Proceedings, Eleventh International Conference on Principles of Knowledge Representation and Reasoning (2008)

Preference Modeling by Weighted Goals with Max Aggregation

Joel Uckelman
ILLC, University of Amsterdam
Plantage Muidergracht 24
1018TV Amsterdam, Netherlands

Ulle Endriss
ILLC, University of Amsterdam
Plantage Muidergracht 24
1018TV Amsterdam, Netherlands

Abstract

Logic-based preference representation languages are promising for expressing preferences over combinatorial domains. Sets of weighted formulas, called goalbases, can be used to define several such languages. How goalbases are translated into utility functions—that is, by what aggregation function this is done—is a crucial component of this type of language. In this paper, we consider the properties of several goalbase languages which use $\max$ as their aggregation function. In particular, we examine the expressivity, succinctness and complexity of such languages.

Introduction

Combinatorial domains present a challenge for preference representation. A feature which makes combinatorial auctions, multiwinner voting, multiagent resource allocation, and recommender systems attractive—their ability to handle synergies among items—is the same feature which makes representing the preferences of agents taking part in them difficult. Explicit representation of agent preferences over states generated by subsets of the available items is impractical for all but the smallest sets of items. Real agents (people) will balk at being asked to rank all the subsets of five items; even a computer will become overwhelmed when faced with keeping an explicit representation for as few as 32 items.[1]

We can overcome this difficulty by using implicit representations of agent preferences, which can often be given more succinctly than an explicit representation can. Implicit representations achieve this by taking advantage of the structure of the preferences being represented. Examples are CP-nets (Boutilier *et al.* 2004) for ordinal preferences, GAI-nets for utility functions which are decomposable in certain ways (Gonzales & Perny 2004), and a host of methods for dealing with voting on interdependent choices (Lang 2004).

Using weighted formulas for preference representation is an idea which originated in penalty logic (Pinkas 1995) and has been applied in voting theory (Lang 2004). In this paper, we focus on languages which use sets of weighted propositional formulas to represent utility functions, following (Chevaleyre, Endriss, & Lang 2006). This framework offers a rich array of languages, far more than are explored there (or here). Chevaleyre, Endriss, & Lang (2006) focus on one particular class of languages, namely those which aggregate their goalbases by summing the weights of satisfied goals. Here, we turn to languages which instead aggregate by taking the maximum-valued satisfied goal. As we shall see, these *max languages* have quite different properties from the sum languages treated in (Chevaleyre, Endriss, & Lang 2006) and (Uckelman & Endriss 2007).

Interesting properties of goalbase languages are their expressivity, succinctness, and complexity—these tell you what utility functions are available in a language, how efficiently they can be represented, and how difficult you can expect computational tasks using the goalbases as input to be. After presenting our framework and discussing related work, we present one section of results for max languages for each of expressivity, succinctness, and complexity; and finally offer some concluding remarks. Please refer to the appendix for proofs of the propositions.

Preference Representation Languages

In this section we define the languages for representing utility functions that are the object of study in this paper and introduce some basic notation. We also discuss related approaches to modeling utility functions and state some simple facts about our languages.

Basic definitions and notation

Fix a finite set $\mathcal{PS}$ of propositional variables. These variables may, for instance, represent goods that the agent whose preferences we wish to model may or may not own: $p \in \mathcal{PS}$ is true if the agent possesses the corresponding good, and false otherwise (but other interpretations are possible as well). Let $\mathcal{L}_{\mathcal{PS}}$ denote the language of propositional logic over $\mathcal{PS}$, using negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$). This language can be used to express goals (formulas the agent would like to hold). A *model* for a formula in this language is a set $M \in 2^{\mathcal{PS}}$, containing exactly those propositional variables that are true in the model. We are interested in modeling *utility functions* $u : 2^{\mathcal{PS}} \to \mathbb{R}$ which map models (which we are also going to refer to as *states* or *alternatives*) to the reals. Utility functions will be represented using *goalbases*.

[1] Storing one four-byte integer for each of 2^{32} states requires 16GB of storage, which is at least four times the amount of RAM usually found in new desktop computers as of the date of writing.

