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:

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

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 $\mathcal{PS}$ of propositional symbols $\Rightarrow$ set $2^{\mathcal{PS}}$ 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 \( u' : PS(k) \rightarrow \mathbb{R} \) 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 $\varphi_i \in L_{PS}$ 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 $u_G$.

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 $H \subseteq L_{PS}$ 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 utility functions that can be generated from goal bases conforming to the restrictions $H$ and $H'$. 
Basic Results

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

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

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

\[
G = \{(p_1 \lor p_2, 2), (p_2, 3), (p_1 \land p_2, 1)\}
\]
## 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>$\subseteq$ normalised concave monotonic</td>
</tr>
</tbody>
</table>
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 $n$-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: $U(n$-cubes, all) is equal to the class of all utility functions (and corresponds to the “explicit form” of writing utility functions).

**Proposition 5** $U(n$-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 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 $u_2$. 

\[\square\]
The Efficiency of Negation

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

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

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

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

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

- $G' = \{(\land X, (-1)^{|X|}) \mid X \subseteq PS\} \in U(\text{positive cubes}, \text{all})$ 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 $(\top, 1) \in G_u \ldots$).
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 \(G \in U(H, H')\) 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 in \(NP\) 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 \(NP\)-complete (reduction from \(SAT\)).

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 \(NP\)-hard.
Three Complexity Results

Proposition 7 \textsc{Max-Utility}($k$-clauses, positive) is NP-complete, even for $k = 2$.

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

Proposition 8 \textsc{Max-Utility}(literals, all) is in P.

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. $\square$

Proposition 9 \textsc{Max-Utility}(positive, positive) is in P.

Proof: Making all propositional symbols true yields maximal utility. $\square$
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 do decide which to use (simplicity versus succinctness).

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

- The complexity results are still preliminary, but may lead somewhere interesting.

- 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).