Expressive Power of Weighted Propositional Formulas for Cardinal Preference Modelling
Extended Abstract[*]

Yann Chevaleyre[1], Ulle Endriss[2], and Jérôme Lang[3]

[1] LAMSADE, Université Paris-Dauphine, France
Email: chevaley@lamsade.dauphine.fr

[2] ILLC, University of Amsterdam, The Netherlands
Email: ulle@illc.uva.nl

[3] IRIT, Université Paul Sabatier, France
Email: lang@irit.fr

Abstract

As proposed in various places, a set of propositional formulas, each associated with a numerical weight, can be used to model the preferences of an agent in combinatorial domains. If the range of possible choices can be represented by the set of possible assignments of propositional symbols to truth values, then the utility of an assignment is given by the sum of the weights of the formulas it satisfies. Our aim in this paper is to establish correspondences between certain types of weighted formulas and well-known classes of utility functions (such as monotonic, concave or k-additive functions). We also briefly comment on the comparative succinctness of different types of weighted formulas for representing the same class of utility functions.

1 Introduction

Many individual or multiagent decision making problems have in their input the preferences of the agent(s) over a set of possible alternatives. Such problems include decision making and planning under uncertainty, multi-criteria decision making and decision support systems, automated group decision making (including auctions, fair division, vote), and distributed decision making (including negotiation).

Saying that the input of a problem contains the preference structure of the agent(s) over the set of alternatives does not imply anything about how these structures are *specified* in the input. Clearly, if the set of alternatives is small, this question is not relevant, since the size of the explicit representation of the preference structure is small as well. This is no longer the case when the set of alternatives is a combinatorial domain: in this case, the set of alternatives is the set of all assignments of each of a given finite set of variables to a value of the corresponding finite domain. Examples are numerous: in combinatorial auctions and

[*]A full version of this paper appears in the Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-2006).

negotiation over resources [9, 12], an alternative is an assignment of each good to an agent; in multiple issue referenda [7], an alternative consists of a truth value (yes or no) for each issue.

For this purpose, many languages have been developed so as to express preference structures as succinctly as possible. These languages differ significantly, depending on whether the preference structure to be expressed is *ordinal* or *cardinal* (numerical). Languages for the succinct representation of ordinal preferences include languages of *ceteris paribus* statements, which range from very expressive languages [13] to syntactical restrictions such as CP-nets [4]. They also include languages based on conditional logics, prioritised logics, and prioritised constraint satisfaction problems (see e.g. [21] for an overview). Languages for the succinct representation of utility functions include graphical models [1, 3, 16, 19], decision trees [5], valued constraint satisfaction problems [2], and bidding languages for combinatorial auctions [6, 22, 24]. Many different issues concerning preference representation languages are worth investigating:

- *Elicitation:* design algorithms to elicit preferences from an agent so as to get an output expressed in a given language.
- *Cognitive relevance:* assess the cognitive relevance of a language by measuring its proximity to the way human agents "know" their preferences and express them in natural language.
- *Expressive power:* identify the set of preference relations or utility functions that can be expressed in a given language.
- *Complexity:* for a given language, determine the computational complexity of tasks such as finding a non-dominated alternative, checking whether an alternative is preferred to another one, whether an alternative is non-dominated, or whether all non-dominated alternatives satisfy a given property.
- *Comparative succinctness:* given two languages L and L', check whether every preference structure expressible in L can also be expressed in L' without a significant (*i.e.* suprapolynomial) increase in size (in which case L' is said to be at least as succinct as L).

Elicitation and complexity have been the subject of much previous work that we will not recall here. Cognitive relevance is somewhat harder to assess, due to its non-technical nature, and to our knowledge it has been rarely studied (see [22] for a short discussion). The last two issues, expressive power and comparative succinctness, have been investigated to a lesser extent. Coste-Marquis *et al.* [11] give a systematic analysis of both issues for *ordinal* preferences, while several other authors [6, 9, 22, 24] investigate these issues for bidding languages for auctions and negotiation (which express valuation functions for bundles of goods).

