Introduction to Logic in Computer Science: Autumn 2007

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

Preference Modelling

- An important topic in *knowledge representation* is the study of languages for expressing *preferences*.
- There are many criteria that we may apply to decide what is a good preference representation language and what isn't.
- This will be an introduction to preference representation when the set of alternatives over which an agent has preferences has a combinatorial structure (i.e. there are many alternatives).

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.

Dinner Plans

Consider the following menu options:

- Starter: fish soup, vegetable soup or salad
- Main: meat or fish
- Wine: red or white
- Dessert: ice cream or tiramisu

So there are 24 possible menus. We don't really want to rank all of them before making a decision.

But we can also not completely decompose the problem into 4 separate problems either (wine choice may depend on mains, etc.).

Committee Elections

Suppose we have to elect a *committee* (not just a single candidate).

If there are k *seats* to be filled from a pool of n *candidates*, then there are $\binom{n}{k}$ possible outcomes.

For $k = 5$ and $n = 12$, for instance, that makes 792 alternatives.

The domain of alternatives has a *combinatorial structure*.

It does not seem reasonable to ask voters to submit their full preferences over all alternatives to the collective decision making mechanism. What would be a reasonable form of balloting?

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 could be measured in terms of some aggregation of individual preferences
For m agents and n resources, there are m^n allocations to consider.

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.

For each agent, there are 2^n alternative bundles to consider.

How should we represent the individual agent preferences?

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.

<u>Remark:</u> 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 $\leadsto$ linear space
But this is *not* a *general method* (not fully expressive).

Explicit Representation (cont.)

For ordinal preferences the situation is even worse. The space complexity required to explicitly describe an ordinal preference ordering over $\mathcal{X}$ is $O(|\mathcal{X}|^2)$. For $\mathcal{X} = 2^{\mathcal{R}}$ this is bad.

$\leadsto$ We need to use something a bit more sophisticated!

Two Frameworks

In the remainder of this lecture we are going to look at two specific frameworks for compact preference representation:

- *CP-nets* for modelling conditional (ordinal) preferences in a *ceteris paribus* fashion
- *Weighted propositional formulas* for modelling utility functions

CP-Nets

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

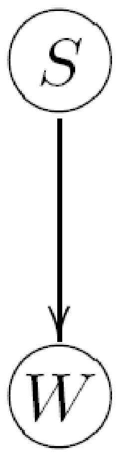
> *"If C is true, all other things being equal, I prefer alternatives satisfying $\varphi \wedge \neg\varphi'$ over those satisf. $\neg\varphi \wedge \varphi'$."*

The "other things" are the truth values of the propositional variables not occurring in φ and φ'.

An important sublanguage of *ceteris paribus* preferences, imposing various restrictions on goals, are *CP-nets*. This part of the lecture is based on the paper by Boutilier et al. (2004). In particular, all the pictures are taken from that paper.

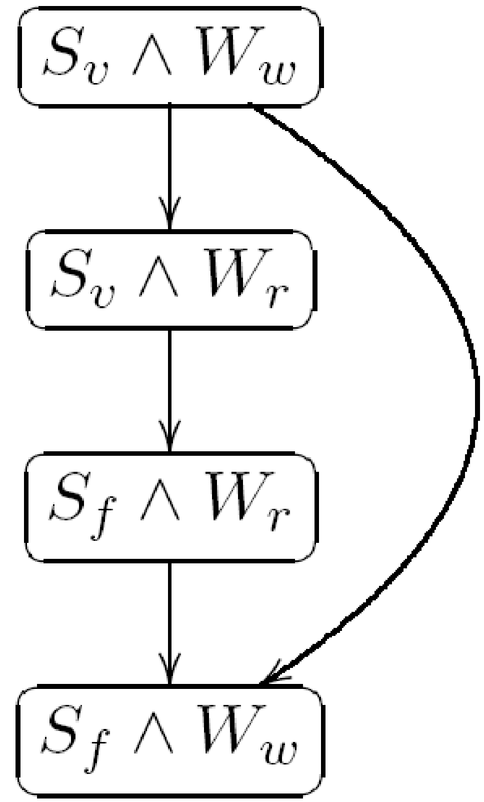
C. Boutilier, R.I. Brafman, C. Domshlak, H.H. Hoos, and D. Poole. CP-nets: A Tool for Representing and Reasoning with Conditional *Ceteris Paribus* Preference Statements. *Journal of AI Research*, 21:135–191, 2004.

