A Natural Language Argumentation Interface for Explanation Generation in Markov Decision Processes ⋆

Thomas Dodson, Nicholas Mattei, and Judy Goldsmith

University of Kentucky
Department of Computer Science
Lexington, KY 40506, USA
tcdodson@gmail.com, nick.mattei@uky.edu, goldsmit@cs.uky.edu

Abstract. A Markov Decision Process (MDP) policy presents, for each state, an action, which preferably maximizes the expected reward accrual over time. In this paper, we present a novel system that generates, in real time, natural language explanations of the optimal action, recommended by an MDP while the user interacts with the MDP policy. We rely on natural language explanations in order to build trust between the user and the explanation system, leveraging existing research in psychology in order to generate salient explanations for the end user. Our explanation system is designed for portability between domains and uses a combination of domain specific and domain independent techniques. The system automatically extracts implicit knowledge from an MDP model and accompanying policy. This policy-based explanation system can be ported between applications without additional effort by knowledge engineers or model builders. Our system separates domain-specific data from the explanation logic, allowing for a robust system capable of incremental upgrades. Domain-specific explanations are generated through case-based explanation techniques specific to the domain and a knowledge base of concept mappings for our natural language model.

1 Introduction

A Markov decision process (MDP) is a mathematical formalism which allows for long range planning in probabilistic environments [2, 15]. The work reported here uses fully observable, factored MDPs [3]. The fundamental concepts use by our system are generalizable to other MDP formalisms; we choose the factored MDP representation as it will allow us to expand our system to scenarios where we recommend a set of actions per time step. A policy for an MDP is a mapping of states to actions that defines a tree of possible futures, each with a probability and a utility. Unfortunately, this branching set of possible futures is a large object with many potential branches that is difficult to understand even for sophisticated users.

The complex nature of possible futures and their probabilities prevents many end users from trusting, understanding, and implementing the plans generated from MDP policies [9]. Recommendations and plans generated by computers are not always trusted

⋆ This work is supported by NSF EAGER grant CCF-1049360.

or implemented by end users of decision support systems. Distrust and misunderstanding are two of the most often user cited reasons for not following a recommended plan or action [13]. For a user unfamiliar with stochastic planning, the most troublesome part of existing explanation systems is the explicit use of probabilities, as humans are demonstrably bad at reasoning with probabilities [18]. Additionally, it is our intuition that the concept of a preordained probability of success or failure at a given endeavor discomforts the average user.

Following the classifications of logical arguments and explanations given by Moore and Parker, our system generates arguments [11]. While we, as system designers, are convinced of the optimality of the optimal action the user may not be so convinced. In an explanation, two parties agree about the truth of a statement and the discussion is centered around *why* the statement is true. However, our system design is attempting to convince the user of the "goodness" of the recommended action; this is an argument.

In this paper we present an explanation system for MDP policies. Our system produces natural language explanations, generated from domain specific and domain independent information, to convince end users to implement the recommended actions. Our system generates arguments that are designed to convince the user of the "goodness" of the recommended action. While the logic of our arguments is generated in a domain independent way, there are domain specific data sources included. These are decoupled from the explanation interface, to allow a high degree of customization. This allows our base system to be deployed on different domains without additional information from the model designers. If an implementation calls for it, our system is flexible enough to incorporate domain specific language and cases to augment its generated arguments. We implement this novel, argument based approach with natural language text in order to closely connect with the user. Building this trust is essential in convincing the user to implement the policy set out by the MDP [13]. Thus, we avoid exposing the user to the specifics of stochastic planning, though we cannot entirely avoid language addressing the inherent probabilistic nature of our planning system.

Our system has been developed as a piece of a larger program working with advising college students about what courses to take and when to take them. It was tested on a subset of a model developed to predict student grades based on anonymized student records, as well as capture student preferences, and institutional constraints at the University of Kentucky [7]. Our system presents, as a paragraph, an argument as to why a student should take a specified set of courses in the next semester. The underlying policy is based on the student's preferences and abilities. This domain is interesting because it involves users who need to reason in discrete time steps about their long term benefits. Beginning students[1] at a university will have limited knowledge about utility theory and represent a good focus population for studying the effectiveness of different explanations.

Model construction, verification and validation is an extremely rich subject that we do not treat in this paper. While the quality of explanations is dependent on the quality and accuracy of a given model we will not discuss modeling accuracy or fidelity.

[1] Students may begin their college careers as Computer Science majors or switch into the major later. We consider students to *begin* with the introductory programming courses, or with the first CS course they take at the University of Kentucky.