In this paper we investigate expressive power for on one of the simplest languages for utility representation, where goals are specified as *propositional logic formulas*, and each goal is associated with a numerical weight. The utility of an alternative is then obtained by summing up the weights of the goals it satisfies. This language has been considered in many places, as have several of its variations (see e.g. [14, 18, 20, 23]). After covering some preliminaries and introducing the problems addressed in this paper in more formal detail in Section 2, we establish a range of correspondence results, between well-known classes of utility functions and different restrictions on the language of weighted propositional formulas, in Section 3. We then discuss, in less detail, succinctness issues in Section 4. Lastly, Section 5 discusses related work and further research directions.

2 Modelling Preferences

Let PS be a finite set of propositional symbols and let $n = |PS|$. $\mathcal{L}_{PS}$ is the propositional language built from PS using the operations of negation, conjunction and disjunction. For any formula $\varphi \in \mathcal{L}_{PS}$, $Var(\varphi)$ denotes the set of propositional symbols occurring in φ. $PS(k)$ is the set of all subsets of PS with at most k symbols (in particular, $PS(1)$ and $PS(n)$ are isomorphic to PS and 2^{PS}, respectively). Elements M of 2^{PS} could be bundles of indivisible goods, agreements in the context of multi-criteria decision making or, more generally, propositional worlds (assigning *true* to every symbol appearing in M and *false* to all other symbols).

2.1 Utility Functions

We now introduce the concept of a *utility function* over propositional worlds and recall the definitions of several well-known classes of utility functions.

Definition 1 *A utility function is a mapping* $u : 2^{PS} \rightarrow \mathbb{R}$.

- u *is normalised iff* $u(\{\,\}) = 0$.
- u *is non-negative iff* $u(X) \geq 0$ *for all* X.
- u *is monotonic iff* $u(X) \leq u(Y)$ *whenever* $X \subseteq Y$.
- u *is modular iff* $u(X \cup Y) = u(X) + u(Y) - u(X \cap Y)$ *for all* X *and* Y.
- u *is concave iff* $u(X \cup Y) - u(Y) \leq u(X \cup Z) - u(Z)$ *whenever* $Y \supseteq Z$.
- u *is convex iff* $u(X \cup Y) - u(Y) \geq u(X \cup Z) - u(Z)$ *whenever* $Y \supseteq Z$.
- u *is* k*-additive iff there exists a mapping* $u' : PS(k) \rightarrow \mathbb{R}$ *such that:*

$$u(X) \quad = \quad \sum \{u'(Y) \mid Y \subseteq X \text{ and } Y \in PS(k)\}$$

Intuitively, concavity means that marginal utility (of obtaining X) decreases as we move to a better starting position (namely from Z to Y). Observe hat u is convex iff $-u$ is concave. Utility functions that are both monotonic and normalised are also known as *capacities*.

The class of k-additive functions, the definition of which is inspired by work in fuzzy measure theory [17] and which recently also have found application in combinatorial auctions [10] and distributed negotiation [9], is probably less well-known than the other classes of functions mentioned in Definition 1. This class is useful in domains where synergies between different items are restricted to bundles of at most k elements. Observe that for $k = n$, *any* utility function is k-additive: $u'(\{\,\}) = u(\{\,\})$ and $u'(X)$ can be defined recursively as $u(X) - \sum_{Y \subset X} u'(Y)$ for all $X \neq \{\,\}$. Also, observe that the class of modular functions coincides with the class of 1-additive functions.

2.2 Weighted Formulas

An alternative approach to representing preferences uses *weighted propositional formulas* [20]. A weighted formula is a pair (φ, α), where φ is a propositional formula in the language $\mathcal{L}_{PS}$ and α is a numerical weight. Intuitively, the degree of satisfaction derived from a particular propositional world (bundle of goods, agreement) may be computed as the sum of the weights of the formulas satisfied by that world.

