Bridging the Gap: Providing Post-Hoc Symbolic Explanations for Sequential Decision-Making Problems with Inscrutable Representations

Sarath Sreedharan [1] **Utkarsh Soni** [1] **Mudit Verma** [1] **Siddharth Srivastava** [1] **Subbarao Kambhampati** [1]

Abstract

As increasingly complex AI systems are introduced into our daily lives, it becomes important for such systems to be capable of explaining the rationale for their decisions and allowing users to contest these decisions. A significant hurdle to allowing for such explanatory dialogue could be the *vocabulary mismatch* between the user and the AI system. This paper introduces methods for providing contrastive explanations in terms of user-specified concepts for sequential decision-making settings where the system's model of the task may be best represented as a blackbox simulator. We do this by building partial symbolic models of the task that can be leveraged to answer the user queries. We empirically test these methods on a popular Atari game (Montezuma's Revenge) and modified versions of Sokoban (a well known planning benchmark) and report the results of user studies to evaluate whether people find explanations generated in this form useful.

1. Introduction

Recent successes in AI have brought the field a lot of attention, and there is a lot of excitement towards deploying AI-based tools for solving various challenges faced in our daily lives. For these systems to be truly effective in the real world, they need to be capable of working with a lay end user. This means not just inferring optimal decisions, but also being able to allow users to raise explanatory queries wherein they can contest the system's decisions. An obstacle to providing explanations to such questions is the fact that the systems may not have a shared vocabulary with its end users or have an explicit interpretable model of the task. More often than not, the system may be reasoning about the task in a high-dimensional space that is opaque to even the developers of the system, let alone a lay user.

While there is a growing consensus within the explainable AI community that end-user explanations need to be framed in terms of user understandable concepts, the focus generally has been on introducing such methods for explaining one-shot decisions such as in the case of classifiers (c.f. (Kim et al., 2018; Ribeiro et al., 2016)). This is unfortunate as explaining sequential decision-making problems present challenges that are mostly absent from the one-shot decision-making scenarios. In these problems, we not only have to deal with possible interrelationship between the actions in the sequence, but may also need to explain conditions for the executability of actions and the cost of executing certain action sequences. Effectively, this means that explaining a plan or policy to a user would require the system to explain the details of the domain (or at least the agent's belief of it) in terms they can understand.

Barring a few exceptions in summarizing policies like (Hayes & Shah, 2017), most work in explaining sequential decision-making problems have thus used a model specified in a shared vocabulary as a starting point for explanation (Chakraborti et al., 2020). Our work aims to correct this by developing methods that are able to field some of the most fundamental explanatory queries identified in the literature, namely contrastive queries, i.e., questions of the form 'why P (the decision proposed by the system) and not Q (the alternative proposed by the user or the foil)? (Miller, 2018), in user-understandable terms. Our methods achieve this by building partial and abstract symbolic models (Section 2) expressed in terms of the user's vocabulary that approximate the task details relevant to the specific query raised by the user. Specifically, we will focus on deterministic tasks where the system has access to a task simulator and we will identify (a) missing preconditions to explain scenarios where the foil raised by the user results in an execution failure of an action and (b) cost function approximations to explain cases where the foil is executable but suboptimal (Section 3). We *learn* such models by interacting with the simulator (on randomly sampled states) while using learned classifiers that detect the presence of user-specified concepts in the simulator states. Figure 1 presents the overall flow of this process with illustrative

[1]School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281 USA. Correspondence to: Sarath Sreedharan <ssreedh3@asu.edu>.

2020 ICML Workshop on Human in the Loop Learning (HILL 2020).

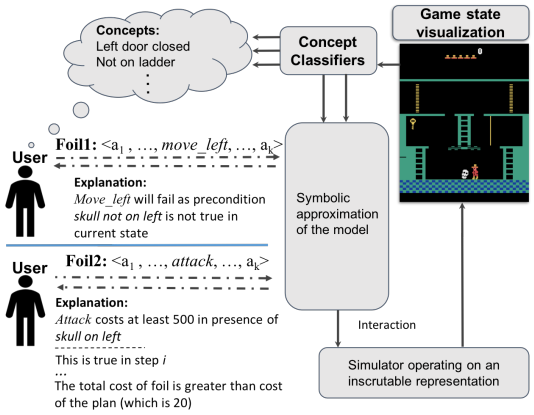

Figure 1. The explanatory dialogue starts when the user presents the system with a specific alternate plan (foil). Here we consider two foils, one that is invalid and another that is costlier than the plan. The system explains the invalid plan by pointing out an action precondition that was not met in the plan, while it explains the foil suboptimality by informing the user about cost function. Each of these model information is expressed in terms of concepts specified by the user which we operationalize by learning a classifier for each concept.

explanations in the context of a slightly updated version of Montezuma's Revenge (Wikipedia contributors, 2019). Our methods also allow for the calculation of confidence over the explanations and explicitly take into account the fact that learned classifiers for user-specified concepts may be noisy. This ability to quantify its belief about the correctness of explanation is an important capability for any post-hoc explanation system that may influence the user's decisions. We evaluate the system on two popular sequential decision making domains, Montezuma's Revenge and a modified version of Sokoban (Botea et al., 2002) (a game involving players pushing boxes to specified targets). We present user study results that show the effectiveness of explanations studied in this paper (Section 5).