The purpose of this work is to generate arguments in a domain-independent way, incorporating domain-specific information only to generate the explanation language. The correctness of the model is therefore irrelevant in the context of validating a method to generate explanations. Through user testing and refinement it is possible to use our work to assist in the construction, verification, and validation of models meant to be implemented with end users.

In the next section we will provide background on MDPs and a brief overview of current explanation systems. In Section 3 we define the model we use as an example domain. Section 4 provides an overview of the system design as well as specific details about the system's three main components: the model based explainer, the case based explainer, and the natural language generator. Section 5 provides examples of the output of our system and an overview of the user study we will use to verify and validate our approach. Section 6 provides some conclusions about the system development so far and our main target areas for future study.

2 Background and Related Work

Markov Decision Processes A MDP is a formal model for planning, when actions are modeled as having probabilistic outcomes. We focus here on factored MDPs [3]. MDPs are used in many areas, including robotics, economics and manufacturing.

Definition 1. *An MDP is a tuple, $\langle S,A,T,R\rangle$, where S is a set of states and A is a set of actions, and $T(s'|s,a)$ is the probability that state s' is reached if a is taken in state s, and $R(s)$ is the reward, or utility, of being in state s. If states in S are represented by variable (attribute) vector, we say that the MDP is* factored.

A policy for an MDP is a mapping $\pi : S \rightarrow A$. The best policy for an MDP is one that maximizes the expected value (Definition 2) [15] within a specified finite or infinite time horizon, or with a guarantee of (unspecified) finiteness. In the case of academic advising, since credits become invalid at the University of Kentucky after 10 years, we assume a fixed, finite horizon [2]. Policies are computed with respect to the expected total discounted reward, where the discount rate γ is such that $0 \leq \gamma < 1$. The optimal policy with respect to discount γ is one that maximizes the total discounted expected value of the start state (see Definition 2) [2, 15].

Definition 2. *The expected value of state s with respect to policy π and discount γ is*

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} T(s' \mid \pi(s), s) * V^{\pi}(s'). \tag{1}$$

The optimal value function V^* is the value function of any optimal policy π^* [2, 15]. We use the optimal policy, and other domain and model information, to generate natural language explanations for users with no knowledge of probability or utility theory.

Explanation Systems Prior work on natural language explanation of MDP policies is sparse, and has focused primarily on what could be called "policy-based explanation," whereby the explanation text is generated solely from the policy. The nature of such systems limits the usefulness of these explanations for users who are unfamiliar with stochastic planning, as the information presented is probabilistic in nature. However, these algorithms have the advantage of being entirely domain-independent. A good example of such a system is Khan et al.'s minimal sufficient explanations [9], which chooses explanatory variables based on the occupation frequency of desired future states. Note that, while the algorithms used in policy-based explanation systems are domain-independent, the explanations generated by such systems often rely on the implicit domain-specific information encoded into the model in the form of action and variable names. Other work has focused on finding the variable which is most influential to determining the optimal action at the current state [5], while using an extensive knowledge-base to translate these results into natural language explanations.

Case-based and model-based explanation systems rely, to different extents, on domain specific information. To find literature on such systems, it is necessary to look beyond stochastic planning. Case-based explanation, which uses a database of prior decisions and their factors, called a case base, is more knowledge-light, requiring only the cases themselves and a model detailing how the factors of a case can be generalized to arbitrary cases. Care must be taken in constructing a case base in order to include sufficient cases to cover all possible inputs. Nugent et al.'s KLEF [14] is an example of a case-based explanation system. A model-based explanation system, however, relies on domain-specific information, in the form of an explicit explanation model.

An explanation interface provides explanations of the reasoning that led to the recommendation. Sinha and Swearingen [17] found that, to satisfy most users, recommendation software employing collaborative filtering must be *transparent*, i.e., must provide not only good recommendations, but also the logic behind a particular recommendation. Since stochastic planning methods are generally not well understood by our intended users, we do not restrict our explanations to cover, for example, some minimum portion of the total reward [9], and instead choose explanation primitives that, while still factual, will be most convincing to the user.

3 Model

For this paper we focus on an academic advising domain. We use a restricted domain for testing which focuses on completing courses to achieve a computer science minor focus at the University of Kentucky. Our research group is also developing a system to automatically generate complete academic advising domains that capture all classes in a university [7]. The long term goal of this ongoing research project is to develop an end-to-end system to aid academic advisors that build probabilistic grade predictors, model student preferences, plan, and explain the offered recommendations.