Definition 2 *A goal base is a set $G = \{(\varphi_i, \alpha_i)\}_i$ of pairs, each consisting of a satisfiable formula $\varphi_i \in \mathcal{L}_{PS}$ and a real number α_i. The utility function u_G generated by G is defined by*

$$u_G(M) \quad = \quad \sum\{\alpha_i \mid (\varphi_i, \alpha_i) \in G \text{ and } M \models \varphi_i\}$$

for all $M \in 2^{PS}$. G is called the generator of u_G.

We shall be interested in the following question:

> *Are there simple restrictions on goal bases such that the utility functions they generate enjoy simple structural properties?*

Interesting candidates for restrictions on formulas include restrictions on the length of formulas as well as the range of propositional connectives appearing in a formula.

Definition 3 *Let $H \subseteq \mathcal{L}_{PS}$ be a restriction on the set of propositional formulas and let $H' \subseteq \mathbb{R}$ be a restriction on the weights allowed in the specification of goals. Regarding formulas, we consider the following restrictions:*

- *A positive formula is a formula with no occurrence of $\neg$; a strictly positive formula is a positive formula that is not a tautology.*
- *A clause is a (possibly empty) disjunction of literals; a k-clause is a clause of length $\leq k$.*
- *A cube is a (possibly empty) conjunction of literals; a k-cube is a cube of length $\leq k$.*
- *A k-formula is a formula φ with $|Var(\varphi)| \leq k$.*

Regarding weights, we consider only the restriction to positive real numbers. Given two restrictions H and H', let $\mathcal{U}(H, H')$ be the class of utility functions that can be generated from goal bases conforming to restrictions H and H'.

Restrictions on formulas can also be combined (e.g. positive clauses are disjunctions of positive literals). We write "all" in case no specific restriction applies. For example, $\mathcal{U}$(*positive k-cubes*, *all*) is the class of utility functions generated by goal bases made up from positive k-cubes and where weights are not subject to any restrictions.

Two goal bases G and G' are said to be *equivalent* ($G \equiv G'$) iff they generate the same utility functions, *i.e.* iff $u_G = u_{G'}$. For instance, we have $\{(\psi \vee \chi, \alpha)\} \equiv \{(\psi, \alpha), (\chi, \alpha), (\psi \wedge \chi, -\alpha)\}$.

3 Correspondence Results

This section provides a whole range of answers to our earlier question regarding the existence of suitable restrictions on goal bases generating utility functions with simple structural properties.

3.1 Basic Results

It turns out that k-additivity plays a central role in characterising the utility functions corresponding to certain types of goal bases. This connection is at its most apparent in the case of positive k-cubes. A k-additive function can be represented by a mapping $u' : PS(k) \to \mathbb{R}$ (see

Definition 1). We can define a bijective function f from such mappings u' onto goal bases G with only positive k-cubes:

$$f : u' \mapsto \{(p_1 \wedge \cdots \wedge p_k, \alpha) \mid u'(\{p_1, \ldots, p_k\}) = \alpha\}$$

Clearly, the utility functions generated by u' and the goal base $f(u')$ are identical. Then, using equivalence-preserving transformations between goal bases, similar to that indicated at the end of Section 2, we can also establish the correspondence of k-additive functions to several other types of restriction. These results are summarised in the following proposition:

Proposition 1 *$\mathcal{U}$(positive k-cubes, all), $\mathcal{U}$(k-cubes, all), $\mathcal{U}$(k-clauses, all), $\mathcal{U}$(positive k-formulas, all), and $\mathcal{U}$(k-formulas, all) are all equal to the class of k-additive utility functions.*

The positive k-clauses do *not* generate the full set of k-additive utility functions, because (due to the fact that $\top$ is not a clause) positive k-clauses do not allow us to assign a non-zero utility to $\{\,\}$:

