Preference Representation with Weighted Goals: Expressivity, Succinctness, Complexity

Joel Uckelman* and **Ulle Endriss**
ILLC, University of Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam, The Netherlands
{juckelma,ulle}@illc.uva.nl

Abstract

The representation of preferences of agents is a central feature in many AI systems. In particular when the number of alternatives to be considered may become large, the use of compact preference representation languages is crucial. The framework of weighted propositional formulas can be used to define several such languages. The central idea is to associate numerical weights with goals specified in terms of propositional formulas, and to compute the utility value of an alternative as the sum of the weights of the goals it satisfies. In this paper, we analyze several properties of languages defined by weighted goals: their expressivity, the relative succinctness of different sublanguages, and the computational complexity of finding the best alternative with respect to a given utility function expressed in terms of weighted goals.

Introduction

Many problems in AI require the representation of and reasoning about the preferences of agents. The alternatives over which agents need to express a preference often have a combinatorial structure. This can, for example, be observed in resource allocation with indivisible goods and in combinatorial auctions (Cramton, Shoham, & Steinberg 2006). The combinatorial nature of such problems makes succinct representation of agent preferences important (Coste-Marquis *et al.* 2004; Lang 2004; Chevaleyre, Endriss, & Lang 2006; Nisan 2006).

To see the importance of preference representation languages, consider bidding in a combinatorial auction. A combinatorial auction is one where bidders may place bids on bundles, and not just single goods. In the most general case, each bidder may submit a bid for each subset of the set of items on auction, and the items are distributed according to some criterion, such as maximizing the revenue for the auctioneer. Already, this description presents two problems: First, the number of subsets of goods is exponential in the number of goods, which is a problem for both the bidders and for the auctioneer; and second, the award criterion is a maximization problem, which is a problem for the auctioneer. In the first case, bidders are faced with having to examine each of exponentially many subsets in order to assign values to them, and, should the bidders succeed at that, the auctioneer must then attempt to use this mountain of data to determine the outcome of the auction. It will be little consolation for the auctioneer to have an efficient winner-determination algorithm in this case, as his input will be exponential! Neither expressing nor computing with explicit representations is feasible for any but the smallest sets of items. In the second case, that of the award criterion, we are usually charged with accepting bids so as to maximize some quantity, be it auctioneer revenue, social welfare, or something else. Explicit representations of bids do not lend themselves to efficient computation, for reasons already stated: They are too long.

Of course, the story changes radically when we no longer insist on explicit representations of bids (or, more generally, of utility functions). An agent's total explicit bid, composed of an atomic bid for each subset, may well have structure which we can exploit for devising a shorter representation. A trivial example, but one which illustrates the point, is that of the bidder who gives a value of €1 to every nonempty set of goods. With n goods, an explicit representation of this bid—i.e., one which lists every subset with the value the bidder places on it—will of course have a length on the order of 2^n, but intuitively we could have a constant-length nonexplicit representation of this bid, as

$$\forall X \supset \emptyset,\ u(X) = €1$$

(or, as we just said in words, "the bidder gives a value of €1 to every nonempty set of goods"). The insight here is that bids may have internal structure which we can be exploit in order to produce concise, and therefore more manageable, representations of those bids.

The representation given in our example, while concise, is too *ad hoc* to be generally useful. If we permit any sort of mathematical expression for defining a bid, we may get short representations in most cases, but be faced with such syntactical variety that our computer will be unable to cope with them all. Hence, there is a need for concise representations, not just on their own, but *in a well-defined language*. Just as the structure of bids makes concise representations possible, the structure of the bidding language makes computation with the bids possible. For a summary of bidding languages, see (Nisan 2006); for a summary of winner determination algorithms, see (Lehmann, Müller, & Sandholm 2006).

*This research was supported by a Marie Curie Early Stage Research fellowship from the GloRiClass project (MESST-CT-2005-020841).

Here we study a particular class of languages for representing preferences: utility functions represented as sets of weighted goals, called goal bases. The central idea is to associate numerical weights with goals specified in terms of propositional formulas, and to compute the utility value of an alternative as the sum of the weights of the goals it satisfies, an idea which originates in penalty logic (Pinkas 1991; Lang 2004). The class of all goal bases (our sets of weighted goals) forms a bidding language. Further bidding languages may be formed by placing restrictions on the kinds of goal bases which are admissible. The structure of goal bases suggests certain sorts of restrictions, namely on the syntactical form of the goals and on the weights. It is important to understand the properties of these languages, both in their own right and so we may identify which languages have potential applications. Already the expressive power of many such languages are known, for which see (Chevaleyre, Endriss, & Lang 2006).

If we have a range of different languages available which can all represent the target class of utility functions, what further criteria should we use to discriminate among them? Our formalism permits the definition of a wide range of languages; the long-term goal is to determine which languages are best for which purposes. Several criteria suggest themselves:

- **Expressivity:** Not all goal base languages are equally expressive, nor do all applications require full expressivity. For example, it may be known beforehand that all bidders in an auction have decreasing marginal utility. Excess expressivity tends to be undesirable because highly expressive languages are computationally harder, so in such a situation we should like to use a bidding language which expresses just the utility functions bidders are likely to have. In order to make such a determination, we need to know what utility functions are expressible in which languages; even better would be to find properties of utility functions which correspond precisely to natural restrictions on goal bases.
- **Succinctness:** Not all goal base languages provide equally concise representations of utility functions which they jointly express. Here again, if it is known beforehand that agents will have utility functions of a certain sort, then we may benefit by selecting a language where those utility functions have efficient representations.
- **Complexity:** Problems such as finding an optimal state are easier in some goal base languages than in others. The problem of finding an optimal state for one goal base (which we call MAX-UTILITY) differs little from that of finding an optimal state across multiple goal bases, and is therefore closely related to the problem of winner determination for processes like combinatorial auctions. If one language has easier decision or search problems than another, this could be a factor in selecting it for some application.

Ideally, we would find a fully expressive, maximally succinct, computationally trivial goal base language and use that in all applications. As there is (demonstrably) no such beast, we must strike a balance between expressivity and succinctness on one hand, and complexity on the other in any potential application. In order to make such comparisons, we must examine the properties of particular goal base languages. Without such results, application designers cannot make informed decisions about which languages are best for their applications. It is for this purpose that we present numerous results for a range of these languages, in hopes of clarifying which languages are interesting.

