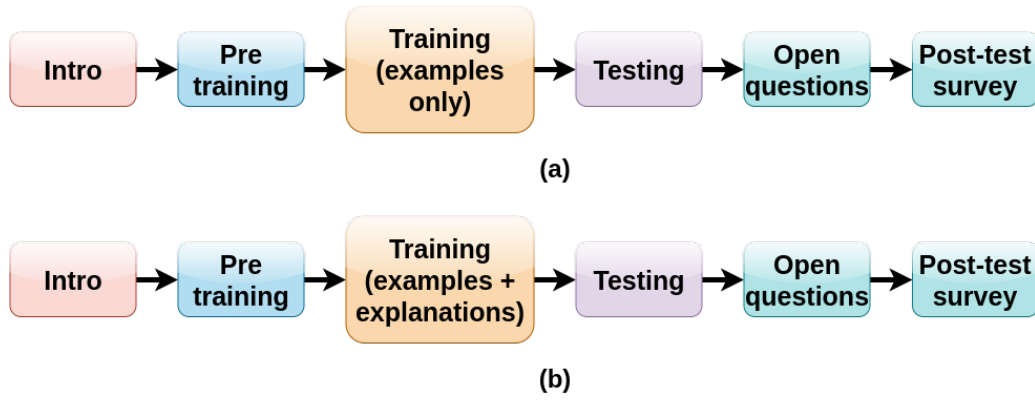


Figure 4.7: Visualisation of experiment design. Design (a) represents the experimental procedure for human self-learning where participants only learn from examples. Machine-aided learning corresponds to design (b) where participants learn from both examples and explanations.

machine-aided groups was evaluated in the same way as in the pre-training.

Each question in pre- and post-training tests was the participant's turn to play. They were asked to select what they considered to be the optimal move. A question category of win_k denotes a game position winnable in k moves of the human player and corresponds to a state winnable by a win_k rule. An exemplary question is shown in Figure 4.4. The post-training questions were rotated and flipped from pre-training questions so the complexity of the solution is maintained. In each test, only 15 questions were given to limit the experiment duration to one hour. The response time of participants was recorded in pre-training and post-training tests.

An exemplary question in the training phase for the machine-aided group is shown in Figure 4.8. In the training stage, the self-learning group did not receive the treatment and was presented with the same game positions without the corresponding visual and textual explanations. The treatment of textual and visual explanations was applied to only the machine-aided group.

For the Island Game experiments, we recorded an English description of the strategy they used for each of the selected post-training questions. Participants were presented with previously submitted answers, one at a time along with a text input box for written answers. Moves for these open questions were selected from post-training with a preference order from wrong and hesitant moves to consistently correct moves. We associated hesitant answers with higher response times. A total of six questions were selected based on individual performance during the post-training.

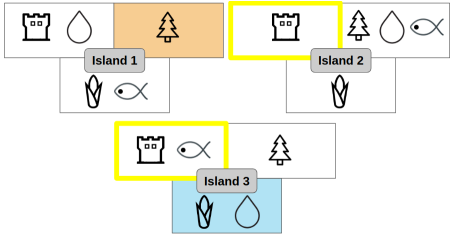
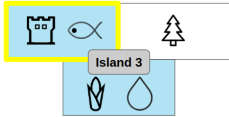
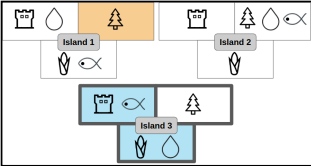
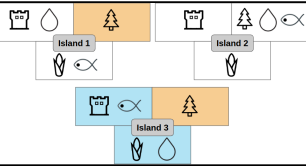
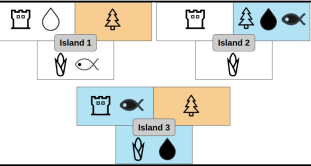
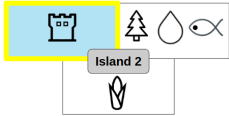
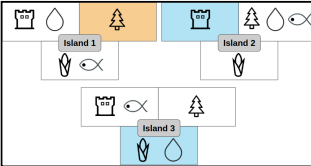
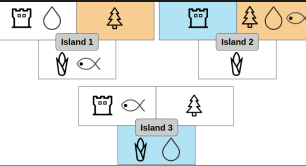
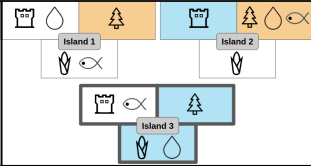
Example	Moves	MIPlain's comments		
	 This is a right move	 You select this territory and obtain 1 pair (Island 3)	 Opponent conquers and prevent you from getting a triplet (Island 3)	 You obtain 2 pairs (Water, Fish) and opponent has no pair
	 This is a wrong move	 Contrast: Not enough pair(s)	 Contrast: Not enough pair(s)	 Contrast: Not enough pair(s)

Figure 4.8: Interface featuring a training example with explanations. Human participants, who play Blue, are confronted with a game position and have to choose between two alternative moves that are highlighted in yellow. When Blue owns two cells on the same island or two pieces of the same resource, related cells or resources are highlighted in bold.

4.5 Results

We conducted three pilot trial experiments using the interface with Noughts and Crosses questions and explanations instead. These experiments were carried out on three samples: an undergraduate student group from Imperial College London, a junior student group from a German middle school and a mixed background group from Amazon Mechanical Turk⁵ (AMT). No consistent explanatory effects were observed for any of the mentioned samples. The problem-solving strategy that humans apply can be affected by factors such as task familiarity, problem difficulty, and motivation. For instance, [265] shows that a rather superficial analogical transfer of a strategy is applied when a problem is too difficult or when there is no reason to gain a more general understanding of a problem. Given that the majority of subjects achieved reasonable initial performance, we ascribe the reason for such results to experience with the game and the complexity of explanations. The game familiarity of adult groups led to less potential for performance improvement. Early middle school students had limited attention and were overwhelmed by information intake. Hence, we focused on specially designed experiment materials, the Island Game, in the follow-up experiments. We present results on task performance and human strategy analysis. Our qualitative analysis of human strategy refers to definitions of textual response quality in Section 4.3.1.

Analysis of human task performance

A sample from Amazon Mechanical Turk and a student sample from the University of Bamberg participated in experiments that used the interface with Island Game questions and explanations. To test hypotheses **H1** to **H5**, we employed a quantitative analysis of test performance and a qualitative analysis of textual answers. A sub-sample with a mediocre performance on pre-training questions of all categories within one standard deviation of the mean was selected for the performance analysis. This aims to discount the ceiling effect (initial performance too high) and outliers (e.g. struggling to use the interface). We evaluate the explanatory effect on human task performance empirically using independent two-sample t-tests. We employed a 5% significance level for testing experimental results.

⁵AMT (www.mturk.com) is an online crowd-sourcing platform used to recruit experiment participants.

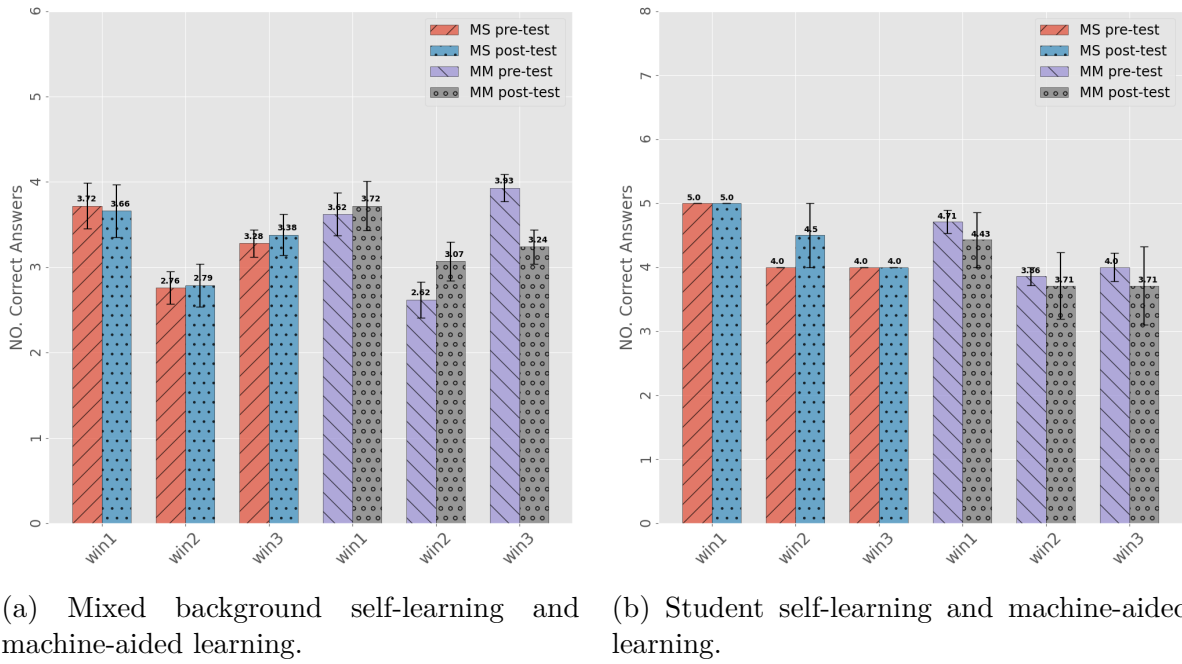


Figure 4.9: The number of correct answers in pre- and post-training in different question categories. Bars show standard errors. We compare the post-training performance of learning from training examples (self-learning) and learning from machine explanations (machine-aided learning) via independent two-sample t-tests. In (a), machine-aided learning led to a higher average human performance in the *win_2* case and a worse average human performance in *win_3* case. In (b), self-learning results on the left have zero standard errors due to a sample size of two. There were no performance changes by machine-aided learning.

It is important to note that while we recognise the potential of alternative experimental design, we can compare the effects of learning with machine explanations against learning from training examples with our two-group approach.

From the AMT sample, we had 90 participants who were 18 to above 65 years old. A sub-sample of 58 participants with a mediocre initial performance was randomly partitioned into two groups, **MS** (Mixed background Self-learning $n = 29$) and **MM** (Mixed background Machine-aided learning, $n = 29$). A different sub-sample of 30 participants completed open questions and was randomly split into two groups, **MSR** (Mixed background Self-learning and strategy Recall, $n = 15$) and **MMR** (Mixed background Machine-aided learning and strategy Recall, $n = 15$).

In Figure 4.9a, in category *win_2*, **MM** post-training had a better comprehension ($p = .028 < .05$) than **MS** post-training while **MM** and **MS** had similar pre-training performance in this category. Results in category *win_2* indicate that explanations have a beneficial effect on **MM**. However, **MM** did not have a better comprehension of *win_1* than **MS** given the same

Table 4.6: The number and accuracy of HQ and LQ responses for groups **MSR**, **MMR**, **SSR**, **SMR** and each question category. For *win_3*, the most mentally challenging category of all three, no HQ response was given.

		<i>win_1</i>	<i>win_2</i>	<i>win_3</i>
MSR	No. HQ / post-train accuracy	9 / .889	1 / 1.00	-
	No. LQ / post-train accuracy	19 / .421	32 / .406	29 / .517
MMR	No. HQ / post-train accuracy	8 / 1.00	2 / 1.00	-
	No. LQ / post-train accuracy	16 / .250	35 / .486	29 / .483
SSR	No. HQ / post-train accuracy	6 / 1.00	1 / 1.00	-
	No. LQ / post-train accuracy	0 / .000	8 / .750	9 / .667
SMR	No. HQ / post-train accuracy	9 / 1.00	9 / .778	-
	No. LQ / post-train accuracy	3 / .000	14 / .571	19 / .737

initial performance. In addition, **MM**'s performance reduced ($p = .005 < .05$) after receiving explanations of *win_3*.

A sample of students involved in a Cognitive Systems course at the University of Bamberg was also examined. We had 13 participants who were 18 to 24 years old and a few outliers between 25 and 54 years. All participants were asked to complete open questions and were randomly split into two groups, **SSR** (Student Self-learning and strategy Recall, $n = 4$) and **SMR** (Student Machine-aided learning and strategy Recall, $n = 9$). In Figure 4.9b, a sub-sample of 9 with a mediocre initial performance was randomly divided into **SS** (Student Self-learning, $n = 2$) and **SM** (Student Machine-aided learning, $n = 7$). The imbalance in the student sample was caused by some participants leaving during the experiment. The machine-aided learning results show large performance variances in post-training as indirect evidence of performance changes. Further experimentation could provide clearer evidence.

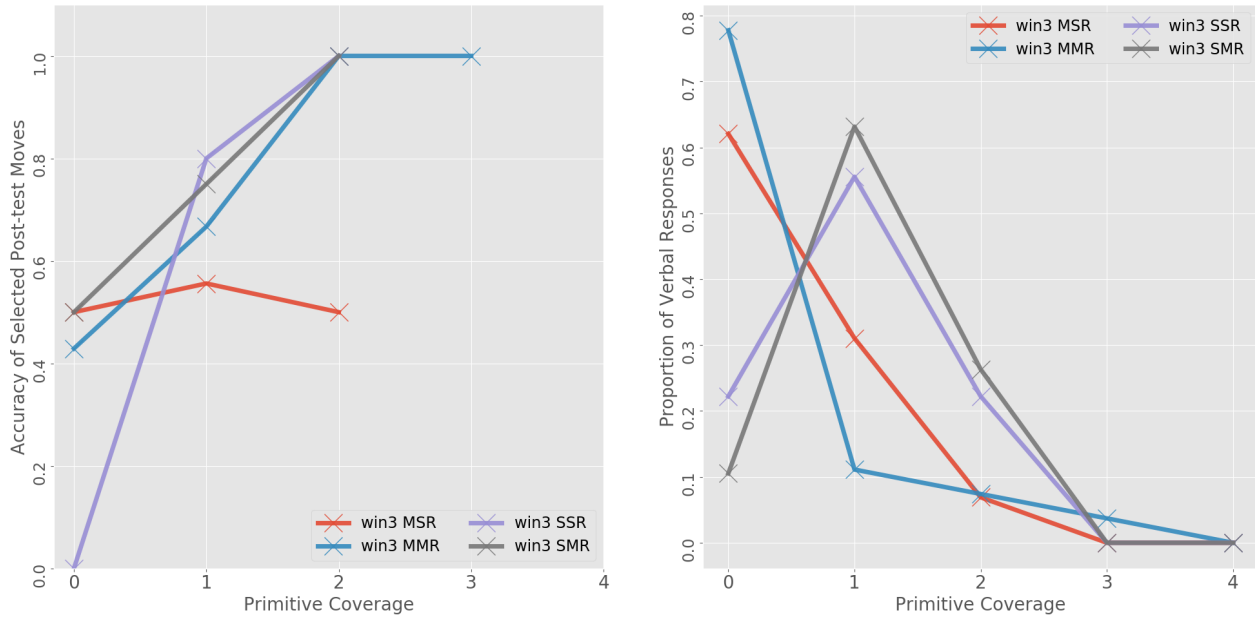
Analysis of human Island Game strategy

In Table 4.6, we identified that participants who were able to provide high-quality responses for their test answers scored higher on these questions. This is not the case for *win_3*, however, due to the high difficulty of providing a good description of the strategy for the *win_3* category. Additionally, in the *win_2* category, both machine-aided groups (**MMR**: $\frac{2}{(2+35)}$, **SMR**: $\frac{9}{(9+14)}$) have greater proportions of high-quality responses than self-learning groups (**MSR**: $\frac{1}{(1+32)}$, **SSR**: $\frac{1}{(1+8)}$). Also, we observed a pattern in which there are fewer HQ responses than LQ responses in the *win_1* and *win_2* categories. This pattern is more significant in the *win_2* category.

Figure 4.10 illustrates the difficulty of providing a good quality textual answer for the non-trivial category *win_3*. Since *win_1* and *win_2* contain fewer predicates, we examine primitive coverage of non-trivial category *win_3*. When counting primitives based on Definition 4.1, we only consider the constraint *number_of_pairs*/3 and ignore the move generator *move*/2 as participants were required to make a move when they answered a question.

In Figure 4.10a, we plotted primitive coverage against the accuracy of post-training answers that were selected as open questions. We observed a major *monotonically increasing trend* in accuracy with respect to primitive coverage. This indicates that high matching between textual answers and the machine-learned theory correlates with high performance. In Figure 4.10b, we observed *downward curves* for **MSR** and **MMR** in the number of textual answers from the lower to the higher primitive coverage. More responses were provided by **SSR** and **SMR** covering *one primitive* than **MSR** and **MMR**. Participants gave very few responses that cover *more than two* primitives. Based on the learned theory⁶ of *MIPlain* in Table 4.2, the results suggest an increasing difficulty to provide complete strategy descriptions of *win_3* beyond two clauses for mixed background groups and four clauses for student groups.

