Fanoos: Multi-Resolution, Multi-Strength, Interactive Explanations for Learned Systems[*]

David Bayani and Stefan Mitsch

Computer Science Department
Carnegie Mellon University, Pittsburgh PA 15213, USA
dcbayani@andrew.cmu.edu, smitsch@cs.cmu.edu

Abstract. Machine learning becomes increasingly important to tune or even synthesize the behavior of safety-critical components in highly non-trivial environments, where the inability to understand learned components in general, and neural nets in particular, poses serious obstacles to their adoption. Explainability and interpretability methods for learned systems have gained considerable academic attention, but the focus of current approaches on only one aspect of explanation, at a fixed level of abstraction, and limited if any formal guarantees, prevents those explanations from being digestible by the relevant stakeholders (e.g., end users, certification authorities, engineers) with their diverse backgrounds and situation-specific needs. We introduce Fanoos, a flexible framework for combining formal verification techniques, heuristic search, and user interaction to explore explanations at the desired level of granularity and fidelity. We demonstrate the ability of Fanoos to produce and adjust the abstractness of explanations in response to user requests on a learned controller for an inverted double pendulum and on a learned CPU usage model.

1 Problem Overview

Explainability and safety in AI—particularly in systems tuned or synthesized using Machine Learning (ML)—are an increasing subject of academic and public concern. As machine learning continues to grow in success and adoption by wide-ranging industries, the impact of these algorithms' behavior on people's lives is becoming highly non-trivial. Unfortunately, many of the most performant contemporary ML algorithms—neural networks (NNs) in particular—are widely considered black-boxes, with the method by which they perform their

[*] This material is based upon work supported by the United States Air Force and DARPA under Contract No. FA8750-18-C-0092. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA. We would like to thank David Held for pointing us to the rl-baselines-zoo repository. Also, we would like to thank Nicholay Topin for supporting our spirits at some key junctures of this work.

duties not being amenable to direct human comprehension. The inability to understand learned components as thoroughly as more traditional software poses serious obstacles to their adoption [6,1,11,27,78,26,79,46] due to safety concerns, difficulty to debug and maintain, and explicit legal requirements, such as the right to an explanation legislation adopted by the European Union[22]. Symbiotic human-machine interactions can lead to safer and more robust agents, but this task requires effective and versatile communication [68,60].

Interpretability of learned systems has been studied in the context of computer science intermittently since at least the late 1980s, particularly in the area of rule extraction (e.g., [5]), inductive logic programming [49], association rule learning [3] and its predecessors (e.g., [55,23]), and in adaptive/non-linear control analysis (e.g., [16]). Interpretability is also motivation to fundamental formal analysis approaches (e.g., [13,75,76,70,37,48]), but gained more significant attention only recently owing in part to its increased impact on daily life [1] with initiatives, such as the DARPA XAI project [28] and the DARPA Assured Autonomy Program [50,51], the IJCAI-XAI workshop [4], and the ICAPS XAIP Workshop [77].

Despite this attention, however, most explanatory-systems developed for ML are hard-coded to provide a single type of explanation with descriptions at a certain fixed level of abstraction and a fixed type of guarantee about the system behavior, if any. This not only prevents the explanations generated from being digestible by multiple audiences (the end-user, the intermediate engineers who are non-experts in the ML component, and the ML-engineer for instance), but in fact limits the use by any single audience since the levels of abstraction and formal guarantees needed are situation and goal specific, not just a function of the recipient's background. When using a microscope, one varies between low- and high- magnification in order to find what they are looking for and explore samples; these same capabilities are desirable for XAI for much the same reasons. For example, when determining the reaction of an autonomous vehicle when a person steps in front of it, most audiences may prefer to ask generally (e.g., "When you detect a person in front of you, what do you do?") and receive a break-down of qualitatively different behaviors for different situations, such as braking when traveling slowly enough, and doing a sharp swerve when traveling too fast to brake. Sometimes, however, an engineer might still want to specify precise starting locations of the car and person and have the car report exact motor commands so to ensure actuators are compliant; the context of use and the audience-type determine which level of abstraction is best, and supporting multiple types of abstractions in turn supports more use-cases and audiences. Further, the explanations for such a component need to range from formal guarantees to rough tendencies—one might want to formally guarantee that the car will avoid collisions always, while it might be sufficient that it usually (but perhaps not always) drives slowly when its battery is low.

The divide between formal and probabilistic explanations also relates to events that are imaginable versus events that may actually occur; formal methods may check every point in a space for conformance to a condition, but if bad behavior only occurs on measure-zero sets, the system would be safe while not being provably so in formalizations lacking knowledge of statistics (e.g., when a car must keep distance >10 cm to obstacles, formally we can get arbitrarily close but not equal; in practice, the difference with ≥10 cm might be irrelevant). Explainable ML systems should enable these sorts of search and smooth variation in need—but at the moment they do not in general.

To address these challenges, we propose a combination of formal verification, heuristic search, and user interaction capable of providing explanations for currently ubiquitous ML methods—such as feed-forward neural networks (FFNNs) and high-dimensional polynomial kernels —at varying levels of abstraction that can be curtailed to users' preferences, with tunable fidelity spanning from formal guarantees to general tendencies of behavior. We introduce Fanoos[1], an algorithm and framework for querying broad classes of learned models.

2 The Fanoos Approach

On a high-level, illustrated in the query process in Fig. 2, Fanoos is an interactive system that allows users to pose a variety of questions grounded in a domain specification (e.g., what environmental conditions cause a robot to swerve left), receive replies from the system, and, as an inner-loop steering the heuristic search and analysis of the learned system, request that explanations be made more or less abstract. An illustration of the process and component interactions can be found in Appendix A, with a fuller example of interaction located in Appendix B.

Crucially, Fanoos provides explanations of high fidelity (being a decompositional approach; see Section 3) while considering whether the explanation should be formally sound or probabilistically reasonable (which removes the "noise" incurred by measure-zero sets that can plague formal descriptions). To this end, we combine formal verification techniques, interactive systems, and heuristic search over knowledge domains in response to user questions and requests.

2.1 Knowledge Domains and User Questions

In the following discussion, let L be the learned system under analysis (which we will assume is piece-wise continuous), q be the question posed by the user, S_I be the (bounded) input-space to L, and S_o be the output space to L, S_{IO} be the joint of the input and output space, and r be the response given by the system. In order to formulate question q and response r, a library listing basic domain information D is provided to Fanoos; D lists what S_I and S_O are and provides a set of predicates, P, expressed over the domain symbols in $S_{IO} = S_I \cup S_O$, i.e., for all $p \in P$, $\mathit{freeVars}(p) \subseteq \mathit{varNames}(S_{IO})$.

```
(Fanoos) when_do_you_usually and(outputtorque_low ,
    ↪ statevalueestimate_high )?
```

Listing 1.1. Question to illuminate input space S_I

[1] "Fanoos" means lantern in Farsi. Our approach shines a light on black-box AI.

Table 1. Description of questions that Fanoos can respond to

Type q_t	Description	Question content q_c			Example
		accepts	illum.	restrictions	
When Do You	Tell all sets (formal consideration of all cases) in the input space S_I that have the potential to cause q_c	Subset s of S_O s.t. there exists a member of s that causes q_c to be true. Found with SAT-solver.	S_I	Cannot contain variables from S_O.	when_do_you $\underbrace{\text{move\_at\_high\_speed?}}_{\text{Predicate } p \in D}$
What Do You Do When	Tell all possible learner responses in the collection of input states that q_c accepts	Subset s of S_I s.t. there exists a member of s that causes q_c to be true. Found with SAT-solver.	S_O	Cannot contain variables from S_I.	what_do_you_do_when *and*(close_to_target_orientation, close_to_target_position)?
What are the Circumstances in Which	Tell information about what input-output pairs occur in the subset of input-outputs accepted by q_c	Subset s of S_{IO} s.t. there exists a member of s that causes q_c to be true. Found with SAT-solver.	S_{IO}	None	what_are_the_circumstances_in_which *and*(close_to_target_position, steer_to_right) *or* move_at_low_speed?
. . . Usually	Statistical tendency. Avoids measure-zero sets that are unlikely seen in practice.	q_c was found to be true at least once via statistical sampling.			when_do_you_usually move_at_low_speed *or* steer_to_left? what_do_you_usually_do_when moving_toward_target_position? what_are_the_usual_circumstances_in_which *and*(close_to_target_position, steer_close_to_center)?