Proposition 2 *$\mathcal{U}$(positive k-clauses, all) is equal to the class of normalised k-additive utility functions.*

Recall that *any* utility function is k-additive for a sufficiently high value of k. Hence, the following correspondence results are easy consequences of the propositions above:

Proposition 3 *$\mathcal{U}$(positive cubes, all), $\mathcal{U}$(positive, all), $\mathcal{U}$(cubes, all), $\mathcal{U}$(clauses, all), and $\mathcal{U}$(all, all) are all equal to the class of all utility functions. $\mathcal{U}$(positive clauses, all) and $\mathcal{U}$(strictly positive, all) are equal to the class of normalised utility functions.*

The central argument in the proof of the next proposition is that the class of modular functions is equal to the class of 1-additive functions.

Proposition 4 *$\mathcal{U}$(literals, all) is equal to the class of modular utility functions; and $\mathcal{U}$(atoms, all) is equal to the class of normalised modular utility functions.*

3.2 Positive Weights

Next we study the classes of utility functions generated by positively weighted formulas. Unsurprisingly, such functions will be non-negative.

Proposition 5 *$\mathcal{U}$(all, positive) and $\mathcal{U}$(cubes, positive) are both equal to the class of non-negative utility functions.*

Again, clauses are less expressive than cubes:

Proposition 6 *$\mathcal{U}$(clauses, positive) is a proper subset of the class of non-negative utility functions.*

Proof. Inclusion of $\mathcal{U}(\textit{clauses}, \textit{positive})$ in the set of non-negative functions follows from Proposition 5. To show that the inclusion is strict, consider the following non-negative utility function:

$$u(\{p,q\}) = 1;\ u(\{p\}) = 0;\ u(\{q\}) = 0;\ u(\{\,\}) = 0$$

Suppose there exists a generator G of u using only positively weighted clauses. Let w_c be the weight associated with clause c. We obtain the following list of constraints:

$$\begin{array}{ll}
(1) & w_p + w_q + w_{p\vee q} + w_{\neg p\vee q} + w_{p\vee\neg q} + w_\top = 1 \\
(2) & w_p + w_{\neg q} + w_{p\vee q} + w_{p\vee\neg q} + w_{\neg p\vee\neg q} + w_\top = 0 \\
(3) & w_{\neg p} + w_q + w_{p\vee q} + w_{\neg p\vee q} + w_{\neg p\vee\neg q} + w_\top = 0 \\
(4) & w_{\neg p} + w_{\neg q} + w_{\neg p\vee q} + w_{p\vee\neg q} + w_{\neg p\vee\neg q} + w_\top = 0 \\
(5) & w_c \geq 0 \text{ for all clauses } c
\end{array}$$

Constraints (2), (3), (4) and (5) give $w_c = 0$ for any clause c, which contradicts (1). □

3.3 Monotonic Functions

The next result characterises the class of normalised monotonic utility functions, also known as *capacities.*

Proposition 7 *$\mathcal{U}(\textit{strictly positive}, \textit{positive})$ is equal to the class of normalised monotonic utility functions.*

Proof. For lack of space we can only give a brief sketch here. Clearly, any utility function generated by positive formulas with positive weights must be monotonic; and by Proposition 3, any function generated by strictly positive formulas is normalised. Hence, every $u \in \mathcal{U}(\textit{strictly positive}, \textit{positive})$ must be a capacity. For the converse, we sketch how to construct a goal base of positively weighted strictly positive formulas for any given capacity u. Consider the utility functions u^k (for $k = 1, \ldots, n$) defined as follows:

$$u^k(X) \quad = \quad \max_{\{x_1..x_k\}\subseteq X, x_1\neq x_2..\neq x_k} u(\{x_1, \ldots, x_k\})$$