⁶The translation of the learned theory into textual and visual explanations does not contain redundant parts.



(a) The accuracy of textual answers increases with respect to the number of primitives covered. (b) The proportion of quality textual answers decreases with the number of primitives covered.

Figure 4.10: *win_3* reuses *win_2* and uses four *number_of_pairs/3* when unfolded. Figure 4.10a also shows that student groups could not provide a significant number of responses covering more than two of these predicates respectively. In Figure 4.10b, both mixed background groups (MSR and MMR) had lower proportions of responses covering one predicate than student groups (SSR and SMR).

4.6 Discussion

Results concerning **H1** to **H5** are summarised in Table 4.7 and 4.8. We assume that (**H1** Null) comprehension does not correlate with textual answer quality. To examine this hypothesis, we analyse the results in two steps. First, the results of HQ responses in two categories (Table 4.6) suggest that being able to provide better textual answers corresponds to a high comprehension. Second, we examined the coverage of primitives (specifically for LQ responses of *win_3*) in textual answers (Figure 4.10a). Evidence in all categories shows a correlation between comprehension and the degree of textual answer matching with explanations. We reject the null hypothesis in all categories which supports **H1**.

In addition, we assume that (**H2** Null) the difficulty for human participants to provide a textual answer is not affected by textual answer quality. Since high response quality is difficult to achieve (Table 4.6) and it is challenging to correctly describe all primitives (Figure 4.10b), we reject this null hypothesis for all categories and **H2** is supported as it is increasingly difficult for participants to provide higher quality textual answers. Hence, two additional trends we

Table 4.7: Hypotheses concerning the quality of textual answers and comprehension. S stands for supported, N denotes not supported, H stands for hypothesis. Test outcomes are presented for *win_1*, *win_2* and *win_3* categories.

H	<i>win_1</i>	<i>win_2</i>	<i>win_3</i>
H1 Human comprehensions manifest in textual answer quality	S	S	S
H2 Difficulty for human participants to provide textual answers increases with textual answer quality	S	S	S
H3 machine-learned theory improves textual answer quality	N	S	N

Table 4.8: Hypotheses concerning the cognitive window and explanatory effects. S stands for supported, H stands for hypothesis, T stands for test outcome.

H	T
H4 Learning a complex theory exceeding the cognitive bound leads to a harmful explanatory effect	s
H5 Applying a learned theory without a low cognitive cost does not lead to a beneficial explanatory effect	s

observed from the same figure suggest two mental barriers to learning. As we assume a human sample is a collection of version space learners, the search space of participants is limited to programs of size *two* (mixed background groups) and *four* (student groups). When H is taken as the student sample and P is the machine-learned theory on winning the Island Game, the cognitive bound $B(\text{win_3}, H) = m^4 * p^{4(j+1)} = 4^4 * 2^{12}$ corresponds to the smallest hypothesis space size for programs with four clauses (four meta-rules are used with at most two body literals in each clause and two primitives *move/2* and *number_of_pairs/3*). This supports the existence of a bound on the hypothesis space size of human learning (Definition 3.15).

Regarding **H3**, we assume that (**H3** Null) machine-learned theory does not improve textual answer quality. Results (Table 4.6) show a higher proportion of HQ responses for machine-aided learning than self-learning in category *win_2*. Thus, for *win_2*, we reject this null hypothesis which means **H3** is supported in category *win_2* where the explanations result in more high-quality textual answers being provided.

Here, we examine Hypothesis 3.16 of a cognitive window. We assume that (**H4** Null) learn-

ing a descriptively complex theory does not affect comprehension harmfully. When P is the program learned by *MIPlain*, $B(P, H)$ bounds for the two samples correspond to the program class with a size no larger than 4. Only *win_3* which has a larger size of seven after unfolding exceeds these cognitive bounds. As harmful effects (Figure 4.9a and 4.9b) have been observed in category *win_3*, this null hypothesis is rejected and **H4** is supported as learning a complex machine-learned theory has a harmful effect on comprehension. We also assume that (**H5** Null) applying a theory without a sufficiently low cognitive cost has a beneficial effect on comprehension. According to Remark 4.2, given sufficient training examples E , *MIGO*'s learning algorithm as $\bar{M}$ and $\phi = \{\text{move}/2, \text{won}/1, \text{number_of_pairs}/3\}$, the predicate *win_1* in *MIPlain*'s learned theory does not have a lower cognitive cost: for all queries x of winning in one move, $\text{Cog}(\text{win}_1, x) \geq \text{CogP}(E, \bar{M}, \phi, x)$. We reject this null hypothesis since no significant beneficial effect has been observed in category *win_1*. Therefore, **H5** is supported and knowledge application that requires cognitive resources does not result in better comprehension.

The performance analysis (Figure 4.9a) demonstrates a comprehension difference between self-learning and machine-aided learning in category *win_2*. While a larger sample size would likely ensure consistency of statistical evidence, the patterns in results suggest more significant results in category *win_2* than *win_1* and *win_3*. The predicate *win_2* in the program learned by *MIPlain* satisfies both constraints on hypothesis space bound for knowledge acquisition and cognitive cost for knowledge application. In addition, our cognitive window theory explains the lack of beneficial effects of predicates *win_1* and *win_3*. The former does not have a lower cognitive cost for execution so operational errors cannot be reduced and there have been no observable effects. The latter is a complex rule with a larger hypothesis space for human participants to search from. Harmful effects have been observed due to partial knowledge being learned.

4.7 Summary

The effects of machine-learned explanations have been investigated in the context of game strategy learning. In addition, we have explored both beneficial and harmful perspectives of AI's explanatory effects on human comprehension. The Island Game is an isomorphic game

design based on Noughts and Crosses. We demonstrate that explanations generated from a machine-learned logic program can enhance human comprehension. To raise awareness of the harmfulness of machine explanations, we also contribute by highlighting results that show logic programs, which have been believed to facilitate human understanding, can negatively affect human comprehension. The presented empirical results support **Sub-thesis 1** and Hypothesis 3.16 of a cognitive window. Given problems whose solutions can be effectively divided into sufficiently small parts, a potential approach to improve the explanatory effectiveness of a machine-learned theory is to divide complex concepts into “chunks”. Each “chunk” of information involves simple-to-execute and short sub-goal explanations. In the next chapter, we investigate the impact of sequential teaching curricula composed of sessions with increasingly difficult tasks and explanations.

Chapter 5

Explanatory machine learning for sequential human teaching

We explain experiments and empirical results supporting **Sub-thesis 1 and 2** from Chapter 1 and Hypothesis 3.17 of sequential teaching curriculum improvement. We include parts from our published Machine Learning Journal paper [2].

5.1 Introduction

In the previous chapter, our empirical results demonstrate that machine-learned explanations are beneficial only if they have the appropriate complexity. To achieve a beneficial effect, logic programs used to generate explanations should not overwhelm human learners with too much information. In addition, logic programs should not put more cognitive loads on working memory than problem solutions that people can develop. These results imply that complex machine-learned logic programs with too much low-level information can be difficult to explain effectively. To address this, this chapter explores sequential curricula to arrange teaching materials into smaller blocks to accommodate the constraints of human learning. Prior research on comprehensibility did not consider human learning from acquired knowledge and the dependency between teaching materials. We apply the E_{ex} framework to account for the impacts of sequential teaching on comprehension.

Human knowledge attainment is considered incremental [37, 318, 103]. Humans are capable of establishing the connections between “chunks” of knowledge [14] and exploiting the mapping from a new problem-solving context to prior knowledge [87]. The dependency between concepts and the order of learning may affect human learning. For example, in the context of teaching algorithms to human students, we may imagine a human teacher using the material in Figure 5.1 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.

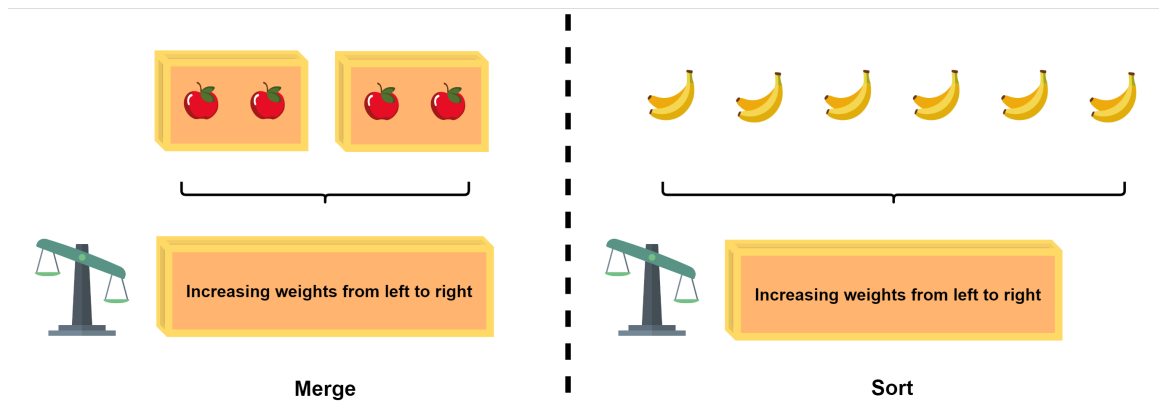


Figure 5.1: Illustration of teaching materials of merging and sorting that can be learned in incremental order (left to right). All fruits have different weights which can be pairwise compared using the balance scale at the corner. On the left-hand side, two sets of apples should be merged into a larger 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.

We conducted a human trial to examine the effects of 1) concept ordering in a sequential teaching curriculum and 2) explanations generated from machine-learned logic theory for teaching efficient sorting strategies. Developing effective sorting approaches requires explicit reasoning [311] which allows us to investigate human inferential ability when building from learned knowledge. In Hypothesis 3.17 of sequential teaching curriculum improvement, we hypothesise that a sequential teaching curriculum can enhance human comprehension when it reduces the size of the hypothesis space associated with the target concept. Our empirical results show that

a sequential teaching curriculum with increasing concept complexity has a beneficial effect on human comprehension. This supports **Sub-thesis 2** and Hypothesis 3.17.

Results indicate that human novices re-discover the known divide-and-conquer algorithms after learning from computational concepts with increasing complexity. We provide evidence of the optimisation of problem-solving strategy due to human studying explanations generated from machine-learned logic rules. When explanations are presented in an incremental teaching curriculum, the most applied strategy correlates to improved task performance. This evidence supports **Sub-thesis 1 and 2**. We observe that explaining a machine-learned logic program without a sufficiently low cognitive cost does not lead to a beneficial effect on human comprehension. This supports Hypothesis 3.16 of a cognitive window.

This chapter is organised as follows. In Section 5.2, we describe modifications to an existing MIL system, *Metagol_O*, for learning a logic program that represents a variant of the merge-sort algorithm. The logic program is divided into two parts, merging and sorting, which correspond to the two problem-solving tasks in our sequential teaching curricula. In Section 5.3, we elaborate on the evaluation of human sorting performance and sorting strategy. Then, we present a set of experimental hypotheses and explain the motivations of these hypotheses. In Section 5.4, we introduce the MaRs-IB test which we use to control participants' cognitive abilities and describe the design of the experimental procedure and materials. In Section 5.5, we show experimental results and explain how they support the experimental hypotheses in Section 5.6.

5.2 Meta-interpretive learning of sorting algorithm

The MIL system *Metagol* [75] 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 [70]. *Metagol_O* [74] makes use of composite objects and actions defined from primitive objects and actions. *Metagol_O* addresses the issue that most ILP systems only consider 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

Table 5.1: Logic programs learned by *Metagol_O*. 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. *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*.

Definition	Rules
merger/2	<pre> 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). </pre>
sorter/2 (after learning merger/2)	<pre> 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). </pre>
sorter/2 (without learning merger/2)	<pre> 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). </pre>

and resource costs of hypothesis programs are minimised by *Metagol_O* using iterative descent [74] 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*. *Metagol_O* converges to hypotheses with the optimal resource complexity and learns resource-efficient robot strategies [74].

We modify the background knowledge of *Metagol_O* for learning a bottom-up variant of the merge-sort algorithm [109] 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. The aforementioned variant takes an input sequence as a list of sub-sequences of

Table 5.2: Meta-rules used by *Metagol_O*. *Metagol_O* uses two meta-rules. 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.

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

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 5.1. To learn the bottom-up variant of merge-sort¹, we define the world state that the robot sorter would see. We define *Lt_expr* which represents a non-empty monotonically increasing sequence of integers $Lt_expr := Int \mid Lt_expr < Lt_expr$. *Metagol_O* is provided with primitives to manipulate expressions which compactly represent ordered sequences.

Example 5.1. *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

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

objects. In particular, a state is defined as a tuple $(exprs, energy, left_bag, right_bag, memory)$ with an arity of 5.

Example 5.2. 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 provide examples of sequential curriculum rank functions based on Definition 3.5. 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 $\lambda_{sort/merge}(sorter, E_{sort}, Metagol_O) = 0$. The merge-then-sort curriculum is represented by $STC(DS, ES, \lambda_{merge/sort}, MS)$ which teaches merging before teaching sorting and the sort-then-merge curriculum is denoted by $STC(DS, ES, \lambda_{sort/merge}, MS)$ which teaches sorting before teaching merging. $\lambda_{merge/sort}$ specifies the ordering of concepts in the merge-then-sort curriculum $STC(DS, ES, \lambda_{merge/sort}, MS)$ and $\lambda_{sort/merge}$ defines the ordering of concepts in the sort-then-merge curriculum $STC(DS, ES, \lambda_{sort/merge}, MS)$.

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 a finite 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

$$A_{comp} = \left\{ \begin{array}{l} parse_exprs/2, compare_nums/2, \\ single_expr/2, recycle_memory/2 \\ drop_bag_remaining/2 \end{array} \right\}$$

for learning $merger/2$ and $sorter/2$ in Table 5.1. In [75], the authors described meta-rules *Chain* and *Tailrec* which are presented in Table 5.2 that allow *Metagol_O* to learn recursive programs. 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*.

Remark 5.1. *Metagol_O is provided with A_{comp} to learn *merger/2* in Table 5.1. Given action definitions $A = A_{prim} \cup A_{comp}$, since composite actions A_{comp} are constructed from multiple primitive actions from A_{prim} , the subset of A_{comp} in *merger/2* is necessary and sufficient 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 $x \in X$.*

In *merger/2*, two *Lt_expr* from *exprs* are first parsed and put 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* expressions in *memory* are then recycled back to *exprs* by *recycle_memory/2*. This process is iterated until there is only one *Lt_expr* left in *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 [283] ensures the expressions in the current state are not less sorted compared with expressions in the immediate next state.

5.3 Experimental framework

For the tasks of merging and sorting, human comprehension corresponds to the competence of participants to predict the ordering of the items. In this section, we explain how we evaluate human sorting accuracy. Human sorting traces are also compared with machine sorting traces to assess the similarity between human and machine sorting strategies. We provide details of the implementation of this comparison.

5.3.1 Evaluating human comprehension of sorting

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 a quantitative 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 [283] 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 misremembered 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 $[0, 1]$. 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.

Definition 5.1. (*Normalised performance score, $\text{perf}(R_h, R_m)$*) 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

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

Example 5.3. 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 $perf(R_{h1}, R_m) = .386 < perf(R_{h2}, R_m) = .657$.

Based on results from multiple trials, the discounting constant c is set to .5 to moderately penalise responses that involve reversed sorted sequences.

5.3.2 Experimental hypotheses

We present the experimental hypotheses with descriptions of our intention in this section. Table 5.3 summarises the experimental variables and treatments. Explanations are not provided for learning sorting to avoid introducing merge prematurely in the curricula that learn sorting before learning merging.

The effect of the curriculum order (**CO**) is examined 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**). We use the normalised performance score $perf$ in Definition 5.1 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

Table 5.3: We consider two independent experimental variables. 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.

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	—

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 $\emptyset$ represents no explanations. 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 are defined as $\lambda_{SM/WEX}$ and $\lambda_{SM/WOEX}$.

The definitions of the rank functions are summarised in Table 5.4. Let $C_{MS/WEX}$ denote the human comprehension of the sequential teaching curriculum that learns merging before learning sorting with explanations of merging. Let $C_{MS/WOEX}$ denote the human comprehension of the sequential teaching curriculum that learns merging before learning sorting without explanations of merging. Let $C_{SM/WEX}$ denote the human comprehension of the sequential teaching curriculum that learns sorting before learning merging with explanations of merging. Let $C_{SM/WOEX}$ denote the human comprehension of the sequential teaching curriculum that learns sorting before learning merging without explanations of merging. We focus on the effect of curriculum order and explanations generated from machine-learned rules on human learning of sorting. In particular, **H1** to **H3** examine the effect of the independent experimental variables **CO** and **EX** on human comprehension. We also assess their relation to Hypothesis 3.16 and 3.17. We use the E_{ex} framework and formulation of sequential teaching curricula (Definition 3.7 to 3.9) to examine the following experimental hypotheses.