2. Background

In this work, we focus on cases where a human observer is trying to make sense of plans proposed by an autonomous system. When the plan differs from the users' expectations, they try to make sense of this disparity by asking why some alternate expected plan was not proposed. The goal then becomes addressing such counterfactual queries in terms of the dynamics of the domain in question, expressed using user-specified concepts. We assume that the decision-making algorithm computes the optimal solution, and thus our focus isn't on how the algorithm came up with the specific decisions, but only on why this action sequence was chosen instead of an alternative that the user expected. We assume access to a deterministic simulator of the form $\mathcal{M}_{\text{sim}} = \langle S, A, T, \mathcal{C} \rangle$, where S represents the set of possible world states, A the set of actions and T the transition function that specifies the problem dynamics. The transition function is defined as $T : S \times A \rightarrow S \cup \{\perp\}$, where $\perp$ corresponds to an invalid absorber-state generated by the execution of an infeasible action. Invalid state could be used to capture failure states that could occur when the agent violates hard constraints like safety constraints. Finally, $\mathcal{C} : S \times A \rightarrow \mathsf{R}$ captures the cost function of executing an action at a particular state (with cost of an infeasible action taken to be infinite). We will overload the transition function T to also work on action sequence, i.e., $T(s, \langle a_1, ..., a_k \rangle) = T(...T(T(s, a_1), a_2), ..., a_k)$. We will look at goal-directed problems in the sense that the decision-making system needs to come up with the plan, i.e., a sequence of actions, $\pi = \langle a_1, .., a_k \rangle$, that will drive the state of the world to a goal state. In general we will use the tuple $\Pi_{\text{sim}} = \langle I, \mathsf{G}, \mathcal{M}_{\text{sim}} \rangle$ to represent the decision making problem, where I is the initial state and G the set of goal states. Moreover a plan is optimal if it achieves the goal and there exists no cheaper plan that can achieve the goal (where $\mathcal{C}(I, \pi)$ is the total cost of executing π).

We will use symbolic action models with preconditions and cost functions (similar to STRIPS models (Geffner & Bonet, 2013)) as a way to approximate the problem for explanations. Such a model can be represented by the tuple $\Pi_{\mathsf{S}} = \langle F_{\mathsf{S}}, A_{\mathsf{S}}, I_{\mathsf{S}}, G_{\mathsf{S}}, \mathcal{C}_{\mathsf{S}} \rangle$, where F_{S} is a set of propositional fluents defining the state space, A_{S} is the set of actions, I_{S} is the initial state, G_{S} is the goal specification. Each valid problem state in the problem is uniquely identified by the subset of fluents that are true in that state (so for any state s, $s \subseteq F_{\mathsf{S}}$). We will use the notation $S_{M_{\mathsf{S}}}$ to denote the space of all possible states for the model $\mathcal{M}_{\mathsf{S}}$. Each action $a \in A_{\mathsf{S}}$ is further described in terms of the preconditions $prec_a$ (specification of states in which a is executable) and the effects of executing the action. We will denote the state formed by executing action a in state s as $a(s)$. We will focus on models where the preconditions are represented as a conjunction of state factors. If the action is executed in a state with missing preconditions, then the execution results in the invalid state ($\perp$). Unlike standard STRIPS models, where the cost of executing action is independent of states, we will be using a state dependent cost function of the form $\mathcal{C}_{\mathsf{S}} : 2^F \times A_{\mathsf{S}} \rightarrow \mathsf{R}$ to capture the cost of valid action executions. Internally, such state models may be represented using conditional cost models of the type discussed in (Geißer et al., 2016). In this paper, we won't try to reconstruct the exact cost function but will rather try to estimate an abstract version of the cost function.

3. Contrastive Explanations

The specific explanatory setting, illustrated in Figure 1, that we are interested in studying involves a decision-making problem specified by the tuple $\Pi_{sim} = \langle I, \mathsf{G}, \mathcal{M}_{\text{sim}} \rangle$ for which the system identifies a plan π. When presented with the plan, the user of the system may either accept it or responds by raising an alternative plan π_f (*the foil*) that they believe should be followed instead. Now the system would need to provide an explanation as to why the plan π may be preferred over the foil π_f in question. The only two possibilities here are that either the foil is inexecutable and hence can not be followed or it is costlier than the plan in question.[1] More formally,

Definition 1 *The plan π is said to be preferred over a foil π_f for a problem $\Pi_{sim} = \langle I, \mathsf{G}, \mathcal{M}_{sim} \rangle$, if either of the following conditions are met, i.e.,*

1. π_f is inexecutable*, which means, either (a) $T(I, \pi_f) \notin \mathsf{G}$, i.e the action sequence doesn't lead to a possible goal state, or (b) the execution of the plan leads to an invalid state, i.e., $T(I, \pi_f) = \perp$.*
2. *Or π_f is* costlier *than π, i.e., $\mathcal{C}(I, \pi) < \mathcal{C}(I, \pi_f)$*

To concretize this interaction, consider an instance from a modified version of Montezuma's revenge (Figure 1). Let's assume the agent starts from the highest platform, and the goal is to get to the key. The specified plan π may require the agent to make its way to the lowest level, jump over the skull, and then go to the key with a total cost of 20. Let us consider a case where the user raises two possible foils that are quite similar to π, but, (a) in the first foil, instead of jumping the agent just moves left (as in it tries to move through the skull) and (b) in the second, instead of jumping over the skull, the agent performs the attack action (not part of the original game, but added here for illustrative purposes) and then moves on to the key. Now using the simulator, the system could tell that in the first case, moving left would lead to an invalid state and in the second case, the foil is more expensive. It may however struggle to explain to the user what particular aspects of the state or state sequence lead to the invalidity or suboptimality. Even efforts to localize parts of its own internal state representation for possible reasons by comparing the foil with similar states where actions are executable or cheaper may be futile, as even what constitutes similar states as per the simulator may be conceptually quite confusing for the user. This scenario thus necessitates the use of methods that are able to express possible explanations in terms that the user may understand.