For instance, $u^1(X) = \max_{x\in X} u(\{x\})$. We are going to show how to construct generators for u^1, $u^2 - u^1$, $u^3 - u^2$ and so forth; the union of these will then be a generator for u.
(Step 1) To construct a generator G^1 for u^1, order the elements p_i of PS such that $u(\{p_1\}) \leq \cdots \leq u(\{p_n\})$.

$$\begin{aligned} G^1 = \{\, & (p_1 \vee \cdots, \vee p_n, u(\{p_1\})), \\ & (p_2 \vee \cdots \vee p_n, u(\{p_2\}) - u(\{p_1\})), \ldots, \\ & (p_n, u(\{p_n\}) - u(\{p_{n-1}\}) \,\} \end{aligned}$$

(Step 2) To construct a generator for $u^{2-1} = u^2 - u^1$, let $\{X_1, \ldots, X_{\binom{n}{2}}\}$ be the set of all 2-ary subsets of PS, ordered in such a way that $u^{2-1}(X_i) \leq u^{2-1}(X_j)$ whenever $i < j$. Observe that $u^{2-1}(X_i)$ is non-negative (due to the monotonicity of u). Now define:

$$\begin{aligned} G^2 = \{\, & (\bigwedge X_1 \vee \cdots \vee \bigwedge X_{\binom{n}{2}}, u^{2-1}(X_1)), \\ & (\bigwedge X_2 \vee \cdots \vee \bigwedge X_{\binom{n}{2}}, u^{2-1}(X_2) - u^{2-1}(X_1)), \ldots, \\ & (\bigwedge X_{\binom{n}{2}}, u^{2-1}(X_{\binom{n}{2}}) - u^{2-1}(X_{\binom{n}{2}-1})) \,\} \end{aligned}$$

G^2 is a generator for $u^2 - u^1$. If we continue using the same method, we can construct generators $G^3, \ldots, G^n$ for $u^3 - u^2$ up to $u^n - u^{n-1}$. $\square$

To exemplify our construction, consider the capacity u with $u(\{p_1\}) = 2$, $u(\{p_2\}) = 5$ and $u(\{p_1, p_2\}) = 6$. We obtain the following goal base:

$$G \;=\; \{(p_1 \vee p_2, 2), (p_2, 3), (p_1 \wedge p_2, 1)\}$$

Also observe that we can model the full set of monotonic utilities by allowing a single goal $(\top, \alpha)$ with (possibly negative) weight α in a goal base that otherwise consists only of strictly positive formulas with positive weights. $\mathcal{U}(\textit{positive}, \textit{positive})$ is the set of non-negative monotonic utility functions.

3.4 Concave Functions

As a final correspondence result, we establish a connection between restrictions on goal bases and concave utility functions.

Proposition 8 *$\mathcal{U}(\textit{positive clauses}, \textit{positive})$ is a subset of the class of normalised concave monotonic utility functions.*

Proof. The fact that any function in $\mathcal{U}(\textit{positive clauses}, \textit{positive})$ is a capacity follows from Proposition 7. So the interesting part is to show that positive clauses with positive weights generate concave utility functions.

Let u be generated by a goal base G of positive clauses with positive weights and let X, Y and Z be propositional worlds such that $Y \supseteq Z$. For positive clauses φ, $X \cup Y \models \varphi$ together with $Y \not\models \varphi$ implies $X \models \varphi$, and $M \models \varphi$ implies $M' \models \varphi$ whenever $M \subseteq M'$. Hence:

$$\begin{array}{l} \{(\varphi, \alpha) \in G \mid X \cup Y \models \varphi \text{ and } Y \not\models \varphi\} \;\subseteq \\ \{(\varphi, \alpha) \in G \mid X \cup Z \models \varphi \text{ and } Z \not\models \varphi\} \end{array}$$

Because all weights α are positive, we immediately obtain the inequation characterising concavity: $u(X \cup Y) - u(Y) \leq u(X \cup Z) - u(Z)$. $\square$

Proposition 8 implies that *positive clauses with negative weights* generate only *convex* utility functions (albeit only negative ones).

