Expressive Power of Weighted Propositional Formulas for Cardinal Preference Modelling

Yann Chevaleyre, Ulle Endriss, Jérôme Lang

Expressive Power of Weighted Propositional Formulas for Cardinal Preference Modelling

Yann Chevaleyre*, Ulle Endriss†, Jérôme Lang‡

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 twofold: (1) to establish correspondences between certain types of weighted formulas and well-known classes of utility functions (such as monotonic, concave or k-additive functions); and (2) to obtain results on the comparative succinctness of different types of weighted formulas for representing the same class of utility functions.

Key words : Preference representation, logic-based languages, expressive power, comparative succinctness, computational complexity

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. These preferences can be either ordinal (*i.e.* preference relations, typically weak orders) or cardinal (*i.e.* utility functions). We make use of the generic word *preference structure* for either a preference relation or a

*LAMSADE, Université Paris-Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France. yann.chevaleyre@lamsade.dauphine.fr

†ILLC, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands. ulle@illc.uva.nl

‡IRIT, Université Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse Cedex 04, France. lang@irit.fr.fr

utility function. 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.[3] Examples are numerous: in combinatorial auctions and negotiation over resources [Cramton *et al.*, 2006; Chevaleyre *et al.*, 2006], an alternative is an assignment of each good to an agent; in multiple issue referenda [Brams *et al.*, 1998], 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. Languages for the succinct representation of ordinal preferences include languages of *ceteris paribus* statements, which range from very expressive languages [Doyle and Wellman, 1991] to syntactical restrictions such as CP-nets [Boutilier *et al.*, 1999], where a weaker expressivity is compensated for by the availability of efficient elicitation and optimisation techniques. They also include languages based on conditional logics, prioritised logics, and prioritised constraint satisfaction problems (see e.g. Lang [2004] for an overview). Languages for the succinct representation of utility functions include graphical models [Bacchusand Grove, 1996; La Mura and Shoham, 1999; Boutilier *et al.*, 2001; Gonzales and Perny, 2004], decision trees [Boutilier *et al.*, 1995], valued constraint satisfaction problems [Bistarelli *et al.*, 1999], and bidding languages for combinatorial auctions [Nisan, 2006; Boutilier and Hoos, 2001; Sandholm, 2002].

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.

[3] Another situation where preference representation is non-trivial is when the set of alternatives is continuous. We leave this issue aside and focus on finite domains only.

- *Expressive power:* identify the set of preference structures 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', determine whether every preference structure that can be expressed in L can also be expressed in L' without a significant (that is, supra-polynomial) 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 Nisan [2006] for a short discussion). Expressive power and comparative succinctness, have been investigated to a lesser extent. Coste-Marquis *et al.* [2004] give a systematic analysis of both issues for *ordinal* preferences, while several other authors [Boutilier and Hoos, 2001; Sandholm, 2002; Chevaleyre *et al.*, 2004; Nisan, 2006] 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 and succinctness for 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 [Pinkas, 1991; Haddawy and Hanks, 1992; Dupin de Saint-Cyr *et al.*, 1994; Lafage and Lang, 2000].

After covering some preliminaries and introducing the problems addressed in this paper in more formal detail in the next section, we first investigate expressivity issues. We focus on a number of possible restrictions on both formulas and weights, and identify the corresponding classes of utility functions. While the results are obvious in extreme cases (if no restriction is imposed, all utility functions can be expressed, with maximal succinctness; if only atomic formulas are allowed then only linear functions can be expressed), there appear to be many cases for which they are non-trivial and particularly interesting, because they correspond to intermediate classes which may realise a good trade-off between simplicity and efficiency. We then present, in less detail, initial results concerning the comparative succinctness of different preference languages. In the final section, we discuss related work and further research directions. In particular, we point out interesting directions for future work regarding the computational complexity of working with different languages based on weighted propositional formulas.