The variables in our factored domain are the required courses for a minor focus in computer science: Intro Computer Programming (ICP), Program Design and Problem Solving (PDPS), Software Engineering (SE), Discrete Mathematics (DM), and Algorithm Design and Analysis (ALGO). We include Calculus II (CALC2) as a predictor

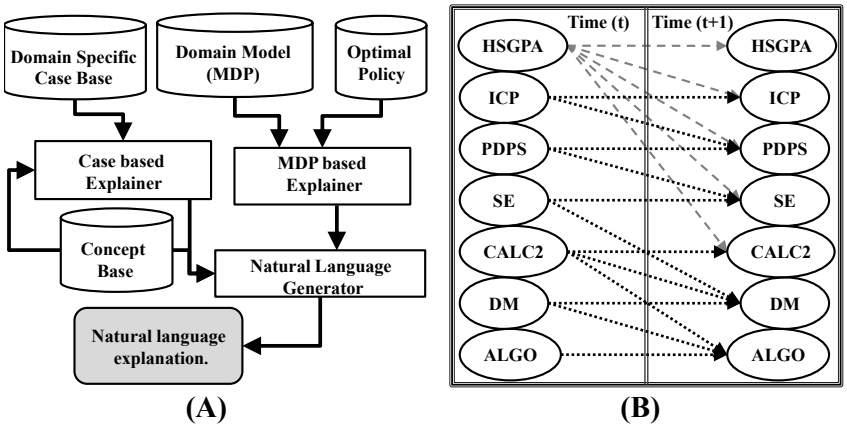

Fig. 1. System organization and data flow (A) and the dynamic decision network (temporal dependency structure) for the academic advising model (B).

course for DM and ALGO due to their strong mathematical components. Each class variable can have values: (G)ood, (P)ass, (F)ail, and (N)ot Taken. An additional variable is high school grade point average, HSGPA; this can have values: (G)ood, (P)ass, (L)ow. The model was hand coded with transition probabilities derived from historic course data at the University of Kentucky.

Each action in our domain is of the form, "Take Course X," and only affects variable X. Figure 1-B shows the temporal dependencies between classes, and implicitly encodes the set of prerequisites due to the near certain probability of failure if prerequisite courses are not taken first. Complex conditional dependences exist between courses due to the possibility of failing a course. CALC2 is not required and we do not place reward on its completion. Taking it correlates with success in DM and ALGO; we want to ensure our model can explain situations where unrewarded variables are important. Most courses in the model have HSGPA, the previous class, and the current class as the priors (except ICP and CALC2 which only have HSGPA as a prior). [2]

The reward function is additive and places a value of 4.0 and 2.0 on Good and Passing grades respectively. Failure is penalized with a 0.0. A discount factor of 0.9 is used to weight early success more than later success. While our current utility function only focuses on earning the highest grades possible as quickly as possible we stress that other utility functions could be used and, in fact, are being developed as part of our larger academic advising research project.

The model was encoded using a variant of the SPUDD format [8] and the optimal policy was found using a local SPUDD implementation developed in our lab [8, 10]. We applied a horizon of 10 steps and a tolerance of 0.01. The model has about 2,400 states and the optimal value function ADD has over 10,000 leaf nodes and 15,000 edges.

[2] HSGPA is a strong predictor of early college success (and college graduation) and GPA's prediction power has been well studied [4].

4 System Overview

Our explanation system integrates a policy-based approach with case-based and model-based algorithms. However, the model-based system is constructed so the algorithm itself is not domain-specific. Rather, the explanation model is constructed from the MDP and resulting policy and relies on domain-specific inputs and a domain-specific language, in the natural language generation module. Thus, we separate the model dependent factors from the model independent methods. This gives our methods high portability between domains.

Figure 1-A illustrates the data flow through our system. All domain specific information has been removed from the individual modules. We think of each of the modules as generating points of our argument while the natural language generator assimilates all these points into a well structured argument to the user. The assimilated argument is stronger than any of the individual points. However, we can remove modules that are not necessary for specific domains, e.g., when a case base cannot be procured. This allows our system to be flexible with respect to a single model and across multiple domains. In addition, system deployment can happen early in a development cycle while other "points" of the argument are brought online. The novel combination of a case-based explainer, which makes arguments from empirical past data, with a model-based explainer, which makes arguments from future predicted data, allows our system to generate better arguments than either piece alone.

A standard use case for our system would proceed as follows: students would access the interface either online or in an advising office. The system would elicit user preferences and course histories (these could also be gleaned from student transcripts). Once this data has been provided to the system, a natural language explanation would explain what courses to take in the coming semester. While our current model recommends one course at a time we will expand the system to include multiple actions per time step.

