Preference Handling—An Introductory Tutorial

Ronen I. Brafman and Carmel Domshlak

■ *We present a tutorial introduction to the area of preference handling—one of the core issues in the design of any system that automates or supports decision making. The main goal of this tutorial is to provide a framework, or perspective, within which current work on preference handling—representation, reasoning, and elicitation—can be understood. Our intention is not to provide a technical description of the diverse methods used but rather to provide a general perspective on the problem and its varied solutions and to highlight central ideas and techniques.*

Our preferences guide our choices. Hence an understanding of the various aspects of preference handling should be of great relevance to anyone attempting to build systems that act on behalf of users or simply support their decisions. This could be a shopping site that attempts to help us identify the most preferred item, an information search and retrieval engine that attempts to provide us with the most preferred pieces of information, or more sophisticated embedded agents such as robots, personal assistants, and so on. Each of these applications strives to take actions that lead to a more preferred state for us: better product, most appropriate articles, best behavior, and so on.

Early work in AI focused on the notion of a *goal*—an explicit target that must be achieved—and this paradigm is still dominant in AI problem solving. But as application domains become more complex and realistic, it is apparent that the dichotomic notion of a goal, while adequate for certain puzzles, is too crude in general. The problem is that in many contemporary application domains, for example, information retrieval from large databases or the web, or planning in complex domains, the user has little knowledge about the set of possible solutions or feasible items, and what she or he typically seeks is the best that's out there. But since the user does not know what is the best achievable plan or the best available document or product, he or she typically cannot characterize it or its properties specifically. As a result, the user will end up either asking for an unachievable goal, getting no solution in response, or asking for too little, obtaining a solution that can be substantially improved. Of course, the user can gradually adjust the stated goals. This, however, is not a very appealing mode of interaction because the space of alternative solutions in such applications can be combinatorially huge, or even infinite. Moreover, such incremental goal refinement is simply infeasible when the goal must be supplied offline, as in the case of autonomous agents (whether on the web or on Mars). Hence, what we really want is for the system to understand our preferences over alternative

 ISSN 0738-4602

choices (that is, plans, documents, products, and so on), and home in on the best achievable choices for us.

But what are preferences? Semantically, the answer is simple. Preferences over some domain of possible choices order these choices so that a more desirable choice precedes a less desirable one. We shall use the term *outcome* to refer to the elements in this set of choices.[1] Naturally, the set of outcomes changes from one domain to another. Examples of sets of possible outcomes could be possible flights, vacation packages, cameras, end results of some robotic application (such as pictures sent from Mars, time taken to complete the DARPA Grand Challenge, and so on). Orderings can have slightly different properties; they can be total (that is, making any pair of outcomes comparable) or partial, strict (that is, no two outcomes are equally preferred) or weak—the basic definitions follow—but that is about it. One can find various discussions in the literature as to when and whether total or weak orderings are appropriate (for an entry point, see, for example, (Hansson 2001b)), but this debate is mostly inconsequential from our perspective, and we make no commitment to one or the other.

Definition 1. A *preference* relation over a set Ω is a transitive binary relation $\succeq$ over Ω. If for every $o, o' \in \Omega$ either $o \succeq o'$ or $o' \succeq o$ then $\succeq$ is a *total* order. Otherwise, it is a *partial* order (that is, some outcomes are not comparable). A preference relation is *strong* if it is antisymmetric (that is, if $o \succeq o'$ then $o' \not\succeq o$), otherwise it is weak.

Given a weak ordering $\succeq$, we can define the induced strong orderings $\succ$ as follows: $o \succ o'$ IFF $o \succeq o'$, but $o' \not\succeq o$. Finally, if $o \succeq o'$ and $o' \succeq o$ then $o \sim o'$, that is, o and o' are equally preferred.

Unfortunately, while the semantics of preference relations is pretty clear, working with preferences can be quite difficult, and there are a number of reasons for that. The most obvious issue is the cognitive difficulty of specifying a preference relation. Cognitive effort is not an issue when all we care about is a single, naturally ordered attribute. For instance, when one shops for a particular iPhone model, all the shopper cares about might be the price of the phone. When one drives to work, all the driver cares about the choice of route might be the driving time. However, typically our preferences are more complicated, and even in these two examples often we care about the warranty, shipping time, scenery, smoothness of driving, and so on. Once multiple aspects of an outcome matter to us, ordering even two outcomes can be cognitively difficult because of the need to consider trade-offs and interdependencies between various attributes, as many readers surely know from their own experience. As the size of the outcome spaces increases, our cognitive burden increases, and additional computational and representational issues come in. How do we order a large set of outcomes? How can the user effectively and efficiently communicate such an ordering to the application at hand? How do we store and reason with this ordering efficiently?

Work in preference handling attempts to address these issues in order to supply tools that help us design systems acting well on our behalf, or help us act well. Preference *elicitation* methods aim at easing the cognitive burden of ordering a set of outcomes, or finding an optimal item. These methods include convenient languages for expressing preferences, questioning strategies based on simple questions, and more. Preference *representation* methods seek ways of efficiently representing preference information. The two are closely related because a compact representation requires less information to be specified, making its elicitation easier. Preference handling *algorithms* attempt to provide efficient means of answering common queries on preference models.

For the remainder of this article, we consider the overall process of preference handling, focusing on the basic questions involved and the key concepts. A number of basic concepts play a central role in our discussion, and these are *models, queries, languages, interpretation, representation,* and *model selection.* Two other central concepts are structure and independence. These concepts will be introduced gradually through concrete examples. They are by no means new or specific to the area of preference handling, but using them properly is crucial for a good understanding and a useful discourse. The choice of examples is admittedly somewhat biased to our own, but we believe that these choices serve the purpose of this tutorial well.

When evaluating the ideas described in this tutorial, the reader should bear in mind that there are many types of decision contexts in which preference handling is required. Each such context brings different demands, and one solution may be great for one context but less so in another. We believe preference-handling settings can be, very roughly, divided into three classes. The first involves online applications with lay users, best exemplified by various online product selecting problems. Here users are reluctant to provide much information, time, and effort; there is no room for user "training"; information is typically qualitative, and no technical knowledge should be expected of the user. Preference-handling techniques in this area must make the most of natural languagelike statements, and be very wise in their selection of questions. The second involves system configuration and design. In these applications preferences drive the behavior of some system the user cares about. In these applications we can expect more effort from the user, and the input language may be a bit more technical. Preference-

handling techniques for such problems can be more sophisticated but must still focus on enabling the user to naturally express his domain knowledge. Finally, there are complex real-world decision problems that require the assistance of a decision analyst. Here, we require tools such as probabilistic modeling and utility functions. Preference-handling techniques should provide the analyst tools that will make it easier for him or her to elicit these complex inputs and efficiently work with them.

Structurally, this tutorial is divided into two parts. The first and central part focuses on preference representation methods. It considers different complete preference models, languages that describe them, and some representational issues. In many application, however, we either don't need or won't get a complete preference model from the user. Thus, the second part deals with the question of what to do when the system is given only partial information about the preference model, and how to obtain as much of the most important parts of the model from the user. At the end of each subsection we provide both bibliographic notes on the material described in it, as well as pointers to related topics.

Further Reading

Work on preference models, algorithms, and representation appears in many areas. Discussion of preference models are abundant in the philosophical literature (Hansson 2001a, Krantz et al. 1971, von Wright 1963, Hallden 1957), in economics and game theory (Arrow and Raynaud 1986, Debreu 1954, von Neumann and Morgenstern 1947, Fishburn 1969), in mathematics (Birkhoff 1948, Davey and Priestley 2002), in operations research and decision analysis (Fishburn 1974, Wald 1950, Bouyssou et al. 2006), in psychology (Tversky 1967, 1969; Kahneman and Tversky 1979, 1984), and in various disciplines of compute science: AI (Doyle and Thomason 1999, Doyle 2004), databases (Agrawal and Wimmers 2000, Kießling 2002, Chomicki 2002), HCI (Gajos and Weld 2005, Chen and Pu 2007), electronic commerce (Sandholm and Boutilier 2006), and more.

Of particular interest to AI researchers are the fields of decision analysis and multicriteria decision making. Practitioners in these areas face diverse problems, often of great complexity, which call for a wide range of tools for modeling preferences, classifying, and sorting options. Some of their applications call for more esoteric mathematical models of preference that differ from the transitive binary relations we focus on here. See, for example, Fishburn (1999), and Oztürk, Tsoukiàs, and Vincke (2005).

Modeling Preferences

Two key questions arise when one approaches a modeling task, namely (1) what is the model? and (2) what questions or queries do we want to ask about this model? To achieve clarity and proper focus, these questions should be answered early on.

A model is a mathematical structure that attempts to capture the properties of the physical, mental, or cognitive paradigm in question. The purpose of the model is to provide a concrete proxy to the, sometimes abstract, paradigm of interest. As such, the model must have intuitive appeal, because it lends meaning to the query answers we compute. Returning to the topic of our discussion, a natural model for preferences is pretty clear—it is an ordering $\succeq$ over the set of possible outcomes, Ω. The answer to the second question varies with our application. Typical examples for queries of interest are finding the optimal outcome, finding an optimal outcome satisfying certain constraints, comparing between two outcomes, ordering the set of outcomes, aggregating preferences of multiple users, finding a preferred allocation of goods, and so on. Once we have a model and a query, we need algorithms that compute an answer to the query given the model. In our tutorial, algorithms will take the back seat, but of course, as computer scientists, we realize that ultimately we must provide them.

The key point is that a preference model must be specified for each new user or each new application (that is, a new set of possible outcomes). Obtaining, or eliciting, the information specifying the model is thus one of the major issues in the process of preference handling. One simple, but ineffective, approach to preference elicitation is to ask the user to explicitly describe the model, that is, an ordering. Clearly, this is impractical unless Ω is very small, and thus we must find an alternative tool that implicitly specifies the model in a compact way.[2] This tool is the language—a collection of possible statements that, hopefully, make it easy to describe common, or natural models. To give meaning to these statements we need an interpretation function mapping sets of such statements to models.

Putting things together, we now have the five basic elements of our metamodel: the *model,* which describes the structure we really care about; the *queries,* which are questions about the model we need to answer; the *algorithms* to compute answers to these queries; the *language,* which lets us implicitly specify models; and an i*nterpretation function* that maps expressions in the language into models. These concepts are depicted and interconnected graphically in figure 1 in which the semantics of the directed arcs is "choice dependence." For instance, as we discuss later, the choice of language depends big time on our assumptions about the

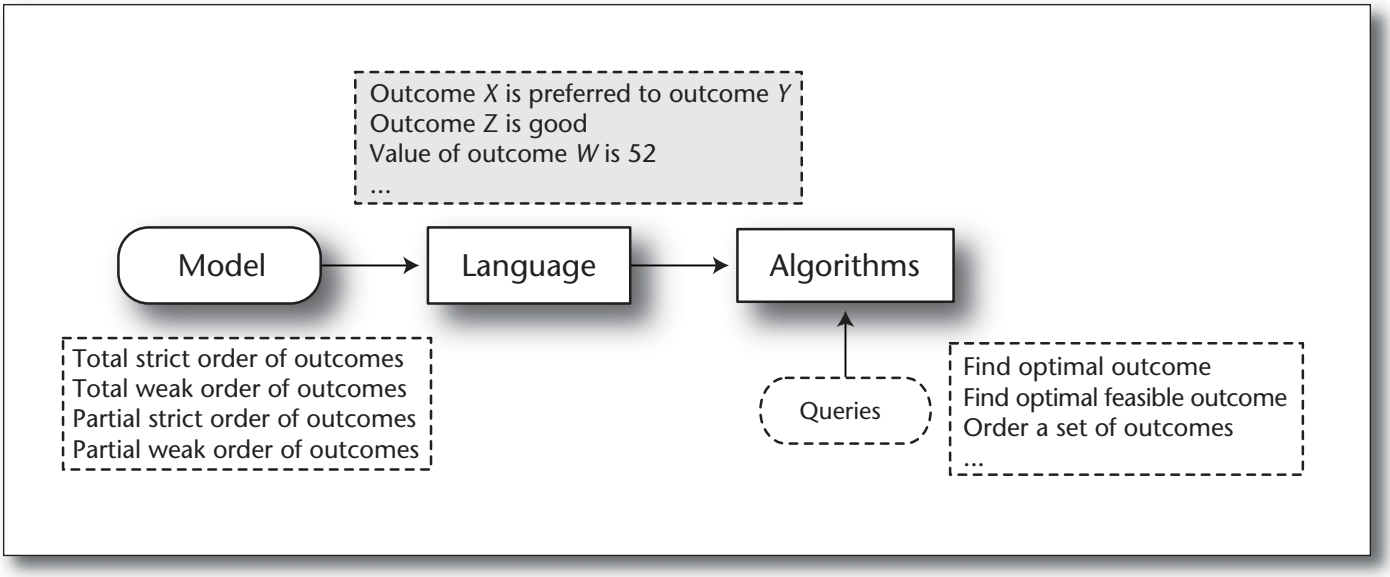

Figure 1. The Metamodel.

preference models of the users. Similarly, the algorithms developed for handling queries about the preferences are typically tailored down to the specifics of the language in use. In what follows, we discuss a few concrete instances of this metamodel. This discussion comes to relate the metamodel to some specific approaches to preference handling, and, synergetically, relate different approaches to each other.

Value Functions, Structure, and Independence

Let us start with the simplest concretization of the metamodel in which the model is a weak total order $\succeq$, and the language is simply the model itself. On the positive side, the model is nice because it is always possible to answer common questions such as "what is the optimal outcome?" or "which of two outcomes is better?" In addition, this combination of model and language makes the choice of interpretation entirely trivial as it should simply be set to the identity function. On the negative side, however, the choice of language here is obviously poor. For instance, imagine having to order hundreds of digital cameras accessible through some online electronics store with their multitude of attributes.

So we are going to gradually try to improve the language. First, rather than specify an explicit ordering of the outcomes, we can ask the user to associate a real value with each outcome. In other words, the language now takes the form of a value function $V : \Omega \rightarrow \Re$, the interpretation function is still simple, notably $o \succeq o' \Leftrightarrow V(o) \geq V(o')$, and the overall concretization of the metamodel this way is depicted in figure 2. At first, this appears an ill-motivated choice. Indeed, having to associate a real value with each outcome seems hardly easier than ordering the outcomes. In fact, while sometimes one may find it more convenient to, for example, assign a monetary value to each outcome rather than order Ω explicitly, most people will probably consider "outcome quantification" to be a more difficult task. However, the main utility of this move is conceptual, as it gives us a clue as to how things might be really improved: While the value function V can always be specified explicitly in the form of a table, there may be more convenient, compact forms for V. If such a compact form exists, and this form is intuitive to the user, then the complexity pitfall of eliciting user preferences on an outcome-by-outcome basis can be eliminated.

But why would V have a compact and intuitive form? Of course, if we treat outcomes as monolithic objects, for example, camera23, camera75, and so on, then such a compact form of V is not something we would expect. However, as the digital camera example suggests, outcomes of typically have some inherent structure, and this structure is given by a set of attributes. For example, flights have departure and arrival times, airline, cost, number of stopovers, and so on. When we purchase a flight ticket, what we care about are the values of these attributes. In the remainder of this tutorial, unless stated otherwise, we assume that we are working with such outcomes. That is, each outcome has some set of attributes, and our choice among these outcomes is driven by the value of these attributes. Formally, we assume the existence and common knowledge of a set of attributes $\mathbf{X} = X_1, \ldots, X_n$ such that $\Omega = \mathcal{X} = \times_{i=1}^{n} Dom(X_i)$, where $Dom(X_i)$ is the domain of attribute X_i. The domains of the attributes are assumed to be finite, although

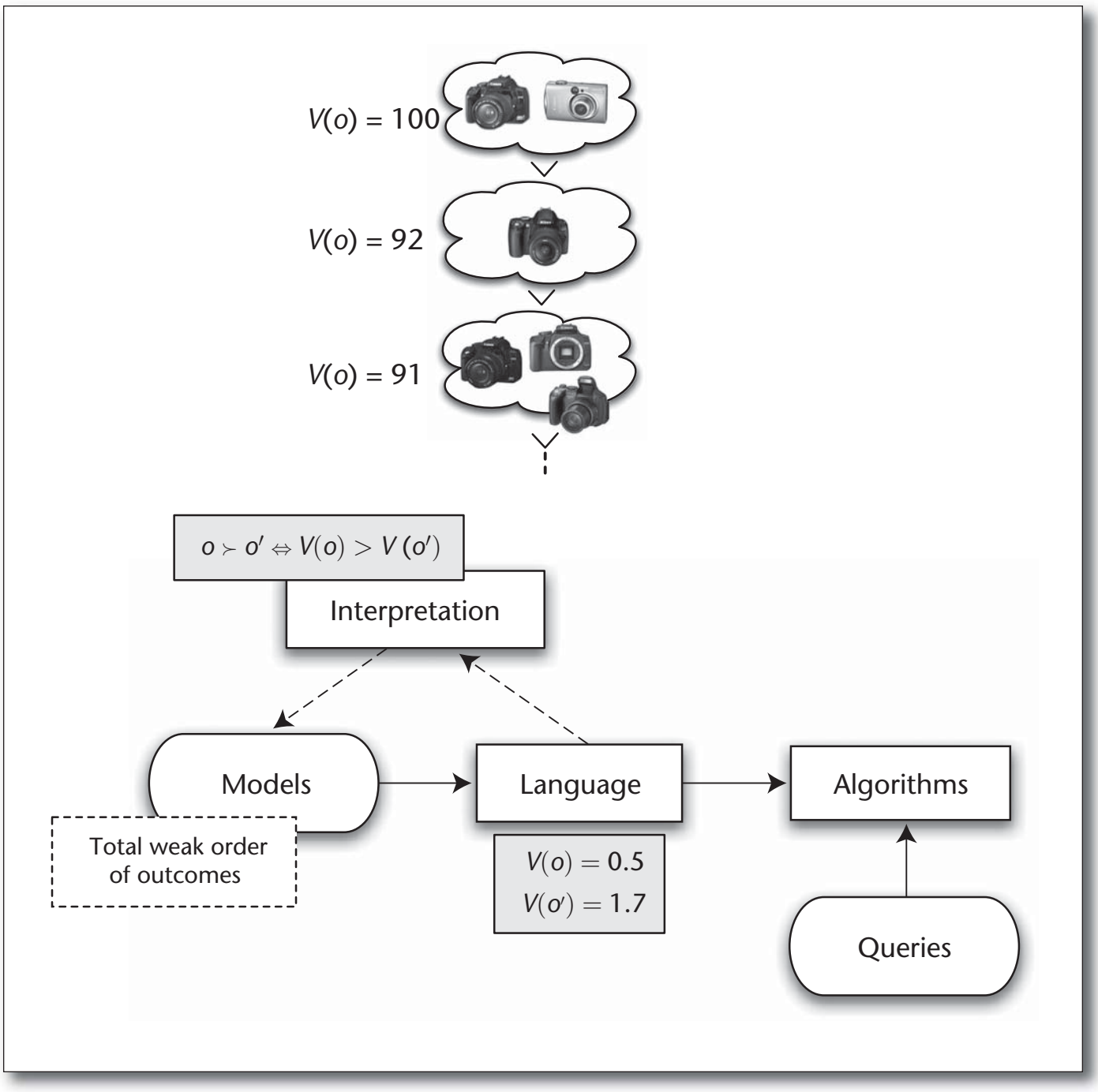

Figure 2. Weak Total Order.

handling infinite yet naturally ordered domains is typically very much similar.