In this paper, we concentrate on two of these properties: comparative succinctness and the computational complexity of finding an optimal alternative. The remainder of this paper is structured as follows: In the next section we list the relevant basic definitions surrounding the specification of utility functions over combinatorial domains by means of weighted propositional formulas. The body of this paper is devoted to the analysis of various restrictions on weighted goals and the languages which arise from them. We first consider the so-called uniqueness property, which assesses whether a language is tight in the sense of having only a single way of specifying any given utility function. In the following section, we examine the expressivity of one natural language, conjunctions of atoms with positive weights. We next consider the relative succinctness of several languages, and finally examine the complexity of finding models which maximize utility within particular languages.

Preliminaries

Here we introduce utility functions, goal bases, and weighted formulas, along with the definitions of various properties of utility functions.

Definition 1 (Utility Functions and Models)**.** *A utility function is a mapping $u : 2^{\mathcal{PS}} \to \mathbb{R}$, where $\mathcal{PS}$ is a fixed, finite set of propositional variables. A* model *is a set $M \subseteq \mathcal{PS}$.*

Definition 2 (Weighted Formulas and Goal Bases)**.** *A* weighted formula *is a pair (φ, w) where φ is a propositional formula in the language $\mathcal{L}_{\mathcal{PS}}$ and $w \in \mathbb{R}$. A* goal base *is a set $G = \{(\varphi_i, w_i)\}_i$ of weighted satisfiable formulas. The utility function u_G generated by the goal base G is*

$$u_G(M) = \sum \{w_i : (\varphi_i, w_i) \in G \text{ and } M \models \varphi_i\}$$

for each $M \in 2^{\mathcal{PS}}$. The set of formulas used in a goal base G is $\mathrm{For}(G) = \{\varphi : (\varphi, a) \in G\}$.

For present purposes, we restrict formulas to contain only the connectives $\neg$, $\wedge$, and $\vee$.

Definition 3 (Goal Base Summation)**.** *If G, G' are goal bases, then*

$$G \oplus G' = \{(\varphi, \sum_{(\varphi,a)\in G} a + \sum_{(\varphi,b)\in G'} b) : \varphi \in \mathrm{For}(G \cup G')\}$$

NB: $\oplus$ *does not* combine formulas which are semantically equivalent but syntactically distinct. E.g., $\{(p, 1)\} \oplus \{(p \wedge p, 1)\} \neq \{(p, 2)\}$.

Definition 4 (Goal Base Size)**.** *Let G be a goal base. Then the* size *of G (written* $\mathrm{size}(G)$*) is defined as the number of occurrences of propositional variables in the formulas of G.*

Definition 5 (Restrictions)**.** *Call $H \subseteq \mathcal{L}_{\mathcal{PS}}$ a restriction on the language of a goal base, and $H' \subseteq \mathbb{R}$ a restriction of the weights of a goal base. Then define $\mathcal{U}(H, H')$ as the class of utility functions which may be generated by goal bases meeting those restrictions.*

Specifically, we consider these restrictions on formulas:

- An *atom* is a member of $\mathcal{PS}$.
- A *literal* is an atom or its negation.
- A *clause* is a (possibly empty) disjunction of literals.
- A *cube* is a (possibly empty) conjunction of literals.
- A *positive* X is a formula of type X free of negations.
- A *Horn formula* is a clause with ≤ 1 positive disjunct.
- A k-formula is a formula with $\leq k$ binary connectives.

When applied to weights, the restrictions *all* and *positive* refer to weights in $\mathbb{R}$ and $\mathbb{R}^+$, respectively. E.g., $\mathcal{U}(\textit{positive 42-clauses}, \textit{positive})$ is the class of utility functions representable by goal bases containing only positively-weighted clauses of at most 42 atoms. Further, we consider these properties of utility functions:

- u is *normalized* iff $u(\emptyset) = 0$.
- u is *nonnegative* iff $u(X) \geq 0$ for all X.
- u is *monotonic* iff $u(X) \geq u(Y)$ for all $X \supseteq Y$.
- u is *modular* iff $u(X \cup Y) = u(X) + u(Y) - u(X \cap Y)$ for all X, Y.
- u is *supermodular* iff $u(X \cup Y) \geq u(X) + u(Y) - u(X \cap Y)$ for all X, Y.

Uniqueness

In this section we consider, in gross terms, how many ways there are to represent a given utility function in some selected classes of goal bases. In particular, we are going to define what it means for a language to have a *unique* representation for any utility function it can express. This is an interesting property, because it suggests that the language in question is parsimonious in its expressivity. The uniqueness property is also of great interest from a technical perspective, as it can be useful for establishing (negative) results on the relative succinctness of different languages (as will be discussed in the section on succinctness).

Definition 6. *A utility function u is* represented *in a language $\mathcal{L}$ if there exists a goal base $G \in \mathcal{L}$ such that $u = u_G$. A utility function u is* uniquely represented *in a language $\mathcal{L}$ if for every maximal pairwise nonequivalent set of formulas Φ meeting the restrictions of $\mathcal{L}$, there is a unique goal base G such that $\mathrm{For}(G) \subseteq \Phi$ and $u_G = u$. (Note that some weights in G may be zero.) A language $\mathcal{L}$ is said to have* unique representations *if every u represented in $\mathcal{L}$ is uniquely represented.*

The problem of determining whether a utility function u has a unique representation in a given language $\mathcal{L}$ amounts to examining the system of linear equations which describes u in $\mathcal{L}$. A language $\mathcal{L}$ has $|\mathcal{L}/_{\equiv}| = m$ distinct nonequivalent formulas, and over a set of atoms $\mathcal{PS}$ there are $2^{|\mathcal{PS}|} = n$ states. Each state $i \in 2^{\mathcal{PS}}$ defines a constraint

$$a_{i1}w_1 + \ldots + a_{im}w_m = b_i$$

where $a_{ij} \in \{0, 1\}$, depending on formula clause j is true in state i; and $b_i = u(X_i)$, where X_i is the set of true atoms in state i. Taken together as matrices $\mathsf{A}\mathsf{x} = \mathsf{b}$, we have:

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

That is, the w_i are the weights, the b_i are the values of the utility function, and the a_{ij} mark which formulas are true in which states. If $n = m$, i.e., if the number of formulas in the language over $\mathcal{PS}$ equals the number of states over $\mathcal{PS}$, then A will be square. This makes available to us a well-known fact from linear algebra, *viz.* that the determinant of the square matrix A is nonzero only when the system has a single, unique solution (Anton 1994, Theorem 2.3.6).