Representational Assumptions A quick note on some of the representational assumptions we are making. The central one we are making is of course that it is possible to approximate the applicability of actions and cost function in terms of high-level concepts. Apart from the intuitive appeal of such models (many of these models have their origin in models from folk psychology), these representation schemes have been widely used to model real-world sequential decision-making problems from a variety of domains and have a clear real-world utility (Benton et al., 2019). We agree that there may be problems where it may not be directly applicable, but we believe this is a sound initial step and applicable to many domains where currently Reinforcement Learning (RL) based decision-making systems are being successfully used, including robotics and games.

Apart from this basic assumption, we make one additional representational assumption, namely, that the precondition can be expressed as a conjunction of positive concepts. Note that the assumption doesn't restricts the applicability of the methods discussed here. Our framework can still cover cases where the action may require non-conjunctive preconditions. To see why, consider a case where the precondition of action a is expressed as an arbitrary propositional formula, $\phi(\mathsf{C})$. In this case, we can express it in its conjunctive normal form $\phi^0(\mathsf{C})$. Now each clause in $\phi^0(\mathsf{C})$ can be treated as a new compound positive concept. Thus we can cover such arbitrary propositional formulas by expanding our concept list with compound concepts (including negations and disjuncts) whose value is determined from the classifiers for the corresponding atomic concepts.

Concept maps: To describe explanations in this setting, the system needs to have access to a mapping from it's internal state representation to a set of high-level concepts that are known to the user (for Montezuma this could involve concepts like *agent being on ladder, holding onto key, being next to a skull etc.*). We will assume each concept corresponds to a propositional fact that the user associates with the task's states and believes that the dynamics of the task are determined by these concepts. This means that as per the user, for each given state, a subset of these concepts may be either present or absent. Our method assumes that we have access to binary classifiers for each concept that may be of interest to the user. The classifiers provide us with a way to convert simulator states to a factored representation. Such techniques have not only been used in explanation (c.f. (Kim et al., 2018; Hayes & Shah, 2017)) but also in works that have looked at learning high-level representations for continuous state-space problems (c.f. (Konidaris et al., 2018)). Let C be the set of classifiers corresponding to the high-level concepts. For state $s \in S$, we will overload the notation C and specify the concepts that are true as $\mathsf{C}(s)$, i.e., $\mathsf{C}(s) = \{c_i | c_i \in \mathsf{C} \wedge c_i(s) = 1\}$ (where c_i is the classifier corresponding to the i^{th} concept, we will overload this notation and also use it to stand for the label of

[1] If the foil is as good as the original plan, then the system could switch to foil without loss of optimality.

the i^{th} concept). The user could specify the set of concepts by identifying positive and negative example states for each concept. These examples could then be used to learn the required classifiers by using algorithms best suited for the internal simulator state representation. This means that the explanatory system should have some method of exposing simulator states to the user. A common way to satisfy this requirement would be by having access to visual representations for the states. The simulator state itself doesn't need to be an image as long as we have a way to visualize it (for example in Atari games where the states can be represented by the RAM state of the game controller but we can still visualize it). The concept list can also be mined from qualitative descriptions of the domain and we can crowd source the collection of example states for each concept.

Explanation using concepts: To explain why a given foil is not preferred over the specified plan, we will present information about the symbolic model expressed in user's vocabulary, $\mathcal{M}^{\mathsf{C}}_S = \langle \mathsf{C}, A^{\mathsf{C}}_S, \mathsf{C}(I), \mathsf{C}(\mathsf{G}), \mathcal{C}^{\mathsf{C}}_S \rangle$. Where $\mathsf{C}(\mathsf{G}) = \bigcap_{s_g \mathit{2} \mathsf{G}} \mathsf{C}(s_g)$ and A^{C}_S contains a definition for each action $a \in A$. The model is a sound abstraction of the simulator $\mathcal{M}_{\text{sim}} = \langle S, T, A, \mathcal{C} \rangle$ for regions of interest $\hat{S} \subseteq S$, in so far as, $\forall s \in \hat{S}$ and $\forall a \in A$, we have an equivalent action $a^{\mathsf{C}} \in A^{\mathsf{C}}_S$, such that $a^{\mathsf{C}}(\mathsf{C}(s)) = \mathsf{C}(T(s,a))$ (assuming $\mathsf{C}(\bot) = \bot$) and $\mathcal{C}^{\mathsf{C}}_S(\mathsf{C}(s), a) = \mathcal{C}(s,a)$. Note that establishing the preference of plan does not require informing the users about the entire model, but rather only the relevant parts. For conciseness, we will use a_i for both the simulator action and the corresponding abstract action in the symbolic model as long as the context allows it to be distinguished.