2 Modelling Preferences

In this section we introduce two approaches to modelling cardinal preferences: by means of classical utility functions and by means of weighted propositional formulas.

We first fix some basic notation. 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 elements (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, coalitions of agents in the context of cooperative games 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 (Utility functions) *A utility function is a mapping* $u : 2^{PS} \to \mathbb{R}$.

- 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 subadditive iff* $u(X \cup Y) \leq u(X) + u(Y) - u(X \cap Y)$ *for all* X *and* Y.
- u *is superadditive iff* $u(X \cup Y) \geq u(X) + u(Y) - u(X \cap Y)$ *for all* X *and* Y.
- u *is concave iff* $u(X \cup Y) - u(Y) \leq u(X \cup Z) - u(Z)$ *for all* X *whenever* $Y \supseteq Z$.
- u *is convex iff* $u(X \cup Y) - u(Y) \geq u(X \cup Z) - u(Z)$ *for all* X *whenever* $Y \supseteq Z$.
- u *is* k*-additive iff there exists a mapping* $m : PS(k) \to \mathbb{R}$ *such that (for all* X*):*

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

Intuitively, concavity means that marginal utility (of obtaining X) decreases as we move to a better starting position (namely from Z to Y). Observe that u is convex iff $-u$ is concave and, similarly, u is superadditive iff $-u$ is subadditive. All concave functions are also subadditive and all convex functions are superadditive (set $Z = X \cap Y$). The class of modular functions is the intersection of the classes of subadditive and superadditive functions. 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 (see e.g. [Grabisch, 1997]) and which recently also have found application in combinatorial auctions [Conitzer *et al.*, 2005] and distributed negotiation [Chevaleyre *et al.*, 2004], 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. We recall the well-known fact that for $k = n$, *any* utility function is k-additive: $m(\{\,\}) = u(\{\,\})$ and $m(X)$ can be defined recursively as $u(X) - \sum_{Y \subset X} m(Y)$ for all $X \neq \{\,\}$. Moreover, the function m such that $u(X) = \sum\{m(Y) \mid Y \subseteq X\}$ is *uniquely* determined; the mapping $u \mapsto m$ is known as the *Möbius inversion* ([Rota, 1964]; see also [Shafer, 1976; Gilboa and Schmeidler, 1992]). Also, the class of modular functions coincides with the class of 1-additive functions. This may be seen as follows. Let X be any non-empty set in 2^{PS} and let $x \in X$. Then the equation characterising modularity implies $u(X) = u(X \setminus \{x\}) + [u(\{x\}) - u(\{\,\})]$. If we apply this step recursively for every element of X, then we end up with the following equation:

$$u(X) \;=\; u(\{\,\}) \;+\; \sum_{x \in X} [u(\{x\}) - u(\{\,\})]$$

Choosing $m(\{\,\}) = u(\{\,\})$ and $m(\{x\}) = u(\{x\}) - u(\{\,\})$, this shows that modularity implies 1-additivity. The converse is easily seen to hold as well.

2.2 Weighted Formulas

An alternative approach to representing preferences uses *weighted propositional formulas*. 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 representing the relative importance of that formula. Intuitively, the degree of satisfaction derived from a particular propositional world (bundle of goods, agreement, coalition) is the sum of the weights of the formulas satisfied by that world.

Definition 2 (Goal bases) *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*

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

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

Summing up the individual weights is particularly suited for modelling utility functions, but other aggregation functions have been investigated as well [Lafage and Lang, 2000]. In this paper we are going to 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. The most obvious restriction on weights would be to allow only positive numbers.

Definition 3 (Restrictions) *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 set of weights allowed in the specification of goals. For 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 $|Var(\varphi)| \leq k$.*

As for weights, we consider only the restriction to the positive reals. 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 the 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, $\mathcal{U}(\textit{positive } k\textit{-cubes}, \textit{all})$ 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. We are also going to consider restrictions to both atoms (propositional symbols) and literals (atoms and their negations). Note that $\top$ is a cube (of length 0), but not a clause (nor is it a literal). The empty clause is equivalent to $\bot$, *i.e.* it is not of interest here, because goals are required to be satisfiable.