Table 5.4: Definitions of the rank functions. In curricula **MS/WEX** and **MS/WOEX**, humans learn merging before learning sorting. 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*.

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	$\lambda_{SM/WOEX}(merger, E_{merge}, \emptyset) = 1$ $\lambda_{SM/WOEX}(sorter, E_{sort}, \emptyset) = 0$

H1: *Learning merging before learning sorting leads to a beneficial effect on human comprehension of sorting with respect to learning sorting before learning merging.*

In **H1**, we examine whether human learners can achieve a 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 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 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.

The hypothesis **H4** focuses on the effect of explanations and relates to Definition 3.1 to 3.4 and Hypothesis 3.16. We suspect that for human learners who have no previous programming experience, learning merging with the presence of explanations can result 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 3.4 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 does not result in a beneficial explanatory effect on human comprehension of merging.*

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

H5: *Learning merging with explanations (**WEX**) 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 Section 5.5, 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.

5.4 Experiments

We have received help from our collaborators at the University of Bamberg in designing and conducting the experiments. My responsibilities were interface implementation², result collection and analysis. We adopt the MaRs-IB test at the beginning of the experiment as a control of participants' cognitive ability. This approach is an alternative to using experimental materials to establish performance benchmarks and does not pre-expose the problem domain. We design training and testing questions based on problems of sorting natural numbers. The numbers are hidden and their ordering can only be accessed via a particular comparison device that we provide. We employ a 2x2 experimental design to examine the effects of 1) curriculum order and 2) machine-learned explanations.

5.4.1 MaRs-IB pre-test

We 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 participant's 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.

²The experiment interface is available on <https://github.com/LAi1997/sequential-teaching>. The interface was created using PsychoPy [234], an open-source package for implementing free interfaces with stimulus presentation and control in Python and JavaScript.

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 [16] or similar material. We argue that using the experimental task will undermine the sequential nature of our experiment, as either, groups 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 [58], 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 [58]. Though there is no normalized score which would be comparable to the IQ, the MaRs-IB test can serve as a measure of

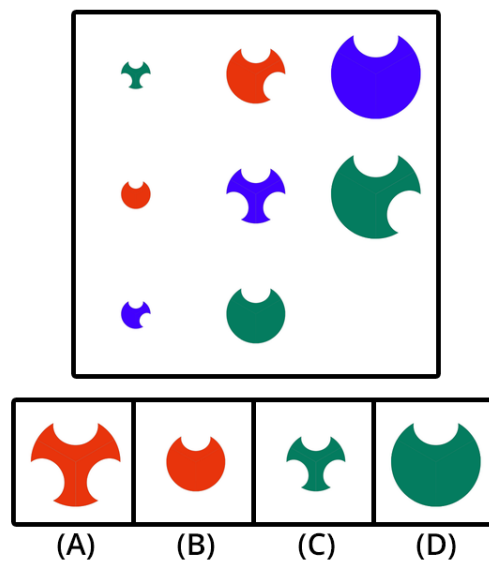


Figure 5.2: Each MaRs-IB item consists of an incomplete 3×3 matrix in which 8 cells with shaped icons depict a modification rule to (an) abstract shape(s). The test asks the participant to fill in the missing piece at the right bottom corner of the 3×3 matrix by choosing one of four provided options (A, B, C and D). The correct answer to this example is A.

the relative cognitive ability of participants. Another concern regards the open access to the material [58] and familiarity with the test from other studies. This risk can be mitigated by randomised arrangement and selection of items from the multiple test sets provided by the authors.

Application

The authors [58] provided three sets of 80 items (Figure 5.2). Each of these sets uses different symbols and is available in three further variations to compensate for colour-vision-deficient participants. The testing procedure described in the validation paper [58] will be consistently completed in under 10 minutes (8 minutes 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 complete three of them correctly. An item in the full test starts with a 500 milliseconds fixation cross, followed by a 100 milliseconds white screen mask [58]. The matrix will then be displayed for 30 seconds, or until the participant responds (whichever happens earlier). After 25 seconds a clock indicates that participants have 5 seconds left for this specific item [58]. Participants complete these items for a total of 8 minutes. The accuracy with which the participant chooses the correct missing pattern in the given time frame corresponds to the pre-test performance.

5.4.2 Materials

Human participants may be quite competent with the task of sorting up to a certain input size. 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.

Problem masking

We designed an isomorphic problem that represented integer sequences by the weights of fruits in piles. This prevents participants from relying on the total ordering of natural numbers. We

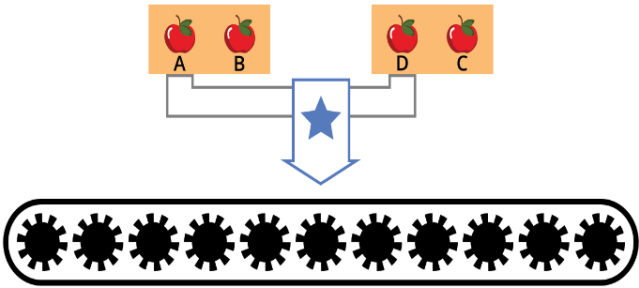
COMPARE weights by typing fruit labels in both LHS and RHS textboxes

Progress 16%

COMPARE



Compare



1. Use the scale on the left to COMPARE weights of TWO fruits by entering the alphabetic CAPITAL labels

2. In EACH ORANGE box, fruits are arranged in INCREASING weights from LEFT to RIGHT

3. Fruits on the CONVEYOR BELT are arranged in INCREASING weights from LEFT to RIGHT

You have 90 SECS to SUBMIT!

Please SELECT the CONVEYOR BELT that has the correct fruit(s) on YELLOW position(s):





Submit

Submit

Figure 5.3: Participants are 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. Participants are taken to the next page where feedback and explanations are presented (Figure 5.4).

Progress 18%

You answer is WRONG!

Initial state

A
 D
 C
 B

This answer is CORRECT!

A
 D
 B
 C

SELECTED >>>
 This answer is WRONG!

Read the feedback and continue
whenever you are ready (60 SECS)

Item C is lighter than item B; append item C

A
 D
 C

Append remaining item(s): B

A
 D
 C
 B

Item C is lighter than item B SO item C should be appended

A
 D
 B

Continue

Figure 5.4: Explanations of merging is shown on this page. 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.

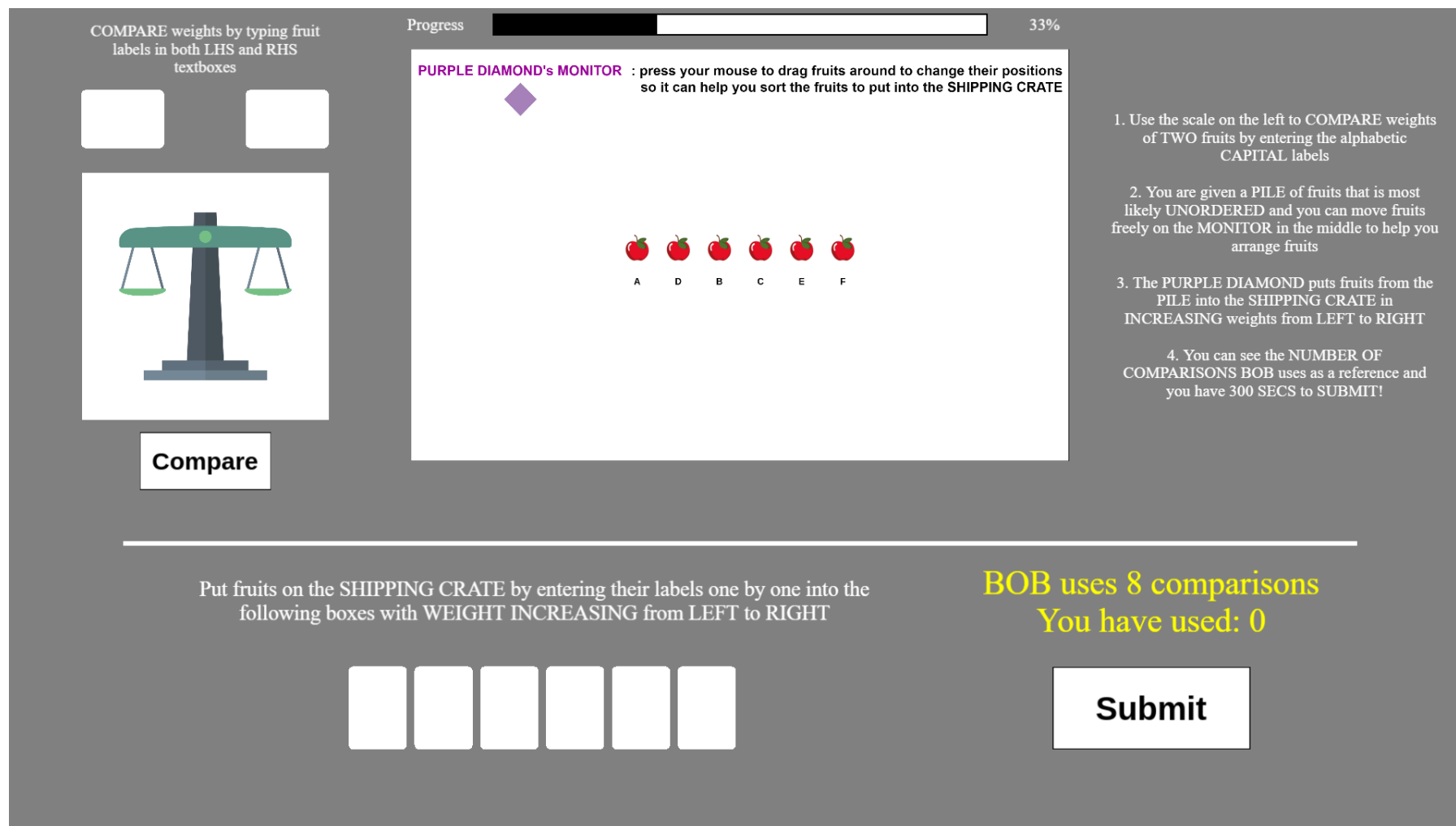


Figure 5.5: Sort training interface. In sort training, participants were advised to sort sequences of fruits (in this case fruits A, D, B, C, E and F) in the centre white box called the “whiteboard” and provided instructions to efficiently use it. The “whiteboard” was a device of the interface which 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.

refer to this problem as the Fruit Weighing problem. Numbers were masked by visual icons of fruits and the numeric values were hidden from participants. A balance scale (Figure 5.3) 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 [279], the new Fruit Weighing problem is 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 consciously recognized [104, 125, 249]. 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 involves two sub-sequences of similar sizes 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.

Problem setup

To make the experiment more engaging, we introduced the background for the Fruit Weighing problem 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 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.

In order 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 Figure 5.3, 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 (Figure 5.6) allows comparison of 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. 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

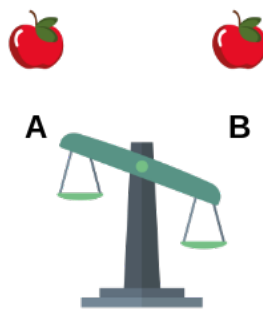


Figure 5.6: The balance on the interface (illustrated by Figure 5.3, 5.4 and 5.5) has two white boxes on two sides where fruit labels can be entered. It would tilt to the side of the fruit that is heavier similar to a physical balance.

select from one of the two output answer choices that had the fruits in the correct order.

Textual and visual explanations

Metagol_O uses predicates with composite objects as arguments to describe the manipulation of *Lt_expr* expressions. A *Lt_expr* expression involving n integers and $n - 1$ “<” symbols is equivalent to a list of size n that contains n integers in increasing order. Given an input list of integers, *Metagol_O* learns *merger/2* and *sorter/2* (Table 5.1) that correspond to a bottom-up variant of the merge-sort algorithm. Both *merger/2* and *sorter/2* take an input set of *Lt_expr* expressions and return a set of a single *Lt_expr* expression. *sorter/2* uses *merger/2* to produce solutions to the Fruit Weighing problem due to its isomorphism to integer sorting. The *sorter/2* program outputs a set that contains a single *Lt_expr* expression which represents a sorted integer sequence.

A part of the experimental interface that contains explanations is presented in Figure 5.4. Given two piles of fruits with increasing weights (A and B, D and C), explanations show whether B

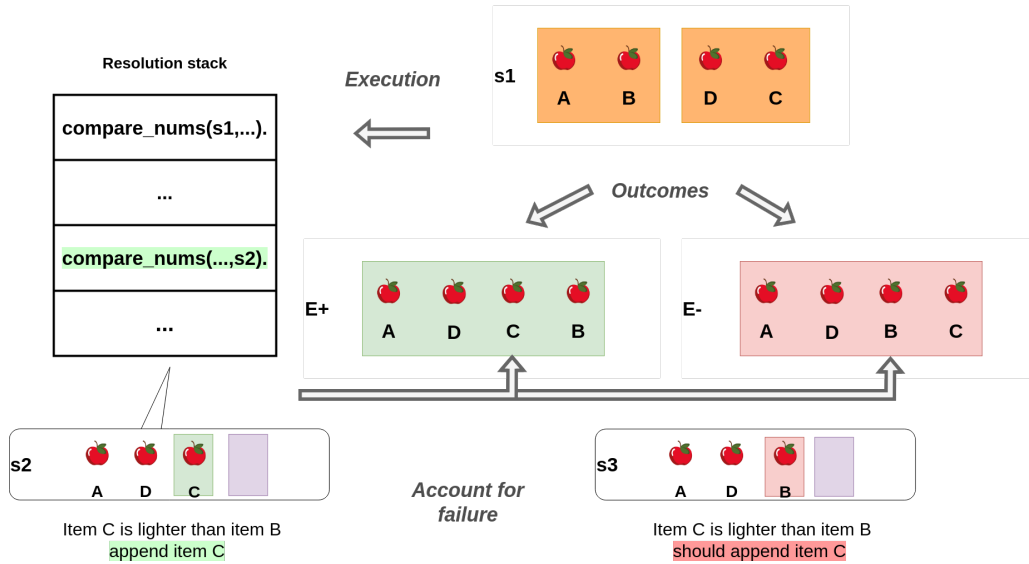


Figure 5.7: Illustration of explanation generation. One set of integer sequences was sampled as training examples and for generating explanations. s_2 and s_3 are outcomes associated with a positive example and a negative example respectively. Given one input integer sequence s_1 and two example integer sequences (s_2 and s_3), logic program *merger/2* is executed on the input *Lt_expr* expression. Intermediate computation expressions are compared with two examples and mapped to equivalent Fruit Weighing representations as two piles of fruits. The error is recognised as s_3 appears inconsistent with the intermediate result which accounts for incorrect decision-making.

and C have been arranged correctly when merging two piles based on their weights. The isomorphism between integer merging and sorting and the designed fruit weighing problem allows mapping from instances from one task to isomorphic instances of the other task.

Results in Psychology [4, 9, 23, 250] show contextualised explanations with appropriate examples as an effective approach for human learning. We produce textual and visual explanations based on first-order programs *merger/2* (Table 5.2). We deliberately generate explanations of merging to avoid pre-mature exposure to experimental material. Texts and visualisations show the correct and incorrect action sequences. Each training example concerns one intermediate merging step. We randomly sample *Lt_expr* of varying lengths as the training examples. These examples are used for training *Metagol_O* too.

Explanations³ provide feedback for a pair of intermediate merging steps which helps contrast the correct and wrong sequences of decision-making. Two outcomes of merging are compared in Figure 5.7. One of the two outputs corresponds to a correctly merged sequence. Given the input integer sequences (two sequences to be merged), the derivation steps and intermediate *Lt_expr* expressions produced by *merger/2* are compared with the incorrect merging outcome. A mapping from primitive definitions to template texts is defined which translates predicates in *merger/2* into natural language. Parts in the resolution tree that are related to primitives in *merger/2* are translated. Accounts of incorrect decision-making are given by suggesting contrastive actions that are correct. Contrastive explanations are frequently used by humans [191] and can be more compact without having a complete causal attribution [159]. In the example shown by Figure 5.7, the correct action to perform at the third position highlighted in s_2 and s_3 should be to append item C. This is due to C being lighter. All visual explanations are generated from the intermediate *Lt_expr* expressions. These expressions are mapped from integer sequences to fruit piles with hidden weights. Differences in the derivation are indicated by highlighting the inconsistent fruit positions in accordance with textual explanations. Hence, in Figure 5.7, the third position is highlighted with green colour in s_2 but coloured red in s_3 . All participants use one set of training examples, so visual and textual explanations are the same and are pre-generated.