For establishing the invalidity of π_f, we just need to focus on explaining the failure of the first failing action a_i, i.e., the last action in the shortest prefix that would lead to an invalid state (which in our running example is the move-left action in the state presented in Figure 1 for the first foil). We can do so by informing the user that the failing action has an unmet precondition, as per the symbolic model, in the state it was executed in. Formally

Definition 2 *For a failing action a_i for the foil $\pi_f = \langle a_1, .., a_i, .., a_n \rangle$, $c_i \in \mathsf{C}$ is considered an explanation for failure if $c_i \in prec_{a_i} \setminus \mathsf{C}(s_i)$, where s_i is the state where a_i is meant to be executed (i.e $s_i = T(I, \langle a_1, .., a_{i\ \ 1} \rangle)$).*

In our example for the invalid foil, a possible explanation would be to inform the user that move-left can only be executed in states for which the concept skull-not-on-left is true; and the concept is false in the given state. This formulation is enough to capture both conditions for foil inexecutability by appending an additional goal action at the end of each sequence. The goal action causes the state to transition to an end state and it fails for all states except the ones in G. Our approach to identifying the minimal information needed to explain specific query follows from studies in social sciences that have shown that selectivity or minimality is an essential property of effective explanations (Miller, 2018).

For explaining the suboptimality of the foil, we have to inform the user about $\mathcal{C}^{\mathsf{C}}_S$. To ensure minimality of explanations, rather than generating the entire cost function or even trying to figure out individual conditional components of the function, we will instead try to learn an abstraction of the cost function $\mathcal{C}^{abs}_s$, defined as follows

Definition 3 *For the symbolic model $\mathcal{M}^{\mathsf{C}}_S = \langle \mathsf{C}, A^{\mathsf{C}}_S, \mathsf{C}(I), \mathsf{C}(\mathsf{G}), \mathcal{C}^{\mathsf{C}}_S \rangle$, an abstract cost function $\mathcal{C}^{abs}_S : 2^{\mathsf{C}} \times A^{\mathsf{C}}_S \rightarrow \mathsf{R}$ is specified as follows $\mathcal{C}^{abs}_S(\mathsf{f}c_1, .., c_k\mathsf{g}, a) = min\mathsf{f}\mathcal{C}^{\mathsf{C}}_S(s,a) \mathit{js\ 2}\ S_{M^{\mathsf{C}}_S} \mathsf{\wedge}\ \mathsf{f}c_1, .., c_k\mathsf{g}\ s\mathsf{g}]$.*

Intuitively, $\mathcal{C}^{abs}_S(\mathsf{f}c_1, .., c_k\mathsf{g}, a) = k$ can be understood as stating that *executing the action a, in the presence of concepts $\{c_1, .., c_k\}$ costs at least k*. We can use $\mathcal{C}^{abs}_S$ in an explanation of the form

Definition 4 *For a valid foil $\pi_f = \langle a_1, .., a_k \rangle$, a plan π and a problem $\Pi_{sim} = \langle I, \mathsf{G}, \mathcal{M}_{sim} \rangle$, the sequence of concept sets of the form $\mathsf{C}_{\pi_f} = \langle \hat{\mathsf{C}}_1, ..., \hat{\mathsf{C}}_k \rangle$ along with $\mathcal{C}^{abs}_s$ is considered a valid explanation for relative suboptimality of the foil (denoted as $\mathcal{C}^{abs}_S(\mathsf{C}_{\pi_f}, \pi_f) > \mathsf{C}(I, \pi)$), if $\forall \hat{\mathsf{C}}_i \in \mathsf{C}_{\pi_f}$, $\hat{\mathsf{C}}_i$ is a subset of concepts presents in the corresponding state (where state is I for $i = 1$ and $T(I, \langle a_1, ..., a_{i\ \ 1} \rangle)$ for $i > 1$) and $\Sigma_{i=f1..kg}\mathcal{C}^{abs}_S(\hat{\mathsf{C}}_i, a_i) > \mathsf{C}(I, \pi)$*

In the earlier example, the explanation would include the fact that executing the action attack in the presence of the concept skull-on-left, will cost at least 500 (as opposed to original plan cost of 20).

4. Identifying Explanations through Sample-Based Trials

For identifying the model parts for explanatory query, we will rely on the agent's ability to interact with the simulator to build estimates. Given the fact that we can separate the two cases at the simulator level, we will keep the discussion of identifying each explanation type separate and only focus on identifying the model parts once we know the failure type.

Identifying failing precondition: To identify the missing preconditions, we will rely on the simple intuition that while successful execution of an action a in the state s_j with a concept C_i doesn't necessarily establish that C_i is a precondition, we can guarantee that any concept false in that state can not be a precondition of that action. This is a common line of reasoning exploited by many of the model learning methods (c.f (Carbonell & Gil, 1990; Stern

& Juba, 2017)). So we start with the set of concepts that are absent in the the state (s_{fail}) where the failing action (a_{fail}) was executed, i.e., poss_prec_set = $\mathsf{C} \setminus \mathsf{C}(s_{\text{fail}})$. We then randomly sample for states where a_{fail} is executable. Each new sampled state s_i where the action is executable can then be used to update the possible precondition set as poss_prec_set = poss_prec_set $\cap$ $\mathsf{C}(s_i)$. That is, if a state is identified where the action is executable but a concept is absent then it can't be part of the precondition. We will keep repeating this sampling step until the sampling budget is exhausted or if one of the following exit conditions is met. (a) In cases where we are guaranteed that the concept list is exhaustive, we can quit as soon as the set of possibilities reduce to one (since there has to be a missing precondition at the failure state). (b) The search results in an empty list. The list of concepts left at the end of exhausting the sampling budget represents the most likely candidates for preconditions. *An empty list here signifies the fact that whatever concept is required to differentiate the failure state from the executable one is not present in the initial concept list* (C). This can be taken as evidence to query the user for more task-related concepts. Any locality considerations for sampled states, like focusing on states close to the plan/foil, can be baked into the sampler. The full specification of the algorithm is provided in the supplementary file (https://bit.ly/38TmDPA).

