An English-Language Argumentation Interface for Explanation Generation with Markov Decision Processes in the Domain of Academic Advising

THOMAS DODSON, University of Pennsylvania
NICHOLAS MATTEI, NICTA and University of New South Wales
JOSHUA T. GUERIN, University of Tennessee at Martin
JUDY GOLDSMITH, University of Kentucky

A Markov Decision Process (MDP) policy presents, for each state, an action, which preferably maximizes the expected utility accrual over time. In this article, we present a novel explanation system for MDP policies. The system interactively generates conversational English-language explanations of the actions suggested by an optimal policy, and does so in real time. 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. 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 MDP-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 used to generate English-language explanations.

Categories and Subject Descriptors: H.1.2 [**Models and Principles**]: User/Machine Systems—*Human factors*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Theory and methods*; H.4.2 [**Information Systems Applications**]: Types of Systems—*Decision support*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Natural language*

General Terms: Design, Algorithms, Experimentation, Human Factors

Additional Key Words and Phrases: Argumentation and explanation in decision support, Markov decision processes

This version supersedes an earlier version which appeared at the 2nd Algorithmic Decision Theory Conference [Dodson et al. 2011].
This work is supported by the National Science Foundation, under grants CCF-1049360 and ITR-0325063. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. NICTA is funded by the Australian Government through the Department of Broadband, Communications and the Digital Economy and the Australian Research Council (ARC) through the ICT Centre of Excellence program.
The majority of the work on this system was completed while the first three authors were attending the University of Kentucky.
Authors' addresses: T. Dodson, Department of Physics and Astronomy, University of Pennsylvania, Philadelphia, PA; N. Mattei, Neville Roach Laboratory, NICTA, Sydney, Australia; email: nsmattei@gmail.com; J. T. Guerin, Department of Computer Science, University of Tennessee at Martin, Martin, TN; J. Goldsmith, Computer Science Department, University of Kentucky, Lexington, KY.

DOI: http://dx.doi.org/10.1145/2513564

ACM Reference Format:
Dodson, T., Mattei, N., Guerin, J. T., and Goldsmith, J. 2013. An English-language argumentation interface for explanation generation with Markov decision processes in the domain of academic advising. *ACM Trans. Interact. Intell. Syst.* 3, 3, Article 18 (October 2013), 30 pages.
DOI: http://dx.doi.org/10.1145/2513564

1. INTRODUCTION

A Markov decision process (MDP) is a mathematical formalism which allows for long-term planning in probabilistic environments [Bellman 1957; Puterman 1994]. 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. "Utility" is sometimes called "reward" in the literature [Puterman 1994]; we choose to use the term utility in this article. In order for a user to *evaluate* the usefulness of a policy, they must take into account every possible sequence of states that could arise from following the optimal actions laid out by the policy, and the likelihoods and utilities of those sequences. Unfortunately, this 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 [Khan et al. 2011].

This is a known issue: recommendations and plans generated by computers are not always trusted 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 [Murray and Häubl 2008]. 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 [Tversky and Kahneman 1974]. Additionally, it is our intuition that the concept of a preordained probability of success or failure at a given endeavor discomfits the average user.

In this article we present an explanation system for MDP policies which attempts to address the issues inherent to evaluation of MDP policies by non-technical users. Specifically, in this work, we focus on advising undergraduate students at a large university on what courses to select in the coming semester(s). Our system produces explanations in conversational English, 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, according to the given utility function—which can be tuned to that user. 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 to 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 conversational English 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 [Murray and Häubl 2008]. 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.

Following the classifications of logical arguments and explanations given by Moore and Parker [2008], our system generates arguments. While we, as system designers, are convinced of the optimality of the optimal action, the user may not be. In

an explanation, however, two parties agree about the truth of a statement and the discussion is centered around *why* the statement is true. As our system design is attempting to convince the user of the "goodness" of the recommended action; this is an argument.

Our system has been developed as a piece of a larger research 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 [Guerin and Goldsmith 2011]. 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 can be tailored to 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 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 throughly in this article. We detail the model employed for the user studies conducted on our system in Section 3 but we do not provide a full description of how the model was constructed. While the quality of explanations is dependent on the quality and accuracy of a given model, we will not discuss modeling accuracy or fidelity in great detail. The purpose of this work is to generate arguments in a domain-independent way, incorporating domain-specific information only to generate the explanation language. We have taken care to construct a model that has basis in reality and we refer the reader to our other publications for a more complete treatment of the subject [Guerin and Goldsmith 2011].

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 English-language template populator. Section 6 provides examples of the output of our system and an overview of the user study used to verify and validate our approach. Section 7 provides some conclusions about the system development so far and our main target areas for future study.

2. BACKGROUND AND RELATED WORK

In our own department, we have seen excellent advising, but also many students who shun advice. It is common, in the latter group, that they arrive at what they hope to be their last semester with an impossible load of required courses. This might be impossible due to scheduling; not all classes are offered every semester, and some coincide. Or it might be impossible because they need too many credits, too many large projects, and too many concurrent courses that are best taken sequentially.

Based on many years of experience and observation in advising, we can say that long-term planning is crucial to student success. It is necessary that students take unpopular or challenging courses early in their careers in order to maintain sufficient progress towards their degrees. It is also important that students actually master the basics, possibly through repeating courses, before moving on to the more enticing topics. Thus, we argue that a long-term view is often necessary for students' academic success. Furthermore, that long-term view must take into account the possibilities of success and failure, and allow sufficient leeway for course repetition or direction change.

2.1. 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 [Boutilier et al. 1999]. MDPs are used in many areas, including robotics, economics and manufacturing.

Definition 2.1. 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 utility, or reward, of being in state s. If states in S are represented by variable (attribute) vectors, we say that the MDP is *factored*.

A policy for an MDP is a mapping $\pi : S \rightarrow A$. The optimal policy for an MDP is one that maximizes the expected value (Definition 2.2) [Puterman 1994] within a specified finite or infinite time horizon, or with a guarantee of (unspecified) finiteness [Hansen 2007]. In the case of academic advising, since credits become invalid at the University of Kentucky after 10 years, we assume a fixed, finite horizon [Bellman 1957]. Policies are computed with respect to the expected total discounted utility, 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 utility of the start state (see Definition 2.2) [Bellman 1957; Puterman 1994].

Definition 2.2. The expected value of state s with respect to policy π and discount γ is

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

The optimal value function V^* is the value function of any optimal policy π^* [Bellman 1957; Puterman 1994]. We use the optimal policy, and other domain and model information, to generate conversational English explanations for users with no knowledge of probability or utility theory.

MDPs have been successfully deployed in other domains. An MDP model, deployed to an online bookstore, was found to generate recommendations which outperformed commercially available collaborative filtering methods in successfully recommending items to users [Shani et al. 2005]. In another study, it was found that users of a vacation planning website were more inclined to follow recommendations generated by an MDP than a linear policy, resulting in a more efficient search session for the user [Mahmood et al. 2009]. MDP methods are, however, not without drawbacks. The costs for considering long-term expected utility, for building *policies* instead of plans, is complexity: computational complexity in computing those policies, and complexity in tracing the effects and benefits of recommended actions. There has been considerable work on the planning side of factored MDPs, which we will not survey here. What we address in this article are algorithms developed to reason about the effects and benefits of actions recommended by an MDP policy.

2.2. Other Planning Methods

There are myriad academic advising software packages available, with varying functionality. Some are degree audit programs, which tell students which requirements they have or have not met for their program. Some are constraint solvers that come up with schedules for one or more semesters. However, most software is simply a list of courses the student should take, either for a single semester, or for their career. Almost none of them look at the likelihood of success in particular courses.

There is a new trend that may look at statistical predictions of success, the so-called Netflix-style advising, at Austin Peay State University and elsewhere [Young 2010]. In

this model, collaborative filtering is used on prior student records to match students with others who had similar records, and to suggest courses that the others took. This can even be modified by taking into account the advisees' preferences [Ray and Sharma 2011].

None of these methods are comprehensive decision-theoretic planners. Few give long-term plans, much less policies. Even the collaborative-filtering methods appear to look at most-likely-to-succeed choices, rather than weighing the utility of courses by the likelihood of success.

Case-based planning methods maintain a library of plans, that is, static action sequences, which are retrieved based on some similarity metric. While case-based planning is, in some limited cases, computationally easier than other planning methods [Liberatore 2005], adapting the retrieved plans for a novel case requires the application of more complicated algorithms, such as artificial neural networks [Paz et al. 2009] or a mixture of domain-specific heuristics, constraint satisfaction, and intervention by a domain expert [Avesani et al. 2000].