Preferential Independence. Although the ability to describe an outcome by means of some preference-affecting attributes **X** is a necessary condition for a value function to be compactly representable, in itself it is not enough. An additional working assumption we need is that typical user preferences exhibit much regularity with respect to the given attributes **X**. The informal notion of "preference regularity" is formally captured by the notion of preferential independence.

Let **Y**, **Z** be a partition of **X**, that is, **Y** and **Z** are disjoint subsets of attributes whose union is **X**. We say that **Y** is preferentially independent of **Z** if for all $y_1, y_2 \in Dom(\mathbf{Y})$ and for every $\mathbf{z} \in Dom(\mathbf{Z})$,

$\mathbf{y}_1\mathbf{z} \succeq \mathbf{y}_2\mathbf{z}$ implies that for all $\mathbf{z}' \in Dom(\mathbf{Z})$: $\mathbf{y}_1\mathbf{z}' \succeq \mathbf{y}_2\mathbf{z}'$.

Intuitively, this means that preferences over **Y**-values do not depend on the value of **Z**, as long as that value is fixed. That is, if, when we fix **Z** to **z**, I weakly prefer $\mathbf{y}_1\mathbf{z}$ to $\mathbf{y}_2\mathbf{z}$, then when we fix **Z** to some other value **z**′, I still weakly prefer $\mathbf{y}_1\mathbf{z}'$ to $\mathbf{y}_2\mathbf{z}'$. For example, if my preference for a car's color is the same regardless of the brand, for example, I always prefer blue to red to white, then we can say that the color of the car is preferentially independent from its brand. (Here we assumed that color and brand are the only two preference-affecting attributes.) Similarly, for most people, price is preferentially independent of all other attributes. That is, given two outcomes that are similar, except for their price, most people would prefer the cheaper one, no matter what fixed concrete value the other preference-affecting attributes take in these outcomes. Importantly, note that when **Y** is preferentially independent of **Z** we can meaningfully talk about the projection of $\succeq$ to **Y**. Even more importantly, preferential independence of **Y** from **Z** allows the user to express his or her preferences between many pairs of outcomes in Ω in terms of **Y** only. For example, if my preference for a car's color is the same regardless of the brand, preference of blue to red implies preference of blue Toyota to red Toyota, blue BMW to red BMW, and so on.

Unlike probabilistic independence, preferential independence is a directional notion. If **Y** is preferentially independent of **Z** it does not follow that **Z** is preferentially independent of **Y**. To see this, suppose that cars have two attributes: price (cheap or expensive) and brand (BMW or Toyota). I always prefer a cheaper car to an expensive one for any fixed brand; that is, I prefer a cheap Toyota to an expensive Toyota, and a cheap BMW to an expensive BMW. However, among cheap cars, I prefer Toyotas, whereas among expensive cars, I prefer BMW. In this case, the ordering modeling my preferences is

(*cheap Toyota*) $\succ$ (*cheap BMW*) $\succ$ (*expensive BMW*) $\succ$ (*expensive Toyota*).

While here price is preferentially independent of brand, brand is not preferentially independent of price.

A weaker notion of independence is that of *conditional preferential independence*. Let **Y**, **Z**, **C** be a partition of **X**. We say that **Y** is conditionally preferentially independent of **Z** given $\mathbf{C} = \mathbf{c}$ if, when we project the ordering to outcomes in which $\mathbf{C} = \mathbf{c}$, we have that **Y** is preferentially independent of **Z**. More explicitly, for all $\mathbf{y}_1$, $\mathbf{y}_2 \in Dom(\mathbf{Y})$ and for every $\mathbf{z} \in Dom(\mathbf{Z})$

$\mathbf{y}_1\mathbf{z}\mathbf{c} \succeq \mathbf{y}_2\mathbf{z}\mathbf{c}$ implies that for all $\mathbf{z}' \in Dom(\mathbf{Z})$: $\mathbf{y}_1\mathbf{z}'\mathbf{c} \succeq \mathbf{y}_2\mathbf{z}'\mathbf{c}$.

We say that **Y** is preferentially independent of **Z** given **C** if for every $\mathbf{c} \in Dom(\mathbf{C})$ we have that **Y** is preferentially independent of **Z** given $\mathbf{C} = \mathbf{c}$. Thus, in the case of conditional preferential independence, our preference over **Y** values does not depend on **Z** values for any fixed value of **C**. Note that the preference ordering over **Y** values can be different given different values of **C**, but for any choice of **C** value, the ordering is independent of the value of **Z**. As an example, suppose that **Y**, **Z**, **C** are, respectively, brand (BMW or Toyota), mileage (low or high), and mechanical inspection report (good or bad). Given a good inspection report, I prefer BMW to Toyota, regardless of the mileage, perhaps because I generally like BMWs, but feel there is no point in a BMW whose mechanical condition is not good. Thus, I have (*BMW,low,good*) $\succ$ (*Toyota,low,good*), and (*BMW,high,good*) $\succ$ (*Toyota,high,good*). On the other hand, given a bad mechanical inspection report, I prefer Toyota to BMW, as they are cheaper to fix, and more reliable. Thus, (*Toyota,low,bad*) $\succ$ (*BMW,low,bad*), and (*Toyota,high,bad*) $\succ$ (*BMW,high,bad*). We see that once the value of the mechanical inspection report is fixed, my preference for brand is independent of mileage, yet the actual preference may change depending on the value of the mechanical inspection report.

Compact Value Functions and Additivity. The strong intuition obtained from the examples in the previous section is that user preferences in many domains exhibit much preferential independence of this or another kind. The question now is how can we leverage such structure. In what follows we'll see at least two answers to this question, and here we'd like to focus on the effect of preferential independence on the structure of the value function.

First, given an attributed representation of the outcome space $\Omega = \times_{i=1,\ldots,n} Dom(X_i)$, we can focus on value functions $V : \times_{i=1,\ldots,n} Dom(X_i) \rightarrow \Re$. As we noted earlier, this in itself buys us a language for specifying *V,* but not the compactness—the size of an explicit tabular representation of V would still be $\prod_{i=1\ldots n} |Dom(X_i)|$. However, depending on the actual preference ordering of the user, more compact forms of *V* are possible. Probably the best-known such form is that of a sum of functions over single attributes, that is,

$$V(X_1,\ldots,X_n) = \sum_{i=1}^{n} V_i(X_i)$$

Functions *V* that have this form are said to be *additively independent.* Intuitively, it seems that such factored form corresponds to some form of independence. Indeed, suppose that we know that a user preference ordering is captured by $V(Color,Brand) = V_1(Color) + V_2(Brand)$. Clearly, for any fixed value of color, the preference of the user over brands is identical, because the choice of col-

or simply fixes V_1. Thus, brand is preferentially independent of color, and similarly, color is preferentially independent of brand. In fact, if

$$V(x_1,\ldots,x_n)=\sum_{i=1,\ldots,n}V_i(x_i)$$

then for any attribute subset **Y** of **X**, **Y** is preferentially independent of **X** \ **Y**. This strong form of independence within **X** is called *mutual independence*.

Value functions in a "factored" form offer one major advantage: they are much easier to specify, and thus much easier to elicit from the users. To see the potentially huge benefit here, note that, if V is known to be additively independent over n Boolean attributes, its specification requires $2n$ values only, as opposed to 2^n values for a tabular, unstructured representation. Some forms of reasoning, too, are much simpler. For instance, finding the optimal outcome with respect to an additively independent V can be done in time O(n) by independently maximizing each of the subfunctions of V. In contrast, the same task with respect to the same function but in an unstructured, tabular form requires exhaustive search in time $O(2^n)$. But an additive independent form is very limiting, too. Basically, it implies that our preferences are unconditional, that is, the preference ordering over attribute values is identical no matter what values the other attributes have. Thus, if you prefer your coffee with milk, then you must prefer your tea with milk, too. Or, if you like a manual transmission in your sports car, you must like it in your minivan, too. Even worse, the strength of preference over attribute values is also fixed—informally, if the added value of milk in your coffee (as opposed to none) is greater than the added value of sugar (as opposed to none), then the same goes for tea.

Generalized Additivity. While additively independent value functions seem a limiting choice of language for preference specification, it *does* seem that our preferences satisfy many independence properties. For example, the only attributes that affect whether I prefer an aisle seat to a window seat on the plane are the duration of the flight and the route. Or, the only attribute that affects how much milk I want in my hot drink is the type of drink, not the amount of sugar, the glass, and so on. To capture such (weaker than additive) forms of independence, we have to shift our language to more expressive functional forms of value functions.

A more general form of additive independence that allows us to be as general as we wish, but not more than that, is that of generalized additive independence (GAI). The generalized additive form of a value function V is

$$V(X_1,\ldots,X_n)=\sum_{i=1}^{k}V_i(\ \ _i)$$

where $\mathbf{Z}_1, \ldots, \mathbf{Z}_k \subseteq \mathbf{X}$ is a cover of the preference-affecting attributes **X**. In other words, V is a sum of k additive factors, *Vi,* where each factor depends on some subset of the attributes **X**, and the attribute subsets associated with different factors need not be disjoint. By allowing for factors containing more than one attribute we enable capturing preferentially-dependent attributes. By allowing factors to be nondisjoint we might substantially reduce factor sizes by enabling different factors to be influenced by the same variable, without having to combine their associated variables into a single, larger factor. For example, the value of a vacation may depend on the quality of facilities and the location. How I assess each may depend on the season—in the summer I prefer a place near the beach, while in the winter I prefer a major city; in the summer I want large outdoor pools and terraces, while in the winter I prefer indoor pools, saunas, and large interiors. By using the language of GAI value functions, V(*Location,Facilities,Season*) can now be given in the form V_1(*Location,Season*) + V_2(*Facilities,Season*).

Note that, in contrast to the language of additively independent functions, the language of GAI functions is fully general. If we choose $k = 1$ and $\mathbf{Z}_1 = \mathbf{X}$, we can represent any value function. In the other extreme, if we choose $k = n$ and $\mathbf{Z}_i = \{X_i\}$, we obtain the language of additively independent value functions. Considering the user efforts required at the stage of preference elicitation, the useful trade-off between expressiveness and compactness seems to lie somewhere in between, with factor size (that is, $|\mathbf{Z}_i|$) being relatively small and the number of factors, k, being also not too large. However, even with this sublanguage of GAI value functions, answering queries may become much harder than with the language of fully additive value functions. This issue brings us to introducing the final component of our metamodel, namely the representation of the user-provided information.

The *representation* is a structure that captures all the information in the the model and is intended to be used directly by our query-answering algorithms. A natural choice for representation is the language itself—after all, the language is designed to facilitate efficient, concise characterization of the model. Indeed, in many cases, the language *is* used as the representation, and the algorithms operate on it. Much work in knowledge representation and reasoning in AI seeks to find various classes of languages for which efficient inference or optimization algorithms exist. Naturally, these classes cannot describe every model concisely, but often they can describe some important, or natural

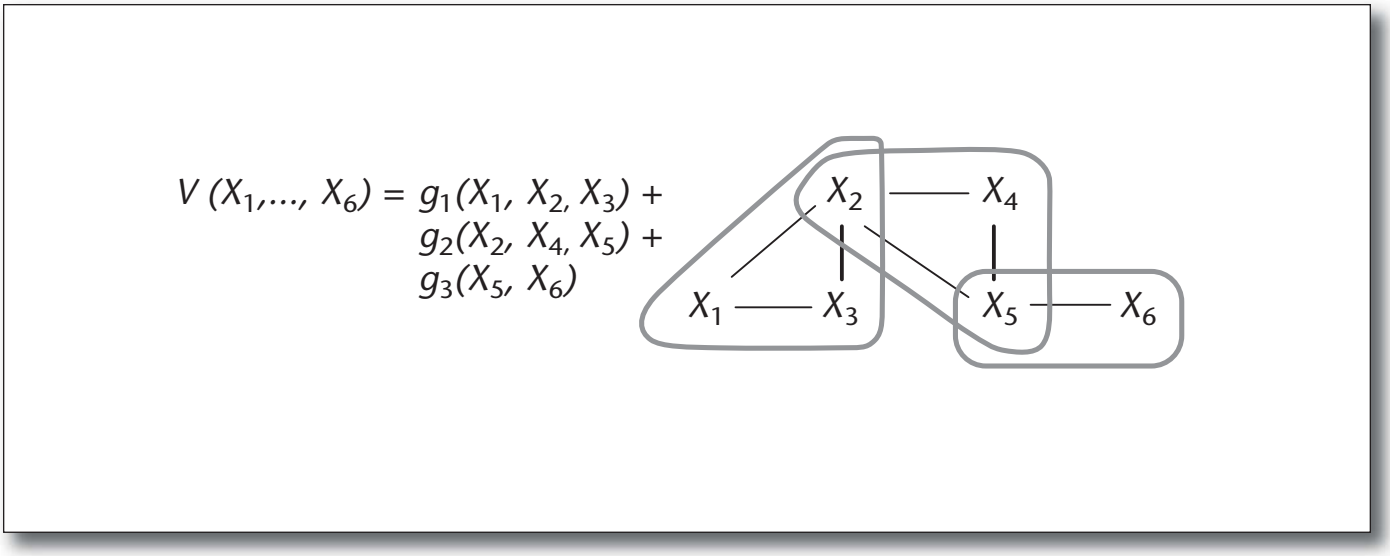

Figure 3. Example of a GAI Value Function and the Induced GAI-Network.

models concisely. Sometimes, however, it is useful to take the input expression and slightly transform or augment it to obtain a representation that provides more intuitive means for analyzing the complexity of query answering and developing efficient query-answering algorithms. Taking a look across various reasoning tasks in AI, typically such a representation comes in a form of an annotated graph, describing dependencies between different components of the problem. Indeed, graphical representations have been found useful for various preference models as well. These representations are compiled from the language (or in some cases, they augment the language as input devices), and inference and optimization algorithms make strong use of the properties of these graphical structures. In particular, the complexity of the algorithms can often be bounded by some graph-topological parameters of these graphs.

GAI value functions V provide us with the first example of this type. The language in this case is a set of tables describing the subfunctions of V. Using these tables, that is, using the language as the actual representation manipulated by the algorithms, it is easy to compute the value of any outcome and thus, to compare any two outcomes with respect to the preferences of the user. However, computing the optimal outcome, that is, finding the value of $\mathbf{X}$ that maximizes V is more difficult. To deal with such queries, it is useful to maintain a GAI-network (Gonzales and Perny 2004). A GAI-network is an annotated graph whose nodes correspond to the attributes in $\mathbf{X}$. An edge connects the nodes corresponding to a pair of attributes X_i, $X_j \in \mathbf{X}$ if X_i and X_j occur jointly in some factor, that is, $\{X_i, X_j\} \subseteq Z_l$ for some $l \in \{1, \ldots, k\}$. Assuming no redundant factors, each of the cliques in this graph corresponds to a factor in V, and it is annotated with a table describing the values of different instantiations of this factor. An example of the graphical core of a GAI-network appears in figure 3.

On the left side of figure 3 we see a GAI value function, and on the right hand side, we see the corresponding GAI-network. In the constraint optimization literature, this structure is called a *cost network* (Dechter 2003), and the problem of finding an assignment with maximal cost for such a network is well studied. Importantly, the algorithms used to solve this problem utilize the graphical structure of the network. This problem is equivalent to that of computing an optimal assignment to $\mathbf{X}$ with respect to a GAI value function. Consequently, the same algorithms can be used here, and they benefit from a representation of the value function in the form of a cost, or GAI, network. Putting things together, in the case of GAI value functions, we see that it is better to store the input in an intermediate representation as a cost or GAI network. The updated metamodel with the components associated with GAI value functions is depicted in figure 4.

