Weighted Propositional Formulas for Cardinal Preference Modelling

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

joint work with Jérôme Lang (IRIT, Toulouse) and Yann Chevaleyre (LAMSADE, Paris-Dauphine)
Main Question

What are appropriate languages for representing preferences in combinatorial domains? Can logic help?

Talk Overview

• Problem: Utility Functions in Combinatorial Domains
• Languages for Representing Utility Functions:
  – “Classical” Utility Functions
  – Weighted Propositional Formulas
• Expressive Power and Correspondence Results
• Comparative Succinctness
• Complexity Issues
• Conclusion and Future Work
Utility Functions in Combinatorial Domains

Let $X$ be a finite set. A *utility function* over the domain $X$ is a mapping from $X$ to the reals:

$$u : X \rightarrow \mathbb{R}$$

Simply listing the utilities for every element of $X$ is only feasible if $X$ is reasonably small.

This is *not* the case if $X$ has a combinatorial structure, as in resource allocation, multi-criteria decision making, elections of committees, . . .

- Resource allocation: set $\mathcal{R}$ of resources $\Rightarrow$ set $2^\mathcal{R}$ of bundles
- General: set $PS$ of propositional symbols $\Rightarrow$ set $2^{PS}$ of models

Fortunately, actual utility functions often exhibit some sort of *structure*, and a suitable preference representation language might be able to capture that structure in a *concise* manner.
Classes of Utility Functions

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

- $u$ is *normalised* iff $u(\{\}\} = 0$.
- $u$ is *non-negative* iff $u(X) \geq 0$.
- $u$ is *monotonic* iff $u(X) \leq u(Y)$ whenever $X \subseteq Y$.
- $u$ is *modular* iff $u(X \cup Y) = u(X) + u(Y) - u(X \cap Y)$.
- $u$ is *concave* iff $u(X \cup Y) - u(Y) \leq u(X \cup Z) - u(Z)$ for $Y \supseteq Z$.

Let $PS(k) = \{ S \subseteq PS \mid \# S \leq k \}$. $u$ is *$k$-additive* iff there exists another mapping $u' : PS(k) \rightarrow \mathbb{R}$ such that (for all $X$):

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

Also of interest: subadditive, superadditive, convex, ...
Why $k$-additive Functions?

The idea comes from fuzzy measure theory (Grabisch and others). Now also used in negotiation and combinatorial auctions.

Again, $u$ is $k$-additive iff there exists a $u' : PS(k) \to \mathbb{R}$ such that:

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

In the context of resource allocation, the value $u'(Y)$ can be seen as the additional benefit incurred from owning the items in $Y$ together, i.e. beyond the benefit of owning all proper subsets.

Example: $u = 4.p + 7.q - 2.p.q + 2.q.r$ is a 2-additive function

The $k$-additive form allows for a parametrisation of synergetic effects:

- $1$-additive $=$ modular (no synergies)
- $|PS|$-additive $=$ general (any kind of synergies)
- \ldots and everything in between
Weighted Propositional Formulas

An alternative approach to preference representation is based on weighted propositional formulas . . .

A goal base is a set $G = \{ (\varphi_i, \alpha_i) \}_i$ of pairs, each consisting of a consistent propositional formula $\varphi_i \in \mathcal{L}_{PS}$ and a real number $\alpha_i$. 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 $u_G$.

Example: $\{ (p \lor q \lor r, 5), (p \land q, 2) \}$

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?
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.

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 $\varphi$ with at most $k$ propositional symbols.

Regarding weights, we consider only the restriction to positive reals.

Given two restrictions $H$ and $H'$, let $\mathcal{U}(H, H')$ be the class of functions that can be generated from goal bases conforming to $H$ and $H'$. 

Ulle Endriss (ulle@illc.uva.nl)
Basic Results

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

**Proof:** Goals \((p_1 \land \cdots \land p_k, \alpha)\) directly correspond to the auxiliary utility function \(u' : \{p_1, \ldots, p_k\} \mapsto \alpha \ldots \)

**Proposition 2** The following are also all equal to the class of \( k\)-additive utility functions: \( \mathcal{U}(k\text{-cubes}, \text{all}), \mathcal{U}(k\text{-clauses}, \text{all}), \mathcal{U}(\text{positive } k\text{-formulas}, \text{all}) \) and \( \mathcal{U}(k\text{-formulas}, \text{all}) \).