2.3. Explanation in Recommender Systems

The bulk of the existing work regarding explanation of the outcomes of recommender systems is in the domain of e-commerce, which makes extensive use of collaborative filtering (CF) algorithms. Sinha and Swearingen [2002] found that to satisfy most users, recommender systems employing CF methods must provide not only good recommendations, but also the logic behind a particular recommendation. This is excessive in the case of MDP policies. The fully transparent explanation for the recommendation generated by an optimal MDP policy is that the action maximizes long-term expected utility, assuming correctness of the utility function for the domain. Herlocker [1999] proposes a "Data-Explorative" model of explanation, in which the recommendation is explained by highlighting key data. In the context of a policy, the only data available are the utility function used to generate that policy and the optimal value function which accompanies the policy. This motivates us to attempt to decompose the utility of the recommended action in some way that allows the user to more easily understand why a particular action is recommended at a given step. A truly data-explorative explanation model would then allow the user the freedom to explore "what-if" scenarios, moving step-by-step through their career. Indeed, as we will see in our user study (Section 6.1), Herlocker et al.'s model of explanation is a desirable feature according to the test users of our system. Therefore, our user study, to some degree, validates the work of Herlocker et al.

Because we are seeking a domain-independent method for generating explanations, and because CF is itself domain-independent, the classification of explanation systems for existing recommender systems provided by Tintarev and Masthoff [2007, 2011] is very instructive. As in the CF case, the challenge is to consider the tradeoffs between different and contradictory explanation goals [Tintarev and Masthoff 2011]. In the case of this particular system, *transparency*, which seeks to explain how the system works, is sacrificed in order to enhance *effectiveness* (helping users make good decisions) and *persuasiveness* (convincing users to accept the recommendation of the system). Such a system is more specifically termed an *argumentation system*, as a goal of persuasiveness implies that the parties involved are not in agreement regarding the optimality of the action in question [Moore and Parker 2008].

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 [Khan et al. 2011], which decomposes the utility of the recommended action in the current state in terms of the expected occupancy frequencies of desired future scenarios (partial assignments of state variables). Explanations are then presented as a set of sentence templates populated by action and variable names.

Definition 2.3. [Khan et al. 2009] The occupancy frequency of state s', starting in state s_0 and executing policy π is

$$\lambda^{\pi}_{s_0}(s') = \delta(s', s_0) + \gamma \sum_{s \in S} T(s' \mid \pi(s), s) \cdot \lambda^{\pi}_{s_0}(s), \tag{2}$$

where

$$\delta(s_1, s_2) = \begin{cases} 1, & s_1 = s_2 \\ 0, & s_1 \neq s_2 \end{cases}$$

Definition 2.4. [Khan et al. 2009] The occupancy frequency of scenario *sc* starting in state s_0 is

$$\lambda^{\pi}_{s_0}(sc) = \sum_{s \in sc} \lambda^{\pi}_{s_0}(s). \tag{3}$$

An example of an explanation of this form, in which only one rewarding scenario need be considered is, taken verbatim from [Khan et al. 2009]:

> Action TakeCS343 & CS448 is the best action because: It is likely to take you to CoursesCompleted = 6, TermNumber = Final about 0.86 times, which is as high as any other action.

Each rewarding scenario corresponds to an explanation template of a similar form, which can be thought of as representing part of the expected value of taking the recommended action from the current state, that is, the action-value function $Q(s, a)$ [Sutton and Barto 1998]. While this particular explanation consists of only a single template, in general the full explanation for $\pi^*(s)$ consists of the minimum number of templates required to explain some portion of the utility, V^{MSE}, such that $Q(s, \pi^*(s)) \geq V^{MSE} > Q(s, a), \quad \forall a \neq \pi^*(s)$ [Khan et al. 2011]. We refer to an approach such as this, which focuses on explaining some suitable portion of the utility function, as a "most coverage" approach.

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. While this particular explanation could be improved by substitution of more accessible phrasing in place of variable names and occupancy frequencies, the system designers chose to publish explanations of the form presented earlier.

Other work has focused on finding the variable which is most influential to determining the optimal action at the current state [Elizalde et al. 2009], 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

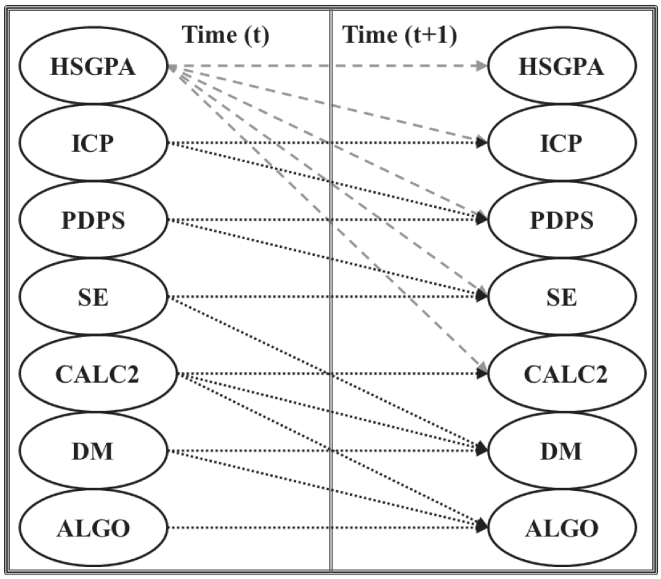

Fig. 1. The dynamic decision network (temporal dependency structure) for the academic advising model. Course grades are influenced by the current grade for that course (if any) and grades for courses that have an arc from t to $t + 1$.

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 [Nugent et al. 2009] 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 [Brüninghaus and Ashley 2003].

3. MODEL

For this article we focus on an academic advising domain. For the purposes of testing our methods, we use restricted domains which focus on completing courses to achieve computer science and psychology minors at the University of Kentucky. Note that while both models were tested in the user study (Section 6), we will use the computer science model as the example model when detailing our methods. Our research group is also developing a system to automatically generate complete academic advising domains that capture all classes in a university [Guerin and Goldsmith 2011]. The long term goal of this ongoing research project is to develop an end-to-end system to aid academic advisors that builds probabilistic grade predictors (e.g., Anthony and Raney [2012]), models student preferences, plans, and explains 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 course for DM and ALGO due to their strong mathematical components. Each class variable can have values: (G)ood - corresponding to A or B, (P)ass - corresponding to C, (F)ail - corresponding to D or F, 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 directly affects variable X. Figure 1 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 utility on its completion. Taking it correlates with success in DM and ALGO; we want to ensure our model can explain situations where variables with no utility are important. Most courses in the model have high school GPA (HSGPA), the previous class, and the current class as the priors (except ICP and CALC2, which only have HSGPA as a prior).[1]

The utility function is additive and places a higher utility value on earning higher grades. The model places a value of 4.0 and 2.0 on Good and Passing grades respectively, while 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 (where quickness is enforced with a relatively strong discount factor of 0.9), we stress that our overall system does not depend on this choice of the utility function. Any additive utility function can be expressed within the underlying MDP and this utility function is used for both policies and explanations. Other utility functions are, in fact, being developed as part of our larger academic advising research project. We hope to be able to eventually capture student preferences over things like the time of day a course occurs and current-semester course loading as well as outcome based utilities such as content mastery and GPA in terms of an additive utility function within our MDP. Our particular utility function simply represents what we believe, based on experiences with students and results from our user study, to be a "least common denominator" of preference among entry-level students. Specifically, it represents a preference for graduation as soon as possible, with a higher GPA being preferable to a lower GPA. We find that most (but not all) students share this baseline preference and therefore, we use it for our testing.

Our particular treatment of utility makes a number of (possibly naïve) simplifying assumptions. We understand that a more complete treatment of student utility is a requirement for actual deployment of a decision-theoretic advising system. Properly integrating trade-offs between time to graduate, courses taken (or skills mastered), and grades earned is one dimension of such a treatment. Along with our investigation of explanations for decision-theoretic planning, acquisition and modeling of more complete models of utility and state transition are being developed as a part of the larger goal of our advising project (e.g., [Guerin and Goldsmith 2011; Guerin et al. 2012]). We discuss some of these related issues in Section 6.1, however we believe that a more thorough and formal treatment of utility is outside the scope of this article.

Transition probabilities for the model were generated by looking at data from many years of computer science or psychology courses, respectively. This past grade assignment data was converted into a transition model (probability of passing any given course given grade assignments in previous courses) for all possible combinations [Guerin and Goldsmith 2011]. The model was encoded using a variant of the SPUDD format [Hoey et al. 1999] and the optimal policy was found using a local SPUDD implementation developed in our lab [Hoey et al. 1999; Mathias et al. 2006]. We applied a horizon of 10 steps (since credits disappear after about 10 semester) during planning. The model has about 2,400 states (all possible combinations of course grades) and the

[1]High school GPA is a strong predictor of early college success (and college graduation) [Camara and Echternacht 2000].