Two goal bases G and G' are said to be *equivalent* (written $G \equiv G'$) iff they generate the same utility functions, *i.e.* iff $u_G = u_{G'}$. The following lemma introduces two equivalence-preserving transformations on goal bases. It shows how to eliminate both negations and disjunctions from inside a conjunction.

Lemma 1 *The following equivalences hold for all goal bases G, formulas $\varphi, \psi, \chi \in \mathcal{L}_{PS}$ and weights $\alpha \in \mathbb{R}$:*

(i) $G \cup \{(\varphi \wedge \neg\psi, \alpha)\} \equiv G \cup \{(\varphi, \alpha), (\varphi \wedge \psi, -\alpha)\}$

(ii) $G \cup \{(\varphi \wedge (\psi \vee \chi), \alpha) \equiv$
$G \cup \{(\varphi \wedge \psi, \alpha), (\varphi \wedge \chi, \alpha), (\varphi \wedge \psi \wedge \chi, -\alpha)\}\}$

Proof. The claims are easily verified by considering all (eight) possible ways of assigning truth values to the formulas φ, ψ and χ. □

A special case of part (i) shows how to eliminate a negation from the outside of a formula: $\{(\neg\psi, \alpha)\}$ can be rewritten as $\{(\top, \alpha), (\psi, -\alpha)\}$. Similarly, setting $\varphi = \top$ in part (ii) provides us with a way of transforming a disjunction into a set of conjunctions: $\{(\psi \vee \chi, \alpha)\}$ can be replaced by $\{(\psi, \alpha), (\chi, \alpha), (\psi \wedge \chi, -\alpha)\}$.

3 Correspondence Results

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

3.1 Basic Results

It turns out that the notion of k-additivity plays a central role in characterising the classes of utility functions corresponding to certain types of goal bases. This connection is at its most apparent in the case of positive k-cubes.

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

Proof. A k-additive function can be represented by a mapping $m : PS(k) \to \mathbb{R}$ (see Definition 1). We can define a bijective function f from such mappings m onto goal bases G with only positive k-cubes:

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

Clearly, the utility functions generated by m and the goal base $f(m)$ are identical. □

Observe that *negative* k-cubes (*i.e.* conjunctions of negative literals of length $\leq k$) also generate the set of all k-additive functions. This may be seen as follows. Let $B(u)$ be

defined by $B(u)(M) = u(\overline{M})$ for all M. If u is generated by G, then $B(u)$ is generated by $\overline{G}$ obtained by replacing every literal in every formula of G by its negation, which shows that $\mathcal{U}(\textit{negative } k\textit{-cubes}, \textit{all}) = B(\mathcal{U}(\textit{positive } k\textit{-cubes}, \textit{all}))$, which is equal to the set of all k-additive utility functions (since the latter is closed under B). In fact, for several of our correspondence results on positive formulas below, there exist similar results for formulas where all literals are negative, even though we are not going to specifically report these here.

Proposition 2 *The following sets are also all equal to the class of k-additive utility functions:*

- $\mathcal{U}(k\textit{-cubes}, \textit{all})$ *and* $\mathcal{U}(k\textit{-clauses}, \textit{all})$*;*
- $\mathcal{U}(\textit{positive } k\textit{-formulas}, \textit{all})$ *and* $\mathcal{U}(k\textit{-formulas}, \textit{all})$.