Our system differs from existing but similar systems such as the one designed by Elizalde et al. [5] in several important ways. First, while an extensive knowledge base will improve the effectiveness of explanations, the knowledge base required by our system to generate basic explanations is minimal, and limited to variables which can be determined from the model itself. Second, our model-based module decomposes recommendations from the MDP in a way that is more psychologically grounded in many domains, focusing on user actions instead of variables [6].

We designed with a "most convincing" heuristic; we attempt to select the factual statements and word framings that will be most influential to our target user base. This is in contrast to existing other similar systems which focus on a "most coverage" heuristic [9]. A most coverage heuristic focuses on explaining some minimal level of utility that would be accrued by the optimal policy. While this method is both mathematically grounded and convincing to individuals who understand probabilistic planning, our intuition is that it is not as convincing to the average individual.

4.1 Model Based Explanation

The model-based module extracts information from the MDP model and a policy of recommended actions on that model. This module generates explanations based on what

comes next — specifically, information about why, in terms of next actions, the recommended action is best. We compare actions in terms of a set of values, called *action factored differential values (AFDVs)* for each possible action in the current state. AFDVs allow us to explain the optimal action in terms of how much better the set of actions at the next state are. E.g., we can model that taking ICP before PDPS is better because taking ICP first improves the expected value of taking PDPS in the next step. We can also highlight how the current action can affect multiple future actions and rewards. This allows our method to explain complex conditional policies without explicit knowledge of the particular conditional. Through the computation of the AFDVs we are able to extract how the current best action improves the expected assignment of one or more variables under future actions.

This method of explanation allows for a salient explanation that focuses on how the current best action will improve actions and immediate rewards in the next state (the next decision point). Many studies have shown empirically that humans use a hyperbolic discounting function and are incredibly risk adverse when reasoning about long term plans under uncertain conditions [6,20]. This discount function places much more value on rewards realized in the short term. In contrast to human reasoning, an MDP uses an exponential discount function when computing optimal policies. The combined effects of human inability to think rationally in probabilistic terms and hyperbolic cognitive discounting means there is a fundamental disconnect between the human user and the rational policy [6, 18]. The disconnect between the two reasoning methods must be reconciled in order to communicate MDP policies to human users in terms that they will more readily understand and trust. This translation is achieved through explaining the long term plan in terms of short term gains with AFDV sets.

To generate a usable set of AFDVs from some state s, we define a method for measuring the value of taking an arbitrary two action sequence and then continuing to follow the given policy, π. Intuitively, a set of AFDVs is a set of two-step look ahead utilities for all the different possible combinations of actions and results. This is accomplished by modifying the general expression for V^{π} to accommodate deviation from the policy in the current state and the set of next states:

$$V_2^{\pi}(s,a_1,a_2) - R(s) = \gamma \sum_{s' \in S} T(s'|s,a_1) \cdot [R(s') + \gamma \sum_{s'' \in S} T(s''|s',a_2) \cdot V^{\pi}(s'')]. \quad (2)$$

Using V_2^{π}, we can then compute a single AFDV object for the action to be explained, $\pi(s)$, by computing the value of the two step sequence $\{\pi(s), a\}$ and the value of another two step sequence $\{a_i, a\}$ and taking the difference,

$$\Delta^{\pi}(s,\pi,a_i,a) = V_2^{\pi}(s,\pi(s),a) - V_2^{\pi}(s,a_i,a). \quad (3)$$

To compute a full set of AFDVs for the explanation action, $\pi(s)$, this computation is done for all $a_i \in A \setminus \pi(s)$ and for all $a \in A$.

In order to choose variables for explanation, we compute, for each i, $\Delta^{\pi}(s,\pi,a_i,a)$, to find out how many actions' utilities will increase after having taken the recommended action. This set of counts gives the number of actions in the current state which cause a greater increase in utility of the action a than the recommended action. We define

$$x_s^{\pi}(a) = |\{i : \Delta^{\pi}(s,\pi,a_i,a) < 0\}|. \quad (4)$$

Note that we may have for all $a \in A : x_s^\pi(a) > 0$, since only the sum of the AFDV set over a_i for the optimal action is guaranteed to be greater than or equal to the sum for any other action. We choose the subset of A for which $x_s^\pi(a)$ is minimal as our explanation variables, and explain $\pi(s)$ in terms of its positive effects on those actions. We can also decompose the actions into corresponding variable assignments and explain how those variables change, leading to higher reward. By focusing on actions we reduce the overall size of the explanation in order to avoid overwhelming the user, while still allowing the most salient variables of the recommended action to be preserved. If more variables are desired, another subset of A can be chosen for which $x_s^\pi(a)$ is greater than the minimum, but less than any other value. While the current method of choosing explanation variables relies on knowledge of the *optimal* policy, the AFDV objects are meaningful for any policy. However, our particular method for choosing the subset of AFDVs for explanation relies on the optimality of the action $\pi(s)$, and would have to be adapted for use with a heuristic policy.