For queries that formally guarantee behavior (see the first three rows in Table 1), we require that the relevant predicates in P be able to expose their internals as first-order formulas; this enables us to guarantee they are satisfied over all members of a given set[2] via typical SAT-solvers (such as Z3 [17]). The other query types require only being able to evaluate question q on a variable assignment provided. The members of P can be generated in a variety of ways, e.g., by forming most predicates through procedural generation and then using a few hand-tailored predicates to capture particular cases [3]. Further, since the semantics of the predicates are grounded, they have the potential to be generated from demonstration.

[2] The box abstractions we introduce in a moment to be more precise.

[3] For example, operational definitions of “high”, “low”, etc., might be derived from sample data by setting thresholds on quantile values—e.g., 90% or higher might be considered “high”.

2.2 Reachability Analysis of L

Having established what knowledge the system is given, we proceed to explain our process. First, users select a question type q_t and the content of the question q_c to query the system. That is, $q = (q_t, q_c)$, where q_t is a member of the first column of Table 1 and q_c is a sentence in disjunctive normal form over a subset of P that obeys the restrictions listed in Table 1. To ease discussion, we will refer to variables and sets of variable-assignments that p accepts (AC) and those that p illuminates (IL), with the intuition being that the user wants to know what configuration of illuminated variables result in the variable configurations accepted by q_c; see Table 1 for example queries. To provide a convenient user experience, auto-completion for question-asking is available, limiting completions to those that obey restrictions imposed by Table 1.

With question q provided, we analyze the learned system L to find subsets in the inputs S_I and outputs S_o that agree with configuration q_c and may over-approximate the behavior of L. Specifically, we use CEGAR [14,13] with boxes (hyper-cubes) as abstractions and a random choice between a bisection and trisection along the longest axis as the refinement process to find the collect of box tuples, B, specified below:

$$
\begin{aligned}
B = \{(B_I^{(i)}, B_O^{(i)}) \in \text{Boxes}(S_I) \times \text{Boxes}(S_O) \mid \\
\Bigg(\Big(AC_q(B_I^{(i)}) \wedge IL_q(B_O^{(i)})\Big) \vee \quad \Big(AC_q(B_O^{(i)}) \wedge IL_q(B_I^{(i)})\Big) \vee \\
\Big(AC_q(B_{IO}^{(i)}) \wedge IL_q(B_{IO}^{(i)})\Big) \Bigg) \wedge B_O^{(i)} \supseteq L(B_I^{(i)})\}
\end{aligned}
$$

where $\text{Boxes}(X)$ is the set of boxes over space X. See Fig. 1 for an example drawn from analysis conducted on the model in Section 4.1. For feed-forward neural nets with non-decreasing activation functions, B may be found by covering the input space, propagating boxes through the network, testing membership to B of the resulting input- and output-boxes, and refining the input mesh as needed over input-boxes that produce output-boxes overlapping with B. The exact size of the boxes found by CEGAR are determined by a series of hyper-parameters, such as the maximum number of refinement iterations or the minimal size abstractions one is willing to consider; for details on such hyper-parameters of CEGAR and other bounded-model checking approaches the interested reader may refer to [14,13,9].

2.3 Generating Descriptions

Having generated B, we produce an initial response, r_0, to the user's query in three steps as follows: (1) for each member of B, we extract the box tuple members that were illuminated by q (in the case where S_{IO} is illuminated, we produce a joint box over both tuple members), forming a set of joint boxes, B'; (2) next, we heuristically search over P for members that describe B' and compute a set of predicates covering all boxes; (3) finally, we format the box

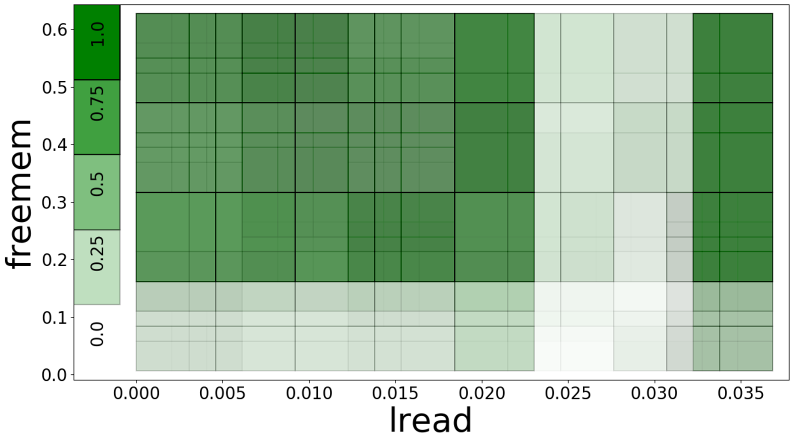


Fig. 1. Example of reachability results. Shown is a 2-D projection of 5-D input-boxes which were selected for retention since their corresponding output-boxes satisfied the user query. Darker areas had boxes with greater volume along the axes not shown; the volumes were normalized in respect to the input-space bounding-box volume over the non-visible axes. A scale is along the y-axis.

covering for user presentation. A sample result answer is shown in listing 1.2, and details on steps (2) and (3) how to produce it follow below.

```
(0.45789160, 0.61440409, 'x Near Normal Levels')
(0.31030792, 0.51991449, 'pole2Angle_rateOfChange Near Normal Levels')
(0.12008841, 0.37943400, 'pole1Angle_rateOfChange High')
(0.06128723, 0.22426058, 'pole2Angle Low')
```

Listing 1.2. Initial answer to question in listing 1.1

Producing a Covering of B' Our search over P for members covering B' is largely based around the greedy construction of a set covering using a carefully designed candidate score.

For each member $b \in B'$ we want to find a set of candidate predicates capable of describing the box and for which we would like to form a larger covering. We find a subset of $P_b \subseteq P$ that is consistent with b in that each member of P_b passes the checks called-for by q_t when evaluated on b (see the Description column of Table 1). This process is made fast by a feasibility check of each member of P on a vector randomly sampled from b, prior to the expensive check for inclusion in P_b. Having P_b, we filter the candidate set further for those members of P_b that appear most specific to b; notice that in our setting, where predicates of varying abstraction level co-mingle in P, it would be of no surprise that P_b contains many members that only loosely fit b. This subset of P_b, P'_b, is formed by sampling outside of b at increasing radii (in the ℓ_∞ sense), collecting those members of P_b that fail to hold true at the earliest radius (see the pseudo-code in Appendix D for further details). Importantly, looking ahead to forming a full covering of B, if none of the predicates fail prior to exhausting[4] this sampling, we report P'_b as empty, allowing us to handle b downstream; this avoids having "difficult" boxes force us to report weak predicates, that would "wash out" more granular details. Further notice that if we want a subset of P_b that was less specific to b than P'_b, we simply perform the CEGAR-analysis so to produce larger boxes—in other words, we try to be specific at this phase under the assumption that the granularity we wanted to describe has been determined earlier.

We next leverage the P'_b sets to construct a covering of B', proceeding in an iterative greedy fashion. Specifically, if C_i is the covering established at iteration i, we increment to C_{i+1} as follows:

$$\begin{aligned}
&C_{i+1} = (C_i \cup \{p_{i+1}\}) \setminus \\
&\qquad\qquad \{p' \in C_i \mid \{b \in B' \mid p' \in P'_b\} \subseteq \{b \in B' \mid p_{i+1} \in P'_b\}\} \text{ with} \\
&p_{i+1} = \text{argmax}_{p \in P \setminus C_i} \text{CoverScore}(p, C_i) \text{ where} \\
&\mathsf{CoverScore}(p, C_i) = \\
&\qquad \sum_{b \in B'} \mathbb{1}(|\mathsf{uncoveredVars}(b, C_i) \cap \mathsf{freeVars}(p)| > 0)\mathbb{1}(p \in P'_b)
\end{aligned}$$

and $\mathsf{uncoveredVars}$ is the set of variables in b that are not constrained by $C_i \cap P_b$; since the boxes are multivariate and our predicates typically only constrain a subset of the variables, we select predicates based on how many boxes would have open variables covered by them. Let C_F be the final covering produced by this process. Notice that C_F is not necessarily an approximately minimal covering of B with respect to members of P—by forcing $p \in P'_b$ when calculating $\mathsf{CoverScore}$, we enforce additional specificity criteria that the covering should adhere to.