Definition 1 (Weighted goals and goalbases)**.** *A* weighted goal *is a pair* (φ, w)*, where* φ *is a satisfiable formula in the language* $\mathcal{L}_{\mathcal{PS}}$ *and* $w \in \mathbb{R}$*. A* goalbase *is a finite set* $G = \{(\varphi_i, w_i)\}_i$ *of weighted goals.*

Goalbases can be interpreted in a variety of ways. For instance, if the weights are taken to represent a ranking of the importance of goals, then this information can be used to define an ordinal preference relation (see e.g., Coste-Marquis *et al.*, 2004). In this paper we are only interested in cardinal preferences (utility functions); still, there are several ways of interpreting a given goalbase (Lang 2004). First, we may either derive utility from goals that are satisfied or incur penalties for goals that are violated by an alternative. Here we shall opt for the former: If $(\varphi, w) \in G$ and an alternative satisfies φ, then this goal yields utility w. Second, we have to decide how to aggregate the weights if more than one goal is satisfied. In principle, any aggregation function $F : 2^{\mathbb{R}} \to \mathbb{R}$ could be used. Lafage and Lang (2000) discuss the basic properties (e.g., commutativity) that such an aggregation function should satisfy.

Definition 2 (Goal bases and utility functions)**.** *A goalbase* G *and an aggregation function* $F : 2^{\mathbb{R}} \to \mathbb{R}$ *generate a utility function* $u_{G,F}$ *by mapping each model* $M \in 2^{\mathcal{PS}}$ *to* $u_{G,F}(M) = F(\{w : (\varphi, w) \in G \text{ and } M \models \varphi\})$*.*

Natural choices for F include summation and $\max$. Given the same goalbase, these two operators produce dramatically different utility functions. E.g., if $G = \{(a, 1) : a \in \mathcal{PS}\}$, then $u_{G,\max}$ is the simple unit-demand utility function ($u(X) = 1$ if $X \neq \emptyset$, 0 otherwise) while $u_{G,\Sigma}$ is the simple additive utility function ($u(X) = |X|$). In case $\max$ is used, we assume $\max \emptyset = -\infty$, i.e., it is often useful to include, say, $(\top, 0)$ in any goalbase by default, so as to obtain well-defined utility functions for all states. We sometimes write u_G rather than $u_{G,F}$ if F is clear from the context. We write $G \equiv_F G'$ if goalbases G and G' define the same utility function when F is used for aggregation.

Weighted propositional formulas provide a framework for defining entire *families* of languages for representing preferences (Chevaleyre, Endriss, & Lang 2006). Any restriction we might impose on goals (e.g., we may only want to allow clauses as formulas) or weights (e.g., we may not want to allow negative weights) and any choice we make regarding F gives rise to a different language.

Definition 3 (Languages and classes of utility functions)**.** *Let* $\Phi \subseteq \mathcal{L}_{\mathcal{PS}}$ *be a syntactic restriction on formulas,* $W \subseteq \mathbb{R}$ *a restriction on weights, and* F *an aggregation function. Then* $\mathcal{L}(\Phi, W, F)$ *is the language defined by these choices and* $\mathcal{U}(\Phi, W, F)$ *is the class of utility functions that can be generated by goalbases belonging to* $\mathcal{L}(\Phi, W, F)$*.*

We say that a language $\mathcal{L}$ is *expressively complete* for a given class of utility functions if it can represent any function in that class. To construct languages, we shall consider the following types of restrictions on formulas:

- *cubes* (conjunctions of literals) and *clauses*;
- *positive* formulas (without negation); and
- k-*formulas* (formulas with at most k atoms).

We also use combinations of these, e.g., 2-*clauses* are clauses of length at most 2, while *pcubes* (short for "positive cubes") are conjunctions of atoms. We write "*atoms*" and "*literals*" instead of 1-*pcubes* and 1-*cubes*.

Regarding weights: We consider weights without any restriction ("*all*") and positive weights ("*pos*", which we shall take to include zero[2]). Concerning aggregation, we concentrate on the $\max$-operator, but also consider summation. For example, $\mathcal{U}(\mathit{cubes}, \mathit{pos}, \Sigma)$ is the class of utility functions that can be expressed by means of goalbases consisting of conjunctions of literals with positive weights, using summation as the aggregator.

In this paper we analyze *max languages* (with $F = \max$) and draw some comparisons with *sum languages* ($F = \Sigma$). When using a max language, only the weight of the most important goal satisfied by a given alternative matters. An example of a simple max language is $\mathcal{L}(\mathit{atoms}, \mathit{all}, \max)$, which allows us to assign a value to any atomic proposition and the utility of a state is equal to the value of the most valuable proposition that is true in that state. $\mathcal{L}(\mathit{atoms}, \mathit{all}, \max)$ is known as the *unit-demand valuation* (of which the aforementioned *simple* unit-demand valuation is a special case) in the literature on combinatorial auctions (Nisan 2006).