For example, the explanation primitive for a set of future actions with $\pi(s) = act\_PDPS$, $x_s^\pi(act\_SE) = x_s^\pi(act\_DM) = 0$, $x_s^\pi(act\_ALGO) = 1$, and $x_s^\pi(a) = 2$ for all other a is:

> The recommended action is *act_PDPS*, generated by examining long-term future reward. It is the optimal action with regards to your current state and the actions available to you. Our model indicates that this action will best prepare you for *act_SE* and *act_DM* in the future. Additionally, it will prepare you for *act_ALGO*.

It is possible to construct pathological domains where our domain independent explainer fails to select a best action. In these rare cases, the explainer will default to stating that the action prescribed by the given policy is the best because it leads to the greatest expected reward; this prevents contradictions between the explanation and policy. The AFDV method will break down if domains are constructed such that the expected reward is 0 within the horizon (2 time steps). This can happen when there are balanced positive and negative rewards. For this reason, we currently restrict our domain independence claims to those domains with only non-negative rewards.

4.2 Case-Based Explanation

Case-based explanation (CBE) uses past performance in the same domain in order to explain conclusions at the present state. It is advantageous because it uses real evidence, which enhances the transparency of the explanation, and analogy, a natural form of explanation in many domains [14]. This argument from past data combined with our model-based argument from predicted future outcomes creates a strong complete argument for the action recommended by the optimal policy. Our case base consists of 2693 distinct grade assignments in 6 distinct courses taken by 955 unique students. This anonymized information was provided by the University of Kentucky, about all courses taken by students who began their academic tenure between 2001 and 2004.

In a typical CBE system, such as KLEF [14], *a fortiori* argumentation is used in the presentation of individual cases. This presents evidence of a strong claim in order to support a weaker claim. In terms of academic achievement, one could argue that if there

is a case of a student receiving a "Fair" in PDPS and a "Good" in SE, then a student who has received a "Good" in PDPS should expect to do at least as well.

In our system, a single case takes the form of: $scenario1 \rightarrow action \rightarrow scenario2$, where a scenario is a partial assignment of state variables, and $scenario2$ occurs immediately after action, which occurs at any time after $scenario1$. In particular, we treat a single state variable assignment, followed by an action, followed by an assignment to single state variable, usually differing from the first, as a single case. For example, a student having received an A in ICP and a B in PDPS in a later semester comprises a single case with $scenario1 = \{var\_ICP = A\} \rightarrow action = take\_PDPS \rightarrow scenario2 = \{var\_PDPS = B\}$. If the same student had also taken CALC2 after having taken ICP, that would be considered a distinct case.

In general, the number of state variables used to specify a case depends on the method in which the case base is used. Two such methods of using a case base are possible: case aggregation and case matching [1]. When using case aggregation, which is better suited to smaller scenarios, the system combines all matching cases into relevant statistics in order to generate arguments. For example, case aggregation in our system would report statistics on groups of students who have taken similar courses to the current student and explain the system recommendation using the success or failure of these groups of students. When using case matching, a small number of cases, whose scenarios match the current state closely, would be selected to generate arguments [14]. Case matching methods are more suited to larger scenarios, and ideally use full state assignments [1]. For example, case matching in our system would show the user one or two students who have identical or nearly identical transcripts and explain the system recommendation using the selected students' transcripts.

Our system uses a case aggregation method, as our database does not have the required depth of coverage of our state-space. There are some states which can be reached by our MDP which have few or no cases. With a larger case base, greater specificity in argumentation is possible by considering an individual case to be the entirety of a single student's academic career. However, presenting individual cases still requires that the case base be carefully pruned to generate relevant explanations. Our system instead presents explanations based on dynamically generated statistics over all relevant cases (i.e., assignments of the variables affected by the recommended action). We select the relevant cases and compute the likelihood of a more rewarding variable assignment under a given action. This method allows more freedom to chose the action for which we present aggregated statistics; the system can pick the most convincing statistics from the set of all previous user actions instead of attempting to match individual cases.