After forming C_F, any boxes that fail to be covered even partially (for example, because P_b or P'_b happen to be empty) are reported with a box-range predicate: atomic predicates that simply list the variable range in the box. In other words, we extend C_F to a set C'_F by introducing new predicates specific to each completely uncovered box so that C'_F does cover all boxes in B'.

Cleaning and Formatting Output for User Having produced C'_F, we collect the content of the covering into a series of conjuncts and disjuncts. Let

$$d_0 = \bigcup_{b \in B'} \{c \subseteq C'_F \mid \forall p \in c. \text{ (p covers b)}\} \ .$$

[4] The operational meaning of "exhausting", as well as the radii sampled, are all parameters stored in the state.

The information needed to compute d_0 is easily gathered from bookkeeping while computing C'_F. Ultimately, the members of d_0 are conjunctions of predicates [5], with their membership to the set being a disjunction. Prior to actually converting d_0 to disjunctive normal form, however, we do the trivial filter of removing any $c \in d_0$ such that there is $c' \in d_0$ where $c' \subsetneq c$, since the set over which $and(c')$ holds is a superset of where $and(c)$ holds. While there are a variety of methods to perform this filter, in practice d_0 is sufficiently small at this stage to allow full member-to-member comparisons. Call d_0 post-filtering d'_0.

Finally, r_0 is constructed by listing each c that exists in d'_0 sorted by two relevance scores: first, the proportion of the volume in B' uniquely covered by c, and second by the proportion of total volume c covers in B'. These sorting-scores can be thought of similarly to recall measures. Specificity is more difficult to tackle, since it would require determining the volume covered by each predicate (which may be an arbitrary first-order formula) across the box bounding the universe, not just the hyper-cubes at hand; this can be approximated for each predicate using set-inversion, but requires non-trivial additional computation for each condition.

2.4 User Feedback and Revaluation

Based on the initial response r_0, users can request a more abstract or less abstract explanation. We view this alternate explanation generation as another heuristic search, where the system searches over a series of states to find those that are deemed acceptable by the user. The states primarily include algorithm hyper-parameters, the history of user interaction including the provided explanations, the question to be answered, and the set B. Abstraction and refinement operators take a current state and produce a new one, often by adjusting the system hyper-parameters and recomputing B. This state-operator model of user response allows for rich styles of interaction with the user, beyond and alongside of the three-valued responses of acceptance, increase, or decrease of the abstraction level show in listing 1.3.

```
(0.11771033, 0.12043966, 'And(pole1Angle Near Normal Levels,
    ↪ pole1Angle_rateOfChange Near Normal Levels, pole2Angle
    ↪ High, pole2Angle_rateOfChange Low, vx Low)')
(0.06948142, 0.07269412, 'And(pole1Angle High,
    ↪ pole1Angle_rateOfChange Near Normal Levels,
    ↪ pole2Angle_rateOfChange High, vx Low, x Near Normal
    ↪ Levels)')
(0.04513659, 0.06282974, 'And(endOfPole2_x Near Normal
    ↪ Levels, pole1Angle Low, pole1Angle_rateOfChange High,
    ↪ pole2Angle High, pole2Angle_rateOfChange Near Normal
    ↪ Levels, x High)')q
```

[5] From here-on, when we refer to a conjuct, we assume it is not in reference to the degenerate 1-ary or 0-ary cases.

```
type letter followed by enter key: b − break and ask a
    ↪ different question,
```

Listing 1.3. Response to "less abstract" than listing 1.2

For instance, a history-travel operator allows the state (and thus r) to return to an earlier point in the interaction process, if the user feels that response was more informative; from there, the user may investigate an alternate path of abstractions. Other operators allow for refinements of specified parts of explanations as opposed to the entire reply; the simplest form of this is by regenerating the explanation without using a predicate that the user specified be ignored. To illustrate details, we describe abstraction operators for increasing or decreasing the abstraction level in the appendix. As a fundamental underlying concept, these operators use a notion of abstractness, as discussed below.

2.5 Capturing the Concept of Abstractness

The exact bounds that delimit abstractness and what makes one thing more or less abstract than another in the lay-sense are often difficult to capture. We consider abstractness a diverse set of relations that subsume the part-of-whole relation, and thus also generally includes the subset relation. For our purposes, defining this notion is not necessary, since we simply wish to utilize the fact of its existence. We understand abstractness to be a semantic concept that shows itself by producing a partial ordering over semantic states (their "abstractness" level) which is in turn reflected in the lower-order semantics of the input-output boxes, and ultimately is reflected in our syntax via explanations of different granularity. Discussions of formalisms most relavent to computer science can be found in [15,65,64] [6] and an excellent discussion of the philosophical underpinnings and extensions can be found in [24].

In this work, the primary method of producing explanations at desired levels of abstraction is entirely implicit—that is, without explicitly tracking what boxes or predicates are considered more or less abstract. This leverages the notion of abstractness inherent in the semantics and refinements of CEGAR in operators that adjust abstraction by adjusting the input-space mesh size, which extends to the verbalization process through the computed covering of boxes.

On the opposite end of the spectrum is explicit expert tuning of abstraction orderings to be used in the system. Fanoos can easily be adapted to leverage expert-labels (e.g., taxonomies as in [63], or simply type/grouping-labels without explicit hierarchical information) to preference subsets of predicates conditionally on user-responses, but for the sake of this paper, we reserve agreement with expert-labels as an independent metric of performance in our evaluation, prohibiting the free use of such knowledge in the algorithm. Further, by forgoing direct supervision, we demonstrate that the concept of abstractness is recoverable from the semantics and structure of the problem itself.

[6] [15] features abstraction in verification, [65] features abstraction at play in interpretating programs, and [64] is an excellent example of interfaces providng a notion of abstractness in network communications.

3 Related Work and Discussion

Many methods are closely related to XAI, stemming from a diverse body of literature and various application domains, e.g., [16,5,3,32,62,57,36,73,8]. Various taxonomies of explanation families have been proposed [45,6,39,43,5,1,27,10,25,58,69,12,59,53,29,11], with popular divisions being (1) between explanations that leverage internal mechanics of systems to generate descriptions (decompositional approaches) versus those that exclusively leverage input-output relations (pedagogical) [7], (2) the medium that comprises the explanation (such as with most-predictive-features [57], summaries of internal states via finite-state-machines [41], natural language descriptions [32,40] or even visual representations [35,40]), (3) theoretical criteria for a good explanation (see, for instance, [46]), and (4) specificity and fidelity of explanation. Overall, most of these approaches advocate for producing human-consumable information—whether it be in natural language, logic, or visual plots—conveying the behavior of the learned system in situations of interest.

Rule-based systems such as expert systems, and work in the (high-level) planning community have a long history of producing explanations in various forms; notably, hierarchical planning [32,47] naturally lends itself to explanations of multiple abstraction levels. All these methods, however, canonically work on the symbolic level, making them inapplicable to most modern ML methods. High fidelity, comprehensible rules describing data points can also be discovered with weakly-consistent inductive logic programming [49] or association rule learning [34,3] typical in data-mining. However, these approaches are typically pedagogical, not designed to leverage access to the internals of the system, and do not offer a variety of descriptions abstractions or strengths. While some extentions of association rule learning (e.g., [63,31,30]) do consider descriptions at various abstraction levels, they only understand abstractness syntatically, requiring complete taxonomies be provided explicitly and a priori ; further, such approaches do not attempt to describe the full datasets they derive from, but only some aspects of them - while support and confidence thresholds may be set sufficaintly low to ensure each transaction is described by at least one rule, the result would be a deluge of highly redundant, low-precision rules lacking most practical value (this may be considered the most extreme case of the "rare itemset problem" as dicussed in [44]). Our approach, by contrast leverages semantic information, attempts to efficiently describe all relevant data instances, and produces descriptions that are necessarly reflective of the mechanism under study. Decision support systems [52,72,20,21,19] typically allow users to interactively investigate data, with operations such as drill-ups in OLAP (OnLine Analytical Processing) cubes analogous to a simple form of abstraction in that setting. The typical notions of analysis, however, largely operate by truncating portions of data distributions and running analytics packages on selected subregions at user's requests, failing to leverage access to the data-generation mechanism when

[7] We have also found this to be referred to as "introspective" explanations versus "rationalizations", such as in [40]

present, and failing to provide explicit abstractions or explicit guarantees about the material it presents.