³A demonstration of textual explanation generation is provided online at <https://github.com/LAi1997/sequential-teaching>.

5.4.3 Method and design

Table 5.5: Design of the experimental groups. Group 1 learned merging before learning sorting and was provided with visual and textual explanations of merging based on machine-learned rules. 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.

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

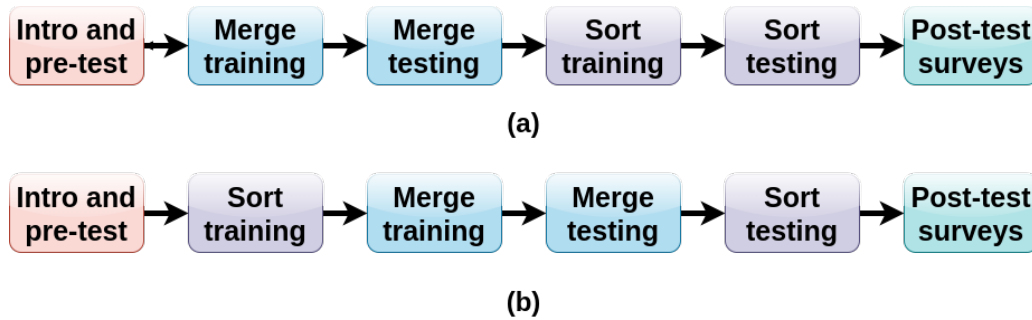


Figure 5.8: In setup (a), participants learned merging before learning sorting (**MS**). In setup (b), participants learned sorting before learning merging (**SM**).

The experiment⁴ was implemented based on a four-group factorial 2×2 design to account for all combinations (Table 5.5.) of the two independent experimental variables, the curriculum order (**CO**) and the presentation of explanations (**EX**). Figure 5.8 shows two curriculum orders. In the curriculum shown by Figure 5.8a, participants received training material on merging first. In the reverse curriculum demonstrated by Figure 5.8b, 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

⁴No approval was required by Imperial College London. Before conducting our experiment, we referred to the ethics approval guideline of Imperial College London (<https://www.imperial.ac.uk/research-ethics-committee/ethics-approval-overview/>). Under the section "Do I need to apply?" on the web page, we assessed that none of the five criteria applied to the work which we were planning to conduct. Hence, our judgement was "No, we do not need to apply". We contacted the Research Governance and Integrity team at Imperial College London since the experiments were carried out and the included data were used by Imperial. They accepted our judgment.

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 Section 5.4.1. 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 before the beginning of the experiment on the 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.

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 ensures all groups experienced the same amount of training and performance tests respecting the curriculum order. When analysing the performance tests, we performed an ANOVA test to inform us if there were differences in the group means and used Tukey's HSD test as a post-hoc test to assess which group means were different.

In merge training, participants first answered the displayed question illustrated by Figure 5.3 and were taken to the next page where feedback and explanations were presented. In Figure 5.4, 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

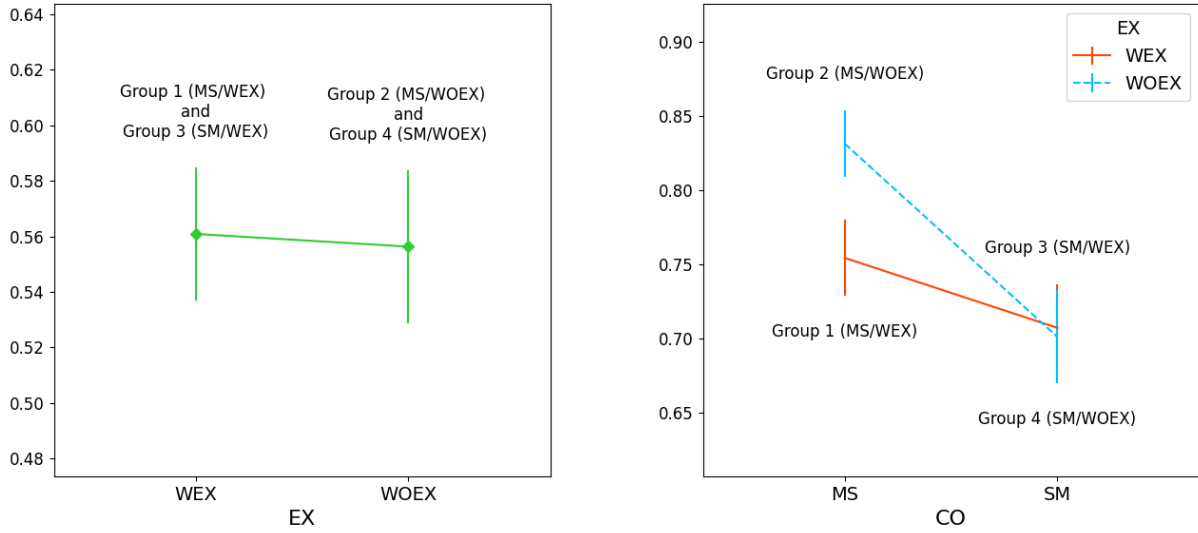
that they believed to have increasing weights from left to right. Another robot instructor Bob helped participants learn to operate the machine “purple diamond” to sort fruits.

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 (Figure 5.5), participants were informed of the number of comparisons their robot instructor Bob made 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.

5.5 Results

Before the experiment, we conducted two experimental trials with one sample of students from the University of Bamberg and another student sample from the Imperial College of 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-



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

Figure 5.9: 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.

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).

Analysis of human 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 5.9a and Figure 5.9b 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 5.9a illustrates almost no change to the mean merge performance score when explanations were introduced. The one-way ANOVA test does not demonstrate a significant effect of the machine-learned theory (**EX**) on the merge performance test score (**PS**). Figure 5.9b illustrates a pattern that switching from the curriculum order **MS** to **SM** results in a reduction in performance. The two-way ANOVA test indicates a significant effect ($F = 10.4, p = .001 < .05$) of curriculum order (**CO**) on sort performance test score (**PS**). The post-hoc Tukey's HSD test shows 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 does 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**).

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 5.4. 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. The

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

Chi-squared test⁵ in this example has a $X^2 = 14.3$ with $p < .025$ which supports the correlation of number pairs in $ht(Ns, M_h)$ and $mt(Ns, M')$. For common pairs of numbers in $ht(Ns, M_h)$ and $mt(Ns, M')$, we obtain their ranks and compute Spearman's rank correlation coefficient. The correlation coefficient is .9 and $p < .025$ which supports 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 $\langle n_i, n_j \rangle$ denoting the comparisons of integers made by M where $n_i, n_j \in Ns$. We employ two statistical tests (Example 5.4) 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 $\langle n_i, n_j \rangle \in Tr$ 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 $\langle n_i, n_j \rangle \in Tr$, we check if $\langle n_i, n_j \rangle$

⁵We add one to all frequency counts to avoid including zeros in the table for the Chi-squared test.

Table 5.6: Mean strategy frequency in sort training and sort performance test. Each human sorting trace in the sort training and sort performance test was matched to one of the following categories of sorting algorithms, bubble sort (*BS*), dictionary sort (*DS*), insertion sort (*IS*), merge sort (*MS*), quick sort (*QS*), a combination of insertion and dictionary sort (*Hybrid*) and unclassified (*Other*). The number of human sorting traces in each category was averaged by the total number of traces. We highlight positive differences greater than .05 in grey and in bold font. In each group, the highlighted difference is the largest increase in algorithm application frequency from sort training to sort performance test.

PS \ Categories	<i>BS</i>	<i>DS</i>	<i>IS</i>	<i>MS</i>	<i>QS</i>	<i>Hybrid</i>	<i>Other</i>
<i>Group 1 (MS/WEX)</i>	–	–	–	–	–	–	–
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
<i>Group 2 (MS/WOEX)</i>	–	–	–	–	–	–	–
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
<i>Group 3 (SM/WEX)</i>	–	–	–	–	–	–	–
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
<i>Group 4 (SM/WOEX)</i>	–	–	–	–	–	–	–
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

is a member of $ht(Ns, M_h)$ and if $\langle n_i, n_j \rangle$ is a member of $mt(Ns, M)$. We 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 [283]. The intention of computing Spearman's rank correlation coefficient is to examine the ordering alignment of comparison pairs.

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

Table 5.7: 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. In Table 5.6, 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.

<i>Group 1 (MS/WEX)</i>		Training		
Quick Sort		Application	No application	Total
Performance test	Application	7	6	13
	No application	0	7	7
	Total	7	13	20

<i>Group 2 (MS/WOEX)</i>		Training		
Merge Sort		Application	No application	Total
Performance test	Application	2	7	9
	No application	0	11	11
	Total	2	18	20

<i>Group 3 (SM/WEX)</i>		Training		
Dictionary Sort		Application	No application	Total
Performance test	Application	3	7	10
	No application	1	9	10
	Total	4	16	20

<i>Group 4 (SM/WOEX)</i>		Training		
Insertion Sort		Application	No application	Total
Performance test	Application	7	11	18
	No application	0	1	1
	Total	7	12	19

Table 5.6 demonstrates the change in the distribution of strategy as the difference in mean strategy frequency from the training and performance test. In addition, Table 5.7 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 5.6, 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 5.7, 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 ($\mu = .679, \sigma = .373$).

Group 2 (**MS/WOEX**) and Group 4 (**SM/WOEX**) did not receive explanations (**WOEX**). From Table 5.6, 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 5.7, 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 sort like approaches ($\mu = .877, \sigma = .285$) in Group 4 achieved a higher mean score ($p = .001 < .05$) than the other strategies ($\mu = .644, \sigma = .399$).

5.6 Discussion

In Table 5.8, 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 [216] 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

Table 5.8: **H1** to **H3** are hypotheses concerning effects of curriculum order and explanations on human sorting comprehension. 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, S is an abbreviation for supported, and N stands for not supported.

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

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 5.7 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.

In the definition of *sorter/2* learned by *Metagol_O* after learning *merger/2* from Table 5.1, the program size is $u = 3$ and the number of predicates including those used in *merger/2* is $p = 8$. in the definition of *sorter/2* learned by *Metagol_O* without learning *merger/2* from 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 Hypothesis 3.17, $3 \cdot \ln(8) < 5 \cdot \ln(6)$ 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 (Figure 5.9b) 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 supports hypothesis **H1** and Hypothesis 3.17.

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 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 supports **H4** and supports Hypothesis 3.16.

Since merging was taught in isolation independent from other secondary or tertiary concepts, we refer to the cognitive window (Remark 5.1) 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 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 in Table 5.6 demonstrates four possible sorting strategy adaptations. From Table 5.7 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

Table 5.9: Summary of the significance of participant's strategy adaptation and performance improvement of the adapted algorithm compared with other algorithms. A tick denotes a significant analysis result, and a cross indicates an insignificant analysis outcome.

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	✓	✓

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 5.9 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**. In Group 1, this positive difference in sorting performance using quick sort supports hypothesis **H5**.

The t-test results of Group 1 and 3 (summarised in Table 5.9) suggest that curriculum order does not affect the performance of adapted strategies since Group 1 experienced the incremen-

tal 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 argue that without the benefit of the incremental curriculum and a compact hypothesis space (Hypothesis 3.17), 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 invention of problem solutions in novices.

Impacts of explanations on performance

As summarised in Table 5.9, 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 [4, 9, 23, 250]. 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.

Impacts of curriculum order on strategy adaptation

Another observation from Table 5.7 and 5.9 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 5.7 and Table 5.9, 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 Hypothesis 3.17. In Hypothesis 3.17, two curricula are compared in terms of the size of the composite hypothesis spaces of learning some target predicates. Incrementally learning concepts can reduce 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 5.9. 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 efficient 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.

5.7 Summary

Previous work on the topic of comprehensibility has investigated the beneficial effects of machine-learned logic theories in a single teaching session. This chapter addresses the increasing emphasis on involving humans in collaborative two-way communications with AI by exploring curricula that contain a sequence of teaching sessions. A sequential teaching curriculum 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 supports **Sub-thesis 2** and Hypothesis 3.17. Results show that when a machine-learned logic program does not have a sufficiently low cognitive cost, explaining it does not lead to a beneficial effect on human comprehension. This supports Hypothesis 3.16.

In addition, adaptations of human sorting strategies have been observed which utilise techniques of divide-and-conquer and recursion. Such strategy adaptations were observed in humans who learned with explanations and in humans who learned concepts with incremental complexity. In the case that humans learn incrementally with explanations, the application of a re-discovered efficient strategy has led to enhanced task performance. This supports **Sub-thesis 1 and 2**. However, further investigations are required to understand better the degree to which sequential teaching curricula with machine-learned explanations facilitate novel human comprehension. We elaborate on future work in Chapter 6.

Chapter 6

Conclusions and future work

Human activities and AI applications are becoming inseparable due to the growing degree of machine-human interactions [88, 192]. Highly interleaving networks of machine-human communication have become the cornerstone of the Industry 4.0 paradigm [220]. Two-way interactions are also essential to the continuous improvement of the performance of AI applications [191]. In education [55] and public service [292], the companionship of intelligent agents compensates for the lack of human resources and actively maintains engaging environments which are particularly valuable.

While AI systems have shown great potential [79, 292] and even demonstrated superhuman performance [126, 277], sufficient information has not been provided in most cases for humans to understand. The opacity of machine learning systems has hindered efforts to produce epistemic results and fulfil AI's needed role in science [144]. In other domains, such barriers can lead to failure to establish trust in high-stakes decision-making, the inability to transfer knowledge to humans, and inconsistent operations. These concerns have been investigated by Explainable Artificial Intelligence (XAI) research [112] which aims to facilitate human understanding and provide explanations of AI systems' outputs, structures, decisions and functions. Currently, there is a lack of operational definition and evaluation of the effects of AI explanations on human comprehension.

We critically demonstrate that explanations generated from machine-learned logic programs can quantifiably improve human performance in complex tasks. This is established on our novel E_{ex}

framework [3, 2] from Chapter 3 which quantitatively evaluates the effects of machine explanations and teaching curricula on human comprehension. Our work relates to Donald Michie’s Ultra-Strong Machine Learning (USML) classification [186]. In USML, a machine learning system acts as a “teacher” who represents its symbolic output to humans and teaches them to achieve high performance. This is modelled by a teaching curriculum in E_{ex} . Our framework builds on an operational definition of comprehensibility based on human out-of-sample predictive accuracy [268]. This notion of human comprehension is limited to logic programs. It does not account for the harmfulness of machine explanations and the perspective that human learning often utilises previously acquired knowledge. These limitations have restricted its scope of application. In contrast, our E_{ex} framework does not have these drawbacks. E_{ex} applies to a wide range of explanation representations and tasks. We account for the harmfulness of explanations and concept order through our definitions of explanatory effects and sequential teaching curricula. Based on E_{ex} , we claim that *changes in human comprehension are quantitatively identifiable and affected by both explanations of machine-learned hypotheses and the presentation order of target concepts* (Chapter 1). We examine this thesis via two sub-theses (Chapter 1):

- Sub-thesis 1** For a non-trivial task, improvement and degradation in human comprehension can be quantitatively identified and are influenced by explanations of machine-learned hypotheses.
- Sub-thesis 2** For a non-trivial task, improvement in human comprehension in sequential curricula can be quantitatively identified and is influenced by the presentation order of target concepts.

We support these sub-theses with results from two publications [3, 2]. As demonstrations of our E_{ex} framework, we study the effects of explaining machine-learned logic programs on human comprehension in the context of game-playing and sorting. Dynamic environments such as games offer varying difficulties for analysing human reasoning [228]. In sorting, efficient strategies are not trivial [311] which allows us to examine human inferential ability in combining already learned knowledge. Logic programs can represent knowledge structurally analogous to human concept hierarchy [14, 44, 223] and implement logic mechanisms of human reasoning

[121, 130, 152]. We have chosen MIL, a variant of ILP, to learn logic programs due to its ability to perform predicate invention and learn recursive concepts. MIL systems *MIGO* [212] and *Metagolo* [74] are modified to learn strategies of a two-player game and sorting. In Chapter 4, **Sub-thesis 1** is supported by results from experiments for teaching humans a two-player game. In Chapter 5, **Sub-thesis 1 and 2** are supported by results from experiments for teaching humans computational sorting algorithms. Specific to logic programs, we have developed a preliminary theory including two hypotheses of a cognitive window of human comprehension and sequential teaching curriculum to account for the limitations in human learning and mental execution. This theory is supported by our empirical results in Chapter 4 and 5.

