Behavior Research Methods (2024) 56:1065–1103
https://doi.org/10.3758/s13428-023-02062-z

Automatic discovery and description of human planning strategies

Julian Skirzyński[1,2] · **Yash Raj Jain**[1] · **Falk Lieder**[1]

Accepted: 6 January 2023 / Published online: 30 May 2023

Abstract

Scientific discovery concerns finding patterns in data and creating insightful hypotheses that explain these patterns. Traditionally, each step of this process required human ingenuity. But the galloping development of computer chips and advances in artificial intelligence (AI) make it increasingly more feasible to automate some parts of scientific discovery. Understanding human planning is one of the fields in which AI has not yet been utilized. State-of-the-art methods for discovering new planning strategies still rely on manual data analysis. Data about the process of human planning is often used to group similar behaviors together. Researchers then use this data to formulate verbal descriptions of the strategies which might underlie those groups of behaviors. In this work, we leverage AI to automate these two steps of scientific discovery. We introduce a method for automatic discovery and description of human planning strategies from process-tracing data collected with the Mouselab-MDP paradigm. Our method utilizes a new algorithm, called Human-Interpret, that performs imitation learning to describe sequences of planning operations in terms of a procedural formula and then translates that formula to natural language. We test our method on a benchmark data set that researchers have previously scrutinized manually. We find that the descriptions of human planning strategies that we obtain automatically are about as understandable as human-generated descriptions. They also cover a substantial proportion of relevant types of human planning strategies that had been discovered manually. Our method saves scientists' time and effort, as all the reasoning about human planning is done automatically. This might make it feasible to more rapidly scale up the search for yet undiscovered cognitive strategies that people use for planning and decision-making to many new decision environments, populations, tasks, and domains. Given these results, we believe that the presented work may accelerate scientific discovery in psychology, and due to its generality, extend to problems from other fields.

Keywords Automatic scientific discovery · Decision-making · Planning · Interpretable strategy discovery · Process-tracing

Introduction

Scientific discovery is a product of scientific inquiry that allows generating and corroborating new insightful hypotheses. In the early days, scientific discovery was seen as a prescriptive method for arriving at new knowledge, by gathering information and refining it with new experiments (see Bacon's *Novum Organum* (Bacon, 1878) or Newton's *Philosophiae Naturalis Principia Mathematica* (Newton, 1687)). The more established philosophical tradition argued that there exists a clear demarcation between the so-called context of discovery and the context of justification (Whewell, 1840; Reichenbach, 1938; Popper, 1935). The former would be a product of a logically unfathomable mental process: the "eureka" moment, that, if anything, could be rather studied by psychology. The latter would concern the proper verification and justification of the discovered theory, and would indeed have a formal structure. However, such an account leaves scientists at the mercy of having a eureka moment. This division was thus challenged. Kuhn (1962), for instance, saw scientific discovery as a complex process of paradigm changes, where increasing amounts of findings that disagree with the current paradigm lead to its change. Importantly for this work, this division was also challenged by early work on AI for problem-solving (Simon & Newell, 1971; Simon, 1973). Therein, the discovery was understood as searching the problem space from the initial state representing current

✉ Julian Skirzyński
julian.skirzynski@tuebingen.mpg.de

[1] Max Planck Institute for Intelligent Systems, Tübingen, Germany

[2] University of California, San Diego, CA 92093, USA

knowledge to the desired goal state. The states transition from one to another by applying simple operators with a predefined meaning. The process of finding the shortest sequence of applied operators would create rules that could later be looked at by a human to determine what search heuristic has been used. The found search heuristic would determine the method for scientific discovery. Regardless of whether you agree with the distinction between discovery versus justification, these works showed that some steps involved in scientific discovery can be automated. Further work in such automation moved beyond the philosophical debate, with the main motivation becoming to aid scientists in their research (Addis et al., 2016).

In this article, we introduce a computational method for assisting scientists in studying human planning. Understanding how people plan is a difficult and time-consuming endeavor. Previous efforts to figure out which strategies and processes people use to make decisions and plan usually entailed manual analysis of the data (Payne, 1993; Willemsen & Johnson, 2011; Callaway et al., 2017; Callaway et al., 2020). The very act of finding the strategies has been thus left to researchers' ingenuity to discover the right patterns in the data. Moreover, previous work has largely failed to characterize people's strategies in detail (but see Jain et al., 2022; Agrawal et al., 2020; Peterson et al., 2021). Our method substantially simplifies that process by using an algorithm called Human-Interpret. Human-Interpret automatically discovers and describes human planning strategies externalized in process-tracing experiments. It achieves that by resorting only to a dictionary of logical primitives and their natural language counterparts that capture basic actions in those experiments. Human-Interpret imitates the input information-gathering operations in terms of procedural logic formulas and translates these formulas into natural language with the input dictionary. Our method runs Human-Interpret a number of times and selects among the strategies it found by applying a majority heuristic. To evaluate our method and, particularly, the Human-Interpret algorithm, we applied it to a planning task where people spend resources on inspecting nodes in a graph in order to find the most rewarding path to traverse. Our method generated descriptions of 4 human planning strategies for that task that were on par with the descriptions created through laborious manual analysis by Jain et al. (2022). Moreover, despite representing only a fraction of all strategies, the found strategies covered half of the relevant cases. Given these results, we believe that the presented work has the potential to facilitate scientific discovery in psychology and perhaps scientific discovery in general. Scientists can now use the help of AI not only for testing their hypotheses about how people make decisions but also for generating them. To apply our approach to new problems, such as multi-alternative risky-choice (Lieder et al., 2017; Peterson et al., 2021), it suffices to run new planning externalization studies and create a new dictionary of logical primitives.

The outline of the article is as follows: We begin by providing background information and summarizing related work pertaining to our method and the benchmark problem used in "Background and related work". In the next section, we describe the whole pipeline of our method for the automatic discovery and description of human planning strategies. "Evaluating our method for discovering human planning strategies" shows the results of our test on the benchmark problem where we compare our automated pipeline with a standard manual approach. Lastly, "General discussion" discusses opportunities for applying our method and directions for future work.

Background and related work

In this section, we detail how AI was used in scientific discovery as well as in scientific discovery for decision making and planning specifically. Additionally, we present the methodology used to measure planning adapted in our studies and a state-of-the-art approach for discovering human planning strategies that we compared to. Lastly, we also describe an important part of our framework for automatic scientific discovery, which is an algorithm that describes planning strategies in an interpretable way by only using demonstrations of these strategies.

Research on automating scientific discovery

One line of research involves the field of computational scientific discovery (Džeroski et al., 2007; Sozou et al., 2017) which models the discovery problem mathematically and advances the discovery of laws or relations using artificial intelligence (AI). BACON (Langley et al., 1987), for instance, was a system for inducing numeric laws from experimental data. Having dependent and independent variables, it created taxonomies of these variables by clustering equally-valued dependent variables and defining new variables as products or ratios of independent variables. Then, Langley et al. (1983) created GLAUBHER that was formulating qualitative laws over the categories in the taxonomy, and STAHL which produced structural theories based on the data leading to anomalous behavior of existing theories (c.f. Kuhn, 1962). Addis et al. (2016) suggested representing theories as programs and using genetic search for best theories. Other systems such as PHINEAS, COAST or IDS are described at length in a review paper by Shrager & Langley (1990). An overview work by Džeroski et al. (2007) shows an extension of this field to mathematical modeling: the background knowledge is represented as generic processes, the data takes the form of time-series,

and discovery takes place by deriving sets of explanatory differential equations.

Another line of research that used AI to help scientists in their endeavors was automatic experimental design. This approach follows the mathematical foundations of design optimization, in which the expected information gain of an experiment defines its usefulness in testing a hypothesis (Myung et al., 2013). Vincent & Rainforth (2017) used this approach to automate the creation of intertemporal choice experiments and risky choice experiments. Ouyang et al. (2016, 2018) went a step further and created a system to automatically find informative scientific experiments in general. Their method expected a formal experiment space, expressed in terms of a probabilistic program as input, and returned a list of experiments ranked by their expected information gain. The experiment that was the highest on that list would provably provide the most information to differentiate between competing hypotheses. Foster et al. (2019) further refined the whole idea by introducing efficient expected information gain estimators.

Using artificial intelligence to understand human planning

The case for using AI in the quest of understanding human planning has been clearly made for one-shot decision-making scenarios (Agrawal et al., 2020; Bhatia & He, 2021; Peterson et al., 2021). Agrawal et al. (2020) fit simple models to the data and optimized them with respect to the regret obtained by comparing simple models' predictions to overly complex models' predictions. After the simple models' predictions converged to those of the complex models, they were used as a proper formalization of one-shot human decision strategies. Peterson et al. (2021) employed artificial neural networks to search for theories of one-shot decision-making. At first, they created a taxonomy of theories that expressed relations between available decision items (such as gambles, and whether the gambles are dependent or independent), and which covered the entire space of possible decision-making theories. Subsequently, they used neural networks to express those theories by imposing different constraints on those networks. The authors gathered a very large data set of human decisions and determined which theory is the best fit based on the networks' performance. Very recent works started going beyond one-shot decision making and showed the utility of machine learning for studying multi-step decision making, i.e. planning. We are aware of 2 such endeavors. Fang et al. (2022) suggest using the data set of human decisions and training machine learning classifiers to learn the association between features extracted from the data (e.g. reaction time), and the decision strategy provided as the label. Benchmark tests with SVM and KNN models showed that this approach is able to correctly discover strategies such as take-the-best. In the paper we compare to, Jain et al. (2022) defined a computational method that assigns planning strategies to human decision data through Bayesian inference. Here, we go further than all the mentioned articles. Although we also consider the issue of i) discovering detailed human multi-step decision strategies (planning strategies), ii) we aim to discover those strategies from data automatically without creating the initial model, whether a complex black-box model (Agrawal et al., 2020), a taxonomy of decision-making theories (Peterson et al., 2021), or a set of possible planning strategies (Fang et al., 2022; Jain et al., 2022).

Methods for measuring how people plan

People's planning for years has been an elusive process that lacked principled analysis tools. To study planning, psychologists firstly focused on one-step decisions and most often relied on educated guesses, i.e. self-constructed mathematical models of human behavior that captured the relationship between inputs and outputs of decision-making (Abelson & Levi, 1985; Westenberg & Koele, 1994; Ford et al., 1989). These methods were not error-proof because they sometimes fit conflicting models equally well to the same data or were too limited to capture the whole decision-making process (Ford et al., 1989; Riedl et al., 2008). To mitigate these drawbacks, scientists have developed *process-tracing* methods that captured the process used in decision-making by also analyzing the context in which each of the steps was taken before reaching a decision (Payne et al., 1978; Svenson, 1979). Among multiple process-tracing paradigms, such as verbal protocols (e.g. in Newell et al. 1972) or conversational protocols (e.g. in Huber et al., 1997), some process-tracing paradigms were computerized and followed human choices by facing people with artificial tasks which required them to take a series of actions before making the final decision. One such process-tracing paradigm called the Mouselab (Payne et al., 1988) paradigm which is used for one-step decision-making tasks was later adapted for studying planning under the name of Mouselab-MDP (Callaway et al., 2017) paradigm. Since we used this methodology in our framework for automatic scientific discovery for planning, we now introduce the Mouselab and Mouselab-MDP paradigms in more detail. At the end, we also introduce a state-of-the-art approach for scientific discovery for planning that uses Mouselab-MDP called the *Computational Microscope*. Since it relies on the manual analysis of data, we treat it as a baseline throughout this paper.

Mouselab

The Mouselab (Payne et al., 1988) paradigm is one of the first computerized process-tracing paradigms. Its goal is to externalize some aspects of cognitive processes taking part in decision-making by engaging people in information acquisition that helps them reach a decision. Mouselab was developed to study multi-alternative risky choice. To do so, it presents participants with an initially occluded payoff matrix whose entries can be (temporarily) revealed by clicking on them. An i, j entry of the $n \times m$ payoff matrix either hides a value for bet i under outcome j or, in the case of the first row, hides a probability for outcome j. The values are expressed in terms of dollars and they indicate the payoff a participant is expected to obtain after selecting gamble i and having outcome j to occur. Participants' task is to choose one of the available $i - 1$ gambles based on the information they gathered by revealing the entries of the matrix. The sequence of clicks externalizes a participant's decision-making process by showing which information they considered. Using the Mouselab paradigm, scientists were able to discover that people choose strategies adaptively (Payne et al., 1988). Despite its usefulness, however, Mouselab is inappropriate to study planning as selecting one gamble does not affect future gambles. Mouselab-MDP was developed to mitigate this shortcoming.

Mouselab-MDP

The Mouselab-MDP paradigm is a generalization of the Mouselab process-tracing paradigm in which a participant's information-acquisition actions affect the availability of his or her future choices (Callaway et al., 2017). By doing so and offering a way to externalize information acquisition, Mouselab-MDP is suitable to study human planning. Concretely, a single decision about choosing a gamble is replaced with a Markov Decision Process (MDP; see Definition 1), and the payoff matrix is replaced with a graphical representation of the MDP, a directed graph with initially occluded nodes.

Definition 1 (Markov Decision Process) A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is a set of states; $\mathcal{A}$ is a set of actions; $\mathcal{T}(s, a, s') = \mathbb{P}(s_{t+1} = s' \mid s_t = s, a_t = a)$ for $s \neq s' \in \mathcal{S}, a \in \mathcal{A}$ is a state transition function; $\gamma \in (0, 1)$ is a discount factor; $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is a reward function.

Clicking on the node reveals a numerical reward or punishment hidden underneath it (see Fig. 1). In the most commonly used setting of the Moueslab-MDP paradigm, a participant's goal is to find the most rewarding path for an agent to traverse from the start node to one of the terminal nodes (nodes without any out-going connections), by minimizing the number of clicks (each click has an associated cost). This formulation was used in a number of papers that studied human planning strategies (Griffiths et al., 2019; Callaway et al., 2020) and led to the creation of cognitive tutors that help people plan better (Lieder et al., 2019; Lieder et al., 2020), the creation of new, scalable and robust algorithms for hierarchical reinforcement learning (Kemtur et al., 2020; Consul et al., 2021), and helped in creating one of the first tools for analyzing human planning in more detail (Jain et al., 2022).

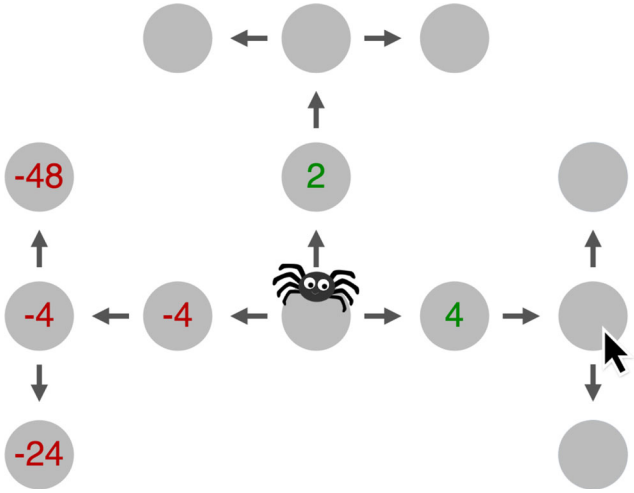

Fig. 1 The Mouselab-MDP environment we used in the paper. The environment is a connected graph of nodes (grey circles) that hide rewards or punishments. The number hidden underneath a node can be uncovered by clicking on it and paying a small fee. The goal is to traverse a path starting in the black node and ending in a node at the highest level so that the sum of rewards along the way minus the cost of clicking was as high as possible

Categorizing planning strategies via the Computational Microscope

The Mouselab-MDP paradigm gives information about the planning processes used by people (Callaway et al., 2020; Callaway et al., 2017). Past research, similar to the research on describing human decision-making, employed formal planning models (i.e. strategies) to describe the planning processes of people, and evaluated those models on data from experiments with human participants, e.g. (Botvinick et al., 2009; Huys et al., 2012; Huys et al., 2015; Callaway et al., 2018; Callaway et al., 2020). Researchers have come up with a number of planning models people could hypothetically use, including classic models of planning such as Breadth First Search, Depth First Search, models that use satisficing, etc. Recently, Jain et al. (2022) created a new computational pipeline in which

a set of manually created planning strategies are given as input to a computational method, called the Computational Microscope, to fit them to human planning data. In more detail, the method takes process-tracing data generated using the Mouselab-MDP paradigm and by using Bayesian inference, categorizes the planning operations from trials of the Mouselab-MDP paradigm into a sequence of these planning strategies. One of its features is that to categorize a trial, it incorporates all the information from all the trials before and after it. The authors of the paper performed a manual inspection of all the process-tracing data (clicks) that participants made in each trial. They first found similar click sequences based on the trials and the features of the trial and then created generalized strategies that replicated these click sequences. This led to the creation of a set of 79 strategies.

One of the drawbacks of their method is the amount of time it took them to manually go through all the participants' data and the scalability of their approach. Since our method aims to develop a method for automatic interpretation of the process-tracing data, which includes the automatic discovery of the planning strategies, we treat their set of planning strategies and their approach as a baseline to compare our framework to.

Finding interpretable descriptions of formal planning strategies (policies): AI-Interpret

Part of our framework utilizes a variant of an algorithm developed to interpret reinforcement learning (RL; see Definition 2) policies: AI-Interpret (Skirzyński et al., 2021).