Proposition 1. *$\mathcal{U}(\textit{positive clauses}, \textit{all})$ has unique representations.*

Proof. Suppose that G contains only positive clauses and generates u_n, where $n = |\mathcal{PS}|$. We can write one constraint for each state except $\emptyset$, so we have $2^n - 1$ constraints. (The constraint for $\emptyset$ can be omitted, since all positive clauses are false in that case.) We have the clause $\bigvee X$ for each nonempty $X \subseteq \mathcal{PS}$, so also $2^n - 1$ distinct nonequivalent clauses, and hence $2^n - 1$ variables for weights.

Enumerate the positive clauses such that the index j codes for the positive clause $\varphi_j = \bigvee\{p_k : j \,\&\, 2^k \neq 0\}$, where '&' stands for *bitwise* conjunction. E.g., $\varphi_7 = p_0 \vee p_1 \vee p_2$, because $7 = 2^0 + 2^1 + 2^2$. Then, let $a_{ij} = 1$ if $i \,\&\, j \neq 0$, and $a_{ij} = 0$ otherwise. This sets $a_{ij} = 1$ iff clause j is true in state i. In other words, each row of A_n is a state, and the ones in a row mark the positive clauses which are true in that state.

Now observe that $\mathsf{A}_1 = [1]$ and A_{n+1} is a block matrix

$$\mathsf{A}_{n+1} = \begin{bmatrix} & & & 0 & & & \\ & \mathsf{A}_n & & \vdots & & \mathsf{A}_n & \\ & & & 0 & & & \\ 0 & \ldots & 0 & 1 & 1 & \ldots & 1 \\ & & & 1 & & & \\ & \mathsf{A}_n & & \vdots & & 1 & \\ & & & 1 & & & \end{bmatrix}$$

where 1 is a matrix of the appropriate size, with every element a 1. The additional rows in A_{n+1} (over A_n) are for states in which p_n, the new variable, is true. The middle row is the state where only p_n is true, and from there down p_n is true in every state. With respect to the other variables, the states in the bottom half repeat the states in the top half. The additional columns in A_{n+1} are for positive clauses which contain p_n. The middle column is for p_n, the degenerate positive clause formed by that variable alone, and the columns thereafter repeat the first $2^{n-1} - 1$ columns with p_n as an additional disjunct. Therefore, the upper left and upper right blocks repeat A_n since no state there makes p_n true; the lower left block repeats A_n since no clause there

contains p_n; and the lower right block is all ones because every state and clause there contains p_n.

Clearly, $\det(\mathsf{A}_1) = 1$. Suppose that $\det(\mathsf{A}_n) \neq 0$ To show that $\det(\mathsf{A}_{n+1}) \neq 0$, we will twice use the following fact about determinants of block matrices:

Fact 1. *For the block matrix* $\left[\begin{smallmatrix}\mathsf{A} & \mathsf{B}\\ \mathsf{C} & \mathsf{D}\end{smallmatrix}\right]$,

$$\det \begin{bmatrix}\mathsf{A} & \mathsf{B}\\ \mathsf{C} & \mathsf{D}\end{bmatrix} = \det(\mathsf{A}) \det(\mathsf{D} - \mathsf{C}\mathsf{A}^{-1}\mathsf{B})$$

where A *is* $m \times m$*,* B *is* $m \times n$*,* C *is* $n \times m$ *and* D *is* $n \times n$*.*

Slice A_{n+1} into blocks like so

$$\mathsf{A} = \left[\begin{array}{ccc|ccccc} & & & 0 & & & \\ & \mathsf{A}_n & & \vdots & & \mathsf{A}_n & \\ & & & 0 & & & \\ \hline 0 & \dots & 0 & 1 & 1 & \dots & 1 \\ & & & 1 & & & \\ & \mathsf{A}_n & & \vdots & & 1 & \\ & & & 1 & & & \end{array}\right] = \mathsf{B}$$

$$\mathsf{C} = \mathsf{D}$$

and note that $\mathsf{A} = \mathsf{A}_n = \mathsf{A}_n^{-1}$ (because the A_i are symmetric about their main diagonal). Further,

$$\mathsf{C}\mathsf{A}^{-1} = \begin{bmatrix}0 & \dots & 0\\ & \mathsf{A}_n & \end{bmatrix} \mathsf{A}^{-1} = \begin{bmatrix}0 & \dots & 0\\ & \mathsf{I} & \end{bmatrix}$$

$$\mathsf{C}\mathsf{A}^{-1}\mathsf{B} = \begin{bmatrix}0 & \dots & 0\\ & \mathsf{I} & \end{bmatrix} \begin{bmatrix}0 & \\ \vdots & \mathsf{A}_n\\ 0 & \end{bmatrix} = \begin{bmatrix}0 & \dots & 0\\ \vdots & \mathsf{A}_n & \\ 0 & & \end{bmatrix}$$

$$\mathsf{D} - \mathsf{C}\mathsf{A}^{-1}\mathsf{B} = 1 - \begin{bmatrix}0 & \dots & 0\\ \vdots & \mathsf{A}_n & \\ 0 & & \end{bmatrix} = \begin{bmatrix}1 & \dots & 1\\ \vdots & 1 - \mathsf{A}_n & \\ 1 & & \end{bmatrix}$$

where I and 1 are an identity matrix and a matrix of ones, respectively, of the appropriate sizes.

Next, slice $\mathsf{D} - \mathsf{C}\mathsf{A}^{-1}\mathsf{B}$ into blocks like so

$$\mathsf{A}' = \left[\begin{array}{c|ccc} 1 & 1 & \dots & 1 \\ \hline 1 & & & \\ \vdots & & 1 - \mathsf{A}_n & \\ 1 & & & \end{array}\right] = \mathsf{B}'$$

$$\mathsf{C}' = \mathsf{D}'$$

and by the Fact, we have that

$$\begin{aligned}\det(\mathsf{D} - \mathsf{C}\mathsf{A}^{-1}\mathsf{B}) &= \det \begin{bmatrix}1 & \dots & 1\\ \vdots & 1 - \mathsf{A}_n & \\ 1 & & \end{bmatrix}\\ &= \det(\mathsf{A}') \det(\mathsf{D}' - \mathsf{C}'\mathsf{A}'^{-1}\mathsf{B}')\\ &= \det[1] \det(1 - \mathsf{A}_n - \begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix} [1] [1 \dots 1])\\ &= \det(1 - \mathsf{A}_n - 1)\\ &= \det(-\mathsf{A}_n)\\ &= -\det(\mathsf{A}_n)\end{aligned}$$

Applying the Fact a second time, we have that $\det(\mathsf{A}_{n+1}) = \det(\mathsf{A}_n) \det(\mathsf{D} - \mathsf{C}\mathsf{A}^{-1}\mathsf{B}) = -\det(\mathsf{A}_n)^2 = -1 \neq 0$, which completes the induction.