Identifying cost function: Now we will employ a similar sampling based method to identify the right cost function abstraction. Unlike the precondition failure case, there is no single action we can choose but rather we need to choose a level of abstraction for each action in the foil (though it may be possible in many cases to explain the suboptimality of foil by only referrring to a subset of actions in the foil). Our approach here would be to find the most abstract representation of the cost function at each step such that of the total cost of the foil becomes greater than that of the specified plan. Thus for a foil $\pi_f = \langle a_1, ..., a_k \rangle$ our objective become

$$\min_{\hat{\mathsf{C}}_1,\ldots,\hat{\mathsf{C}}_k} \Sigma_{i=1..k} k\hat{\mathsf{C}}_i k \text{ subject to } \mathcal{C}_s^{abs}(\mathsf{C}_{\pi_f}, \pi_f) > \mathsf{C}(I, \pi)$$

For any given $\hat{\mathsf{C}}_i$, $\mathcal{C}_s^{abs}(\hat{\mathsf{C}}_i, a_i)$ can be approximated by sampling states randomly and finding the minimum cost of executing the action a_i in states containing the concepts $\hat{\mathsf{C}}_i$. We can again rely on a sampling budget to decide how many samples to check and enforce required locality within sampler. Similar to the previous case, *we can identify the insufficiency of the concept set by the fact that we aren't be able to identify a valid explanation*. The algorithm can be found in the supplementary file (https://bit.ly/38TmDPA).

Confidence over explanations: Though both the methods discussed above are guaranteed to identify the exact model in the limit, the accuracy of the methods is still limited by practical sampling budgets we can employ. So this means it is important that we are able to establish some level of confidence in the solutions identified. To assess confidence,

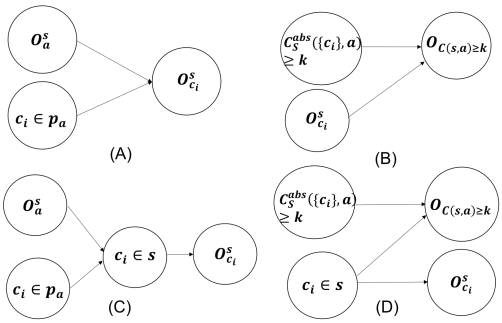

Figure 2. A simplified probabilistic graphical models for explanation inference, Subfigure (A) and (B) assumes classifiers to be completely correct, while (C) and (D) presents cases with noisy classifier.

we will follow the probabilistic relationship between the random variables as captured by Figure 2 (A) for precondition identification and Figure 2 (B) for cost calculation. Where the various random variables captures the following facts: O_a^s - indicates that action a can be executed in state s, $c_i \in p_a$ - concept c_i is a precondition of a, $O_{c_i}^s$ - the concept c_i is present in state s, $\mathcal{C}_s^{abs}(\{c_i\}, a) \geq k$ - the abstract cost function is guaranteed to be higher than or equal to k and finally $O_{C(s,a)\ \ k}$ - stands for the fact that the action execution in the state resulted in cost higher than or equal to k. We will allow for inference over these models, by relying on the following simplifying assumptions - (1) the distribution of concepts over the state space is independent of each other, (2) the distribution of all non-precondition concepts in states where the action is executable is the same as their overall distribution across the problem states (which can be empirically estimated), (3) cost distribution of an action over states corresponding to a concept that does not affect the cost function is identical to the overall distribution of cost for the action (which can again be empirically estimated). The second assumption implies that you are as likely to see a non-precondition concept in a sampled state where the action is executable as the concept was likely to appear at any sampled state (this distribution is denoted as p_{c_i}). While the third one implies that for a concept that has no bearing on the cost function for an action, the likelihood that executing the action in a state where the concept is present will result in a cost greater than k will be the same as that of the action execution resulting in cost greater than k for a randomly sampled state ($p_{C(.,a)\ \ k}$).

For a single sample, the posterior probability of explanations for each case can be expressed as follows: For precondition estimation, updated posterior probability for a positive observation can be computed as $P(c_i \ 2 \ p_a j O_{c_i}^s \wedge O_a^s) = (1 \quad P(c_i \ 2 \ p_a j O_{c_i}^s \wedge O_a^s))$, where

$$P(c_i \ 2 \ p_a j O_{c_i}^s \wedge O_a^s) = \frac{p_{c_i} \quad P(c_i \ 2 \ p_a)}{P(O_{c_i}^s j O_a^s)}$$

and for the case of cost function approximation

$$P(C_s^{abs}(fc_ig, a) \quad kjO^s_{c_i} \wedge O_{C(s,a)} \quad _k) = \frac{P(C_s^{abs}(fc_ig, a) \quad k)}{P(C_s^{abs}(fc_ig, a) \quad k)) + p_{C(.,a)} \quad _k \quad P(:C_s^{abs}(fc_ig, a) \quad k))}$$

Full derivation of above formulas can be found in the supplementary file (https://bit.ly/38TmDPA). The distribution used in the cost explanation, can either be limited to distribution over states where action a_i is executable or allow for the cost of executing an action in a state where it is not executable to be infinite.