Definition 2 (Reinforcement learning) Reinforcement learning (RL) is a class of methods that perform iterations over trials and evaluation on a given MDP in order to find the optimal policy π^* which maximizes the expected reward (Sutton & Barto, 2018). A deterministic policy π is a function $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that controls an agent's behavior in an MDP and a nondeterministic policy π is a function $\pi : \mathcal{S} \rightarrow Prob(\mathcal{A})$ that defines a probability distribution over the actions in the MDP. The reward r_t represents the quality of performing action a_t in state s_t. The cumulative return of a policy is a sum of its discounted rewards obtained in each step of interacting with the MDP, i.e. $G_t^\pi = \sum_{i=t}^{\infty} \gamma^t r_t$ for $\gamma \in [0, 1]$. The expected reward $J(\pi)$ of policy π is equal to $J(\pi) = \mathbb{E}(G_0^\pi)$.

In contrast to existing methods for interpretability in RL that generate complex outputs: decision trees with algebraic constraints (Liu et al., 2018), finite-state automata (Araki et al., 2019), or programs (Verma et al., 2018) to represent policies, AI-Interpret generates simple and shallow disjunctive normal form formulas (DNFs; equivalent to decision trees, see Definition 3) that express the strategy in terms of pre-defined logical predicates.

Definition 3 (Disjunctive Normal Form) Let $f_{i,j}, h : \mathcal{X} \rightarrow \{0, 1\}, i, j \in \mathbb{N}$ be binary-valued functions (predicates) on domain $\mathcal{X}$. We say that h is in disjunctive normal form (DNF) if the following property is satisfied:

$$h(\boldsymbol{x}) = (f_{1,1}(\boldsymbol{x}) \wedge ... \wedge f_{1,n_1}((\boldsymbol{x})) \vee ... \vee (f_{m,1}((\boldsymbol{x}) \wedge ... \wedge f_{m,n_m}((\boldsymbol{x})) \quad (1)$$

and $\forall i, j_1 \neq j_2, \; f_{i,j_1} \neq f_{i,j_2}$. In other words, h is a disjunction of conjunctions.

Studies presented by Skirzyński et al. (2021) show that transforming this output into flowcharts, which use natural language instead of predicates, is easily understood by people, and can even help in improving their planning skills. Moreover, AI-Interpret is an imitation learning method (see Definition 4) and interprets policies via their demonstrations. Due to these reasons, we decided to use it in order to achieve our goal: find descriptions of human planning strategies by using data from process-tracing experiments.

Definition 4 (Imitation learning) Imitation learning (IL) is the problem of finding a policy $\hat{\pi}$ that mimics transitions provided in a data set of trajectories $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^M$ where $s_i \in \mathcal{S}, a_i \in \mathcal{A}$ (Osa et al. 2018).

On a high level, AI-Interpret uses 4 inputs. If $\mathcal{S}$ is a set of states and $\mathcal{A}$ is a set of actions in a given environment, AI-Interpret accepts a data set of demonstrations (state-actions pairs) $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^N, s_i \in \mathcal{S}, a_i \in \mathcal{A}$ generated by some policy π. Additionally, it also accepts the set of predicates $\mathcal{L}$ that act as feature detectors. Those feature detectors evaluate to $True$ or $False$ depending on the action to be taken and the current state, i.e. $f : \mathcal{S} \times \mathcal{A} \rightarrow \{0, 1\}$. On top of that, AI-Interpret uses a parameter d denoting the maximum depth of the DNF (decision tree), and the ratio of the expected rewards α. AI-Interpret uses $\mathcal{D}$ and $\mathcal{L}$ to find DNF formula ψ of size at most d. Formula ψ is required to induce policy π_ψ with an expected reward of at least α of π's expected reward. AI-Interpret achieves that by transforming each state-action pair in $\mathcal{D}$ into a vector of predicate valuations and clustering the set of these vectors into coherent groups of behaviors. In each iteration, the clustered vectors are used as positive examples for a DNF learning method, and suboptimal state-action pairs generated by looking at possible actions in existing states, serve as negative examples. The DNF learning method is called Logical Program Policies (LPP; (Silver et al., 2020)).

LPP defines a prior distribution for the predicates in $\mathcal{L}$, and uses the MAP estimation and decision-tree learning methods to find the most probable DNF formulas that accept the positive examples and reject the negative examples. AI-Interpret uses LPP to find DNF ψ that achieves the expected reward defined by α and has depth limited by d. In case of a failure, it removes the least promising cluster (the smallest weighted posterior) to try to describe the remaining data.

A new method for discovering and describing human planning strategies

We created a method that enables (cognitive) scientists to generate descriptions of human planning strategies using data from sequential decision-making experiments conducted with the Mouselab-MDP paradigm (Callaway et al., 2020; Callaway et al., 2017). As illustrated in Fig. 2, our method comprises the following steps: 1) collecting and pre-processing process-tracing data on human planning, 2) setting up a vocabulary of logical predicates that can be used to describe people's strategies, 3) running our new algorithm 10 times to automatically discover strategies possibly used by people, 4) applying a choice heuristic to select which of those strategies are accurate. Importantly, during strategy discovery, our algorithm automatically describes the found strategies as step-by-step procedures. The first four subsections describe each of these four steps in turn. We also detail the algorithm itself. The last section reports on the technical details of setting up the initial code base for our pipeline and can be skipped.

Data collection and data preparation

To use our method for discovering human strategies, the data collected in the experiments is required to meet certain criteria. Firstly, the experiment has to externalize people's planning by using computerized process-tracing paradigms for planning (see "Externalization" in Fig. 2). In our benchmark studies we used the Mouselab-MDP paradigm (Callaway et al., 2020; Callaway et al., 2017), since to our knowledge, it is the only such paradigm that is available so far. To study one-shot decision-making, on the other hand, one could use the standard Mouselab paradigm (Payne et al., 1988), ISLab (Cook & Swain, 1993), MouseTrace (Jasper & Shapiro, 2002) or other similar environments. Secondly, the experiments need to gather participants' *planning operations* that we define as sequences of state-action pairs generated by each of the participants. The states are defined in terms of the information that the participant has collected about the task environment. The actions are the information-gathering operations that the participant performs to arrive at their plan. Thirdly, the gathered planning operations should be saved in a CSV file that would additionally contain labels of the experimental block they came from (e.g. 'train' or 'test') and labels corresponding to participant id. In our case, this data was separated into distinct CSVs and custom Mouselab-MDP functions extracted states, actions, blocks, and ids from them into a new (Python) object. The result of this process is visually presented in Fig. 2 as the "Planning operations" arrow.

Creating the Domain Specific Language

Humans build sentences by using words (and build words by using phonemes). In order to automatically build descriptions of strategies used by people in the planning process-tracing experiments, we also need a vocabulary of some primitives. Due to the algorithmic setup of the method introduced in this section, here, the required set of primitives comprises logical predicates. Following Skirzyński et al. (2021) whose method is an important part of our pipeline, the predicates serve as feature-detectors and are formally defined as mappings from the set of state-action pairs to booleans, that is $f : \mathcal{S} \times \mathcal{A} \rightarrow \{0, 1\}$. Practically, the predicates describe the state $s \in \mathcal{S}$, the action $a \in \mathcal{A}$, or the particular characteristic that action a has in state s. For instance, predicate `is_observed(s,a)` might denote that node number a in the Mouselab MDP paradigm has not yet been clicked and its value is hidden in state s. Later, this predicate could be used to define a very simple DNF that allows clicking all the nodes that have not yet been clicked, i.e. `not(is_observed(s,a))`. The second step in our pipeline thus considers creating a set of predicates that describe the process-tracing environment. Further in the text, we will call that set of predicates the *Domain Specific Language* (DSL). Setting up the DSL is visually presented in Fig. 2 as "DSL creation", and as we discussed, occurs after externalizing human planning.

As one can notice, creating an appropriate Domain Specific Language is a non-trivial task that is important to the success of the whole method. The process of creating the DSL usually entails studying the structure of the task and the strategies people are thought to use in that task. For instance, if we were to create a DSL for a multi-alternative risky choice environment (such as Mouselab from "Mouselab") we would first adhere to the existing knowledge of the various decision strategies people are thought to use in this task (Payne, 1993) and the elementary operations they are composed of (Bettman et al., 1990). Inspecting those strategies would help us determine the primitives needed to describe them, and these primitives would serve as initial predicates of the constructed DSL. In the case of risky choice, we would focus on characterizing information-gathering operations. Furthermore, we would

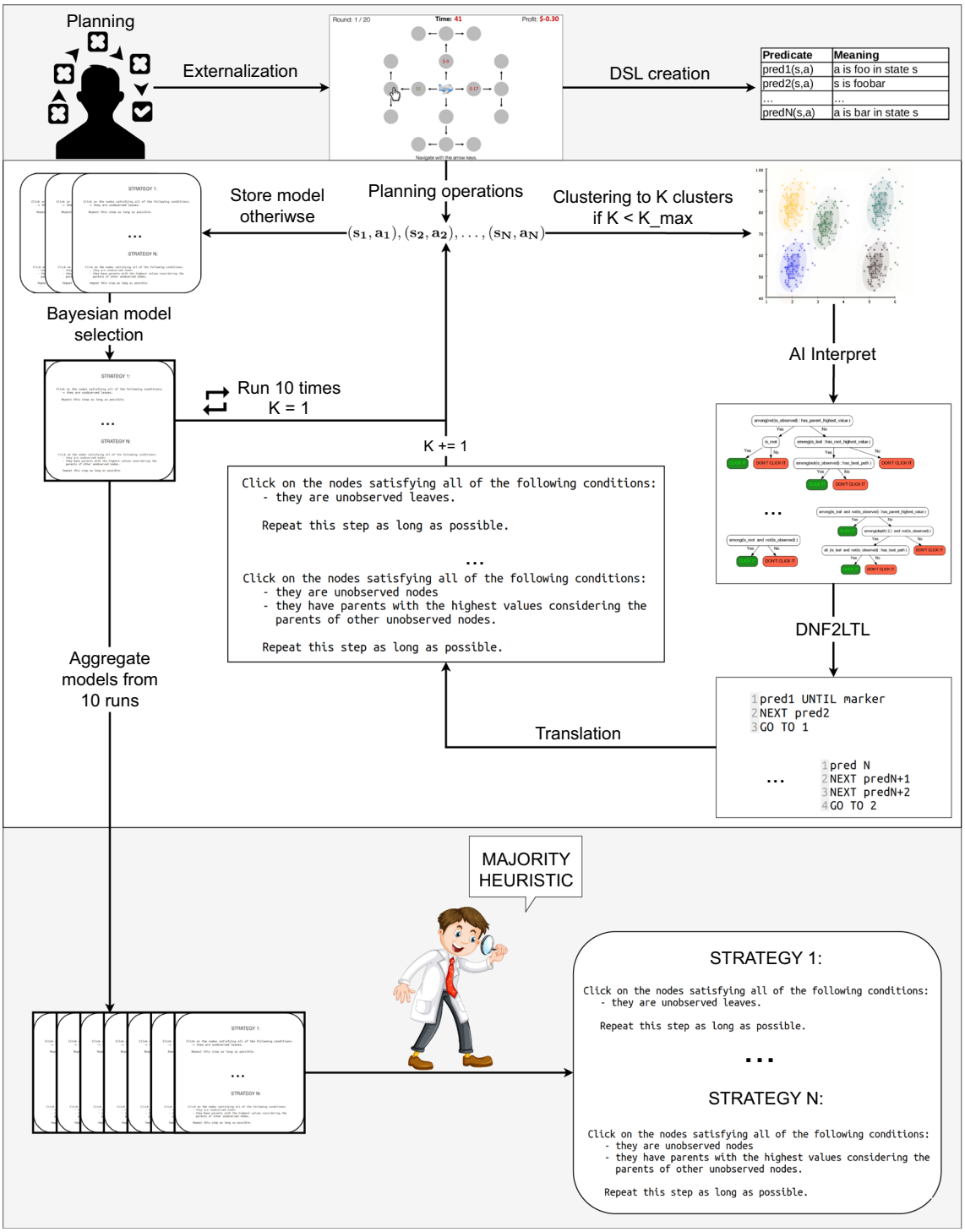

Fig. 2 Diagram representing our method for discovering and describing human planning strategies. The method assumes externalizing human planning first and gathering human planning operations in an experiment. Then it assumes creating a Domain Specific Language (DSL) to describe the environment in which participants made decisions. Afterward, the method assumes running Human-Interpret, an algorithm that automatically extracts and describes possible strategies used by people. Human-Interpret starts by performing clustering on the gathered data to create mathematical representations of planning strategies – policies. Then, it uses the AI-Interpret (Skirzyński et al., 2021) algorithm supplied with the constructed DSL and the found policies to discover formulaic descriptions of the strategies. Later, it transforms the formulaic descriptions into procedural instructions expressed in linear temporal logic and translates linear temporal logic formulas into natural language. Human-Interpret gradually increases the number of clusters and uses Bayesian model selection to output the best set of strategies it found. To generate the final result, our method assumes running Human-Interpret 10 times and applying a choice heuristic on 10 outputted sets

study the environment itself to expand the DSL with new primitives that described the properties of the environment. Those predicates could describe the gambles (e.g., most promising, second most promising, least explored, most explored, etc.) and the attributes (e.g., most informative one, one we know least about, etc.). By following this process, we would obtain a DSL with multiple predicates, some of which would stand for primitives known to be useful in describing people's planning heuristics, and some would denote characteristics of the environment that were yet unused. The algorithm's task would be to combine these primitives in one description so that they fit the sequences of planning operations externalized in the experiments.

Obtaining interpretable descriptions of human planning strategies via Human-Interpret

The third step of our method assumes using the algorithm for discovering interpretable descriptions of people's planning strategies. This algorithm finds a set of planning strategies people used based on the data from process-tracing experiments and then generates interpretable descriptions of those strategies in the form of procedural formulas. Since the AI-Interpret method of Skirzyński et al. (2021) plays a vital role in our pipeline, we call this new algorithm Human-Interpret.

Big picture

Human-Interpret is an algorithm that discovers human planning strategies externalized in a process-tracing paradigm in the form of natural language descriptions. As illustrated in Fig. 2, Human-Interpret clusters the participants' click sequences. It then applies the AI-Interpret algorithm to each cluster separately. AI-Interpret creates logical formulas that describe the planning strategies, and Human-Interpret translates those logical formulas into procedural descriptions in natural language. To maximize the likelihood of the final set of strategies obtained through this process, Human-Interpret also iterates through a higher and higher number of strategies to discover and eventually performs Bayesian model selection (Raftery, 1995) over the runs with respect to human data (see the transitions labeled "Run 10 times" and "Bayesian model selection" in Fig. 2). Technically, in each run, Human-Interpret executes 4 subroutines.

Firstly, Human-Interpret uses human data to generate demonstrations for AI-Interpret. It does so by clustering human data into an input number of subsets represented as policies and sampling those policies. Demonstrations obtained that way take the form of representative sequences of information-gathering actions (clicks) paired with knowledge states in which they were performed (states) (see the transition labeled as "Clustering to K clusters" that goes from the sequence of planning operations $(s_i, a_i)_{i=1}^{N}$ to the set of clusters in Fig. 2).

Secondly, Human-Interpret utilizes AI-Interpret to build descriptions of the sampled demonstrations. See the arrow labeled as "AI-Interpret" in Fig. 2 to visualize this process: AI-Interpret takes demonstrations from the clusters (middle left) and describes them using the created DSL (upper right) to output a number of flowcharts/DNF formulas describing planning strategies found in the demonstrations (middle right). The way in which these algorithms operate makes them robust to imperfections of the DSL. That is, even a highly incomplete DSL created according to the recipe from "Creating the Domain Specific Language" would likely result in approximate descriptions of some strategies. The reason is that AI-Interpret selects the subset of planning operations that are the easiest to describe with the existing DSL. Hence, although constructing the DSL does require human intelligence and domain knowledge, the robustness of our algorithm makes creating an appropriate DSL simpler than it would otherwise be.

Thirdly, Human-Interpret enhances the interpretability of the found formulas using the method from Becker et al. (2022) and turns the formulas into procedural descriptions mimicking the linear temporal logic formalism (see arrow "DNF2LTL" in Fig. 2 and consult Definition 5).

Definition 5 (Linear Temporal Logic) Let $\mathcal{P}$ be the set of propositional variables p (variables that can be either true or false), let $\neg, \wedge, \vee$ be standard logical operators for negation, AND, and OR, respectively, and let $\mathbf{X}, \mathbf{U}, \mathbf{W}$ be modal operators for NEXT, UNTIL, and UNLESS, respectively. Linear temporal logic (LTL) is a logic defined on (potentially infinite) sequences of truth assignments of propositional variables. LTL formulas are expressions that state which of the variables are true, and when they are true in the sequences. Whenever this agrees with the actual truth assignment in an input sequence, then we say that a formula is true.

Formally, for α and β being LTL formulas, we define a formula to be expressed in LTL inductively: ψ is an LTL formula if $\psi \in \mathcal{P}$ (ψ states that one of the variables is true in the first truth-assignment in the sequence), $\psi = \neg\alpha$ (ψ is a negation of an LTL formula), $\psi = \alpha \vee \beta$ (ψ is a disjunction of two LTL formulas), $\psi = \alpha \wedge \beta$ (ψ is a conjunction of two LTL formulas), $\psi = \mathbf{X}\alpha$ (ψ states that LTL formula α is true starting from the next truth-assignment in the sequence) or $\psi = \alpha\mathbf{U}\beta$ (ψ states that LTL formula α is true until some truth-assignment in the sequence where LTL formula β becomes true).