Further Reading

Value functions originate in the decision theory literature, for example, Wald (1950). The notion of preferential independence is discussed in depth by Keeney and Raiffa (1976) who also consider their relation to additive forms. Generalized additive independence was introduced by Fishburn (1969) and was later made popular in the AI community by Bacchus and Grove (1995). GAI value functions are essentially identical to weighted CSPs (Dechter 2003), that is, CSPs in which a value is associated

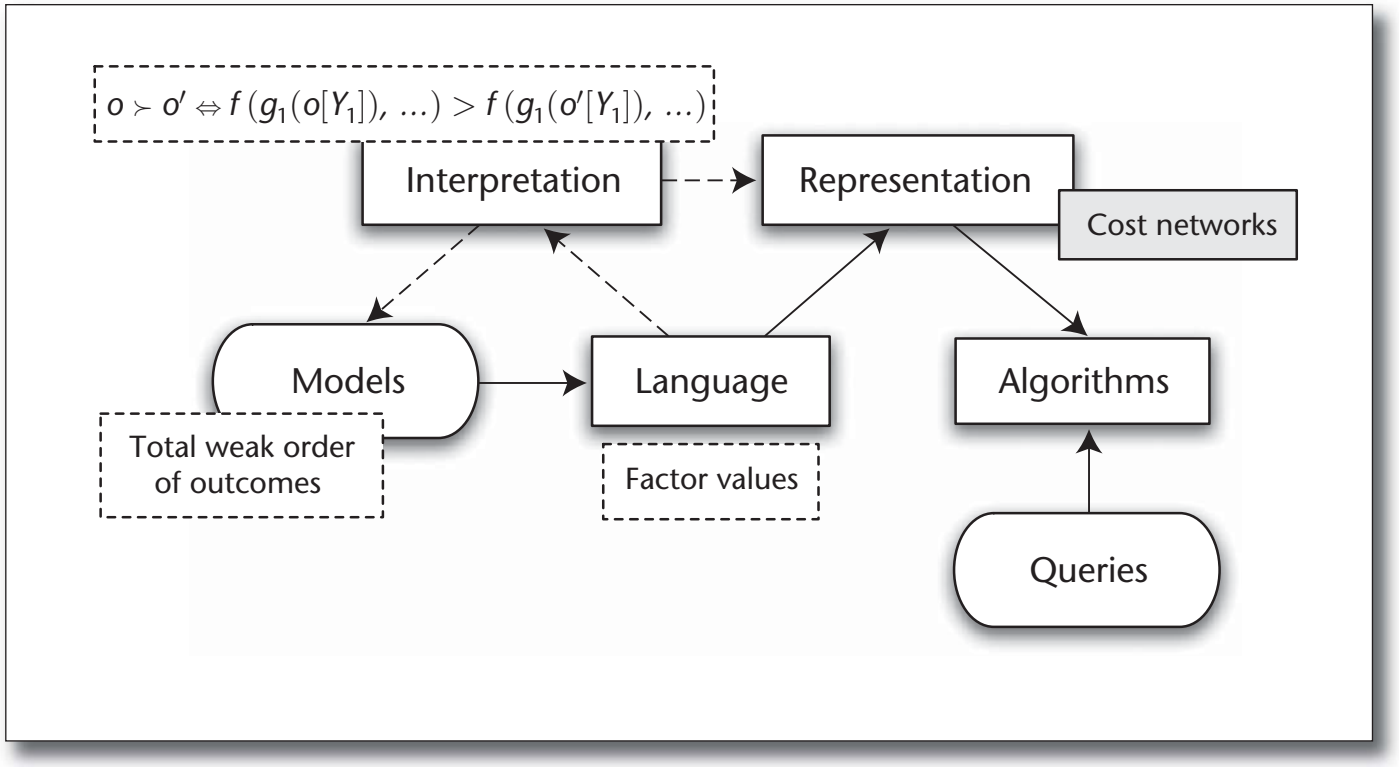

Figure 4. Metamodel for GAI Value Functions.

with each possible assignment to a constrained set of variables. In fact, the optimization problem of finding the optimal assignment to a GAI value function/cost network/weighted CSP is identical to that of finding most probable explanation (MPE) in graphical probabilistic models such as Bayes nets or join-trees (Pearl 1988, Lauritzen and Spiegelhalter 1988). Indeed, there is, in general, close correspondence between probabilistic reasoning and preferences. Technically, a joint probability distribution over random variables $x_1, \ldots, x_n$ also describes an ordering over the set of possible assignments. While the joint probability form is multiplicative, the logarithm is additive and looks much like an additive value function. Value (and utility) functions can be normalized to look like probabilities, but there are no natural corresponding notions of marginal probability and conditional probability in the case of preferences (although see Shoham [1997] for a slightly different perspective).

Value functions map tuples to integers, or reals. More generally, mappings to other structures can be defined, as has been done in the framework of soft constraints (Bistarelli, Montanari, and Rossi 1997; Bistarelli et al. 1999). This formalism can be viewed as generalizing the notion of GAI, as it allows for both new ranges for the function and aggregation and comparison operators other than addition and maximization.

Value functions over the Cartesian product of a set of attribute domains are one way of combining preference for different aspects of the problem. One can think of each factor in a value function as representing one criterion for selecting elements. Alternatively, one way to induce attributes on an object is to rate it according to multiple criteria. Multicriteria decision making is an important field of operations research and decision analysis that is concerned with the question of how to work with multiple, possibly conflicting criteria without necessarily combining them into a single value function (or more generally, a total order). See Bouyssou et al. (2006) for a detailed description of techniques in this area.

Partial Orders and Qualitative Languages

The language provided by GAI value functions for describing total orders is clearly much more convenient than an explicit ordering of the set of outcomes. However, the effort required to specify them is still too large for many applications, and the quantitative information the users are required to provide is typically not very intuitive for them. Therefore, while system designers may be willing to spend the time and effort required to specify a value function, casual, lay users, such as shoppers at online stores, are highly unlikely to do so.

So what properties would we like a language to have so that lay users will feel comfortable express-

ing their preferences or answering preference queries?[3] First, such a language must be based upon information that users find cognitively easy to reflect upon and express. Second, the information communicated in such language should be reasonably easy to interpret. Third, the language should allow a quick (and therefore, compact) specification of "natural" preference orderings. Finally, it should be possible to efficiently answer queries about the user preferences. Considering this list of expectations, it appears unavoidable that such a language be based on pieces of qualitative information about the model. Indeed, while quantitative information appears much harder to introspect upon, natural language statements, such as "I prefer SUVs to minivans," or "I like vanilla," appear quite often in our daily conversations (as opposed, say, to the "vanilla is worth $5.2 to me").

So what types of qualitative statements of preference can we expect users to provide? One simple class of statements corresponds to explicit comparison between alternatives, such as "I like this car more than that one," or "I prefer this flight to that flight." Similar statements are example-critiquing statements of the form "I want a car like this, but in blue." These statements are very easy to interpret, in part because they indicate an ordering relation between two alternative, completely specified outcomes. However, *because* these statements are very specific, they provide only a small amount of information about the preference model of the user. Another class of qualitative preference statements are generalizing statements, such as "In a minivan I prefer automatic transmission to manual transmission," or perhaps a more extreme "I prefer any red car to any blue car." Such statements are as natural in our daily speech as the more concrete comparison statements, yet they provide much more information because a single generalizing statement implicitly encodes many comparisons between concrete outcomes. Syntactically, generalizing preference expressions take the form

$$S = \{s_1, \ldots, s_m\} = \{\langle \varphi_1 \mathcal{R}_1 \psi_1 \rangle, \ldots, \langle \varphi_m \mathcal{R}_m \psi_m \rangle\},$$

where each $s_i = \varphi_i \, \mathcal{R}_i \, \psi_i$ is a single preference statement, with ψ_i, φ_i being some logical formulas over $\mathbf{X}$; $\mathcal{R}_i \in \{\succ, \succeq, \sim\}$; and $\succ$, $\succeq$, and $\sim$ have the standard semantics of strong preference, weak preference, and preferential equivalence, respectively.

For an illustration, the statements

s_1 "SUV is at least as good as a minivan"

and

s_2 "In a minivan, I prefer automatic transmission to manual transmission"

can be written as

$(X_{type} = SUV) \succeq (X_{type} = minivan)$

and

$(X_{type} = minivan \wedge X_{trans} = automatic) \succ (X_{type} = minivan \wedge X_{trans} = manual)$.

By adding such generalizing statements into our language, we facilitate more compact communication of the preference model. This benefit, however, can come with a price at the level of language interpretation because the precise meaning that a user puts into generalizing statements is not always clear. While the statement "I prefer any vegetarian dish to any meat dish" is easy to interpret, the statement "In a minivan I prefer automatic transmission to manual transmission" can be understood in numerous ways. For instance, under what is called the totalitarian semantics, this statement gets interpreted as "I always prefer a minivan with an automatic transmission to one with a manual transmission" (that is, regardless of the other properties the two cars may have). While sometimes reasonable, in general it assumes too much about the information communicated by the statement. Another possible interpretation is "I prefer a typical/normal minivan with automatic transmission to a typical/normal minivan with a manual transmission." Of course, we now have to define "typical" and "normal," a thorny issue by itself. Yet another possible interpretation, under what is called *ceteris paribus* ("all else being equal") semantics, is "I prefer a minivan with automatic transmission to one with a manual transmission, provided all other properties are the same." This is probably the most conservative natural interpretation of such generalizing statements, and as such, it is also the weakest. That is, it provides the weakest constraints on the user's preference model. But because it is conservative, it is also unlikely to reach unwarranted conclusions, which makes it probably the most agreed-upon general scheme for statement interpretation. However, the list of alternative interpretations that has been suggested in the literature on preferences and in the area of nonmonotonic reasoning is much longer, and the only solid conclusion we can probably draw from this situation is that the role of the interpretation function becomes very important in metamodels relying on qualitative statements of preference.

With this picture in mind, in the remainder of this section we consider a specific instance of a complete metamodel in which a qualitative language is used. Specifically, we consider the language of statements of preference on the values of single attributes. This specific language underlies the preference specification approach called *CP-networks* (Boutilier et al. 2004a), and the components of this metamodel instance are described in figure 5.

The model of user preferences assumed in the CP-networks approach is that of partial orders over outcomes. The reason is practical. Total orders are difficult to specify. Ultimately, to specify a total

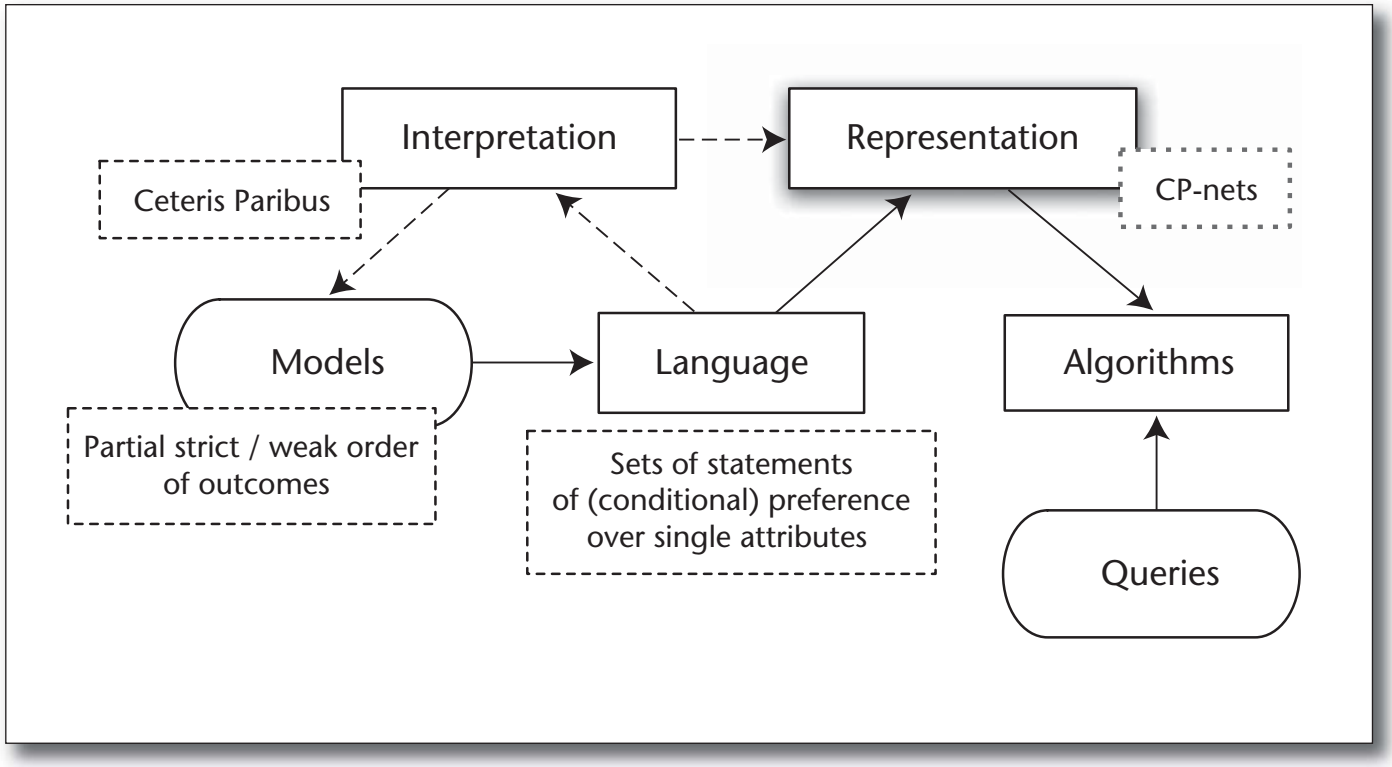

Figure 5. Metamodel for CP-Nets.

order, we need information that allows us to compare any two outcomes. That is quite a lot of information, and generally it is unlikely that casual users will be ready to provide it. In addition, in practice we often don't need to compare every pair of outcomes. For instance, this information might not be essential for the tasks of identifying the most preferred outcomes, ordering reasonably well a given set of outcomes, and so on.

The language employed in the CP-networks approach is the language of (possibly conditioned) preference statements on the values of single attributes. For instance, in the statement "I prefer black sports cars to red sports cars," the preference is expressed over attribute *Ext-Color,* with the value of *Category* conditioning the applicability of the statement only to pairs of sports cars. Formally, the language of CP-networks consists of preferences expressions of the form

$$S \subseteq \left\{ \mathbf{y} \wedge x_i \succ \mathbf{y} \wedge x_j \middle| X \in \mathbf{X}, \mathbf{Y} \subseteq \mathbf{X} \setminus \{X\}, \right. \\ \left. x_i, x_j \in Dom(X), \mathbf{y} \in Dom(\mathbf{Y}) \right\}$$

Note that $\mathbf{Y}$ can be any subset of attributes excluding the referent attribute X, and each statement $\mathbf{y} \wedge x_i \succ y \wedge x_j$ in the expression says that, in some context captured by assigning $\mathbf{y}$ to $\mathbf{Y}$, the value x_i is preferred to x_j for X.

The interpretation function used in CP-networks corresponds to the ceteris paribus interpretation of individual statements, and then transitive closure of the preference orderings induced by these statements. Specifically, a statement $\mathbf{y} \wedge x_i \succ \mathbf{y} \wedge x_j$ is interpreted as communicating a preference ordering in which outcome o is preferred to outcome o' whenever o and o agree on all attributes in $\mathbf{X} \setminus \{X\}$, they both assign the value $\mathbf{y}$ to $\mathbf{Y}$, and o assigns x_i to X while o' assigns x_j to X. For example, consider the statement s_1 in figure 6, which can be written as: $X_{category} = minivan \wedge X_{ext\text{-}color} = red \succ X_{category} = minivan \wedge X_{ext\text{-}color} = white$. s_1 induces the following preference ordering on the tuples in the table on the top right: $t_1 \succ t_3$ and $t_2 \succ t_4$. These relationships follow from the ceteris paribus semantics because both t_1, t_3 and t_2, t_4 are minivans and they both assign the same value to all other attributes (*Int-Color* in this case). However, t_1's *Ext-Color* is preferred to t_3's. In contrast, $t_1 \succ t_4$ does not follow from $s1$ because t_1 and t_4 assign different values to *Int-Color.* Considering now a set of statements as a single coherent preference expression, the interpretation function in the CP-networks approach takes the transitive closure of the orderings induced by each of the statements s_i. For example, we saw that s_1 induces $t_2 \succ t_4$. From s_4 we can similarly deduce $t_1 \succ t_2$ because these are two red cars whose other attributes (*Category* in this case) are the same. Thus, the preference ordering induced by the overall expression provides us with $t_1 \succ t_4$.