4 Comparative Succinctness

Different restrictions on goal bases constitute different *languages* for describing utility functions. In this section, we make a first step towards analysing the comparative *succinctness* of such languages.

4.1 Defining Succinctness

A language L' for expressing utility functions is said to be *at least as succinct* as another language L iff there exists a polysize reduction for any utility function expressed in L to the same utility function expressed in L' (see also [8, 11]). In our case, languages are restrictions $\mathcal{U}(H, H')$.

Definition 4 *Let L and L' be two sets of goal bases. We say that L' is at least as succinct as L, denoted by $L \preceq L'$, iff there exist a mapping $f : L \to L'$ and a polynomial function p such that:*

- *$G \equiv f(G)$ for all $G \in L$; and*
- *$size(f(G)) \leq p(size(G))$ for all $G \in L$.*

Here the *size* of a goal base is the sum of the lengths of the formulas in that goal base. If $L \preceq L'$ and $L' \preceq L$, then L and L' are as succinct as each other: they express the same sets of utilities in the same order of size. It may also be the case that two languages are incomparable, that is, neither $L \preceq L'$ nor $L' \preceq L$ holds. The strict order associated with $\preceq$ is denoted by $\prec$.

We are interested in comparing the succinctness of different languages that have the same expressive power. Note that, if $H, H' \subseteq \mathcal{L}_{PS}$ and $H'' \subseteq \mathbb{R}$ with $\mathcal{U}(H, H") \equiv \mathcal{U}(H', H")$, then $H \subseteq H'$ implies $\mathcal{U}(H, H") \preceq \mathcal{U}(H', H")$. In this case the polysize reduction is simply the identity function.

4.2 An Incomparability Result

The most basic way of representing a utility function would be to explicitly list all propositional worlds with a non-zero utility. This directly corresponds to goal bases consisting solely of cubes, each of which contains either p or $\neg p$ as a conjunct for every propositional symbol $p \in PS$ (let us refer to such cubes as *n-cubes*). Clearly, $\mathcal{U}(n\text{-}cubes, all)$ is equal to the class of all utility functions. An alternative form of representation is based on the notion of k-additivity and uses the auxiliary function u' to define utility functions [9, 17]. This directly corresponds to goal bases consisting only of positive cubes.

As shown elsewhere [9], these two forms of representation are *incomparable*. Hence, $\mathcal{U}(n\text{-}cubes, all)$ and $\mathcal{U}(positive\ cubes, all)$ are also incomparable. The following two utility functions can be used to prove the mutual lack of a polysize reduction:

- The function $u_1(M) = |M|$ can be generated by a goal base of just n positive cubes of length 1, but we require $2^n - 1$ n-cubes to generate u_1.
- The function u_2, with $u_2(M) = 1$ for $|M| = 1$ and $u_2(M) = 0$ otherwise, can be generated by a goal base of n n-cubes, but we require $2^n - 1$ positive cubes to generate u_2.

4.3 The Efficiency of Negation

Recall that both $\mathcal{U}(positive\ cubes, all)$ and $\mathcal{U}(cubes, all)$ are equal to the class of all utility functions (Proposition 3). However, as the next proposition states, the representation of utility functions based on cubes is strictly more succinct than the representation based on positive cubes alone:

Proposition 9 $\mathcal{U}(positive\ cubes, all) \prec \mathcal{U}(cubes, all)$.