Proof. Any positive k-cube $(p_1 \wedge \cdots \wedge p_k, \alpha)$ can be rewritten as a set of k-clauses (using an arbitrary additional propositional symbol p):

$$\{(\neg p_1 \vee \cdots \vee \neg p_k, -\alpha), (p, \alpha), (\neg p, \alpha)\}$$

Hence, $\mathcal{U}(\textit{positive } k\textit{-cubes}, \textit{all}) \subseteq \mathcal{U}(k\textit{-clauses}, \textit{all})$. Clearly, $\mathcal{U}(\textit{positive } k\textit{-cubes}, \textit{all})$ is also included in both $\mathcal{U}(k\textit{-cubes}, \textit{all})$ and $\mathcal{U}(\textit{positive } k\textit{-formulas}, \textit{all})$, and all of the classes mentioned are included in $\mathcal{U}(k\textit{-formulas}, \textit{all})$.

Using Lemma 1, we can transform any goal base consisting of k-formulas into a goal base of positive k-cubes, *i.e.* we also get $\mathcal{U}(k\textit{-formulas}, \textit{all}) \subseteq \mathcal{U}(\textit{positive } k\textit{-cubes}, \textit{all})$. Hence, all of the sets of utility functions mentioned earlier are equivalent. The claim then follows immediately from Proposition 1. □

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 $\{\,\}$. We therefore obtain the following weaker result:

Proposition 3 $\mathcal{U}(\textit{positive } k\textit{-clauses}, \textit{all})$ *is equal to the class of normalised k-additive utility functions.*

Proof. First observe that positive k-clauses augmented with $\top$ can generate all k-additive utility functions. This immediately follows from case (ii) of Lemma 1 and Proposition 1. Without lack of generality, we may assume that any goal base G of positive k-clauses augmented with $\top$ includes exactly one weighted goal of the form $(\top, \alpha)$. It then remains to be shown that u_G is normalised iff $\alpha = 0$. This is clearly so, because $u_G(\{\,\}) = \alpha$

holds due to the fact that $\{\}$ falsifies all positive clauses. □

Next we list a number of further basic results, all of which are simple consequences of the results on k-additive utility functions for the special cases of $k = n$ and $k = 1$.

Proposition 4 *The following sets are all equal to the class of all utility functions:*

- $\mathcal{U}(\textit{positive cubes}, \textit{all})$ *and* $\mathcal{U}(\textit{positive}, \textit{all})$*;*
- $\mathcal{U}(\textit{cubes}, \textit{all})$*,* $\mathcal{U}(\textit{clauses}, \textit{all})$ *and* $\mathcal{U}(\textit{all}, \textit{all})$*.*

Proof. Recall that *any* utility function is k-additive for a sufficiently high value of k. The claim then follows from Propositions 1 and 2. □

$\mathcal{U}(\textit{positive cubes}, \textit{all})$ corresponds to the *marginal contribution nets* of Ieong and Shoham [2005], who also point out that this language is fully expressive.

Proposition 5 $\mathcal{U}(\textit{positive clauses}, \textit{all})$ *is equal to the class of normalised utility functions.*

Proof. This is a corollary to Proposition 3. □

Proposition 6 $\mathcal{U}(\textit{strictly positive}, \textit{all})$ *is also equal to the class of normalised utility functions.*

Proof. As any positive clause is a strictly positive formula, by Proposition 5, any normalised function must belong to $\mathcal{U}(\textit{strictly positive}, \textit{all})$. Vice versa, if G is a set of strictly positive formulas then $u_G(\{\}) = 0$, because $\{\}$ falsifies all strictly positive formulas. □

Proposition 7 $\mathcal{U}(\textit{literals}, \textit{all})$ *is equal to the class of modular utility functions.*

Proof. First recall that the class of modular functions is equal to the class of 1-additive functions. Therefore, by Proposition 2, $\mathcal{U}(1\textit{-cubes}, \textit{all})$ is equal to the class of modular functions. The set of 1-cubes is the set of literals together with $\top$. The claim then follows from the fact that we can rewrite $\{(\top, \alpha)\}$ as $\{(p, \alpha), (\neg p, \alpha)\}$ using any propositional symbol p. □

Proposition 8 $\mathcal{U}(\textit{atoms}, \textit{all})$ *is equal to the class of normalised modular utility functions.*