The last conceptual component of the metamodel is the representation of the input information. In the CP-networks approach, the preference

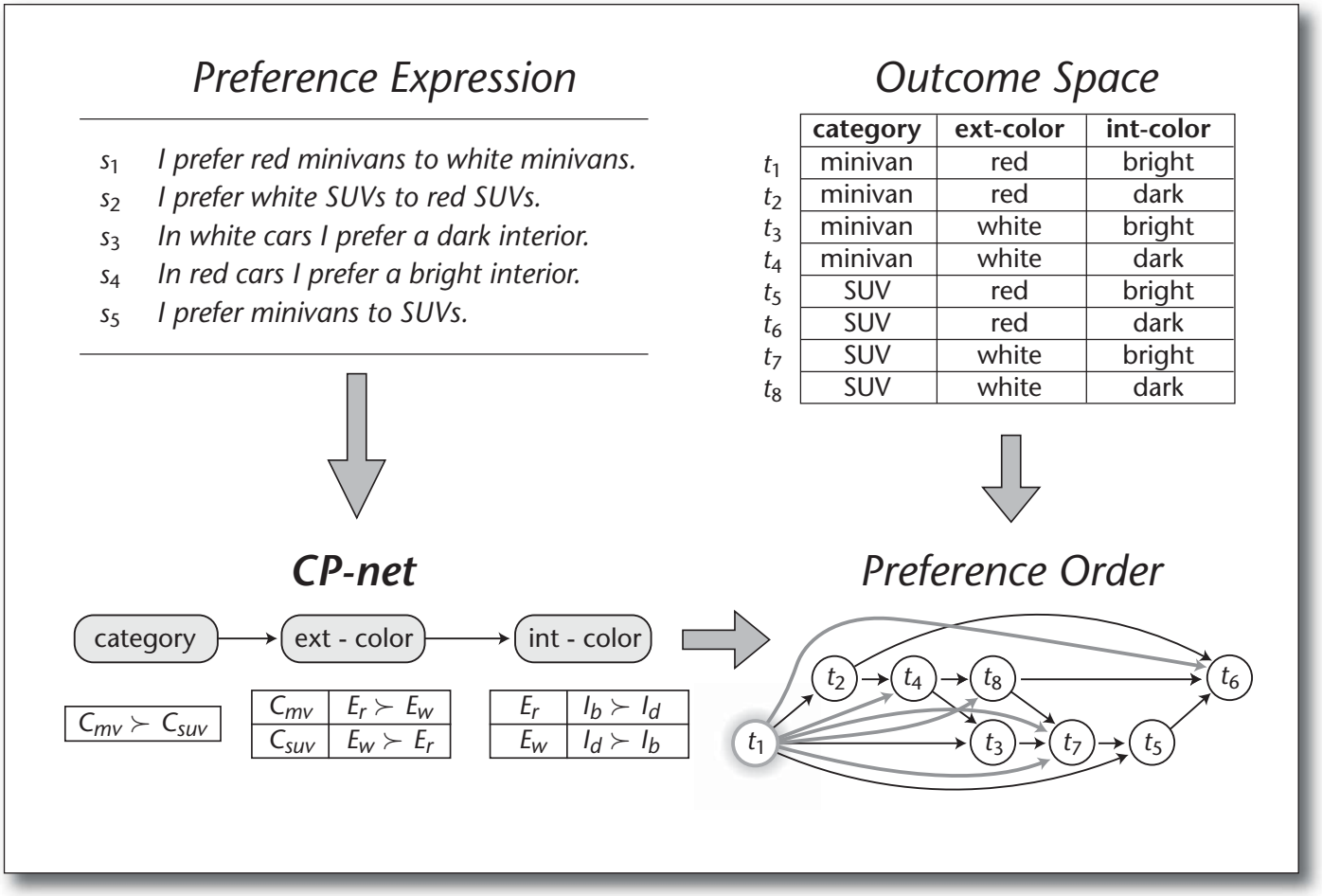

Figure 6. CP-Net Example: Statements, CP-Net, Induced Preference Relation.

expression is represented using an annotated digraph (that gives the name to the approach). In this digraph, nodes correspond to outcome attributes, and edges capture the conditioning structure of the preference statements. Specifically, for each statement $\mathbf{y} \wedge x_i \succ \mathbf{y} \wedge x_j$ we add an edge from each node representing an attribute $Y' \in \mathbf{Y}$ to the node representing attribute X, because our preference for X values depends on the value of Y'. Finally, each node is annotated with all statements that describe preferences for its value. Figure 6 depicts the CP-network induced by statements $\{s_1, \ldots, s_5\}$. For example, because of statement s_1 in which the preference for exterior color depends on category, *Category* is a parent of *Ext-Color* in the network. In that figure we can also see the relevant preference statements in the table of each node. On the lower right side of this figure we see the induced preference relation. Note that for clarity we introduced only the ordering relations that follow directly from the statements and just a few of the relations obtained by transitive closure.

Recall that earlier we emphasized the notion of independence. How does it come into play in CP-networks? The answer is that it is a by-product of the interpretation semantics. Notice how the ceteris paribus semantics and the notion of conditional preferential independence are closely related. In both some context is fixed, and then the order over values of some attribute set is identical regardless of the (fixed) value of the remaining attributes. In CP-networks, the ceteris paribus interpretation forces the attribute associated with a node to be preferentially independent of all other attributes, given the value of the parent attributes.

But what does the graphical structure of CP-networks really buy us? While users need not be aware of this structure, it was shown to play an important role in computational analysis of various queries about the preference model. First, the graph structure helps us ensure statement consistency. In general, a preference statement is inconsistent if it induces a cycle of strict preferences in the preference ordering. In that case, we can prove that $o \succ o$ for some outcome o. In the case of CP-networks, we know that if the CP-net is acyclic, then the preference statement it represents must be consistent. Note that the acyclicity of the network can be recognized in time linear in |X|—this

is probably the simplest example of a query that can be efficiently answered by using the graphical representation of the input expression. Second, when the network is acyclic, we can topologically sort its nodes. This topological ordering is a standard subroutine in various algorithms that operate on CP-networks. For instance, this ordering can be used to find an optimal outcome in time linear in $|\mathbf{X}|$.[4] Third, it can be shown that the complexity of comparison queries, that is, answering the query "does $o \succ o'$ hold," varies depending on the structure of the network from polynomial time to PSPACE-complete.

Finally, given the complexity of comparisons, it appears that finding a preferentially consistent total order of a given set of outcomes would be difficult as well. By a consistent total order we mean a total order such that if $o \succ o'$ holds then o will appear before o' in that order. Surprisingly, perhaps, this is not the case, and we can consistently order a set of outcomes quickly. The reason for this is that we have more flexibility in doing this. For two given outcomes, o, o', a comparison query asks whether $o \succ o'$ holds. In contrast, an "ordering query" asks for a consistent ordering of o, o'. To order o before o', we need only know that $o \not\prec o'$, which is easier to check. In fact, an ordering query needs only establish that one of $o \not\succ o'$ or $o' \not\succ o$ holds. When the CP-network is acyclic we can do this in time linear in $|\mathbf{X}|$. Of course, it is not obvious that, given that ordering two outcomes is easy, so is ordering an arbitrary number of outcomes, as there may be some global rather than local constraints involved. But fortunately, the same ideas apply, and we can order a set of N items in $O(N \log(N)|\mathbf{X}|)$ time.

Altogether, we see that the graphical representation in CP-nets holds much useful information about the induced preference order. This example provides good evidence that representation deserve careful attention as an important component of the preference handling metamodel.

Further Reading

CP-nets were introduced in Boutilier et al. (2004a, 2004b) and have generated various extensions, enhancements, and analysis, such as Wilson (2004); Rossi, Venable, and Walsh (2004); Goldsmith et al. (2005); and Brafman, Domshlak, and Shimony (2006). Preference languages and logics have been discussed by philosophers (Hansson 2001a) and AI researchers (Brewka, Niemela, and Truszczynski 2003; Tan and Pearl 1994; Doyle and Wellman 1994; Boutilier 1994). In fact, the best known interpretation for nonmonotonic logics is known as the preference semantics (Shoham 1987), and thus most nonmonotonic logics, such as Kraus, Lehmann, and Magidor (1990); McCarthy (1980); and Reiter (1980), can be viewed as expressing preference statements, usually associated with aggressive interpretations that attempt to deduce as much information as possible.

Graphical models play an important role in many areas of AI, most notably probabilistic reasoning (Pearl 1988) and constraint satisfaction (Dechter 2003). An early use in preference modeling appeared in (Bacchus and Grove 1995), where it was used to model independence in utility functions. More recent models include expected-utility networks (La Mura and Shoham 1999), which provide a model that combines preference and probabilities combined, GAI-networks, which model GAI utility functions (Gonzales and Perny 2004), UCP-net (Boutilier, Bacchus, and Brafman 2001), which provide a directed graphical model for value and utility functions, and CUI networks, which build on the notion of conditional utility independence to provide more compact representations for certain utility functions (Engel and Wellman 2006).

Preference Compilation

So far, we have seen two general approaches to preference specification. The more quantitative value function-based specifications are very simple to interpret and support efficient comparison and ordering operations. However, they are based on an input language that few users will find natural. The qualitative models that utilize generalizing statements attempt to provide users with intuitive input expressions; however, it is not always clear how to interpret these statements (for example, "I like Chinese restaurants"), comparisons may be difficult, and depending on the class of input statements, ordering may be difficult too. Things can become particularly hairy if we want to allow diverse types of input statements. On the one hand, this is clearly desirable, as it gives more flexibility to the users. On the other hand, the interpretation and interaction between heterogeneous statements may not be easy to specify. Likewise, developing efficient query-answering algorithms for such heterogeneous expressions becomes much more problematic.

Preference compilation techniques attempt to make the best of both worlds by mapping diverse statements into a single, well-understood representation, notably that of compact value functions. As outlined in figure 7, this mapping translates qualitative statements into constraints on the space of possible value function. As a simple example, if I say "car A is preferred to car B," then this puts a constraint on V in the form of $V(A) > V(B)$. Typically these constraints are consistent with multiple value functions, and depending on the method, we then answer queries by using one of the consistent value functions. Note that by working with one specific consistent value function, we

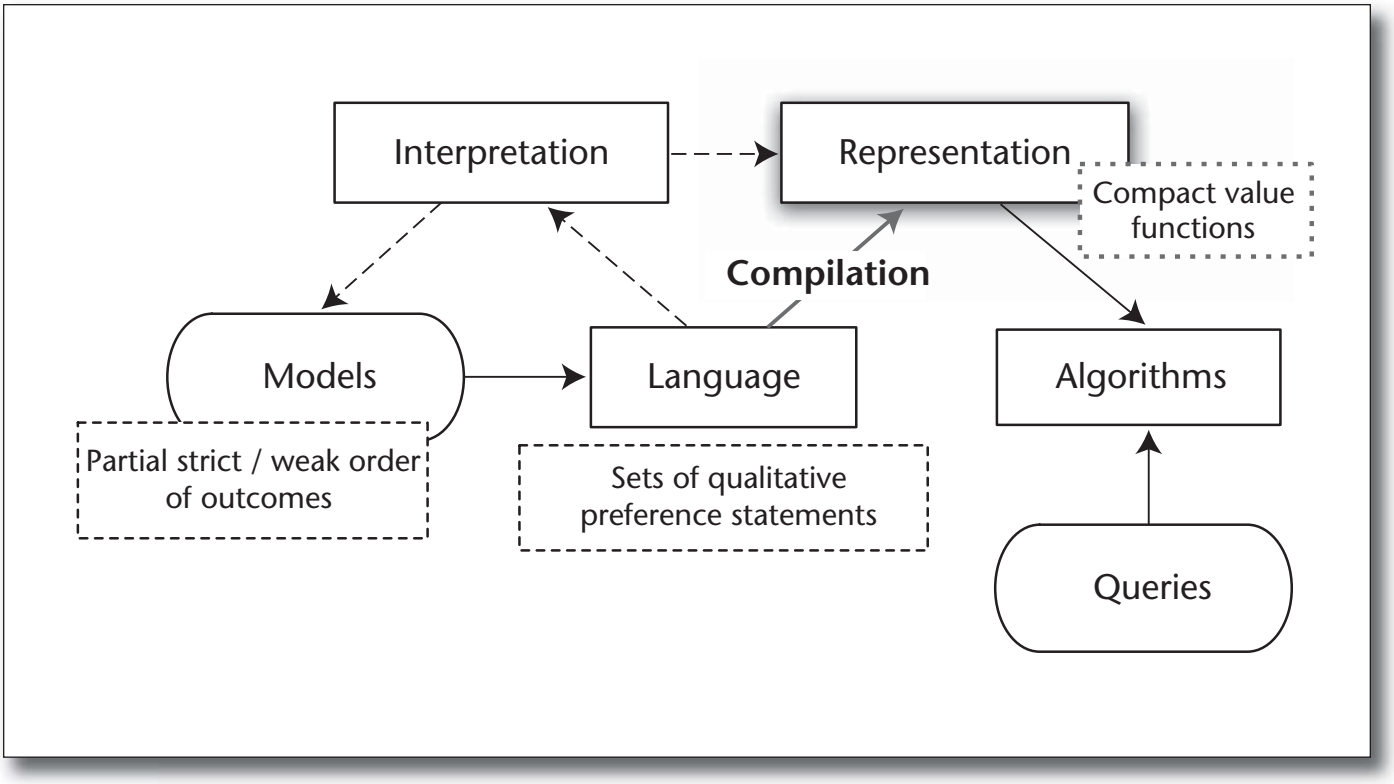

Figure 7. Preference Compilation.

are committing to one of many total orders that extend the partial order described by the input; that is, we are making additional, not necessarily warranted, assumptions about the user's preference. Depending on the application and the types of queries we want to answer, this may or may not be appropriate. We return to this issue later in this tutorial.

In what follows we describe two classes of techniques for value function compilation. One is more canonical, following the classical works on conjoint measurement, whereas the other is based on more novel ideas, as well as on some computational techniques used extensively in the area of discriminative machine learning.

Structure-Based Compilation. One of the advantages of generalizing statements is that a single statement can express much information. Thus, with a compact structure, such as a CP-network, we can express much information about the ordering of outcomes. If we were to compile a set of such statements into an arbitrary value function, we would lose this conciseness—we might get a structure that is exponentially large in |**X**|, and therefore expensive to store and work with. Structure-based compilation attempts to maintain the beneficial compact structure of input statements in the resulting value function so that time and space complexity remains low. Another reason for maintaining a compact target function is that it possesses many independence properties. These properties seem desirable, as they are usually implicitly implied by the input statements.

The general paradigm of structure-based compilation works as follows. We would like to show that some class of statements can be mapped into some class of value functions and that this mapping can be done efficiently. Let's consider a very simple example to illustrate this idea. For that, we restrict ourselves to a simple language of unconditional preference statements over single attributes such as "I prefer red cars to blue cars." Next, we specify a target class of value functions, and in our case this will be the class of additively independent value functions. Note that this class of functions satisfies the desired goal of mapping into compact value functions—each additively independent value function requires only $O(n)$ space as we need to associate a numeric value with (and only with) every attribute value. For example, if we have an attribute *Color* with values *blue, red,* and *black,* we will have a function V_{color} associated with this attribute and we must specify $V_{color}(blue)$, $V_{color}(red)$, $V_{color}(black)$. Finally, we must specify a mapping from our preference expressions to the selected target class of functions. The mapping usually corresponds to mapping each statement to a set of constraints on the value function. For example, "I prefer red cars to blue cars" maps into $V_{color}(blue) < V_{color}(red)$. Once we generate the entire set of con-

straints, any values for the parameters that satisfy these constraints is a valid value function. For example $V_{color}(blue) = 10$; $V_{color}(red) = 15$ is a valid value function with respect to the sole statement, as noted.

At this point we want to show that this mapping works in general. That is, if we have a consistent set of statements S in the given language, then there exists an assignment to the appropriate set of parameters that yields a value function V representing the preference order induced by S.[5] For instance, in this particular case, it is trivial to show that given consistent orderings over the values of individual attributes, we can come up with an additively independent value function that is consistent with these orderings. In general, representation theorems of interest usually (1) define a concrete class of input statements; (2) define the structure of the resulting value function (as a function of some properties of the input statements), and show that it is not too large; (3) prove that, for consistent input expressions in the chosen class, there exist value functions consistent with these expressions and obeying the selected structure. Finally, we also need a compilation theorem, that is, a theorem that shows that this mapping process can be performed efficiently.

Naturally, there are more interesting results than the very simple example considered here. For example, Brafman and Domshlak (2008) show that it is possible to compile to compact GAI value functions rather complicated, heterogeneous expressions consisting of (1) conditional preference statements over single attributes, for example, "In sports car from model year 2008 and beyond, I prefer the color red to blue," (2) conditional statements of attribute importance, for example, "Given that I'm flying in business class at night, the airline choice is more important than the seating assignment," and (3) pairwise comparisons of concrete outcomes, for example, "Your car is better than mine." The compilation scheme for this language is already much more involved and thus we won't get into its detailed discussion. In general, however, the area of structure-based preference compilation is still largely unexplored, and we believe that many interesting results in this direction are still to be discovered.

Structure-Free Compilation. While the amount of concrete results on structure-based preference compilation is still limited, it appears that the structure-based approach has two important limitations in general. The first is that a particular target structure for the value function is assumed. As we know, such structure is intimately tied with the notion of independence. Thus, in essence, the structure-based approach attempts to augment the user expression with additional independence assumptions. These assumptions may not be valid. They may even be inconsistent with the user's statements. The second limitation is that our ability to provably model heterogeneous types of statements is limited. That is, the languages for which we have representation and compilation theorems are limited.

The two assumptions are closely related. Because we insist on a special structure with its related independence assumptions, we lose our modeling flexibility, and consequently the expressive power of input statements we can handle. However, we insisted on such structure for a reason—we wanted a compactly representable value function. The structure-free compilation approach removes the inherent independence assumptions of the structure-based approach while maintaining our ability to efficiently compute the value of each outcome. It does this by utilizing kernel-based methods, popular in machine learning (Muller et al. 2001). Of course, there is no entirely free lunch, and some other assumptions are hidden behind these technique—and we will try to understand them later on.

The basic idea behind structure-free compilation is as follows: starting with the original n attributes $\mathbf{X}$, we schematically define a much larger (exponential in n) space of new attributes. These attributes, called features in what follows, correspond to all possible combinations of attribute values. The key property of the high-dimensional space defined by these features is that *any* value function defined in terms of the original set of attributes has an additively independent decomposition over the new features. The value of an outcome corresponds to the dot product between a vector of weights, one associated with each feature, and the vector of feature values provided by the outcome. Computing this dot product explicitly is infeasible, as these vectors' length is exponential in n. However, under certain conditions, dot products can be computed efficiently using kernal functions, and this is exactly what is exploited here. We now provide a more detailed explanation.