Having shown that $\det(\mathsf{A}_n)$ is nonzero, it follows that the system has exactly one solution. Therefore G is the unique positive-clause generator of u_n. □

Naturally, many languages lack the uniqueness property. The next result states two examples.

Proposition 2. *$\mathcal{U}$(clauses, all) and $\mathcal{U}$(Horn, all) lack unique representations.*

Proof. The utility function $u(X) = 1$ if $p \notin X$ and $u(X) = 0$ otherwise, can be represented in Horn clauses as $\{(\neg p, 1)\}$ or $\{(\top, 1), (p, -1)\}$. □

Finally, we mention here that $\mathcal{U}$(*positive cubes*, *all*) also has unique representations. This may be shown using the same method as in the proof of Proposition 1. Alternatively, we can give an inductive argument that the weight of each cube is uniquely determined, starting with the weight for $\top$, then those for atoms, and so on (Chevaleyre *et al.* 2004). The same also follows from the close relationship between the language based on positive cubes and the *Möbius inversion*, which is itself uniquely determined (Rota 1964; Grabisch 1997).

Expressivity and Correspondence

A central property to be analyzed for any preference representation language is its *expressivity*. The expressivity of weighted propositional formulas for representing utility functions has been studied at length by Chevaleyre *et al.* (2006): Without restrictions on either goals or weights, *any* utility function can be specified. Full expressive power can also be retained if we restrict goals to either cubes or clauses, or even just positive cubes. A restriction to positive clauses, on the other hand, forces utility functions to be normalized. Literals can generate any modular function, and only those, while k-formulas correspond to so-called k-additive utility functions (Grabisch 1997). Finally, $\mathcal{U}$(*positive*, *positive*) is exactly the class of nonnegative monotonic utility functions. Here we prove one further result along these lines:

Proposition 3. *$\mathcal{U}$(positive cubes, positive) is equal to the class of nonnegative monotonic utility functions u such that for all X, $u(X) \geq \sum_{Y \subset X} u(Y) \cdot (-1)^{|X \setminus Y|+1}$.*

Proof. First, notice that $w_{\bigwedge Y} = u(Y) - \sum_{Z \subset Y} w_{\bigwedge Z}$, and so $\sum_{Y \subset X} w_{\bigwedge Y} = \sum_{Y \subset X} (u(Y) - \sum_{Z \subset Y} w_{\bigwedge Z})$. Expanding weights all the way down gives us

$$\sum_{Y \subset X} w_{\bigwedge Y} = \sum_{Y \subset X} u(Y) \cdot (-1)^{|X \setminus Y|+1}$$

which we will use in both directions.

($\Longrightarrow$) Suppose that $u \in \mathcal{U}$(*positive cubes*, *positive*).

- u is nonnegative since all weights are positive.

- u is monotonic: Fix $A \subseteq \mathcal{PS}$, $x \in \mathcal{PS}$.

$$u(A \cup \{x\}) = \sum_{X \subseteq A \cup \{x\}} w_{\bigwedge X} \geq \sum_{X \subseteq A} w_{\bigwedge X} = u(A)$$

since $2^A \subseteq 2^{A \cup \{x\}}$ and all $w_\varphi \geq 0$.
- $u(X) \geq \sum_{Y \subset X} u(Y) \cdot (-1)^{|X \setminus Y|+1}$: $u(X) = w_{\bigwedge X} + \sum_{Y \subset X} w_{\bigwedge Y} = w_{\bigwedge X} + \sum_{Y \subset X} u(Y) \cdot (-1)^{|X \setminus Y|+1}$, and $w_{\bigwedge X} \geq 0$.

($\Longleftarrow$) Suppose that u satisfies all three properties. Let G be the unique representation of u in positive cubes, and $w_{\bigwedge X}$ an arbitrary weight in G. $w_{\bigwedge X} + \sum_{Y \subset X} w_{\bigwedge Y} = u(X)$ by definition, and $u(X) \geq \sum_{Y \subset X} w_{\bigwedge Y}$ by the third property. So $w_{\bigwedge X} + \sum_{Y \subset X} w_{\bigwedge Y} \geq \sum_{Y \subset X} w_{\bigwedge Y}$, i.e., $w_{\bigwedge X} \geq 0$. Therefore, all weights in G are positive (since we do not count zero weights as being part of G) and so $u \in \mathcal{U}(\textit{positive cubes}, \textit{positive})$. □

Observe how the second part of the proof relies on the uniqueness property of the language of positive cubes (see previous section), and also the fact that, without a restriction to positive weights, the language of positive cubes in known to be fully expressive (Chevaleyre, Endriss, & Lang 2006).

To date, the important property of (normalized monotonic) supermodularity has escaped a precise characterization in terms of restrictions on weighted goals. Proposition 3 indicates that a corresponding language lies between $\mathcal{U}(\textit{positive cubes}, \textit{positive})$ and $\mathcal{U}(\textit{positive cubes}, \textit{all})$. Moreover, it is possible that $\mathcal{U}(\textit{positive cubes}, \textit{positive})$ could serve as an approximation of the class of supermodular utility functions.

Succinctness

In this section, we consider the relative succinctness of languages. If two languages are equally expressive, then the one with shorter representations is preferable, other things being equal. Intuitively, one language is more succinct than another if for every representation in the second, there is an equivalent but shorter representation in the first. Succinctness results are much easier to come by when the less succinct language has unique representations, since in those cases we can be assured that there is no as-yet-undiscovered representation more efficient than the one we have. As a result, all of the succinctness results presented here rely on uniqueness results presented above.

The succinctness definition given in Chevaleyre *et al.* (2006) does not permit the comparison of languages which differ in expressive power; however, this restriction is unnecessary and it might occasionally be useful to compare the succinctness of such languages within their areas of expressive overlap. The following definitions provide the concepts and notation for doing so.