Proof. Atoms are strictly positive literals, *i.e.* the claim follows from Propositions 6 and 7. □

3.2 Non-negative Functions

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

Proposition 9 *$\mathcal{U}(all, positive)$ and $\mathcal{U}(cubes, positive)$ are both equal to the class of non-negative utility functions.*

Proof. It is obvious that $\mathcal{U}(all, positive)$, and *a fortiori* $\mathcal{U}(cubes, positive)$, are contained in the set of all non-negative utility functions. For the converse inclusion, it is enough to show that any non-negative utility function can be generated by positively weighted cubes. So suppose u is such a non-negative utility function and define $G = \{(form(M), u(M)) \mid M \in 2^{PS}\}$, where:

$$form(M) \;=\; \bigwedge\{x \mid x \in M\} \wedge \bigwedge\{\neg x \mid x \in PS \setminus M\}$$

We have $u = u_G$, *i.e.* u is in $\mathcal{U}(cubes, positive)$). □

Again, clauses are less expressive than cubes:[4]

Proposition 10 *$\mathcal{U}(clauses, positive)$ is a proper subset of the class of non-negative utility functions.*

Proof. Inclusion of $\mathcal{U}(clauses, positive)$ in the set of non-negative functions follows from Proposition 9. 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:

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

[4] But observe that the restrictions on the functions that can still be expressed are different than for Proposition 5. While positive clauses with general weights generate all normalised functions, general clauses with positive weights do not only *not* generate all normalised (non-negative) utility functions, but also some non-normalised functions.

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

Likewise, $\mathcal{U}(k\text{-}\textit{clauses}, \textit{positive})$ is a proper subset of the class of non-negative k-additive utility functions.

3.3 Monotonic Functions

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

Proposition 11 *$\mathcal{U}(\textit{strictly positive}, \textit{positive})$ is equal to the class of normalised monotonic utility functions.*

Proof. Clearly, any utility function generated by positive formulas with positive weights must be monotonic; and by Proposition 6, 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:

$$u^k(X) \;=\; \max\{u(X') \mid X' \subseteq X \text{ and } |X'| \leq k\}$$

For instance, $u^1(X) = \max_{x \in X} u(\{x\})$ and $u^n = u$ (because of monotonicity). 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 the utility function u^n, and hence for u.

(Step 1) To construct a generator G^1 for u^1, order the elements p_i of PS such that $u(\{p_1\}) \leq \cdots \leq u(\{p_n\})$.

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

Clearly, G^1 is a generator for u^1.

(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 $u^n - u^{n-1}$. The union of $G^1, \ldots, G^n$ will then be a generator for the sum of $u^1, u^2 - u^1, \ldots, u^n - u^{n-1}$; that is, it will be a generator for $u = u^n$. □

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

(Step 1) Ordering the elements of $\{p_1, p_2\}$ gives $u(\{p_1\}) < u(\{p_2\}$, therefore, $G^1 = \{(p_1 \vee p_2, 2), (p_2, 3)\}$. G^1 is a generator for u^1, where $u^1(\{\}) = 0$, $u^1(\{p_1\}) = 2$, $u^1(\{p_2\}) = 5$, $u^1(\{p_1, p_2\}) = 5$.

(Step 2) Since $\{p_1, p_2\}$ is the only 2-ary subset of $\{p_1, p_2\}$, $G^2 = \{(p_1 \wedge p_2, u^{2-1}(\{p_1, p_2\}))\}$. Now, $u^{2-1}(\{p_1, p_2\}) = u^2(\{p_1, p_2\}) - u^1(\{p_1, p_2\}) = u(\{p_1, p_2\}) - 5 = 1$. Therefore, we obtain the following goal base:

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

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

Proposition 12 *$\mathcal{U}(\textit{positive clauses}, \textit{positive})$ is a subset of the class of normalised concave monotonic utility functions.*

Proof. The fact that any utility function from the set $\mathcal{U}(\textit{positive clauses}, \textit{positive})$ is a capacity follows from Proposition 11. 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:

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

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

We do not know whether the converse inclusion holds as well. Note that Proposition 12 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 [Cadoli *et al.*, 1996; Coste-Marquis *et al.*, 2004]). In our case, languages are restrictions $\mathcal{U}(H, H')$ or, more generally, sets of goal bases.

