# Computational Social Choice: Spring 2007

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

# Preference Representation

Collective decision making is driven by the interests of individuals, who must be able to *communicate preferences* (directly through full revelation, or indirectly via "moves" in a game).

- So far, we have treated this topic only very *abstractly*, by saying that agents "have" a utility function or "report" a ranking.
- Preferences representation in *combinatorial domains:*
  - electing a committee of size $k$ from amongst $n$ candidates requires expressing preferences over $\binom{n}{k}$ possible committees;
  - negotiation over $n$ goods requires expressing preferences over $2^n$ alternative bundles.
- In this lecture, we are going to review and compare different *preference representation languages*.

# Plan for Today

- General *requirements* on preference representation languages
- Distinguish *cardinal* and *ordinal* preference structures
- Different *classes* of utility functions (cardinal preferences): monotonic, dichotomous, modular, concave utilities ...
- Review of languages for representing utility functions: *explicit form*, *k-additive form*, *weighted goals*, ...
- Discussion of properties of different representation languages: *expressive power* and *comparative succinctness*
- Review of languages for ordinal preference representation: *prioritised goals* and *ceteris paribus preferences*

# Preference Representation Languages

The following questions should be addressed when you investigate a preference representation language:

- *Cognitive relevance:* How close is a given language to the way in which humans would express their preferences?
- *Elicitation:* How difficult is it to elicit the preferences of an agent so as to represent them in the chosen language?
- *Expressive power:* Can the chosen language encode all the preference structures we are interested in?
- *Succinctness:* Is the representation of (typical) structures succinct? Is one language more succinct than the other?
- *Complexity:* What is the computational complexity of related decision problems, such as comparing two alternatives?

We are going to concentrate on expressive power and succinctness.

## Cardinal and Ordinal Preferences

A *preference structure* represents an agent's preferences over a set of alternatives $\mathcal{X}$. There are different types of preference structures:

- A *cardinal* preference structure is a (*utility* or *valuation*) function $u : \mathcal{X} \rightarrow Val$, where $Val$ is usually a set of numerical values such as $\mathbb{N}$ or $\mathbb{R}$.
- An *ordinal* preference structure is a *binary relation* $\preceq$ over the set of alternatives (reflexive, transitive and connected).

Note that we shall assume that $\mathcal{X}$ is finite.

# Some Observations

- *Intrapersonal comparison:* ordinal and cardinal preferences allow for comparing the satisfaction of an agent for different alternatives
- *Interpersonal comparison:* ordinal preferences don't allow for interpersonal comparison ("Ann likes $x$ more than Bob likes $y$")
- *Preference intensity:* ordinal preferences cannot express preference intensity; cardinal preferences can (subject to *Val* being numerical)
- *Representability:* a connected ordinal preference relation $\preceq$ is representable by a utility function $u$: $x \preceq y$ iff $u(x) \leq u(y)$
- *Cognitive relevance:* hard to make general statements, but at least ordinal preferences don't require reasoning with numerical utilities
- *Explicit representation:* the explicit representation of cardinal and ordinal preferences have space complexity $O(|\mathcal{X}|)$ resp. $O(|\mathcal{X}|^2)$

## Preferences in Resource Allocation Scenarios

Let $\mathcal{R}$ be a finite set of indivisible *resources* (goods) with $|\mathcal{R}| = n$.

Assume there are *no externalities:* agent preferences only depend on their assigned bundle (not on, say, the allocation as a whole) $\leadsto$ need to model preference structures over $\mathcal{X} = 2^{\mathcal{R}}$

Hence, the explicit representation has *exponential* space complexity.

Possible ways out:

- only consider *restricted classes* of preference structures, which may allow for a more concise representation; and/or
- consider (and compare) *different representation languages*.

We start with the case of utility functions ...

# Classes of Utility Functions

Now a utility function is a mapping $u : 2^{\mathcal{R}} \to \mathbb{R}$.

- $u$ 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 *dichotomous* iff $u(X) = 0$ or $u(X) = 1$
- $u$ is *modular* iff $u(X \cup Y) = u(X) + u(Y) - u(X \cap Y)$
- $u$ is *additive* iff $u(X) = \sum_{x \in X} u(\{x\})$

Important: For the above definitions, the respective (in)equalities are understood to hold for *all* bundles $X, Y \subseteq \mathcal{R}$.