6.1 Summary of contributions

Contribution 1: Explanatory effects of machine-learned logic theories on human comprehension (supports Sub-thesis 1)

In Chapter 3, we propose the E_{ex} framework, a representation-independent assessment framework that quantitatively evaluates the effects of machine explanations on human comprehension. We present a preliminary theory of a cognitive window that includes two constraints on a logic program: 1) the size of the hypothesis space associated with the program and 2) the cognitive complexity for approximating the load on working memory during the human execution of the program. We define a cognitive cost of a logic program Cog as an estimation of this cognitive complexity. We postulate the existence of a cognitive window: when a machine-learned logic program has low descriptive complexity and its execution is cognitively straightforward, explaining the program improves human comprehension.

Importantly, we demonstrate that machine explanations generated from logic programs can measurably improve human comprehension using E_{ex} [3]. Human trials have been carried out in the domain of a simple two-player game. In Chapter 4, empirical results show that a machine-learned theory has a beneficial effect on human comprehension when it has low descriptive and executional complexity. This result supports **Sub-thesis 1**. In addition, our results show when logic programs have high descriptive or executional complexity, they have either harmful

or no effect as the human task performance becomes worse or remains the same. This supports **Sub-thesis 1**. Both results support our cognitive window hypothesis.

Contribution 2 Effects of a sequential teaching curriculum on human comprehension (supports Sub-thesis 1 and 2)

In the E_{ex} framework (Chapter 3), we provide operable definitions of sequential teaching curricula to account for the impacts of curriculum order on human comprehension. Sequential teaching curricula [2] accommodate a learning mechanism via the division of top-level learning tasks into smaller lower-level tasks. In Chapter 3, we hypothesise that a sequential teaching curriculum can improve human comprehension if it can introduce appropriate concepts to reduce the size of the hypothesis space for learning the concepts. Experiments were conducted for teaching sorting algorithms (Chapter 5). Our analysis of empirical results demonstrates that a sequential teaching curriculum with increasing concept complexity has a beneficial effect on human comprehension. This supports **Sub-thesis 2** and our hypothesis on human comprehension improvement by a sequential teaching curriculum. We observe no human comprehension improvement when explanations do not have a sufficiently low cognitive cost. This supports our cognitive window hypothesis. An interesting phenomenon of human knowledge re-discovery is highlighted. Novices in computational algorithms unintentionally were found to develop recognised ideas or concepts known to experts. These human learners were able to re-discover some known efficient computational algorithms and apply them effectively when they learn incrementally with machine explanations. This supports **Sub-thesis 1 and 2**.

6.2 Future Work

With respect to the limitations of our contributions, we discuss future work that could be carried out on the evaluation of comprehensibility, potential aspects for optimisation and application of sequential teaching curricula for facilitating discovery and human performance improvement.

6.2.1 Evaluation of comprehensibility

Scope of experimentation

In our cognitive window and sequential teaching hypotheses, a human-comprehensible logic program is associated with a small hypothesis space and should have a low cognitive executional complexity. These hypotheses are supported by empirical evidence in two experiments involving human participants. Owing to limited experimental resources, we have not been able to examine our E_{ex} framework and hypotheses in other domains. Future work could build on existing results and the E_{ex} to investigate the comprehensibility of machine explanation in other non-trivial tasks. Engaging and interactive game-like tasks are appropriate candidates for future experiment settings. These dynamic environments allow analyses of human reasoning ability at different levels [228]. Other activities such as discovery in science provide a basis for experimentation of how machine explanations impact real-world human reasoning [228]. The ability of AI to learn and present interesting concepts in a human-comprehensible format is a key perspective of machine intelligence. There is an increasing emphasis on applying XAI techniques to drive scientific discovery [316, 144]. Empirical demonstrations could examine the USML [186] where a machine learning system can obtain knowledge and explain it in a way such that scientists can reason about it. To obtain insights into human reasoning in these situations, future studies could emphasise more on human trace analysis. In our experiments, we have observed adaptations to efficient problem solutions from human traces. This implies a trace-based assessment can provide additional information on changes in human problem solutions.

Scope of application

In Chapter 3, we have discussed how the E_{ex} framework can be applied to evaluating explanations from different machine-learning systems. We have not been able to explore explanations of machine-learned outputs other than logic programs. Our definitions of comprehensibility cover explanations produced by Turing machines in a finite number of steps. These definitions are representation-independent. This allows maximum freedom of representations that can be

tested, e.g. decision rules, graphs, natural language texts or hybrid representations. In addition, any explanation generation algorithm represented by a Turing machine that terminates can be considered as an “explainer”. With an increasing number of methods for explaining machine learning systems [222], we offer an operational notion of comprehensibility for machine learning systems in general.

In Chapter 2, we have suggested applications of the E_{ex} framework for human-AI communication. While we acknowledge the reviewed areas are not exhaustive, we consider our work to be useful for at least these research directions (EBL, ABML, Machine Teaching, Virtual Bargaining and Dialogue Systems). While there is an emphasis on improving the communication between humans and AI systems, the effect of communication on humans is often assumed without validation. Our E_{ex} framework provides an operational assessment of how well people can understand and follow the communicated information. For instance, similar to our demonstration in 4, our E_{ex} framework can be used to test user misinterpretation as a harmful effect of explanations for EBL. For ABML, argumentative explanations may provide guidance to resolve conflicts between arguments and offer justifications for decision-making in “black box” machine learning systems. The E_{ex} framework can quantitatively assess if teaching argumentative explanations can help users become better at decision-making for a prediction task. We have conducted our studies in non-trivial human reasoning domains using E_{ex} and developed a preliminary theory supported by empirical results. The E_{ex} offers a potential solution to close the gap between theories and applications in Machine Teaching. Our framework can also be applied to assess human comprehension of flexible and instantaneous signals by AI. Insights obtained can be used to optimise virtual conventions in real-world human-AI interactions such as autonomous driving and dynamic games.

Recent advancements in Dialogue Systems have focused on large language models. Evaluations of language models and their applications are often too subjective or too context-independent [150]. Most of them have been objectively evaluated based on “ground truths” input-output data [41]. These assessments are disjoint from the real use cases of the language models. Recent efforts aim to build dynamic and interactive language model assessment platforms [150, 63]. However, empirical evaluations, for example, involving humans scoring the helpfulness and satisfaction of explanations, are dependent on user preferences. These tests do not objectively

measure the extent to which the participants have learned from the explanations. The ability of language models to enrich domain-specific representations (e.g. logic programs) for human understanding and performance could be examined via our framework. A use case of the framework would be to assess the translation to human-like texts from logic programs that predict some function. We can then examine human interactions with language models where the objective is to improve the human application performance of the domain knowledge.

Low quality explanations

The quality of explanations and the effect of explanations on humans require different evaluation criteria that should be distinguished [222]. We clarify that the E_{ex} framework is an operational evaluation that can be used to represent comprehensibility from the human user perspective. It cannot reflect the quality of explanation from the view of the machine-learned model. However, it can be employed to examine the effect of low-quality explanations on human comprehension. Various properties such as consistency and correctness are of interest in XAI [124, 222]. As an example, explanations can be generated from a machine-learned model given identical inputs and the effect of inconsistency on human comprehension can be reflected by human task performance. Also, for complex problem-solving tasks, precise labelling of domain instances might not be feasible and learning a completely correct predictive model is not possible. When adopting the E_{ex} framework, the objective of the machine “teacher” would be to provide low-error explanations in its perspective as frequently as possible to ensure maximum explanation correctness.

Encouraging human self-explanations is relevant when training examples are incorrectly labelled and machine explanations do not cover all training examples. Future work could also focus on comprehensibility through both human self-explanations and machine explanations. People often look for an explanation in the case of unexpected events from their perspectives [191]. Self-explanation has been shown to support learning and reasoning [165]. When an explanation inconsistent with an example is presented, questions such as “why is the example E labelled A rather than B ?” demand contrastive self-explanations. Humans can steer clear of the pitfalls of the machine-learned model and improve their performance based on contrastive self-

explanations [243]. People could be encouraged to derive their explanations from additional training examples from an oracle. Given the frequent reports of errors by large language models, such interaction may enhance the safety of applications [20] by avoiding factually incorrect information often presented in their responses [161]. Experimentally, this also requires more interactive teaching curricula to be implemented.

Standardisation of comprehension evaluation platforms

In our empirical studies, we have carefully designed experimental procedures to ensure consistent results. However, since our experiments in Chapter 4 and 5 have been conducted separately, participant samples vary in size. This is due to the difficulty of recruiting participants. To address this, an open and standardised evaluation platform is necessary. Standardisation of benchmarks is essential for evaluating the performance of machine learning systems. Platforms and challenges such as the ImageNet [258] have provided important sources for standardising performance comparison of different machine learning systems. Given some appropriate problem-solving tasks, the comprehensibility of explanations provided by machine learning systems may also be examined and benchmarked. Participation in such a challenge requires a system to learn from a set of training examples. The system is asked to classify some test examples and provide explanations for them. These explanations are given to train newly recruited human participants. Afterwards, participants should be given some test questions to classify unseen examples based on the E_{ex} framework. The predictive accuracy of human participants would be recorded as the comprehensibility of the machine explanations. With significantly large participant groups, machine explanation approaches could be compared by contrasting human learning performance.

Non-explainability and human intelligibility

In our E_{ex} framework, we consider an output by any Turing machine that terminates as a machine explanation. Machine explanations differ only in their effectiveness to accommodate human comprehension. It means the question of whether it is impossible for any machine to generate an explanation of a problem, is trivial. There could always be some machine

that returns its input to the human learner. These outputs might not necessarily be useful for human comprehension but they are categorised as machine explanations in the E_{ex} . Yet, one non-trivial question is how to define the “hardness” of providing a machine explanation. We discuss one tentative account of this notion here. The time complexity of an explanation generation algorithm as a function of its input size can be viewed as an indication of the “difficulty” in explaining. Problems that require algorithms bounded by higher time complexity to explain can be seen as harder explanation problems. In future work, the connection between non-explainability and computational complexity theory could be further explored.

E_{ex} can be extended to examine human intelligibility and human declarative understanding of explanations. In Michie’s view [182], effective computational representations for human understanding and validation should be both human-intelligible and human-executable. An explanation is human-intelligible when humans can memorise and describe the associated concepts. A human-executable explanation can be validated by humans through mental calculation. However, from the perspective of “explainees”, the failure of explanations to be intelligible or executable by humans may imply non-effectiveness in certain situations. Owing to the limitation of cognitive capacities, complex computations are not always operable by humans. A person may be able to write an implementation of merge sort to arrange a long sequence but may not always be able to perform the sorting mentally. On the other hand, a declarative understanding requires the awareness and knowledge of factual information. A declarative comprehension of a machine algorithm can allow a computer scientist to identify algorithmic properties that are universally true for all inputs of the algorithm.

Currently, these aspects of human comprehension and human explicit reasoning have not been explored. We can extend E_{ex} to account for situations in which humans can collaborate with a programming assistant to encode human problem solutions as computer programs. This approach is related to Human-Assisted Computation [272]. Programs written by humans would then be executed by another machine which alleviates the computational burden of the human learners. The comparison between the explanatory effects would then be possible based on the performance of written programs before and after learning the explanations. In addition, humans could be given a pool of insightful statements and tested on their accuracy in predicting the correct insights after studying explanations. Future work could focus on clarification and

empirical demonstration of various forms of human comprehension.

6.2.2 Optimisation of comprehensibility

Representation of explanations

This thesis has not explored the effect of different representations on human comprehension. Results in Psychology have shown that appropriate representation of information can lead to significant improvements in human task performance and learning [141, 177]. Frameworks that support the selection of representation [201, 25, 282] are only applied subjectively without means to quantify the cognitive costs of representations [245]. Empirical evaluations of a narrow range of representation schema [254, 178, 261, 319] provide limited insights on optimising the general effectiveness of representations.

Recent work on a representation choice framework called *rep2rep* [245] investigates a quantified approach to recommend effective representation systems. The *rep2rep* framework removes the need for subjectivity by specifying representational systems in a defined language and computing cognitive costs between different representational systems. Cognitive costs arise from hypothesised costs in registering the components of a representation, parsing structure, interpreting symbols, and developing a solution based on relevant background knowledge [245]. Results [245, 246] demonstrated that optimising cognitive costs from specifications of mathematical and cognitive properties can yield an effective encoding of math problems. Our framework can be used to operationally evaluate the effect of different representations on human comprehension. Based on the *rep2rep* approach, we may define a more comprehensive cognitive cost function and a representation specification language. Future work built on E_{ex} can extend the theory of cognitive window to objectively compare representations beyond logic programs.

Concept invention and background knowledge.

Currently, machine learning systems are still limited in their ability to determine appropriate background knowledge and invent meaningful concepts for knowledge transfer. Large language models have received great attention due to their capability of communicating with humans

via natural language [320]. These models are often constrained by training data from known textual corpora and cannot effectively generate new knowledge beyond what has been documented. A challenging problem remains to be addressed: how should we design machine learners that can invent concepts intelligible and executable by humans? We can see such machine learners would need to be USML. Our E_{ex} framework provide an assessment measure for machine inventions that are both intelligible and executable. As a starting point, we may refer to a notion of invention defined in ILP (Definition 2.8). All new and reusable predicate definitions learned automatically are machine inventions. We can apply the E_{ex} framework to investigate the effects of concepts invented by different machine learning systems on human comprehension. Our results show that explaining a logic program of game strategies that uses an additional predicate can improve human comprehension. This extra predicate definition introduced in background knowledge checks the number of near-win patterns and presents a “shortcut” for human decision-making. However, the creation of this predicate definition still relies on expert knowledge and has not been automated. Future work could derive insights from human comprehension results to automatically optimise machine inventions.

Results in Psychology [45] suggest that reducing the complexity of mental execution and goal maintenance in working memory can improve human task performance and thus enhance comprehensibility. Recent results [76] demonstrate that optimising the textual and computational complexity of logic programs can lead to better learning performance of MIL systems. Future work could consider optimising background knowledge used in machine explanations. In ILP, for example, comprehension-friendly background knowledge may be identified by incorporating a measurement of cognitive cost into ILP systems. Background knowledge recognised this way could be hypothesised to produce effective explanations. Empirically, E_{ex} could be applied to examine these hypotheses. This optimisation approach could also be extended beyond ILP for identifying background knowledge with a high explanatory impact that enhances human-AI communication.

Curriculum order and example selection

As an initial attempt to integrate our framework with interactive teaching curricula, we have examined a simple curriculum setting. For more refined partitions of concepts or complex concepts that depend on multiple sub-concepts, curricula may need to contain more than two teaching material blocks. For effective teaching curricula, we expect the order of teaching material blocks to be a crucial factor. In Psychology [223, 217], human knowledge acquisition can be viewed as the hierarchisation of concepts. Each node in this structure is associated with the information of a sub-concept and may be dependent on multiple sub-nodes. However, since a curriculum involves a sequence of teaching blocks, each focusing on one concept, the hierarchy of concepts could be linearised. Future work could explore the problem of finding a linearisation of teaching materials for high human comprehension.

Another interesting perspective concerns the optimisation of training example selection to improve human comprehension. In the context of scientific discovery, designing informative experiments to explain some hypothesised phenomenon requires deliberate thinking. The question of “which experiments should be performed?” may not be immediately apparent due to the limit of experimental time and resources. The topic of optimising training examples has been explored in Machine Learning. Some related approaches are the Active Learning paradigm [271] and Machine Teaching [321]. Given some objective functions, frameworks in Machine Teaching [97, 171] aim to recognise a set of training examples that maximises the objective value. On the other hand, Active Learning iteratively consults a data source to label training examples that have maximal entropy [197]. This approach is considered effective in domains where training labels are not readily available. Both of these approaches optimise for some objective specification. This objective can be adapted for identifying training examples with the highest expected human comprehension. Further studies could build on the E_{ex} framework to examine the effects of different active learning and machine teaching methods on human comprehension.

6.2.3 Applications of sequential teaching curricula

Conditions of effective human re-discovery

Future work can explore the phenomenon of human re-discovery in complex human reasoning situations. Our experiments show the creativeness of human learning in exploiting the analogical transfer of partial solution structure. In the human trial described in Chapter 5, we have employed a mixed approach that allows a higher degree of freedom for learners to re-discover computationally efficient algorithms. We observed adaptations of human sorting strategies to utilise techniques of divide-and-conquer and recursion. Such strategy adaptations were observed in humans who learned with explanations and in humans who learned concepts with incremental complexity. 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 developed sorting strategies by employing divide-and-conquer and recursion-like techniques commonly used by efficient machine sorting algorithms. While not all inspected human learners in the experiment 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 itself an achievement for those novices of programming. Discovery encouraging learning allows individualised understanding [36]. This can help trigger sudden realisations of previously incomprehensible concepts. However, further investigations are required to gain a better understanding of the degree to which sequential teaching curricula with machine explanations facilitate human re-discovery.