More closely related to our work are approaches to formally analyze neural networks to extract rules, ensure safety, or determine decision stability, which we discuss in more detail below. Techniques related to our inner-loop reachability analysis have been used for stability or reachability analysis in systems that are otherwise hard to analyze analytically. Reachability analysis for FFNNs based on abstract interpretation domains, interval arithmetic, or set inversion has been used in rule extraction and neural net stability analysis [5,18,66,74] and continues to be relevant, e.g., for verification of multi-layer perceptrons [56], estimating the reachable states of closed-loop systems with multi-layer perceptrons in the loop [78], estimating the domain of validity of neural networks [2], and analyzing security of neural networks [71]. While these works provide methods to extract descriptions that faithfully reflect behavior of the network, they do not generally ensure descriptions are comprehensible by end-users, do not explore the practice of strengthening descriptions by ignoring the effects of measure-zero sets, and do not consider varying description abstraction.

The high-level components of our approach can be compared to [33], where hand-tunable rule-based methods with natural language interfaces encapsulate a module responsible for extracting information about the ML system, with explanation generation in part relying on minimal set-covering methods to find predicates capturing the model states. Extending this approach to generate more varying-resolution descriptions, however, does not seem like a trivial endeavor, since (1) it is not clear that the system can appropriately handle predicates that are not logically independent, and expecting experts to explicitly know and encode all possible dependencies can be unrealistic, (2) the system described does not have a method to vary the type of explanation provided for a given query when its initial response is unsatisfactory, and (3) the method produces explanations by first learning simpler models via MDPs. Learning simpler models by sampling behavior of more sophisticated models is an often-utilized, widely applicable method to bootstrap human understanding (e.g. [10,41,28]), but it comes at the cost of failing to leverage substantial information from the internals of the targeted learned system. Crucially, such a technique cannot guarantee the fidelity of their explanations in respect to the learned system being explained, in contrast to our approach.

In [54], the authors develop vocabularies and circumstance-specific human models to determine the parameters of the desired levels of abstraction, specificity and location in robot-provided explanations about the robot's specific, previous experiences in terms of trajectories in a specific environment, as opposed to the more generally applicable conditional explanations about the internals of the learned component generated by Fanoos. The particular notions of abstraction and granularity from multiple, distinct, unmixable vocabularies of [54] evaluate explanations in the context of their specific application and are not immediately applicable nor easily transferable to other domains. Fanoos, by contrast, does not require separate vocabularies and enables descriptions to include multiple abstraction levels (for example, mixing them as in the sentence “House X and a 6m large patch on house Y both need to be painted”).

Closest in spirit to our work are planning-related explanations [62][8], providing multiple levels of abstraction with a user-in-the-loop refinement process, but with a focus on markedly different search-spaces, models of human interaction, algorithms for description generation and extraction, and experiments. Further, we attempt to tackle the difficult problem of extracting high-level symbolic knowledge from systems where such concepts are not natively embedded, in contrast to [62], who consider purely symbolic constructs.

In summary, current approaches focus on single aspects of explanations, fixed levels of abstraction, and inflexible guarantees about the explanations given. We argue that an interleaving between automated formal techniques, search heuristics, and user interaction is necessary to achieve the desired flexibility in explanations and the desired adjustable level of fidelity.

4 Experiments and Results

In this section we discuss empirical demonstrations of Fanoos’s ability to produce and adjust descriptions across two different domains. The code implementing our method, the models analyzed, the database of raw-results, and the analysis code used to generate the results presented will be released in the near future.

4.1 Systems Analyzed

We analyze learned-systems from robotics control and more traditional ML predictions to demonstrate the applicability to diverse domains. Information on the predicates available for each domain can be found in Table 2.

Table 2. Summary statistics of predicates in each domain

	CPU	IDP
Input space predicates	33	62
Output space predicates	19	12
Joint input-output space predicates	8	4
Total with MA (more abstract) label	20	15
Total with LA (less abstract) label	40	63

Inverted Double Pendulum (IDP) The control policy for an inverted double-pendulum—similar to the basic inverted single pendulum example in control—is

[8] We note that [62] was published after the core of our approach was developed; both of our thinkings developed independantly.

tasked to keep a pole steady and upright; the pole consists of two segments, an under-actuated one attached to the end of the first, actuated one, where both are rotationally free in the same plane. This substantially complicates the control-problem, since multi-pendulum systems are known to exhibit chaotic behavior [38,42]. The trained policy was taken from reinforcement learning literature[9]. The seven-dimensional observation space is [x, vx, pole2_endpoint, pole1angle, pole1angle_rateOfChange, pole2angle, pole2angle_rateOfChange], the bounding box for which can be found in Table 4 located in Appendix C. The output is a torque in $[-1, 1]Nm$ and a state-value, which is not a priori bounded. Internal to the analyzed model is a transformation to convert the observations we provide to the form expected by the networks—chiefly, the angles are converted to sines and cosines and observations are standardized in respect to the mean and standard deviation of the model's training runs. The values chosen for the input-space bounding box were inspired by the 5% and 95% quantile values over a test-run of the model in the rl-zoo framework. We expanded the input-box beyond this range to allow for the examination of rare inputs and observations the model was not necessarily trained on (e.g., while the train and test environments exit whenever the end of the second segment was below a certain height, in real applications, a user may want to know how the system attempts to recover in such an unseen situation). Whether or not the analysis stays in trained regions depends on the content of the user's question, which may either include or exclude these previously unseen regions.

CPU Usage (CPU) We also analyze a more traditional ML algorithm for a non-control task — a polynomial kernel regression for modeling. Specifically, we use a three-degree fully polynomial basis over a 5-dimensional input space[10] to linearly regress-out a three-dimensional vector. We trained our model using the publicly available data from [67][11]. The observations are

$$[lread, scall, sread, freemem, freeswap]$$

and the response variables we predict are

$$[lwrite, swrite, usr]\ .$$

We opted to analyze an algorithm with a degree-3 polynomial feature-set after normalizing the data in respect to the minimum and maximum of the training set since this achieved the highest performance—over 90% accuracy—on a 90%-10% train-test split of the data compared to similar models with 1,2, or 4 degree kernels[12]. While the weights of the kernel may be interpreted in some sense (such

[9] `https://github.com/araffin/rl-baselines-zoo`, trained using PPO2 [61] which, as an actor-critic method, uses one network to produce the action, and one to estimate state-action values.

[10] The input space includes cross-terms and the zero-degree element—e.g., x^2y and 1 are members.

[11] Dataset available at `https://www.openml.org/api/v1/json/data/562`

[12] Note that while we did do due-diligence in producing a performant and soundly-trained model, the primary point is to produce a model worthy of analysis.

as indicating which individual feature is, by itself, most influential), how the model behaves over the original input space is far from clear from these weights due to the joint correlation between the features and non-linear transformations of the input values. For analysis convenience, we transform the input space to be normalized in the same fashion as the observations the model was trained and evaluated on (alternatively, we could have kept the original observation space and put the normalization as part of the model-pipeline). The bounds of our input-space bounding-box were determined from the 5% and 95% quantiles for each input-variable over the full, normalized dataset; the exact values can be found in Table 5 located in Appendix C.

4.2 Experiment Design

We tested Fanoos on the listed domains using synthetically generated interactions, with the goal of determining whether our approach properly changes the description abstractness in response to the user request. The domain and question type were randomly chosen among the options listed. The questions themselves were randomly generated to have up to four disjuncts, each with conjuncts of length no more than four; conjuncts were ensured to be distinct, and only predicates respecting the constraints of the question-type were used. Interaction with Fanoos post-question-asking was randomly selected from four alternatives (here, MA means "more abstract" and LA means "less abstract"):

- Initial refinement of 0.25 ; make LA; make MA; exit
- Initial refinement of 0.125 ; make MA; make LA; exit
- Initial refinement of 0.20 ; make LA; make MA; exit
- Initial refinement of 0.10 ; make MA; make LA; exit

For the results presented here, over 130 of these interactions were held, resulting in several hundred question-answer-descriptions.

4.3 Metrics

We evaluated the abstractness of *each* response Fanoos provided using several metrics across the following categories: reachability analysis, structural description, and expert labeling.

Reachability Analysis We compare the reachability analysis results when producing descriptions of different abstraction levels, which call for different levels of refinement. Specifically, we record statistics about the input-boxes generated during the CEGAR-like analysis, normalized to the input-space bounding box so that each axis is in $[0, 1]$ to yield comparable results across domains. The metrics give a rough sense of the abstractness notion implicit in the size of boxes and how they relate to descriptions:

- Volume of the box (product of its side lengths).