The structure-free approach works by taking a first step that is rather unintuitive—it maps the original problem where we work with n given attributes into a problem in which we have $\exp(n)$ attributes. Assuming here that the attributes $\mathbf{X}$ are all binary valued, the outcomes Ω described in terms of $\mathbf{X}$ are schematically mapped into a space $\mathcal{F} = \Re^{4n}$ using a certain mapping $\Phi : \Omega \mapsto \mathcal{F}$. The mapping Φ relates the attributes $\mathbf{X}$ and dimensions of $\mathcal{F}$ as follows. Let $\mathbf{F} = \{\mathfrak{f}_1, \ldots \mathfrak{f}_{4n}\}$ be a labeling of the dimensions of $\mathcal{F}$, and $\mathcal{D} = \cup_{i=1\ldots n} Dom(X_i) = \{x_1, \overline{x_1}, \ldots x_n, \overline{x_n}\}$; be the union of attribute domains in $\mathbf{X}$. Let val : $\mathbf{F} \rightarrow 2^{\mathcal{D}}$ be a bijective mapping from the dimensions of $\mathcal{F}$ onto the power set of $\mathcal{D}$, uniquely associating each dimension $\mathfrak{f}_i$ with a subset $\text{val}(\mathfrak{f}_i) \subseteq \mathcal{D}$, and vice versa. Let $\text{Var}(\mathfrak{f}_i) \subseteq \mathbf{X}$ denote

the subset of attributes "instantiated" by val(f_i). For example, if val(f_i) = {x_2, $\overline{x_3}$, x_{17}}, then Var(f_i) = {X_2, X_3, X_{17}}. Given that, for each $\mathbf{x} \in \mathcal{X}$ and $f_i \in \mathbf{F}$,

$$\Phi(\mathbf{x})[i] = \begin{cases} 1, & \text{val}(f_i) \neq \theta \wedge \text{val}(f_i) \subseteq \mathbf{x} \\ 0, & \text{otherwise} \end{cases}$$

That is, geometrically, Φ maps each n-dimensional vector $\mathbf{x} \in \mathcal{X}$ describing an outcome to the 4n-dimensional vector in $\mathcal{F}$ that uniquely encodes the set of all projections of x onto the subspaces of $\mathcal{X}$. For example, suppose that we have three attributes describing cars: *Color* (*red, blue*), *Category* (*sedan, minivan*), *New* (*new, used*). In this case, the features f_i will correspond to *red, red* $\wedge$ *new, sedan* $\wedge$ *used,* and so on, and we would map a car (*red, used, sedan*) into a truth assignment to features that assigns true to the features associated with {*red*}, {*used*}, {*sedan*}, {*red, used*}, {*red, sedan*}, {*used, sedan*}, {*red, used, sedan*}.

Let us now consider the compilation of preference expressions in terms of constraints on additively independent value functions in the new space $\mathcal{F}$. Assuming here that the referents of the statements in S correspond to propositional logic formulas over $\mathbf{X}$, consider an arbitrary comparative statement $\varphi \succ \psi$. Let $\mathbf{X}_\varphi \subseteq \mathbf{X}$ (and similarly $\mathbf{X}_\psi$) be the variables involved in φ, and $M(\varphi) \subseteq Dom(\mathbf{X}\varphi)$ be the set of all φ's models in the subspace of Ω defined by $\mathbf{X}_\varphi$. For instance, if $\mathbf{X} = \{X_1, \ldots, X_{10}\}$, and $\varphi = X_1 \vee X_2$, then $\mathbf{X}_\varphi = \{X_1, X_2\}$, and $M(\varphi) = \{x_1x_2, \overline{x_1}\, x_2, x_1\, \overline{x_2}\}$. Given that, each statement $\varphi \succ \psi$ is compiled into a set of $|M(\varphi)| \times |M(\psi)|$ linear constraints

$$\forall \mathbf{m}_\varphi \in M(\varphi), \forall \mathbf{m}_\psi \in M(\psi).$$
$$\sum_{f_i : \text{val}(f_i) \in 2^{\mathbf{m}_\varphi}} w_i > \sum_{f_j : \text{val}(f_j) \in 2^{\mathbf{m}_\psi}} w_j$$

where $2^{\mathbf{m}}$ denotes the set of all nonempty value subsets of the local model m. For example, statement $(X_1 \vee X_2) \succ (\neg X_3)$ (for example, "It is more important that the car is powerful or fast than not having had an accident") is compiled into

$$w_{x1} + w_{x2} + w_{x1x2} > w_{\overline{x3}}$$
$$w_{x1} + w_{\overline{x2}} + w_{x1\overline{x2}} > w_{\overline{x3}}$$
$$w_{\overline{x1}} + w_{x2} + w_{\overline{x1}x2} > w_{\overline{x3}}$$

Putting together such constraints resulting from compiling all the individual preference statements in the expression, we obtain a constraint system $\mathfrak{C}$ corresponding to the entire expression S. At first view, solving $\mathfrak{C}$ poses numerous complexity issues. First, though this constraint system is linear, it is linear in the exponential space $\mathfrak{R}^{4n}$. Second, the description size of $\mathfrak{C}$, and, in fact, of each individual constraint in $\mathfrak{C}$, can be exponential in n. Interestingly, it turns out that these complexity issues can be overcome by using some duality techniques from optimization theory (Bertsekas, Nedic, and Ozdaglar 2003) and reproducing kernel Hilbert spaces (RKHS) (Kimeldorf and Wahba 1971). The bottom line is that these techniques allow for computing a solution for $\mathfrak{C}$ (corresponding to the desired value function $V : \mathcal{F} \mapsto \mathfrak{R}$), and subsequently computing the values of V on the outcomes Ω, without requiring any explicit computation in $\mathfrak{R}^{4n}$.[6]

Further Reading

The idea of finding factored value functions that represent some preference relation is discussed at length in Keeney and Raiffa (1976). Conjoint measurement theory (Green and Rao 1971; Green, Krieger, and Wind 2001) specifically considers the problem of generating a value function describing a user (or a population) preference based on answers to preference queries. Most of this work seeks an additive independent model, though some extensions exist (Bouyssou and Pirlot 2005). Compilation methods were investigated in AI by McGeachie and Doyle (2004), Domshlak and Joachims (2007), and Brafman and Domshlak (2008). Much recent work in the machine-learning community is devoted to rank learning problems, where we attempt to learn a ranking function on some input based on examples. See for example Crammer and Singer (2003), Radlinski and Joachims (2007), and Burges et al. (2005).

Uncertainty and Utility Functions

In our discussion so far, we assumed that our preferences were over outcomes that we can obtain with certainty. For example, when we ordered digital cameras (for example, based on their zoom, number of mega-pixels, and cost) our implicit assumption was that we actually have a choice among concrete cameras. This may sound completely obvious, but consider the following situation. My company offers a new perk—each week it will buy me a lottery ticket, but I have to choose which lottery to participate in (for example, California Lottery, Nevada, British Columbia, Upper Saxon, and so on). So I'm faced with the following decision problem: determine the most desirable lottery from a set of possible lotteries. Each lottery offers a certain set of possible prizes, and to make things simple, we assume that the odds for each prize are known. Alternatively, suppose that I'm spending the weekend in Las Vegas, and I love playing roulette-style games. I know that different hotels have different such games with different prizes and different odds, and I want to choose the best place to gamble.

Gambling and lotteries may sound like insignificant applications of preference theories. It turns out, however, that the concept of a lottery, or a gamble, is far more general than "real" lotteries.

Basically, a lottery represents a (possibly available to us) choice that can lead to various concrete outcomes, but where we do not know ahead of time which outcome will prevail. For example, consider the problem of selecting a digital camera. If the attributes we care about are indeed, megapixels, zoom, and cost, then no lottery is involved because we can directly obtain the outcome we select. But what if we also care about reliability, defined as length of time until the camera starts to malfunction? Needless to say that we cannot really select the value of this attribute. Instead, we might be able to associate with this attribute a probability distribution over its values. Similarly, what if what we care about in a flight is its actual (in contrast to the published!) arrival time. Again, we cannot select a concrete arrival time simply because we don't control it. What we could do is, for example, choose the airline based on its past record. Thus, a flight is a lottery over arrival times (among other things), and by selecting different flights we are selecting different lotteries. Similarly, consider an actuation command sent to one of NASA's rovers on Mars. We don't really know ahead of time how it will affect the rover. This depends on the complex dynamics of the rover itself, the terrain, and the current weather conditions on Mars. None of these parameters are known precisely. Hence, when we choose to send this command, we're actually choosing a lottery over various concrete changes of the rover's state.

Thus, lotteries over concrete outcomes are a general model of choice under uncertainty, that is, the problem of choosing among actions with uncertain outcomes. But this begs the question: how can we work with preferences over such choices? Clearly, if we're able to order the set of possible lotteries, we're done. However, it's really unclear how even to approach this problem. This is where von Neumann and Morgenstern's theory of utilities (von Neumann and Morgenstern 1947) comes in. This seminal theory explains how we can associate a real value with each concrete outcome—its *utility*—such that one lottery is preferred to another if and only if the expected utility of its outcomes is greater than that of the other.

This result seems magical, and in some sense it is, but does come with a price. First, note that simply ordering the outcomes is not going to be enough. We will have to work harder and come up with something like a value function. However, whereas in a value function we really care only about the relative ordering of outcomes' values, in a utility function the actual value plays a significant role. In addition, the whole technique works only when the target ordering over lotteries satisfies certain properties. On the positive side, these properties make a lot of sense; that is, you would expect a "reasonable" decision maker to satisfy these postulates. On the negative side, it is well known that peoples' orderings often do not satisfy these properties, and a lot depends on framing issues. There's been a lot of work on this topic, especially by economists, who have attempted to come up with weaker or simply different properties (and there's a very nice and accessible textbook on this topic (Kreps 1988)). Here, however, we'll stick to the classical result.

Formally, our model here has the following elements:

Ω—Set of possible concrete outcomes

$\mathcal{L} = \Pi(\Omega)$—Set of possible lotteries (that is, probability distributions) over Ω

$L \subseteq \mathcal{L}$—Set of available lotteries over Ω (that is, possible choices or actions)

If $l \in \mathcal{L}$ and $o \in \Omega$, we use $l(o)$ to denote the probability that lottery l will result in outcome o. Our first step will be to assume that we seek an ordering over the entire family $\Pi(\Omega)$. This makes things simpler mathematically. The actual elicitation technique is affected by the size of Ω only.

Our second step is to define the notion of a complex lottery. Imagine that my employer has now added a new metalottery to its list of offered lotteries. In this lottery, we win lottery tickets. Say, with probability 0.4 we get a California lottery ticket denoted c, and with probability 0.6 we get a Nevada lottery ticket denoted n. A fundamental assumption we're going to make is that this new complex lottery, denoted $0.4c + 0.6n$, is equivalent to a simple lottery. For example, suppose the California lottery gives out \$1,000,000 with probability 0.1, and nothing otherwise, while the Nevada lottery gives out \$2,000,000 with probability 0.05, and nothing otherwise. We can transform the new complex lottery into a simple lottery by looking at the joint distribution. Thus, the probability that we get \$1,000,000 is 0.1 * 0.4 = 0.04; the probability that we get \$2,000,000 is 0.05 * 0.6 = 0.03 and the probability that we get nothing is 0.93. Next, we present the three properties that preference over lotteries must satisfy for the von Neumann Morgenstern theorem to hold:

Axiom 1. Order. $\succeq$ is a Total Weak Order. For every $l, l' \in$ L at least one of $l \succeq l'$ or $l' \succeq l$ holds.

Axiom 2. Independence/Substitution. For every lottery p, q, r and every $a \in [0, 1]$ if $p \succeq q$ then $ap + (1 - a)r \succeq aq + (1 - a)r$.

Axiom 3. Archimedean/Continuity. If p, q, r are lotteries s.t. $p \succ q \succ r$ then $\exists a, b \in (0, 1)$ such that $ap + (1 - a)r \succeq q \succeq bp + (1 - b)r$.

Let's take a closer look at each one of these axioms. The first is pretty straightforward: We must agree that, in principle, any two lotteries are comparable. The second axiom is a bit harder to parse, yet it formalizes a very intuitive assumption about preferences over lotteries. Basically, the sec-

ond axiom says that if you prefer oranges at least as much as apples then, no matter how you feel about carrots, you'll like a lottery that gives you an orange with probability p and a carrot with probability $1 - p$ at least as much as a lottery that gives you an apple with probability p and a carrot with probability $1 - p$. Thus, in both lotteries we get a carrot with the same probability, but with the remaining probability, the first lottery gives us a more preferred outcome, and so we like it better.

To understand the third axiom imagine that you strictly prefer oranges to apples and apples to carrots. Imagine that we slightly "dilute" your oranges by making a lottery that gives you oranges with probability 1 – a and carrots with probability a. The axiom says that there is always (possibly very small, but still positive) such a dilution for which you still prefer this lottery to getting apples for sure. It is not surprising that this is called the continuity axiom. Similarly, if we take the carrots and only slightly improve them by giving you some chance of getting an orange, there is always such a prospect that is still worse than getting apples for sure. Intuitively, this axiom is implying that nothing can be too bad (for example, hell) or too good (for example, heaven) so that just a little bit of it could transform a good thing into an arbitrarily bad one, or a bad thing into an arbitrarily good one.

It turns out that these three axioms together buy us a lot. Specifically, the von Neumann Morgenstern theorem states that a binary relation over $\mathcal{L}$ satisfies axioms 1–3 if and only if there exists a function $U : \Omega \rightarrow \Re$ such that

$$p \succeq q \Leftrightarrow \sum_{o \in \Omega} U(o)p(o) \geq \sum_{o \in \Omega} U(o)q(o)$$

Moreover, U is unique up to affine (= linear) transformations.

U is called a *utility function,* and the aforementioned condition says that a lottery p is at least as preferred as a lottery q, iff the expected utility of the possible outcomes associated with p is as high as the expected utility of the outcomes associated with q.

Once again, the metamodel instantiated for the setup of preferences over lotteries is depicted in figure 8. Note that, in practice, the existence of the utility function should be somehow translated to a method for obtaining that function. One way to do that is as follows. First, we order the set of concrete outcomes. Because utilities are unique up to affine transformations, we can fix an arbitrary value for the best and worst outcomes. We'll use 1 for the best and 0 for the worst. Next, we need to assign a value to every other outcome. Let o_b denote the best outcome and o_w denote the worst outcome. Consider an arbitrary outcome o. The utility of o is given by the value p determined in response to the following question: For what value p is it the case that you are indifferent between getting o for sure and a lottery in which you obtain o_b with probability p and o_w with probability $1 - p$.

For example, suppose that we're ordering food. Restaurants are characterized by two attributes: whether they sell junk food or healthy food, and whether the food is spicy or not. Here are the steps we take to generate the utility function:

Step One: Order outcomes best to worst: (unspicy, healthy) $\succeq$ (spicy, junk food) $\succeq$ (spicy, healthy) $\succeq$ (unspicy, junk food)

Step Two: Assign utility to best and worst outcomes: U(unspicy, healty) : = 1, U(unspicy, junk food) : = 0,

Step Three: Ask for values p and q such that (a) (spicy, healthy) ~ p(unspicy, healthy) + (1 – p)(unspicy, junk food) and (b) (spicy, junk food) ~ q(unspicy, healthy) + (1 – q)(unspicy, junk food) 4. Assign: U(spicy, healthy) : = p, U(spicy, junk food) : = q.

Although we see that there is a clear methodology for assigning utilities to outcomes, it is apparent that we can't expect lay users to be able to come up with these answers without much help. Indeed, specifying a utility function is much harder, cognitively, than specifying an ordering, and direct utility elicitation is not likely to be very useful in online setting. Current research in the area attempts to cope with this limitation in at least three different ways. First, it is possible to use here, as well, the structural assumptions such as generalized additive independence to decompose the specification of complex utility functions. Second, by using data from previous users, one can learn about typical utility functions and later calibrate them using information about the current user. Finally, one can exploit knowledge about the current decision problem to characterize only the information about the user's utility function required to address the current decision problem. We shall consider some of these issues in the next section.

Further Reading

The theory of expected utility appeared in Jon von Neumann and Oscar Morgenstern's seminal book *Theory of Games and Economic Behavior* (1947). Since then, many economists have tried to provide similar axiomatizations of choice behavior and this topic is discussed by Kreps (1988) and Fishburn (1982). One of the most beautiful results in this area is presented in Leonard Savage's *The Foundation of Statistics* (1972). Axiomatizations are important because they expose the basic assumptions of a model. We can also check the validity of a model by verifying, directly or indirectly, whether a user's preferences satisfy the relevant set of axioms. In particular, experiments conducted by

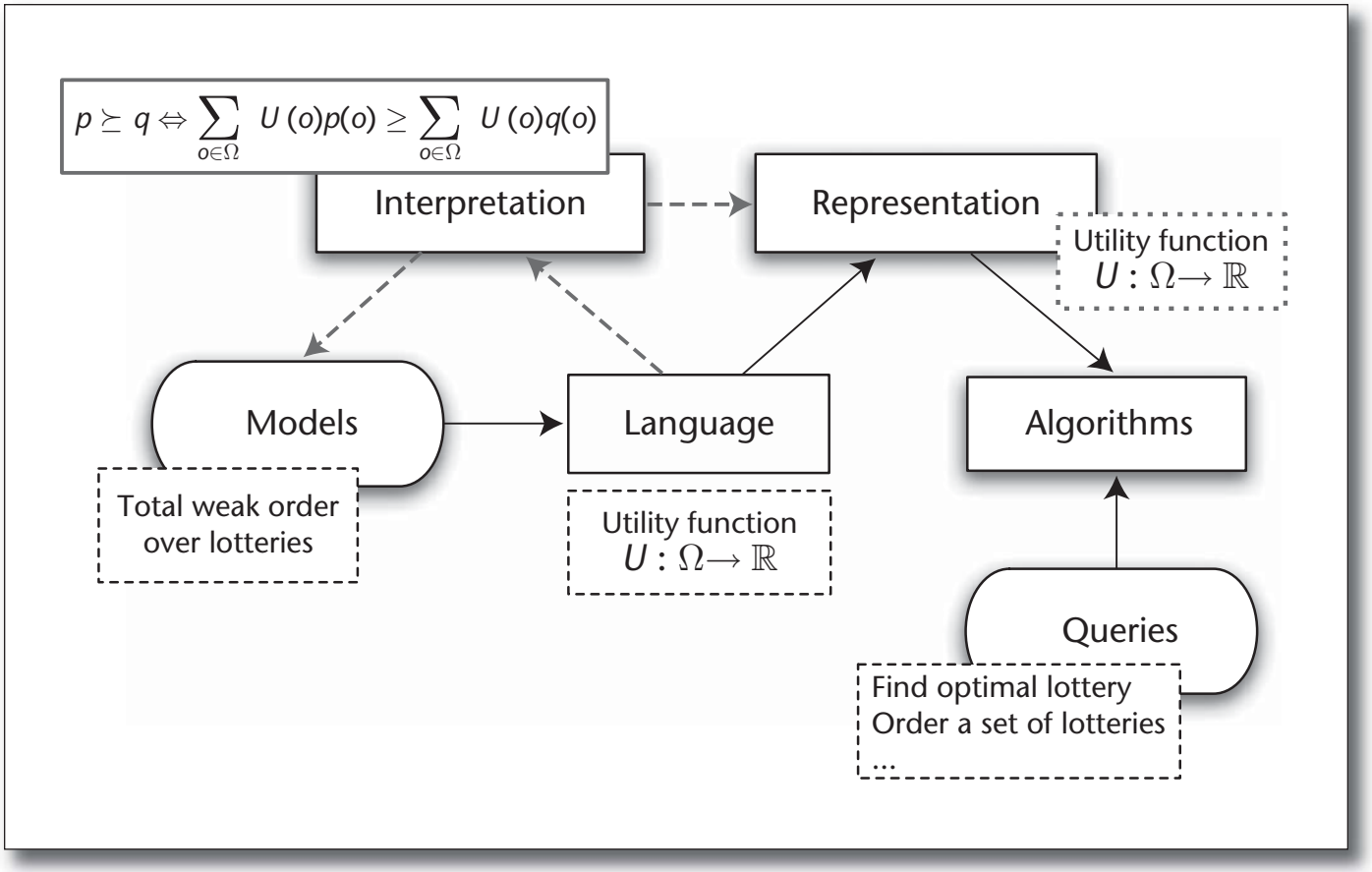

Figure 8. Metamodel for Utility Functions.

behavioral economists show that people's behavior often violates the von Neumann and Morgenstern axioms (Kahneman and Tversky 1979).