Our method accomplishes this selection in a domain-independent way using the ordered variable assignments stored in the concept base. We use a separate configuration file, called a concept base, to store any domain specific information. We separate this data from the explanation system in order to maintain domain independence. In our system, there is a single required component of the concept base which must be defined by the system implementer; an ordering in terms of reward value over the assignments for each variable, with an extra marker for a valueless assignment that allows us to easily generate meaningful and compelling case-based explanations. The mapping could also be computed from the model on start-up, but explicitly enumerating the ordering in

the concept base allows the system designer to tweak the case-based explanations in response to user preferences by reordering the values and repositioning the zero-value marker.

For a given state, s, for each variable v_i affected by $\pi(s)$, we consider the naïve distribution, $\phi(v_i)$, over the values of v_i from cases in the database. We compute the conditional distribution, $\phi(v_i|s)$, over the values of v_i given the values to all other variables in s. Then, for each conditional distribution, we examine the probability of a rewarding assignment. We then sort the distributions in order from most rewarding to least, by comparing each one to the probability of receiving the assignment from any of the naïve distributions. Conditional distributions which have increased probability of rewarding assignments over the naïve distributions are then chosen to be used for explanation.

For a student in a state such that $var\_ICP = Good$, $var\_CALC2 = Good$, and $\pi(s_e) = act\_PDPS$: since act_PDPS influences only var_PDPS, three grade distributions will be generated over its values: one distribution for all pairs with $var\_ICP = Good$, one with $var\_CALC2 = Good$, and one over all cases which have some assignment for var_PDPS. If, in the case base, 200 students had $var\_ICP = Good$ and $var\_PDPS \neq NotTaken$ with 130 "Good" assignments, 40 "Fair", and 30 "Poor", giving a [0.65, 0.20, 0.15] distribution; 150 students had $var\_CALC2 = Good$ and $var\_PDPS \neq NotTaken$ with 100 "Good", 30 "Fair", and 20 "Poor", giving a [0.67, 0.20, 0.13] distribution; while 650 students had $var\_PDPS \neq NotTaken$ with 300 "Good", 250 "Fair", and 100 "Poor", giving a [0.47, 0.38, 0.15] distribution, then the distributions indicate that such assignments increase the probability of receiving $var\_PDPS = Good$, and the generated explanation primitive is:

> Our database indicates that with either $var\_ICP = Good$ or $var\_CALC2 = Good$, you are more likely to receive $var\_PDPS = Good$ in the future.

4.3 Natural Language Generator

In explanations generated by our system, particular emphasis is placed on displaying probabilities in terms that are more comfortable to the target user base, undergraduate students. A verbal scale has some inherent problems. In medical decision making, Witterman et al. found that experienced doctors were more confident using a verbal, rather than numeric, scale [21]. Unfortunately, Renooij [16] reports large variability of the numerical values assigned to verbal expressions between subjects. However, Renooij found that there was a high level of inter-subject consistency and intra-subject consistency over time, in the ordering of such verbal expressions. Additionally, numerical interpretations of ordered lists of verbal expressions were less variable than interpretations of randomly ordered lists [16]. Thus, our explanations replace numerical probabilities with a system of intuitively ordered adverb phrases: *very likely* ($p > 0.8$), *likely* ($p > 0.5$), *unlikely* ($p < 0.5$), and *very unlikely* ($p < 0.2$). Since words at the extremes of the scale are less likely to be misinterpreted, *nearly certain* ($p > 0.95$) and *nearly impossible* ($p < 0.05$) could also be added to the scale.

Though these cutoffs work well for expressing the probabilities of state changes predicated on some action in an MDP model, they are not well suited for expressing the probability of a particular variable assignment with some underlying distribution.

In this case, our system simply uses *less likely* and *more likely* for effects which cause the probability of the particular value to be less than or greater than the probability in the naïve distribution.

While MDP-based explanations can be generated in a domain-independent way, producing domain-independent natural language explanations is more problematic. The only domain semantics available from the MDP are the names of the actions, variables, and values. These labels, however, tend to be abbreviated or otherwise distorted to conform to technical limitations. Increasing the connection between the language and domain increases the user trust and relation to the system by communicating in language specific to the user [13, 17]. Our system uses a relatively simple concept base which provides mappings from variable names and assignments to noun phrases, and action names to verb phrases. This is an optional system component; the domain expert should be able to produce this semantic mapping when constructing the MDP model.

All of these mappings are stored in the concept base as optional components. The template arguments that are populated by the explanation primitives are also stored in the concept base. Each explanation module only computes the relations between variables. It is up to the interface designer to establish the mappings and exact wordings in the concept base. We allow for multiple templates and customizable text, based on state or variable assignment, to be stored in the concept base. This flexible component allows for as much or as little domain tailoring as is required by the application.