Definition 4 (Succinctness) *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 $\prec$ (*i.e.* $L \prec L'$ iff $L \preceq L'$ but not $L' \preceq L$).

We are interested in comparing the succinctness of different languages that have the same expressive power (*i.e.* that can generate the same class of utility functions). 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. We call this the *explicit form*. 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, $\mathcal{U}(n\text{-}\textit{cubes}, \textit{all})$ is equal to the class of all utility functions.

As discussed earlier, the concept of k-additivity gives rise to a different representation, which we call the k-*additive form*. The k-additive form directly corresponds to goal bases consisting only of positive cubes (see proof of Proposition 1). As shown elsewhere [Chevaleyre *et al.*, 2004], the explicit form and the k-additive form of representing utility functions are *incomparable* with respect to succinctness. This means that $\mathcal{U}(n\text{-}\textit{cubes}, \textit{all})$ and $\mathcal{U}(\textit{positive cubes}, \textit{all})$ are also incomparable. The following two utility functions can be used to prove the mutual lack of a polysize reduction (details may be found in [Chevaleyre *et al.*, 2004]):[5]

- 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 $\mathcal{U}(\textit{positive cubes}, \textit{all})$ and $\mathcal{U}(\textit{cubes}, \textit{all})$ are equal to the class of all utility functions (Proposition 4). However, as the next proposition states, the representation of

[5] In that paper [Chevaleyre *et al.*, 2004], cardinality rather than size is used as a measure for succinctness. Note, however, that comparative succinctness results coincide for the two approaches as long as only formulas of polynomial length occur (as is the case for cubes of any kind).

utility functions based on cubes is strictly more succinct than the representation based on positive cubes alone:[6]

Proposition 13 $\mathcal{U}(\textit{positive cubes}, \textit{all}) \prec \mathcal{U}(\textit{cubes}, \textit{all})$.

Proof. Clearly, $\mathcal{U}(\textit{positive cube}, \textit{all}) \preceq \mathcal{U}(\textit{cubes}, \textit{all})$, because every positive cube is also a cube (*i.e.* the polysize reduction here is identity). To show that the representation based on general cubes is *strictly* more succinct, we consider the family of utility functions u_n, for $n \geq 1$, where $u_n : 2^{\{p_1,\ldots,p_n\}} \to \mathbb{R}$ is defined by $u_n(\{\,\}) = 1$ and $u_n(M) = 0$ for all $M \neq \{\,\}$. u_n is generated by the goal base $G = \{(\neg p_1 \wedge \cdots \wedge \neg p_n, 1)\}$. That is, using general cubes, u_n 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:

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

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

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

Next, the Möbius inversion shows that the goal base generating u_n is in fact *uniquely* determined if only positive cubes are available:[7] Indeed, 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_n, the language based on cubes is indeed strictly more succinct than the language based on positive cubes. □

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

[6] This has also been observed by Ieong and Shoham [2005].