**Proof:** Use equivalence-preserving transformations of goal bases such as \( G \cup \{(\varphi \land \neg \psi, \alpha)\} \equiv G \cup \{(\varphi, \alpha), (\varphi \land \psi, -\alpha)\} \)

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

**Proof:** \((\top, \alpha)\) cannot be rewritten as a positive clause \ldots
Monotonic Utility

Proposition 4 $U(\text{strictly positive, positive})$ is equal to the class of normalised monotonic utility functions.

Example: Take the normalised monotonic function $u$ with $u(\{p_1\}) = 2$, $u(\{p_2\}) = 5$ and $u(\{p_1, p_2\}) = 6$. We obtain the following goal base:

$$G = \{(p_1 \vee p_2, 2), (p_2, 3), (p_1 \land p_2, 1)\}$$
## Overview of Correspondence Results

<table>
<thead>
<tr>
<th>Formulas</th>
<th>Weights</th>
<th>Utility Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>cubes/clauses/all</td>
<td>general</td>
<td>all</td>
</tr>
<tr>
<td>positive cubes/formulas</td>
<td>general</td>
<td>all</td>
</tr>
<tr>
<td>positive clauses</td>
<td>general</td>
<td>normalised</td>
</tr>
<tr>
<td>strictly positive formulas</td>
<td>general</td>
<td>normalised</td>
</tr>
<tr>
<td>$k$-cubes/clauses/formulas</td>
<td>general</td>
<td>$k$-additive</td>
</tr>
<tr>
<td>positive $k$-cubes/formulas</td>
<td>general</td>
<td>$k$-additive</td>
</tr>
<tr>
<td>positive $k$-clauses</td>
<td>general</td>
<td>normalised $k$-additive</td>
</tr>
<tr>
<td>literals</td>
<td>general</td>
<td>modular</td>
</tr>
<tr>
<td>atoms</td>
<td>general</td>
<td>normalised modular</td>
</tr>
<tr>
<td>cubes/formulas</td>
<td>positive</td>
<td>non-negative</td>
</tr>
<tr>
<td>clauses</td>
<td>positive</td>
<td>$\subseteq$ non-negative</td>
</tr>
<tr>
<td>strictly positive formulas</td>
<td>positive</td>
<td>normalised monotonic</td>
</tr>
<tr>
<td>positive clauses</td>
<td>positive</td>
<td>$\subset^*$ normalised concave monotonic</td>
</tr>
</tbody>
</table>

*In the paper we only show $\subseteq$, but Joel Uckelman has recently found an example that shows that the two languages are in fact distinct.*
Comparative Succinctness

If two languages can express the same class of utility functions, which should we use? An important criterion is 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 $f : L \rightarrow L'$ and a polynomial function $p$ such that:

- $G \equiv f(G)$ for all $G \in L$ (they generate the same functions); and
- $\text{size}(f(G)) \leq p(\text{size}(G))$ for all $G \in L$ (polysize reduction).

Write $L \prec L'$ (strictly less succinct) iff $L \preceq L'$ but not $L' \preceq L$.

Two languages can also be incomparable with respect to succinctness.
An Incomparability Result

Let complete cubes $\subseteq L_{PS}$ be the restriction to cubes of length $n = |PS|$, containing either $p$ or $\neg p$ for every $p \in PS$.

Fact: $U$(complete cubes, all) is equal to the class of all utility functions (and corresponds to the “explicit form” of writing utility functions).

Proposition 5 $U$(complete cubes, all) and $U$(positive cubes, all) are incomparable (in view of their succinctness).

Proof: The following two functions can be used to prove the mutual lack of a polysize reduction:

- $u_1(M) = |M|$ can be generated by a goal base of just $n$ positive cubes of length 1, but we need $2^n - 1$ complete cubes for $u_1$.

- 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 $n$ complete cubes, but we require $2^n - 1$ positive cubes to generate $u_2$. □
The Efficiency of Negation

Recall that both $U(\text{positive cubes, all})$ and $U(\text{cubes, all})$ are equal to the class of all utility functions. So which should we use?