Related languages

Much work on preference modeling using weighted goals has concentrated on representing ordinal preferences (Lang 2004; Coste-Marquis *et al.* 2004). Here we shall limit our discussion to languages for representing utility functions.

Chevaleyre, Endriss, & Lang (2006) have analyzed the expressivity of *sum languages* in detail and given several succinctness results. Further succinctness as well as complexity results are proved in (Uckelman & Endriss 2007).

Boutilier and Hoos (2001) suggest a variant of $\mathcal{L}(\mathit{pos}, \mathit{pos}, \Sigma)$, the sum language of positive formulas with positive weights, as a means for communicating bids in a combinatorial auction. There are two differences between their language and $\mathcal{L}(\mathit{pos}, \mathit{pos}, \Sigma)$. First, they also allow for the logic connective XOR, which we do not consider here. Including additional connectives can improve succinctness, so is attractive from a pragmatic point of view, but it does not make an important difference as far as the basic principles of the approach are concerned. Second, Boutilier and Hoos (2001) also allow for weights to be assigned to subformulas of goals. The utility of an alternative is then computed as the sum of the weights of all the subformulas satisfied. This does allow us to express some utility functions more concisely, but it does not affect succinctness in the technical sense to be defined later in this paper (to be precise, goalbase size decreases by at most a quadratic factor), nor does it affect the expressivity of the language.

Lafage and Lang (2000) discuss the definition of utility functions in terms of weighted goals from an axiomatic point of view. These authors have considered different aggregators, including summation and $\max$. An important difference from our languages is that they aggregate disutilities of

[2]Zero is not needed as a weight for sum languages because $\sum \emptyset = 0$, but is needed for max languages because $\max \emptyset = -\infty$.

violated goals rather than utilities of satisfied goals. In the case of summation, for a given goalbase G, both approaches lead to the same utility function (modulo “shifting” by a constant).[3] However, for $\max$ there are important differences: If we compare alternatives M and M', we may prefer M if we go by the most important goal satisfied but M' if we go by the most important goal violated.

Another important group of languages for modeling cardinal preferences are the OR/XOR family of bidding languages for combinatorial auctions (Nisan 2006). While these languages also use logical connectives, the connectives are interpreted differently there. In the OR language, a set of atomic bids $\langle B, p\rangle$, each consisting of a bundle of goods and a price, is taken to represent a function that maps any set of goods X to the maximal sum of prices that is achievable by accepting any set of non-overlapping bids so that X covers all the goods mentioned in those accepted bids. We now give a possible translation of the OR bidding language into a sum language in our framework:

1. Interpret goods as propositional variables and replace each bid $\langle B, p\rangle$ by the weighted goal $(\bigwedge B, p)$.
2. For each variable q in the resulting goalbase, replace the ith occurrence of q by the new variable q_i.
3. For any $i \neq j$, add $(q_i \wedge q_j, -\Omega)$ to the goalbase, with some excessively large value Ω.

The last step ensures that satisfying any two goals corresponding to overlapping bundles is never an attractive option. For all other alternatives, the utility values induced by this goalbase (using summation for the aggregation) are the same as the those defined by the original OR-bid.

The XOR bidding language is easier to interpret in our framework. For this language, the auctioneer may accept at most one atomic bid per bidder. This means that the value of a set of goods X is the highest price attached to any of its subsets within the atomic bids submitted. This is equivalent to $\mathcal{L}(\mathit{pcubes}, \mathit{pos}, \max)$ in our framework. Combinations of XOR and OR have also been considered in the literature (Nisan 2006; Sandholm 2002).

Superfluous goals

For max languages, some weighted goals in a goalbase may never contribute to the utility of an alternative. We conclude this section by introducing the terminology for speaking about this kind of situation and stating some simple facts about potentially superfluous goals.

Definition 4 (Properties of weighted goals)**.**

- *Call* $(\varphi, w_\varphi) \in G$ dominated *if there exists a* $(\psi, w_\psi) \in G$ *such that* $\varphi \models \psi$ *and* $w_\varphi < w_\psi$.
- *Call* $(\varphi, w_\varphi) \in G$ active *in a state* M *when* $M \models \varphi$ *and for all* ψ, *if* $M \models \psi$ *then* $w_\psi \leq w_\varphi$.
- *Call* $(\varphi, w_\varphi) \in G$ superfluous *if for every state* M *where* (φ, w_φ) *is active, the set* $\{(\psi, w_\psi) : M \models \psi$ *and* w_ψ *is maximal*$\}$ *is not a singleton.*

Note that if there are no models for which (φ, w) is active, then the superfluity condition is fulfilled vacuously, so (φ, w) is superfluous in that case.