Proof. Clearly, $\mathcal{U}(positive\ cube, all) \preceq \mathcal{U}(cubes, all)$, because every positive cube is also a cube. To show that the representation based on general cubes is *strictly* more succinct, we consider the utility function u with $u(\{\}) = 1$ and $u(M) = 0$ for all $M \neq \{\}$. If $PS = \{p_1, \ldots, p_n\}$, then u is generated by the goal base $G = \{(\neg p_1 \wedge \cdots \wedge \neg p_n, 1)\}$. That is, using general cubes,

u can be generated from a goal base with a single weighted formula of length n. Now, consider the following goal base using positive cubes alone:

$$G' \quad = \quad \{(\bigwedge X, (-1)^{|X|}) \mid X \subseteq PS\}$$

That is, every cube of length k gets the weight $(-1)^k$. Observe that G' generates u, *i.e.* $u = u_{G'}$:

$$u_{G'}(M) = \sum_{X \subseteq M} (-1)^{|X|} = \sum_{k=0}^{|M|} \binom{|M|}{k} (-1)^k = 0^{|M|}$$

Next, we are going to show that the goal base generating u is in fact *uniquely* determined if only positive cubes are available:[1] The only positive cube satisfied by $\{\,\}$ is $\top$. Hence, we must have $(\top, 1) \in G'$. But then we must have $(p, -1) \in G'$ for every propositional symbol $p \in PS$ to ensure $u(\{p\}) = 0$. This in turn fully determines the weights of cubes with two conjuncts, and so forth. Thus, because the size of G' is *exponential* in the number of propositional symbols in PS and because no other goal base using positive cubes can generate u, the language based on cubes is indeed strictly more succinct than the language based on positive cubes. $\square$

This result shows that the inclusion of negation into a representation language for cardinal preferences can make that language strictly more succinct.

5 Conclusion

We have further analysed the language of weighted propositional formulas previously studied by several authors. Most of our results concern the *expressive power* of this language; we have established several correspondences between certain types of weighted formulas and well-known classes of utility functions. We have then studied the *comparative succinctness* of languages based on different types of weighted formulas that can represent the same class of utility functions.

In this paper, we have focussed exclusively on the additive interpretation of weighted propositional formulas. Other aggregation functions can be considered, such as maximum or more general functions (e.g. [2] in the CSP framework). Weighted formulas together with maximum as the aggregation function have been considered in various places, including the so-called XOR language for combinatorial auctions [24], which furthermore restricts formulas to positive cubes. Comparing the simple (but yet expressive) framework of weighted goals with the various languages designed for combinatorial auctions (a synthesis of which is given in [22]) is an issue for further research.

While this paper establishes a number of interesting results on the expressive power and comparative succinctness of weighted formulas for cardinal preference modelling, it also raises a multitude of further questions. As concerns expressive power, further correspondence results are needed to fully understand the relationship between restrictions on goal bases and different classes of utility functions. As concerns succinctness, our observation that the inclusion of negation into a language significantly improves succinctness in the case of cubes immediately raises the question whether this remains true for more general formulas: Is $\mathcal{U}(all, all)$ strictly

[1]Without loss of generality, we assume that no goal base contains two or more logically equivalent formulas.

more succinct than $\mathcal{U}(\textit{positive}, \textit{all})$? We conjecture: yes. Another interesting question would be whether $\mathcal{U}(\textit{all}, \textit{all})$ is strictly more succinct than $\mathcal{U}(\textit{cubes}, \textit{all})$. Again, we conjecture: yes.

A further important area for future research concerns the *complexity* of working with different languages of weighted formulas. For instance, let Max-Utility(H, H') be the following decision problem: given a goal base $G \in \mathcal{U}(H, H')$ and an integer K, check whether there exists a world $M \in 2^{PS}$ such that $u_G(M) \geq K$. Obviously, Max-Utility is in NP for the full language of weighted formulas, since $u_G(M) \geq K$ can be checked in polynomial time. Clearly as well, the general problem is NP-complete, due to its straightforward reduction from Sat [15]. More interestingly, for sublanguages such as $\mathcal{U}(k\textit{-clauses}, \textit{positive})$, Max-Utility is also NP-complete, even for $k = 2$. This can be shown via a reduction from Max2Sat [15]. Simpler languages such as $\mathcal{U}(\textit{literals}, \textit{all})$, on the other hand, give rise to polynomial decision problems: assuming that G contains every literal exactly once (possibly with weight 0), making a propositional symbol p true iff the weight of p is greater than the weight of $\neg p$ results in an alternative with maximal utility. Max-Utility$(\textit{positive}, \textit{positive})$ is also in P, because making *all* propositional symbols true will result in maximal utility. We shall leave a full analysis of these issues to a future occasion.