▶ What is the connection between modular and additive utilities?

# Modular and Additive Utilities

Modularity and additivity are really just two different names for the same thing (well, almost):

**Proposition 1** *A utility function is additive iff it is both modular and normalised.*

Proof: "$\Rightarrow$": obvious ✓

"$\Leftarrow$": Let $X \subseteq \mathcal{R}$, $x \in X$.
From modularity, we get $u(X) = u(X \setminus \{x\}) + u(\{x\}) - u(\{\,\})$.
As $u$ is normalised, we obtain $u(X) = u(X \setminus \{x\}) + u(\{x\})$.
If we iterate this step $|X|$ times, we get $u(X) = \sum_{x \in X} u(\{x\})$. ✓

## More Classes of Utility Functions

A few more commonly used classes of utility functions:

- $u$ is *submodular* iff $u(X \cup Y) \leq u(X) + u(Y) - u(X \cap Y)$
- $u$ is *supermodular* iff $u(X \cup Y) \geq u(X) + u(Y) - u(X \cap Y)$
- $u$ is *concave* iff $u(X \cup Y) - u(Y) \leq u(X \cup Z) - u(Z)$ for $Y \supseteq Z$
  - Intuition: marginal utility (of obtaining $X$) decreases as we move to a better starting position (namely from $Z$ to $Y$)
- $u$ is *convex* iff $u(X \cup Y) - u(Y) \geq u(X \cup Z) - u(Z)$ for $Y \supseteq Z$

## Observations

The following relationships amongst some of these classes of utility functions are easily checked:

- submodular $\cap$ supermodular $=$ modular
- $u$ submodular iff $-u$ supermodular
- $u$ concave iff $-u$ convex
- concave $\subseteq$ submodular (Proof: set $Z = X \cap Y$)
- convex $\subseteq$ supermodular

# 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 $u : 2^{\mathcal{R}} \rightarrow \mathbb{R}$ 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.

Remark: Of course, any additive utility function *could* be encoded very concisely: just store the utilities for individual goods + the information that this is an additive function $\rightsquigarrow$ linear space
But this is *not* a *general method* (not fully expressive).

# 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 $\alpha^T$* for bundles $T \subseteq \mathcal{R}$.

- In the context of resource allocation, the value $\alpha^T$ 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 2** *Any utility function is representable in $k$-additive form for some $k \leq |\mathcal{R}|$.*

Proof: For any utility function $u$, we can define coefficients $\alpha^X$:

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

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

Proof sketch: 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 $|T| = 3$, ...

Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. *Multiagent Resource Allocation with k-add. Utility Functions*. DIMACS-LAMSADE Workshop 2004.

# Weighted Propositional Formulas

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

Notation: finite set of propositional letters $PS$ (representing goods); propositional language $\mathcal{L}_{PS}$ over $PS$ can describe requirements.

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

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

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

▶ If we restrict goals to *conjunctions of atoms* (of length $\leq k$), then this corresponds directly to the *$k$-additive form*.

# Program-based Representations

Yet another approach to representing preferences would be to define utilities in terms of a *program:* input bundle, output utility value. But not just any program will do. Requirements:

- it must be possible to efficiently validate that a given string constitutes a *syntactically correct program*; and
- we have to have an effective method of *computing the output* of the program for any given input.

Dunne *et al.* (2005) propose such a program-based approach based on so-called *straight-line programs* (warning: rather technical).

One result says that any function computable by a deterministic TM in time $T$ is representable by an SLP with $O(T \log T)$ lines.

P.E. Dunne, M. Wooldridge, and M. Laurence. The Complexity of Contract Negotiation. *Artificial Intelligence*, 164(1–2):23–46, 2005.

## Ordinal Preferences

Next we are going to look into different languages for representing *ordinal* preference structures.

Recall that an *explicit representation* of an ordinal preference relation $\preceq$ over $2^n$ alternatives requires space up to $O(2^n \cdot 2^n)$: for each pair of bundles, say which one is preferred.