5 Discussion and Study Proposal

Our system successfully generates natural language explanations in real time using domain-independent methods, while incorporating domain specific language for the final explanation. The concept base allows designers to insert custom language as a preamble to any or all of the recommendations. This allows the user interface designer flexibility as to how much domain, modeling, and computational information to reveal to the end user.

The runtime complexity of our system, to generate an explanation for a given state, is $\mathcal{O}(n^2)$ where n is the number of actions in the MDP model. Almost all the computational burden is experienced when computing the AFDVs. These could, for very large domains, be precomputed and stored in a database if necessary. This complexity is similar to the computational requirements imposed by other MDP explanation systems [9] and is easily within the abilities of most modern systems for domains with several thousand states.

Our concept base includes text stating that recommendations depend on grades (outcomes) the student has received previously, and on the user's preferences. In many applications we expect that users do not want to know how every decision in the system is made; we are building convincing arguments for a general population, not computer scientists. While technically inclined people may want more information regarding the model construction and planning, it is our feeling that most users want to understand what they should do now. Thus, our example explanation does not explain or exhibit the entire policy. The important concept for our end users is not the mathematical structure of a policy, but that future advice will depend on current outcomes. After language substitution, the generated explanations look like:

> The recommended action is taking Introduction to Program Design and Problem Solving, generated by examining possible future courses. It is the optimal course with regards to your current grades and the courses available to you. Our model indicates that this action will best prepare you for taking Introduction to Software Engineering and taking Discrete Mathematics in the future. Additionally, it will prepare you for taking Algorithm Design and Analysis. Our database indicates that with either a grade of A or B in Introductory Computer Programming or a grade of A or B in Calculus II, you are more likely to receive a grade of A or B in Introduction to Program Design and Problem Solving, the recommended course.

This form of explanation offers the advantage of using multiple approaches. The first statement explains the process of generating an MPD policy, enhancing the transparency of the recommendation in order to gain the trust of the user [17]. It makes clear that the planning software is considering the long-term future, which may inspire confidence in the tool. The second statement relies solely on the optimal policy and MDP model. It offers data about expected future performance in terms of the improvement in value of possible future actions, the AFDVs. The AFDVs are computed using an optimal policy. That means the policy maximizes expected, long term reward. This part of the explanation focuses on the near future to explain actions which may only be preferable because of far future consequences. The shift in focuses leverages the users inherent bias towards hyperbolic discounting of future rewards [6]. The last statement focuses on the student's past performance in order to predict performance at the current time step and explains that performance in terms of variable assignments. This paragraph makes an analogy between the user's performance and the aggregated performance of past students. Argument from analogy is very relevant to our domain — academic advisors often suggest, for example, that advisees talk to students who have taken the course from a particular professor. Additionally, the case-based explanation module can be adapted to take into account user preferences, and therefore make more precise analogies.

User Study We have recently received institutional approval for a large, multi-staged user study. We informally piloted the system with computer science students at our university, but this informal test fails to address the real issues surrounding user interfaces. Our study will use students from disciplines including psychology, computer science, and electrical engineering, and advisors from these disciplines. We will compare the advice generated by our system and its "most convincing" approach to other systems which use a "most coverage" (with respect to rewards) approach. We will survey both students and advisors to find what, if any, difference exists between these two approaches. We will also test differences in framing advice in positive and negative lights. There is extensive literature about the effects of goal framing on choice and we hope to leverage this idea to make our recommendations more convincing [19].

By approaching a user study from both the experts' and users' viewpoints we will learn about what makes good advice in this domain and what makes convincing arguments in many more domains. A full treatment of this study, including pilot study, methodology, instrument development, and data analysis will fill another complete paper. We did not want to present a token user study. Quality evaluation methods must

become the standard for, and not the exception to, systems that interact with non-expert users such as the one developed here.

6 Conclusion and Future Work

In this work we have presented a system and design which generates natural language explanations for actions generated by MDPs. This system uses a novel mix of case-based and model-based techniques to generate highly salient explanations. The system design abstracts the domain dependent knowledge from the explanation system, allowing it to be ported to other domains with minimal work by the domain expert. The generated explanations are grounded both psychologically and mathematically for maximum impact, clarity, and correctness. The system operates in real time and is scalable based on the amount of domain specific information available.

Automatic planning and scheduling tools generate recommendations that are often not followed by end users. As computer recommendations integrate deeper into everyday life it becomes imperative that we, as computer scientists, understand why and how users implement recommendations generated by our systems. The framework here starts to bridge the gap between mathematical fundamentals and user expectations.