Example: Dinner



$$\boxed{S_f \succ S_v}$$

S_f	$W_w \succ W_r$
S_v	$W_r \succ W_w$

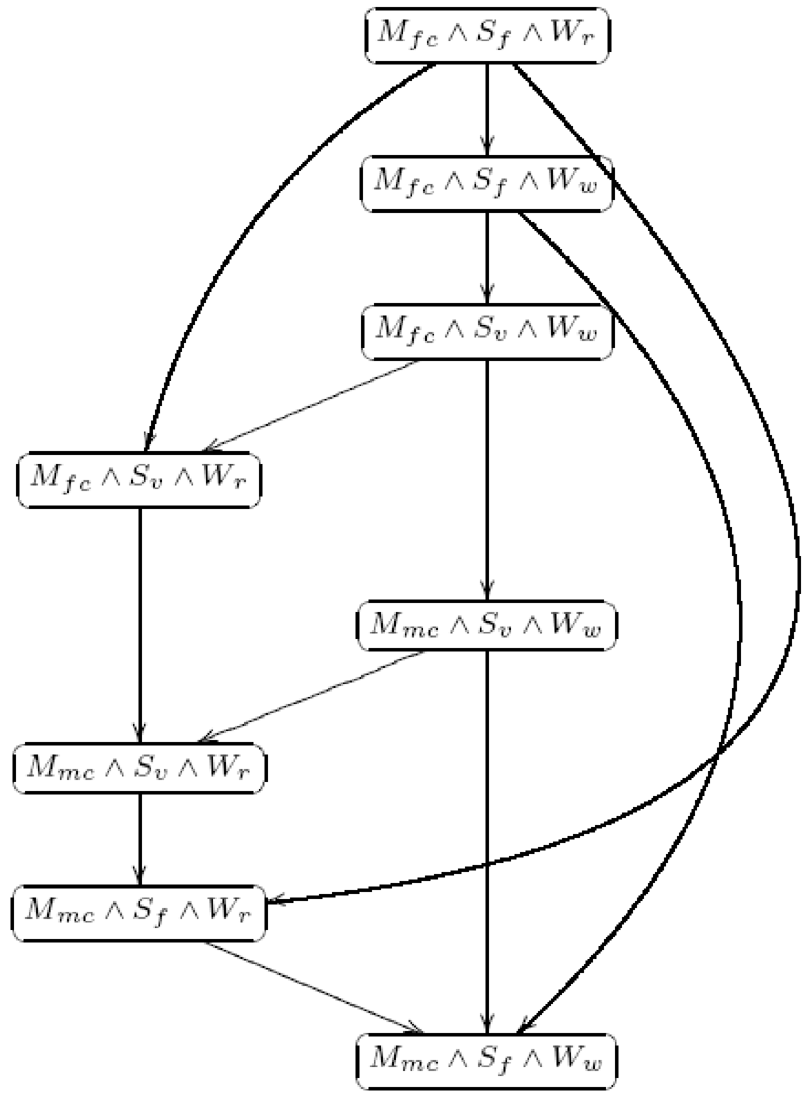


Example: Dinner II

$M_{mc} \succ M_{fc}$

M_{mc}	$S_f \succ S_v$
M_{fc}	$S_v \succ S_f$

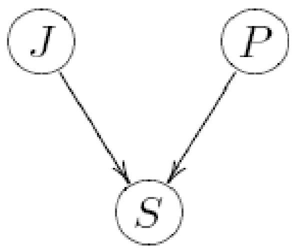
S_f	$W_w \succ W_r$
S_v	$W_r \succ W_w$



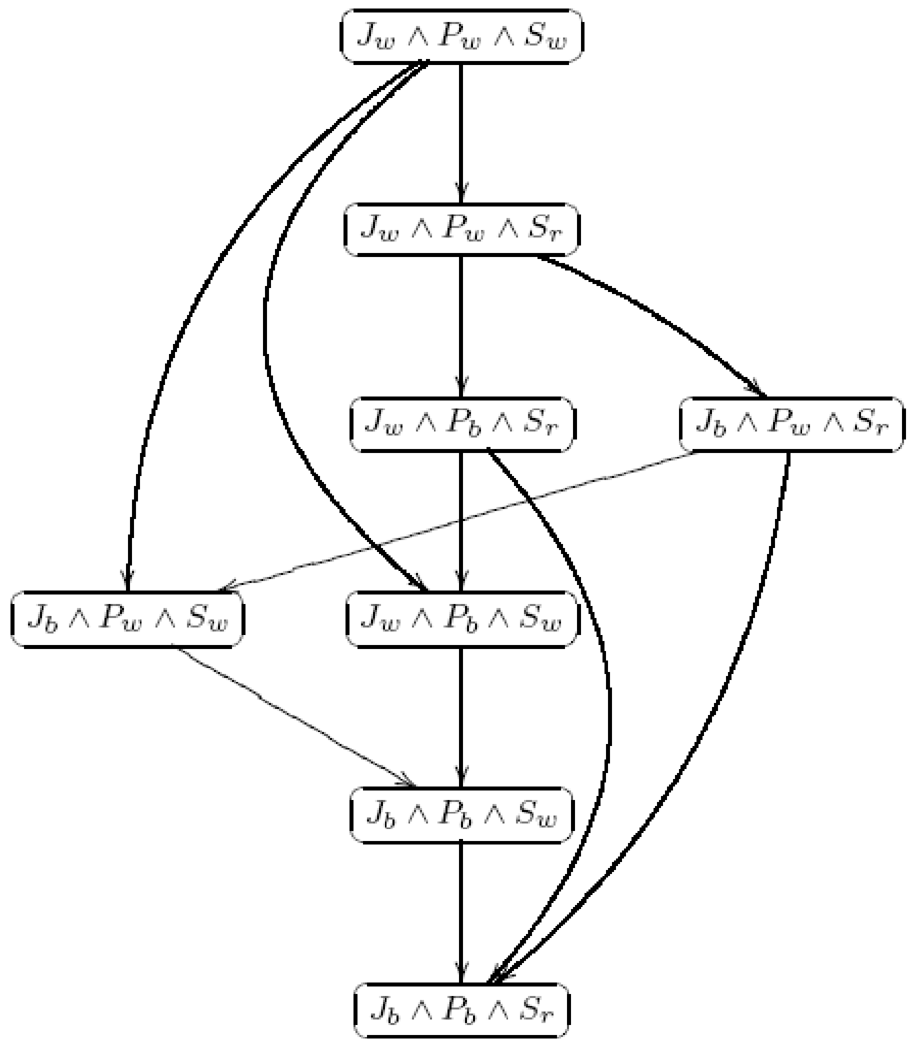
Example: Evening Dress

$J_b \succ J_w$

$P_b \succ P_w$



$J_b \wedge P_b$	$S_r \succ S_w$
$J_w \wedge P_b$	$S_w \succ S_r$
$J_b \wedge P_w$	$S_w \succ S_r$
$J_w \wedge P_w$	$S_r \succ S_w$



Definition

A CP-net over variables $V = \{X_1, \ldots, X_n\}$ is a directed graph G over V whose nodes are annotated with *conditional preference tables* for each X_i. Each such table (for X_i) associates a total order with each instantiation of the parents of X_i in the graph.

A given preference ordering $\succ$ may or may not *satisfy* a given CP-net (semantics as expected).

To date, most technical results pertain to *acyclic* CP-nets. E.g.:

Proposition 1 *Every acyclic CP-net is satisfiable.*

Some Complexity Results

The following results apply to acyclic CP-nets:

- *Outcome optimisation:* What is the best alternative?
  $O(n)$ — easy algorithm: start from most important variables and set each variable to its most preferred value

- *Dominance queries:* Does the CP-net N force $N \models o \succ o'$?
  NP-hard in general (upper bound not known), but tractable for special cases, e.g. $O(n^2)$ for binary-valued tree-structured nets

- *Ordering queries:* Is $o \succ o'$ consistent with N, i.e. $N \not\models o' \succ o$?
  $O(n)$ to check whether $N \not\models o' \succ o$ or $N \not\models o \succ o'$

Weighted Propositional Formulas

Next we are going to look at a language for modelling utility functions. The basic idea is to use propositional logic to express *goals* and to add up the *weights* of the goals satisfied for a particular alternative.