Fourthly, Human-Interpret expresses the found procedural descriptions in natural language. It does so by translating each predicate into a sequence of words, abiding by

Table 1 Explanation of the parameters used by Human-Interpret and other methods utilized by Human-Interpret

Algorithm	Parameter	Explanation
Human-Interpret	exp_id	Name of the experiment for which the clusters of state-action pairs will be created. The name is used to identify proper folders with the data.
	num_participants	Number of participants whose data from the experiment to consider.
	block	Part of the experiment from which the data is taken (e.g. 'test'). Depends on which identifiers have been used in the experiment.
	max_num_clusters	The maximum number of probabilistic (EM) clusters to create.
	criterion	Criterion for performing Bayesian model selection on models with differing numbers of clusters.
	num_trajs	Number of sequences of planning operations generated by the EM clusters used to find the descriptions of the clusters.
	$\mathcal{L}$	Domain Specific Language of predicates to create descriptions from.
	`DICT`	Predicate to natural language expression dictionary.
Expectation-Maximization	features	Set of functions which defines features in the environment for externalizing planning. The features are used by the softmax models that symbolize the EM clusters. Might be related or unrelated to the DSL.
	tolerance	Minimum change in the likelihood of the state-action sequences' assignment to EM clusters that allows further iterations of the EM algorithm.
	change_tolerance	Minimum relative change between the likelihood in the previous and current iteration that allows further iterations of the EM algorithm.
AI-Interpret	interpret_size, ai_tolerance, num_rollouts, num_ai_clusters	See Skirzyński et al. (2021).
	expert_reward	Average reward of the optimal policy in the environment
	max_divergence	The maximum difference in formula-induced policy's expected reward and the mean reward of the interpreted policy measured proportionally to the expert reward

predicate-to-expression translations and syntax rules predefined in a special predicate dictionary. We represent this step by the arrow "Translation" in Fig. 2.

The pseudo-code for Human-Interpret can be found in the Algorithm 1 box; the explanation of its parameters is shown in Table 1.

Examples

Consider the following two examples to better understand the workflow of Human-Interpret: Assume we are searching for planning strategies used by humans in the Mouselab MDP visible in Fig. 3a) and b).

The planning operations sequences we observed – so the nodes clicked to uncover rewards – are represented by numbers written next to the click arrows. The sample click sequences visible in the figures start with the node labeled as 1, the next element is the node labeled as 2, then 3, 4, etc. To relate to our pipeline shown in Fig. 2, the figures show planning operations externalized in a process-tracing paradigm, the Mouselab-MDP. Assume we observed multiple click sequences as in Fig. 3a) and b), and the clustering step resulted in 2 policies that were able to reproduce them. The click sequences seen in the figures can thus additionally serve to visualize the output of the clusters from the clustering step in Fig. 2. Assume we further created a Domain Specific Language that includes predicates describing the nodes, and the attributes of the clicks on the nodes. Relevant predicates that evaluated to true when node number i was clicked are written below i in Fig. 3a) and b). As we can see, in Fig. 3a) the first 3 clicks activated predicate `is_root`, i.e. they considered clicking the nodes of the MDP closest to the start, black node. Similarly, in Fig. 3b) the first clicks activated `is_leaf` predicate

Input: Id of the experiment exp_id;
Number of participants $num\_participants$;
Block $block$;
Max number of EM clusters $max\_num\_clusters$;
Bayesian criterion $criterion$;
Number of sequences generated by the EM clusters num_trajs;
Domain specific language $\mathcal{L}$;
Features for the EM softmax models $features$;
Tolerance for the EM clustering $tolerance$;
Tolerance ratio for the EM clustering $change\_ratio$;
Maximum size of the DNF's conjunctions in AI-Interpret $interpret\_size$;
Tolerance for AI-Interpret $ai\_tolerance$;
Number of rollouts for AI-Interpret $num\_rollouts$;
Number of clusters for AI-Interpret $num\_ai\_clusters$;
Expert reward in the environment $expert\_rew$;
Maximum divergence from the expert max_div;
Threshold for choosing the unless/until conditions $threshold$;
Allowed unless/until predicates $allowed\_predicates$;
Redundant predicates $redundant\_predicates$;
Predicate dictionary `DICT`

Output: Set of procedural formulas $\mathcal{P}$;

$\mathcal{P}_{best} = []$
for K in 1:$max\_num\_clusters$ **do**
  $\mathcal{P} = []$
  $planning\_operations \leftarrow$ load_human_data(exp_id, $num\_participants$, $block$)
  $EM\_clusters \leftarrow$ cluster_human_data($planning\_operations$, K, $features$, $tolerance$, $change\_ratio$)
  for c in $EM\_clusters$ **do**
    $demos \leftarrow$ generate_demonstrations(c, num_trajs)
    $DNF \leftarrow$ AI-Interpret($demos$, $interpret\_size$, $ai\_tolerance$, $num\_rollouts$, $num\_ai\_clusters$, $expert\_rew$, max_div, $\mathcal{L}$)
    $proc\_formula \leftarrow$ DNF2LTL(DNF, $threshold$, $allowed\_predicates$, $redundant\_predicates$)
    $pruned\_proc\_formula \leftarrow$ prune($proc\_formula$)
    $\mathcal{P}$.append($pruned\_proc\_formula$)
  end for
  $\mathcal{P} \leftarrow$ translate($\mathcal{P}$, `DICT`)
  if $criterion(\mathcal{P}) > criterion(\mathcal{P}_{best})$ **then**
    $\mathcal{P}_{best} = \mathcal{P}$
  end if
end for
return $\mathcal{P}_{best}$

Algorithm 1 Pseudocode for the Human-Interpret algorithm.

(nodes furthest to the black node), whereas predicate `is_max_observed` evaluated to $False$ for all but for the last node in the sequence when a big reward was observed. In all cases, predicate `is_observed` evaluated to $False$, i.e. each click was made on a node that has not been clicked yet. Human-Interpret uses multiple sequences of planning operations generated by the clusters alongside the predicates activated in each step of the sequences and runs AI-Interpret. AI-Interpret outputs a DNF formula for each cluster to capture the dynamics of the sequences. Assume that for Fig. 3a) the output was `(not(is_observed) and is_root and not(all_roots_observed))` `OR (not(is_observed) and all_roots_observed)`.

Since clicks other than the first 3 for sequences such as the one in Fig. 3a) activated many predicates, and only the predicate `not(is_observed)` evaluated to $True$ for all of them, AI-Interpret found the best possible DNF formula that agreed with all the elements in the sequences. For cluster that generated sequences as the one in Fig. 3b), the found DNF formula was `not(is_observed) and is_leaf and not(is_max_observed)`. This DNF captures that the click sequences always started with clicking the leaf nodes and terminated whenever the maximum

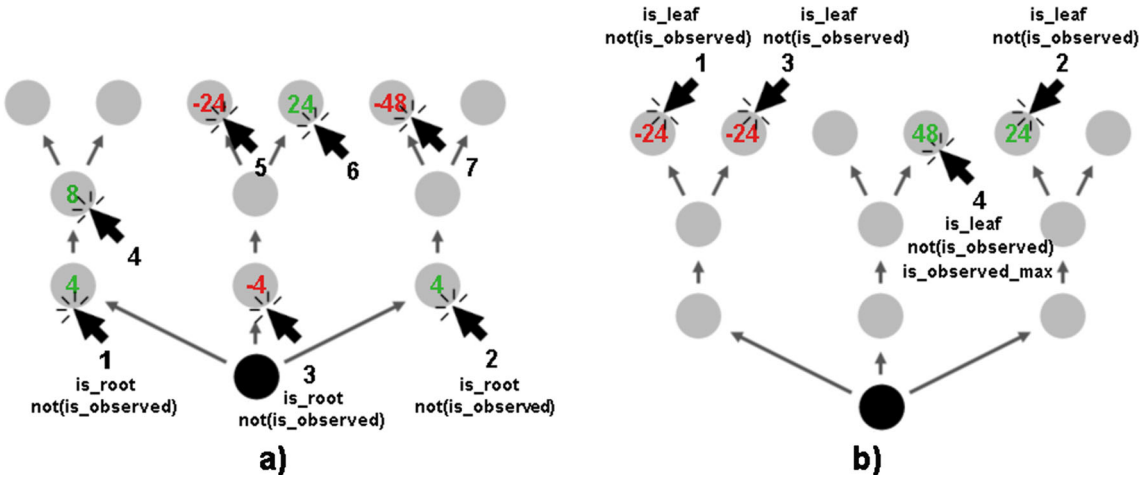

Fig. 3 Sample sequences of planning operations externalized in the Mouselab-MDP paradigm. Numbers below the click arrows denote the ordering of the planning operations (clicks), whereas the predicates written below the numbers denote relevant elements of the Domain Specific Language active when using a given planning operation

value was observed. In the next step, Human-Interpret uses an algorithm created by Becker et al. (2022) to transform the DNF formulas into linear temporal logic formulas. For the provided DNF formulas, the output would be `not(is_observed) and is_root and UNTIL all_roots_observed THEN True UNTIL IT STOPS APPLYING` (the expression after `THEN` captures complete uncertainty) and `not(is_observed) and is_leaf UNTIL is_max_observed`. Finally, Human-Interprets translates the procedural descriptions into natural language by making use of a predefined dictionary. The final output we would get for our examples if we used the dictionary same as in this paper would be *1. Click on the nodes satisfying all of the following conditions: they are unobserved roots. Repeat this step until all the roots are observed. 2. Terminate or click on some random nodes and then terminate. Repeat this step as long as possible.* for the first cluster exemplified by the click sequence in Fig. 3a) and *1. Click on the nodes satisfying all of the following conditions: they are unobserved leaves. Repeat this step until the previously observed node uncovers a 48.* for the second cluster exemplified by the sequence in Fig. 3b).

Human-Interpret would repeat these steps across all runs that consider different total numbers of clusters, say from 1 to 10, and employ Bayesian model selection to decide which set of strategies has the highest score under selected Bayesian criterion (e.g. marginal likelihood). This set would become the final output of the algorithm.

Clustering planning operations into planning strategies

Human-Interpret begins the process by extracting planning strategies out of sequences of planning operations gathered in the process-tracing experiment. Let $\mathcal{D} = \{(\tau_{1i})_{i=1}^{K_1}, (\tau_{2i})_{i=1}^{K_2} \ldots, (\tau_{Mi})_{i=1}^{K_M}\}$ denote the set of planning operations belonging to M participants where $\tau_{ji} = ((s_l^{ji}, a_l^{ji}))_{l=1}^{L^{ji}}$ is the i-th sequence of planning operations generated by participant j, and L^{ji} is its length. Human-Interpret utilizes the Expectation Maximization (EM) (Dempster et al., 1977; Moon, 1996) algorithm to fit a probabilistic clustering model to the state-action sequences in $\mathcal{D}$ and extracts k planning strategies (the clusters $\pi_1, \pi_2 \ldots, \pi_k$). Each planning strategy corresponds to a softmax policy of the form given in Eq. 2.

$$\pi_i(a \mid s, w_i) = \frac{\exp(\mathbf{f}(s,a)^\intercal w_i)}{\sum_{k=1}^{T} \exp(\mathbf{f}(s,a_k)^\intercal w_i)} \tag{2}$$

Each softmax policy (π_i) is represented by weights w_i assigned to P different features $\mathbf{f} = [\phi_1, \phi_2, \ldots, \phi_P]$ of the state-action pair where $\phi_i : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{X}_\phi$. These features were partially derived from the DSL and partially hand-designed. There were 19 of them and they can be found in Appendix A.4. The aim of the EM algorithm is to find the planning strategies by clustering the click sequences in $\mathcal{D}$ into k clusters, with each cluster being represented by some softmax policy π_i. It does this by optimizing the set of weights $W = (w_1, w_2, \ldots, w_k)$ that maximize the total likelihood ($\mathcal{M}$) of the click sequences under all the clusters. $\mathcal{M}$ is described in Eq. 3.

$$\mathcal{M}(\mathcal{D} \mid W) = \sum_{i=1}^{k} \sum_{C \in \mathcal{D}} \sum_{(s,a) \in C} \pi_i(a \mid s, w_i) \tag{3}$$

After obtaining the policies represented by the weights W, they are discretized to form new uniform policies ($\overline{\pi}_i$) as described in Eq. 4 that assign a uniform probability to actions with the highest probability according to π_i, that is

to optimal actions. Policies $\overline{\pi_i}$ are then used to create a data set of demonstrations $\overline{\mathcal{D}} = \{(s_i, a_i)\}_{i=1}^{L}$ which contains L planning operations generated through some fixed number of rollouts.

$$\overline{\pi}_i(a \mid s) = \begin{cases} 0, & \text{if } \pi_i(s,a) \neq \max\limits_{k \leq P} \pi(s, a_k) \\ 1/|\{a_i : \pi_i(s,a_i) = \max\limits_{k \leq P} \pi(s,a_k)\}| & \text{otherwise} \end{cases} \tag{4}$$

Finding formulaic descriptions of planning strategies

After computing policies $\overline{\pi}_i$ and generating the data set of demonstrations $\overline{\mathcal{D}}$, Human-Interpret essentially runs AI-Interpret (Skirzyński et al., 2021). The only modification to the AI-Interpret algorithm that is introduced by Human-Interpret relates to the fact that the input demonstrations no longer represent the optimal policy, but *some*, often imperfect policy mimicking humans: $\overline{\pi}_i$ (further called the interpreted policy). Because of that, the expected reward for policies induced by candidate formulas cannot make at least α of the expected reward for the interpreted policy (see the Background section), since it leads to errors. For instance, in a situation when the interpreted policy achieves a reward of R, and the optimal policy achieves a reward of $100R$, a formula that induces a policy with reward $50R$ meets the criterion defined by α, but is a poor approximation to $\overline{\pi}_i$. Because of that, Human-Interpret uses *divergence* instead of the expected reward ratio. If π_f is a policy induced by formula f found by AI-Interpret and π_{opt} is the optimal policy for the studied environment (here, the Mouselab MDP), Human-Interpret computes the divergence of f as the ratio between the difference in rewards for the interpreted policy $\overline{\pi}_i$ and π_f, and the expected reward of the optimal policy, i.e. $\text{div}(\pi_f) = \frac{J(\pi_f) - J(\overline{\pi}_i)}{J(\pi_{opt})}$. Consequently, Human-Interpret searches for solutions whose size is limited by d and for which the divergence is at most α (see the Background section). Note that introducing that modification requires the modeler, or the algorithm, to compute the optimal policy for the environment or at least know its approximation.

Extracting procedural descriptions from logical formulas

The output produced by the modified AI-Interpret algorithm we defined above is a DNF formula f^*. Following the finding that procedural descriptions are easier to grasp for people than flowcharts (Becker et al., 2022), Human-Interpret uses the method presented in Becker et al. (2022) to transform f^* into a logical expression written in linear temporal logic. The DNF2LTL algorithm described in Becker et al. (2022) produces such an expression by separating the DNF into conjunctions and finding the dynamics of their changes in truth valuations using the initial set of demonstrations inputted to AI-Interpret. The output, which we will call a *procedural formula*, separates the conjunctions with NEXT commands and instructs to follow each conjunction until some condition occurs unless another condition occurs. The conditions are chosen among the predicates or 2-element disjunctions of predicates from the DSL introduced in "Creating the Domain Specific Language", or simply read (for the UNTIL condition) "until it applies", which denotes a special logical operator. Since the output formula might be overly complex after that process, in the next step the algorithm prunes some of the predicates appearing in the conjunctions. Concretely, predicates are greedily removed one by one so as to increase the probability of people's planning operations under the strategy described by the shortened formula.

Translating to natural language

Once the procedural formulas are generated, it is possible to obtain fully understandable descriptions of human planning strategies by transforming the predicates and the operators appearing in the formulas into natural language. In our case, the procedural formulas are expressed in natural language by using a predefined predicate-to-expression dictionary that provides a direct translation for each predicate (as a sequence of words), and controls which words should be added, and in what order, to mimic the meaning of LTL operators.

Bayesian model selection

Human-Interpret uses Bayesian model selection according to Bayesian inference with a uniform prior on the number of clusters (Kass & Raftery, 1995). In more detail, it considers K runs of human data clustering and cluster description, where the output of run i posits the existence of i clusters and establishes model i. Each cluster for a model is defined as a mixture of two policies. The first policy is induced by the procedural description constructed for the cluster. The second policy serves as an error model. These two policies assign a uniform probability to actions allowed and disallowed by the procedural description constructed for the cluster, respectively. The weights assigned to both

policies are cluster-dependent free parameters with prior sampled using Beta functions where alpha and beta are hyperparameters (i.e., $\epsilon_i \sim \text{Beta}(\alpha, \beta)$ and $\epsilon_i \in [0, 1]$). Mathematically, the clustering model $P(\tau)$ for a sequence of planning operations $\tau = (s_i, a_i)_{i=1}^{T}$, for K clusters represented by K policies $\pi_1, \ldots, \pi_K$, and for K error models $\overline{\pi_1}, \ldots, \overline{\pi_K}$ took the following form:

$$P(\tau \mid \epsilon) = \sum_{i=1}^{K} 1/K * P_i(\tau; \epsilon_i) \tag{5}$$

$$= \sum_{i=1}^{K} 1/K * \prod_{j=1}^{T} \left(\epsilon_i * \overline{\pi_i}(s_j, a_j) + (1 - \epsilon_i) * \pi_i(s_j, a_j)\right). \tag{6}$$

Human-Interpret selects the best model using the equations above according to the input Bayesian criterion: either the marginal likelihood, the Bayesian Information Criterion (BIC) (Konishi & Kitagawa, 2008), or the Akaike Information Criterion (Vrieze, 2012).