Machine-human two-way learning via behavioural debugging

An exciting prospect of sequential teaching curricula is that machines and humans could take up more active roles to enable two-way cooperative learning [275]. Sequential teaching curricula can be extended to represent the coordination between humans and machines in a back-and-forth fashion. One perspective is explicitising human implicit decision-making by machine-

learned theories via a process known as behavioural cloning [31]. Programs are learned by machines to represent goal-oriented knowledge in a form similar to the human conceptual structure. Traces of human problem solutions can be provided as inputs to a machine learning system and used to model decision-making processes. Hence, MIL is an appropriate candidate for cloning since it can iteratively learn recursive programs by inventing sub-goal predicates. However, cloned rules associated with demanding operations and complex environments could be difficult for humans to understand [19]. When the objective is to improve human task performance, direct presentations of faulty rules would not be the most effective approach. To address this, optimised sequential curricula can be designed to “debug” human knowledge progressively in terms of dependency and complexity. In incremental curricula, cloned sub-goals are compared with machine-learned sub-goals. Mistakes can be identified and corrected via program revisions [155]. This could lead to a “clean-up” effect on human behaviours which can be validated via empirical experiments by presenting appropriate explanations in sequential training. These corrections in a human strategy are helpful in intelligent tutoring [317] and algorithmic debugging [274] to make humans and machines interactively learn from each other.

6.3 Summary of thesis achievements

We have explored in this thesis an operational assessment of the explanatory effects of AI in interactive machine-human teaching. Most XAI studies do not emphasise empirical results and do not provide unambiguous definitions. Our novel E_{ex} framework provides an objective definition of comprehensibility based on human out-of-sample predictive accuracy. Notably, E_{ex} is representation-independent and flexible for evaluating human comprehension in a wide variety of prediction tasks. It accounts for the beneficial and harmful effects of machine explanations. Experimentally, we use E_{ex} to examine the impacts of explaining logic programs on human comprehension of a simple game. Importantly, we demonstrate that explaining machine-learned logic programs can improve human task performance. Results support our preliminary theory of a cognitive window of human comprehension which requires a machine-learned logic program to be informatively compact with low goal maintenance cost for it to improve human comprehension.

In addition, we expand the E_{ex} framework to account for cases where complex concepts are partitioned into sequentially taught sessions. In our empirical study, we examine the effects of sequentially teaching concepts with increasing or decreasing complexity. We observe an improvement in task performance from humans who learn with increasing concept complexity. This supports our hypothesis that a reduction in the size of the associated hypothesis space of the target concept implies an improvement in human comprehension. We obtain empirical results which demonstrate human’s potential for re-discovering computational algorithms and optimised problem-solving strategies in a limited time frame.

We believe that the frameworks and results presented in this thesis have opened inspiring avenues of interdisciplinary research in AI which will encourage the XAI community to further examine AI-human interactions via operational empirical approaches.

Bibliography

- [1] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.
- [2] L. Ai, J. Langer, S. H. Muggleton, and U. Schmid. Explanatory machine learning for sequential human teaching. *Machine Learning*, 112:3591–3632, 2023.
- [3] L. Ai, S. H. Muggleton, C. Hocquette, M. Gromowski, and U. Schmid. Beneficial and harmful explanatory machine learning. *Machine Learning*, 110:695–721, 2021.
- [4] V. Aleven and K. R. Koedinger. An effective metacognitive strategy: learning by doing and explaining with a computer-based cognitive tutor. *Cognitive Science*, 26(2):147–179, 2002.
- [5] L. Alfieri, P. Brooks, N. Aldrich, and H. R. Tenenbaum. Does discovery-based instruction enhance learning? *Journal of Educational Psychology*, 103:1–18, 2011.
- [6] H. Allahyari and N. Lavesson. User-oriented assessment of classification model understandability. *11th Scandinavian Conference on Artificial Intelligence*, 2011.
- [7] E. Allender. Applications of Time-Bounded Kolmogorov Complexity in Complexity Theory. In O. Watanabe, editor, *Kolmogorov Complexity and Computational Complexity*, EATCS Monographs on Theoretical Computer Science, pages 4–22. Springer, Berlin, Heidelberg, 1992.
- [8] J. R. Anderson. *The adaptive character of thought*. The adaptive character of thought. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, US, 1990. Pages: xii, 276.

- [9] J. R. Anderson, J. M. Fincham, and S. Douglass. The role of examples and rules in the acquisition of a cognitive skill. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 23(4):932, 1997.
- [10] J. R. Anderson, N. Kushmerick, and C. Lebiere. The tower of hanoi and goal structures. In *Rules of the Mind*, pages 121–142. Psychology Press, 1993.
- [11] J. R. Anderson and R. Thompson. Use of analogy in a production system architecture. In *Similarity and Analogical Reasoning*, page 267–297. Cambridge University Press, USA, 1989.
- [12] R. Andrews, J. Diederich, and A. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373–389, 1995.
- [13] S. Anjomshoe, A. Najjar, D. Calvaresi, and K. Främling. Explainable agents and robots: Results from a systematic literature review. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, page 1078–1088, 2019.
- [14] Y. Anzai and H. A. Simon. The theory of learning by doing. *Psychological Review*, 86(2):124–140, 1979.
- [15] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, et al. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.
- [16] W. Arthur Jr. and D. V. Day. Development of a short form for the raven advanced progressive matrices test. *Educational and Psychological Measurement*, 54(2):394–403, 1994.
- [17] N. Asghar, P. Poupart, X. Jiang, et al. Deep active learning for dialogue generation. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics*, page 78–83, 2017.
- [18] M. Bain and S. H. Muggleton. Learning optimal chess strategies. In *Machine Intelligence*, pages 291–309. Oxford University Press, Inc., New York, NY, USA, 1995.

- [19] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, 1995.
- [20] Y. Bang, S. Cahyawijaya, N. Lee, et al. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint*, 2023.
- [21] P. Baroni, A. Rago, and F. Toni. From fine-grained properties to broad principles for gradual argumentation: A principled spectrum. *International Journal of Approximate Reasoning*, 105:252–286, 2019.
- [22] D. Beal and M. R. B. Clarke. The construction of economic and correct algorithms for king and pawn against king. *Advances in Computer Chess*, 2, 1980.
- [23] D. C. Berry and D. E. Broadbent. Implicit learning in the control of complex systems. In *Complex problem solving: The European Perspective*, pages 131–150. Lawrence Erlbaum Associates, Inc., 1995.
- [24] J. Bitner and B. Hansche. Topics concerning the kpk endgame. Technical report, Urbana-Champaign: University of Illinois, 1976.
- [25] A. Blackwell and T. Green. *Notational systems—the cognitive dimensions of notations framework*, chapter HCI models, theories, and frameworks: toward an interdisciplinary science. Morgan Kaufmann, 2003.
- [26] H. Blockeel. Hypothesis Language. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning*, pages 507–511. Springer US, Boston, MA, 2010.
- [27] H. Blockeel and L. De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1):285–297, 1998.
- [28] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik–Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.
- [29] M. A. Bramer. An optimal pattern-based algorithm for king and pawn against king. *Advances in Computer Chess*, 2, 1980.
- [30] M. A. Bramer. Pattern-based representations of knowledge in the game of chess. *International Journal of Man-Machine Studies*, 16(4):439–448, 1982.

- [31] I. Bratko, T. Urbančič, and C. Sammut. Behavioural cloning: Phenomena, results and problems. *IFAC Proceedings Volumes*, 28(21):143–149, 1995.
- [32] M. Brent, R. Chris, and W. Sandra. Explaining explanations in AI. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, page 279–288, 2019.
- [33] G. Brewka, T. Eiter, and M. Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.
- [34] C. E. Briguez, M. C. D. Budán, C. A. D. Deagustini, et al. Argument-based mixed recommenders and their application to movie suggestion. *Expert Systems with Applications*, 41(14):6467–6482, 2014.
- [35] T. Brown, B. Mann, N. Ryder, et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Proceedings of the 34th International Conference on Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.
- [36] J. S. Bruner. The art of discovery. *Harvard Educational Review*, 31:21–32, 1961.
- [37] J. S. Bruner, J. J. Goodnow, and G. A. Austin. *A Study of Thinking*. New York, NY: Wiley, 1956.
- [38] N. Bulso, M. Marsili, and Y. Roudi. On the complexity of logistic regression models. *Neural Computation*, 31(8):1592–1623, 2019.
- [39] A. Bundy and B. Mitrovic. Reformation: A domain-independent algorithm for theory repair. Technical report, University of Edinburgh, 2016.
- [40] A. Bundy, E. Philalithis, and X. Li. Modelling virtual bargaining using logical representation change. In S. H. Muggleton and N. Chater, editors, *Human-Like Machine Intelligence*. Oxford university Press, Oxford, UK, 2020.
- [41] R. Burnell, W. Schellaert, J. Burden, et al. Rethink reporting of evaluation results in AI. *Science*, 380(6641):136–138, 2023.
- [42] R. M. J. Byrne. Suppressing valid inferences with conditionals. *Cognition*, 31(1):61–83, 1989.

- [43] R. Caballero, A. Riesco, and J. Silva. A survey of algorithmic debugging. *ACM Computing Surveys*, 50:1–35, 2017.
- [44] J. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. *Machine Learning*, 11:26, 1985.
- [45] P. Carpenter, M. Just, and P. Shell. What one intelligence test measures: A theoretical account of the processing in the raven progressive matrices test. *Psychological Review*, 97:404–431, 1990.
- [46] L. Carstens and F. Toni. Improving out-of-domain sentiment polarity classification using argumentation. *ICDMW*, page 1294–1301, 2015.
- [47] R. Caruana, H. Kangarloo, J. D. Dionisio, U. Sinha, and D. Johnson. Case-based explanation of non-case-based learning methods. In *Proceedings of the AMIA Symposium*, page 212–215, 1999.
- [48] A. Cawsey. Generating interactive explanations. In *Proceedings of the 9th National Conference on Artificial Intelligence*, page 86–91, 1991.
- [49] A. Cawsey. *Explanation and Interaction: The Computer Generation of Explanatory Dialogues*. MIT press, 1992.
- [50] A. Cawsey. Planning interactive explanations. *International Journal of Man-Machine Studies*, 38:169–199, 1993.
- [51] G. J. Chaitin. Information-Theoretic Limitations of Formal Systems. *Journal of the ACM*, 21(3):403–424, 1974.
- [52] G. J. Chaitin. Algorithmic information theory. *IBM Journal of Research and Development*, 21(4):350–359, 1977.
- [53] C.-H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud. Explaining Image Classifiers by Counterfactual Generation. In *Proceedings of the International Conference on Learning Representations*, 2018.

- [54] N. Chater, I. Vlaev, and M. Grinberg. A new consequence of simpson’s paradox: stable cooperation in one-shot prisoner’s dilemma from populations of individualistic learners. *Journal of Experimental Psychology: General*, 137:403–421, 2008.
- [55] B. A. Chaushi, B. Selimi, A. Chaushi, and M. Apostolova. Explainable Artificial Intelligence in Education: A Comprehensive Review. In L. Longo, editor, *Explainable Artificial Intelligence*, Communications in Computer and Information Science, pages 48–71, 2023.
- [56] C. Chen, Y. Liu, X. Zhang, and S. Xie. Scalable Explanation of Inferences on Large Graphs. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 982–987, 2019. ISSN: 2374-8486.
- [57] V. Cherkassky, X. Shao, F. M. Mulier, and V. N. Vapnik. Model complexity control for regression using VC generalization bounds. *IEEE Transactions on Neural Networks*, 10(5), 1999.
- [58] G. Chierchia, D. Fuhrmann, L. J. Knoll, B. P. Pi-Sunyer, A. L. Sakhardande, and S. J. Blakemore. The matrix reasoning item bank (MaRs-IB): novel, open-access abstract reasoning items for adolescents and adults. *Royal Society open science*, 6(190232), 2019.
- [59] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, volume 30, page 4302–4310. Curran Associates, Inc., 2017.
- [60] W. J. Clancey. The epistemology of a rule-based expert system —a framework for explanation. *Artificial Intelligence*, 20(3):215–251, 1983.
- [61] M. R. B. Clarke. A quantitative study of king and pawn against king. *Advances in Computer Chess*, 1:108–118, 1977.
- [62] W. W. Cohen. Grammatically biased learning: Learning logic programs using an explicit antecedent description language. *Artificial Intelligence*, 68(2):303–366, 1994.

- [63] K. M. Collins, A. Q. Jiang, S. Frieder, L. Wong, M. Zilka, U. Bhatt, T. Lukasiewicz, Y. Wu, J. B. Tenenbaum, W. Hart, T. Gowers, W. Li, A. Weller, and M. Jamnik. Evaluating Language Models for Mathematics through Interactions. *arXiv preprint*, 2023.
- [64] D. Corapi, A. Russo, and E. Lupu. Inductive Logic Programming in Answer Set Programming. In D. Hutchison, T. Kanade, J. Kittler, et al., editors, *Inductive Logic Programming*, volume 7207, pages 91–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [65] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (4th ed.)*. MIT Press and McGraw-Hill, 2022.
- [66] P. T. Cox and T. Pietrzykowski. Causes for events: their computation and applications. In J. Siekmann, editor, *Lecture Notes in Computer Science: Proceedings of the 8th International Conference on Automated Deduction*, page 608–621. Springer-Verlag, 1986.
- [67] R. C. Craig. Directed versus independent discovery of established relations. *Journal of Educational Psychology*, 47:223–234, 1956.
- [68] M. W. Craven and J. W. Shavlik. Using sampling and queries to extract rules from trained neural networks. In *Proceedings of the 11th International Conference on Machine Learning*, page 37–45, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
- [69] M. W. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, page 24–30, Cambridge, MA, USA, 1995. MIT Press.
- [70] A. Cropper. *Efficiently Learning Efficient Programs*. Ph.D. thesis, Imperial College London, 2017.
- [71] A. Cropper. Playgol: Learning programs through play. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 6074–6080, 2019.
- [72] A. Cropper, S. Dumancic, R. Evans, and S. H. Muggleton. Inductive logic programming at 30. *Machine Learning*, 111:147–172, 2021.
- [73] A. Cropper and R. Morel. Learning programs by learning from failures. *Machine Learning*, 2021.

- [74] A. Cropper and S. H. Muggleton. Learning efficient logical robot strategies involving composable objects. In *Proceedings of the 24th International Conference on Artificial Intelligence*, page 3423–3429, 2015.
- [75] A. Cropper and S. H. Muggleton. Metagol system. <https://github.com/metagol/metagol>, 2016.
- [76] A. Cropper and S. H. Muggleton. Learning efficient logic programs. *Machine Learning*, 108:1063–1083, 2019.
- [77] D. D. Cummins, T. Lubart, O. Alksnis, and R. Rist. Conditional reasoning and causation. *Memory & Cognition*, 19(3):274–282, 1991.
- [78] R. Cummins. ‘How does it Work?’ vs. ‘What are the Laws?’: Two Conceptions of Psychological Explanation. In R. Cummins, editor, *The World in the Head*. Oxford University Press, 2010.
- [79] C. M. Cutillo, K. R. Sharma, L. Foschini, et al. Machine intelligence in health-care—perspectives on trustworthiness, explainability, usability, and transparency. *npj Digital Medicine*, 3, 2020.
- [80] K. Cyras, D. Letsios, R. Misener, and F. Toni. Argumentation for explainable scheduling. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, pages 2752–2759, 2019.
- [81] K. Cyras, A. Rago, E. Albini, P. Baroni, and F. Toni. Argumentative XAI: A Survey. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 4392–4399, Montreal, Canada, 2021. International Joint Conferences on Artificial Intelligence Organization.
- [82] R. M. Dawes and R. H. Thaler. Anomalies: cooperation. *Journal of Economic Perspectives*, 2:187–197, 1988.
- [83] L. De Raedt and S. Džeroski. First-order jk-clausal theories are pac-learnable. *Artificial Intelligence*, 70(1–2):375–392, 1994.

- [84] A. Dejl, P. He, P. Mangal, H. Mohsin, B. Surdu, E. Voinea, E. Albini, P. Lertvitayakumjorn, A. Rago, and F. Toni. Argflow: A Toolkit for Deep Argumentative Explanations for Neural Networks. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, pages 1761–1763, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.
- [85] R. B. Dekker, F. Otto, and C. Summerfield. Curriculum learning for human compositional generalization. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 119, 2022.
- [86] R. Devidze, F. Mansouri, L. Haug, Y. Chen, and A. Singla. Understanding the power and limitations of teaching with imperfect knowledge. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 2647–2654, 2020.
- [87] Z. Dienes and J. Perner. A theory of implicit and explicit knowledge. *Behavioral and Brain Sciences*, 22(5):735–808, 1999.
- [88] R. Drath and A. Horch. Industrie 4.0: hit or hype? *IEEE Industrial Electronics Magazine*, 8(2):56–58, 2014.
- [89] S. Džeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine Learning*, 43:7–52, 2001.
- [90] T. Ellman. Explanation-based learning: A survey of programs and perspectives. *ACM Computing Surveys*, 21(2):163–221, 1989.
- [91] R. Evans, M. Bošnjak, L. Buesing, et al. Making Sense of Raw Input. *Artificial Intelligence*, 299(C):103521, 2021.
- [92] R. Evans and E. Grefenstette. Learning Explanatory Rules from Noisy Data, 2018.
- [93] Z. T. Fernando, J. Singh, and A. Anand. A study on the Interpretability of Neural Retrieval Models using DeepSHAP. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 1005–1008, New York, NY, USA, 2019.