Definition 7. *Let $\mathcal{L}$ be a language for (i.e., a class of) goal bases. We define:*

- $\mathcal{U}(\mathcal{L}) = \{u_G : G \in \mathcal{L}\}$
- $\mathrm{Rep}_{\mathcal{L}}(\mathcal{U}) = \{G \in \mathcal{L} : u_G \in \mathcal{U}\}$
- $\mathcal{L}_{\cap \mathcal{L}'} = \mathrm{Rep}_{\mathcal{L}}(\mathcal{U}(\mathcal{L}) \cap \mathcal{U}(\mathcal{L}'))$

$\mathcal{U}(\mathcal{L})$ is the set of utility functions representable in the language $\mathcal{L}$. $\mathrm{Rep}_{\mathcal{L}}(\mathcal{U})$ is the set of *$\mathcal{L}$-representations* of the utility functions in the set $\mathcal{U}$. $\mathcal{L}_{\cap \mathcal{L}'}$ is the *expressive intersection* of the languages $\mathcal{L}$ and $\mathcal{L}'$ as represented in $\mathcal{L}$. It is easy to see that given any two goal-base languages $\mathcal{L}$ and $\mathcal{L}'$, $\mathcal{L}_{\cap \mathcal{L}'}$ and $\mathcal{L}'_{\cap \mathcal{L}}$ are equally expressive.

Definition 8. *$\mathcal{L} \preceq \mathcal{L}'$ ($\mathcal{L}'$ is at least as succinct as $\mathcal{L}$) iff there exist a function $f : \mathcal{L}_{\cap \mathcal{L}'} \to \mathcal{L}'_{\cap \mathcal{L}}$ and a polynomial p such that $G \equiv f(G)$ and $\mathrm{size}(f(G)) \leq p(\mathrm{size}(G))$ for all $G \in \mathcal{L}_{\cap \mathcal{L}'}$.*

$\mathcal{L} \sim \mathcal{L}'$, $\mathcal{L} \prec \mathcal{L}'$, and $\mathcal{L} \perp \mathcal{L}'$ (to be read as *equally succinct*, *less succinct*, and *incomparable*) are defined from $\preceq$ in the standard way. This definition permits us to compare the succinctness of languages which differ in expressivity. (Notice that languages with no overlap in their expressivity would be considered equally succinct according to this definition.)

Cubes and clauses are natural languages for representing common utility functions, particularly those which confer bonuses or penalties for combinations of items (cubes: Boardwalk *and* Park Place, or assault *and* battery), or those which reward good-enough options but give an additional bonus for specificity (clauses: turning a Phillips screw with a slotted or Phillips screwdriver works, but turning a Phillips screw with a Phillips screwdriver is better). Here, we compare the relative succinctness of cubes and clauses and their positive restrictions.

Proposition 4. $\mathcal{U}(\textit{positive cubes}, \textit{all}) \prec \mathcal{U}(\textit{cubes}, \textit{all})$.

Proof. (Chevaleyre, Endriss, & Lang 2006, Prop. 13). □

Proposition 5. $\mathcal{U}(\textit{positive clauses}, \textit{all}) \prec \mathcal{U}(\textit{clauses}, \textit{all})$.

Proof. Clearly $\mathcal{U}(\textit{positive clauses}, \textit{all}) \preceq \mathcal{U}(\textit{clauses}, \textit{all})$, since every positive clause is a clause. Consider the family of utility functions $u_n : 2^{\{p_1, \ldots, p_n\}} \to \mathbb{R}$ where

$$u_n(X) = \begin{cases} 1 & \text{if } X = \mathcal{PS} \\ 0 & \text{otherwise} \end{cases}$$

u_n may be represented in clauses as

$$\{(\top, 1), (\bigvee \{\neg p : p \in \mathcal{PS}\}, -1)\}$$

the length of which increases linearly with n. u_n may be represented in positive clauses as

$$\{(\bigvee X, w_{\bigvee X}) : \emptyset \subset X \subseteq \mathcal{PS}\}$$

where

$$w_{\bigvee X} = \begin{cases} 1 & \text{if } |X| \text{ is odd} \\ -1 & \text{if } |X| \text{ is even} \end{cases}$$

Proof that the $w_{\bigvee X}$ define the correct utility function: First, let $\binom{n}{\text{even}}$ and $\binom{n}{\text{odd}}$ be the number of even and odd subsets of n, respectively. $\binom{n}{\text{even}} = \binom{n}{\text{odd}}$: For odd n, every even-sized subset X has a unique odd-sized complement $\bar{X}$, and vice versa. For even n, notice that $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$, so $\binom{n}{\text{odd}} = \sum_{\substack{0 \leq i \leq n \\ i \text{ odd}}} \binom{n}{i} = \binom{n}{1} + \binom{n}{3} + \cdots + \binom{n}{n-1} = \binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \binom{n-1}{3} + \cdots + \binom{n-1}{n-2} + \binom{n-1}{n-1} =$

$\sum_{0\leq i\leq n-1}\binom{n-1}{i} = 2^{n-1}$, which is exactly half of the subsets. Next, suppose for purposes of this proof that $w_{\bigvee\emptyset} = 0$, even though $\top$ is not a positive clause. Now,

$u(\mathcal{PS}) = 1$**:** $\binom{\mathcal{PS}}{\text{even}} = \binom{\mathcal{PS}}{\text{odd}}$, i.e., there are equal numbers of even and odd sized subsets of $\mathcal{PS}$, with a corresponding nonzero weight for each except $\emptyset$. Thus $\binom{\mathcal{PS}}{\text{odd}} - (\binom{\mathcal{PS}}{\text{even}} - 1) = 1$ is the difference between the number of nonzero odd-sized and even-sized weights. Since all odd-sized weights are 1 and all nonzero even-sized weights are -1, 1 is also the *sum* of all even- and odd-sized weights.

$u(X) = 0$**,** $X \subset \mathcal{PS}$**:** The active weights are those $w_{\bigvee Y}$ for which $X \cap Y \neq \emptyset$. There are $2^{\mathcal{PS}\setminus X}$ *in*active weights (now including $w_{\bigvee\emptyset}$), half of which are for odd-sized disjunctions ($\binom{\mathcal{PS}\setminus X}{\text{even}} = \binom{\mathcal{PS}\setminus X}{\text{odd}}$). Since the total number of even- and odd-sized weights are equal, we have also that $\binom{\mathcal{PS}}{\text{even}} - \binom{\mathcal{PS}\setminus X}{\text{even}} = \binom{\mathcal{PS}}{\text{odd}} - \binom{\mathcal{PS}\setminus X}{\text{odd}}$, which are the total number of active even- and odd-sized weights, respectively. Since $\emptyset$ is inactive, the sums of active even- and odd-sized weights are opposite, and so together sum to 0.

