Weighted Propositional Formulas for Preference Representation in Combinatorial Domains

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

[joint work with Yann Chevaleyre (Paris),
Jérôme Lang (Toulouse) and Joel Uckelman (Amsterdam)]

Talk Overview

We will take inspiration from the field of *collective decision making*.
In particular, I shall mention two applications:

- *multiagent resource allocation*
- *voting theory: electing a commitee*

We will concentrate on relevant *knoweldge representation* issues, particularly on languages for describing *utility functions* over *combinatorial domains* (needed to represent agent preferences):

- the *explicit form* of representation (not very clever);
- the *k-additive form* (a lot more attractive);
- logic-based languages based on *weighted formulas* and their properties: *expressivity*, *succinctness*, *complexity*

Multiagent Resource Allocation

Scenario: several agents and a set $\mathcal{R}$ of indivisible resources

Task: decide on an allocation of resources to agents, e.g. by means of negotiation or an auction; the quality of a solution can be measured in terms of a suitable aggregation of the individual preferences

Individual agents model their preferences in terms of *utility functions* $u : 2^{\mathcal{R}} \to \mathbb{R}$. In particular, the utility assigned to a bundle is *not* (necessarily) the sum of the utilities or the individual items.

▶ How should we *represent* the individual agent preferences?

Issues that matter for this kind of application:

- Can we *express* all the preference structures (utility functions) that we may come across?
- Can we express them in a *concise* manner?

Explicit Representation

The *explicit form* of representing a utility function u consists of a table listing for every bundle $X \subseteq \mathcal{R}$ the utility $u(X)$. By convention, table entries with $u(X) = 0$ may be omitted.

- the explicit form is *fully expressive:*
  any utility function $u : 2^{\mathcal{R}} \to \mathbb{R}$ may be so described
- the explicit form is *not concise:* it may require up to 2^n entries

Even very simple utility functions may require exponential space: e.g. the additive function mapping bundles to their cardinality.

The k-additive Form

- A utility function is *k-additive* iff the utility assigned to a bundle X can be represented as the sum of marginal utilities for subsets of X with cardinality $\leq k$ (*limited synergies*).

- The *k-additive form* of representing utility functions:

$$u(X) \quad = \quad \sum_{T \subseteq X} \alpha^T \qquad \text{with } \alpha^T = 0 \text{ whenever } |T| > k$$

  Example: $u = 3.x_1 + 7.x_2 - 2.x_2.x_3$ is a 2-additive function

- That is, specifying a utility function in this language means specifying the *coefficients α^T* for bundles $T \subseteq \mathcal{R}$.

- In the context of resource allocation, the value α^T can be seen as the additional benefit incurred from owning the items in T *together*, *i.e.* beyond the benefit of owning all proper subsets.

Expressive Power

The k-additive form is *fully expressive*, if we choose k large enough:

Proposition 1 *Any utility function is (uniquely) representable in k-additive form for some $k \leq |\mathcal{R}|$.*

Proof: For any utility function u, we can define coefficients α^X:

$$\alpha^{\{\,\}} = u(\{\,\})$$
$$\alpha^X = u(X) - \sum_{T \subset X} \alpha^T \quad \text{for all } X \subseteq \mathcal{R} \text{ with } X \neq \{\,\}$$

Hence, $u(X) = \sum_{T \subseteq X} \alpha^T$, which is k-additive for $k = |\mathcal{R}|$. ✓

The k-additive form allows for a *parametrisation* of synergies:

- 1-additive = modular (no synergies)
- $|\mathcal{R}|$-additive = general (any kind of synergies)
- ... and everything in between

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 languages for defining utilities. 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:

- $u \equiv f(u)$ for all $u \in L$ (they represent the same functions); and
- $size(f(u)) \leq p(size(u))$ for all $u \in L$ (polysize reduction).

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

Two languages can also be *incomparable* in view of succinctness.