Using noisy concept classifiers: Given how unlikely it is to have access to a perfect classifier for any concept, a more practical assumption to adopt could be that we have access to a noisy classifier. However, we assume that we also have access to a probabilistic model for its prediction. That is, we have access to a function $P_C : C \rightarrow [0, 1]$ that gives the probability that the concept predicted by the classifier is actually associated with the state. Such probability functions could be learned from the test set used for learning the classifier. Allowing for the possibility of noisy observation generally has a more significant impact on the precondition calculation than the cost function approximation. Since we are relying on just generating a lower bound for the cost function, we can be on the safer side by under-approximating the cost observations received (though this could lead to larger than required explanation). In the case of precondition estimation, we can no longer use a single failure (execution of an action in a state where the concept is absent) as evidence for discarding the concept. Though we can still use it as an evidence to update the probability of the given concept being a precondition. We can remove a particular possible precondition from consideration once the probability of it not being a precondition crosses a specified threshold.

To see how we can incorporate these probabilistic observations into our confidence calculation, consider the updated relationships presented in Figure 2 (C) and (D) for precondition and cost function approximation. Note that in previous sections, we made no distinction between the concept being part of the state and actually observing the concept. Now we will differentiate between the classifier saying that a concept is present ($O^s_{c_i}$) from the fact that the concept is part of the state ($c_i \in C(S)$). Now we can use this updated model for calculating the confidence. We can update the posterior of a concept not being a precondition given a negative observation ($O^s_{:c_i}$) using the formula

$$P(c_i \text{ 🙂 } p_a jO^s_{:c_i} \wedge O^s_a) = \frac{P(O^s_{:c_i} jc_i \text{ 🙂 } p_a \wedge O^s_a) \quad P(c_i \text{ 🙂 } p_a jO^s_a)}{P(O_{:c_i} jO^s_a)}$$

Similarly we can modify the update for a positive observation to include the observation model and also do the same for the cost explanation. For calculation of cost confidence, we will now need to calculate $P(O_{C(s,a)} \quad _k jc_i \text{ 🙂 } C(s), C_s^{abs}(fc_ig, a) \quad k)$. This can either be empirically calculated from samples with true label or we can assume that this value is going to be approximately equal to the overall distribution of the cost for the action.

The derivations for all of the above expressions and formulas for the other cases can be found in the supplementary file.

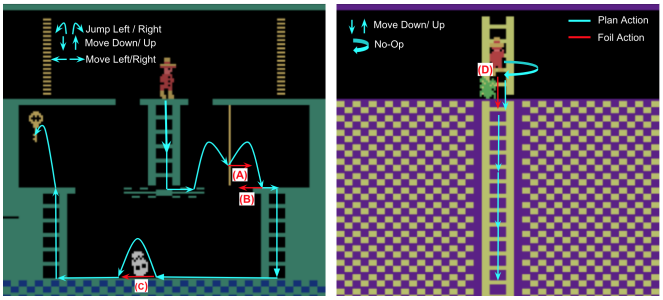

Figure 3. Montezuma Foils: Left Image shows foils for screen 1, (A) Move right instead of Jump Right (B) Go left over the edge instead of using ladder (C) Go left instead of jumping over the skull. Right Image shows foil for screen 4, (D) Move Down instead of waiting.

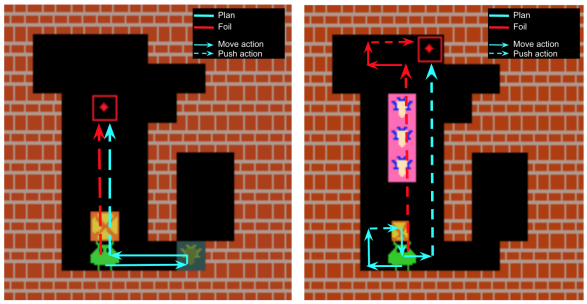

Figure 4. Sokoban Foils: Left Image shows foils for Sokoban-switch, note that the green cell will turn pink once the agent passes it. Right Image shows foil for screen Sokoban-cell.

5. Evaluation

For validating the soundness of the methods discussed before, we tested the approach on the open-AI gym's deterministic implementation of Montezuma's Revenge (Brockman et al., 2016) for precondition identification and two modified versions of the gym implementation of Sokoban (Schrader, 2018) for cost based foil evaluation. The first version of Sokoban included a switch the player could turn on to reduce the cost of pushing the box (we will refer to this version as Sokoban-switch) and the second version (Sokoban-cells) included particular cells from which it is costlier to push the box. We used RAM-based state representation for Montezuma and images of game state for the Sokoban variations. To add richer preconditions to the settings, we added a wrapper over the original simulators for all the games to render

any action that fails to change the current agent state as an action failure. For Montezuma, we selected four invalid foils for the game, three from screen 1 and one from screen 4 and for Sokoban, we selected one valid but suboptimal foil for each variation. The specifics of the games, foils, plans and state representation are provided in the supplementary file (https://bit.ly/38TmDPA). The plan and foils used can be found in Figure 3 (for Montezuma) and Figure 4 (for Sokoban variants).