- Sum of the box side lengths. Unlike the box volume, this measure is at least as much as the maximum side length.
- Number of boxes used to form the description.

The volume and summed-side-lengths are distributions, reported in terms of the minimum, maximum, median, and sum of the values.

Description Structure Fanoos responds to the user with a weighed description in disjunctive normal form. This structure is summarized as follows to give a rough sense of how specific each description is by itself:

- Number of disjuncts, including atomic predicates
- Number of conjuncts, excluding atomic predicates[13]
- Number of named predicates: atomic user-defined predicates that occur anywhere in the description, i.e., excluding box-range predicates of conjuncts of atomic predicates.
- Number of box-range predicates that occur anywhere (i.e., in conjuncts as well as stand-alone).

The Jaccard score and overlap coefficients below are used to measure similarity in the verbage used in two descriptions.

- Jaccard score: general similarity between two descriptions, viewing the set of atomic predicates used in each description as a bag-of-words.
- Overlap coefficient: measures whether one description is simply a more "verbose" variant of the other, in the sense that the set of atomic predicates of one is a subset of the other using $\frac{|S_1 \cap S_2|}{min(|S_1|, |S_2|)}$, where S_1 and S_2 are the sets of predicates used in the two descriptions.

Expert Labeling As humans, we have a priori knowledge about which atomic predicates describe more abstract notions in the world than others, and as such can evaluate the responses based on usage of more vs. less abstract verbage. It is important to note that this approach—on descriptions built from atomic predicates—yields an informative approximation rather than a true measure of abstractness for the following reasons: it is not clear that the abstractness of a description's components translates in an obvious fashion to the abstractness of the whole (in a similar vein, we do not rule out the possibility that predicates of the same level in the partially ordered set of abstractness can be combined to descriptions of different abstractness[14]). This phenomenon becomes more pronounced in coarsely grained partitions, where nuances are hidden in the partitions. For simplicity we choose two classes, more abstract (MA) vs. less abstract (LA), in the measures below:

[13] By excluding atomic predicates, this provides some rough measure of the number of "complex" terms.

[14] For example, just because two description use verbage from the same expert-labeled category of abstractness, it does not mean the two descriptions have the same level of abstractness.

- Number of predicates accounting for multiplicity, i.e., if an atomic predicate q has label MA and occurs twice in a sentence, it contributes two to this score.
- Number of unique predicates: e.g., if an atomic predicate q has label MA and occurs twice in a sentence, it contributes one to this score.
- Prevalence: ratio of unique predicates to the total number of atomic predicates in a description. This measure is particularly useful when aggregating the results of multiple descriptions into one distribution, since the occurrence of predicates is statistically coupled with the length of descriptions; under a null hypothesis of random generation of content, one would expect longer sentences to contain more MA,LA predicates, but expect the proportion to remain constant.

Each of the above measures have obvious counter-parts for predicates with MA/LA labels. We note that prevalence will not necessarily sum to 1, since box-range predicates are atomic predicates without either label.

4.4 Results

Running the experiments described in Section 4.2, we collected a series of states and the summary statistics on them described in Section 4.3. Since we are chiefly interested in evaluating whether a description changes to reflect the abstraction requested by the user, we examine the relative change in response to user interaction. Specifically, for pre-interaction state S_t and post-interaction state S_{t+1}, we collect metrics $m(S_{t+1}) - m(S_t)$ for each domain-response combination. This same process is used for the Jaccard score and overlap coefficients, except the values in question are computed as $m(S_{t+1}, S_t)$. Our results are summarized in Table 3 on the medians of these distributions.

As can be seen, the reachability and structural metrics follow the desired trends: when the user requests greater abstraction (MA), the boxes become larger, and the sentences become structurally less complex—namely, they become shorter (fewer disjuncts), have disjuncts that are less complicated (fewer explicit conjuncts, hence more atomic predicates), use fewer unique terms overall (reduction in named predicates) and resort less often to referring to the exact values of a box (reduction in box-range predicates). Symmetric statements can be made for when requests for less abstraction (LA) are issued. From the overlap and Jaccard scores, we can see that the changes in response complexity are not simply due to increased verbosity—simply adding or removing phrases to the descriptions from the prior steps—but also the result of changes in the verbage used; this is appropriate since abstractness is not exclusively a function of description specificity.

Trends for the expert labels are similar, though more subtle to interpret. We see that use of LA-related terms follows the trend of user requests with respect to multiplicative- and uniqueness-counts (increases for LA-requests, decreases for MA-requests), while being less clear with respect to prevalence (uniform 0 scores). For use of MA terms, we see that the prevalence is correlated with

Table 3. Median relative change in description before and after Fanoos adjusts the abstraction in the requested direction

			CPU	CPU	IDP	IDP
		Request	LA	MA	LA	MA
Reachability		Boxes	8417.5	-8678.0	2.0	-16.0
	Sum side lengths	Max	-1.125	1.125	-1.625	1.625
		Median	-1.187	1.188	-2.451	2.438
		Min	-0.979	0.986	-2.556	2.556
		Sum	21668.865	-22131.937	582.549	-553.007
	Volume	Max	-0.015	0.015	-0.004	0.004
		Median	-0.003	0.003	-0.004	0.004
		Min	-0.001	0.001	-0.003	0.003
		Sum	-0.03	0.03	-0.168	0.166
Structural		Jaccard	0.106	0.211	0.056	0.056
		Overlap coeff.	0.5	0.714	0.25	0.25
		Conjuncts	1.0	-2.0	0.5	-2.5
		Disjuncts	7.0	-7.5	2.0	-2.5
		Named preds.	1.0	-1.0	1.0	-4.5
		Box-Range preds.	2.0	-2.0	1.5	-1.5
Expert	MA terms	Multiplicative	3.0	-3.0	24.0	-20.0
		Uniqueness	0.0	0.0	1.0	-1.5
		Prevalence	-0.018	0.014	-0.75	0.771
	LA terms	Multiplicative	20.0	-21.5	68.5	-86.0
		Uniquness	2.0	-2.0	12.0	-14.0
		Prevalence	0.0	0.0	0.0	0.0

user requests in the expected fashion (decrease on LA requests, increase on MA requests). Further, we see that this correlation is mirrored for the MA counts when taken relative to the same measures for LA terms. Specifically, when a user requests greater abstraction (MA), the counts for LA terms decrease far more than those of MA terms, and the symmetric situation occurs for requests of lower abstraction (LA), as expected. While they depict encouraging trends, we take these expert-label measures with caution, due to the fragility of reasoning about the complete description's abstractness based on its constituents (recall that the abstractness of a description is not necessarily directly linked to the abstractness of its components). Nevertheless, these results—labelings coupled with the structural trends—lend solid support to the notion that Fanoos can recover substantial elements of an expert's notions about abstractness by leveraging the grounded semantics of the predicates.

5 Conclusions And Future Work

Fanoos is an explanation framework for ML components mixing technologies ranging from heuristic search to classic verification techniques. We have demonstrated that our approach is capable of producing and navigating explanations at multiple, adjustable levels of granularity and strength. We will continue to explore this direction's potential, and hope that the community finds inspiration in both the methodology and philosophical underpinnings presented here.

As future work, we are exploring the use of active learning leveraging user interactions to select from a collection of operators, with particular interest in bootstrapping the learning process using operationally defined oracles to approximate users. In addition to this, we plan to explore more advanced data-driven predicate generation to accelerate construction of knowledge bases; an early candidate is learning generalized Hough transforms [7] given their representational flexibility, amenability to human review, and intuitiveness of the extrapolations that may be necessary. Finally, this style of work lends itself to engineering improvements on the reachability computations curtailed to ML systems.