Fact 1. *Fix* $(\varphi, w_\varphi) \in G$. *Then:*

1. *If* (φ, w_φ) *is dominated, then* (φ, w_φ) *is never active.*
2. *If* (φ, w_φ) *is never active, then* (φ, w_φ) *is superfluous.*
3. *If* (φ, w_φ) *is superfluous, then* $G \equiv_{\max} G \setminus \{(\varphi, w_\varphi)\}$.

Note, however, that *superfluous* does not imply *never active*: For example, in $\{(a, 1), (b, 1), (a \wedge b, 1)\}$, $(a \wedge b, 1)$ is superfluous but nonetheless active in the state $\{a, b\}$.

Fact 2. *If* G *contains no superfluous formulas, then every* $(\varphi, w) \in G$ *has a state in which it is uniquely active.*

We will be interested in goalbases that are *minimal* in the sense of there being no smaller goalbase that would generate the same utility function.

Fact 3. *If* G *is a minimal goalbase for a utility function* $u \in \mathcal{U}(\Phi, W, \max)$, *then* G *contains no superfluous formulas.*

Expressivity

In this section we present several expressivity results, answering the question what classes of utility functions can be represented using which languages. We concentrate on max languages. For similar results for sum languages we refer to (Chevaleyre, Endriss, & Lang 2006). We begin by establishing some simple results regarding equivalences between languages, which allow us to narrow down the range of languages to be considered in the remainder of the section. We then characterize the expressivity of the most important (distinct) languages.

Equivalences

We are interested in comparing languages generated by different types of goalbases, such as positively weighted clauses or literals with arbitrary weights. Next we establish several equivalences amongst languages and thereby show that we actually only need to consider a subset of all the languages that can be defined in this manner. Furthermore, if we are interested only in monotone utility functions, then we can further reduce the range of languages to consider.

We first show that disjunction is not a helpful connective for max languages:

Proposition 4. $G \cup \{(\varphi_1 \vee \cdots \vee \varphi_n, w)\} \equiv_{\max} G \cup \{(\varphi_i, w) : 1 \leq i \leq n\}$.

This tells us that with $\max$ as our aggregator, disjunctions as main connectives do not contribute to a language’s expressivity. In fact, as any formula has an equivalent representation in disjunctive normal form, this tells us that disjunction can *never* increase expressive power of a language. In particular, we get the following equivalences between languages:

Corollary 5. *Fix* $W \subseteq \mathbb{R}$. *Then:*

1. $\mathcal{U}(\mathit{pclauses}, W, \max) = \mathcal{U}(\mathit{atoms}, W, \max)$.

[3]Under summation, for a given goalbase G, in our setting we have $u_G(M) = \sum\{w : (\varphi, w) \in G$ and $M \models \varphi\}$, while in the penalty framework the *disutility* of M is $du_G(M) = \sum\{w : (\varphi, w) \in G$ and $M \models \neg\varphi\}$. The sum $u_G(M) + du_G(M)$ is constant; it is equal to $\sum\{w : (\varphi, w) \in G\}$ for any M.

2. $\mathcal{U}(\textit{clauses}, W, \max) = \mathcal{U}(\textit{literals}, W, \max)$.
3. $\mathcal{U}(\textit{pos}, W, \max) = \mathcal{U}(\textit{pcubes}, W, \max)$.
4. $\mathcal{U}(\textit{all}, W, \max) = \mathcal{U}(\textit{cubes}, W, \max)$.

The same is *not* true for sum languages. E.g., under summation clauses are more expressive than literals: For $W = \mathbb{R}$, clauses can express *all* utility functions, while literals can express only modular functions (Chevaleyre, Endriss, & Lang 2006). For each of the above equivalences, there is a set of weights W which violate it under summation.

A utility function u is called *monotone* if $M \subseteq M'$ implies $u(M) \leq u(M')$. Monotonicity is a reasonable assumption for many applications, in particular if propositional variables are interpreted as goods. Next we show that negation is not a helpful operation in case we are only interested in modeling monotone functions.