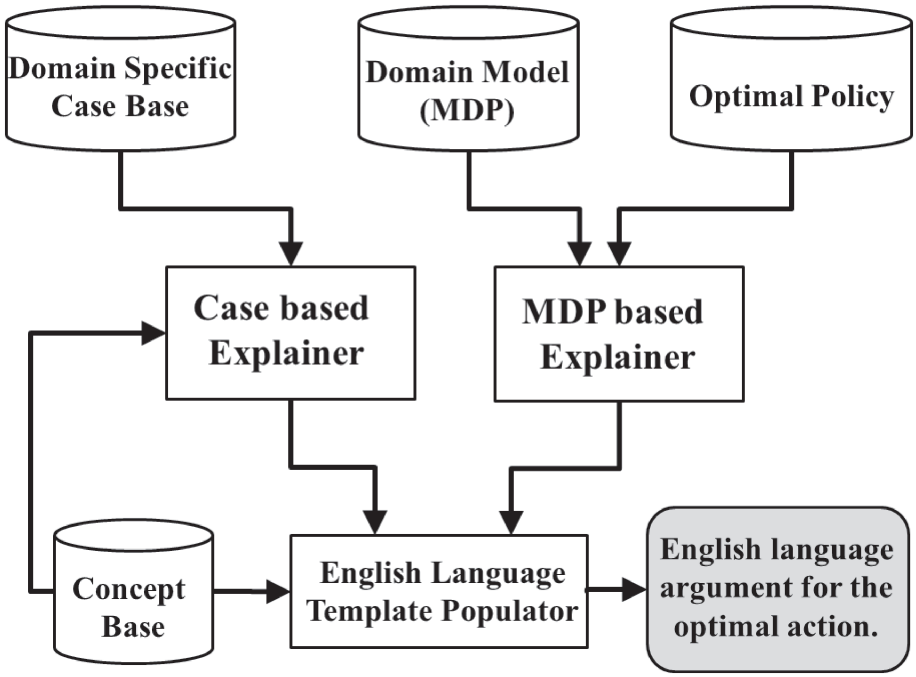

Fig. 2. The system organization and data flow: independent data sources feed into the reasoning modules that generate our arguments. The results of the argument generation modules are fed into the English Template Populator (ETP) module which combines the arguments with context specific information.

optimal value function has over 10,000 possible terminating configurations and 15,000 possible action paths to a terminating assignment.

4. SYSTEM OVERVIEW

Our explanation system integrates a policy-based approach with case-based and MDP-based algorithms. However, the MDP-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 domain-specific language in the English-language template populator module only. Thus, we isolate the domain-specific factors, giving our methods high portability between domains.

Figure 2 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 English-language template populator 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, for instance, 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 MDP-based explainer, which makes arguments from future predicted data, allows our system to generate better arguments than either piece alone.

For example, after all processing has been complete our system generates explanations similar to the following.

> 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.

The first two sentences contain an explanation of what the system is going to be doing in order to build some trust with the user [Sinha and Swearingen 2002]. The argument in the next two sentences comes strictly from the MDP model and the MDP policy fed to the recommendation system. The last two sentences contain information from the case-base only. The arguments generated from the MDP components and the case-base components are then fed through our template populator in order to generate English language explanations for the users. In the next several sections we will detail the inner workings of each of the components and how they fit together to create highly convincing arguments on behalf of the optimal policy.

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). Currently our system can only take in what grades students have in courses. We plan to augment the system with the ability to capture important user preference data as suggested in our user study (see Section 6.1). Once this data has been provided to the system, a explanation, in conversational English, 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.

4.1. MDP-Based Explanation

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 [Frederick et al. 2002; Tversky and Kahneman 1992]. A hyperbolic discounting function can make preference reversals possible: a user may prefer a different course of action before long term rewards have been realized [Read 2004]. We want to avoid this problem: if we highlight short and intermediate term rewards, we may be able to convince students to stick with the plans selected with the support of the system.

In contrast to human reasoning, an MDP uses an exponential discount function when computing optimal policies. Preference reversals cannot occur with an exponential discounting function, but we may still encounter other problems. Humans may not place appropriate discounts on their short and long term rewards; a classic example of this is investing in personal retirement plans. Many people *know* that, in order to maximize their long term utility they should save for retirement (large, long term reward), but that new flat screen TV would be great to have now (smaller, short term reward). Students may not be correctly accounting for future rewards; we want them to understand that some actions have very high valued rewards that are not realized in the short term.

The combined effects of human inability to think rationally in probabilistic terms and hyperbolic discounting of future utility means there is often a fundamental disconnect between the human user and the rational policy [Frederick et al. 2002; Tversky and Kahneman 1974]. 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.

The MDP-based explanation module extracts information from the MDP model of the domain and a policy of recommended actions for that model. It attempts to translate the reasoning method of the MDP to the reasoning method of the human user by explaining the long term plan in terms of short term gains. Specifically, the module outputs information about why, in terms of actions at the next decision point, the recommended action is best at the current decision point.

In order to facilitate comparing actions in this way, we will define an object which is the decrease in the value of taking the action a in timestep $t+1$ if some action a_i is taken instead of $\pi(s)$ in timestep t. This object is called the *action-factored differential value (AFDV)* of a following a_i (at state s under some policy π), $\Delta^{\pi}(s,\pi,a_i,a)$. This object is simply the difference in the expected utility of executing π and the expected utility of deviating from π by executing a_i 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. For example, if $\pi(s) = act\_ICP$, and we compute that the action-factored differential values of *act_PDPS* following any other a_i are all negative, we can point out that taking ICP before PDPS is better because taking ICP first improves the expected value of taking PDPS in the next time step more than any other action. We can also highlight how the current action can affect multiple future actions and utilities by considering different actions at timestep $t+1$. 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 utility associated with taking one or more future actions. This method allows for a salient explanation that focuses on how the current best action will improve actions and immediate utility in the next state.

To generate a usable set of AFDVs from some state s, we first 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:

$$V_2^{\pi}(s,a_1,a_2) - R(s) = \gamma \sum_{s' \in S} T(s'|s,a_1) \cdot [R(s') + \gamma \sum_{s'' \in S} T(s''|s',a_2) \cdot V^{\pi}(s'')]\,. \tag{4}$$

Using V_2^{π}, we can then compute a single AFDV object for a following a_i instead of $\pi(s)$ by computing the values of the two-step sequences $\{\pi(s),a\}$ and $\{a_i,a\}$ and taking the difference,

$$\Delta^{\pi}(s,\pi,a_i,a) = V_2^{\pi}(s,\pi(s),a) - V_2^{\pi}(s,a_i,a). \tag{5}$$

To compute a full set of AFDVs for explanation, this computation is done for all $a \in A$ (since we would like to consider an arbitrary action in timestep $t+1$) and for all $a_i \in A \setminus \pi(s)$ (since $\Delta^{\pi}(s,\pi,\pi(s),a)$ is identically 0).

In order to choose actions for explanation, we count, for each next-step action $a \in A$, the number of initial actions $a_i \in A \setminus \pi(s)$ for which $\Delta^{\pi}(s,\pi,a_i,a)$ is negative. We therefore define

$$x_s^{\pi}(a) = |\{i : \Delta^{\pi}(s,\pi,a_i,a) < 0\}| \tag{6}$$

as the number of actions in the current state, s, which cause a greater increase in utility of the action a than the recommended action.

Note that we may have for all $a \in A : x_s^{\pi^*}(a) > 0$, since by definition

$$\sum_{a \in A} \Delta^{\pi^*}(s, \pi^*, \pi^*(s), a) > \sum_{a \in A} \Delta^{\pi^*}(s, \pi^*, a_i, a) \quad \forall a_i \neq \pi^*(s). \tag{7}$$

That is, only the *sum* of the AFDV set is is guaranteed to be maximal for the optimal action.

We choose the subset of A for which $x_s^{\pi}(a)$ is minimal as our explanation actions, and explain $\pi(s)$ in terms of its positive effects on those actions. We could also decompose the actions into corresponding variable assignments and explain how those variables change, leading to higher utility. 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 features of the recommended action to be preserved. While our intuition is that reducing the overall length of the explanation is desirable, if a system designer wanted to present more explanation actions, 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 actions 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 following.

> The recommended action is *act_PDPS*, generated by examining long-term future benefits. 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*.

Algorithm 1 presents the algorithm for computing $V_2^{\pi}(s, a_1, a_2)$. It relies on a function particular to the implementation, APPLYACTION, which takes a state and an action and returns the set of possible next states and their associated probabilities. In general, APPLYACTION is linear in the size of the state-space, however, in a factored MDP, it can be easily precomputed for each action. In the worst case, the function $V_2^{\pi}(s, a_1, a_2)$ is in $\mathcal{O}(|S|^2)$.