Concept learning: For Montezuma, we specified ten concepts for each screen and for the Sokoban variations we used a survey to collect the set of concepts. The survey allowed participants to interact with the game through a web interface, and at the end, they were asked to specify game concepts that they thought were relevant for particular actions. For Sokoban-switch, we collected data from six participants and received 25 unique concepts and for Sokoban-cell we collected data from seven participants and received 38 unique concepts. In both domains, we wrote scripts to identify positive and negative examples for each concept from randomly sampled states of the game. For Sokoban variants, we rejected any concept that resulted in less than ten samples after 1000 episodes of sampling and consequently we focused on just 20 concepts for Sokoban-switch and 32 concepts for Sokoban-cell respectively. We used AdaBoost Classifier (Freund et al., 1999) on Montezuma, and Convolutional Neural Networks (CNNs) for Sokoban variants. The CNN architecture involved four convolutional layers followed by three fully connected layers that gave a binary classification output. Montezuma classifiers had an average accuracy of 99.72%, while that for Sokoban-switch was 99.46% and for Sokoban-cell was 99.34%.

Explanation identification: As mentioned previously, we ran the search for identifying preconditions for Montezuma's foils and cost function identification on Sokoban. From the collected list of concepts, we doubled the final concept list used by including negations of each concept. So for Montezuma we used 20 concepts per screen, and 40 and 64 concepts were used for Sokoban-switch and Sokoban-cell. The probabilistic models for each classifier were calculated from the corresponding test sets. For precondition identification, the search was run with a sampling budget of 500 and a cutoff probability of 0.01 for each concept. The search was able to identify the expected explanation for each foil and had a mean confidence of 0.5044 for foils in screen 1 and a confidence value of 0.8604 for the foil in screen 4. The ones in screen 1 had lower probabilities since they were based on more common concepts and thus their presence in the executable states was not a strong evidence for them being a precondition. For cost function identification, the search was run with a sampling budget of 750 and all the calculations, including both computing the concept distribution and updating the probability of explanation, were limited to states where the action was executable. Again the search was able to find the expected explanation. We had an average confidence of 0.9996 for the Sokoban-switch and 0.998 for the Sokoban-cell.

User study: With the basic explanation generation method in place, we were interested in evaluating if users would find such an explanation helpful. Specifically, the hypothesis we tested were

Hypothesis 1*: Missing precondition information is a useful explanation for action failures.*

Hypothesis 2*: Abstract cost functions are a useful explanation for foil suboptimality.*

To evaluate this, we performed a user study with all the foils used along with the generated explanation and a simple baseline explanation. For precondition case (H1), the baseline involved pointing out just the failing action and the state it was executed. For the cost case (H2), it involved pointing out the exact cost of executing each action in the foil. The users were asked to choose the one they believed was more useful (the choice ordering was randomized to ensure the results were counterbalanced) and were also asked to report on a Likert scale the completeness of the chosen explanation. For each foil, we took the explanation generated by the search and converted it into text by hand. The subjects were also given the option to provide suggestions on what they think would help improve the completeness of the explanation in a free text field. For H1, we collected 20 replies in total (five per foil) and 19 out of the 20 participants selected precondition based explanation as the choice. On the question of whether the explanation was complete, we had an average score of 3.35 out of 5 on the Likert scale (1 being not at all complete and 5 being complete). For H2, we again collected 20 replies in total (ten per foil) and found 14 out of 20 participants selected the concept-based explanation over the simple one. The concept explanations had on average a completeness score of 3.36 out of 5. The results seems to suggest that in both case people did prefer the concept-based explanation over the simple alternative. The completeness results suggest that people may like, at least in some cases, to receive more information about the model.

6. Related Work

There is an increasing number of works investigating the use of high-level concepts to provide meaningful post-hoc explanations to the end-users. The representative works in this direction include TCAV (Kim et al., 2018) and its various offshoots like (Luss et al., 2019) that have focused on one-shot decisions. Authors of (Hayes & Shah, 2017) have looked at the use of high-level concepts for policy summaries. They use logical formulas to concisely characterize

various policy choices, including states where a specific action may be selected (or not). Unlike our work, they are not trying to answer why the policy ends up choosing a specific action (or not). (Waa et al., 2018) looks at addressing suboptimality of foils while supporting interpretable features, but it requires the domain developer to specifically encode positive and negative outcomes to each action. Another related work is the approach studied in (Madumal et al., 2020). Here, they are also trying to characterize dynamics in terms of high-level concepts. Though in their example, they assume that the full structural relationship between the various variables is provided upfront. The explanations discussed in this paper can also be seen as a special case of Model Reconciliation explanation (c.f (Chakraborti et al., 2017)), where the human model is considered to be empty and our use of abstractions is also connected to the HELM explanatory framework introduced in (Sreedharan et al., 2018). Usefulness of preconditions as explanation has also been considered in other explanatory works like (Winikoff, 2017; Broekens et al., 2010). Our effort to associate action cost to concepts could also be contrasted to efforts in (Juozapaitis et al., 2019) and (Anderson et al., 2019) which explain in terms of interpretable reward components. Unfortunately, their method relies on having reward function being represented using interpretable components.