# Prioritised Goals

Again, associate goods with propositional letters in $PS$ and bundles with models $M \in 2^{PS}$. *Goals* can be expressed as formulas in the propositional language $\mathcal{L}_{PS}$.

Instead of weights, we now have a *priority relation* over goals. Assuming this priority relation is a total order, it can be represented by a function $rank : \mathbb{N} \to \mathbb{N}$ mapping each (index of a) goal to its rank. By convention, a *lower rank* means *higher priority*.

A *goal base* is now a finite set of goals with an associated rank function: $G = \langle\{\varphi_1, \ldots, \varphi_m\}, rank\rangle$.

▶ Ideally, all goals will get satisfied. But if not, how can we extend a priority relation over goals to a preference relation over models?

## Combining Priorities

There are several options (convention: $\min(\{\,\}) = +\infty$):

- *Best-out ordering:*

  $M \preceq M'$ iff $\min\{\mathit{rank}(i) \mid M \not\models \varphi_i\} \leq \min\{\mathit{rank}(i) \mid M' \not\models \varphi_i\}$

  That is, preference depends (only) on the rank of the most important goal that is being violated.

- *Discrimin ordering:*
  Let $d(M, M') = \min\{\mathit{rank}(i) \mid M \not\models \varphi_i \text{ and } M' \models \varphi_i\}$ be the rank of the most important "discriminating" goal.

$$M \preceq M' \text{ iff } d(M, M') \leq d(M', M) \text{ or}$$
$$\{\varphi_i \mid M \models \varphi_i\} = \{\varphi_i \mid M' \models \varphi_i\}$$

## Combining Priorities (cont.)

- *Leximin ordering:*
  Let $d_k(M) = |\{\varphi_i \mid M \models \varphi_i \text{ and } \mathit{rank}(\varphi_i) = k\}|$ be the number of goals of rank $k$ that are satisfied by alternative $M$.

$M \preceq M'$ iff (1) for all $k$: $d_k(M) = d_k(M')$ or

(2) there exists a $k$ such that $d_k(M) < d_k(M')$

and for all $j < k$: $d_j(M) = d_j(M')$

## Properties

- None of the three variants of combining prioritised goals leads to a *fully expressive* preference representation language.
- The *best-out ordering* and the *leximin ordering* result in *connected* preference relations, but the *discrimin ordering* typically does not.
- For the *strict* preference relations we have:
  - *best-out preference* entails *discrimin preference*; and
  - *discrimin preference* entails *leximin preference*

# Ceteris Paribus Preferences

In the language of *ceteris paribus* preferences, preferences are expressed as statements of the form $C : \varphi > \varphi'$, meaning:

> *"If $C$ is true, all other things being equal, I prefer alternatives satisfying $\varphi \wedge \neg\varphi'$ over those satisf. $\neg\varphi \wedge \varphi'$."*

The "other things" are the truth values of the propositional variables not occurring in $\varphi$ and $\varphi'$. A preference relation can be constructed as the transitive closure of the union of individual preference statements.

Discussion: interesting from a *cognitive* point of view (close to human intuition), but of rather *high complexity*.

An important sublanguage of *ceteris paribus* preferences, imposing various restrictions on goals, are *CP-nets*.

C. Boutilier *et al*. CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. *JAIR*, 21:135–191, 2004.

# Summary

We have reviewed several preference representation languages for both cardinal and ordinal preference structures.

- The computational aspects of preference representation are crucial in *combinatorial domains* (such as resource allocation).
- We have emphasised *expressive power* and *succinctness*.
- Languages considered (there are many more):
  - *cardinal:* explicit form, $k$-additive form, weighted goals, and program-based representations of utility functions
  - *ordinal:* prioritised goals and ceteris paribus statements

# References

For an in-depth survey of logic-based languages for representing preferences, refer to:


- J. Lang. Logical Preference Representation and Combinatorial Vote. *Annals of Mathematics and Artificial Intelligence*, 42(1):37–71, 2004.


For a concise overview of the role of preference representation in the context of multiagent resource allocation, consult:


- Y. Chevaleyre *et al.* Issues in Multiagent Resource Allocation. *Informatica*, 30:3–31, 2006. (Sect. Preference Representation)