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: A survey on explainable artificial intelligence (xai). IEEE Access **6**, 52138–52160 (2018)
2. Adam, S.P., Karras, D.A., Magoulas, G.D., Vrahatis, M.N.: Reliable estimation of a neural network's domain of validity through interval analysis based inversion. In: 2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015. pp. 1–8 (2015). https://doi.org/10.1109/IJCNN.2015.7280794, `https://doi.org/10.1109/IJCNN.2015.7280794`
3. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Acm sigmod record. vol. 22, pp. 207–216. ACM (1993)
4. Aha, D.W., Darrell, T., Pazzani, M., Reid, D., Sammut, C., Stone, P.: Ijcai 2017 workshop on explainable artificial intelligence (xai). Melbourne, Australia, August (2017)
5. Andrews, R., Diederich, J., Tickle, A.: Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-Based Systems **6**, 373–389 (12 1995). https://doi.org/10.1016/0950-7051(96)81920-4
6. Anjomshoae, S., Najjar, A., Calvaresi, D., Främling, K.: Explainable agents and robots: Results from a systematic literature review. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems (2019)
7. Ballard, D.: Generalising the hough transform to detect arbitary patterns. Pattern Recognition **13** (1981)
8. Benz, A., Jäger, G., Van Rooij, R., Van Rooij, R.: Game theory and pragmatics. Springer (2005)
9. Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y., et al.: Bounded model checking. Advances in computers **58**(11), 117–148 (2003)
10. Biran, O., Cotton, C.: Explanation and justification in machine learning: A survey. In: IJCAI-17 workshop on explainable AI (XAI). vol. 8, p. 1 (2017)
11. Chakraborti, T., Kulkarni, A., Sreedharan, S., Smith, D.E., Kambhampati, S.: Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 29, pp. 86–96 (2019)

12. Chuang, J., Ramage, D., Manning, C., Heer, J.: Interpretation and trust: Designing model-driven visualizations for text analysis. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 443–452. ACM (2012)
13. Clarke, E., Fehnker, A., Han, Z., Krogh, B., Stursberg, O., Theobald, M.: Verification of hybrid systems based on counterexample-guided abstraction refinement. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 192–207. Springer (2003)
14. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: International Conference on Computer Aided Verification. pp. 154–169. Springer (2000)
15. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages. pp. 238–252 (1977)
16. David, Q.: Design issues in adaptive control. IEEE Transactions on Automatic Control **33**(1) (1988)
17. De Moura, L., Bjørner, N.: Z3: An efficient smt solver. In: Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. TACAS'08/ETAPS'08, Springer-Verlag, Berlin, Heidelberg (2008), http://dl.acm.org/citation.cfm?id=1792734.1792766
18. Driescher, A., Korn, U.: Checking stability of neural narx models: An interval approach. IFAC Proceedings Volumes **30**(6), 1005–1010 (1997)
19. Eom, H.B., Lee, S.M.: A survey of decision support system applications (1971–april 1988). Interfaces **20**(3), 65–79 (1990)
20. Eom, S., Kim, E.: A survey of decision support system applications (1995–2001). Journal of the Operational Research Society **57**(11), 1264–1278 (2006)
21. Eom, S.B., Lee, S.M., Kim, E., Somarajan, C.: A survey of decision support system applications (1988–1994). Journal of the Operational Research Society **49**(2), 109–120 (1998)
22. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) (2016), http://data.europa.eu/eli/reg/2016/679/oj
23. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., et al.: Advances in knowledge discovery and data mining, vol. 21. AAAI press Menlo Park (1996)
24. Floridi, L.: The method of levels of abstraction. Minds and machines **18**(3), 303–329 (2008)
25. Friedrich, G., Zanker, M.: A taxonomy for generating explanations in recommender systems. AI Magazine **32**(3), 90–98 (2011)
26. Garcıa, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research **16**(1), 1437–1480 (2015)
27. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) **51**(5), 93 (2019)
28. Gunning, D.: Explainable artificial intelligence (xai). Defense Advanced Research Projects Agency (DARPA), nd Web **2** (2017), https://www.darpa.mil/attachments/XAIProgramUpdate.pdf
29. Hailesilassie, T.: Rule extraction algorithm for deep neural networks: A review. arXiv preprint arXiv:1610.05267 (2016)

30. Han, J., Fu, Y.: Discovery of multiple-level association rules from large databases. In: VLDB. vol. 95, pp. 420–431. Citeseer (1995)
31. Han, J., Fu, Y.: Mining multiple-level association rules in large databases. IEEE Transactions on knowledge and data engineering **11**(5), 798–805 (1999)
32. Hayes, B., Scassellati, B.: Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 5469–5476. IEEE (2016)
33. Hayes, B., Shah, J.A.: Improving robot controller transparency through autonomous policy explanation. In: 2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI. pp. 303–312. IEEE (2017)
34. Hipp, J., Güntzer, U., Nakhaeizadeh, G.: Algorithms for association rule mining-a general survey and comparison. SIGKDD explorations **2**(1), 58–64 (2000)
35. Huang, S.H., Held, D., Abbeel, P., Dragan, A.D.: Enabling robots to communicate their objectives. Autonomous Robots **43**(2), 309–326 (Feb 2019). https://doi.org/10.1007/s10514-018-9771-0, https://doi.org/10.1007/s10514-018-9771-0
36. Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks (2017). https://doi.org/10.1007/978-3-319-63387-9_5, https://doi.org/10.1007/978-3-319-63387-9_5
37. Kearfott, R.B.: Interval computations: Introduction, uses, and resources. Euromath Bulletin **2**(1), 95–112 (1996)
38. Kellert, S.H.: In the wake of chaos: Unpredictable order in dynamical systems. University of Chicago press (1993)
39. Kim, J., Rohrbach, A., Darrell, T., Canny, J., Akata, Z.: Textual explanations for self-driving vehicles. In: Proceedings of the European conference on computer vision (ECCV). pp. 563–578 (2018)
40. Kim, J., Rohrbach, A., Darrell, T., Canny, J.F., Akata, Z.: Textual explanations for self-driving vehicles (2018). https://doi.org/10.1007/978-3-030-01216-8_35, https://doi.org/10.1007/978-3-030-01216-8_35
41. Koul, A., Fern, A., Greydanus, S.: Learning finite state representations of recurrent policy networks (2019), https://openreview.net/forum?id=S1gOpsCctm
42. Levien, R., Tan, S.: Double pendulum: An experiment in chaos. American Journal of Physics **61**(11), 1038–1044 (1993)
43. Lipton, Z.C.: The mythos of model interpretability. arXiv preprint arXiv:1606.03490 (2016)
44. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 337–341 (1999)
45. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence (2018)
46. Miller, T., Howe, P., Sonenberg, L.: Explainable ai: Beware of inmates running the asylum or: How i learnt to stop worrying and love the social and behavioural sciences. arXiv preprint arXiv:1712.00547 (2017)
47. Mohseni-Kabir, A., Rich, C., Chernova, S., Sidner, C.L., Miller, D.: Interactive hierarchical task learning from a single demonstration. In: Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction. pp. 205–212. ACM (2015)
48. Moore, R.E.: Interval analysis, vol. 4. Prentice-Hall Englewood Cliffs, NJ (1966)
49. Muggleton, S.: Inductive logic programming: issues, results and the challenge of learning language in logic. Artificial Intelligence **114**(1-2), 283–296 (1999)

50. Neema, S.: Assured autonomy. DARPA Research Program.
51. Neema, S.: Assured autonomy (2017), https://www.darpa.mil/attachments/AssuredAutonomyProposersDay_Program%20Brief.pdf
52. Palaniappan, S., Ling, C.: Clinical decision support using olap with data mining. International Journal of Computer Science and Network Security **8**(9), 290–296 (2008)
53. Papadimitriou, A., Symeonidis, P., Manolopoulos, Y.: A generalized taxonomy of explanations styles for traditional and social recommender systems. Data Mining and Knowledge Discovery **24**(3), 555–583 (2012)
54. Perera, V., Selveraj, S.P., Rosenthal, S., Veloso, M.: Dynamic generation and refinement of robot verbalization. In: 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). pp. 212–218. IEEE (2016)
55. Piatetsky-Shapiro, G., Frawley, W.: Knowledge discovery in databases, 1991
56. Pulina, L., Tacchella, A.: An abstraction-refinement approach to verification of artificial neural networks. In: International Conference on Computer Aided Verification. pp. 243–257. Springer (2010)
57. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier (2016). https://doi.org/10.1145/2939672.2939778, https://doi.org/10.1145/2939672.2939778
58. Richardson, A., Rosenfeld, A.: A survey of interpretability and explainability in human-agent systems. In: XAI Workshop on Explainable Artificial Intelligence. pp. 137–143 (2018)
59. Roberts, M., Monteath, I., Sheh, R., Aha, D., Jampathom, P., Akins, K., Sydow, E., Shivashankar, V., Sammut, C.: What was i planning to do. In: ICAPS workshop on explainable planning. pp. 58–66 (2018)
60. Rosenthal, S., Biswas, J., Veloso, M.: An effective personal mobile robot agent through symbiotic human-robot interaction. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1. pp. 915–922. International Foundation for Autonomous Agents and Multiagent Systems (2010)
61. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
62. Sreedharan, S., Madhusoodanan, M.P., Srivastava, S., Kambhampati, S.: Plan explanation through search in an abstract model space pp. 67–75 (2018)
63. Srikant, R., Agrawal, R.: Mining generalized association rules (1995)
64. Standardization, I.: Iso/iec 7498-1: 1994 information technology–open systems interconnection–basic reference model: The basic model. International Standard ISOIEC **74981**, 59 (1996)
65. Tennent, R.D.: The denotational semantics of programming languages. Commun. ACM **19**(8), 437–453 (Aug 1976). https://doi.org/10.1145/360303.360308, https://doi.org/10.1145/360303.360308
66. Thrun, S.: Extracting rules from artificial neural networks with distributed representations. In: Advances in neural information processing systems. pp. 505–512 (1995)
67. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: Openml: Networked science in machine learning. SIGKDD Explorations **15**(2), 49–60 (2013). https://doi.org/10.1145/2641190.2641198, http://doi.acm.org/10.1145/2641190.2641198
68. Veloso, M.M., Biswas, J., Coltin, B., Rosenthal, S.: Cobots: Robust symbiotic autonomous mobile service robots. In: IJCAI. p. 4423 (2015)