Algorithm 2 presents the algorithm for computing the actions for explanation. If computing the two-step lookahead values is in $\mathcal{O}(|S|^2)$, then this algorithm is in $\mathcal{O}(|S|^2|A|^2)$, as it loops over all actions except the recommended action in the current timestep for every possible action in the next timestep.

It is possible to construct pathological domains where our domain-independent explainer cannot find a suitable action for explanation. 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 utility; this prevents contradictions between the explanation and policy. The AFDV method will break down if domains are constructed such that the expected utility is 0 within the horizon (2 time steps). This can happen when there are balanced positive and negative utilities. For this reason, we currently restrict our domain independence claims to those domains with only non-negative utilities. This, however, is not a significant restriction, as the utility function can always be rescaled by an arbitrary constant.

Algorithm 1 Compute $V_2^\pi(s, a_1, a_2)$

Require: $R(s), \pi(s), V^\pi \in \mathcal{O}(1)$
  $\Sigma \leftarrow 0$
  $S', P' \leftarrow$ ApplyAction(a_1, s)
  for all $s' \in S'$ **do**
    $\sigma \leftarrow 0$ {temporary variable to sum over the inner s'' loop}
    $S'', P'' \leftarrow$ ApplyAction(a_2, s')
    for all $s'' \in S''$ **do**
      $\sigma \leftarrow \sigma + P''(s'') \times V^\pi(s'')$
    end for
    $\sigma \leftarrow \gamma \times \sigma$
    $\sigma \leftarrow \sigma + R(s')$
    $\Sigma \leftarrow \Sigma + P'(s') \times \sigma$
  end for
  $\Sigma \leftarrow \gamma \times \Sigma$
  $\Sigma \leftarrow \Sigma + R(s)$
  return Σ

Algorithm 2 Compute actions for explanation

  $minval \leftarrow \infty$
  for all $a \in A$ **do**
    $x[a] \leftarrow 0$
    for all $a_i \in A \setminus \pi(s)$ **do**
      $\Delta \leftarrow V_2^\pi(s, \pi(s), a) - V_2^\pi(s, a_i, a)$
      if $\Delta < 0$ **then**
        $x[a] \leftarrow x[a] + 1$
      end if
    end for

    if $x[a] < minval$ **then**
      $minval \leftarrow x[a]$
    end if
  end for

  $E \leftarrow \emptyset$
  for all $a \in A$ **do**
    if $minval = x[a]$ **then**
      $E \leftarrow E \cup \{a\}$
    end if
  end for

  return E

According to the classification scheme presented by Tintarev and Masthoff [2011], our method of explanation is knowledge- or utility-based. The user's needs are encoded in the utility function, and our explanation is a decomposition of the expected utility by action. We have used a single, simple utility function for this example, but the explanation system does not rely on that—or any particular—utility function. We assume that, when it is deployed, the utility function will be personalized. It is likely that, with not too much additional work, we would be able to use the preference

representation to further refine our explanations. For instance, if the planning system were to elicit preferences in the form of an additive independent function, we should be able to leverage that independence to describe individual subgoals, and further customize the explanations that are generated.

4.2. Case-Based Explanation

While the utility function which specifies the optimal MDP policy is often derived from statistical relationships in past data, the arguments generated from that policy, in general, look forward from the current state. In our system, the policy-based argument attempts to convince the user that the recommended action will improve the expected utility of some action in a future time step.

However, we would also like to argue, if possible, that the immediate outcomes of the action recommended in the current state are desirable based on the current variable assignments (in our domain, this represents past academic performance). While the MDP model allows us to assign a probability to each possible outcome of the recommended action, that is precisely the type of argument which our system eschews. Instead, to make an argument in this form, we employ case-based explanation (CBE), which uses data about 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 [Nugent et al. 2009].

This argument from past data combined with our MDP-based argument from predicted future outcomes creates a strong complete argument for the action recommended by the optimal policy. In the computer science domains, our case base consists of $2,693$ distinct grade assignments in 6 distinct courses taken by 955 unique students. In the psychology domain, our case base consists of $11,117$ distinct grade assignments in 7 distinct courses taken by $6,395$ 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 [Nugent et al. 2009], 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 "Pass" 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 $scenario1 = \{var\_ICP = Good\} \rightarrow action = \{take\_PDPS\} \rightarrow scenario2 = \{var\_ICP = Good, var\_PDPS = Good\}$[2]. 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 [Aamodt and Plaza 1994]. 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,

[2]Recall from Section 3 that we made the choice to represent both A and B with "Good" and C with "Pass".

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 [Nugent et al. 2009]. Case matching methods are more suited to larger scenarios, and ideally use full state assignments [Aamodt and Plaza 1994]. 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.

Privacy concerns aside, 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 statistics generated dynamically from all relevant cases. A case is considered relevant if it consists of an assignment to a variable which matches an assignment in the user's current state as well as an assignment to the variable or variables affected by the action recommended in the current state. We select the relevant cases and compute the likelihood of a variable assignment with higher immediate utility 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.

We separate this method from the explanation system in order to maintain domain independence. This is done with a separate configuration file, called a concept base, used to store any domain-specific information. Our method accomplishes the selection of relevant cases in a domain-independent way using the ordered variable assignments stored in the concept base. 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 utility over the assignments for each variable in the factored utility function, 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 first consider the naïve distribution, $\Phi(v_i)$, over the values of v_i from cases in the database. For example, suppose that a student is in a state such that $var\_ICP = Good$, $var\_CALC2 = Good$, and $\pi(s_e) = act\_PDPS$. Since act_PDPS influences only var_PDPS, we must generate a naïve distribution, $\Phi(var\_PDPS)$. In this particular example, the zero-value marker over the ordered variable assignments is placed between "Poor" and "NotTaken." Even though both values are not rewarded in our model, we would like to include cases which correspond to students who have taken a course and failed. Thus, if 650 cases are found such that $var\_PDPS \neq NotTaken$, with 300 "Good," 250 "Fair," and 100 "Poor," $\Phi(var\_PDPS) = [0.47, 0.38, 0.15]$.

We then compute a conditional distribution, $\Phi(v_i \mid v_j)$, for each other value v_j in s. 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," $\Phi(var\_PDPS \mid var\_ICP = Good) = [0.65, 0.20, 0.15]$. Additionally, if 150 students had $var\_CALC2 = Good$ and $var\_PDPS \neq NotTaken$ with 100 "Good," 30 "Fair," and 20 "Poor," we would compute $\Phi(var\_PDPS \mid var\_CALC2 = Good) = [0.67, 0.20, 0.13]$. Note that we did not compute

$\Phi(var\_PDPS \mid var\_DM = NotTaken)$ because of the placement of the zero value marker. We would, however, still like to present statistics such as $\Phi(var\_ICP \mid var\_ICP = Poor)$ to account for students who may fail a course and wish to retake it. The zero value marker gives the system the flexibility to consider certain interesting variable assignments while neglecting other cases which would otherwise be equivalent with respect to the utility function.

For each conditional distribution, we examine the expected utility of the explanation action according to the distribution, and compare each with the expected utility under the naïve distribution. The conditional distribution which predicts the greatest increase in expected utility over the naïve distribution is then chosen to be used for explanation. Returning to the given example, since values of "Good," "Fair," and "Poor" in our model are 4.0, 2.0, and 0.0 respectively, the naïve distribution predicts an immediate utility of 2.64, while the conditional distributions predict utilities of 3.0 and 3.08.

Conditional distributions which predict a decrease in expected utility compared to the naïve distribution are not considered. Since only the explanation action is considered in the current state, and only distributions which increase the probability of assignments with positive utility are considered, this method cannot produce explanations which conflict with the policy-based explanation component.

In the given example, the conditional distributions indicate that a value of $var\_CALC2 = Good$ increases the utility of taking action act_PDPS, and moreoever, increases the utility more than any other assignment in the current state. The generated explanation primitive is therefore as follows.

> 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.

Though we do not present individual cases, the explanation is generated from information contained in the cases. Regardless of the number of cases used to create a statistic, we believe that we have preserved the core method of CBE: explaining a solution in terms of past solutions to similar problems. We chose to aggregate case information because, in practice, it simplified an incredibly rich topic (preparing and presenting a database of cases). While this is a departure from what is traditionally considered to be case-based explanation, we believe that the terminology is suitable. However, it is important to make a clear distinction between case-based reasoning in the general sense and the specific implementation which we chose to use to present case information.