Heuristically choosing strategies used by people

In the last step of our method, we aggregate the results of 10 runs performed by Human-Interpret, i.e. 10 Bayesian-optimal sets of strategies, and select the final output. In order to do that, we employ the majority heuristic. According to the heuristic, a strategy belongs to the output, if it was discovered in at least 7 of the models/runs. Otherwise, it is most likely noise. Our heuristic uses 7 as the majority criterion since it was found to maximize the accuracy of our method during evaluation (see "Measuring the reliability of our computational method ").

Technical details regarding using our method

Here, we present technical details connected to installing and using our method for discovering and describing human planning strategies. We equipped the initial code base written in Python 3.6 with 1) data from 4 planning experiments ran in different versions of the Mouselab-MDP paradigm, 2) a Domain Specific Language (DSL) of logical primitives used to generate procedural formulas, and 3) a dictionary of predicate-to-expression entries for transforming a formula into natural language. Each of these elements is either a parameter or a hyperparameter of Human-Interpret. A description of the experiments can be found in Jain et al. (2022), whereas the DSL is detailed in Appendix A.2 and contained in one of the files in the code base, similar to the dictionary. Thanks to the initial values for those parameters, it is possible to use our method without performing prior research. Moreover, one may extend our research by slightly modifying the DSL or running Human-Interpret on a different data set. The steps involved in setting up the whole method are as follows:

1. Download data needed in the pipeline and the source code for Human-Interpret by cloning the appropriate GitHub repository using the command:

   `git clone https://github.com/RationalityEnhancement/InterpretableHumanPlanning.git`

   The repository includes four data sets that are contained in the folder `data/human`. Refer to Jain et al. (2022) for a detailed description of the experiments they come from.
2. Access the root directory of the downloaded source code and install the needed Python dependencies:

   `pip3 install -r requirements.txt`
3. Run Human-Interpret on either of the available data sets by typing

   `python3 pipeline.py ...run <r> ...experiment_id <name> ...max_num_strategies <max> ...num_participants <num_p> ...expert_reward <exp_rew> ...num_demos <demos> ...begin <b>`

   Parameter `run` is the run id of the call to the function. Then, `experiment_id` corresponds to the name of the experiment that resulted in one of the four data sets. As written in Table 1, `max_num_strategies` quantifies the maximum number of strategies that could exist in the data set and should be described; parameter `expert_reward` defines the maximum reward obtainable in the Mouselab MDP defining each of the experiments; `num_participants` state that the data of the first `<num>` participants should be extracted from the data set; `num_demos` controls how many strategy demonstrations to use in Human-Interpret; `begin` controls which model to begin with (how many clusters to consider in the first model to then incrementally increase that number thus defining consecutive models). The available names, number of all tested participants, and corresponding expert rewards are provided in the readme file included in the source code. The output of this command is saved in the `interprets_procedure` folder in a text file named according to the following structure (and smaller files without the prefix):

   `BEST_strategies_<name>_<max>_<num_p>_<demos>_run<run>`.

   The files contain procedural formulas describing the clusters, their natural language descriptions, and a set of statistics associated with those descriptions (e.g.

how well they fit the experimental data, how big they are, etc.)

4. To reproduce the exact set of strategies from this paper, run the following code for *i* in 1:10:

```
python3 pipeline.py
...run i ...experiment v1.0
...max_num_strategies 17 ...begin
17 ...num_participants 0
...expert_reward 39.97 ...num_demos
128
```

Information regarding other parameters available for tuning the Human-Interpret algorithm can be found in Table 1 and in file `pipeline.py`. The set of logical primitives serving for our DSL may be accessed by navigating to `RL2DT/PLP/DSL.py`. Finally, the dictionary is included in file `translation.py`. To apply our method to different problems consult `pipeline.py`, `README.md`, "Data collection and data preparation", and "Creating the Domain Specific Language".

Evaluating our method for discovering human planning strategies

In this section, we evaluate the reliability of our method on data from a planning externalization experiment that utilized the Mouselab-MDP paradigm. We measure reliability threefold. First, we analyze the quality of all the strategies that Human-Interpret discovered automatically. Second, we measure the observational error of the final set of strategies outputted by our method with respect to strategies discovered through laborious manual inspection of the same data set. Third, we compare the performance of our computational method to the manual method. The first subsection focuses on the setup of our method: it describes the planning experiment, the vocabulary of logical primitives, and the parameters for Human-Interpret. The second subsection numerically describes all of the discovered strategies. The third subsection measures the observational error of our method. The last subsection compares our method to the manual method.

Setup of the benchmark problem

Planning experiment

In our benchmark problem, we used a process-tracing experiment on human planning conducted according to the Mouselab-MDP paradigm (see "Mouselab-MDP"), namely the first experiment presented in Jain et al. (2022). This experiment (which we will refer to as the increasing variance experiment) used the Mouselab-MDP environment shown in Fig. 1 where the rewards hidden underneath the nodes were different in every trial but were always drawn from the same probability distribution. The nodes closest to the spider's starting position at the center of the web had rewards sampled uniformly from the set $\{-4, -2, 2, 4\}$. Nodes one step further away from the spider's starting position had rewards sampled uniformly from the set $\{-8, -4, 4, 8\}$. Finally, the nodes furthest away from the starting position of the spider harbored rewards sampled uniformly from the set $\{-48, -24, 24, 48\}$. The fee for clicking a node to uncover its reward was 1. The 180 participants who took part in this experiment were divided into 3 groups that differed in what kind of feedback was provided during the trials. The control group received no feedback. The second group received feedback on their first move. The third group received feedback on every click they made that was designed to teach them the planning strategy that is optimal for the task environment. There were always 2 blocks: a training block and a test block, with 20 trials in the training block and 10 trials in the test block. As stated in the previous sections, the experiment utilized the Mouselab-MDP process-tracing paradigm and operationalized people's planning strategies in terms of the clicks (planning operations) they performed.

Domain Specific Language (DSL) and translation dictionary

We adopted the DSL of predicates from our work on AI-Interpret which was also conducted on Mouselab-MDP environments (Skirzyński et al., 2021). Generally, the DSL consisted of over 14000 predicates generated according to hand-made context-free grammar. The predicates were defined on state-action pairs, where states were represented as Python objects capturing the uncovered and covered rewards in the Mouselab-MDP, and each action denoted the ID of the node to be clicked. The predicates described the current state of the Mouselab-MDP or the actions available in that state. The state is described in terms of the clicked nodes, the termination reward, and other properties. The actions are described in graph theoretic terms, such as the depth of the clicked node, whether it is a leaf node, whether its parents or children have been observed, and so on. The DSL included two crucial second-order predicates: the *among* predicate asserts that a certain condition (given by a first-order predicate) holds among a set of nodes defined by another first-order predicate, and the *all* predicate asserts that all the nodes satisfying a certain condition also satisfy another condition. Detailed descriptions of the predicates in the DSL we used are available in Appendix A.2.

The dictionary we used for translating the resulting procedural formulas is an adapted version of the dictionary from Becker et al. (2022). In our dictionary, we changed natural language translations of most predicates to use graph

theoretic jargon (such as leaves, roots, etc.). Moreover, our translations always begin with "*Click on the nodes satisfying all the following conditions:*" if there are any non-negated predicates in the formula and list the non-negated predicates as conditions. The difference with the original dictionary is also that more complex predicates (such as those which included other predicates as their argument) are broken down into 2 or more conditions, whereas in the original dictionary each predicate has its unique translation. More details on how the translation was created can be found in our project's repository in the `translation.py` file.

Parameters for the Human-Interpret algorithm

To run Human-Interpret, we used the default parameters for the AI-Interpret algorithm, the DSL described in the previous section, the data from all of the participants in the experiment described in "Planning experiment" (that is 180), and data from both blocks of the experiment (i.e., the training block and the test block). We ran Human-Interpret with 1-20 clusters and had it perform model-order selection according to the BIC. The model selection was set to disregard clustering models for which Human-Interpret was unable to produce a description for all clusters. Our rationale for doing so was that models that do not describe all the clusters are not useful for the purpose of understanding human planning. The DSL we used and the predicates we used for the DNF2LTL algorithm can also be found in Appendix A.2 and A.3. All parameters are listed in Table 5 in Appendix A.1. The more elusive parameters, such as interpret_size or num_trajs were selected based on the simulations and interpretability experiments presented by Skirzyński et al. (2021).

Measuring the reliability of Human-Interpret

Method

To assess the reliability of Human-Interpret, we ran clustering and description subroutines of Human-Interpret 21 times to output 21 maximum BIC models and analyzed each subset of 10 models with respect to the ground truth, i.e. the strategies found by Jain et al. (2022). In total, the runs found 21 distinct strategies (found in Appendix A.5), where 10 of the 21 strategies were rediscovered with respect to the manual analysis by Jain et al. (2022). Since every strategy outputted by Human-Interpret is represented in terms of a cluster of planning operations and the policy mixture describing the cluster, we measured the average of the mean likelihood per planning operation in the clusters under the policy mixtures assigned to the unique strategy. For comparison purposes, we additionally measured the likelihoods under two additional methods: the random method represented by a policy that always assigns the same probability to all possible planning operations (including termination), and the Computational Microscope (Jain et al., 2022) whose policies are programmatic versions of the strategies Jain et al. (2022) discovered manually.

Technical specification

We performed our benchmark tests on a computing cluster equipped with Intel-based CPU nodes with 1TB of RAM and 64 cores, and NVIDIA V100 GPU accelerators with 16 GB HBM2 and 30 GB of RAM. Our implementation of Human-Interpret was predominantly CPU-based and relied on multi-threading. Hence, a single job made use of only 1 GPU unit and 1 CPU core. Thanks to the cluster, however, we were able to use parallelization and run all jobs with a different number of clusters (21 * 20 = 420 jobs) in parallel.

Results

Figure 4 shows the aggregated results of the runs by reporting the average improvement in the mean likelihood over the random method of both Human-Interpret and the Computational Microscope across all the discovered strategies. As it can be seen, the planning operations are over twice as better fitted to the strategies that represent them than if they were assigned at random. Unsurprisingly, the same planning operations are also better fitted to the manually found strategies from the Computational

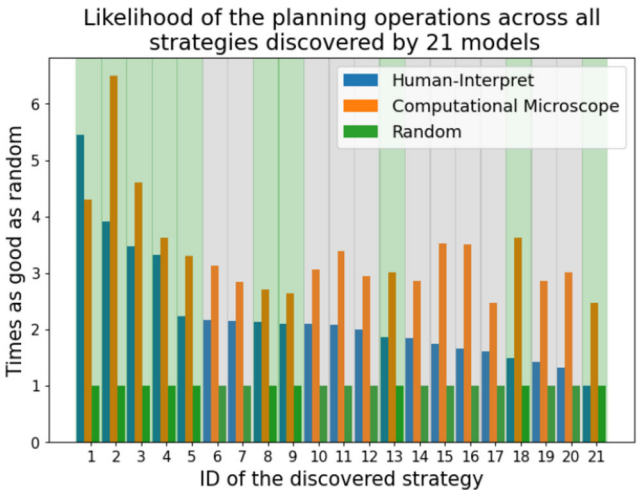

Fig. 4 Improvement over the random strategy in terms of the standard mean likelihood per planning operation for 3 methods: Human Interpret, the Computational Microscope, and a random method that always assigns equal probability to all possible actions. The listed strategies are those found in the initial 21 strategy models. The bars show improvement in terms of the mean likelihood per planning operation. Background colors encode whether a strategy was chosen as ground truth (green) or not (grey)

Table 2 Error statistics for the output of our method with respect to the manually discovered ground truth

Run	F-1 score	precision	recall	accuracy	stability	variability
Mean	$0.53 \pm 4e^{-5}$	$0.8 \pm 2e^{-4}$	0.4 ± 0	0.67 ± 0	$0.795 \pm 1e^{-3}$	$1.45 \pm 1e^{-3}$
Median	0.53	0.8	0.4	0.67	–	–

The mean run represents a run with average statistics, whereas the median run represents a run with median values for all the statistics. The numbers after the ± sign indicate standard errors

Microscope than to the ones automatically discovered by Human-Interpret.

Measuring the reliability of our computational method

Method

The output of our method depends on how the outputs of the 10 runs of Human-Interpret are aggregated. The majority heuristic used to perform this aggregation has one parameter: the number of runs that have to agree on a discovered strategy. Therefore, we evaluated the reliability and reproducibility of our method using cross-validation. This involved splitting the set of all 10-run subsets into a training set that contained 70% of all 300 thousand subsets and a test set that contained the remaining 30% (about 100 thousand subsets). We selected the value of the parameter by maximizing the following accuracy metric on the training set:

$$acc(S) = \frac{|b \in B : b \in G| + |b \notin B : b \notin G|}{|S|}, \tag{7}$$

where G is the ground truth, S is the set of all strategies, and B is the set of strategies discovered by our computational method. The parameter value that optimized the method's accuracy on the training set was 7. We then evaluated the method's performance with the chosen parameter value on the test set. We measured the method performance using four standard metrics from signal detection theory: accuracy, precision, recall, and F1 score (McNicol, 2005). A method's precision is the proportion of strategies discovered by our method that correspond to the ground truth strategies, whereas recall defines the proportion of the total number of the ground truth strategies our method discovers. Formally, the precision $prec$, recall rec and $F1$ measures are defined as

$$prec(S) = \frac{|b \in B : b \in G|}{|B|}, \tag{8}$$

$$rec(S) = \frac{|b \in B : b \in G|}{|G|}, \tag{9}$$

$$F1(S) = 2 \cdot \frac{prec(S) \cdot rec(S)}{prec(S) + rec(S)}. \tag{10}$$

We chose F1-score as the secondary measure of reliability because it emphasized the number of true positives. Lastly, we also measure the stability of our method by estimating the proportion of runs that output the median result we report here, and its variability that quantifies how many strategies are in the difference set if 2 outputs are different.

Results

As summarized in Table 2, our method has high precision of 80%, moderately high accuracy of 67%, and rather low recall of 40%. The F1-score of 0.53 indicates that the method presents good reliability. The stability of our method is equal to 79.5%, meaning that there is roughly an 80% chance of generating the same results when the method is run twice. If the results are different, however, they differ by 1.45 strategies on average. We will refer to this statistic as the method's *variability*. The most common example of non-zero variability in our tests was that the smaller set was missing one strategy or that each set contained one strategy that was missing from the other one. The standard errors are mostly null, meaning that all these estimates are robust. Jointly, these numbers mean that around 80% out of all strategies discovered by our computational method is actually used by people in planning, and these strategies represent slightly below half of all the ground truth strategies Human-Interpret can currently discover. Using the example of the median run of our method with respect to the F1-score, it discovers 5 strategies where 4 are actually used by people, and 1 represents noise. The stability and the variability of the method indicate this is a representative output. We discuss how relevant the discovered strategies are in the next section.

Evaluating and comparing strategies discovered by our computational method to strategies discovered through manual inspection

In this section, we show the descriptions of human planning strategies discovered in the median run of our method, list their statistics, and compare them to the strategies that Jain et al. (2022) found through manual inspection. In this and the following sections, we address the discovered strategies as Strategy i for $i \in [21]$ relating to the numeration

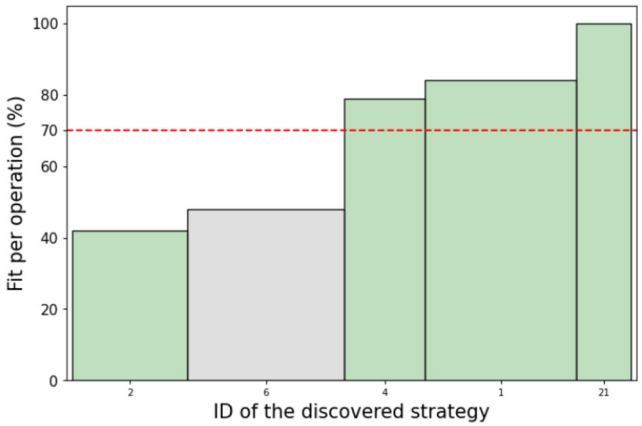

Fig. 5 Plot showing the quality of strategies found by our method, measured as the fit per operation (FPO), depending on their size. Strategies are labeled on the x-axis via their unique ID (see Table 3 for a reference). The width of the bar corresponds to the size of the cluster representing the strategy. The height of the bar corresponds to the FPO measure. The red line indicates the average FPO across all clusters. The green color indicates strategies that belong to the ground truth, whereas the grey color indicates noise

introduced in Fig. 5. Some statistics on the strategies are reported with the standard error.

Ground truth strategies Human-Interpret outputted descriptions for 5 strategies seen in Table 3: Strategies 1, 2, 4, 6, and 21. Four of those strategies concur with the ground truth; only Strategy 6 is a false positive. The 4 rediscovered strategies jointly account for 51.3% of people's planning operations (Jain et al., 2022). Exactly 2 of the 4 strategies: Strategy 2 and Strategy 4, are mentioned as one of the most frequent according to Jain et al. (2022). There is one more frequent strategy that our method did not discover. However, our analysis of Strategy 1 indicates that manual analysis heavily underrepresented its frequency (Jain et al., 2022). It is because the average fit per planning operation (FPO) is significantly higher under this strategy than under the strategy assigned by the Computational Microscope. Since Strategy 1 is a special case of Strategy 4, and Strategy 4 has a frequency of over 36% in the manual analysis, we suspect that many planning operations assigned to Strategy 1 were labeled as Strategy 4 by the manual analysis. Hence, Strategy 1 was underrepresented despite the fact it seems to be often used by people.