Thus all $2^n - 1$ positive clauses receive nonzero weights. By Proposition 1, this representation is unique; its length increases exponentially with n. Hence over normalized utility functions (the expressive overlap of $\mathcal{U}(\textit{positive clauses}, \textit{all})$ and $\mathcal{U}(\textit{clauses}, \textit{all})$), clauses are strictly more succinct than positive clauses. □

Here we show that positive cubes and positive clauses (with arbitrary weights) are incomparable with respect to succinctness:

Proposition 6.

$$\mathcal{U}(\textit{positive cubes}, \textit{all}) \perp \mathcal{U}(\textit{positive clauses}, \textit{all}).$$

Proof. $\mathcal{U}(\textit{positive clauses}, \textit{all}) \not\preceq \mathcal{U}(\textit{positive cubes}, \textit{all})$: The family of utility functions $u_n(X) = 1$ if $X \neq \emptyset$ and $u_n(X) = 0$ otherwise, is represented uniquely and linearly as $\{(\bigvee\mathcal{PS}, 1)\}$ in positive clauses. $w_{\bigwedge X} = (-1)^{|X|}$ in positive cubes, so the unique representation there assigns nonzero weights to all $2^n - 1$ positive cubes.

$\mathcal{U}(\textit{positive clauses}, \textit{all}) \not\succeq \mathcal{U}(\textit{positive cubes}, \textit{all})$: Let $u_n(X)$ be the same family of utility functions as in Proposition 5. $u_n(X)$ is represented uniquely and linearly as $\{(\bigwedge\mathcal{PS}, 1)\}$ in positive cubes, but the representation in positive clauses is exponential, as shown in Proposition 5. □

The fact that $\mathcal{U}(\textit{positive cubes}, \textit{all})$ and $\mathcal{U}(\textit{positive clauses}, \textit{all})$ are incomparable in terms of succinctness indicates that these two classes are best suited to representing very different kinds of utility functions. (As mentioned above, positive cubes seem good for awarding bonuses for completing sets, while positive clauses seem good for awarding bonuses for acquiring the first item in a set.) Given their incomparable sublanguages, the following result is surprising:

Proposition 7. $\mathcal{U}(\textit{cubes}, \textit{all}) \sim \mathcal{U}(\textit{clauses}, \textit{all})$.

Proof. Follows immediately from Lemma 1. □

Lemma 1. *If* $\mathcal{L}_{\textit{cubes}} \subseteq \Phi$ *or* $\mathcal{L}_{\textit{clauses}} \subseteq \Phi$*,* $\mathcal{L}_{\textit{cubes}} \subseteq \Psi$ *or* $\mathcal{L}_{\textit{clauses}} \subseteq \Psi$*, and* $\Phi, \Psi \subseteq \mathcal{L}_{\textit{cubes}} \cup \mathcal{L}_{\textit{clauses}}$*, then* $\mathcal{U}(\Phi, \textit{all}) \sim \mathcal{U}(\Psi, \textit{all})$.

Proof. Suppose that $G \in \mathcal{U}(\Phi, \textit{all})$. Enumerate $(\varphi_i, w_i) \in G$. We construct an equivalent goal base G'. Let $G_0 = G$,

$$G_{i+1} = \begin{cases} (G_i \setminus \{(\varphi_i, w_i)\}) \\ \qquad \cup \{(\neg\varphi_i, -w_i), (\top, w_i)\} & \text{if } \varphi_i \notin \Psi \\ G_i & \text{otherwise} \end{cases}$$

and let $G' = G_{|G|}$.

The transformation produces an equivalent goal base: By (Chevaleyre, Endriss, & Lang 2006, Lemma 1), $G_i \equiv G_{i+1}$ for all i, so $G = G_1 \equiv G_2 \equiv \ldots \equiv G_{|G|-1} \equiv G_{|G|} = G'$.

The transformation produces a goal base in the appropriate language: Suppose that $\varphi \in \Phi$. At minimum, Ψ contains either every clause or every cube. If φ is a clause, then $\neg\varphi$ is a cube, and vice versa. Hence at least one of φ and $\neg\varphi$ are in Ψ. $\top$ is both a cube ($\bigwedge\emptyset$) and a clause ($p \vee \neg p$), so $\top \in \Psi$ regardless. Thus $G' \in \mathcal{U}(\Psi, \textit{all})$.

The transformation produces a goal base as succinct as the original: If φ is a cube, then φ requires the same number of atoms and binary connectives as as $\neg\varphi$ (written as a clause); similarly, if φ is a clause. The only increase in size between G and G' can come from the addition of $\top$, so we have that $|G'| \leq |G| + 1$.

Therefore, $\mathcal{U}(\Phi, \textit{all}) \succeq \mathcal{U}(\Psi, \textit{all})$. By the same argument $\mathcal{U}(\Phi, \textit{all}) \preceq \mathcal{U}(\Psi, \textit{all})$. So $\mathcal{U}(\Phi, \textit{all}) \sim \mathcal{U}(\Psi, \textit{all})$. □

From this, it follows that there can be no gain in succinctness by adding clauses to a cubes language with arbitrary weights, or vice versa.

Complexity

In this section, we analyze the effect that restrictions on goal bases have on the complexity of answering questions about the utility functions they represent, focusing specifically on the problem MAX-UTILITY—finding a model which produces maximal utility, expressed as a decision problem.

Definition 9. *The decision problem* MAX-UTILITY(H, H') *is defined as: Given a goal base* $G \in \mathcal{U}(H, H')$ *and an integer* K*, check whether there is a model* $M \in 2^{\mathcal{PS}}$ *where* $u_G(M) \geq K$.

MAX-UTILITY is clearly in NP for the unrestricted language, since whether $u_G(M) \geq K$ is polynomially checkable, and is NP-complete via a reduction from MAXSAT (Garey & Johnson 1979). MAX-UTILITY for $\mathcal{U}(\textit{literals}, \textit{all})$ and $\mathcal{U}(\textit{positive}, \textit{positive})$ is polynomial. (Respectively: Make p true iff $w_p > w_{\neg p}$ for each atom p; and make all atoms true.) Generally, however, MAX-UTILITY is NP-complete for languages which permit both positive and negative weights or both positive and negative literals.

Proposition 8. MAX-UTILITY$(k\textit{-cubes}, \textit{pos})$ *is NP-complete for* $k \geq 2$.