4.3. Time to Graduation

Many of the subjective responses to our system in the user study (see Section 6.1) asked specifically for information about how long it would take to graduate from the current state, and how the recommended action would help them towards that goal. In order to reason about time to graduation, we must first discuss how graduation, and time to graduation, are represented in our domains. For CS and Psychology minors; graduation is represented by any scenario (partial assignment of state variables) in which the student has achieved a *Good* or *Pass* in the required courses. Our concept base also allows us to give a meaningful name to each explanation goal, so that we can use the word "graduation" rather than enumerating the variable assignments that comprise the individual scenarios. This grouping of explanation goal scenarios into explanation goals via the concept base also allows us the flexibility to define multiple explanation goals. For example, graduation with a passing GPA and graduation with a good GPA could be considered separately.

For example, the explanation goal with the label "graduation" in the CS minor domain is given by the set of scenarios {$var\_ICP = Good$ or $var\_ICP = Pass$, $var\_PDPS = Good$ or $var\_PDPS = Pass$, $var\_SE = Good$ or $var\_SE = Pass$, $var\_DM = Good$ or $var\_DM = Pass$, $var\_ALGO = Good$ or $var\_ALGO = Pass$}. Another possible explanation goal could be labeled "a passing grade in Algorithm Design and Analysis", and would consist of scenarios {$var\_ALGO = Good$ or $var\_ALGO = Pass$}. For the sake of brevity the method will be presented using the latter scenario.

One method of producing a "time to goal" explanation would be to use a deterministic version of the domain and an optimal deterministic planner or other search algorithm to find the minimal time to a goal. There are at least two potential issues with this method. First, we are concerned—based on years of interactions with students—that it will be interpreted as a promise, namely, "you *will* graduate in four semesters". Second, building a deterministic model and including search or planning code in our system would require significant overhead, especially for an approach that we believe is likely to be misleading. We therefore would like to consider an approach which is similar to our other methods, leveraging either the model and optimal policy or the case base.

A model-based explainer could be used to generate an argument of the form, "There is a 45% chance of achieving $var\_ALGO = Good$ in four or fewer steps". However, the results of our user study indicate that over half of students surveyed consider how past students have performed in their situation when selecting courses (see Section 6.1). In light of these results, we feel that arguments generated from the case base may be more convincing to our target users. Note that only certain types of case bases, specifically those which consist of inherently sequential cases and also include temporal information, will have the information required to implement "time to goal" measures. A generic case base can still be used to generate information about whether an explanation goal was attained from the current state, or to compute the proportion of past users who attained a given explanation goal from the user's current state.

When temporal information is included, there is a relatively simple algorithm: construct a career for every unique student in the case base, and check each career against the assignments to the user's current state and the set of goal scenarios to be considered for explanation. In this context, a career refers to the complete set of cases for a single unique student. This is more properly a redefinition of a single case to encompass an entire student's academic career, but for the sake of clarity, we will continue to refer to this set as a career. In the worst case, every case of every career must be checked against the user's current state, giving an algorithm which is linear in the number of cases. Constructing careers is also linear in the number of cases, though in practice, precomputing careers from the case base does offer an improvement in performance over reconstructing careers for each new explanation.

In our case base, the last time step in which a career shares any variable assignment with the user can be taken as the time step before the current state, while the first time step in which the career satisfies any one of the set of goal scenarios can be taken as the time step after which the explanation goal of which the scenario is a part is achieved. For example, if the user is in a state with $var\_ICP = Good, var\_PDPS = Good$, and we are considering an explanation goal $\{var\_ALGO = Good, var\_ALGO = Pass\}$, and we find a career in the case base where $var\_PDPS = Good$ was accomplished in time step 4, while $var\_ALGO = Good$ was accomplished in time step 7, the career can be said to represent the accomplishment of the explanation goal, "a passing grade in Algorithm Design and Analysis," from the current state in three time steps. Similarly, a career that has $var\_PDPS = Good$ in time step 4 and $var\_ALGO = Pass$ in time step 8 can be said to represent the accomplishment of the same explanation goal in four time steps.

Once again, we choose to aggregate statistics from the case base rather than present individual cases, so each career in the case base will be checked against the user's current state.

If we have found ten careers in the case base which accomplish any one of the set of goal scenarios, broken down as five careers of three time steps and five careers of four time steps, we may say that the explanation goal was achieved, on average, in four or fewer time steps. Additionally, if we are recommending that the user take *act_SE*, and the careers in the case base have *var_SE* = *Good* in the first time step after the time step matching the current state (time step 5 in the given example), we may generate an explanation of the following.

> Past students have taken *Software Engineering* and accomplished their goal of achieving *a passing grade in Algorithm Design and Analysis* in four or fewer *semesters* from the current state.

This is the strongest form of this type of explanation, because we have cases that exactly match the user's current state and recommended action. It is also possible that there is not an exact match in the case base. In this case, we may relax the condition that careers have the recommended action in the appropriate timestep and generate an explanation of the following form.

> Past students accomplished their goal of achieving *a passing grade in Algorithm Design and Analysis* in four or fewer *semesters* from the current state.

While this explanation does not recommend the optimal action, it does convey information about the time to the goal scenario, demonstrating to the user that we are considering their specific goals.

It should be possible to extend this approach to generic subgoals, however care must be taken in choosing subgoals for explanation. Because the optimal policy is specific to the utility function used during the planning phase, only certain subgoals, possibly only those which are scenarios with high utility, can be explained with reference to the policy, that is, using the stronger form of the argument. This is a possible direction for future research, as is validating the efficacy of the case-based "time to goal" method via user studies.

While this particular explanation method relies on features of the case base which may not be available in all domains, it was motivated by user study data which is necessarily domain-specific. It is not always possible, or even advisable, to preserve domain independence for its own sake. A system which is meant to be deployed should take into account features of the domain which allow it to satisfy the needs of its users. While the bulk of our argumentation methods were developed with domain independence in mind, we nevertheless remain open to including domain-specific explanation modules, such as this one, when the use case arises.

4.4. English-Language Template Populator

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 inherent problems. In medical decision making, Witteman et al. [2007] found that experienced doctors were more confident using a verbal, rather than numeric, scale. Renooij [2001] reports large inter-subject variability of the numerical values assigned to verbal expressions. 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 [Renooij 2001]. 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 [Murray and Häubl 2008; Sinha and Swearingen 2002]. 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. COMPARISON TO EXISTING SYSTEMS

The work reported here uses fully observable, factored MDPs [Boutilier et al. 1999]. The fundamental concepts used 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.

Our system differs from existing but similar systems such as the one designed by Elizalde et al. [2009] 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 MDP-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 [Frederick et al. 2002].

Additionally, Bohnenberger et al. used a combination of MDP plans and graphical explanations to help shoppers move through a shopping mall [Bohnenberger et al. 2005]. This system was explaining the plan as its primary user interface and was not, necessarily, attempting to justify why the plan was the best. Users were given several alternative routes to select from in order to provide confidence in the system itself.

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 other, similar, systems which focus on a "most coverage" heuristic, for instance, Khan et al.'s minimal sufficient explanation (MSE) system [Khan

et al. 2011]. A most coverage heuristic focuses on explaining a minimal level of utility that would be accrued by the optimal policy (see Section 2.3 for a more detailed explanation). 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. The MSE algorithm also leverages the concept of decomposing the total utility into individual scenarios, each with their own utility, but the MSE algorithm explains those scenarios in terms of occupancy frequency. Our approach to this problem is similar in spirit, though much less complete. While the MSE algorithm constructs scenarios with high utility on the fly, we will encode an explanation goal, a set of goal scenarios for explanation, in the concept base.

6. DISCUSSION AND USER STUDY

Our system successfully generates conversational English 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 worst-case runtime complexity of our system, to generate an explanation for a given state, is $\mathcal{O}(|S|^2|A|^2)$, where S is the set of all states and A is the set of actions. 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 [Khan et al. 2011] 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 the current format we assume the user's preferences are for a high GPA and to graduate as soon as possible. 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, decision-theoretic planning techniques are not necessarily well understood by our target users. 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 this.

> 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 MDP policy, enhancing the transparency of the recommendation in order to gain the trust of the user [Sinha and

Swearingen 2002]. It makes clear that the planning software is considering the long-term future. 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, which are computed using an optimal policy (which maximizes expected long-term utility). This part of the explanation focuses on the near future to explain actions which may only be preferable because of far future consequences. This shift in focus, from far to near, leverages the user's inherent bias towards wanting to realize rewards as soon as possible [Frederick et al. 2002]. The last statement in the explanation 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 statement 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.