Explicit vs. k-additive Form

Proposition 2 *The explicit and the k-additive form are incomparable in view of succinctness.*

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

- $u_1(X) = |X|$: representing u_1 requires $|\mathcal{R}|$ non-zero coefficients in the k-additive form (*linear*); but $2^{|\mathcal{R}|} - 1$ non-zero values in the explicit form (*exponential*).
- $u_2(X) = 1$ for $|X| = 1$ and $u_2(X) = 0$ otherwise: requires $|\mathcal{R}|$ non-zero values in the explicit form (*linear*); but $2^{|\mathcal{R}|} - 1$ non-zero coefficients in the k-additive form (*exponential*): $\alpha^T = 1$ for $|T| = 1$, $\alpha^T = -2$ for $|T| = 2$, $\alpha^T = 3$ for $|T| = 3$, ...

Remark: Still, for most utility functions occurring in practice, the k-additive form can be expected to be more succinct.

Committee Elections

How should we elect a committee with k seats from amongst n candidates? The usual approach is to extend standard voting rules:

- *Plurality voting:* each voter can vote for their preferred candidate and the candidate receiving the most votes wins
- *Approval voting:* each voter can approve of as many canddiates as they wish and the candidate receiving the most approvals wins

A naïve way of extending each would be to make the top k candidates winners. But neither method is very expressive:

- Plurality ballots can only express preferences where one candidate has utility 1 and the rest utility 0.
- Approval ballots can only express preferences where a subset of candidates each has utility 1 and each candidate in the complement has utility 0.

Example

Suppose we have a voter with the following preferences:

$$\text{Alice, Bob} \succ \text{neither} \succ \text{both}$$

What ballot should this voter cast under plurality (approval) voting?

Observe that these preferences would be expressible using either the *explict form* or the *k-additive form:*

$\{a\}$	1
$\{b\}$	1
$\{a, b\}$	-1

$$a + b - 3.a.b$$

▶ Besides having to express typical preferences in a *concise* way, we would also like to be able to do so in a *natural* manner . . .

Weighted Propositional Formulas

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

Let PS be a set of propositional symbols (resources, candidates) and let $\mathcal{L}_{PS}$ be the propositional language over PS.

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

Example: $\{(p \vee q \vee r, 5), (p \wedge q, 2)\}$

Question

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 φ 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'.

Basic Results

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

Proposition 4 *The following are also all equal to the class of* k*-additive utility functions:* $\mathcal{U}(k\textit{-cubes}, \textit{all})$*,* $\mathcal{U}(k\textit{-clauses}, \textit{all})$*,* $\mathcal{U}(\textit{positive } k\textit{-formulas}, \textit{all})$ *and* $\mathcal{U}(k\textit{-formulas}, \textit{all})$*.*

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

Normalised Utility Functions

A utility function $u : 2^{PS} \rightarrow \mathbb{R}$ is called *normalised* iff $u(\{\,\}) = 0$.

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

Proof: $(\top, \alpha)$ cannot be rewritten as a positive clause . . . ✓

Monotonic Utility

A utility function $u : 2^{PS} \to \mathbb{R}$ is called *monotonic* iff $u(X) \leq u(Y)$ whenever $X \subseteq Y$.

Proposition 6 *$\mathcal{U}(strictly\ positive, positive)$ 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:

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

Overview of Correspondence Results

Formulas	**Weights**		**Utility Functions**
cubes/clauses/all	general	$=$	all
positive cubes/formulas	general	$=$	all
positive clauses	general	$=$	normalised
strictly positive formulas	general	$=$	normalised
k-cubes/clauses/formulas	general	$=$	k-additive
positive k-cubes/formulas	general	$=$	k-additive
positive k-clauses	general	$=$	normalised k-additive
literals	general	$=$	modular
atoms	general	$=$	normalised modular
cubes/formulas	positive	$=$	non-negative
clauses	positive	$\subset$	non-negative
strictly positive formulas	positive	$=$	normalised monotonic
positive formulas	positive	$=$	non-negative monotonic
positive clauses	positive	$\subset$	normalised concave monotonic

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

An Incomparability Result

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

Fact: $\mathcal{U}(complete\ cubes, all)$ is equal to the class of *all* utility functions (and corresponds to the "explicit form" of writing utility functions).

Proposition 7 *$\mathcal{U}(complete\ cubes, all)$ and $\mathcal{U}(positive\ cubes, all)$ are incomparable in view of succinctness.*

Proof: This is in fact equivalent to the earlier result on the incomparability of the explicit and the k-additive form. ✓

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. So which should we use?