Proof. The decision problem MAX k-CONSTRAINT SAT is defined as: Given a set C of k-cubes in $\mathcal{PS}$ and an integer K, check whether there is a model $M \in 2^{\mathcal{PS}}$ which satisfies at

least K of the k-cubes in C. MAX-UTILITY(k-*cubes*, *pos*) is a weighted version of MAX k-CONSTRAINT SAT, which is NP-complete for $k \geq 2$ (Ausiello *et al.* 1999, LO12, Appendix B). □

Proposition 9. MAX-UTILITY(k-*Horn*, *pos*) *is NP-complete for* $k \geq 2$.

Proof. This is a weighted version of MAX HORN 2-SAT, which is NP-complete (Jaumard & Simeone 1987, Proposition 3.1). □

Proposition 10. MAX-UTILITY(*positive* k-*cubes*, *all*) *is NP-complete for* $k \geq 2$.

Proof. MAX-UTILITY(k-*cubes*, *pos*) is NP-complete for $k \geq 2$; MAX-UTILITY(k-*cubes*, *all*) contains it for any fixed k and so is NP-complete also. We exhibit a polynomial reduction of MAX-UTILITY(k-*cubes*, *all*) to MAX-UTILITY(*positive* k-*cubes*, *all*). Given a goal base $G \in \mathcal{U}(k\text{-cubes}, all)$, construct G' as follows:

1. For each $(\bigwedge X, w) \in G$:
   (a) Let $X' = \{x : x \in X\} \cup \{\bar{x} : \neg x \in X\}$.
   (b) Put $(\bigwedge X', w) \in G'_0$.
2. Normalize G'_0 to $[-1, 1]$.
3. Let $\delta = \sum\{|w| : (\varphi, w) \in G'_0\}$.
4. Let $\alpha = \delta + 1$, and $\beta = -3\delta - 3$.
5. For each $x \in \mathcal{PS}$, put
$$(x, \alpha), (\bar{x}, \alpha), (x \wedge \bar{x}, \beta) \in G'_1$$
6. Let $G' = G'_0 \oplus G'_1$.

Here, $\overline{\mathcal{PS}}$ is $\mathcal{PS}$ with a bar over each propositional variable.

Lemma 2. *Fix* $A \subseteq \mathcal{PS} \cup \overline{\mathcal{PS}}$ *such that* $x, \bar{x} \notin A$. *Then* $u_{G'}(A \cup \{x, \bar{x}\}) < u_{G'}(A) < u_{G'}(A \cup \{x\}), u_{G'}(A \cup \{\bar{x}\})$.

Proof. Note that for any two models M, N we have that $|u_{G'_0}(M) - u_{G'_0}(N)| \leq \delta$. δ is a (not necessarily tight) upper bound on the utility change in G'_0 between arbitrary models. This fact is used below to bound away the terms $u_{G'_0}(A \cup \{x\})$ and $u_{G'_0}(A \cup \{x, \bar{x}\})$:

$$\begin{aligned} u_{G'}(A \cup \{x\}) &= u_{G'_0}(A \cup \{x\}) + u_{G'_1}(A \cup \{x\}) \\ &= u_{G'_0}(A \cup \{x\}) + u_{G'_1}(A) + w_x^{G'_1} \\ &= u_{G'_0}(A \cup \{x\}) + u_{G'_1}(A) + \delta + 1 \\ &\geq u_{G'_0}(A) - \delta + u_{G'_1}(A) + \delta + 1 \\ &> u_{G'_0}(A) + u_{G'_1}(A) = u_{G'}(A) \end{aligned}$$

Similarly, $u_{G'}(A \cup \{\bar{x}\}) > u_{G'}(A)$. Finally,

$$\begin{aligned} u_{G'}(A \cup \{x, \bar{x}\}) &= u_{G'_0}(A \cup \{x, \bar{x}\}) + u_{G'_1}(A \cup \{x, \bar{x}\}) \\ &= u_{G'_0}(A \cup \{x, \bar{x}\}) \\ &\quad + u_{G'_1}(A) + w_x^{G'_1} + w_{\bar{x}}^{G'_1} + w_{x \wedge \bar{x}}^{G'_1} \\ &= u_{G'_0}(A \cup \{x, \bar{x}\}) + u_{G'_1}(A) - \delta - 1 \\ &\leq u_{G'_0}(A) + \delta + u_{G'_1}(A) - \delta - 1 \\ &< u_{G'_0}(A) + u_{G'_1}(A) = u_{G'}(A) \end{aligned}$$

□

If M' is a model in $\mathcal{PS} \cup \overline{\mathcal{PS}}$, let $M = M' \setminus \overline{\mathcal{PS}}$. By the Lemma, we have that every model optimal for $u_{G'}$ will contain exactly one of x and $\bar{x}$ for all $x \in \mathcal{PS}$. (If M' contains both x and $\bar{x}$, it could gain at least 1 utility by removing both; if M' has neither, it could gain at least 1 utility by adding one.) Call a model M' in $\mathcal{PS} \cup \overline{\mathcal{PS}}$ *full* if for every $x \in \mathcal{PS}$ either $x \in M'$ or $\bar{x} \in M'$, and *bivalent* if for every $x \in \mathcal{PS}$ either $x \notin M'$ or $\bar{x} \notin M'$. Whenever M' is full and bivalent, M will be a model in $\mathcal{PS}$.

All of the operations applied in generating G' from G are order-preserving over full, bivalent models: Consider G'_0 prior to normalization. $u_{G'_0}(X') = u_G(X)$ for all models X. Normalization is order-preserving. Every full, bivalent model is optimal for $u_{G'_1}$, since all full, bivalent models have the same value ($w_x = w_{\bar{y}}$ and $w_{x \wedge \bar{x}} = w_{y \wedge \bar{y}}$ for all $x, y \in \mathcal{PS}$) and by the Lemma all nonfull or nonbivalent models are strictly dominated. Adding G'_1 to G'_0 increases every atomic weight by α, which is order-preserving; and increases $w_{x \wedge \bar{x}}$ by β, which has no effect at all since $x \wedge \bar{x}$ is false on every bivalent model. Therefore, if $u_{G'}(X') < u_{G'}(Y')$ where X' and Y' are full and bivalent, then $u_G(X) < u_G(Y)$.