Our domain sits at an interesting intersection of persuasion and explanation. We want to provide the users with enough information to make informed decisions and hope that they make ones that are optimal. There are some cases, such as convincing mentally disabled people to wash their hands [Hoey et al. 2012], where persuasion is the main driver of explanation. In our case we hope that by providing an honest explanation that is geared towards highlighting why a recommended course of action is beneficial, students will choose the course of action we have laid out for them. We want students to have an appropriate level of confidence in the recommended action as this may help them to stick to decisions when the going gets rough.

6.1. User Study

We conducted a large user study encompassing both domain experts (advisors) and target users of our system. Our study goals were manifold. We compare the advice generated by our system and its "most convincing" approach to other systems which use a "most coverage" (with respect to utility) approach. We investigate when and where users and experts would use our system as well as subjective user and expert assessments of our system on various features. Finally, we attempt to understand what factors users and experts would want to add to our system.

We surveyed 65 students enrolled in introductory computer science courses. These courses are open to all students, so a variety of majors are represented including computer science, computer engineering, electrical engineering, physics, math, and mechanical engineering. We also surveyed 130 students enrolled in introductory psychology courses. Again, these courses are open to all students and the majors of the students surveyed included psychology, biology, social work, family sciences, and undecided majors. This variety of majors and focuses allows us to make more general statements about the types of advice that students with different training would prefer. We surveyed students about additional questions regarding their perceptions of the advising process and specific factors affecting their decisions. These additional questions are interesting to practitioners and educators who seek to better understand the student mindset. We do not provide a full analysis of the survey results in this article. Instead, we are focused on the effectiveness of our system from a software usability perspective [Wickens et al. 1998; Wohlin et al. 2000] and we refer the reader to our other work on the topic, currently in preparation [Mattei et al. 2012].

We also conducted a survey of 10 advisors in order to gain perspective on how domain experts feel about our system and to validate our results against their advice.

Table I. System Output Comparison

Our System	MSE Based System
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.	Action TakeCS343 & CS448 is the best action because: It is likely to take you to CoursesCompleted = 6, TermNumber = Final about 0.86 times, which is as high as any other action.

Note: Side by side comparison of explanations generated by our system and example explanation taken verbatim from Khan et al.'s MSE based system [Khan et al. 2011].

The advisors were computer science faculty advisors, general College of Engineering advisors, and general College of Arts and Sciences advisors.

6.1.1. Users. We have data from 195 students from multiple classes and majors. Each student was asked a variety of questions about their GPA, major, and overall factors that they considered when receiving course selection advice. We were unable to have each student come into our lab and interact with the system individually so we generated paper surveys that were handed out to students in a variety of classes. In our paper questionnaires for each of computer science (CS) and psychology (PSY), we generated two fictional students. Both students are about half way through completing a minor in their respective course of study. One student is doing very well (about a 3.5 GPA) while the other student is struggling (2.3 GPA). For each treatment our system generated advice for the student given their progress in the degree and we generated explanations based on the minimal sufficient explanation (MSE) algorithm presented by Khan et al. [2011]. The wording for Khan et al.'s MSE explanation is taken verbatim from their published study, the numbers are calculated for our domain. An example of each type of explanation is given in Table I.

Table I illustrates the dichotomy of abstraction levels in the explanations generated by the two systems. Both approaches are mathematically correct. However, in a user-facing system, the way that the advice is framed is nearly as important for achieving the desired outcome of explaining or convincing the user of the optimality of the recommendation, as distrust and misunderstanding are two of the most often user cited reasons for not following a recommended plan or action [Murray and Häubl 2008]. Therefore we attempt to compare *perceptions* of the given advice. We believe that the use of appropriate language is a crucial feature in explanation systems; mathematical correctness is not enough. An attempt could be made to modify the language in the advice generated by the MSE system in order to equalize as many factors as possible, however, we felt it was more appropriate to use the explanations as published [Khan et al. 2011].

A full example of the survey instrument is available in the Appendix. We generated two versions of our survey for each minor (4 total versions for the students): in the first version our system presented a recommendation for the high achieving student and MSE advice for the struggling student, while in the second version of the survey the advice was switched (so our system would explain the struggling student). The

Table II. Summary of User Study Results

	Metric	Argument Approach	MSE Approach
CS Group	More Convincing	85.7%	14.3%
	Correctness Rating Median	4/5	4/5
	Clarity Rating Median	4/5*	2/5
	Convincingness Rating Median	3/5*	2/5
PSY Group	More Convincing	89.5%	10.5%
	Correctness Rating Median	4/5*	3/5
	Clarity Rating Median	4/5*	2/5
	Convincingness Rating Median	4/5*	3/5
All Responses	More Convincing	88.6%	11.4%
	Correctness Rating Median	4/5*	3/5
	Clarity Rating Median	4/5*	2/5
	Convincingness Rating Median	4/5*	3/5

Note: Median scores for the systems divided into the CS survey group, PSY survey group, and all respondents. Entries denoted with a (*) are statistically significant at $\alpha = 0.05$.

students for each major were split randomly into two groups and only saw one version of the survey. We chose this method in an attempt to control for possible bias as we felt that students may rate the positive advice as better since positive framings are generally preferred.

While none of the students in our survey were receiving advice specifically for their situation, asking users to put themselves in a closely related persons' shoes should preserve the results of our survey and is a common practice in human-factors and psychology research [Wickens et al. 1998]. The questionnaire presents the subject with a fictional student and gives the subject the full transcript of the fictional student, a listing of all the courses (with descriptions) required for a minor in the particular field (PSY or CS), the high school GPA of the fictional student, and how many semesters they had been attending college.

Since we controlled the courses where we distributed the surveys, we selected courses where we would have a high density of students who were about halfway through the coursework for a minor in the respective department. From the demographic portions of the survey we know that most (more than 75%) of the students who took the survey in CS and PSY were within 2 semesters (plus or minus) of the fictional students they were answering questions about. While the students who completed the survey had, in general, higher GPA's than the fictional poor performing student, many had GPA's close to the fictional high achieving student. Since the students were answering questions about fictional students, who were very much like them, we feel that the students who completed our survey were in a position to judge all of the properties of the system that we were testing. After the students received advice from the system, they were asked a variety of follow-up questions about their overall preference, additions to the system they would like to see, and how they would use the system if it was provided to them.

When asked which system they felt to be more convincing, our core metric, 36 of 42 (85%) of students receiving the CS version selected our system and 111 of 124 (90%) of students receiving the PSY version selected our system. We asked the students to select some key factors they used when making decisions about what courses to take in future semesters. We allowed the students to select multiple features or add their own to this question. Validating our reliance on analogy as an argumentation method, 38 of 62 (61%) in the CS group and 65 of 130 (50%) in the PSY group indicated that they considered the performance of past students in their situation. Overall, 47 of 62 (75%)

in the CS group and 104 of 130 (80%) in the PSY group indicated that they considered how past students in their situation performed and/or how a course would prepare them for future courses to be important when making a decision. The students also seemed to be very goal focused: 42 of 62 (68%) in the CS group and 100 of 130 (76%) in the PSY group responded that course requirements were an important factor in deciding what courses to take. While our system does not explicitly detail the course requirement structure we did verify that our two chosen methods, analogy and short-term utility, play a significant role in communicating with users.

After reading the advice from each system we asked the students to rate the advice on a Likert scale from 1 to 5 on the factors of correctness, convincingness, and clarity [Wickens et al. 1998]. For all the following analysis we compared the groups and systems with Mann-Whitney U tests, which assumes equal variances but ordinal data, and used $\alpha = 0.05$ as is standard in software testing [Wohlin et al. 2000] and higher education surveys [de Winter and Dodou 2010]. For all the questions, for both systems, and both groups, there was no statistically significant difference between the median responses based on what was being explained. This means that there was likely no effect on the students' perception of our system or the MSE based system because the system was explaining a good or a bad student. This is a positive result as we would like our system to be able to work with students who are doing well and students who are struggling. A summary of our analysis is shown in Table II.