Proposition 6. *Fix G. If $u_{G \cup \{(\bigwedge X \wedge \neg a, w)\}, \max}$ is monotone, then $G \cup \{(\bigwedge X \wedge \neg a, w)\} \equiv_{\max} G \cup \{(\bigwedge X, w)\}$.*

The following result shows that we can further reduce the range of languages to consider if we limit ourselves to monotone utility functions. It follows immediately from Proposition 6 and Corollary 5. (Note that $\bigwedge \emptyset = \top$.)

Corollary 7. *Let* MONO *be the class of monotone utility functions. Fix $W \subseteq \mathbb{R}$. Then:*

1. $\mathcal{U}(\textit{clauses}, W, \max) \cap \text{MONO} =$
   $\mathcal{U}(\textit{atoms} \cup \{\top\}, W, \max) \cap \text{MONO}$.
2. $\mathcal{U}(\textit{all}, W, \max) \cap \text{MONO} = \mathcal{U}(\textit{pcubes}, W, \max) \cap \text{MONO}$.

Correspondence results

Corollary 5 tells us that the interesting languages, expressivity-wise, are those based on cubes, positive cubes, literals, and atoms. We prove that cubes are expressively complete for the full range of utility functions and that positive cubes correspond to the class of monotone functions:

Proposition 8. *$\mathcal{U}(\textit{cubes}, \textit{all}, \max)$ is the class of all utility functions.*

Proposition 9. *$\mathcal{U}(\textit{pcubes}, \textit{all}, \max)$ is the class of monotone utility functions.*

As mentioned earlier, $\mathcal{U}(\textit{atoms}, \textit{all}, \max)$ is the class of unit-demand valuations. We are not aware of a property of utility functions referred to in the literature that would characterize $\mathcal{U}(\textit{literals}, \textit{all}, \max)$. This is a generalization of the unit-demand valuation which also allows us to specify a value for *not* receiving a particular item.

By restricting the set of weights W we can capture classes of utility functions with a particular range. $\mathcal{U}(\textit{pcubes}, \textit{pos}, \max)$, for instance, is the class of non-negative monotone functions. This class is known to be equal to $\mathcal{U}(\textit{pos}, \textit{pos}, \Sigma)$ (Chevaleyre, Endriss, & Lang 2006).[4] This is a case where a syntactically simple languages is more expressive with max than with sum.

On the other hand, some very simple classes of utility functions are hard to capture using max aggregation. For instance, a utility function u is called *modular* iff $u(M \cup M') = u(M) + u(M') - u(M \cap M')$ for all $M, M' \in 2^{\mathcal{PS}}$. Modular functions are nicely captured by $\mathcal{U}(\textit{literals}, \textit{all}, \Sigma)$ (Chevaleyre, Endriss, & Lang 2006). However, there is no natural restriction to formulas that would allow us to characterize the modular functions under max aggregation. On the contrary, among the max languages considered here, only $\mathcal{L}(\textit{cubes}, \textit{all}, \max)$ can express all modular functions, and this language is so powerful that it can actually express *all* utility functions.

[4]To be precise, $\mathcal{U}(\textit{pcubes}, \textit{pos}, \max) \supseteq \mathcal{U}(\textit{pos}, \textit{pos}, \Sigma)$, because in the max language we can also express partially defined functions returning $-\infty$ for some states.

Succinctness

An important criterion for selecting one language over another for some application is, as mentioned previously, the size of the representations of utility functions in that language. (Recall that this is why we seek to avoid explicit representations of utility functions.) In this section, we present both absolute and comparative notions of succinctness, so that we can make precise claims about the efficiency of various goalbase languages.

Absolute succinctness

Definition 5. *The* length of a formula φ *is the number of occurrences of atoms it contains. The* size of a weighted goal (φ, w) *is the length of φ plus the number of bits needed to store w ($\log w$ bits). The* size of a goalbase G, *written as* $\text{size}(G)$, *is the sum of the sizes of the weighted goals in G.*

Often we consider families of utility functions $\{u_n\}_{n \in \mathbb{N}}$ where $n = |\mathcal{PS}_n|$. ($\mathcal{PS}_n$ is a set containing n propositional variables.) Suppose that we have a corresponding family of goalbases $\{G_n\}_{n \in \mathbb{N}}$ for which $u_n = u_{G_n}$. Unless the number of bits required to represent the weights in G_n grows superexponentially in n, the size contributed by the weights can be safely ignored when considering how $\text{size}(G_n)$ grows with n, since $\log c^{p(n)}$ is polynomial in n for fixed constants c and polynomials p. Every family of utility functions considered here has weights which are independent of n, and so we disregard the size of the weights in our succinctness results.