Preference Elicitation

Our discussion so far has focused on models and languages for representing preferences. Some of the languages discussed were motivated by the need to make the preference specification process more rapid and more accessible to lay users. However, in many problem domains we can't expect to obtain perfect information about the user's preferences because of the time and effort involved. Thus, the first question that arises is what can we do with partial information. Next, given that we can expect only partial information, we would like to obtain the most useful information possible given the user's time and effort limitations. Thus, the second question is how to make the elicitation process effective.

Working with Partial Specifications

The abstract problem of working with a partial specification has close similarities to the basic problem of statistical machine learning, in particular to classification. In classification problems, we must classify objects based on partial knowledge of the true model underlying the classification. In the standard setting of the classification problem, this partial knowledge corresponds to a set of properly classified objects from the space of all possible objects. Given such partial knowledge, the process of classification typically boils down to inferring either a single classification model, or a (possibly weighted) set of such models, from the space of all models in the chosen model class. The latter set of possible models is called a *hypothesis space*. Our situation is somewhat similar—our hypothesis space corresponds to the set of all possible preference model instances. For example, if our models are total orders, then these would be all possible total orders over the relevant set of outcomes. Or if our model is some GAI value function with a particular set of factors, then the hypothesis space includes all possible factor values. We, too, are given only partial information, and have to make decisions based on this information only. However, our situation with respect to the partial information is not exactly the same as in the standard setting of statistical machine learning. The main difference stems from the form in which the partial

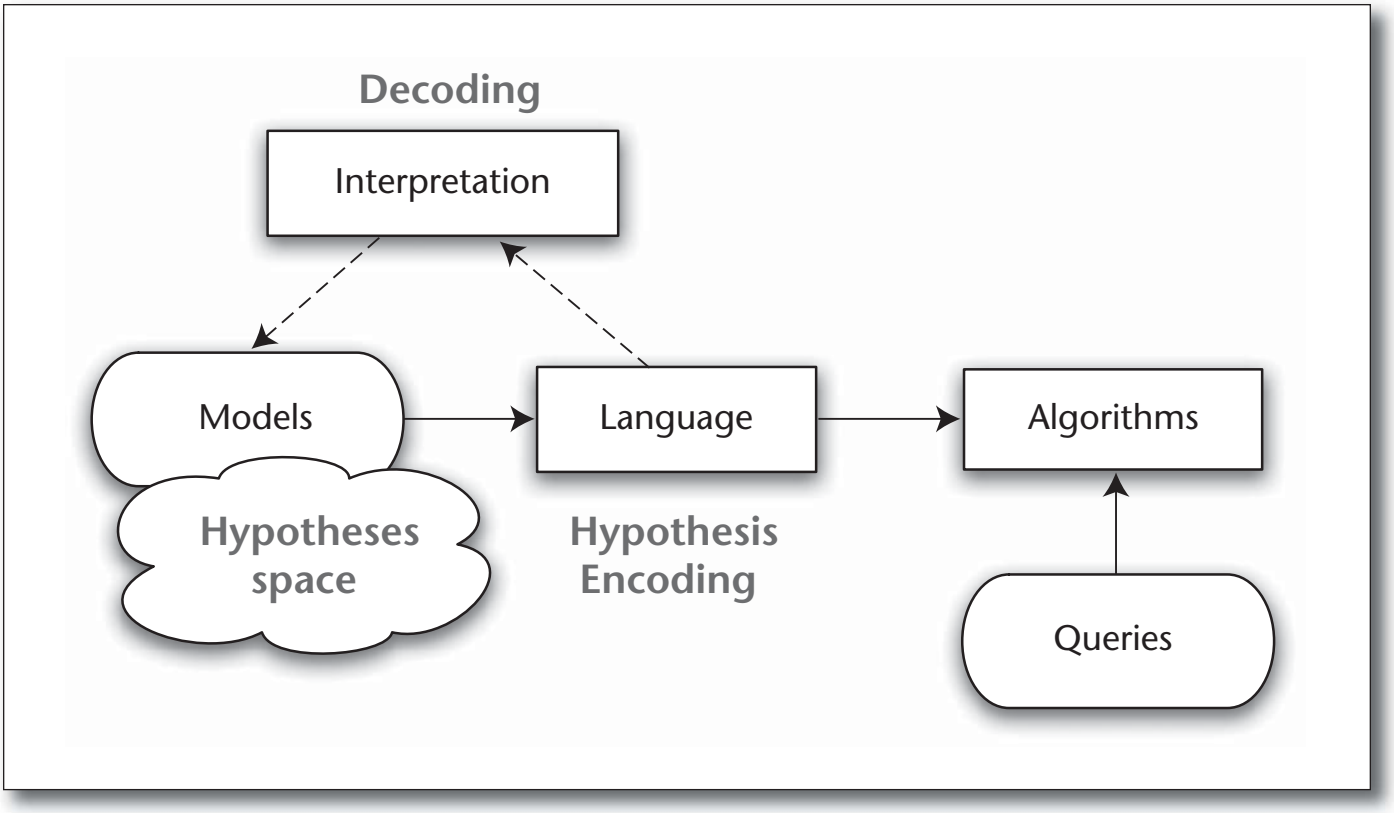

Figure 9. Model Selection Process.

knowledge is obtained. In statistical machine learning the partial knowledge is given by the examples of the concept to be learned, and the examples are assumed to be sampled from independent and identically-distributed random variables reflecting the topology of the object space. In contrast, our partial knowledge is typically given by a set of more global constraints on the original hypothesis space, with the constraints corresponding to some restrictions on values, some relationships between attractiveness of outcomes, generalizing statements, and so on. In that sense, while being very similar to the inference process in statistical machine learning, our inference process is closer to more general settings of inference via constraint optimization such as, for example, inference about probability distributions in Shore and Johnson (1980).

Figure 9 relates this discussion and the concepts involved to our metamodel. When we have a complete specification, the inference process leaves us with a single model, as shown in figure 10. However, in most practical situations, multiple models are consistent with the data, and we must specify how information is mapped into the model space. One of the basic choices we must make is whether a prior distribution is defined over the hypothesis space or not.

The Maximum-Likelihood Approach. Suppose we have a prior distribution over models. In many situations, this is quite natural. Consider, for example an e-commerce site visited by many consumers. As data from multiple users accumulates, the site can form a reasonable probabilistic model of user preferences. This distribution can be used as a prior over the hypothesis space. Now, two standard models for selecting model parameters can be used. In the maximum-likelihood method, we start with a prior distribution over the preference models and update this distribution using the user statements. Typically, the user statements simply rule out certain models, and the probability mass is divided among the remaining models. At this point, the most likely model consistent with the user statements is selected and used for the decision process.

For example, suppose that we are helping the user select a digital camera and the user cares only about two features: number of megapixels M, and its weight W. Suppose that our hypothesis class contains only additive value functions of the form $c_m \cdot M + c_w \cdot W$. We have a prior distribution $p_{m,w}$ over c_m and c_w. Suppose that all we know is that the user prefers some camera with parameters (m_1, w_1) to another camera with (m_2, w_2). From this we can conclude that $c_m m_1 + c_w w_1 > c_m m_2 + c_w w_2$, that is,

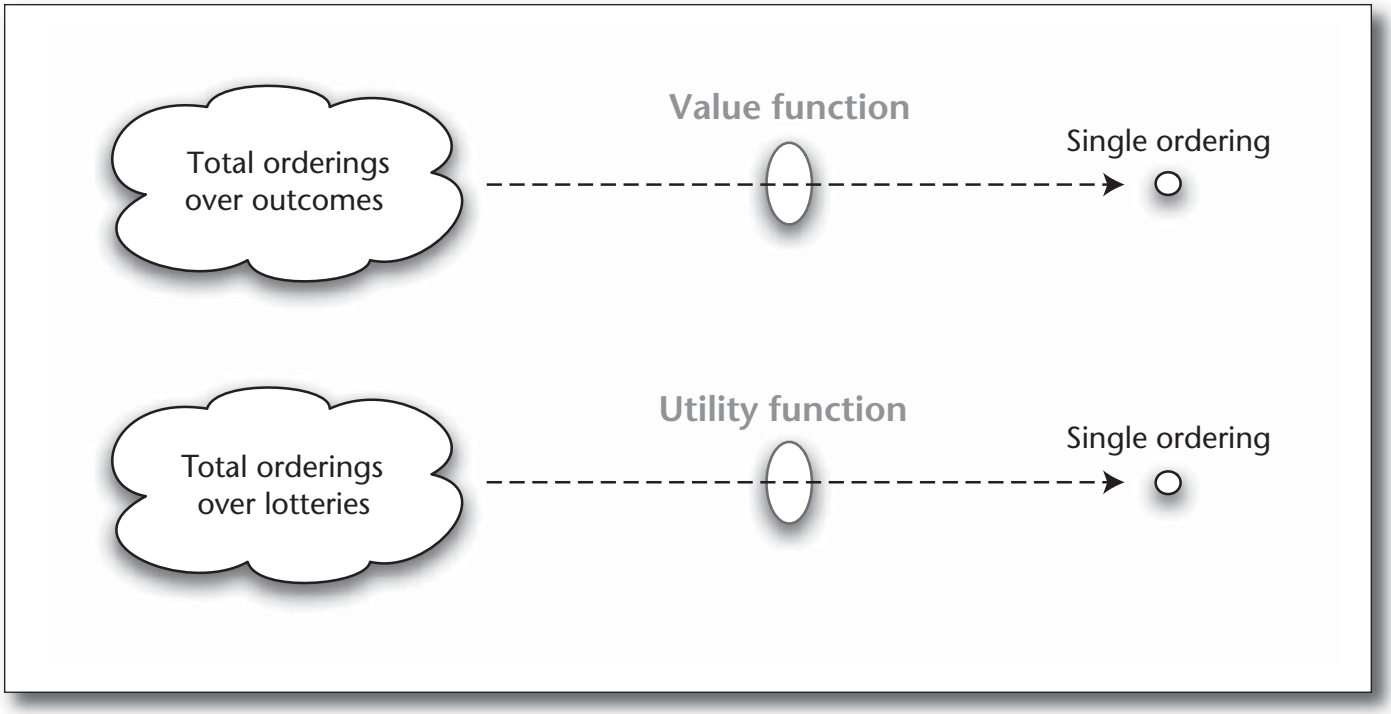

Figure 10. Selection With Full Information.

$c_m(m_1 - m_2) > c_w(w_2 - w_1)$. The value function selected according to the maximum-likelihood principle will have the coefficients c_m, c_w such that:

$$\left(c_m^*, c_w^*\right) = \text{argmax}_{(c_m, c_w)} \begin{array}{l} \left(p_{m,w}(c_m, c_w):\right. \\ c_m(m_1 - m_2) > \\ \left. c_w(w_2 - w_1)\right) \end{array}$$

Let's consider some of the models we discussed earlier in light of the maximum-likelihood approach, starting with CP-nets. One of the special features of CP-nets is that there is always a default model—namely, the model that satisfies all and only the preference relations derived from the input statements. Thus, in CP-nets, the prior distribution could be any distribution, while the posterior distribution over partial orders $\succ$ is

$$p(\succ) \sim \begin{cases} 1, & \succ \text{ assumes all and only} \\ & \text{all the information in } N \\ 0, & \text{otherwise} \end{cases}$$

Evidently, this is not a very interesting application of the maximum-likelihood principle.

Next, consider the structured value function compilation. Suppose that $p(V)$ is our prior distribution over value functions. Then, the posterior distribution is obtained by removing elements that are inconsistent with the constraints and renormalizing (that is, doing standard probabilistic conditioning). The situation with the structureless value compilation approach is the same, except that it requires a concrete prior

$$p(V) \sim -e^{\|wv\|^2}$$

where w is the vector of coefficients of a high-dimensional linear value function. The picture we have in both of the compilation approaches can be illustrated in figure 11. That is, we start with a large weighted hypothesis space (that is, a prior distribution over models), obtain information from the user that usually rules out some of the models, ending up with a subset of the original hypothesis space, from which we need to select a single model—the model with highest weight, possibly requiring some other selection criteria to break ties.

The Bayesian Approach. The Bayesian approach is identical to the maximum-likelihood approach, except that instead of selecting the most probable model and answering queries using it, this method maintains the entire posterior probability distribution and uses all models to answer a query, weighted by their respective probability. Thus, considering the picture above, we work with the entire intermediate purple set, but giving different weight to different elements.

As a simple example, let $\mathcal{O}$ be a set of total orders—our hypothesis space—and let $p : \mathcal{O} \to [0, 1]$ be the posterior distribution over $\mathcal{O}$ obtained after updating our prior with the user's input. Sup-

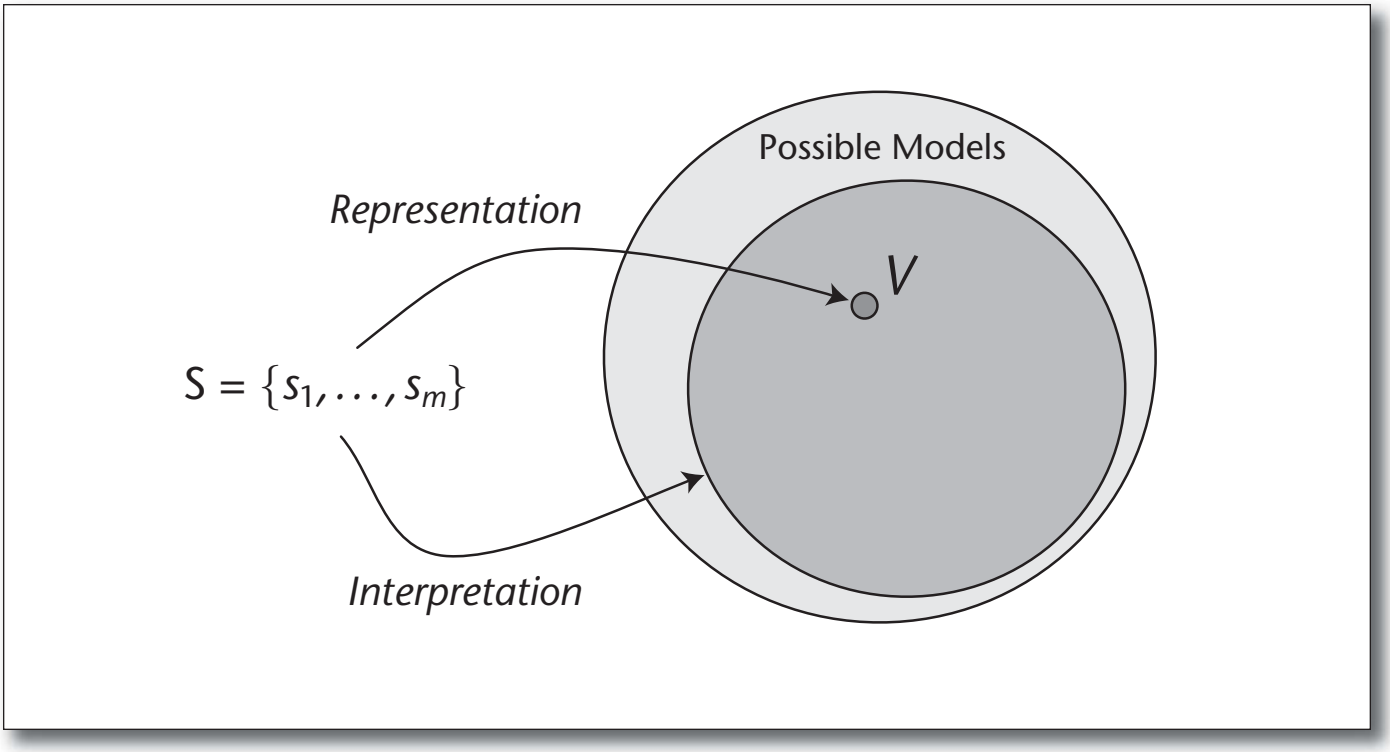

Figure 11. Model Selection in Value Function Compilation.

pose that we want to answer the query "$o_1 \prec o_2$?" We answer positively if $p(\{\prec \in \mathcal{O} : o_1 \prec o_2\}) > p(\{\prec \in \mathcal{O} : o_1 \not\prec o_2\})$. That is, we check whether p places more weight on models in which o_1 is less preferred than o_2 than on models in which this is not the case.