Suppose that M' is optimal for $u_{G'}$. It follows from the Lemma that M' is full and bivalent, and so it follows from the above that M is optimal for u_G. This completes the reduction of MAX-UTILITY(k-*cubes*, *all*) to MAX-UTILITY(*positive* k-*cubes*, *all*). Generating G' from G and recovering M from M' are linear in the size of G and $\mathcal{PS}$, respectively, so the reduction is polynomial. Hence MAX-UTILITY(*positive* k-*cubes*, *all*) is NP-complete. □

Since $\mathcal{U}$(*positive* k-*cubes*, *all*) generates the class of k-additive functions, Proposition 10 is closely related to a known NP-completeness result for winner determination in combinatorial auctions with k-additive utility functions (Chevaleyre *et al.* 2004). The advantage of our method is that it can easily be adapted to also obtain an NP-completeness result for positive clauses:

Proposition 11. MAX-UTILITY(*positive* k-*clauses*, *all*) *is NP-complete for* $k \geq 2$.

Proof. Similar to the proof for the NP-completeness of MAX-UTILITY(*positive* k-*cubes*, *all*). Given a goal base $G \in \mathcal{U}(k\text{-clauses}, all)$, construct G' as follows:

1. For each $(\bigvee X, w) \in G$:
   (a) Let $X' = \{x : x \in X\} \cup \{\bar{x} : \neg x \in X\}$.
   (b) Put $(\bigvee X', w) \in G'_0$.
2. Normalize G'_0 to $[-1, 1]$.
3. Let $\delta = \sum\{|w| : (\varphi, w) \in G'_0\}$.
4. Let $\alpha = -2\delta - 2$, and $\beta = 3\delta + 3$.
5. For each $x \in \mathcal{PS}$, put
$$(x, \alpha), (\bar{x}, \alpha), (x \vee \bar{x}, \beta) \in G'_1$$
6. Let $G' = G'_0 \oplus G'_1$.

As in the previous proof, construction of G' from G is order-preserving over full, bivalent models. ($x \vee \bar{x}$ is

true in every full, bivalent model and hence the disjunctive weights do not disturb the ordering.) Hence by the same argument, MAX-UTILITY(k-*clauses*, *all*) reduces polynomially to MAX-UTILITY(*positive* k-*clauses*, *all*), and hence MAX-UTILITY(*positive* k-*clauses*, *all*) is NP-complete. □

Conclusion

We have examined the properties of various goal base languages. In particular, we have exhibited two which have the uniqueness property and shown their expressive power, devised a framework for comparison of and compared the succinctness of representations in several languages, and shown that MAX-UTILITY, the problem of finding maximal assignments, is NP-complete for several simple languages.

In particular, the results permit us to offer the following observations for application designers:

- It may be possible to approximate supermodularity with positive cubes, or with a language very near to positive cubes. Proposition 3 indicates that the positive cubes language captures all but a restrictive class of supermodular utility functions.
- There is no loss of concision when restricting a language containing clauses and cubes with arbitrary weights to a language containing only cubes or only clauses.
- Moving from positive clauses or cubes to general clauses or cubes can result in greater concision, with no increase in the complexity of MAX-UTILITY.
- Most languages are NP-complete for MAX-UTILITY, which, broadly speaking, appears to happen whenever both positive and negative literals or both positive and negative weights are permitted in a language.

Many avenues are open for further research. Intuitive correspondence results for several common properties of utility functions are unknown (sub- and supermodularity, concavity and convexity), as is the precise expressivity of some simple languages (clauses and positive clauses, with positive weights).

There is potential for the fruitful application of goal base languages in both combinatorial auctions (as a bidding language) and in committee elections (to extend the expressivity of voting methods). In the context of combinatorial auctions, goal base languages may provide more natural ways of expressing bids, especially when bidders' utility functions can be expected to have certain properties. Some goal base languages may also prove better than others when dealing with preference elicitation, either for agents who must construct a goal base, for agents who wish not to reveal their complete valuation, or for tabulators who wish to solicit as little information from agents as possible in order to do determine winners (Sandholm & Boutilier 2006). In the context of committee elections, these languages can provide voters with ways of expressing (dis)synergies among candidates.

References

Anton, H. 1994. *Elementary Linear Algebra*. Wiley & Sons, seventh edition.

Ausiello, G.; Crescenzi, P.; Gambosi, G.; Kann, V.; Marchetti-Spaccamela, A.; and Protasi, M. 1999. *Complexity and Approximation*. Springer-Verlag.

Chevaleyre, Y.; Endriss, U.; Estivie, S.; and Maudet, N. 2004. Multiagent resource allocation with k-additive utility functions. In *Proc. DIMACS-LAMSADE Workshop on Computer Science and Decision Theory*, Annales du LAMSADE 3, 83–100.

Chevaleyre, Y.; Endriss, U.; and Lang, J. 2006. Expressive power of weighted propositional formulas for cardinal preference modelling. In *Proc. 10th Intl. Conference on Principles of Knowledge Representation and Reasoning (KR-2006)*, 145–152. AAAI Press.

Coste-Marquis, S.; Lang, J.; Liberatore, P.; and Marquis, P. 2004. Expressive power and succinctness of propositional languages for preference representation. In *Proc. 9th Intl. Conference on Principles of Knowledge Representation and Reasoning (KR-2004)*, 203–212. AAAI Press.

Cramton, P.; Shoham, Y.; and Steinberg, R., eds. 2006. *Combinatorial Auctions*. MIT Press.

Garey, M., and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Co.

Grabisch, M. 1997. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems* 92:167–189.

Jaumard, B., and Simeone, B. 1987. On the complexity of the maximum satisfiability problem for Horn formulas. *Information Processing Letters* 26(1):1–4.

Lang, J. 2004. Logical preference representation and combinatorial vote. *Annals of Mathematics and Artificial Intelligence* 42(1–3):37–71.

Lehmann, D.; Müller, R.; and Sandholm, T. 2006. The winner determination problem. In Cramton et al. (2006). 297–317.

Nisan, N. 2006. Bidding languages for combinatorial auctions. In Cramton et al. (2006). 215–232.

Pinkas, G. 1991. Propositional nonmonotonic reasoning and inconsistency in symmetric neural networks. In *Proc. 12th Intl. Conference on Artificial Intelligence (IJCAI-1991)*. Morgan-Kaufmann Publishers.

Rota, G.-C. 1964. On the foundations of combinatorial theory I: Theory of Möbius functions. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 2(4):340–368.

Sandholm, T., and Boutilier, C. 2006. Preference elicitation in combinatorial auctions. In Cramton et al. (2006). 233–263.