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

Formulas	**Weights**		**Utility Functions**	**Reference**
cubes/clauses/all	general	$=$	all	Prop. 4
positive cubes/formulas	general	$=$	all	Prop. 4
positive clauses	general	$=$	normalised	Prop. 5
strictly positive formulas	general	$=$	normalised	Prop. 6
k-cubes/clauses/formulas	general	$=$	k-additive	Prop. 2
positive k-cubes/formulas	general	$=$	k-additive	Prop. 1 & 2
positive k-clauses	general	$=$	normalised k-additive	Prop. 3
literals	general	$=$	modular	Prop. 7
atoms	general	$=$	normalised modular	Prop. 8
cubes/formulas	positive	$=$	non-negative	Prop. 9
clauses	positive	$\subset$	non-negative	Prop. 10
strictly positive formulas	positive	$=$	normalised monotonic	Prop. 11
positive clauses	positive	$\subseteq$	normalised concave monotonic	Prop. 12

Table 1: Summary of Correspondence Results

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. Our correspondence results are summarised in Table 1. We have then made initial steps towards analysing the *comparative succinctness* of languages based on different types of weighted formulas that can represent the same class of utility functions. In particular, we have seen that the language of weighted cubes, while not more expressive, is strictly more succinct than the language based on positive cubes.

In this paper, we have focussed exclusively on the additive interpretation of weighted propositional formulas. Other aggregation functions can be considered, such as maximum [Dubois *et al.*, 1994] or more general functions (see, for instance, the work of Bistarelli *et al.* [1999] in the CSP framework). Weighted formulas together with maximum as the aggregation function have been considered in various places, including for instance the so-called XOR language for combinatorial auctions [Sandholm, 2002], 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 by Nisan [2006]) 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 open questions. As concerns expressive power, further corre-

spondence results are needed to fully understand the relationship between restrictions on goal bases and different classes of utility functions. For instance, it would be very interesting to obtain precise characterisations of the classes of *superadditive* and *subadditive* functions in terms of goal bases. 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 $\mathcal{U}(\textit{all}, \textit{all})$ strictly 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 [Garey and Johnson, 1979]. 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 [Garey and Johnson, 1979].

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$(positive, \textit{positive})$ 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.

This paper has previously appeared in the Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-2006).

References

[Bacchus and Grove, 1996] 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.

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

[Boutilier and Hoos, 2001] 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.

[Boutilier *et al.*, 1995] 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.

[Boutilier *et al.*, 1999] 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.

[Boutilier *et al.*, 2001] 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.

[Brams *et al.*, 1998] Steven J. Brams, D. Marc Kilgour, and William S. Zwicker. The paradox of multiple elections. *Social Choice and Welfare*, 15(2):211–236, 1998.

[Cadoli *et al.*, 1996] 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.

[Chevaleyre *et al.*, 2004] Yann Chevaleyre, Ulle Endriss, Sylvia Estivie, and Nicolas 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.

[Chevaleyre *et al.*, 2006] Yann Chevaleyre, Paul E. Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaître, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A. Rodríguez-Aguilar, and Paulo Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

[Conitzer *et al.*, 2005] Vincent Conitzer, Tuomas W. Sandholm, and Paolo Santi. Combinatorial auctions with k-wise dependent valuations. In *Proc. 20th National Conference on Artificial Intelligence (AAAI-05)*. AAAI Press, 2005.

[Coste-Marquis *et al.*, 2004] 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.

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

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

[Dubois *et al.*, 1994] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In D. M. Gabbay et al., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 439–513. Oxford University Press, 1994.

[Dupin de Saint-Cyr *et al.*, 1994] 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.

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

[Gilboa and Schmeidler, 1992] I. Gilboa and D. Schmeidler. Canonical representation of set functions. Technical report, Northwestern University Kellogg Graduate School of Management, 1992. Discussion Paper No. 986.

[Gonzales and Perny, 2004] 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.

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

[Haddawy and Hanks, 1992] 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.

[Ieong and Shoham, 2005] Samuel Ieong and Yoav Shoham. Marginal contribution nets: A compact representation scheme for coalitional games. In *Proc. 6th ACM Conference on Electronic Commerce (EC-2005)*. ACM Press, 2005.

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

[Lafage and Lang, 2000] Celine Lafage and Jérôme 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.

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

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

[Pinkas, 1991] 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.

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

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

[Shafer, 1976] G. Shafer. *A mathematical theory of evidence*. Princeton University Press, 1976.