Noisy strategies When it comes to the noisy strategies not represented in the ground truth, our method returned 1 such output. This output represents a particular behavior of Human-Interpret, namely an oversimplification of the strategies caused by the data selection process. Strategy 6, despite being very similar to one of the manually found strategies, lacks an early termination constraint. It was discovered in this form due to the operation selection process implemented by Human-Interpret (through AI-Interpret) where certain planning operations are removed from the set to simplify creating a description. Relevant data was hence likely removed and the strategy eventually became too specific: it considered only a subset of the planning operations. Since the strategy was chosen even after running Human-Interpret multiple times and applying the majority heuristic, the removed data must have been indescribable with the current DSL.

Quantitative analysis of the strategies Here, we report descriptive statistics about the complexity of the automatically generated descriptions, the frequencies of the discovered strategies, how well they corresponded to the EM clusters, and how well they explain the sequences of planning operations they are meant to describe. These statistics suggest that our method enables us to find reasonable solutions of a very promising quality, especially considering that our method discovered those strategies without any labeled training examples or human feedback. On average, the discovered descriptions agreed with the softmax models of the clusters on 84% ± 8% of the planning operations. That quantity, which we call the *formula-cluster fit* (FCF) is defined as the average of two proportions. The first one is the proportion of planning operations generated according to the inferred description that agreed with the choices of the corresponding softmax model. The second one is the equivalent proportion obtained by generating the demonstrations according to the softmax models and then evaluating them according to the descriptions. In both cases, we performed 10000 rollouts. Further, the average likelihood of the planning operations within the clusters reached 70% ± 11% of the likelihood that would have been achieved if all planning operations perfectly followed the descriptions. We call that proportion the *fit per operation* (FPO). The quality of individual cluster descriptions, as measured in terms of the FPO, is depicted in Fig. 5. Since the average measurements over all clusters do not take into account the importance of clusters, we decided to perform the same computations using cluster size-dependent weighted averages. As larger clusters represent the most often used strategies, and these are the strategies we are predominantly interested in from a psychological standpoint, weighted averages capture the measured statistics with higher accuracy. After weighing our statistics we found that the descriptions achieved a similar formula-cluster fit (FCF) of 86% ± 4%, and the fit per planning operation (FPO) of 66% ± 3%. We also tried excluding the unique cluster describing the random policy, which trivially achieves the FPO of 1, which caused the average to drop to 62% ± 3%. Those results suggest that the more accurate estimate of the fit with respect to planning operations is

Table 3 The strategies discovered by applying our computational method to the benchmark problem

ID	Strategy descriptions	Statistics					
		FR	FCF	FON	FPO	C	N
4	**Description:** 1. Click on the nodes satisfying all of the following conditions: – they are unobserved leaves. Repeat this step until all the leaves are observed or the previously observed node uncovers a 48. **Summary:** Explore final outcomes until observing +48 **Manual counterpart:** Search for the best possible final outcome	16.7% 36.6%	0.88	0.86	0.75	5	10
1	**Description:** 1. Click on a node satisfying all of the following conditions: – it is an unobserved leaf. 2. Unless the previously observed node uncovers a 48, in which case stop at the previous step, click on a node satisfying all of the following conditions: – it is an unobserved leaf. click in this way under the condition that: – the previously observed node was its sibling. 3. GOTO step 1 unless all the leaves are observed or the previously observed node uncovers a 48. **Summary:** Explore final outcomes until observing +48 sibling by sibling. **Manual counterpart:** Explore final outcomes with a preference for nodes in the same sub-tree of the root	31% 0.36%	0.98	0.89	0.77	13	10
21	**Description:** 1. Terminate or click on some random nodes and then terminate. Repeat this step as long as possible. **Summary:** Random planning. **Manual counterpart:** Random planning.	4.3% 1.2%	0.53	1.0	1.0	1	10
2	**Description:** Do not click. **Summary:** No planning. **Manual counterpart:** No planning	22.6% 13.2%	1.0	0.46	0.26	1	9
6	**Description:** 1. Click on a node satisfying all of the following conditions: – it is an unobserved leaf. 2. Click on the nodes satisfying all of the following conditions: – they are unobserved leaves. Click in this way as long as: – the previously observed node was their sibling. Repeat this step as long as possible. **Summary:** Explore all final outcomes sibling by sibling **Manual counterpart:** –	6.3% –	*0.73	0.61	0.36	7	9

The strategies are listed along side their ID, their automatically generated description, the summary we created by hand, and the name (or numbers) of the corresponding strategy (strategies) discovered manually by Jain et al. (2022). FR denotes the frequency of the strategy; FCF (fit cluster-formula) averages two proportions: formula demonstrations agreeing with the softmax clusters and vice-versa measured using 10000 demonstrations; FON (fit optimal-non-optimal) quantifies how often people's planning operations in the cluster agreed with the description; FPO (fit per operation) is the ratio between the average likelihood per planning operation belonging to the cluster and the average likelihood per planning operation for the (policy induced by the) cluster's description; C, which stands for complexity, is the number of individual predicates in the description; N is the number of clusters with the given description. Frequency (FR) is the only statistic measured for the manually found strategies. The statistics are averaged across all models in the median run of our method

Springer

closer to 65%, whereas the fit with respect to the formulas is indeed around 85%.

Quantitative comparison against the manual method Generally, our method outputted 5 strategies that cover 51% of the ground truth planning operations and describe them almost 3 times better than chance. In the case of the manual inspection performed by Jain et al. (2022), the results were higher, and the average improvement was almost 4 times as high as the random method across 79 strategies. It has to be noted, however, that our method analyzed only up to 20 clusters (thus strategies) contrary to Jain et al. (2022). Hence, the strategies discovered through our method were more coarse. Despite that limitation, our method managed to achieve a good accuracy score of 67%, all within mere 20 days, which included problem analysis and DSL creation (around 2 weeks), running the code (around 5 days), and analyzing the results (around 1 day). In contrast, it took Jain et al. (2022) about 120 days to finish their manual analysis. This means that our method could have accelerated the process of strategy discovery by over 3 months – 100 days – compared to manual inspection. Jointly, these results suggest that our method can be used to speed up research on the strategies of human planning and decision-making. A more detailed summary of the comparison between the performance of our automated strategy discovery and description method with the manual approach by Jain et al. (2022) can be found in Table 4.

Table 4 Comparison between our method for automatically finding and describing human planning strategies and the manual approach by Jain et al. (2022)

	Method	
Statistic	Automatic analysis	Manual analysis
Runtime	**20 days**	120 days
Discovered strategies	5	79
Accuracy	0.67	**1**
Success rate	81%	**100%**
Ground truth represented	51%	**100%**
Likelihood per click	0.31 ± 0.06	$\mathbf{0.43 \pm 0.04}$
Times as good as random	2.99 ± 0.7	$\mathbf{3.93 \pm 0.61}$

Runtime is the number of days it took to generate the result. Discovered strategies counts how many strategies were generated by the method. Accuracy measures the accuracy of the method according to Eq. 7. Success rate is the proportion of sequences of planning operations that the method eventually described. Ground truth represented measures the ground truth proportion of planning operations that are represented by the ground truth strategies discovered by the method. Likelihood per click is the average likelihood per a planning operation. Times as good as random quantifies the improvement in average likelihood per operation over the random model which assigns equal probability to all actions that are possible in a given step

General discussion

The main contribution of our research is automating the process of scientific discovery in the area of human planning. By using our method, scientists are no longer left at the mercy of their own ingenuity to notice and correctly interpret the right patterns in human behavior; instead, they can rely on computational methods that do so reliably. Concretely, we developed the first method for the automatic discovery and description of human planning strategies. Our method's pipeline comprises 4 steps: The first step is to run an experiment that externalizes human planning operations. The second step is to create a Domain Specific Language (DSL) of logical predicates that describe the task environment and the planning operations. The third step is to run the generative imitation learning algorithm that we created and present in this paper, called Human-Interpret. Human-Interpret discovers the strategies externalized in the experiment by creating generative softmax models (the generative step) and describes them by procedural rules formulated in the DSL by imitating their rollouts (the imitation learning step). Finally, the fourth step is to apply a heuristic on the output produced by Human-Interpret to choose the final set of strategies.

Advantages

Human-Interpret is the first-ever automatic method that can discover that people use previously unknown planning strategies and what they are. This is a step towards leveraging artificial intelligence to facilitate scientific discovery in research on human planning and decision-making. The main benefits of this approach are that it is more objective, potentially capable of discovering strategies that human researchers might overlook, and that it can be applied to many large data sets that human researchers do not have the capacity to analyze manually. The following paragraphs summarize the strengths of our method in more detail.

Reliability The tests we ran on a benchmark planning problem revealed that the reliability of our method is comparable to that of a manual human analysis when it comes to the most frequent strategies. Firstly, the automatically discovered strategy descriptions were clear and understandable (see Table 3), mimicking the rigor and clarity of man-made descriptions. Secondly, the overall accuracy of our method was good, reaching 67% while keeping precision as high as 80%. This means that the strategies found by our method were highly likely to be actually used by people. Finally, our automated approach found 2 out of 3 most important, frequently used strategies assuming the division from Jain et al. (2022), and provided

evidence to label one additional strategy as relevant. The one remaining important strategy was not expressable with the current DSL (we explain that in paragraph *Imperfect DSL*), and hence was not outputted. These results indicate that our method discovered almost the same set of important strategies as scientists did despite working with almost no human supervision. Reliability of our method with respect to less frequent strategies likely requires improvements in the created DSL, as well as running Human-Interpret with many more clusters.

Time complexity Applying our computational method was much faster than manual human analysis. It took us approximately 6 days before we obtained the final output via Human-Interpret and heuristic analysis of its output, whereas studying human planning without the aid of AI took about 120 days. We hence sped up the whole process 20 times. If we additionally included the time needed to set up a proper DSL (refer to "Domain Specific Language (DSL) and translation dictionary" to see how it was created), the total time would accrue to 20 days. This is still a 6-fold improvement over the manual analysis. We also expect that for new problems, the reported time connected to creating the DSL would not be exceedingly different. The reason is that by building a DSL for the Mouselab-MDP we already cover a wider variety of problems: the problems we can represent concern deliberate decisions that involve selecting, processing, and integrating multiple pieces of information. Since that description applies to the majority of problems in the domain of planning, our DSL may serve as a basis for their study, and be simply extended. As extending existing work is much faster than creating something new, we think this process would not last longer than in our case. When it comes to the time required to run Human-Interpret itself, it could differ depending on the size of the considered problem.[1] In general, however, Human-Interpret can currently handle reasonably sized Mouselab-MDPs and if one is ready to sacrifice the possibility of discovering a larger number of strategies by selecting fewer clusters, it can even run in less than 3 days. Analyzing the output of Human-Interpret takes no longer than 1-2 days, irrespective of other parameters.

Generality As we mentioned in the previous paragraph, our method could, in principle, be extended to a wide range of (sequential) decision problems. For instance, consider playing chess. If we wanted to inspect human strategies in chess, the process-tracing paradigm would have to record the sequence of moves and countermoves the player is considering while deciding what to do next. To achieve this, we could provide the player a second chessboard and ask them to use it to play through what might happen depending on which move they choose next. The belief state of the metalevel MDP would encode the sequence of moves and positions the player has considered up to a given time. The predicates would encode features that predict how uncertain the player should be about alternative moves' potential. This would include how often they have simulated playing the move, how many steps they looked ahead, and how much the most recent simulations have changed the player's estimate of the move's quality (cf. Russek et al., 2022). The design of those predicates could also be informed by how grandmasters represent chess positions (e.g., Chase and Simon, 1973; McGrath et al., 2021). The experiment would ask chess players to use the second chessboard to select which moves to analyze in what order before selecting the best move on the first chessboard. We could then apply Human-Interpret to their externalized planning operations. Human-Interpret would learn interpretable strategies that procedurally described which moves to think about, depending on how they are related to the current position and the player's previous planning operations. We might thereby discover the clever planning strategies that enable chess players to efficiently identify excellent moves.

Summary The main benefit of using the introduced pipeline is that it is more objective than visual inspection. Moreover, from a long-term perspective, our automatic method is a promising step toward leveraging artificial intelligence to accelerate scientific discovery in research on decision-making and planning. Rather than trying to manually discover one strategy at a time, our method makes it possible to run many large experiments with many different environments and discover the most representative strategies people use across those environments at once. A scientist interested in human planning could then invest his or her resources to other components of their research while waiting for the computations to finish. Afterward, they could either start inspecting the data in search of more detailed strategies or use the found strategies as hypotheses to test in experimental studies. Further, our method is more objective than the subjective and potentially biased manual approach. Strategies and their descriptions are assigned based on mathematical likelihood and they are provably optimal under such probabilistic criteria, whereas people could introduce strategies based on their preconceptions and imperfect knowledge. Computers can tirelessly apply our rigorous, systematic procedure to all trials of all participants of numerous data sets. Our systematic, objective method might thereby be able to discover important strategies that scientists might otherwise overlook; one example thereof is Strategy 1 in Table 3 which likely belongs to the

[1] For a more detailed discussion of this issue, see the *Computational complexity* paragraph of the next section.

set of frequently used strategies, but the manual analysis incorporated most of it to its generalization (Strategy 4).

Limitations

A major limitation of our method is that it was unable to rediscover all the strategy types documented by Jain et al. (2022). In this section, we discuss the underlying bottlenecks and other limitations of our method.

Imperfect clustering The softmax clusters did not always contain planning operations that fit well together. For instance, the No planning Strategy 2 had a poor fit score of 0.26 with respect to people's planning operations (FPO), although the fit between the description and the cluster softmax model (FCF) was virtually perfect (1.0). This means the description was an accurate representation of the Human-Interpret cluster and the poor FPO came from incompatible planning operations. Most likely, the softmax strategies could not capture the underlying logic of the clusters, as planning operations clustered together by the generative part of Human-Interpret were too diverse.

Incomplete DSL Our analysis indicates that the DSL we used was incapable of capturing all of the strategies, which was the most evident in the case of the 3rd most important, frequent planning strategy from Jain et al. (2022) called "Consecutive second maximum strategy". This strategy was used in around 6% of the trials of the planning experiment. It observes the final outcomes in a random order but stops planning after it encounters two outcomes with the second-highest reward consecutively. The reason it was not discovered by our method was that the DSL did not include a predicate that described the second-highest reward.

Disregarding evidence for some planning strategies Table 3 shows differences in the coverage (FR) between the same strategies found by our pipeline and manual analysis. The main discrepancy is that the automatically discovered strategies have a higher frequency than the human-discovered strategies. Besides Strategies 1 and 4 which we believe to be improperly represented by the manual analysis, those differences stem from the simplification inherent to our method and used by AI-Interpret – it finds a description that fits some subset of the elements. Note that since the softmax clusters represent human planning operations when a demonstration (planning operation) of the softmax cluster is rejected by the algorithm, a human operation that corresponded to this demonstration (e.g. it followed the same planning principle) is also rejected. A rejection of a cluster-generated planning operation in AI-Interpret might occur in two situations: 1) either it is indescribable due to the imperfect DSL, or 2) it differs from other operations generated by the softmax cluster due to the imperfect clustering. In fact, we computed that AI-Interpret utilized only 67% of the cluster-generated planning operations to find the descriptions. A part of human planning operations was thus also rejected when finding a description, but still counted towards the coverage of the strategy. Based on our measure of the quality of the discovered descriptions with respect to human planning operations, i.e. the FPO which was as high as 70%, the upper bound for the number of human planning operations that were completely disregarded by Human-Interpret is probably around 30%. This issue might be solved in future work by studying more clusters and improving the DSL.

Computational complexity One of our method's limitations is that it requires a considerable amount of compute time. Because the required amount of time increase with the maximal number of clusters considered by the method, the method's computational complexity is a bottleneck to how many strategies can be discovered within a given amount of time. From the preliminary runs we tried and other gathered data, we suspect it would take about 4 times longer to run our code with up to 80 clusters. As much as this is still manageable, our experiments served mainly as a proof of concept, and thus we decided on fewer clusters. In fact, we believe that even though we did not manage to find strategies that are as fine-grained as the manually found ones due to computation time concerns, Human-Interpret can be scaled to a larger number of clusters. In our initial runs, the code took about 40 days before terminating. After optimizing our code using hashing and parallelization, the required amount of compute time dropped to 5 days. The remaining computational bottlenecks concern i) clustering sequences of planning operations via the EM algorithm (up to 2.5 days), ii) evaluating all the predicates in the DSL on all the states from the demonstrated sequences (up to 1 hour per cluster), iii) computing DNF formulas based on this evaluation (up to 1 hour per cluster). Further code optimization in those areas may improve the time complexity of the method to an even larger degree. However, our method's computational complexity will likely remain a limiting factor for some practical applications.

Bias towards fewer strategies. We strove for a balance between parsimony and expressiveness of our strategies by introducing the complexity parameters (preferring smaller decision trees in AI-Interpret, limiting their size). It is possible that by foregoing these parameters, we could discover a larger number of strategies. They could, however, be less interpretable, not always distinct, and sometimes try to explain the randomness inherent to some human decision-making.