69. Ventocilla, E., Helldin, T., Riveiro, M., Bae, J., Boeva, V., Falkman, G., Lavesson, N.: Towards a taxonomy for interpretable and interactive machine learning. In: XAI Workshop on Explainable Artificial Intelligence. pp. 151–157 (2018)
70. Walter, E., Jaulin, L.: Guaranteed characterization of stability domains via set inversion. IEEE Transactions on Automatic Control **39**(4), 886–889 (April 1994). https://doi.org/10.1109/9.286277
71. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: 27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018. pp. 1599–1614 (2018), https://www.usenix.org/conference/usenixsecurity18/presentation/wang-shiqi
72. Wasylewicz, A.T.M., Scheepers-Hoeks, A.M.J.W.: Clinical Decision Support Systems, pp. 153–169. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-99713-1_11, https://doi.org/10.1007/978-3-319-99713-1_11
73. Wellman, H.M., Lagattuta, K.H.: Theory of mind for learning and teaching: The nature and role of explanation. Cognitive development **19**(4), 479–497 (2004)
74. Wen, W., Callahan, J.: Neuralware engineering: develop verifiable ann-based systems. In: Proceedings IEEE International Joint Symposia on Intelligence and Systems. pp. 60–66. IEEE (1996)
75. Wen, W., Callahan, J., Napolitano, M.: Towards developing verifiable neural network controller. Department of Aerospace Engineering, NASA/WVU Software Research Laboratory (1996)
76. Wen, W., Napolitano, M., Callahan, J.: Verifying stability of dynamic soft-computing systems (1997)
77. XAIP: XAIP 2018: Proceedings of the 1st Workshop on Explainable Planning (2018)
78. Xiang, W., Johnson, T.T.: Reachability analysis and safety verification for neural network control systems. CoRR **abs/1805.09944** (2018), http://arxiv.org/abs/1805.09944
79. Yasmin, M., Sharif, M., Mohsin, S.: Neural networks in medical imaging applications: A survey. World Applied Sciences Journal **22**(1), 85–96 (2013)

A Fanoos Structural Overview

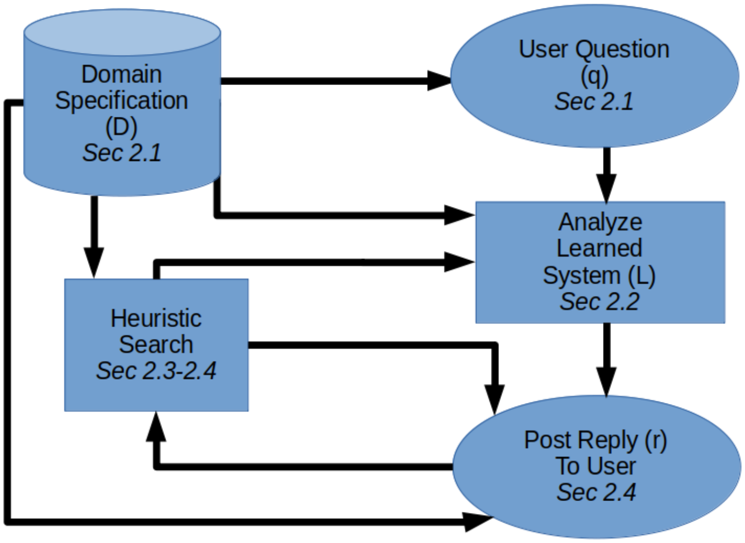


Fig. 2. User-interaction with Fanoos

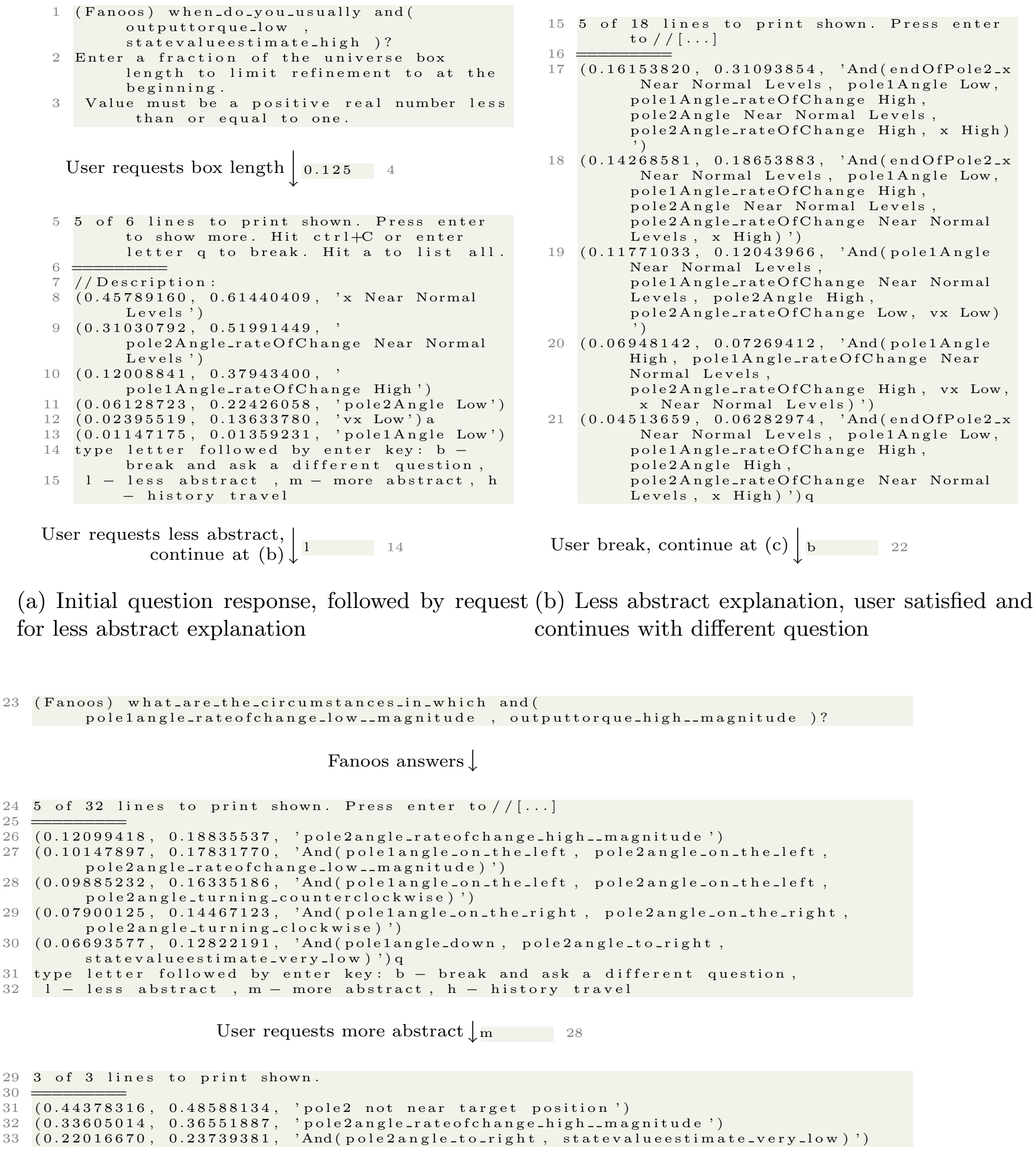
Fig. 2 illustrates the component interactions in Fanoos. Sections detailing the component are italicized. Components requiring user interaction are oval, internal modules are rectangular, and the knowledge database cylindrical.