For the correctness question, there was no significant difference between the CS and PSY group for our system ($NCS = 49$, $NPSY = 122$, mean rank of 75.82 for CS and 90.09 for PSY; $U = 3488$, $Z = -1.988$, and $p = 0.0728$) or the MSE explanations ($NCS = 50$, $NPSY = 115$, mean rank of 88.16 for CS and 80.76 for PSY; $U = 2617$, $Z = 0.9379$, and $p = 0.3507$). Our system has a somewhat higher variance between the CS group and the PSY group ($\sigma^2 = 1.022$ versus $\sigma^2 = 0.8736$) for the correctness score. This trend continues throughout all of our metrics including clarity and convincingness. We speculate that this is because some of the respondents in the CS group wanted more technical answers and marked our system down accordingly for its lack of hard numbers. Over all students our system was given a higher median correctness score $\tilde{x} = 4/5$ versus the MSE system $\tilde{x} = 3/5$ and this was a statistically significant result ($NAA = 171$, $NMSE = 165$, mean rank of 189.96 for our system and 146.26 for the MSE system; $U = 17776.5$, $Z = -4.271$, and $p = 1.7 \times 10^{-6}$). Generally, all students felt the advice generated from the systems to be correct and there were very few low outlying scores. The mean score for correctness from the PSY group on our system was slightly higher, $\bar{x} = 4.05$, versus the CS group, $\bar{x} = 3.76$. In general, across both systems, the PSY group seemed to express more trust in the computer system in general in their written comments than the CS students did. Even when some of the PSY students seemed to not be able to understand the MSE explanations, they would label the explanation as correct because, in the words of one anonymous survey taker, "the computer is probably correct." This level of trust did not seem to exist in the CS students as their written comments were more concerned with how the model was created and what was going on, "under the hood."

For the question of clarity, there was no significant difference between the CS and the PSY group for our system ($NCS = 49$, $NPSY = 116$, mean rank of 77.11 for CS and 89.57 for PSY; $U = 3424.5$, $Z = -1.5717$, and $p = 0.1164$) or the MSE based explanations ($NCS = 50$, $NPSY = 115$, mean rank of 78.94 for CS and 84.76 for PSY; $U = 3078$, $Z = -0.7499$ and $p = 0.3507$). There was a significant difference in the perceived clarity for our system ($\tilde{x} = 4/5$) versus the MSE based system ($\tilde{x} = 2/5$) and our system was much more preferred overall ($NAA = 171$, $NMSE = 165$, mean rank of 214.90 for our system and 120.42 for the MSE system; $U = 22040$, $Z = -9.1245$,

and $p = 2.2 \times 10^{-16}$). Our use of domain specific knowledge and careful choice of explanation language was most likely the cause for our systems' higher clarity score.

Finally, for the question of convincingness, there was no significant difference between the CS and the PSY group for our system ($NCS = 49$, $NPSY = 116$, median rank of 72.47 for CS and 87.45 for PSY; $U = 3358$, $Z = -1.9223$, and $p = 0.05449$) or the MSE based explanations ($NCS = 50$, $NPSY = 115$, mean rank of 75.17 for the CS group and 86.40 for PSY; $U = 3266.5$, $Z = 1.4306$, and $p = 0.1532$). We find again a somewhat high variance between the means for our system between the CS students and the PSY students ($\sigma^2 = 1.38$ versus $\sigma^2 = 0.80$). We feel, that for convincingness, this is a response both to the lack of hard numbers in our system and the more linear nature of the CS curriculum. Students in the CS minor have a well defined linear sequence of courses that they must take and that reinforcing this linearity decreases the convincingness of our system. Again, there was a statistically significant preference of students in both groups for our system ($\tilde{x} = 4/5$) over the MSE based system ($\tilde{x} = 3/5$), with our system being rated as more convincing ($NAA = NMSE = 165$, median rank of 202.96 for our system and 128.04 for the MSE system; $U = 19793.5$, $Z = -7.3145$, and $p = 6.4 \times 10^{-14}$).

When asked about their possible usage patterns, 37 of 44 (84%) in the CS group and 142 of 165 (86%) of the PSY group, responded they would use the system at home before and/or while talking to an advisor. Students were allowed to select multiple answers to this question to indicate the different possible ways they would use the system. There was a very small group of students, 7 of 44 (15%) for CS and 27 of 165 (16%) for PSY, that said they would use our system instead of talking to an advisor. There was also a small group of students, 7 of 44 (15%) for CS and 11 of 165 (9%) of PSY, who said they would either not use our system or use it for requirements checking only. When students were asked if they would make use of the advising feature if it was integrated with our university's course requirement checking feature, 24 of 44 (55%) for CS and 88 of 120 (73%) for PSY, responded that they would often or always use the recommendation feature.

We also asked the students in both categories what additional questions they would have if they received our advice and how they would make our system better. Of the students who responded to these question, 38 students in the PSY group and 25 in the CS group, more than 50% of the students in both groups had questions about subjective factors of courses. These questions included what the professor was like, and would taking two certain courses concurrently make for a particularly difficult semester. They also hinted at individual variation in preferences for number of projects versus number of exams. At this juncture, our system cannot provide such advice to students, as it would require abandoning domain-independent methods. The question of concurrent actions has not been addressed in this system, either in the policies or the explanations, and is left as a direction for future work.

About 50% of the PSY group and 40% of the CS group wanted to work through some "what if" scenarios. These included rearranging proposed courses and looking at different expected time to graduation and other factors. If these users had been able to interact with our explanation system they could have built and tested these "what if" scenarios in real time, a true benefit of our system. Since the user study, we have implemented features that would allow our system to discuss time to graduation (as it is a definable subgoal within the model).

There was a small fraction, less than 8% of CS students and no PSY students, who wanted to see more numbers in our system instead of our word explanations. A handful of students (less than 5%) wanted to know what we meant by "augment." They asked for more specific learning factors that a course would improve and how this would

translate to their future success. Our current model does not deal with identifying student aptitudes for things like mathematics or reading comprehension. Accounting for these factors would require a more complicated MDP with explicit modelling of tested aptitudes, or a partially observable MDP (POMDP) which could infer aptitudes from student grades. Additionally, about 10% of students in both groups expressed interest in working through whole plans of study for multiple semesters. These are interesting extensions to our work and we hope to address them in future versions of our system.

6.2. Advisors

Our advisor survey covers 10 advisors from several different walks of advising. Included are several faculty advisors and several advisors who advise students in multiple areas within a single college. While our smaller sample size does not allow us to present as a complete a statistical comparison as we would like, we can still draw some conclusions about how advisors see the role of our system.

The advisor survey differed several key ways from the survey shown in the Appendix. The advisors were given the student examples as a narrative of a fictional student including all their grades in the courses they had completed. The advisors were asked to give advice to these students. The advisors were then shown two new fictional students and advice generated by our system (no advice from the MSE system was included in the advisor study). The advisors were asked to rate the advise generated by our system across the same set of factors that we used in the student surveys.

Nearly all advisors, across all categories, saw requirements as the most important priority when recommending courses to students. This criterion was rated as the first priority for 9 of 10 advisors surveyed. In stark contrast to the students, 7 of 10 advisors rated drawing analogy between the current student and past student performance as the least important aspect of advising. Advisors rated our data as being generally correct with a median of 4/5 and generally clear with a median of 3/5. The advisors saw our advice for the struggling student as less clear and less correct because our system did not (and could not) engage the student in a discussion about choosing another major. In fact, when advisors did raise issues about the quality of our advice, it was generally in response to subjective factors. Advisors felt that our advice, while technically correct in most instances, left out many important factors that could only be gleaned and responded to by an in person interview.

The issue of subjective factors was key for the advisors. They felt, "there is no need to put a computer between two humans that need to communicate." In our sample, it was very obvious that advisors were worried that students, if given access to our system, would skip the person to person advising process in favor of a machine. This was reflected in that 7 of the 10 advisors said they would never or not often use our system or recommend our system to students. All three of the advisors who suggested giving students access to our system did so under the caveat that students should be required to still meet with a human advisor to clear up any comments or concerns that the student would have.

6.3. Study Discussion

We feel that our user study was successful in verifying our chosen psychological heuristics and confirming the correctness and clarity of our explanations. However, the extremely personal nature of advising prompted a somewhat adversarial reaction from some users and domain experts. We wholeheartedly agree that our system could never replace actual human contact and we never meant to suggest it would, could, or should. We would endorse the use of our system in an actual academic setting only as a

decision support tool; as one component of student advising including career counselling and face-to-face time with caring professionals.

Despite the emotional response, we feel our study validates our overall approach. Many of the students responded positively to the framing and manner of our advice. Many even identified our psychological techniques as being what they sought when asking for advice. We feel that, with our novel use of analogy and focus on immediate utility, we have created a system that is both flexible and applicable to multiple domains.

Our study should not be taken as an attempt to demonstrate that the AFDV approach to generating explanations is superior to the MSE approach. We feel that we have demonstrated that our approach to *framing* the explanations is more effective in all criteria and that we have validated the psychological heuristics which motivated our particular method of explanation. A direct comparison of the two approaches would require modification of the explanation language generated by the MSE system so that the abstraction levels are similar. Such a study may well reveal that the more effective mode of explanation depends on individual user preferences and ranking of subgoals within the domain. Indeed, the "time to subgoal" method of explanation which we added to the system in response to user feedback is more similar in spirit to the MSE method than the AFDV method, supporting that hypothesis.