7. Conclusion

We view the approaches introduced in the paper as the first step towards designing more general symbolic explanatory methods for sequential decision-making problems that operate on inscrutable representations. The current methods facilitate generation of explanations in user-specified terms for sequential decisions by allowing users to query the system about alternative plans. We implemented these method in multiple domains and evaluated the effectiveness of the explanation using user studies. While contrastive explanations are answers to questions of the form "Why P and not Q?", we have mostly focused on refuting the foil (the "not Q?" part). This is because, in the presence of a simulator, it is easier to show why the plan is valid by simulating the plan and presenting the state trace. We can further augment such traces with the various concepts that are valid at each step of the trace. Also, note that the methods discussed in this paper can still be used if the user's questions are specified in term of temporal abstraction over the agent's action space. As long as the system can simulate the foils raised by the user, we can keep the rest of the methods discussed in the paper the same. In future, we would like to investigate more complex tasks, such as those with stochastic dynamics or partial observability. Also, we could look at extending the methods to support partial foils (where the user only specifies some part of the plan) and develop methods that allow for efficient acquisitions of new concepts from the user.

As mentioned, the readers can find the supplementary file containing the algorithm pseudocodes, derivation of all the formulas (along with formulas for noisy classifier cases) and additional experiment details at the link `https://bit.ly/38TmDPA`.

Acknowledgments

Kambhampati's research is supported in part by ONR grants N00014-16-1-2892, N00014-18-1-2442, N00014-18-1-2840, N00014-9-1-2119, AFOSR grant FA9550-18-1-0067, DARPA SAIL-ON grant W911NF-19-2-0006, NSF grants 1936997 (C-ACCEL), 1844325, NASA grant NNX17AD06G, and a JP Morgan AI Faculty Research grant.

References

Anderson, A., Dodge, J., Sadarangani, A., Juozapaitis, Z., Newman, E., Irvine, J., Chattopadhyay, S., Fern, A., and Burnett, M. Explaining Reinforcement Learning to Mere Mortals: An Empirical Study. In *IJCAI*, 2019.

Benton, J., Lipovetzky, N., Onaindia, E., Smith, D. E., and Srivastava, S. (eds.). *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019*, 2019. AAAI Press. URL `https://aaai.org/ojs/index.php/ICAPS/issue/view/239`.

Botea, A., Müller, M., and Schaeffer, J. Using abstraction for planning in sokoban. In *International Conference on Computers and Games*, pp. 360–375. Springer, 2002.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *CoRR*, abs/1606.01540, 2016. URL `http://arxiv.org/abs/1606.01540`.

Broekens, J., Harbers, M., Hindriks, K., Van Den Bosch, K., Jonker, C., and Meyer, J.-J. Do you get it? user-evaluated explainable bdi agents. In *German Conference on Multiagent System Technologies*, pp. 28–39. Springer, 2010.

Carbonell, J. G. and Gil, Y. Learning by experimentation: The operator refinement method. In *Machine learning*, pp. 191–213. Elsevier, 1990.

Chakraborti, T., Sreedharan, S., Zhang, Y., and Kambhampati, S. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *IJCAI*, pp. 156–163, 2017.

Chakraborti, T., Sreedharan, S., and Kambhampati, S. The emerging landscape of explainable ai planning and decision making. *arXiv preprint arXiv:2002.11697*, 2020.

Freund, Y., Schapire, R., and Abe, N. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

Geffner, H. and Bonet, B. A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1): 1–141, 2013.

Geißer, F., Keller, T., and Mattmüller, R. Abstractions for planning with state-dependent action costs. *ICAPS*, 2016.

Hayes, B. and 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.

Juozapaitis, Z., Koul, A., Fern, A., Erwig, M., and Doshi-Velez, F. Explainable Reinforcement Learning via Reward Decomposition. In *IJCAI Workshop on explainable AI (XAI)*, 2019.

Kim, B., M., W., Gilmer, J., C., C., J., W., , Viegas, F., and Sayres, R. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV) . *ICML*, 2018.

Konidaris, G., Kaelbling, L. P., and Lozano-Perez, T. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

Luss, R., Chen, P.-Y., Dhurandhar, A., Sattigeri, P., Zhang, Y., Shanmugam, K., and Tu, C.-C. Generating contrastive explanations with monotonic attribute functions. *arXiv preprint arXiv:1905.12698*, 2019.

Madumal, P., Miller, T., Sonenberg, L., and Vetere, F. Explainable reinforcement learning through a causal lens. In *AAAI*, 2020.

Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 2018.

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

Schrader, M.-P. B. gym-sokoban. https://github.com/mpSchrader/gym-sokoban, 2018.

Sreedharan, S., Srivastava, S., and Kambhampati, S. Hierarchical expertise level modeling for user specific contrastive explanations. In *IJCAI*, pp. 4829–4836, 2018.

Stern, R. and Juba, B. Efficient, safe, and probably approximately complete learning of action models. *arXiv preprint arXiv:1705.08961*, 2017.

Waa, J., Diggelen, J. v., Bosch, K., and Neerincx, M. Contrastive Explanations for Reinforcement Learning in Terms of Expected Consequences. In *IJCAI Workshop on explainable AI (XAI)*, 2018.

Wikipedia contributors. Montezuma's revenge (video game) — Wikipedia, the free encyclopedia, 2019. URL https://en.wikipedia.org/w/index.php?title=Montezuma%27s_Revenge_(video_game)&oldid=922152932. [Online; accessed 14-January-2020].

Winikoff, M. Debugging agent programs with why? questions. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 251–259, 2017.