Future work

Tackling the limitations As stated above, the main limitations of our method are imperfect softmax clustering, an incomplete DSL, and the computational bottlenecks of AI-Interpret. We could improve the clustering by imposing a no-diversity penalty on the models. Hypothetically, this would make the softmax models as different as possible and they could thus capture more diverse kinds of behaviors, for instance as in Eysenbach et al. (2018). Alternatively, we could obtain a better clustering by incorporating a different method for choosing which number of softmax clusters is optimal. Here, we used the maximum marginal likelihood or the highest BIC to differentiate between competing models (understood as numbers of clusters), but there might be better ways for accounting for model complexity, such as performing cross-validation. We see improving the DSL as an iterative refinement process in which we would add the necessary predicates to the DSL, check the results of running our computational method with the new DSL against the manual analysis, identify missing predicates, if any, and then return to the first step. Just a few iterations of this process should render a DSL capable of describing most of the strategies used by people in the Mouselab-MDP environment. We elaborate on this approach in more detail below. Both of those enhancements, however, would not be feasible without first making our computational method run faster. Thus, upgrading the implementation of AI-Interpret should be one of the first next steps.

Designing a Human-in-the-Loop version of Human-Interpret As mentioned in the previous paragraph, Human-Interpret failed to find one of the relevant strategies due to an incomplete DSL. Specifying a sufficiently expressive DSL is a general challenge for all applications of Human-Interpret. One potential remedy could be to first write an extensive version of the DSL that accounts for many, perhaps redundant predicates, and then use genetic algorithms (Kuhn & Johnson, 2019) to find a subset of the DSL that allows Human-Interpret to obtain the highest average likelihood. However, because this approach requires many iterations, this may be intractable with the current computational limitations of Human-Interpret. An iterative, human-in-the-loop version of Human-Interpret could be a more tractable remedy. To help the user refine the DSL, the algorithm could highlight trials that were difficult to classify. The user could then investigate those trials and update the DSL accordingly for a subsequent run. One possible way of achieving this is monitoring the percentage of trials whose average likelihood is not substantially different under the automatically ascribed strategy than under the random strategy. This could be formalized as the average planning operation likelihoods differing by less than δ. We call this property δ-similarity. Let P_i denote the proportion of δ-similar trials for iteration i. Assuming P_0 is 1 and all the trials are trivially assigned to the random strategy with no DSL, then whenever $P_{i-1} - P_i < \epsilon$ for some non-negative ϵ after the change in the DSL, then the algorithm could output the final set of strategies. Otherwise, the algorithm could output all δ-similar trials from iteration i for the user to analyze and find possible missing predicates that capture the poorly described behavior. Otherwise, the algorithm could output δ-similar trials from iteration i for the human to analyze and find possible missing predicates that capture the poorly described behavior. This version of the algorithm would include substantial human input and two additional hyperparameters: $\delta \in [0, 1]$ for measuring fit relative to the random policy, and $\epsilon \in [0, 1]$ for measuring improvement in the likelihood between iterations. We hence call it (δ, ϵ)-Human-Interpret. The current version of Human-Interpret is a special case of this generalization, that is $(\delta, 1)$-Human-Interpret which always stops after the initial setup of the DSL and does not focus on δ-similar trials.

Merging Human-Interpret with the Computational Microscope Besides ameliorating the performance of our method, a worthwhile future work direction is to combine the strategy discovery method presented in this article with our computational process-tracing method for measuring which planning strategy each participant used on each trial of the experiment – the Computational Microscope (Jain et al., 2022). We hypothesize that our method could serve to establish the basic set of strategies one could use as input to the microscope. Then, a human planning sequence could be automatically assigned to one of the automatically discovered strategies, instead of the manually discovered strategies. In this way, automatic strategy discovery might make it possible to speed up the development of equivalent computational process-tracing methods for other tasks and domains and to improve the repertoire of strategies that those methods use to describe the temporal evolution of people's cognitive strategies.

Researching other scenarios Finally, future work should seek to apply our method for the automatic discovery and description of human strategies to other scenarios. The methodology we presented in this paper can be easily extended to decision-making in other domains, such as risky choice, intertemporal choice, and multi-attribute decision-making (Lieder et al., 2017). Our method could thereby help distill the existing process-tracing data on these tasks (Mouselab and eye-tracking) into detailed process models of the specific heuristics people use to make those kinds of decisions. It would likely lead to the discovery of new heuristics and a more mechanistic understanding of some of the heuristic processes that are already known. It would also

help decision researchers go beyond studying individual decision strategies to automatically identify the entire toolbox of all strategies that decision-makers have available. Our method could additionally be used to accelerate the slow process of strategy discovery because it could be simultaneously applied to process-tracing data from many different types of decision environments. In that way, we could also gain insights into how the types of heuristics people use differ across different environments. Because our approach is rather general, we believe it has the potential to accelerate scientific discovery in several areas of the cognitive and behavioral sciences.

Appendix A

A.1: Human-Interpret parameters

Table 5 lists the values of the parameters of the Human-Interpret method that we used in our benchmark planning environment (the Mouselab MDP). The goal of the benchmark test was to discover and describe strategies used in this environment by people.

A.2: Defining the Domain Specific Language

Our Domain Specific Language consisted of a number of logical predicates. Every predicate we defined accepts (at least) two arguments: (belief) state of the environment b and computation/action c. In our case, the state relates to the list of expected values of nodes in the Mouselab MDP, whereas the computation is the number of the node to click, with 0 reserved for termination. The Mouselab MDP we used in the benchmark test had the form of a tree, hence a lot of the predicates made use of notions used for the tree graph structures. The meaning of the predicates, presented below in alphabetical order, is the following:

Table 5 Values for the parameters of Human-Interpret that were used in the benchmark test

Parameter	Value
exp_id	v1.0
num_participants	180
block	train & test
num_clusters	20
num_demos	128
$\mathcal{L}$	See "Appendix A.2"
features	See "Appendix A.4"
tolerance	1e−4
change_tolerance	1e−5
interpret_size	5
ai_tolerance	0.025
num_rollouts	10000
num_ai_clusters	4
expert_reward	39.97
max_divergence	0.2
threshold	0.5
allowed_predicates	See "Appendix A.3"
redundant_predicates	See "Appendix A.3"

A

`all`**(b,c,pred$_1$,pred$_2$)** : All the nodes in the MDP that satisfy $pred_1$ also satisfy $pred_2$.

`among`**(b,c,pred$_1$,pred$_2$)** : This node is among all the nodes in the MDP that satisfy $pred_1$ and inside that set it also satisfies $pred_2$.

`are_branch_leaves_observed`**(b,c)** : This node has successor leaves which are all observed.

`are_leaves_observed`**(b,c)** : All leaf nodes have been observed.

`are_roots_observed`**(b,c)** : All nodes on level 1 have been observed.

D

`depth`**(b,c,d)** : This node lies on level d.

H

`has_best_path`**(b,c,list)** : This node lies on a path for which the sum of expected rewards is the highest for the paths on which other nodes in $list$ lie.

`has_child_highest_value`**(b,c,list)** : This node has a child with an observed value that is higher than any other observed child's value for the nodes from $list$.

`has_child_highest_level_value`**(b,c)** : This node's child has the maximum possible value on its level.

`has_child_lowest_value`**(b,c,list)** : This node has a child with an observed value that is lower than any other observed child's value for the nodes from $list$.

`has_child_lowest_level_value`**(b,c)** : This node's child has the minimum posible value of its level.

has_largest_depth**(b,c,list)** : This node is the deepest in the tree among the nodes from $list$.

has_leaf_highest_value**(b,c,list)** : This node has a successor that is a leaf with an observed value that is higher than any other observed successor-leaf's value for the nodes from $list$.

has_leaf_highest_level_value**(b,c)** : This node leads to an uncovered leaf that has the maximum possible value on its level.

has_leaf_lowest_value**(b,c,list)** : This node has a successor that is a leaf with an observed value that is lower than any other observed successor-leaf's value for the nodes from $list$.

has_leaf_lowest_level_value**(b,c)** : This node leads to an uncovered leaf that has the minimum possible value on its level

has_most_branches**(b,c,list)** : This node belongs to the largest number of paths among the nodes in $list$.

has_parent_highest_value**(b,c,list)** : This node has a parent with an observed value that is higher than any other observed parent's value for the nodes from $list$.

has_parent_highest_level_value**(b,c)** : This node's parent has the maximum possible value on its level.

has_parent_lowest_value**(b,c,list)** : This node has a parent with an observed value that is lower than any other observed parent's value for the nodes from $list$.

has_parent_lowest_level_value**(b,c)** : This node's parent has the minimum possible value of its level.

has_root_highest_value**(b,c,list)** : This node has an ancestor on level 1 with an observed value that is higher than any other observed 1st-level ancestor's value for the nodes from $list$.

has_root_highest_level_value**(b,c)** : This node ca n be accessed through an observed node on level 1 which has the highest value on level 1.

has_root_lowest_value**(b,c,list)** : This node has an ancestor on level 1 with an observed value that is lower than any other observed 1st-level ancestor's value for the nodes from $list$.

has_root_lowest_level_value**(b,c)** : This node can be accessed through an observed node on level 1 which has the minimum value on level 1.

has_smallest_depth**(b,c,list)** : This node is the shallowest in the tree among the nodes from $list$.

I

is_ancestor_max_val**(b,c)** : One of the ancestors of this node is uncovered and has the maximum possible value in the MDP.

is_leaf**(b,c)** : This node is a leaf.

is_max_in_branch**(b,c)** : This node lies on a path with an uncovered maximum possible value in the MDP.

is_2max_in_branch**(b,c)** : This node lies on a path with 2 uncovered maximum possible values in the MDP.

is_observed**(b,c)** : This node was already clicked and is observed.

is_on_highest_expected_value_path**(b,c)** : This node lies on a path that has the highest expected value.

is_positive_observed**(b,c)** : There is a node with a positive value observed.

is_previous_observed_max**(b,c)** : The previously observed node uncovered the maximum possible value in the MDP.

is_previous_observed_max_leaf**(b,c)** : The previously observed node is a leaf and it uncovered the maximum possible value in the MDP.

is_previous_observed_max_level**(b,c)** : The previously observed node uncovered a maximum possible value on that level.

is_previous_observed_max_nonleaf**(b,c)** : The previously observed node isn't a leaf and it uncovered the maximum possible value in the MDP.

is_previous_observed_max_root**(b,c)** : The previously observed node lies on level 1 and it uncovered the maximum possible value in the MDP.

is_previous_observed_min**(b,c)** : The previously observed node uncovered the minimum possible value in the MDP.

`is_previous_observed_min_level`**(b,c)** : The previously observed node uncovered a minimum possible value on that level.

`is_previous_observed_parent`**(b,c)** : The previously observed node is the parent of this node.

`is_previous_observed_sibling`**(b,c)** : The previously observed node is one of the siblings of this node.

`is_root`**(b,c)** : This node is one of the nodes on level 1.

`is_successor_max_val`**(b,c)** : One of the successors of this node is uncovered and has the maximum possible value in the MDP.

O

`observed_count`**(b,c,n)** : There are at least n observed nodes.

T

`termination_return`**(b,c,e)** : The expected reward after stopping now is $\geq e$.

The DSL we used for studying Mouselab MDP policies was generated through a probabilistic context-free grammar with the following format:

START → `all`(PREDS_AMONG_PRED_DEPTH)
START → `all`(PREDS_AMONG_PRED_LEAF)
START → `all`(PREDS_AMONG_PRED_ROOT)
START → `all`(PREDS_AMONG_PRED_PARENT)
START → `all`(PREDS_AMONG_PRED_CHILD)
START → `all`(PREDS_AMONG_PRED_PRED)
START → `among`(PREDS_AMONG_PRED_DEPTH)
START → `among`(PREDS_AMONG_PRED_LEAF)
START → `among`(PREDS_AMONG_PRED_ROOT)
START → `among`(PREDS_AMONG_PRED_PARENT)
START → `among`(PREDS_AMONG_PRED_CHILD)
START → `among`(PREDS_AMONG_PRED_PRED)
START → `among`(PREDS_DEPTH)
START → `among`(PREDS_LEAF)
START → `among`(PREDS_ROOT)
START → `among`(PREDS_PARENT)
START → `among`(PREDS_CHILD)
START → `among`(PREDS_PRED)
START → PRED
START → GENERAL_PRED

PREDS_AMONG_PRED_DEPTH →
PREDS_DEPTH, AMONG_CHILD

Listing 1 Probabilistic context-free grammar that generates the predicates used by AI-Interpret and in consequence by Human-Interpret. Probability of each production is uniform with respect to the non-terminal symbol on its left hand-side

PREDS_AMONG_PRED_DEPTH →
PREDS_DEPTH, AMONG_PARENT
PREDS_AMONG_PRED_DEPTH →
PREDS_DEPTH, AMONG_LEAF
PREDS_AMONG_PRED_DEPTH →
PREDS_DEPTH, AMONG_ROOT

PREDS_AMONG_PRED_LEAF →
PREDS_LEAF, AMONG_CHILD
PREDS_AMONG_PRED_LEAF →
PREDS_LEAF, AMONG_PARENT
PREDS_AMONG_PRED_LEAF →
PREDS_LEAF, AMONG_ROOT
PREDS_AMONG_PRED_LEAF →
PREDS_LEAF, AMONG_PRED

PREDS_AMONG_PRED_ROOT →
PREDS_ROOT, AMONG_CHILD
PREDS_AMONG_PRED_ROOT →
PREDS_ROOT, AMONG_PARENT
PREDS_AMONG_PRED_ROOT →
PREDS_ROOT, AMONG_LEAF
PREDS_AMONG_PRED_ROOT →
PREDS_ROOT, AMONG_PRED

PREDS_AMONG_PRED_PARENT →
PREDS_PARENT, AMONG_CHILD
PREDS_AMONG_PRED_PARENT →
PREDS_PARENT, AMONG_LEAF
PREDS_AMONG_PRED_PARENT →
PREDS_PARENT, AMONG_ROOT
PREDS_AMONG_PRED_PARENT →
PREDS_PARENT, AMONG_PRED

PREDS_AMONG_PRED_CHILD →
PREDS_CHILD, AMONG_PARENT
PREDS_AMONG_PRED_CHILD →
PREDS_CHILD, AMONG_LEAF
PREDS_AMONG_PRED_CHILD →
PREDS_CHILD, AMONG_ROOT
PREDS_AMONG_PRED_CHILD →
PREDS_CHILD, AMONG_PRED

PREDS_AMONG_PRED_PRED →
PREDS_PRED, AMONG_CHILD
PREDS_AMONG_PRED_PRED →
PREDS_PRED, AMONG_PARENT
PREDS_AMONG_PRED_PRED →
PREDS_PRED, AMONG_LEAF
PREDS_AMONG_PRED_PRED →
PREDS_PRED, AMONG_ROOT
PREDS_AMONG_PRED_PRED →
PREDS_PRED, AMONG_PRED

PREDS_DEPTH → DEPTH
PREDS_DEPTH → DEPTH `and` LEAF
PREDS_DEPTH → DEPTH `and` ROOT
PREDS_DEPTH → DEPTH `and` PARENT
PREDS_DEPTH → DEPTH `and` CHILD

Listing 1 (continued)

```
PREDS_DEPTH → DEPTH and DEPTH
PREDS_DEPTH → DEPTH and PRED

PREDS_LEAF → LEAF
PREDS_LEAF → LEAF and LEAF
PREDS_LEAF → LEAF and ROOT
PREDS_LEAF → LEAF and PARENT
PREDS_LEAF → LEAF and CHILD
PREDS_LEAF → LEAF and PRED

PREDS_ROOT → ROOT
PREDS_ROOT → ROOT and ROOT
PREDS_ROOT → ROOT and PARENT
PREDS_ROOT → ROOT and CHILD
PREDS_ROOT → ROOT and PRED

PREDS_PARENT → PARENT
PREDS_PARENT → PARENT and PARENT
PREDS_PARENT → PARENT and CHILD
PREDS_PARENT → PARENT and PRED

PREDS_PARENT → PARENT
PREDS_PARENT → PARENT and PARENT
PREDS_PARENT → PARENT and CHILD
PREDS_PARENT → PARENT and PRED

PREDS_CHILD → CHILD
PREDS_CHILD → CHILD and CHILD
PREDS_CHILD → CHILD and PRED

PREDS_PRED → PRED
PREDS_PRED → PRED and PRED

AMONG_PRED → has_lowest_depth |
has_largest_depth
AMONG_PRED → has_best_path | has_most_paths

AMONG_CHILD → has_child_highest_value |
has_child_lowest_value
AMONG_PARENT → has_parent_highest_value |
has_parent_lowest_value
AMONG_LEAF → has_leaf_highest_value |
has_leaf_lowest_value
AMONG_ROOT → has_root_highest_value |
has_root_lowest_value

DEPTH → depth(DEP) | not(depth(DEP))
LEAF → has_leaf_highest_level_value |
has_leaf_lowest_level_value
LEAF → not(has_leaf_highest_level_value)
LEAF → not(has_leaf_lowest_level_value)
ROOT → has_root_highest_level_value |
has_root_lowest_level_value
ROOT → not(has_root_highest_level_value)
ROOT → not(has_root_lowest_level_value)
PARENT → has_parent_highest_level_value |
has_parent_lowest_level_value
PARENT → not(has_parent_highest_level_value)
PARENT → not(has_parent_lowest_level_value)
```

Listing 1 (continued)

```
CHILD → has_child_highest_level_value |
has_child_lowest_level_value
CHILD → not(has_child_highest_level_value)
CHILD → not(has_child_lowest_level_value)
PRED → is_leaf | is_root | is_max_in_branch
PRED → is_2max_in_branch |
are_branch_leaves_observed
PRED → not(is_leaf) | not(is_root) |
not(is_max_in_branch)
PRED → not(is_2max_in_branch)
PRED → not(are_branch_leaves_observed) |
not(is_observed)

GENERAL_PRED → is_previous_observed_max |
is_positive_observed
GENERAL_PRED → are_leaves_observed
GENERAL_PRED → are_roots_observed |
is_previous_observed_positive
GENERAL_PRED → is_previous_observed_parent
GENERAL_PRED → is_previous_observed_sibling
GENERAL_PRED → is_previous_observed_min
GENERAL_PRED → is_previous_observed_max_
               nonleaf
GENERAL_PRED → is_previous_observed_max_leaf
GENERAL_PRED → is_previous_observed_max_root
GENERAL_PRED → is_previous_observed_max_
               level(DEP)
GENERAL_PRED → is_previous_observed_min_
               level(DEP)
GENERAL_PRED → observed_count(NUM) |
termination_return(RET)

NUM → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8
DEP → 1 | 2 | 3
RET → -30 | -25 | -15 | -10 | 0 | 10 | 15 | 25
| 30
```

Listing 1 (continued)

A.3: Defining redundant and allowed predicates

The `redundant_predicates` parameter of HumanInterpret controls which predicates are to be removed from the DNF formula before it is turned into linear temporal logic formula. The `allowed_predicates` parameter specifies which predicates are considered for the until and unless conditions. We used the following values:

`allowed_predicates` = [All predicates derived from GENERAL_PRED in the CFG above, *is_max_in_branch*, *are_branch_leaves_observed*]

`redundant_predicates` = [All predicates of the type `all(b,c,pred1,pred2), allowed_predicates`]

A.4: Defining features for the softmax models

The generative models created in the generative step of the Human-Interpret algorithm take form of softmax functions shown in Eq. 2. Those function are defined on a set of features derived from the Mouselab MDP in a given (belief) state b while taking a given computation/action c (node to click). We used the following features in the benchmark test:

C

`count_observed_node_branch`**(b,c)** : What is the minimum of the number of observed nodes on branches that pass through the given node.