- [94] K. Forbus. Software social organisms: Implications for measuring AI progress. *AI Magazine*, 37:85–90, 2016.
- [95] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the 15th International Conference on Machine Learning*, page 144–151, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.
- [96] A. A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explorations Newsletter*, 15:1–10, 2014.
- [97] Z. Gao, C. Ries, H. U. Simon, and S. Zilles. Preference-based teaching. *Journal of Machine Learning Research*, 18:1012–1043, 2017.
- [98] P. Gardenfors. *Belief revision*. Cambridge University Press, 1992.
- [99] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. New York: Freeman, 1979.
- [100] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. Clingo = ASP + Control: Preliminary Report. *Technical Communications of the 13th International Conference on Logic Programming*, 14:1–9, 2014.
- [101] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080, 1988.
- [102] M. Genesereth and V. K. Chaudhri. *Introduction to Logic Programming*. Springer Cham, 2020.
- [103] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40(1):11–61, 1989.
- [104] D. Gentner and R. Landers. Analogical reminding: a good match is hard to find. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, page 607–613, 1985.
- [105] C. Ginet. In defense of a non-causal account of reasons explanations. *The Journal of Ethics*, 12:229–237, 2008.

- [106] M. Gleicher. A framework for considering comprehensibility in modeling. *Big data*, 4(2):75–88, 2016.
- [107] E. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.
- [108] S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50:20–31, 1995.
- [109] H. H. Goldstine and J. Neumann. Planning and coding of problems for an electronic computing instrument, part II, volume 2. In *John von Neumann Collected Works, Volume V: Design of Computers, Theory of Automata and Numerical Analysis*, pages 152–214. Pergamon Press, Oxford, England, 1963.
- [110] Y. Goyal, Z. Wu, J. Ernst, et al. Counterfactual Visual Explanations. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2376–2384. PMLR, 2019. ISSN: 2640-3498.
- [111] R. Guidotti, A. Monreale, S. Ruggieri, et al. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):1–42, 2018.
- [112] D. Gunning, M. Stefik, J. Choi, and others. XAI-explainable artificial intelligence. *Science Robotics*, 4(37), 2019.
- [113] B. Hancock, A. Bordes, P. Mazare, et al. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, page 3667–3684, 2019.
- [114] G. H. Harman. The Inference to the Best Explanation. *Philosophical Review*, 74(1):88–95, 1965.
- [115] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *24th Annual Symposium on Foundations of Computer Science*, pages 439–445, 1983. ISSN: 0272-5428.
- [116] M. D. Hauser, N. Chomsky, and W. T. Fitch. The faculty of language: what is it, who has it, and how did it evolve? *Science*, 298:1569–1579, 2002.

- [117] D. Hilbert and W. Ackermann. Principles of mathematical logic. *American Mathematical Society*, 69, 1950.
- [118] D. J. Hilton. Conversational processes and causal explanation. *Psychological Bulletin*, 107:65–81, 1990.
- [119] D. J. Hilton and B. R. Slugoski. Knowledge-based causal attribution: The abnormal conditions focus model. *Psychological Review*, 93:75–88, 1986.
- [120] M. Hind, D. Wei, M. Campbell, et al. Ted: Teaching ai to explain its decisions. *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 2019.
- [121] J. R. Hobbs. Abduction in natural language understanding. In *The Handbook of Pragmatics*. Stevenage: Peter Peregrinus, 2008.
- [122] C. Hocquette. *Efficient Instance and Hypothesis Space Revision in Meta-Interpretive Learning*. Ph.D. thesis, Imperial College London, 2022.
- [123] R. R. Hoffman and G. Klein. Explaining Explanation, Part 1: Theoretical Foundations. *IEEE Intelligent Systems*, 32(3):68–73, 2017.
- [124] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman. Measures for explainable AI: Explanation goodness, user satisfaction, mental models, curiosity, trust, and human-AI performance. *Frontiers in Computer Science*, 5, 2023.
- [125] K. J. Holyoak and K. Koh. Surface and structural similarity in analogical transfer. *Memory and Cognition*, 15(4):332–340, 1987.
- [126] F. Hsu. IBM’s Deep Blue Chess grandmaster chips. *IEEE Micro*, 19(2):70–81, 1999.
- [127] L. Hu, R. Wu, T. Li, and L. Wang. Quadratic upper bound for recursive teaching dimension of finite VC classes. In *Proceedings of the 30th Conference on Learning Theory, COLT*, page 1147–1156, 2017.
- [128] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule-based predictive models. *Decision Support Systems*, 51:141–154, 2011.

- [129] P. N. Johnson-Laird. *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Harvard University Press, USA, 1986.
- [130] P. N. Johnson-Laird. Deductive reasoning. *WIREs Cognitive Science*, 1(1):8–17, 2010.
- [131] F. Jotterand and C. Bosco. Keeping the “human in the loop” in the age of artificial intelligence. *Science and Engineer Ethics*, 26:2455–2460, 2020.
- [132] T. Kaminski, T. Eiter, and K. Inoue. Exploiting Answer Set Programming with External Sources for Meta-Interpretive Learning. *Theory and Practice of Logic Programming*, 18(3-4):571–588, 2018.
- [133] M. Kamkar, N. Shahmehri, and P. Fritzson. Bug localization by algorithmic debugging and program slicing. In *International Workshop on Programming Language Implementation and Logic Programming*, page 60–74, 1990.
- [134] M. J. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, 1994.
- [135] S. T. Kedar-Cabelli and L. T. Mccarty. Explanation-based generalization as resolution theorem proving. In *Proceedings of the 4th International Workshop on Machine Learning*, pages 383–389, 1987.
- [136] B. Kim, C. Rudin, and J. Shah. The Bayesian case model: a generative approach for case-based reasoning and prototype classification. In *Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS’14*, pages 1952–1960, Cambridge, MA, USA, 2014. MIT Press.
- [137] S. S. Y. Kim, E. A. Watkins, O. Russakovsky, R. Fong, and A. Monroy-Hernández. "Help Me Help the AI": Understanding How Explainability Can Support Human-AI Interaction. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI '23*, pages 1–17, New York, NY, USA, 2023. Association for Computing Machinery.
- [138] R. D. King, K. E. Whelan, F. M. Jones, et al. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247–252, 2004.

- [139] J. E. Kittel. An experimental study of the effect of external direction during learning on transfer and retention of principles. *Journal of Educational Psychology*, 48:391–405, 1957.
- [140] E. Kitzelmann and U. Schmid. Inductive synthesis of functional programs: An explanation based generalization approach. *Journal of Machine Learning Research*, 7:429–454, 2006.
- [141] P. B. Kohl and N. D. Finkelstein. Student representational competence and self-assessment when solving physics problems. *Physical Review Special Topics - Physics Education Research*, 1(1), 2005.
- [142] A. N. Kolmogorov. On tables of random numbers. *Sankhya: The Indian Journal of Statistics*, 207(25):369–375, 1963.
- [143] R. Kowalski and D. Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2:227–260, 1971.
- [144] M. Krenn, R. Pollice, S. Y. Guo, et al. On scientific understanding with artificial intelligence. *Nature Reviews Physics*, 4:761–769, 2022.
- [145] H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: a joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, page 1675–1684, 2016.
- [146] M. Lange. *Because Without Cause: Non-Causal Explanations in Science and Mathematics*. Oxford University Press, 2016.
- [147] P. Langley. The changing science of machine learning. *Machine Learning*, 82:275–279, 2021.
- [148] M. Law. *Inductive Learning of Answer Set Programs*. PhD thesis, Imperial College London, UK, 2018.
- [149] M. Law, A. Russo, and K. Broda. Inductive Learning of Answer Set Programs. In E. Fermé and J. Leite, editors, *Logics in Artificial Intelligence*, Lecture Notes in Computer Science, pages 311–325, Cham, 2014. Springer International Publishing.

- [150] M. Lee, M. Srivastava, A. Hardy, et al. Evaluating Human-Language Model Interaction. *Transactions on Machine Learning Research*, 2023.
- [151] M. Lee and A. Thompson. Guided instruction in logo programming and the development of cognitive monitoring strategies among college students. *Journal of Educational Computing Research*, 16:125–144, 1997.
- [152] E. Lemke, H. Klausmeier, and C. Harris. Relationship of selected cognitive abilities to concept attainment and information processing. *Journal of Educational Psychology*, 58:27–35, 1967.
- [153] H. J. Levesque. Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17(4):355–389, 1988.
- [154] J. Li, A. Miller, S. Chopra, et al. Learning through dialogue interactions by asking questions. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.
- [155] X. Li, A. Bundy, and A. Smaill. Abc repair system for datalog-like theories. In *10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 2, pages 335–342. Scite Press, 2018.
- [156] B. Y. Lim, A. K. Dey, and D. Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 2119–2128, 2009.
- [157] T. S. Lim, W. Y. Loh, and Y. S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, 2000.
- [158] D. Lin, E. Dechter, K. Ellis, J. B. Tenenbaum, and S. H. Muggleton. Bias reformulation for one-shot function induction. In *Proceedings of the 23rd European Conference on Artificial Intelligence*, pages 525–530, 2014.
- [159] P. Lipton. Contrastive explanation. *Royal Institute of Philosophy Supplement*, 27:247–266, 1990.

- [160] Z. Lipton. The mythos of model interpretability. *Communications of the ACM*, 61:36–43, 2018.
- [161] N. F. Liu, T. Zhang, and P. Liang. Evaluating verifiability in generative search engines. *arXiv preprint*, 2023.
- [162] J. W. Lloyd. *Foundations of Logic Programming*. Springer Science and Business Media, 2012.
- [163] T. Lombrozo. The structure and function of explanations. *Trends in Cognitive Sciences*, 10:464–470, 2006.
- [164] T. Lombrozo. Explanation and Abductive Inference. In K. J. Holyoak and R. G. Morrison, editors, *The Oxford Handbook of Thinking and Reasoning*, page 0. Oxford University Press, 2012.
- [165] T. Lombrozo. Explanatory Preferences Shape Learning and Inference. *Trends in Cognitive Sciences*, 20(10):748–759, 2016.
- [166] T. Lombrozo and N. Z. Gwynne. Explanation and inference: mechanistic and functional explanations guide property generalization. *Frontiers in Human Neuroscience*, 8, 2014.
- [167] J. MacGlashan, M. K. Ho, R. T. Loftin, B. Peng, G. Wang, D. L. Roberts, M. E. Taylor, and M. L. Littman. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.
- [168] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere. A grounded interaction protocol for explainable artificial intelligence. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, page 1033–1041, 2019.
- [169] B. F. Malle. *How the Mind Explains Behavior: Folk Explanations, Meaning, and Social Interaction*. MIT Press, 2004.
- [170] B. F. Malle. Attribution theories: How people make sense of behavior. In *Theories in social psychology*, pages 72–95. Wiley Blackwell, Hoboken, NJ, US, 2011.

- [171] F. Mansouri, Y. Chen, A. Vartanian, X. Zhu, and A. Singla. Preference-based batch and sequential teaching: towards a unified view of models. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 9199–9209, Red Hook, NY, USA, 2019. Curran Associates Inc.
- [172] D. W. Massaro and N. Cowan. Information processing models: Microscopes of the mind. *Annual Review of Psychology*, 44:383–425, 1993.
- [173] R. E. Mayer. 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, 2004.
- [174] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, page 165–172, 2013.
- [175] J. McCarthy. What is artificial intelligence? Available electronically at [http:// www-formal.stanford.edu/jmc/whatisai/whatisai.html](http://www-formal.stanford.edu/jmc/whatisai/whatisai.html). Accessed: 2023-10-15.
- [176] J. McCarthy. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, 1959.
- [177] D. E. Meltzer. Relation between students’ problem-solving performance and representational format. *American Journal of Physics*, 73(5):463–478, 2005.
- [178] W. Meulemans, N. H. Riche, B. Speckmann, B. Alper, and T. Dwyer. KelpFusion: A Hybrid Set Visualization Technique. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1846–1858, 2013.
- [179] R. Michalski. A theory and methodology of inductive learning. *Machine Learning: Symbolic Computation*, 1983.
- [180] D. Michie. Experiments on the mechanization of game-learning part I. Characterization of the model and its parameters. *The Computer Journal*, 6:232–236, 1963.
- [181] D. Michie. “Memo” Functions and Machine Learning. *Nature*, 218(5136):19–22, 1968.
- [182] D. Michie. Experiments on the mechanization of game-learning part II. Rule-based learning and the human window. *The Computer Journal*, 25:105–113, 1982.

- [183] D. Michie. Information and complexity in chess. *Advances in Computer Chess*, 3:139–143, 1982.
- [184] D. Michie. Inductive rule generation in the context of the fifth generation. *Machine Learning Workshop*, 1983.
- [185] D. Michie. Measuring the knowledge content of expert programs. In *On Machine Intelligence*. Ellis Horwood, Chichester, second revised edition edition, 1986.
- [186] D. Michie. Machine learning in the next five years. In *Proceedings of the 3rd European Working Session on Learning*, pages 107–122. Pitman, 1988.
- [187] D. Michie. Machine intelligence and the human window. *Applied Artificial Intelligence*., 5(1):1–10, 1991.
- [188] D. Michie, M. Bain, and J. Hayes-Michie. Cognitive models from sub-cognitive skills. *Knowledge-Based Systems in Industrial Control*, pages 71–99, 1990.
- [189] D. Michie and R. Camacho. Building symbolic representations of intuitive real-time skills from performance data. *Machine Intelligence*, 13:385–418, 1994.
- [190] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.
- [191] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.
- [192] D. Minh, H. Wang, Y. Li, and T. N. Nguyen. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, 2021.
- [193] J. B. Misyak and N. Chater. Virtual bargaining: A theory of social decision-making. *Philosophical Transactions of the Royal Society B*, 369, 2014.
- [194] J. B. Misyak and N. Chater. The spontaneous creation of systems of conventions. In *Proceedings of the 39th Annual Meeting of the Cognitive Science Society*, 2017.
- [195] J. B. Misyak, T. Noguchi, and N. Chater. Instantaneous conventions: the emergence of flexible communicative signals. *Psychological Science*, 27(12):1550–1561, 2016.

- [196] T. M. Mitchell. The need for biases in learning generalizations. Technical report, Rutgers University, 1980.
- [197] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
- [198] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [199] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: a unifying view. *Machine Learning*, 1(1):47–80, 1986.
- [200] V. Mnih, K. Kavukcuoglu, D. Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.
- [201] D. Moody. The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, 2009.
- [202] J. D. Moore and C. L. Paris. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational linguistics*, 19:651–694, 1993.
- [203] M. Mozina, J. Zabkar, and I. Bratko. Argument based machine learning. *Artificial Intelligence*, 171:922–937, 2007.
- [204] S. Muggleton, L. De Raedt, D. Poole, et al. ILP turns 20. *Machine Learning*, 86(1):3–23, Jan. 2012.
- [205] S. H. Muggleton. Inductive logic programming. *New Generation Computing*, 8:295–318, 1991.
- [206] S. H. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
- [207] S. H. Muggleton. Hypothesizing an algorithm from one example: the role of specificity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 381(2251), 2023.

- [208] S. H. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the 5th International Conference on Machine Learning*, pages 339–352, 1988.
- [209] S. H. Muggleton, W.-Z. Dai, C. Sammut, A. Tamaddoni-Nezhad, J. Wen, and Z.-H. Zhou. Meta-Interpretive Learning from Noisy Images. *Machine Learning*, 107(7):1097–1118, 2018.
- [210] S. H. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629–679, 1994.
- [211] S. H. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*, pages 368–381, 1990.
- [212] S. H. Muggleton and C. Hocquette. Machine discovery of comprehensible strategies for simple games using meta-interpretive learning. *New Generation Computing*, 37:203–217, 2019.
- [213] S. H. Muggleton, D. Lin, N. Pahlavi, and A. Tamaddoni-Nezhad. Meta-interpretive learning: application to grammatical inference. *Machine Learning*, 94:25–49, 2014.
- [214] S. H. Muggleton, D. Lin, and A. Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. *Machine Learning*, 100:49–73, 2015.
- [215] S. H. Muggleton, U. Schmid, C. Zeller, A. Tamaddoni-Nezhad, and T. Besold. Ultra-strong machine learning: Comprehensibility of programs learned with ILP. *Machine Learning*, 107:1119–1140, 2018.
- [216] B. B. Murdock. The serial position effect of free recall. *Journal of Experimental Psychology*, 64(5):482–488, 1962.
- [217] G. L. Murphy and M. E. Lassaline. *Hierarchical structure in concepts and the basic level of categorization*, page 93–131. Psychology Press, MIT Press, 1997.
- [218] P. M. Murphy and M. J. Pazzani. ID2-of-3: Constructive induction of m-of-n concepts for discriminators in decision trees. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 183–187. Morgan Kaufmann, 1991.