Our current model recommends one course at a time. We will be expanding the system to include multiple actions per time step. This requires a planner that can handle factored actions, and requires that we adjust the explanation interface. We expect that explanations will consist of three parts, not necessarily all present in each response. The first will answer the question, "Why this particular course/atomic action?" The second will answer, "Why these two/few courses/atomic actions together?" And the third will look at the entire set. Answers to the first type of query will be very similar to what is described here, but will take into account whether the effects are on simultaneous or future courses. Answers to the second type will build directly on the information generated to answer the first type. We expect that answers to "Why this set of courses" will depend on the constraints given on sets of courses/atomic actions, such as "You are only allowed to take 21 credits per semester, and your transcript indicates that you/people with records like yours do best with about 15 per semester."

Our model based module extracts information from the MDP model and a policy of recommended actions on that model. Finding optimal policies for factored MDPs is PSPACE-hard [12]. We assumed, in the development of this system, that the optimal policy is available. Given a heuristic policy, our system will generate consistent explanations, but they will not necessarily be as convincing. We would like to extend our work and improve the argument interface when only heuristic policies are available.

Acknowledgements

This work is partially supported by NSF EAGER grant CCF-1049360. We would like to thank the members of the UK-AILab, especially Robert Crawford, Joshua Guerin, Daniel Michler, and Matthew Spradling for their support and helpful discussions. We are also grateful to the anonymous reviewers who have made many helpful recommendations for the improvement of this paper.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI communications 7(1), 39–59 (1994)
2. Bellman, R.: Dynamic Programming. Princeton University Press (1957)
3. Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intellgence Research 11, 1–94 (1999)
4. Camara, W.J., Echternacht, G.: The SAT I and high school grades: utility in predicting success in college. RN-10, College Entrance Examination Board, New York (2000)
5. Elizalde, F., Sucar, E., Noguez, J., Reyes, A.: Generating explanations based on Markov decision processes. In: Mexican International Conf. on Artificial Intelligence. pp. 51–62 (2009)
6. Frederick, S., Loewenstein, G., O'Donoghue, T.: Time discounting and time preference: A critical review. Journal of Economic Literature 40, 351–401 (2002)
7. Guerin, J.T., Crawford, R., Goldsmith, J.: Constructing dynamic bayes nets using recommendation techniques from collaborative filtering. Tech report, University of Kentucky (2010)
8. Hoey, J., St-Aubin, R., Hu, A., Boutilier, C.: SPUDD: Stochastic planning using decision diagrams. In: Proc. UAI. pp. 279–288 (1999)
9. Khan, O., Poupart, P., Black, J.: Minimal sufficient explanations for factored Markov decision processes. In: Proc. ICAPS (2009)
10. Mathias, K., Williams, D., Cornett, A., Dekhtyar, A., Goldsmith, J.: Factored mdp elicitation and plan display. In: Proc. ISDN, AAAI (2006)
11. Moore, B., Parker, R.: Critical Thinking. McGraw-Hill (2008)
12. Mundhenk, M., Lusena, C., Goldsmith, J., Allender, E.: The complexity of finite-horizon Markov decision process problems. JACM 47(4), 681–720 (2000)
13. Murray, K., Häubl, G.: Interactive consumer decision aids. In: Wierenga, B. (ed.) Handbook of Marketing Decision Models, pp. 55–77. Springer (2008)
14. Nugent, C., Doyle, D., Cunningham, P.: Gaining insight through case-based explanation. JIIS 32, 267–295 (2009)
15. Puterman, M.: Markov Decision Processes. Wiley (1994)
16. Renooij, S.: Qualitative Approaches to Quantifying Probabilistic Networks. Ph.D. thesis, Institute for Information and Computing Sciences, Utrecht University, The Netherlands (2001)
17. Sinha, R., Swearingen, K.: The role of transparency in recommender systems. In: CHI '02 Conference Companion. pp. 830–831 (2002)
18. Tversky, A., Kahneman, D.: Judgement under uncertainty: Heuristics and biases. Science 185, 1124–1131 (1974)
19. Tversky, A., Kahneman, D.: Rational choice and the framing of decisions. The Journal of Business 59(4), 251–278 (1986)
20. Tversky, A., Kahneman, D.: Advances in prospect theory: Cumulative representation of uncertainty. Journal of Risk and uncertainty 5(4), 297–323 (1992)
21. Witteman, C., Renooij, S., Koele, P.: Medicine in words and numbers: A cross-sectional survey comparing probability assessment scales. BMC Med. Informatics and Decision Making 7(13) (2007)