**Proposition 6** $U(\text{positive cubes, all}) \prec U(\text{cubes, all})$. [“less succinct”]

**Proof:** Clearly, $U(\text{positive cubes, all}) \preceq U(\text{cubes, all})$, because any positive cube is also a cube.

Now consider $u$ with $u(\{\}) = 1$ and $u(M) = 0$ for all $M \neq \{\}$:

- $G = \{(\neg p_1 \land \cdots \land \neg p_n, 1)\} \in U(\text{cubes, all})$ has linear size and generates $u$.

- $G' = \{((\land X, (-1)^{|X|}) \mid X \subseteq PS\} \in U(\text{positive cubes, all})$ has exponential size and also generates $u$.

On the other hand, the generator of $u$ must be unique if only positive cubes are allowed (start with $(\top, 1) \in G_u \ldots$). □
Recent Results

Some of these results apply to pairs of languages of different expressive power. In that case, the succinctness relation is understood to hold with respect to the class of utility functions in their intersection.

**Proposition 7** $U(\text{positive clauses, all}) \prec U(\text{clauses, all})$

**Proposition 8** $U(\text{complete cubes, all}) \prec U(\text{cubes, all})$

**Proposition 9** $U(\text{positive cubes, all})$ and $U(\text{positive clauses, all})$ are incomparable (in view of their succinctness).

**Proposition 10** $U(\text{positive cubes, all}) \prec U(\text{positive, all})$

**Proposition 11** $U(\text{cubes, all}) \prec U(\text{all, all})$


Complexity

Other interesting questions concern the complexity of reasoning about preferences. Consider the following decision problem:

\textbf{Max-Utility}(H, \ H')

\textbf{Given:} Goal base \( G \in \mathcal{U}(H, H') \) and \( K \in \mathbb{Z} \)

\textbf{Question:} Is there an \( M \in 2^{PS} \) such that \( u_G(M) \geq K \)?

Some basic results are straightforward:

- \textbf{Max-Utility}(H, H') is \textit{in \textsc{NP}} for any choice of \( H \) and \( H' \), because we can always check \( u_G(M) \geq K \) in polynomial time.

- \textbf{Max-Utility}(all, all) is \textsc{NP-complete} (reduction from \textsc{Sat}).

More interesting questions would be whether there are either

(1) “large” sublanguages for which \textbf{Max-Utility} is still polynomial,

or (2) “small” sublanguages for which it is already \textsc{NP-hard}.
Three Complexity Results

Proposition 12 \textsc{Max-Utility}($k$-clauses, positive) is \textit{NP-complete}, even for $k = 2$.

\textbf{Proof:} Reduction from \textsc{Max2Sat} (NP-complete): “Given a set of 2-clauses, is there a satisfiable subset with cardinality $\geq K$”? \hfill $\square$

Proposition 13 \textsc{Max-Utility}(literals, all) is in \textit{P}.

\textbf{Proof:} Assuming that $G$ contains every literal exactly once (possibly with weight 0), making $p$ true iff the weight of $p$ is greater than the weight of $\neg p$ results in a model with maximal utility. \hfill $\square$

Proposition 14 \textsc{Max-Utility}(positive, positive) is in \textit{P}.

\textbf{Proof:} Making all propositional symbols true yields maximal utility. \hfill $\square$

\begin{itemize}
  \item Joel has several more of these results. So far, it appears that \textsc{Max-Utility}(H, H’) is either NP-complete or linear (when in P).
\end{itemize}
Conclusion and Future Work

• Comparison of two ways of modelling utility functions, used in different communities (*expressive power/correspondence results*).

• If two languages are equally expressive, we need to use other criteria to decide which to use (simplicity versus *succinctness*).

• Another important property of a language for modelling utilities is the *computational complexity* of related reasoning tasks.

• This is ongoing work; we want to collect more results of this type to get a clearer picture of the general situation.

• Investigate other *aggregation functions* (than sum-taking) for weighted propositional formulas (such as *max*).

• Investigate connections to *bidding languages* for combinatorial auctions (e.g. XOR-language = *max* of positive cubes).