B Example User Interactions

To demonstrate typical user-interactions with our system, we present here a sample of manual interactions in the spirit of other systems (e.g., [4,6]). In the interest of space, we do not list the meaning of all individual predicates. For those interested, the definition is provided with the code forthcoming. In practice, if users want to know more about exactly what each predicate means operationally (e.g., the exact conditions that each predicate tests for), they can look it up in the domain specification[15]—a large part of the point of this system is to provide functionality beyond just cross-referencing code.

Whenever we insert a comment in the interface-trace that was not originally there, we put `//` at the beginning of the line. Notice that our code uses a Unix-style interaction in the spirit of the `more` command, so not to flood the screen.

[15] This is easily facilitated by open-on-click hyperlinks and/or hover text.



```
(Fanoos) when_do_you_usually and(
     outputtorque_low ,
     statevalueestimate_high )?
Enter a fraction of the universe box
     length to limit refinement to at the
     beginning.
 Value must be a positive real number less
     than or equal to one.
```

```
0.125
```

```
5 of 6 lines to print shown. Press enter
     to show more. Hit ctrl+C or enter
     letter q to break. Hit a to list all.
==========
//Description:
(0.45789160, 0.61440409, 'x Near Normal
     Levels')
(0.31030792, 0.51991449, '
     pole2Angle_rateOfChange Near Normal
     Levels')
(0.12008841, 0.37943400, '
     pole1Angle_rateOfChange High')
(0.06128723, 0.22426058, 'pole2Angle Low')
(0.02395519, 0.13633780, 'vx Low')a
(0.01147175, 0.01359231, 'pole1Angle Low')
type letter followed by enter key: b -
     break and ask a different question,
 l - less abstract , m - more abstract, h
     - history travel
```

```
l
```

(a) Initial question response, followed by request for less abstract explanation

```
5 of 18 lines to print shown. Press enter
     to//[...]
==========
(0.16153820, 0.31093854, 'And(endOfPole2_x
      Near Normal Levels, pole1Angle Low,
     pole1Angle_rateOfChange High,
     pole2Angle Near Normal Levels,
     pole2Angle_rateOfChange High, x High)
     ')
(0.14268581, 0.18653883, 'And(endOfPole2_x
      Near Normal Levels, pole1Angle Low,
     pole1Angle_rateOfChange High,
     pole2Angle Near Normal Levels,
     pole2Angle_rateOfChange Near Normal
     Levels, x High)')
(0.11771033, 0.12043966, 'And(pole1Angle
     Near Normal Levels,
     pole1Angle_rateOfChange Near Normal
     Levels, pole2Angle High,
     pole2Angle_rateOfChange Low, vx Low)
     ')
(0.06948142, 0.07269412, 'And(pole1Angle
     High, pole1Angle_rateOfChange Near
     Normal Levels,
     pole2Angle_rateOfChange High, vx Low,
      x Near Normal Levels)')
(0.04513659, 0.06282974, 'And(endOfPole2_x
      Near Normal Levels, pole1Angle Low,
     pole1Angle_rateOfChange High,
     pole2Angle High,
     pole2Angle_rateOfChange Near Normal
     Levels, x High)')q
```

```
b
```

(b) Less abstract explanation, user satisfied and continues with different question

```
(Fanoos) what_are_the_circumstances_in_which and(
     pole1angle_rateofchange_low__magnitude , outputtorque_high__magnitude )?
```

```
5 of 32 lines to print shown. Press enter to//[...]
==========
(0.12099418, 0.18835537, 'pole2angle_rateofchange_high__magnitude')
(0.10147897, 0.17831770, 'And(pole1angle_on_the_left, pole2angle_on_the_left,
     pole2angle_rateofchange_low__magnitude)')
(0.09885232, 0.16335186, 'And(pole1angle_on_the_left, pole2angle_on_the_left,
     pole2angle_turning_counterclockwise)')
(0.07900125, 0.14467123, 'And(pole1angle_on_the_right, pole2angle_on_the_right,
     pole2angle_turning_clockwise)')
(0.06693577, 0.12822191, 'And(pole1angle_down, pole2angle_to_right,
     statevalueestimate_very_low)')q
type letter followed by enter key: b - break and ask a different question,
 l - less abstract , m - more abstract, h - history travel
```

```
m
```

```
3 of 3 lines to print shown.
==========
(0.44378316, 0.48588134, 'pole2 not near target position')
(0.33605014, 0.36551887, 'pole2angle_rateofchange_high__magnitude')
(0.22016670, 0.23739381, 'And(pole2angle_to_right, statevalueestimate_very_low)')
```

(c) Next question, initial response, and user request to make more abstract

Fig. 3. A sample user session with Fanoos on the inverted double pendulum example

C Input-Space Bounding Box Values

In this section, we list the input-space bounding boxes used in our experiments. We list the values here up to four significant figures. Listings with further precision can be found in the code bases.

Table 4. Inverted double pendulum input-space boxes

Variable name	Lower bound	Upper bound
x	-1	1
vx	-0.8	0.8
pole2_endpoint[1]	-0.5	0.5
pole1angle	-0.2	0.2
pole1angle_rateOfChange	-0.6	0.6
pole2angle	-0.04	0.04
pole2angle_rateOfChange	-0.7	0.7

[1] *pole2_endpoint* is a delta-value with respect to x. That is, in the observation given to the model to standardize, we add x to the value reported for *pole2_endpoint*. This choice is motivated by the fact that the model was trained on the *pole2_endpoint* position being measured in free-space, despite the fact that sensible values for this in an observation are highly dependent on x, the horizontal position of the cart's center.

Table 5. CPU Usage input-space boxes

Variable name	Lower bound	Upper bound
lread	0.0	0.0369
scall	0.0095	0.4245
sread	0.0028	0.0992
freemem	0.0061	0.6275
freeswap	0.4324	0.8318

D Pseudo-Code for Specific-Selection Subroutine

Pseudo-code for our method of finding the most-specific conditions for a box are in Algorithm 1. In our code, we used $\ell = c\exp(\alpha \times c)$ where $c = [1.0, 1.01, 1.05, 1.1, 1.2, 1.4, 1.8, 2.6]$ and α is a non-negative real-valued parameter we store and manipulate in the state. Similarly, n is stored in the states.

```
input : box to fit, $b$; number of random samples to try, $n$; list of predicates to try, $P$; a list of strictly increase real numbers of length starting with 1.0, $\ell$
output: A set of indices into $P$ of the most specific predicates, $s$
$s \leftarrow \{\}$;
$dimsCovered \leftarrow \{\}$;
$bCenter \leftarrow getBoxCenter(b)$;
$bDim = getDimension(b)$;
for $i \leftarrow 0$ to $length(\ell) - 1$ do
    $lowerR \leftarrow \ell[i]$;
    $upperR \leftarrow \ell[i+1]$;
    $innerB \leftarrow ((b - bCenter(b)) \times lowerR) + bCenter(b)$;
    $outterB \leftarrow ((b - bCenter(b)) \times upperR) + bCenter(b)$;
    $randomSamples \leftarrow \{\}$;
    for $j \leftarrow 1$ to $n$ do
        $randomSamples \leftarrow randomSamples \cup \{getRandVecBetweenBoxes(innerBox, outterBox)\}$;
    end
    for $pIndex \leftarrow 0$ to $length(P)$ do
        if $pIndex \in s$ then
            continue;
        end
        else if $freeVars(P[pIndex]) \subseteq dimsCovered$ then
            continue;
        end
        foreach $v \in randomSamples$ do
            /* We evaluate the predicate at index pIndex on v to see if it returns false */
            if $\neg P[pIndex].eval(v)$ then
                $s \leftarrow s \cup \{pIndex\}$;
                $dimsCovered \leftarrow dimsCovered \cup \text{freeVars}(P[pIndex])$;
                break;
            end
        end
    end
    if $length(dimsCovered) == bDim$ then
        return $s$;
    end
end
return $s$;
```

Algorithm 1: Find Most Specific Consistent Predicates