Proposition 8 *$\mathcal{U}(positive\ cubes, all) \prec \mathcal{U}(cubes, all)$.* ["less succinct"]

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

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

- $G = \{(\neg p_1 \wedge \cdots \wedge \neg p_n, 1)\} \in \mathcal{U}(cubes, all)$ has *linear* size and generates u.
- $G' = \{(\bigwedge X, (-1)^{|X|}) \mid X \subseteq PS\} \in \mathcal{U}(positive\ cubes, all)$ 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 $(\top, 1) \in G_u \ldots$). ✓

Cubes and Clauses

Proposition 9 *$\mathcal{U}(positive\ cubes, all)$ and $\mathcal{U}(positive\ clauses, all)$ are incomparable in view of succinctness (over normalised functions).*

Proof: Need to find counterexamples for both directions: one language can express it succinctly and the other not. Need to appeal to uniqueness property for the latter (non-trivial for positive clauses). ✓

Proposition 10 $\mathcal{U}(cubes, all) \sim \mathcal{U}(clauses, all)$ ["equally succinct"]

Proof: Use equivalence-preserving transformastions of goal bases such as $G \cup \{(\varphi \vee \psi, \alpha)\} \equiv G \cup \{(\neg\varphi \wedge \neg\psi, -\alpha), (\top, \alpha)\}$. Given that weights labelling the same formula (here $\top$) can be combined, this increases the cardinality of the goal base by at most 1. ✓

Complexity

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

> MAX-UTILITY(H, H')
> **Given:** Goal base $G \in \mathcal{U}(H, H')$ and $K \in \mathbb{Z}$
> **Question:** Is there an $M \in 2^{PS}$ such that $u_G(M) \geq K$?

Some basic results are straightforward:

- 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.
- MAX-UTILITY(*all*, *all*) is *NP-complete* (reduction from SAT).

More interesting questions would be whether there are either (1) "large" sublanguages for which MAX-UTILITY is still polynomial, or (2) "small" sublanguages for which it is already NP-hard.

Three Complexity Results

Proposition 11 $\textsc{Max-Utility}(\mathit{k\text{-}clauses}, \mathit{positive})$ *is NP-complete, even for* $k = 2$.

<u>Proof:</u> Reduction from $\textsc{Max2Sat}$ (NP-complete): "Given a set of 2-clauses, is there a satisfiable subset with cardinality $\geq K$?". ✓

Proposition 12 $\textsc{Max-Utility}(\mathit{literals}, \mathit{all})$ *is in P.*

<u>Proof:</u> 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. ✓

Proposition 13 $\textsc{Max-Utility}(\mathit{positive}, \mathit{positive})$ *is in P.*

<u>Proof:</u> Making *all* propositional symbols true yields maximal utility. ✓

Back to Voting

Some very simple languages correspond to the sets of legal ballots for two well-known voting rules (to elect a single candidate):

- *Plurality voting:* vote for your preferred candidate (the candidate receiving the most votes wins): $\mathcal{U}(atom, \{1\})$
- *Approval voting:* approve of as many canddiates as you wish (the candidate receiving the most approvals wins): $\mathcal{U}(atoms, \{1\})$

Propositional logic seem a suitable language for expressing voter preferences over commitees, so maybe this could be extended.

Winner determination could be modelled as MAX-UTILITY wrt. the sum of the goal bases sent by each voter, and a goal base encoding the constraints on the size of the committee (with very high weights).

Conclusion

Compact preference representation in combinatorial domains is relevant to a number of applications, and weighted goals are an interesting class of languages for doing this. Ongoing work:

- Fill in missing technical results on expressivity, succincness and complexity to get global picture
- Aggregation operators other than $\sum$ (particularly $\max$)
- Applications: committee elections, distributed negotiation, combinatorial auctions

Y. Chevaleyre, U. Endriss, and J. Lang. *Expressive Power of Weighted Propositional Formulas for Cardinal Preference Modelling*. Proc. KR-2006.

J. Uckelman and U. Endriss. *Preference Representation with Weighted Goals: Expressivity, Succinctness, Complexity*. Proc. AiPref-2007.