D

`depth_count`**(b,c)** : What is the number of observed nodes on the same depth as the given node.

`depth`**(b,c)** : What is the depth of the given node.

G

`get_level_observed_std`**(b,c)** : What is the standard deviation for the values of observed nodes at the same level as the given node.

H

`hp_0`**(b,c)** : Does the path that the node lies on has a value greater than 0.

I

`immediate_successor_count`**(b,c)** : What is the number of observed children of the given node.

`is_leaf`**(b,c)** : Is the given node a leaf or not.

`is_previous_max`**(b,c)** : Did the previously observed node uncover the maximum possible value in the MDP or not.

`is_root`**(b,c)** : Is the given node a root or not.

M

`most_promising`**(b,c)** : Whether the given node lies on the path with the highest expected reward or not.

O

`observed_height`**(b,c)** : What is the maximum number of consecutively observed nodes on the path of the given node, starting from its children.

P

`parent_observed`**(b,c)** : Whether the parent of the given node is observed or not.

`previous_observed_successor`**(b,c)** : Whether the previously observed node was the parent of the given node or not.

S

`second_most_promising`**(b,c)** : Whether the given node lies on the path with the second highest expected reward or not.

`siblings_count`**(b,c)** : What is the number of observed siblings of the given node.

`soft_satisficing`**(b,c)** : What is the expected reward of terminating now and traversing the most promising path.

`successor_uncertainty`**(b,c)** : What is the total standard deviation for the children of the given node.

T

`termination_constant`**(b,c)** : The value of this feature is 0 for the termination action and is -1 for all the nodes.

U

`uncertainty`**(b,c)** : What is the total standard deviation for the nodes on the same depth as the given node.

A.5: Benchmark test results

As mentioned in the main text, the 21 runs of Human-Interpret returned a set of 21 unique strategies. Here, we extend Table 3 from the main text, and in Table 6 present all of the discovered strategies alongside their LTL formulas. Note that sometimes strategies discovered in different runs differed by 1 predicate (e.g. `depth(3)` vs. `is_leaf`) and so we listed equivalent predicates in brackets.

Table 6 Strategies found in the median combined 10-run of Human-Interpret listed with their ID from the main text

ID	Generated formulas and descriptions	Statistics					
		FR	FCF	FON	FPO	N	GT
1	**Summary:** Search for the best possible outcome sibling by sibling. **Description:** 1. Click on a node satisfying all of the following conditions: – it is an unobserved leaf. 2. Unless the previously observed node uncovers a 48, in which case stop at the previous step, click on a node satisfying all of the following conditions: – it is an unobserved leaf. click in this way under the condition that: – the previously observed node was its sibling. 3. GOTO step 1 unless all the leaves are observed or the previously observed node uncovers a 48. **LTL formula:** `among(not(is_observed) and is_leaf) AND NEXT among(not(is_observed) and is_leaf) and is_previous_observed_sibling UNLESS is_previous_observed_max_leaf LOOP FROM among (not(is_observed) and is_leaf) UNLESS (are_leaves_observed or is_previous_observed_ max_leaf)`	32.3%	0.98	0.91	0.84	21	Y
2	**Summary:** No planning. **Description:** Do not click. **LTL formula:** `False UNTIL IT STOPS APPLYING`	24.6%	1	0.51	0.41	20	Y
3	**Summary:** Click all immediate outcomes. **Description:** 1. Click on the nodes satisfying all of the following conditions: – they are unobserved roots. Repeat this step as long as possible. **LTL formula:** `among(not(is_observed) and is_root) UNTIL IT STOPS APPLYING`	4.2%	1	0.74	0.61	3	Y
4	**Summary:** Search for the best possible outcome. **Description:** 1. Click on the nodes satisfying all of the following conditions: – they are unobserved leaves. Repeat this step until all the leaves are observed or the previously observed node uncovers a 48. **LTL formula:** `among(not(is_observed) and is_leaf) UNTIL (are_leaves_observed or is_previous_observed_max_leaf)`	17.2%	0.88	0.87	0.79	21	Y

Table 6 (continued)

ID	Generated formulas and descriptions	Statistics					
		FR	FCF	FON	FPO	N	GT
6	**Summary:** Click all final outcomes sibling by sibling. **Description:** 1. Click on a node satisfying all of the following conditions: – it is an unobserved leaf. 2. Click on a node satisfying all of the following conditions: – it is an unobserved leaf. – the previously observed node was its sibling 3. GOTO step 1. LTL formula: `among(not(is_observed) and is_leaf) AND NEXT among(not(is_observed) and is_leaf) and is_previous_observed_sibling LOOP FROM among(not(is_observed) and is_leaf)` Description: 1. Click on the nodes satisfying all of the following conditions: – they are unobserved nodes that lead to leaves whose value is different from -48 – they are located on the highest level considering the unobserved nodes that lead to leaves whose value is different from -48. Repeat this step until a node with a positive value is observed or the previously observed node uncovers a -48. 2. Click on a node satisfying all of the following conditions: – it is an unobserved node that leads to a leaf whose value is different from -48 – it is located on the highest level considering the unobserved nodes that lead to leaves whose value is different from -48 – it is the previously observed node was its sibling. 3. GOTO step 1 unless all the leaves are observed or all the roots are observed. **LTL formula:** `among(not(has_leaf_lowest_level_value) and not(is_observed) : has_largest_depth) UNTIL (is_positive_observed or is_previous_observed_min_level(3)) AND NEXT among(not(has_leaf_lowest_level_value) and not(is_observed) : has_largest_depth) and is_previous_observed_sibling LOOP FROM among(not(has_leaf_lowest_level_value) and not(is_observed) : has_largest_depth) UNLESS (are_leaves_observed or are_roots_observed)`	33.5%	0.74	0.65	0.47	18	N
5	**Summary:** Best first search. **Description:** 1. Click on the nodes satisfying all of the following conditions: – they are unobserved nodes – they have parents with the highest values considering the parents of other unobserved nodes. Repeat this step as long as possible. **LTL formula:** `among(not(is_observed) : has_parent_highest_value) UNTIL IT STOPS APPLYING`	3.8%	0.63	0.67	0.64	7	Y

Table 6 (continued)

ID	Generated formulas and descriptions	Statistics					
		FR	FCF	FON	FPO	N	GT
	Summary: Depth first search.						
	Description:	9.1%	0.84	0.72	0.6	7	Y
	1. Click on a node satisfying all of the following conditions: – it is an unobserved node – it has a parent with the lowest value considering the parents of other unobserved nodes. 2. Click on the nodes satisfying all of the following conditions: – they are unobserved non-roots – they have parents with the highest values considering the parents of other unobserved non-roots. 7 Repeat this step as long as possible. 3. GOTO step 1.						
	LTL formula: `among(not(is_observed) : has_parent_[lowest|highest]_value) AND NEXT among(not(is_observed) and not(is_root) : has_parent_highest_value) UNTIL IT STOPS APPLYING LOOP FROM among(not(is_observed) : has_parent_[lowest|highest]_value)`						
	Summary: Depth first search.						
	Description:	9.1%	0.84	0.72	0.6	7	Y
	1. Click on a node satisfying all of the following conditions: – it is an unobserved node – it has a parent with the lowest value considering the parents of other unobserved nodes. 2. Click on the nodes satisfying all of the following conditions: – they are unobserved non-roots – they have parents with the highest values considering the parents of other unobserved non-roots. 7 Repeat this step as long as possible. 3. GOTO step 1.						
	LTL formula: `among(not(is_observed) : has_parent_[lowest|highest]_value) AND NEXT among(not(is_observed) and not(is_root) : has_parent_highest_value) UNTIL IT STOPS APPLYING LOOP FROM among(not(is_observed) : has_parent_[lowest|highest]_value)`						
	Summary: Click on a node and all its successors. Repeat or terminate.						
	Description:	4%	0.58	0.84	0.73	5	Y
	1. Click on a random node or terminate. 2. Click on the nodes satisfying all of the following: – they are unobserved non-roots – they have parents with the lowest (highest) values considering 9 the parents of other unobserved non-roots. Repeat this step as long as possible. 3. GOTO step 1.						
	LTL formula: `True AND NEXT among(not(is_observed) and not(is_root) : has_parent_[lowest|highest]_value) UNTIL IT STOPS APPLYING GO TO True`						

Table 6 (continued)

ID	Generated formulas and descriptions	Statistics					
		FR	FCF	FON	FPO	N	GT
8	**Summary:** Depth first search until finding the best final outcome. **Description:** 1. Click on a node satisfying all of the following conditions: – it is an unobserved node – it has a parent with the lowest value considering the parents of other unobserved nodes. Click in this way under the condition that: – the previously observed node uncovered something else than a 48. 2. Click on the nodes satisfying all of the following conditions: – they are unobserved non-roots – they have parents with the highest values considering the parents of other unobserved non-roots. Repeat this step as long as possible. 3. GOTO step 1 unless all the leaves are observed or the previously observed node uncovers a 48. **LTL formula:** `among(not(is_observed) : has_parent_[lowest|highest]]_value) and not(is_previous_observed_max_leaf) AND NEXT among(not(is_observed) and not(is_root) : has_parent_highest_value) UNTIL IT STOPS APPLYING LOOP FROM among(not(is_observed) : has_parent_[lowest|highest]_value) and not(is_previous_observed_max_leaf)`	6.0%	0.92	0.71	0.71	1	Y
11	**Summary:** Depth first search on a random branch. **Description:** 1. Click on a node satisfying all of the following conditions: – it is an unobserved node – it has a parent with the lowest value considering the parents of other unobserved nodes. 2. Click on the nodes satisfying all of the following conditions: – they are unobserved non-roots – they have parents with the highest values considering the parents of other unobserved non-roots. Repeat this step as long as possible. **LTL formula:** `among(not(has_child_lowest_level_value) and not(is_observed) : has_parent_[lowest|highest]_value) AND NEXT among(not(is_observed) and not(is_root) : has_parent_highest_value) UNTIL IT STOPS APPLYING`	3.8%	0.75	0.63	0.63	1	N
14	**Summary:** Click all immediate outcomes and continue at random. **Description:** 1. Click on the nodes satisfying all of the following conditions: – they are unobserved roots. Repeat this step until all the roots are observed. 2. Terminate or click on some random nodes and then terminate. Repeat this step as long as possible. **LTL formula:** `among(not(is_observed) and is_root) UNTIL are_roots_observed AND NEXT True UNTIL IT STOPS APPLYING`	2%	0.68	0.99	0.98	4	N

Table 6 (continued)

ID	Generated formulas and descriptions	Statistics					
		FR	FCF	FON	FPO	N	GT
16	**Summary:** Click a random path or a branch. **Description:** 1. Click on a node satisfying all of the following conditions: – it is a leaf – lies on a best path. 2. Click on the nodes satisfying all of the following conditions: – they are leaves – lie on best paths. Click in this way as long as: – the previously observed node was their sibling. Repeat this step until a node with a positive value is observed. 3. Click on the nodes satisfying all of the following conditions: – they are unobserved nodes – they have children with the highest values considering the children of other unobserved nodes. Repeat this step as long as possible. **LTL formula:** `among(is_leaf : has_best_path) AND NEXT among(is_leaf : has_best_path) and is_previous_observed_sibling UNTIL is_positive_observed AND NEXT among(not(is_observed) : has_child_highest_value) UNTIL IT STOPS APPLYING`	5.7%	0.55	0.47	0.34	3	N
13	**Summary:** Click all final outcomes. **Description:** 1. Click on the nodes satisfying all of the following conditions: – they are unobserved leaves. Repeat this step until it stops applying. **LTL formula:** `among(not(is_observed) and is_leaf) UNTIL IT STOPS APPLYING`	5.6%	0.69	0.74	0.55	6	Y

Table 6 (continued)

ID	Generated formulas and descriptions	Statistics					
		FR	FCF	FON	FPO	N	GT
	Summary: Click two sibling final outcomes and maybe one random outcome.						
	Description*:	5.9%	0.45	0.48	0.25	5	N
	1. Click on random nodes. Do not click on the nodes satisfying either of the following conditions: – they are non-leaves that have children with the non-highest value on their level. 2. Click on random nodes. Click in this way as long as: – the previously observed node was their sibling. Do not click on the nodes satisfying either of the following conditions: – they are non-leaves that have children with the non-highest value on their level. Repeat this step until this node belongs to a subtree with all leaves already observed. 3. Click on a random node or terminate. **LTL formula:** `not(among(not(has_child_highest_level_value) and not(is_leaf))) UNTIL (is_positive_observed or is_previous_observed_min_level(3)) AND NEXT not(among(not(has_child_highest_level_value) and not(is_leaf))) and is_previous_observed_sibling UNTIL[are_branch_leaves_observed AND NEXT True`						
20	**Description:** 1. Click on a node satisfying all of the following conditions: – it is an unobserved leaf. 2. Click on random nodes. Click in this way as long as: – the previously observed node was their sibling. Do not click on the nodes satisfying either of the following conditions: – they are nodes belonging to a subtree with some unobserved leaves. Repeat this step as long as possible. **LTL formula:** `among(is_leaf and not(is_observed)) AND NEXT is_previous_observed_sibling and not(are_branch_leaves_observed)) UNTIL IT STOPS APPLYING`						
	Summary: Search for the best final outcome in pairs.						
	Description:	4%	0.58	0.84	0.73	2	N
	1. Click on a node satisfying all of the following conditions: – it is an unobserved leaf 2. Click on a nodes satisfying all of the following conditions: – it is an unobserved leaf Click in this way under the condition that – the previously observed node was its sibling. 3. GOTO step 1 unless all the leaves are observed or the previously observed node uncovers a 48. **LTL formula:** `among(is_leaf and not(is_observed)) AND NEXT among(is_leaf and not(is_observed)) and is_previous_observed_sibling LOOP FROM among(is_leaf and not(is_observed)) UNLESS (are_leaves_observed or [is_previous_observed_max_leaf | is_previous_observed_max])`						

Table 6 (continued)

ID	Generated formulas and descriptions	Statistics					
		FR	FCF	FON	FPO	N	GT
	Summary: Click 2 sibling final outcomes.						
	Description:	6.1%	0.54	0.54	0.37	6	N
	1. Click on a node satisfying all of the following conditions: – it is an unobserved leaf 2. Click on the nodes satisfying all of the following conditions: – they are unobserved leaves						
15	Click in this way as long as: – the previously observed node was their sibling. Repeat this step as long as possible.						
	LTL formula: `among(is_leaf and not(is_observed)) AND NEXT among(is_leaf and not(is_observed)) and is_previous_observed_sibling UNTIL IT STOPS APPLYING`						
	Summary: Random planning.						
	Description:	4.1%	0.57	1.0	1.0	20	Y
21	1. Terminate or click on some random nodes and then terminate. Repeat this step as long as possible.						
	LTL formula: `True UNTIL IT STOPS APPLYING`						
	Summary: Depth first search on two random branches.						
	Description:	3%	0.7	0.66	0.54	2	N
	1. Click on a node satisfying all of the following conditions: – it is an unobserved node – it has a parent with the lowest value considering the parents of other unobserved nodes. 2. Click on the nodes satisfying all of the following conditions: – they are unobserved non-roots – they have parents with the highest values considering the parents of other unobserved non-roots. Repeat this step as long as possible. 3. Click on a node satisfying all of the following conditions: – it is an unobserved node – it has a parent with the lowest value considering the parents of other unobserved nodes.						
12	4. Click on the nodes satisfying all of the following conditions: – they are unobserved non-roots – they have parents with the highest values considering the parents of other unobserved non-roots. Repeat this step as long as possible.						
	LTL formula: `among(not(has_child_lowest_level_value) and not(is_observed) : has_parent_[lowest|highest]_value) AND NEXT among(not(is_observed) and not(is_root) : has_parent_highest_value) UNTIL IT STOPS APPLYING AND NEXT mong(not(has_child_lowest_level_value) and not(is_observed) : has_parent_[lowest|highest]_value) AND NEXT among(not(is_observed) and not(is_root) : has_parent_highest_value) UNTIL IT STOPS APPLYING`						