Although conceptually elegant, the Bayesian approach often requires the specification of another element to answer queries. The problem is that it is not always clear how to use the posterior probability to answer queries. For example, suppose we seek an optimal outcome. One approach we might take would be to assign to each outcome the sum of probabilities of all models in which this outcome is ranked highest, and return the outcome with the highest score. Thus, intuitively, the total value associated with an outcome is the probability that it is the best one.

Now, imagine that we have n possible outcomes, and our posterior model puts all of its weight on $n - 1$ equally likely different orderings. The top element in each of these orderings is different, but the second element is always the same outcome o. Thus, o's weight in this case would be 0, but intuitively it appears to be the best choice. To address this problem we can add a scoring function. The scoring function assigns a score to each possible answer within each model. The best answer is then the one with the highest expected score. In this example, we implicitly used a score of 1 for an the outcome ranked highest and 0 for all other outcomes. We could have used a different score, such assigning $n - j + 1$ to the outcome ranked jth, obtaining a different result.

Let's consider the earlier example of the purchase of a digital camera. Recall, we have a distribution $p_{m,w}$ over c_m and c_w, and a constraint due to our single observation that $c_m(m1 - m2) > c_w(w_2 - w_1)$. Let's use m_C and w_C to denote the number of megapixels and the weight of some camera C. The score associated with camera C in a particular model with coefficients c_m, c_w is $c_m m_C + c_w w_C$. Thus, the expected score of a camera is

$$\alpha \int_{\{c_m, c_w : c_m(m1-m2) > c_w(w_2-w_1)\}} p_{m,w}(c_m, c_w) \left(c_m m_C + c_w w_C\right) dc_m dc_w,$$

where α is some normalizing constant. Thus, the best camera is the one that maximizes this value, that is,

$$\arg\max_{c \text{ is a camera}} \int_{\{c_m, c_w : c_m(m1-m2) > c_w(w_2-w_1)\}} p_{m,w}(c_m, c_w) \left(c_m m_C + c_w w_C\right) dc_m dc_w.$$

A natural scoring function when our hypothesis class if utility functions is the utility of an outcome. Consider the case of a probability distribution over utility functions, and suppose we seek the optimal choice. Each choice is associated with

	V_1	V_2	V_3	V_4
o_1	1	4	2	2
o_2	2	3	4	3
o_3	3	3	3	4
o_4	4	1	1	3

Table 1. Four Feasible Value Functions over Four Outcomes.

a value in each model—its expected utility within that model. But we also have a distribution over models; thus, we can use the expected expected utility, that is, replacing

$$p \succeq q \Leftrightarrow \sum_{o \in \Omega} U(o)p(o) \geq \sum_{o \in \Omega} U(o)q(o)$$

with

$$p \succeq q \Leftrightarrow \sum_{U} p(U) \sum_{o \in \Omega} U(o)p(o)$$
$$\geq \sum_{U} p(U) \sum_{o \in \Omega} U(o)q(o).$$

Regret-Based Scores. Although prior distributions over preference models are natural in some cases, they are difficult to obtain in others. Without them, we need to replace the notion of the "most probable model" by some notion of "best" model. One attractive way to do this is using the concept of *regret.*

Again, suppose that our target class is the class of value or utility functions. We have a set of possible candidates, as in the structure-based compilation approach, and we need to select one. Suppose the the user's true (and unknown) value function is *V,* and suppose that we select some function *V*.* How bad can this be? This depends on *V, V*,* and the query we are trying to answer. Suppose that we are trying to find the most preferred outcome. If both *V* and *V** have the same most preferred outcome, then we're very happy. But suppose that *o* is the most preferred outcome according to *V,* and *o** is the most preferred outcome according to *V*.* In that case, we're going to recommend *o** to the user, instead of *o.* How unhappy would this make the user? Well, the user values *o* at $V(o)$ and *o** at $V(o^*)$, so we could quantify the user's regret at getting *o** instead of *o* by $V(o) - V(o^*)$. Thus, we can also view this as the regret associated with selecting *V** instead of *V.*

As an example, consider table 1 of value functions for a domain with four possible outcomes. Suppose that V_1 is the true value function, but we select V_2. V_2 assigns the highest value to o_1. The true value of o_1 is 1. The true optimal value is 4, assigned to o_4. Our regret is thus 4 – 1 = 3.

When we select *V** as the value function, we don't know what the true value function *V* is, so we can't compute $V(o) - V(o^*)$. A cautious estimate would consider the worst case scenario. This is called the maximal regret. More formally, if $\mathcal{V}$ is the set of candidate value functions, then

$$\text{Regret}(V^*|\mathcal{V}) = \max_{V \in \mathcal{V}}(V(\text{argmax}_o V(o)) - V(\text{argmax}_{o'} V^*(o'))),$$

that is, the worst case across all choices of *V* of the difference between the value according to *V* of the best *V* outcome and the best *V** outcome. For instance, considering our example above we see that our maximal regret at having chosen V_2 would be if the true value function is V_1. Given the notion of regret, an obvious choice of a value function among the set of alternative possible values functions is the one that minimizes regret. Specifically, the minmax regret value function V_{mr} satisfies

$$V_{mr} = \text{argmin}_{V \in \mathcal{V}} \text{Regret}(V|\mathcal{V}).$$

In our example, V_3 and V_4 minimize max regret because we can see that the maximal regret associated with V_1, V_2, V_3, V_4 is 3, 3, 2, 2, respectively. Thus, V_4 would be a choice that minimizes regret. Notice that in our case, this is equivallent to saying that selecting outcome o_3 is the choice that minimizes regret. Indeed, rather than talk about the regret associated with a value function, we can simply (and more generally) talk about the regret associated with a particular answer to the query at hand. This regret is a function of the set of candidate value functions.

Regret-based criteria can be used in different contexts. When a probability distribution over possible models is not given, we can use minmax regret to select a single model or select an optimal answer. When a probability distribution over models is available, we still need to associate a score with each possible answer to a query. The maximal regret can be used to score an answer in a model, and expected regret can be used to rate different answers. Thus, in our above example, if we associate a distribution of (0.5, 0.1, 0.1, 0.3) with $\{V_1, V_2, V_3, V_4\}$, we can compute the expected regret associated with each outcome. For example, the regret associated with the choice o_1 is 3, 0, 2, 2. Taking expectation, we obtain 2.3.

Finally, a word of caution. Although minmax regret is an intuitive choice criterion, its semantics is problematic, as there is no clear meaning associated with measures of difference in value or utility,

except for special cases of what are known as measurable value functions (Krantz et al. 1971). Thus, it is best thought of as an intuitive heuristic method.

Preference Elicitation

Our discussion so far on preference specification took a user-driven perspective—the information flow was considered to be unidirectional: from the user to the system. Under the assumption of eventual acquisition of the complete model of user preferences there is no real difference between purely user-driven and mixed-initiative preference specification. This is not the case, however, with partial model acquisition, and in what follows we consider preference specification in that setting more closely.

Clearly, if we can get only part of the model, then which part we get is quite important. In fact, to answer certain queries accurately, a partial model suffices, provided it has the right information. Preference elicitation is a process driven by the system that aims at improving the quality of the information it has about the user's preferences while decreasing the user's cognitive burden as much as possible.

The main benefit of the system-driven setting is that questions can be asked sequentially and conditionally. The questions we ask at each point take into account the answers we received earlier, as well as the query. There are a number of ways to approach this problem. When, as often is the case, the task is to recognize an optimal outcome of interest, *k*-item queries are often used. In this case, the user is shown *k* alternative choices and is asked either to select the best one among them or to rank them. When the user selects an item out of *k,* he or she is basically giving an answer to $k - 1$ explicit comparisons (that is, telling us that the chosen outcome is at least as preferred as the $k - 1$ other outcomes). Each such answer eliminates all orderings inconsistent with these answers.

A simple way of implementing this approach is by considering all possible total orders as possible hypotheses, and then ruling them out as information is obtained. For example, if the choice is among five items $o_1, \ldots, o_5$ and $k = 2$, we simply ask the user to compare pairs of items. If, for example, the user indicates that $o_2 \succ o_3$ and $o_1 \succ o_4$, orderings of the form …, o_4, …, o_1, …. and …, o_3, …, o_2 …. are eliminated.

However, this is a slow process that is likely to require many iterations. A more efficient process utilizes a smaller hypothesis space, for example, one that is based on some GAI-decomposition. In that case, each query farther constrains the possible values of the value function factors, and some generalization is taking place; that is, one can rule out additional orderings. For example, suppose that a vacation choice depends on two attributes, location and facility, and I learned that the user prefers a spa in Barcelona to a spa in Madrid. Without farther assumptions, only orderings that are inconsistent with this choice can be ruled out. However, if we assume that the value function is additive, or if we assume preferential independence, then we can also rule out orderings in which a rented apartment in Madrid is preferred to an otherwise similar apartment in Barcelona.

The aforementioned process appears somewhat ad hoc, and a more principled approach is desirable. In particular, even if we decide to use *k*-item queries, it is not clear how to select which query to ask. This problem can be approached from a decision-theoretic perspective. As we noted earlier, preference elicitation is a sequential process. We start with some target query and (possibly empty) knowledge of the agent's preferences and we can ask the user various questions. Typically, we don't know ahead of time how many questions the users will agree to answer, and so we must have a response to the original query at each point.

To have a well-posed sequential decision problem we need to model the state of the system and to have a valuation for alternative states as a start. The most natural way to model the system's state is in terms of some beliefs over a set of possible preference ordering. This takes us back to issues we considered in the context of partial models. In that setting, too, we needed to represent a set of possible models, possibly with some probability distribution defined over them, and to assign a valuation to each such set. The same answers we gave there can be used here, too. Thus, the system's state could be modeled as a set of possible orderings or value functions. We can farther refine this model by introducing a distribution over preference models to capture our uncertainty. The value of a model depends on the query at hand, but again, considering the problem of optimal choice, we can use measures such as expected utility loss or maximal regret.

At this point, we can start using standard ideas from decision theory to analyze the problem of preference elicitation. Let's start with a context in which we can pose a single preference query. What is the most appropriate query? This problem is analogous to the problem of selecting an observation in a decision context. That is, we have some beliefs about the current state of the world and we need to make a decision. The value of this decision depends on the actual state of the world, yet before we make the decision, we can observe the value of some feature of the current state. Which feature should we observe?

The answer given in decision theory is that we should compute the value of information associated with each feature we could observe, and observe the feature with maximal value of information.

The value of information of a feature is a measure of how much "happier" we would be if we learned the value of this feature. Considering a setting where our beliefs are captured using a distribution p over the state of the world, the value of information of feature f is computed as follows: Let s denote some state of the world, and let $V(s, a)$ denote the value of doing a at state s. Recall that for us s would denote some preference model, and a would be a possible query, such as "what is the best item?" Given the distribution p, $V(p, a)$ is the expected value of a according to p, that is, $E_p[V(s, a)]$. Let a_p denote the action with the highest expected value, that is, $a_p = \text{argmax}_a E_p[V(s, a)]$. Now we can define the value of query q given belief state p. Suppose that q has k possible answers, $r_1, \ldots, r_k$. Let p_i denote the probability p conditioned on r_i. Let a_i denote the best action according to p_i. The value of information of query q is

$$VI(q) = \Sigma_i\, p(r_i)V(p_i, a_i) - V(p, a).$$

Thus, the value of a query is some aggregate of the value of the world after we get an answer to that query. The value of the world, in our case, would be how happy we would be making the best decision in that world. For example, when we assume a probability distribution over possible preference models, and we evaluate such a distribution in terms of its minmax regret value, then the best query is the one that will minimize the expected minmax regret value of the resulting state.

We illustrate these ideas using the example we used earlier. Consider the four possible value functions over four objects as in table 1. Our current beliefs are: $pr(V_1) = 0.5$, $pr(V_2) = 0.1$, $pr(V_3) = 0.1$, $pr(V_4) = 0.3$. The score we give each outcome in each state is simply its value. In our current state (as captured by our current beliefs) the outcomes have the following expected values: 1.7, 2.3, 3.3, 3.1. Thus, the best choice is o_3 with value 3.3. Suppose that we can ask a query of the form: "what is the value of outcome *o?*" and let's see what is the value of asking: "what is $v(o_4)$." There are 3 possible answers, with the following probabilities according to our current beliefs: 4 with probability 0.5, 1 with probability 0.2, and 3 with probability 0.3. If we get the answer "$v(o_4) = 4$" then we conclude that V_1 is the only possible value function. The best action at that point (that is, the best answer to the query) would be to choose o_4 with a value of 4. If we learn that "$v(o_4) = 1$" then our posterior belief will assign V_2 and V_3 probability of 0.5 each. At this point, the best action or answer would be o_2 with expected value of 3.5. Finally, if we learn that "$v(o_4) = 3$" we conclude that the value function is V_4 and choose o_3 and obtain a value of 4. We can now compute the expected value of our state following the observation of $v(o_4)$, that is, $0.5 \cdot 4 + 0.2 \cdot 3.5 + 0.3 \cdot 4 = 3.9$. Thus the value of query "what is $v(o_4)$" is $3.9 - 3.3 = 0.6$. You can verify for yourself that this query has the highest value of information (among queries of this structure).

In the aforementioned example we assumed a distribution over possible models. However, similar ideas (though perhaps less well founded) can be applied when do not use a prior. In that case, a natural metric is minimal regret loss. Asking, as above, for the value of o_4, if we learn that it is 4 or that it is 3, we know the precise value function. In that case, our regret is 0. If we learn that $v(o_4) = 1$, then both V_2 and V_3 are possible. In that case, our minimal regret is 1 (obtained by suggesting item o_2). Thus, recalling from an earlier discussion that our minimal regret when all four value functions were possible was 2, the minimal regret loss associated with the question "what is $v(o_4)$" is $2 - 1 = 1$.

Using the idea of value of information, we can decide which query to asks, but this answer has two weaknesses. It deals with a single query, rather than a sequence of queries, and it ignores the cognitive effort required to answer this query. For both problems we can use well-known solutions from decision theory. The second problem is relatively easy to deal with, provided we have some way of assigning cost to each query. Then, rather than talk about value of information, we can talk about net value of information (NVI), where

$$NVI(q) = VI(q) - \text{Cost}(q).$$

For the first problem there are two standard solutions. The first is to act greedily, or myopically, and always pose the query with maximal NVI. This myopic behavior might not lead to an optimal sequence. For example, suppose we have two queries to ask. There may be one query that has large *VI*, say 1, and high cost, say 0.4, but following which, most queries have little value, almost 0. On the other hand, there is a query with moderate *VI*, say 0.5, but no cost, following which we can pose another similar query. However, it is computationally relatively cheap to compute the myopic selection. The other option is to try to compute the optimal sequential choice. Typically, this requires knowing the number of queries ahead of time (although this can be overcome) and while an optimal querying strategy results, the computational costs are exponential in the number of steps.

Finally, there is a very elegant model that captures all the above considerations nicely, although it comes with a heavy computational price tag. According to this model, the problem of preference elicitation is best modeled using a partially observable Markov decision process (POMDP). A POMDP has four key elements: a set S of possible states of the world, a set A of possible actions, a set Ω of possible observations, and a reward function R. Intuitively, we are modeling a decision maker that at each decision point can select an action from A. This action affects the state of the world, but that

state is not directly observable to the agent. Instead, the agent can observe an element in Ω, which is a possibly noisy feature of the current state. Actions can have cost or generate rewards, and that cost or reward may depend on the state of the world in which they are executed. Because the agent cannot directly observe the state of the world, what it does observe induces a distribution over the state of the world, called its belief state.

The POMDP framework is really perfect for modeling the intricacies of the elicitation process. The possible states of the world correspond to possible preference models in our hypothesis space, for example, value functions, or value functions with a particular factorization. The actions can be divided into two types: queries, which model the queries we can ask the agent, and actions, which can be used to model the actions we will eventually take on behalf of that agent. These actions could model a final selection of an item of choice, or they could model more intricate choices that the system might face whose evaluation requires knowledge of the agent's value function. The observations correspond to responses the user might make to the system's queries. Finally, the reward function models the cost of some of the queries (capturing their cognitive cost, for example) and the possible value to the agent of actions such as selecting one particular item. Naturally, the value of an item depends on the agent's preferences, that is, on the current state of the world. We can also model the probability that the agent will no longer agree to answer questions by adjusting the state space. Once a POMDP model is formulated, there are standard techniques for solving it. However, exact solution methods are impractical, and approximate solution methods work well for only moderately large state spaces. Thus, to apply this idea, we will have to limit ourselves to a few hundreds of possible preference models, at least if we are to rely on the current state of the art.

Starting with an initial belief state, that is, a distribution over the preference models, the policy generated by solving the POMDP will tell us what query to ask next. In some precise sense, this is the best query we could ask at this point taking all issues into consideration. Once the agent responds to this query, our distribution over models will be updated, and again, we can use the model to decide on our next query.

A good preference elicitation strategy balances the expected effort required by the user with the expected quality of the final choice made by the system based on the user's responses. The POMDP model is the theoretically best motivated, but the other options can also lead to good strategies. Making these approaches practical is an ongoing research issue that involves many aspects of the problem. These start with modeling the initial belief state. This initial distribution is often obtained by learning the preferences of a large population of users. Next, there is the problem of representing this distribution compactly. In general, the number of possible preference models is very large, thus, some parametric model is desirable. Next, there is an issue of modeling the cognitive burden of each query. Finally, computational techniques for obtaining approximately optimal elicitation strategies are needed. For example, the preference elicitation POMDP has special structure and properties that could be exploited by the solution algorithm.