The results on the following slides are taken from the two papers cited below.

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.

Classes of 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 $u(X) \geq 0$.
- 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 $u' : PS(k) \to \mathbb{R}$ such that (for all X):

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

Also of interest: subadditive, superadditive, convex, ...

Why k-additive Functions?

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

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

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.

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

The *k-additive form* allows us to parametrise synergetic effects:

- 1-additive = modular (no synergies)
- $|PS|$-additive = general (any kind of synergies)
- … and everything in between

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

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

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

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

Proposition 3 *The following are also all equal to the class of k-additive utility functions: $\mathcal{U}$(k-cubes, all), $\mathcal{U}$(k-clauses, all), $\mathcal{U}$(positive k-formulas, all) and $\mathcal{U}$(k-formulas, all).*

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

Proposition 4 *$\mathcal{U}$(positive k-clauses, all) is equal to the class of normalised k-additive utility functions.*

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

Monotonic Utility

Proposition 5 *$\mathcal{U}$(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:

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

Some Expressivity 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

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 (classes of goal bases).

L is no more succinct than L' ($L \preceq L'$) iff there exist a mapping $f : L \to L'$ and a *polynomial* function p such that:

- $u_G \equiv u_{f(G)}$ 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).

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

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

Proposition 6 *$\mathcal{U}(\textit{complete cubes}, \textit{all})$ and $\mathcal{U}(\textit{positive cubes}, \textit{all})$ 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. $\square$

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. So which should we use?

Proposition 7 $\mathcal{U}(\textit{positive cubes}, \textit{all}) \prec \mathcal{U}(\textit{cubes}, \textit{all})$.

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

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

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

Some Succinctness Results

$$
\begin{array}{rcl}
\mathcal{L}(pcubes, all) & \perp & \mathcal{L}(complete\ cubes, all) \\
\mathcal{L}(pcubes, all) & \prec & \mathcal{L}(cubes, all) \\
\mathcal{L}(pcubes, all) & \prec & \mathcal{L}(positive, all) \\
\mathcal{L}(pclauses, all) & \prec & \mathcal{L}(clauses, all) \\
\mathcal{L}(pcubes, all) & \perp & \mathcal{L}(pclauses, all) \\
\mathcal{L}(cubes, all) & \sim & \mathcal{L}(clauses, all)
\end{array}
$$

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 8 Max-Utility(*k-clauses*, *positive*) *is NP-complete, even for* $k = 2$.

Proof: Reduction from Max2Sat (NP-complete): "Given a set of 2-clauses, is there a satisfiable subset with cardinality $\geq K$?". □

Proposition 9 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. □

Proposition 10 Max-Utility(*positive*, *positive*) *is in P.*

Proof: Making *all* atoms true yields maximal utility. □

Some Complexity Results

- $\textsc{Max-Utility}(\textit{literals}, \textit{all})$ is in P.
- $\textsc{Max-Utility}(\textit{positive}, \textit{positive})$ is in P.
- $\textsc{Max-Utility}(k\textit{-clauses}, \textit{positive})$ is NP-complete for $k \geq 2$.
- $\textsc{Max-Utility}(k\textit{-cubes}, \textit{positive})$ is NP-complete for $k \geq 2$.
- $\textsc{Max-Utility}(\textit{positive } k\textit{-clauses}, \textit{all})$ is NP-complete for $k \geq 2$.
- $\textsc{Max-Utility}(\textit{positive } k\textit{-cubes}, \textit{all})$ is NP-complete for $k \geq 2$.

Conclusion

Preference representation in combinatorial domains is relevant to a number of applications.

CP-nets and weighted propositional formulas are two proposals for compact preference representation in this area.

Interesting questions to consider:

- How expressive is the language under consideration?
- Which language is more succinct for certain structures?
- What is the complexity of relevant decision problems?
- How do I best elicit preferences from a user?
- What features make a language "natural"?