Table 6 (continued)

ID	Generated formulas and descriptions	Statistics					
		FR	FCF	FON	FPO	N	GT
	Summary: Depth First Search until a positive value.						
	Description: 1. Click on a node satisfying all of the following conditions: – it is an unobserved node – it has a parent with the lowest value considering the parents of other unobserved nodes. 2. Click on the nodes satisfying all of the following conditions: – they are unobserved non-roots – they have parents with the highest values considering the parents of other unobserved non-roots. Repeat this step as long as possible 3. Click on a node satisfying all of the following conditions: – it is an unobserved node – it has a parent with the lowest value considering the parents of other unobserved nodes. 4. Click on the nodes satisfying all of the following conditions: – they are unobserved non-roots – they have parents with the highest values considering the parents of other unobserved non-roots. – this step until all the roots are observed or the previously observed node uncovers a positive value 5. GOTO step 3.	2.8%	0.85	0.68	0.47	1	N
17	**LTL formula:** `among(not(is_observed) : has_parent_lowest_value) AND NEXT among(not(is_root) and not(is_observed) : has_parent_highest_value) UNTIL IT STOPS APPLYING AND NEXT among(not(is_observed) : has_parent_lowest_value) AND NEXT among(not(is_root) and not(is_observed) : has_parent_highest_value) UNTIL (are_roots_observed or is_previous_observed_positive)`						

Table 6 (continued)

ID	Generated formulas and descriptions	Statistics					
		FR	FCF	FON	FPO	N	GT
18	**Summary:** Click sibling final outcomes and a middle node. **Description:** 1. Click on random nodes. Do not click on the nodes satisfying either of the following conditions: – they are non-leaves that have children with the non-highest value on their level. Repeat this step until a node with a positive value is observed or the previously observed node uncovers a -48. 2. Click on random nodes. Click in this way as long as: – the previously observed node was their sibling. Do not click on the nodes satisfying either of the following conditions: – they are non-leaves that have children with the non-highest value on their level. Repeat this step until a node with a positive value is observed or the termination rewrd is -30. 3. Click on a node satisfying all of the following conditions: – it is a non-leaf that has a child with the non-highest value on its level. **LTL formula:** `not(among(not(has_child_highes_level_value) and not(is_leaf))) UNTIL (is_positive_observed or is_previous_observed_min) AND NEXT not(among(not(has_child_highest_level_value) and not(is_leaf))) and is_previous_observed_sibling UNTIL (is_positive_observed or termination_reward(-30)) AND NEXT among(not(has_child_highest_level_value) and not(is_leaf))`	5.5%	0.47	0.5	0.24	1	Y

Table 6 (continued)

ID	Generated formulas and descriptions	Statistics					
		FR	FCF	FON	FPO	N	GT
	Summary: Depth First Search until a positive valued path.						
19	Description: 1. Click on the nodes satisfying all of the following conditions: – they are unobserved nodes that lead to leaves whose value is different from -48 – they are located on the highest level considering the unobserved nodes that lead to leaves whose value is different from -48. Repeat this step until a node with a positive value is observed or the previously observed node uncovers a -48. 2. Click on the nodes satisfying all of the following conditions: – it is an unobserved node that leads to a leaf whose value is different from -48 – it is located on the highest level considering the unobserved nodes that lead to leaves whose value is different from -48. – it is the previously observed node was its sibling. 3. GOTO step 1 unless all the leaves are observed or all the roots are observed. LTL formula: `among(not(has_leaf_lowest_level_value) and not(is_observed) : has_largest_depth) UNTIL (is_positive_observed or is_previous_observed_min) AND NEXT among(not(has_leaf_lowest_level_value) and not(is_observed) : has_largest_depth) and is_previous_observed_sibling LOOP FROM among(not(has_leaf_lowest_level_value) and not(is_observed) : has_largest_depth) UNLESS (are_leaves_observed or are_roots_observed)`	3.9%	0.82	0.52	0.41	1	N

For each strategy, we provide automatically generated descriptions that represent that strategy, and a summary of that strategy that we created by hand. FR denotes the frequency of the strategy; FCF (fit cluster-formula) averages two proportions: formula demonstrations agreeing with the softmax clusters and vice-versa measured using 100000 demonstrations; FON (fit optimal-non-optimal) quantifies how often people's planning operations in the cluster agreed with the description; FPO (fit per operation) is the ratio between the average likelihood per planning operation belonging to the cluster and the average likelihood per planning operation for the (policy induced by the) cluster's description in general; N is the total number of clusters encoding a strategy

*One strategy listed steps 1 and 2 twice

Funding Open Access funding enabled and organized by Projekt DEAL. This project was funded by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039B.

Data Availability Anonymized data from the experiment we used as our benchmark experiment is available at https://github.com/RationalityEnhancement/InterpretableHumanPlanning/tree/main/data/human.

Code Availability The code for Human-Interpret is available at https://github.com/RationalityEnhancement/InterpretableHumanPlanning.

Declarations

Ethics approval and consent to participate Not applicable

Consent for Publication Not applicable.

Conflict of Interests The authors declare that they have no conflicts of interest or competing interests.

References

Abelson, R., & Levi, A. (1985). Decision making and decision theory. In Lindzey, G., & Aronson, E. (Eds.) *Handbook of social psychology*. Hillsdale, NJ: Erlbaum.

Addis, M., Sozou, P.D., Lane, P.C., & Gobet, F. (2016). Computational scientific discovery and cognitive science theories. In *Computing and philosophy, Springer, pp 83–97*.

Agrawal, M., Peterson, J. C., & Griffiths, T. L. (2020). Scaling up psychology via scientific regret minimization. *Proceedings of the National Academy of Sciences*, *117*(16), 8825–8835.

Araki, B., Vodrahalli, K., Leech, T., Vasile, C. I., Donahue, M., & Rus, D. (2019). Learning to Plan with Logical Automata. In *Robotics: Science and systems conference (RSS), Messe Freiburg, Germany*, (pp. 1-9).

Bacon, F. Fowler, T. (Ed.) (1878). Oxford: Clarendon Press.

Becker, F., Skirzyński, J., Van Opheusden, B., & Lieder, F. (2022). Boosting human decision-making with ai-generated decision aids. Computational Brain & Behavior.

Bettman, J. R., Johnson, E. J., & Payne, J. W. (1990). A componential analysis of cognitive effort in choice. *Organizational behavior and human decision processes*, *45*(1), 111–139.

Bhatia, S., & He, L. (2021). Machine-generated theories of human decision-making. *Science*, *372*(6547), 1150–1151.

Botvinick, M. M., Niv, Y., & Barto, A. G. (2009). Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition*, *113*(3), 262–280.

Callaway, F., Lieder, F., Krueger, P. M., & Griffiths, T. L. (2017). Mouselab-MDP: A new paradigm for tracing how people plan. In *The 3rd Multidisciplinary Conference on Reinforcement Learning and Decision Making, Ann Arbor, MI, https://osf.io/vmkrq/*.

Callaway, F., Lieder, F., Das, P., Gul, S., Krueger, P. M., & Griffiths, T. (2018). A resource-rational analysis of human planning.

Callaway, F., van Opheusden, B., Gul, S., Das, P., Krueger, P., Lieder, F., & Griffiths, T. (2020). Human planning as optimal information seeking. Manuscript under review.

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive psychology*, *4*(1), 55–81.

Consul, S., Heindrich, L., Stojcheski, J., & Lieder, F. (2021). Improving human decision-making by discovering efficient strategies for hierarchical planning. arXiv preprint arXiv: 210200521.

Cook, G. J., & Swain, M. R. (1993). A computerized approach to decision process tracing for decision support system design. *Decision Sciences*, *24*(5), 931–952.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, *39*(1), 1–22.

Džeroski, S., Langley, P., & Todorovski, L. (2007). Computational discovery of scientific knowledge. In *Computational Discovery of Scientific Knowledge, Springer, pp 1–14*.

Eysenbach, B., Gupta, A., Ibarz, J., & Levine, S. (2018). Diversity is all you need: Learning skills without a reward function. arXiv preprint arXiv: 180206070.

Fang, J., Schooler, L., & Shenghua, L. (2022). Machine learning strategy identification: a paradigm to uncover decision strategies with high fidelity. Behavior Research Methods, pp. 1–22.

Ford, J. K., Schmitt, N., Schechtman, S. L., Hults, B. M., & Doherty, M. L. (1989). Process tracing methods: contributions, problems, and neglected research questions. *Organizational behavior and human decision processes*, *43*(1), 75–117.

Foster, A., Jankowiak, M., Bingham, E., Horsfall, P., Teh, Y. W., Rainforth, T., & Goodman, N. (2019). Variational bayesian optimal experimental design. arXiv preprint arXiv: 190305480.

Griffiths, T. L., Callaway, F., Chang, M. B., Grant, E., Krueger, P. M., & Lieder, F. (2019). Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, *29*, 24–30.

Huber, O., Wider, R., & Huber, O. W. (1997). Active information search and complete information presentation in naturalistic risky decision tasks. *Acta Psychologica*, *95*(1), 15–29.

Huys, Q. J., Eshel, N., O'Nions, E., Sheridan, L., Dayan, P., & Roiser, J. P. (2012). Bonsai trees in your head: how the Pavlovian system sculpts goal-directed choices by pruning decision trees. PLoS Computational Biology 8(3).

Huys, Q. J. M., Lally, N., Faulkner, P., Eshel, N., Seifritz, E., Gershman, S. J., . . . , Roiser, J. P. (2015). Interplay of approximate planning strategies. *Proceedings of the National Academy of Sciences*, *112*(10), 3098–3103.

Jain, Y. R., Callaway, F., Griffiths, T. L., Dayan, P., He, R., Krueger, P. M., & Lieder, F. (2022). A computational process-tracing method for measuring people's planning strategies and how they change over time. Behavior Research Methods, pp. 1–43.

Jasper, J., & Shapiro, J. (2002). Mousetrace: A better mousetrap for catching decision processes. *Behavior Research Methods Instruments, & Computers*, *34*(3), 364–374.

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, *90*(430), 773–795.

Kemtur, A., Jain, Y. R., Mehta, A., Callaway, F., Consul, S., Stojcheski, J., & Lieder, F. (2020). Leveraging machine learning to automatically derive robust planning strategies from biased models of the environment. In *Proceedings of the 42nd Annual Conference of the Cognitive Science Society, pp 2405–2411*.

Konishi, S., & Kitagawa, G. (2008). Information criteria and statistical modeling. Springer Science & Business Media.
Kuhn, M., & Johnson, k. (2019). Feature engineering and selection: A practical approach for predictive models. CRC Press.
Kuhn, T. S. (1962). *The structure of scientific revolutions*. Chicago: University of Chicago Press.
Langley, P., Zytkow, J. M., Bradshaw, G. L., & Simon, H. A. (1983). Three facets of scientific discovery. In *IJCAI, Citeseer, pp 465–468*.
Langley, P., Simon, H. A., Bradshaw, G. L., & Zytkow, J. M. (1987). *Scientific discovery: Computational explorations of the creative processes*. Cambridge: MIT press.
Lieder, F., Krueger, P. M., & Griffiths, T. (2017). An automatic method for discovering rational heuristics for risky choice. In *CogSci*.
Lieder, F., Callaway, F., Jain, Y., Krueger, P., Das, P., Gul, S., & Griffiths, T. (2019). A cognitive tutor for helping people overcome present bias. In *RLDM 2019*.
Lieder, F., Callaway, F., Jain, Y. R., Das, P., Iwama, G., Gul, S., . . . , Griffiths, T. L. (2020), Leveraging artificial intelligence to improve people's planning strategies. Manuscript in revision.
Liu, G., Schulte, O., Zhu, W., & Li, Q. (2018). Toward interpretable deep reinforcement learning with linear model u-trees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp 414–429*.
McGrath, T., Kapishnikov, A., Tomašev, N., Pearce, A., Hassabis, D., Kim, B., . . . , Kramnik, V. (2021). Acquisition of chess knowledge in alphazero. arXiv preprint arXiv: 211109259.
McNicol, D. (2005). *A primer of signal detection theory*. London: Psychology Press.
Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal processing magazine*, *13*(6), 47–60.
Myung, J. I., Cavagnaro, D. R., & Pitt, M. A. (2013). A tutorial on adaptive design optimization. *Journal of mathematical psychology*, *57*(3-4), 53–67.
Newell, A., Simon, H. A., et al. (1972). *Human problem solving vol 104*. Englewood Cliffs, NJ: Prentice-Hall.
Newton, I. (1687). *Philosophiae naturalis principia mathematica*. London: William Dawson & Sons Ltd.
Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., Peters, J., et al. (2018). An algorithmic perspective on imitation learning. *Foundations and Trends®, in Robotics*, *7*(1-2), 1–179.
Ouyang, L., Tessler, M. H., Ly, D., & Goodman, N. (2016). Practical optimal experiment design with probabilistic programs. arXiv preprint arXiv: 160805046.
Ouyang, L., Tessler, M. H., Ly, D., & Goodman, N. D. (2018). Webppl-oed: A practical optimal experiment design system. In *CogSci*.
Payne, J. W., Braunstein, M. L., & Carroll, J. S. (1978). Exploring predecisional behavior: an alternative approach to decision research. *Organizational Behavior and Human Performance*, *22*(1), 17–44.
Payne, J. W., Bettman, J. R., & Johnson, E. J. (1988). Adaptive strategy selection in decision making. *Journal of experimental psychology: Learning, Memory, and Cognition*, *14*(3), 534.
Payne, J. W. (1993). *The adaptive decision maker*. Cambridge: Cambridge University Press.
Peterson, J. C., Bourgin, D. D., Agrawal, M., Reichman, D., & Griffiths, T. L. (2021). Using large-scale experiments and machine learning to discover theories of human decision-making. *Science*, *372*(6547), 1209–1214.
Popper, K. (1935). *The Logic of Scientific Discovery*. Evanston, IL: Routledge.
Raftery, A. E. (1995). Bayesian model selection in social research. *Sociological methodology*, *25*, 111–163.
Reichenbach, H. (1938). *Experience and prediction an analysis of the foundations and the structure of knowledge*. Chicago: The University of Chicago Press.
Riedl, R., Brandstätter, E., & Roithmayr, F. (2008). Identifying decision strategies: a process-and outcome-based classification method. *Behavior research methods*, *40*(3), 795–807.
Russek, E., Acosta-Kane, D., van Opheusden, B., Mattar, M.G., & Griffiths, T. (2022). Time spent thinking in online chess reflects the value of computation. PsyArXiv.
Shrager, J., & Langley, P. Shrager, J., & Langley, P. (Eds.) (1990). *Computational approaches to scientific discovery*. San Mateo, CA: Morgan Kaufmann.
Silver, T., Allen, K. R., Lew, A. K., Kaelbling, L. P., & Tenenbaum, J. (2020). Few-shot bayesian imitation learning with logical program policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, (Vol. 34, pp. 10251-10258).
Simon, H. A. (1973). Does scientific discovery have a logic? *Philosophy of science*, *40*(4), 471–480.
Simon, H. A., & Newell, A. (1971). Human problem solving: The state of the theory in 1970. *American Psychologist*, *26*(2), 145.
Skirzyński, J., Becker, F., & Lieder, F. (2021). Automatic discovery of interpretable planning strategies. *Machine Learning*, *110*, 2641–2683.
Sozou, P. D., Lane, P. C., Addis, M., & Gobet, F. (2017). Computational scientific discovery. Springer handbook of model-based science 719–734.
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. Cambridge: MIT press.
Svenson, O. (1979). Process descriptions of decision making. *Organizational behavior and human performance*, *23*(1), 86–112.
Verma, A., Murali, V., Singh, R., Kohli, P., & Chaudhuri, S. (2018). Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning, PMLR, pp 5045–5054*.
Vincent, B. T., & Rainforth, T. (2017). The darc toolbox: automated, flexible, and efficient delayed and risky choice experiments using bayesian adaptive design. PsyArXiv October 20.
Vrieze, S. I. (2012). Model selection and psychological theory: a discussion of the differences between the akaike information criterion (aic) and the bayesian information criterion (bic). *Psychological methods*, *17*(2), 228.
Westenberg, M. R., & Koele, P. (1994). Multi-attribute evaluation processes: Methodological and conceptual issues. *Acta Psychologica*, *87*(2-3), 65–84.
Whewell, W. (1840). The philosophy of the inductive sciences: founded upon their history, vol 1. JW Parker.
Willemsen, M., & Johnson, E. (2011). Visiting the decision factory: observing cognition with MouselabWEB and other information acquisition Methods, pp. 19–42.