Further Reading

Preference elicitation and value of information have been first studied in the areas of decision analysis and psychology, where they remain a topic of great importance (Tversky 1972, Keeney and Raiffa 1976, Howard and Matheson 1984, French 1986). This line of research has been extended in artificial intelligence, with a focus on automating the process of preference elicitation (Ha and Haddawy 1997, 1999; Torrens, Faltings, and Pu 2002; Pu and Faltings 2004; Faltings, Torrens, and Pu 2004; Braziunas and Boutilier 2005; Payne, Bettman, and Johnson 1993; Smith and McGinty 2003). Casting preference elicitation for policy optimization as a properly defined decision process was first suggested in the papers by Chajewska, Koller, and Parr (2000); Chajewska et al. (1998), and then extended in Boutilier (2002), which suggested the POMDP-based formulation. Preference elicitation under the minmax-regret model selection criterion has been studied in Boutilier et al. (2006) and Braziunas and Boutilier (2007). Note that here our discussion is focused on handling user preferences in "single-agent" settings; for an overview of recent works on preference elicitation in multiagent settings such as in (combinatorial) auctions see Sandholm and Boutilier (2006).

Conclusion

The importance of preference handling techniques for many areas of artificial intelligence and decision support systems is apparent. This area poses conceptual challenges, cognitive challenges, computational challenges, and representational challenges. A large body of work on this topic has accumulated. But there is ample room for additional ideas and techniques. Indeed, aside from the classical work of von Neuman and Morgenstern and techniques in the area of conjoint measurement theory, which basically deal with eliciting the value of additive value functions, most of the ideas described in this tutorial have yet to filter to real-world applications.

In this context, it is important to recall our three

rough categories of applications. In the case of the online consumer world, we believe the technology for more sophisticated online sales assistants is ripe. Although it may perhaps not be universally applicable, we believe that there are markets in which more sophisticated online assistants would be highly appreciated. Naturally, many issues affect their acceptance beyond the sheer power of the preference elicitation technology they provide. In the case of application design, we believe that more work on tools and much education are required for ideas to filter through. Finally, in the area of decision analysis, some techniques for better elicitation of GAI utility functions, a well as qualitative techniques for preparatory analysis, could play an important role.

Acknowledgements

We would like to thank Alexis Tsoukiàs for useful discussions and detailed comments. Partial support for Ronen Brafman was provided by the Paul Ivanier Center for Robotics Research and Production Management, by the Lynn and William Frankel Center for Computer Science. Partial support for Carmel Domshlak was provided by BSF Award 2004216 of United States–Israel Binational Science Foundation. Both authors are supported by COST Action IC0602.

Notes

1. For readers familiar with *decision theory*, this term comes with some baggage, and so we will note that at this stage, we focus on choice under certainty.

2. One possibility is to elicit only a partial model and use it to answer queries. See Working with Partial Specifications.

3. Note that the term *preference query* denotes queries made to users regarding their preferences, while just *queries* denote the questions we wish to answer using the preference model.

4. When the CP-net is fully specified, that is, an ordering over the domain of each attribute is specified for every possible assignment to the parents, we know that a single most preferred assignment exists. When the CP-net is not fully specified, or when we have additional hard constraints limiting the feasible assignments, then a number of Pareto-optimal assignments may exist—that is, assignments o such that for $o \not\prec o'$ for any other feasible o'.

5. Note that, technically, a concrete representation theorem would require some definition of consistency at the level of the input statements.

6. Presenting the computational machinery here is simply infeasible, and thus the reader is referred to Domshlak and Joachims (2007).

References

Agrawal, R., and Wimmers, E. L. 2000. A Framework for Expressing and Combining Preferences. In *Proceedings of the ACM SIGMOD International Conference on Management of Data,* 297–306. New York: Association for Computing Machinery.

Arrow, K. J., and Raynaud, H. 1986. *Social Choice and Multicriterion Decision Making*. Cambridge, MA: The MIT Press.

Bacchus, F., and Grove, A. 1995. Graphical Models for Preference and Utility. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence,* 3–10. San Francisco: Morgan Kaufmann Publishers.

Bertsekas, D.; Nedic, A.; and Ozdaglar, A. 2003. *Convex Analysis and Optimization*. Nashua, NH: Athena Scientific.

Birkhoff, G. 1948. *Lattice Theory,* volume 25. Providence, RI: American Mathematical Society.

Bistarelli, S.; Fargier, H.; Montanari, U.; Rossi, F.; Schiex, T.; and Verfaillie, G. 1999. Semiring-Based CSPs and Valued CSPs: Frameworks, Properties, and Comparison. *Constraints* 4(3): 275–316.

Bistarelli, S.; Montanari, U.; and Rossi, F. 1997. Semiring-Based Constraint Solving and Optimization. *Journal of the ACM* 44(2): 201–236.

Boutilier, C. 1994. Toward a Logic for Qualitative Decision Theory. In *Proceedings of the Third Conference on Knowledge Representation* (KR–94), 75–86. San Francisco: Morgan Kaufmann Publishers.

Boutilier, C. 2002. A POMDP Formulation of Preference Elicitation Problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 239–246. Menlo Park, CA: AAAI Press.

Boutilier, C.; Bacchus, F.; and Brafman, R. 2001. UCP-Networks: A Directed Graphical Representation of Conditional Utilities. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence,* 56–64. San Francisco: Morgan Kaufmann Publishers.

Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; and Poole, D. 2004a. CP-nets: A Tool for Representing and Reasoning about Conditional Ceteris Paribus Preference Statements. *Journal of Artificial Intelligence Research* 21: 135–191.

Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; and Poole, D. 2004b. Preference-Based Constrained Optimization with Cp-Nets. *Computational Intelligence* (Special Issue on Preferences in AI and CP) 20(2): 137–157.

Boutilier, C.; Patrascu, R.; Poupart, P.; and Schuurmans, D. 2006. Constraint-Based Optimization and Utility Elicitation Using the Minimax Decision Criterion. *Artificial Intelligence* 170(8–9): 686–713.

Bouyssou, D., and Pirlot, M. 2005. Following the Traces: An Introduction to Conjoint Measurement without Transitivity and Additivity. *European Journal of Operational Research* 163(2): 287–337.

Bouyssou, D.; Marchant, T.; Pirlot, M.; Tsoukias, A.; and Vincke, P. 2006. *Evaluation and Decision Models with Multiple Criteria: Stepping Stones for the Analyst.* Berlin: Springer.

Brafman, R. I., and Domshlak, C. 2008. Graphically Structured Value-Function Compilation. *Artificial Intelligence* 172(2–3).

Brafman, R. I.; Domshlak, C.; and Shimony, S. E. 2006. On Graphical Modeling of Preference and Importance. *Journal of Artificial Intelligence Research* 25: 389–424.

Braziunas, D., and Boutilier, C. 2005. Local Utility Elicitation in GAI Models. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence,* 42–49. Arlington, VA: AUAI Press.

Braziunas, D., and Boutilier, C. 2007. Minimax Regret Based Elicitation of Generalized Additive Utilities. In *Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence,* 25–32. Arlington, VA: AUAI Press.

Brewka, G.; Niemela, I.; and Truszczynski, M. 2003. Answer Set Optimization. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*. San Francisco: Morgan Kaufmann Publishers.

Burges, C. J. C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. N. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of the International Conference on Machine Learning,* 89–96. New York: Association for Computing Machinery.

Chajewska, U.; Getoor, L.; Norman, J.; and Shahar, Y. 1998. Utility Elicitation As A Classification Problem. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence,* 79–88. San Francisco: Morgan Kaufmann Publishers.

Chajewska, U.; Koller, D.; and Parr, R. 2000. Making Rational Decisions Using Adaptive Utility Elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 363–369. Menlo Park, CA: AAAI Press.

Chen, L., and Pu, P. 2007. Preference-Based Organization Interface: Aiding User Critiques in Recommender Systems. In *Proceedings of the Eleventh International Conference on User Modeling,* 77–86. Berlin: Springer-Verlag.

Chomicki, J. 2002. Querying with Preferences Intristic. In P*roceedings of the Eighth International Conference on Extending Database Technology*, 34–51, LNCS 2287. Berlin: Springer.

Crammer, K., and Singer, Y. 2003. A Family of Additive Online Algorithms for Category Ranking. *Journal of Machine Learning Research* 3: 1025–1058.

Davey, B. A., and Priestley, H. A. 2002. *Introduction to Lattices and Order.* Cambridge, UK: Cambridge University Press.

Debreu, G. 1954. Representation of a Preference Ordering By A Numerical Function. In *Decision Processes,* ed. R. Thrall, C. Coombs, and R. Davis, 159–166. New York: John Wiley.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Domshlak, C., and Joachims, T. 2007. Efficient and Nonparametric Reasoning Over User Preferences. *User Modeling and User-Adapted Interaction* (Special issue on Statistical and Probabilistic Methods for User Modeling).17(1-2): 41–69.

Doyle, J. 2004. Prospects for Preferences. *Computational Intelligence* 20(2): 111–136.

Doyle, J., and Thomason, R. H. 1999. Background to Qualitative Decision Theory. *AI Magazine* 20(2): 55–68.

Doyle, J., and Wellman, M. 1994. Representing Preferences as Ceteris Paribus Comparatives. In Decision-Theoretic Planning: Papers from the AAAI Spring Symposium, 69–75, AAAI Technical Report SS-94-06. Menlo Park, CA: AAAI Press.

Engel, Y., and Wellman, M. P. 2006. CUI Networks: A Graphical Representation for Conditional Utility Independence. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Faltings, B.; Torrens, M.; and Pu, P. 2004. Solution Generation with Qualitative Models of Preferences. *International Journal of Computational Intelligence and Applications* 7(2): 246–264.

Fishburn, P. C. 1969. *Utility Theory for Decision Making*. New York: John Wiley & Sons.

Fishburn, P. 1974. Lexicographic Orders, Utilities, and Decision Rules: A Survey. *Management Science* 20(11): 1442–1471.

Fishburn, P. C. 1982. *The Foundations of Expected Utility.* Dordrecht, Holland: Reidel.

Fishburn, P. 1999. Preference Structures and Their Numerical Representations. *Theoretical Computer Science* 217(2): 359–383.

French, S. 1986. *Decision Theory.* New York: Halsted Press.

Gajos, K., and Weld, D. 2005. Preference Elicitation for Interface Optimization. In *Proceedings of the Eighteenth Annual ACM Symposium on User Interface Software and Technology* (UIST). New York: Association for Computing Machinery.

Goldsmith, J.; Lang, J.; Truszczynski, M.; and Wilson, N. 2005. The Computational Complexity of Dominance and Consistency In CP-nets. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence,* 144–149. Menlo Park, CA: AAAI Press.

Gonzales, C., and Perny, P. 2004. GAI Networks for Utility Elicitation. In *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning,* 224–234. Menlo Park, CA: AAAI Press.

Green, P. E., and Rao, V. R. 1971. Conjoint Measurement for Quantifying Judgmental Data. *Journal of Marketing Research* 8: 355–363.

Green, P. E.; Krieger, A. M.; and Wind, Y. 2001. Thirty Years of Conjoint Analysis: Reflections and Prospects. *Interfaces* 31(3): 56–73.

Ha, V., and Haddawy, P. 1997. Problem-Focused Incremental Elicitation of Multiattribute Utility Models. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence,* Providence, Rhode Island, 215–222. San Francisco: Morgan Kaufmann Publishers.

Ha, V., and Haddawy, P. 1999. A Hybrid Approach to Reasoning with Partially Elicited Preference Models. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence,* Stockholm, Sweden. San Francisco: Morgan Kaufmann Publishers.

Hallden, S. 1957. *On the Logic of Better.* Lund, Sweden: Gleerup.

Hansson, S. O. 2001a. Preference Logic. In *Handbook of Philosophical Logic,* volume 4, ed. D. M. Gabbay and F. Guenthner, second edition. 319–394. Dortrecht, Holland: Kluwer

Hansson, S. O. 2001b. *The Structure of Values and Norms*. Cambridge, UK: Cambridge University Press.

Howard, R. A., and Matheson, J. E. 1984. *Readings on the Principles and Applications of Decision Analysis*. Menlo Park, CA: Strategic Decision Group.

Kahneman, D., and Tversky, A. 1979. Prospect Theory: An Analysis of Decisions Under Risk. *Econometrica* 47(2): 313–327.

Kahneman, D., and Tversky, A. 1984. Choices, Values, and Frames. *American Psychologist* 39: 341–350.

Keeney, R. L., and Raiffa, H. 1976. *Decision with Multiple Objectives: Preferences and Value Tradeoffs*. New York: Wiley.

Kießling, W. 2002. Foundations of Preferences In Database Systems. In *Proceedings of 28th International Conference on Very Large Data Bases* (VLDB). San Francisco: Morgan Kaufmann Publishers.

Kimeldorf, G., and Wahba, G. 1971. Some Results on Tchebycheffian Spline Functions. *Journal of Mathematical Analysis and Applications* 33: 82–95.

Krantz, D. H.; Luce, R. D.; Suppes, P.; and Tversky, A. 1971. *Foundations of Measurement.* New York: Academic Press.

Kraus, S.; Lehmann, D.; and Magidor, M. 1990. Nonmonotonic Reasoning, Preferential Models, and Cumulative Logics. *Artificial Intelligence* 44: 167–207.

Kreps, D. M. 1988. *Notes on the Theory of Choice*. Boulder, CO: Westview Press.

La Mura, P., and Shoham, Y. 1999. Expected Utility Networks. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence,* Stockholm, Sweden, 367–373. San Francisco: Morgan Kaufmann Publishers.

Lauritzen, S. L., and Spiegelhalter, D. J. 1988. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems (with Discussion). *Journal of Royal Statistical Society*, Series B 50.

McCarthy, J. 1980. Circumscription: A Form of Nonmonotonic Reasoning. *Artificial Inteligence* 13(1-2): 27–39.

McGeachie, M., and Doyle, J. 2004. Utility Functions for Ceteris Paribus Preferences. *Computational Intelligence* 20(2): 158–217. (Special Issue on Preferences in AI).

Muller, K. R.; Mika, S.; Ratsch, G.; Tsuda, K.; and Scholkopf, B. 2001. An Introduction to Kernel-Based Learning Algorithms. *IEEE Neural Network*s 12(2): 181–201.

Oztürk, M.; Tsoukiàs, A.; and Vincke, P. 2005. Preference Modeling. In *Multiple Criteria Decision Analysis: State of the Art Surveys,* ed. J. Figueira, S. Greco, and M. Ehrgott, 27–72. Berlin: Springer Verlag.

Payne, J.; Bettman, J.; and Johnson, E. 1993. *The Adaptive Decision Maker.* Cambridge, UK: Cambridge University Press.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.

Pu, P., and Faltings, B. 2004. Decision Tradeoff Using Example Critiquing and Constraint Programming. *Constraints: An International Journal* 9(4): 289–310.

Radlinski, F., and Joachims, T. 2007. Active Exploration for Learning Rankings From Clickthrough Data. In *Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 570–579. New York: Association for Computing Machinery.

Reiter, R. 1980. A Logic for Default Reasoning. *Artificial Inteligence* 13: 81–132.

Rossi, F.; Venable, K. B.; and Walsh, T. 2004. mCP Nets: Representing and Reasoning with Preferences of Multiple Agents. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence,* 729–734. Menlo Park, CA: AAAI Press.

Sandholm, T., and Boutilier, C. 2006. Preference Elicitation in Combinatorial Auctions. In *Combinatorial Auctions*, ed. P. Cramton, Y. Shoham, and R. Steinberg, chapter 10, 233–264. Cambridge, MA: MIT Press.

Savage, L. 1972. *The Foundations of Statistics*, 2nd ed. New York: Dover.

Shoham, Y. 1987. A Semantics Approach to Nonmonotonic Logics. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 388–392. Los Altos, CA: William Kaufmann, Inc.

Shoham, Y. 1997. A Symmetric View of Probabilities and Utilities. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1324–1329.

Shore, J. E., and Johnson, R. W. 1980. Axiomatic Derivation of the Principle of Maximum Entropy and the Principle of Minimum Crossentropy. *IEEE Transactions on Information Theory* 26(1): 26–37.

Smith, B., and McGinty, L. 2003. The Power of Suggestion. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence,* 127–132. San Francisco: Morgan Kaufmann Publishers.

Tan, S. W., and Pearl, J. 1994. Qualitative Decision Theory. In *Proceedings of the Twelfth National Conference on Artificial Intelligence, 9*28–933. Menlo Park, CA: AAAI Press.

Torrens, M.; Faltings, B.; and Pu, P. 2002. SmartClients: Constraint Satisfaction as a Paradigm for Scaleable Intelligent Information Systems. *Constraints* 7(1): 49–69.

Tversky, A. 1967. A General Theory of Polynomial Conjoint Measurement. *Journal of Mathematical Psychology* 4(1): 1–20.

Tversky, A. 1969. Intransitivity of Preferences. *Psychological Review* 76: 31–48.

Tversky, A. 1972. Elimination by Aspects: A Theory of Choice. *Psychological Review* 79: 281–299.

von Neumann, J., and Morgenstern, O. 1947. *Theory of Games and Economic Behavior*, 2nd ed.. Princeton, NJ: Princeton University Press.

von Wright, G. H. 1963. *The Logic of Preference: An Essay.* Edinburgh, Scotland: Edinburg University Press.

Wald, A. 1950. *Statistical Decision Functions*. New York: John Wiley.

Wilson, N. 2004. Extending CP-Nets with Stronger Conditional Preference Statements. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence,* 735–741. Menlo Park, CA: AAAI Press.

Ronen Brafman is an associate professor at the Department of Computer Science at Ben-Gurion University in Israel. He received his Ph.D. in computer science from Stanford University in 1996 and was a postdoctoral fellow at the University of British Columbia. His research work focuses on various aspects of decision making and decision support, including preference handling, classical and decision theoretic planning, and reinforcement learning. He serves as an associate editor for the *Journal of AI Research* and is a member of the editorial board of the *Artificial Intelligence Journal*.

Carmel Domshlak is a senior lecturer at the Faculty of Industrial Engineering and Management in Technion. His research interests are in modeling and reasoning about preferences, automated planning and reasoning about action, and knowledge-base information systems. He received his Ph.D. in computer science from Ben-Gurion University in 2002 for his work on preference representation models and was a postpdoctoral fellow at the Intelligent Information Systems Institute at Cornell University. He is a member of the editorial board of the *Journal of AI Research*.