Our study was also successful in that it highlighted the need to augment the modes of interaction between the system and the user. Both the students and the advisors wanted our system to be more interactive, though in very different ways. The users of our system wanted to address time to completion of subgoals, which we have since addressed in our system design. This is in line with the students wanting to manually check "what-if" scenarios; "if I get an A in this class, what should I do next." Making the ability to work through these scenarios easy and intuitive will be a focus of our future UI design as we have already augmented our system design to compute the results.

The advisors, however, want to see a different kind of interactivity; namely between the student and the advisor. The most promising approach we could take would be to leverage our system to identify and refer students to human advisors. In keeping with the feedback of our surveyed professionals, we do not feel it is in the best interests of the students to convert our system into a full fledge replacement for advisors. If our system were deployed on a department or university level we could use it to identify students who should really meet with an advisor or career counselor. This feature is already capable of being implemented in our system; we can provide static advice to classes of students (i.e., those who have a low GPA or fail a certain class). In these cases our system can give a static piece of advice such as, "please see Dr. X in the main advising office." Additionally, we would want to implement some tracking and reporting function on the back-end of a fully designed system. Note that features such as these are well outside the scope of our research project and would be more important if developing the system for actual commercial implementation.

7. CONCLUSION AND FUTURE WORK

In this work we have presented a system and design which generates conversational English explanations for actions generated by MDPs. This system uses a novel mix of case-based and MDP-based techniques to generate highly salient explanations. The system design abstracts the domain-specific 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 the specificity of explanations 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 should I take this particular course/atomic action?" The second will answer, "Why should I take 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 should I take 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 [Mundhenk et al. 2000]. 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.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

We would like to thank Elizabeth Mattei for her help with the statistical analysis, Jennifer Doerge for her helpful feedback on advising (and generally being an example of a great advisor), and the members of the UK AI–Lab, especially Robert Crawford, 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 article.

REFERENCES

Aamodt, A. and Plaza, E. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun. 7*, 1, 39–59.

Anthony, A. and Raney, M. 2012. Bayesian network analysis of computer science grade distributions. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE'12)*.

Avesani, P., Perini, A., and Ricci, F. 2000. Interactive case-based planning for forest fire management. *Appl. Intell. 13*, 41–57.

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press.

Bohnenberger, T., Jacobs, O., Jameson, A., and Aslan, I. 2005. Decision-theoretic planning meets user requirements: Enhancements and studies of an intelligent shopping guide. In *Proceedings of the 3rd International Conference on Pervasive Computing*. 279–296.

Boutilier, C., Dean, T., and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *J. Artif. Intell. Resear. 11*, 1–94.

Brüninghaus, S. and Ashley, K. 2003. Combining case-based and model-based reasoning for predicting the outcome of legal cases. In *Proceedings of the 4th International Conference on Case-Based Reasoning*.

Camara, W. J. and Echternacht, G. 2000. *The SAT I and High School Grades: Utility in Predicting Success in College*. College Board Res. Notes 10, College Entrance Examination Board, New York.

de Winter, J. and Dodou, D. 2010. Five-point Likert items: t test versus Mann-Whitney-Wilcoxon. *Pract. Assess. Resear. Eval. 15*, 11, 2.

Dodson, T., Mattei, N., and Goldsmith, J. 2011. A natural language argumentation interface for explanation generation in Markov decision processes. In *Proceedings of the 2nd International Conference on Algorithmic Decision Theory*. 42–55.

Elizalde, F., Sucar, E., Noguez, J., and Reyes, A. 2009. Generating explanations based on Markov decision processes. In *Proceedings of the Mexican International Conference on Artificial Intelligence*. 51–62.

Frederick, S., Loewenstein, G., and O'Donoghue, T. 2002. Time discounting and time preference: A critical review. *J. Econ. Lit. 40*, 351–401.

Guerin, J. T. and Goldsmith, J. 2011. Contructing a dynamic Bayes net model of academic advising. In *Proceedings of the Uncertainty in Artificial Intelligence Workshop on Bayesian Modeling Applications*.

Guerin, J. T., Hanna, J. P., Ferland, L., Mattei, N., and Goldsmith, J. 2012. The academic advising planning domain. In *Proceedings of the 3rd Workshop on the International Planning Competition*. 1–5.

Hansen, E. A. 2007. Indefinite-horizon POMDPs with action-based termination. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI'07)*. AAAI Press, 1237–1242.

Herlocker, J. L. 1999. Explanations in recommender systems. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems: Workshop on Interacting with Recommender Systems*.

Hoey, J., Boutilier, C., Poupart, P., Olivier, P., Monk, A., and Mihailidis, A. 2012. People, sensors, decisions: Customizable and adaptive technologies for assistance in healthcare. *ACM Trans. Intell. Syst. Technol. 2*, 4.

Hoey, J., St.-Aubin, R., Hu, A., and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*. 279–288.

Khan, O., Poupart, P., and Black, J. 2009. Minimal sufficient explanations for factored Markov decision processes. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*.

Khan, O., Poupart, P., and Black, J. 2011. Automatically generated explanations for Markov decision processes. In *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, 144–163.

Liberatore, P. 2005. On the complexity of case-based planning. *J. Exp. Theor. Comput. Sci. 17*, 3, 283–295.

Mahmood, T., Ricci, F., and Venturini, A. 2009. Improving recommendation effectiveness: Adapting a dialogue strategy in online travel planning. *Inf. Tech. Tourism 11*, 4.

Mathias, K., Williams, D., Cornett, A., Dekhtyar, A., and Goldsmith, J. 2006. Factored MDP elicitation and plan display. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*.

Mattei, N., Dodson, T., Guerin, J. T., and Goldsmith, J. 2012. Attitudes and expectations of undergraduate students in STEM and non-STEM disciplines. Manuscript.

Moore, B. and Parker, R. 2008. *Critical Thinking*. McGraw-Hill.

Mundhenk, M., Lusena, C., Goldsmith, J., and Allender, E. 2000. The complexity of finite-horizon Markov decision process problems. *J. ACM 47*, 4, 681–720.

Murray, K. and Häubl, G. 2008. Interactive consumer decision aids. In *Handbook of Marketing Decision Models*, B. Wierenga, Ed., Springer, 55–77.

Nugent, C., Doyle, D., and Cunningham, P. 2009. Gaining insight through case-based explanation. *J. Intell. Inf. Syst. 32*, 267–295.

Paz, J. F. D., Rodriguez, S., Bajo, J., and Corchado, J. M. 2009. Mathematical model for dynamic case-based planning. *Int. J. Comput. Math. 86*, 10–11, 1719–1730.

Puterman, M. 1994. *Markov Decision Processes*. Wiley.

Ray, S. and Sharma, A. 2011. A collaborative filtering based approach for recommending elective courses. In *Proceedings of the 5th International Conference on Information Intelligence, Systems, Technology and Management (ICISTM'11)*. S. Dua, S. Sahni, and D. Goyal Eds., Springer, 330–339.

Read, D. 2004. Intertemporal choice. In *The Blackwell Handbook of Judgement and Decision Making*, D. J. Koehler and N. Harvey Eds., Oxford University Press.

Renooij, S. 2001. Qualitative approaches to quantifying probabilistic networks. Ph.D. thesis, Institute for Information and Computing Sciences, Utrecht University, Netherlands.

Shani, G., Heckerman, D., and Brafman, R. I. 2005. An MDP-based recommender system. *J. Mach. Learn. Res. 6*, 1265–1295.

Sinha, R. and Swearingen, K. 2002. The role of transparency in recommender systems. In *CHI'02 Conference Companion*. 830–831.

Sutton, R. and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press.

Tintarev, N. and Masthoff, J. 2007. A survey of explanations in recommender systems. In *Proceedings of the 23rd IEEE International Conference on Data Engineering Workshop*. 801–810.

Tintarev, N. and Masthoff, J. 2011. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor Eds., Springer, 479–510.

Tversky, A. and Kahneman, D. 1974. Judgement under uncertainty: Heuristics and biases. *Science 185*, 1124–1131.

Tversky, A. and Kahneman, D. 1992. Advances in prospect theory: Cumulative representation of uncertainty. *J. Risk Uncertainty 5*, 4, 297–323.

Wickens, C., Lee, J., Liu, Y., and Gordon-Becker, S. 1998. *Introduction to Human Factors Engineering*. Addison-Wesley.

Witteman, C., Renooij, S., and Koele, P. 2007. Medicine in words and numbers: A cross-sectional survey comparing probability assessment scales. *BMC Med. Inf. Decis. Making 7*, 13.

Wohlin, C., Rune, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. 2000. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers.

Young, J. R. 2010. The Netflix effect: When software suggests students' courses. *Chron. Higher Educ*.

Received February 2012; revised September 2012, December 2012; accepted January 2013