- [219] R. Mühlhoff. Human-aided artificial intelligence: Or, how to run large computations in human brains? Toward a media sociology of machine learning. *New Media & Society*, 22(10):1868–1884, 2020.
- [220] M. Nardo, D. Forino, and T. Murino. The evolution of man–machine interaction: the role of human in industry 4.0 paradigm. *Production and Manufacturing Research*, 8(1):20–34, 2020.
- [221] J. F. Nash. Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences*, volume 36, pages 48–49, 1950.
- [222] M. Nauta, J. Trienes, S. Pathak, et al. From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI. *ACM Computing Surveys*, 55(13s):295:1–295:42, 2023.
- [223] A. Newell. *Unified Theories of Cognition*. Harvard University Press, USA, 1990.
- [224] J. Ni, T. Young, V. Pandelea, et al. Recent advances in deep learning based dialogue systems: a systematic survey. *Artificial Intelligence Review*, 2022.
- [225] S. Nienhuys-Cheng and R. Wolf. *Foundations of Inductive Logic Programming*. Springer-Verlag New York, Inc., 1997.
- [226] H. Nilsson and P. Fritzson. Algorithmic debugging for lazy functional languages. In *Programming Language Implementation and Logic Programming*, page 385–399, 1992.
- [227] L. Novick and K. Holyoak. Mathematical problem solving by analogy. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 17:398–415, 1991.
- [228] M. R. Oaksford and N. Chater. Reasoning theories and bounded rationality. In *Rationality: Psychological and philosophical perspectives*, International library of psychology, pages 31–60. Taylor & Frances/Routledge, Florence, KY, US, 1993.
- [229] M. R. Oaksford and N. Chater. *Rationality In An Uncertain World: Essays In The Cognitive Science Of Human Understanding*. Routledge, London, 1998.
- [230] A. Passerini, P. Frasconi, and L. De Raedt. Kernels on prolog proof trees: Statistical learning in the ilp setting. *Journal of Machine Learning Research*, 7(11):307–342, 2006.

- [231] R. E. Passingham and S. P. Wise. *The Neurobiology of the Prefrontal Cortex: Anatomy, Evolution, and the Origin of Insight*. Oxford University Press, 2012.
- [232] J. Pearl and D. Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books., 2018.
- [233] C. S. Peirce. *Collected papers of Charles Sanders Peirce*. Harvard University Press, 1974.
- [234] J. W. Peirce, J. R. Gray, S. Simpson, M. R. MacAskill, R. Höchenberger, H. Sogo, E. Kastman, and J. Lindeløv. Psychopy2: experiments in behavior made easy. *Behavior Research Methods*, 2019.
- [235] D. C. Penn, K. J. Holyoak, and D. J. Povinelli. Darwin’s mistake: Explaining the discontinuity between human and nonhuman minds. *Behavioral and Brain Sciences*, 31:109–130, 2008.
- [236] G. D. Plotkin. A note on inductive generalization. In *Machine Intelligence*, pages 153–163. American Elsevier, 1970.
- [237] K. Popper. *The Logic of Scientific Discovery*. Routledge, 2002.
- [238] E. Puiutta and E. Veith. Explainable reinforcement learning: A survey. In A. Holzinger, P. Kieseberg, A. Tjoa, and E. Weippl, editors, *Lecture Notes in Computer Science: Machine Learning and Knowledge Extraction*, pages 77–95. Springer International Publishing, 2020.
- [239] J. R. Quinlan. Learning efficient classification procedures and their application to chess end games. In *Machine Learning: An Artificial Intelligence Approach*, pages 463–482. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.
- [240] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.
- [241] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.
- [242] J. R. Quinlan and R. M. Cameron. Induction of logic programs: FOIL and related systems. *New Generation Computing*, 13:287–312, 1995.

- [243] J. Rabold, M. Siebers, and U. Schmid. Generating contrastive explanations for inductive logic programming based on a near miss approach. *Machine Learning*, 111:1799–1820, 2022.
- [244] A. N. Rafferty, E. Brunskill, T. L. Griffiths, and P. Shafto. Faster teaching via POMDP planning. *Cognitive science*, 40:1290–1332, 2016.
- [245] D. Raggi, G. Stapleton, A. Stockdill, M. Jamnik, G. G. Garcia, and P. Cheng. How to (re)represent it? In *Proceedings of the 32nd IEEE International Conference on Tools with Artificial Intelligence*, page 1224–1232, 2020.
- [246] D. Raggi, A. Stockdill, M. Jamnik, G. G. Garcia, H. Sutherland, and P. Cheng. Inspection and selection of representations. In C. Kaliszyk, E. Brady, A. Kohlhase, and C. Sacerdoti-Coen, editors, *Intelligent Computer Mathematics (CICM), Lecture Notes in Computer Science*, volume 11617, page 227–242. Springer, 2019.
- [247] A. Rago, O. Cocarascu, C. Bechlivanidis, D. Lagnado, and F. Toni. Argumentative explanations for interactive recommendations. *Artificial Intelligence*, 296:1–22, 2021.
- [248] A. Rago, O. Cocarascu, and F. Toni. Argumentation-Based Recommendations: Fantastic Explanations and How to Find Them. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 1949–1955, 2018.
- [249] S. K. Reed, C. C. Ackinclose, and A. A. Voss. Selecting analogous problems: Similarity versus inclusiveness. *Memory and Cognition*, 18(1):83–98, 1990.
- [250] S. K. Reed and C. A. Bolstad. Use of examples and procedures in problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 17(4):753, 1991.
- [251] B. Rehder. A causal-model theory of conceptual representation and categorization. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, 29(6):1141–1159, 2003.
- [252] M. T. Ribeiro, S. Singh, and C. Guestrin. "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*, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.
- [253] M. Robnik-Šikonja and M. Bohanec. Perturbation-Based Explanations of Prediction Models. In J. Zhou and F. Chen, editors, *Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent*, Human–Computer Interaction Series, pages 159–175. Springer International Publishing, Cham, 2018.
- [254] P. Rodgers, L. Zhang, and H. Purchase. Wellformedness Properties in Euler Diagrams: Which Should Be Used? *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1089–1100, 2012.
- [255] P. Rodríguez, S. Heras, J. Palanca, et al. An educational recommender system based on argumentation theory. *AI Communications*, 30(1):19–36, 2017.
- [256] L. Rolf, G. Kern-Isberner, and G. Brewka. Argumentation-Based Explanations for Answer Sets Using ADF. In M. Balduccini, Y. Lierler, and S. Woltran, editors, *Logic Programming and Nonmonotonic Reasoning*, Lecture Notes in Computer Science, pages 89–102, Cham, 2019. Springer International Publishing.
- [257] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215, 2019.
- [258] O. Russakovsky, J. Deng, H. Su, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.
- [259] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.
- [260] S. J. Russell. Preliminary steps toward the automation of induction. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 477–484, 1986.
- [261] B. Saket, A. Endert, and C. Demiralp. Task-Based Effectiveness of Basic Visualizations. *IEEE transactions on visualization and computer graphics*, 25(7):2505–2512, 2019.
- [262] D. Sally. Conversation and cooperation in social dilemmas: a meta-analysis of experiments from 1958 to 1992. *Rationality and Society*, 7:58–92, 1995.

- [263] B. Sanchez-Lengeling, J. Wei, B. Lee, E. Reif, P. Wang, W. Qian, K. McCloskey, L. Colwell, and A. Wiltchko. Evaluating Attribution for Graph Neural Networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 5898–5910. Curran Associates, Inc., 2020.
- [264] U. Schmid. Interactive learning with mutual explanations in relational domains. In *Human-Like Machine Intelligence*. Oxford University Press, 2021.
- [265] U. Schmid and J. Carbonell. Empirical evidence for derivational analogy. In *Proceedings of the 21st Annual Conference of the Cognitive Science Society*, 2000.
- [266] U. Schmid and B. Finzel. Mutual explanations for cooperative decision making in medicine. *KI-Künstliche Intelligenz*, 34(2):227–233, 2020.
- [267] U. Schmid and E. Kitzelmann. Inductive rule learning on the knowledge level. *Cognitive Systems Research*, 12:237–248, 2011.
- [268] U. Schmid, C. Zeller, T. Besold, A. Tamaddoni-Nezhad, and S. H. Muggleton. How does predicate invention affect human comprehensibility? In *Proceedings of the 26th International Conference on Inductive Logic Programming*, pages 52–67, 2017.
- [269] N. Sendi, N. Abchiche-Mimouni, and F. Zehraoui. A new Transparent Ensemble Method based on Deep learning. *Procedia Computer Science*, 159:271–280, 2019.
- [270] R. Setiono and W. K. Leow. Fernn: an algorithm for fast extraction of rules from neural networks. *Applied Intelligence*, 12:15–25, 2000.
- [271] B. Settles. *Active Learning*. Springer Cham, 2012.
- [272] D. Shahaf and E. Amir. Towards a Theory of AI Completeness. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, 2007.
- [273] A. Shapiro and T. Niblett. Automatic induction of classification rules for a chess endgame. In M. Clarke, editor, *Advances in Computer Chess*, volume 3, pages 73–91. Pergamon, Oxford, 1982.
- [274] E. Y. Shapiro. *Algorithmic Program Debugging*. Ph.D. thesis, Yale University, 1982.

- [275] T. B. Sheridan. Human-robot interaction: status and challenges. *Human Factors*, 58(4):525–532, 2016.
- [276] E. H. Shortliffe. A rule-based computer program for advising physicians regarding antimicrobial therapy selection. In *Proceedings of the Annual ACM Conference*, ACM '74, page 739, New York, NY, USA, 1974. Association for Computing Machinery.
- [277] D. Silver, A. Huang, C. Maddison, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.
- [278] H. A. Simon. *Decision making: Descriptive, normative, and prescriptive interactions*, chapter Rationality as process and as product of thought, page 58–77. Cambridge University Press, Cambridge, UK, 1988.
- [279] H. A. Simon and J. R. Hayes. The understanding process: Problem isomorphs. *Cognitive Psychology*, 8:165–190, 1976.
- [280] M. Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology Boston, 2006.
- [281] K. Sokol and P. Flach. Explainability fact sheets: a framework for systematic assessment of explainable approaches. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 56–67, New York, NY, USA, 2020. Association for Computing Machinery.
- [282] C. D. Souza. *The Semiotic Engineering of Human-Computer Interaction*. MIT Press, 2005.
- [283] C. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15(1), 1904.
- [284] D. Sperber and H. Mercier. *The Enigma of Reason: A New Theory of Human Understanding*. Allen Lane, 2017.
- [285] A. Srinivasan. The aleph manual. *Machine Learning at the Computing Laboratory, Oxford University*, 2001.

- [286] M. J. E. Sternberg and S. H. Muggleton. Structure activity relationships (SAR) and pharmacophore discovery using inductive logic programming (ILP). *QSAR and Combinatorial Science*, 22(5):527–532, 2003.
- [287] S. Stumpf, A. Bussone, and D. O’sullivan. Explanations considered harmful? user interactions with machine learning systems. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, 2016.
- [288] J. Sweller. Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science*, 12(2):257–285, 1988.
- [289] J. Sweller, P. A. Kirschner, and R. E. Clark. Why minimally guided teaching techniques do not work: A reply to commentaries. *Educational Psychologist*, 42(2):115–121, 2007.
- [290] P. Tamagnini, J. Krause, A. Dasgupta, and E. Bertini. Interpreting black-box classifiers using instance-level visual explanations. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 1–6, 2017.
- [291] S. Tärnlund. Horn clause computability. *BIT Numerical Mathematics*, 17(2):215–226, 1977.
- [292] E. J. Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature Medicine*, 25:44–56, 2019.
- [293] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Society*, volume 2, page 230–265, 1937.
- [294] A. M. Turing. *Systems of Logic Based on Ordinals*. Ph.D. thesis, Princeton University, 1938.
- [295] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.
- [296] M. T. Ullman. Contributions of memory circuits to language: the declarative/procedural model. *Cognition*, 92(1):231–270, 2004.
- [297] T. Urbančič and I. Bratko. Reconstructing human skill with machine learning. In *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 498–502, 1994.

- [298] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
- [299] M. van Emden and A. Vellino. From Chinese Room to Human Window. *ICGA Journal*, 33:127–139, 2010.
- [300] I. Van Rooij. The Tractable Cognition Thesis. *Cognitive Science*, 32(6):939–984, 2008.
- [301] I. van Rooij and T. Wareham. Parameterized Complexity in Cognitive Modeling: Foundations, Applications and Opportunities. *The Computer Journal*, 51(3):385–404, 2008.
- [302] M. Van Someren. A perspective on machine learning and comprehensibility from knowledge acquisition. In *Proceedings IJCAI-95 Workshop on Machine Learning and Comprehensibility*, pages 21–26, 1995.
- [303] A. Vassiliades, N. Bassiliades, and T. Patkos. Argumentation and explainable artificial intelligence: a survey. *The Knowledge Engineering Review*, 36:e5, 2021.
- [304] A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, volume 30, page 6000–6010. Curran Associates, Inc., 2017.
- [305] X. Wang, S. Wang, X. Liang, et al. Deep Reinforcement Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2022.
- [306] C. Watkins. *Learning from Delayed Rewards*. Ph.D. thesis, University of Cambridge, 1989.
- [307] L. Wells and T. Bednarz. Explainable AI and reinforcement learning - a systematic review of current approaches and trends. *Frontiers in Artificial Intelligence*, 4, 2021.
- [308] J. Whitehill and J. Movellan. Approximately optimal teaching of approximately optimal learners. *IEEE Transactions on Learning Technologies*, pages 152–164, 2018.
- [309] M. Wiering and M. Van Otterlo, editors. *Reinforcement Learning: State-of-the-Art*, volume 12 of *Adaptation, Learning, and Optimization*. Springer, Berlin, Heidelberg, 2012.

- [310] D. A. Wilkenfeld and T. Lombrozo. Inference to the Best Explanation (IBE) Versus Explaining for the Best Inference (EBI). *Science & Education*, 24(9-10):1059–1077, 2015.
- [311] R. A. Williamson, V. K. Jaswal, and A. N. Meltzoff. Learning the Rules: Observation and Imitation of a Sorting Strategy by 36-Month-Old Children. *Developmental psychology*, 46(1):57–65, 2010.
- [312] A. C. Yao. Decision tree complexity and Betti numbers. *Journal of Computer and System Sciences*, 55(1):36–43, 1997.
- [313] T. Yeo, P. Kamalaruban, A. Singla, A. Merchant, T. Asselborn, L. Faucon, P. Dillenbourg, and V. Cevher. Iterative classroom teaching. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence and 31st Innovative Applications of Artificial Intelligence Conference and 9th AAAI Symposium on Educational Advances in Artificial Intelligence*, 2019.
- [314] T. Zahavy, N. B. Zrihem, and S. Mannor. Graying the black box: understanding DQNs. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.
- [315] V. F. Zambaldi, D. C. Raposo, A. Santoro, et al. Deep reinforcement learning with relational inductive biases. In *Proceedings of the International Conference on Learning Representations*, 2019.
- [316] C. Zednik and H. Boelsen. Scientific exploration and explainable artificial intelligence. *Minds and Machines*, 32:219–239, 2022.
- [317] C. Zeller and U. Schmid. Automatic generation of analogous problems to help resolving misconceptions in an intelligent tutor system for written subtraction. In *Workshops Proceedings for the 24th International Conference on Case-Based Reasoning*, pages 108–117, 2016.
- [318] C. Zeller and U. Schmid. A human like incremental decision tree algorithm: Combining rule learning, pattern induction, and storing examples. In *Learning, Knowledge, Data, Analytics*, 2017.

- [319] J. Zhang. The nature of external representations in problem solving. *Cognitive Science*, 21(2):179–217, 1997.
- [320] W. X. Zhao, K. Zhou, J. Li, et al. A survey of large language models. *arXiv preprint*, 2023.
- [321] X. Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, volume 29, 2015.