References

[1] F. Bacchus and A. J. Grove. Utility independence in a qualitative decision theory. In *Proc. 5th International Conference on Principles of Knowledge Representation and Reasoning (KR-1996)*. Morgan Kaufmann Publishers, 1996.

[2] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Frameworks, properties and comparison. *Constraints*, 4(3):199–240, 1999.

[3] C. Boutilier, F. Bacchus, and R. Brafman. UCP-networks: A directed graphical representation of conditional utilities. In *Proc. 17th Conference on Uncertainty in Artificial Intelligence (UAI-2001)*. Morgan Kaufmann Publishers, 2001.

[4] C. Boutilier, R. Brafman, H. Hoos, and D. Poole. Reasoning with conditional *ceteris paribus* preference statements. In *Proc. 15th Conference on Uncertainty in Artificial Intelligence (UAI-1999)*. Morgan Kaufmann Publishers, 1999.

[5] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI-1995)*. Morgan Kaufmann Publishers, 1995.

[6] C. Boutilier and H. Hoos. Bidding languages for combinatorial auctions. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*. Morgan Kaufmann Publishers, 2001.

[7] S. J. Brams, D. M. Kilgour, and W. S. Zwicker. The paradox of multiple elections. *Social Choice and Welfare*, 15(2):211–236, 1998.

[8] M. Cadoli, F. Donini, P. Liberatore, and M. Schaerf. Comparing space efficiency of propositional knowledge representation formalisms. In *Proc. 5th International Conference on Principles of Knowledge Representation and Reasoning (KR-1996)*. Morgan Kaufmann Publishers, 1996.

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

[10] V. Conitzer, T. W. Sandholm, and P. Santi. Combinatorial auctions with k-wise dependent valuations. In *Proc. 20th National Conference on Artificial Intelligence (AAAI-05)*. AAAI Press, 2005.

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

[12] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

[13] J. Doyle and M. P. Wellman. Preferential semantics for goals. In *Proc. 9th National Conference on Artificial Intelligence (AAAI-1991)*. AAAI Press, 1991.

[14] F. Dupin de Saint-Cyr, J. Lang, and T. Schiex. Penalty logic and its link with Dempster-Shafer theory. In *Proc. 10th Conference on Uncertainty in Artificial Intelligence (UAI-1994)*. Morgan Kaufmann Publishers, 1994.

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

[16] C. Gonzales and P. Perny. GAI networks for utility elicitation. In *Proc. 9th International Conference on Principles of Knowledge Representation and Reasoning (KR-2004)*. AAAI Press, 2004.

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

[18] P. Haddawy and S. Hanks. Representations for decision-theoretic planning: Utility functions for deadline goals. In *Proc. 4th International Conference on Principles of Knowledge Representation and Reasoning (KR-1994)*. Morgan Kaufmann Publishers, 1992.

[19] P. La Mura and Y. Shoham. Expected utility networks. In *Proc. 15th Conference on Uncertainty in Artificial Intelligence (UAI-1999)*. Morgan Kaufmann Publishers, 1999.

[20] C. Lafage and J. Lang. Logical representation of preferences for group decision making. In *Proc. 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000)*. Morgan Kaufmann Publishers, 2000.

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

[22] N. Nisan. Bidding languages for combinatorial auctions. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. MIT Press, 2006.

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

[24] T. W. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.