In absolute terms, there is a strong dependency of the size of representations in max languages on the size of the range of the utility function being represented:

Proposition 10. $|G| \geq |\text{ran}\, u_{G,\max}|$.

While the size of range of a utility function serves as a lower bound on the size of its representation in any max language, there is no such relationship for sum languages: E.g., if $G = \{(a_i, 2^i) : a_i \in \mathcal{PS}\}$, then $u_{G,\Sigma}$ has a large range (every value in $0, \ldots, 2^{|\mathcal{PS}|} - 1$) despite that G is itself small (using only $|\mathcal{PS}|$ atoms).

Proposition 11. *Let $\{G_n\}_{n \in \mathbb{N}}$ be a family of minimal goalbases such that $G_n \in \mathcal{L}(\textit{pcubes}, \textit{pos}, \max)$ and $|\mathcal{PS}_n| = n$. Then $\text{size}(G_n)$ is polynomial iff $|G_n|$ is polynomial.*

Furthermore, by Proposition 10 we have that if $|G_n|$ is polynomial then $|\text{ran}\, u_{G_n}|$ is polynomial. However, the converse does not hold: Let $G_n = \{(\bigwedge X, 1) : |X| = \binom{n}{n/2}\} \cup$

$\{(\top, 0)\}$. Here, $|\operatorname{ran} u_{G_n}| = 2$ but $|G_n| = \binom{n}{n/2}$ is superpolynomial and G_n is minimal in $\mathcal{L}(\mathit{pcubes}, \mathit{pos}, \max)$.

Several sum languages are sufficiently restrictive as to have exactly one minimal representation of each representable utility function, a property we call *unique representations*. This also occurs for at least one max language:

Proposition 12. *Minimal representations in the language* $\mathcal{L}(\mathit{pcubes}, \mathit{pos}, \max)$ *are unique.*

Furthermore, the proof of this proposition shows how to construct the minimal representation for any representable utility function.

Relative succinctness

Frequently one language contains shorter representations of the same utility functions than does another language. Here we offer a definition of relative succinctness to make this notion precise:

Definition 6. *Let $\mathcal{L}$ and $\mathcal{L}'$ be goalbase languages, F and F' aggregators, and $\mathcal{U}$ a class of utility functions for which $(\mathcal{L}, F)$ and $(\mathcal{L}', F')$ are expressively complete. Then $(\mathcal{L}, F) \preceq_{\mathcal{U}} (\mathcal{L}', F')$ iff there exists a function $f : \mathcal{L} \to \mathcal{L}'$ and a polynomial p such that for all $G \in \mathcal{L}$, if $u_{G,F} \in \mathcal{U}$ then $u_{G,F} = u_{f(G),F'}$ and* $\operatorname{size}(f(G)) \leq p(\operatorname{size}(G))$.

Read $(\mathcal{L}, F) \preceq_{\mathcal{U}} (\mathcal{L}', F')$ as: $(\mathcal{L}', F')$ is at least as succinct as $(\mathcal{L}, F)$ over the class $\mathcal{U}$. When $(\mathcal{L}', F')$ is strictly more succinct than $(\mathcal{L}, F)$—that is, in no case are representations more than polynomially worse, and in at least one case, they are exponentially better in $(\mathcal{L}', F')$—we write $(\mathcal{L}, F) \prec_{\mathcal{U}} (\mathcal{L}', F')$. When we have nonstrict succinctness in both directions, we write $(\mathcal{L}, F) \sim_{\mathcal{U}} (\mathcal{L}', F')$; when we have nonstrict succinctness in neither direction, i.e., incomparability, we write $(\mathcal{L}, F) \perp_{\mathcal{U}} (\mathcal{L}', F')$. Whenever a succinctness relation appears unsubscripted (i.e., without an explicit class of comparison), then implicitly

$$\mathcal{U} = \{u_{G,F} : G \in \mathcal{L}\} \cap \{u_{G',F'} : G' \in \mathcal{L}'\},$$

which is the *expressive intersection* of $(\mathcal{L}, F)$ and $(\mathcal{L}', F')$.

This definition of succinctness is a generalization of those given in (Chevaleyre, Endriss, & Lang 2006) and (Uckelman & Endriss 2007). The definition from the former does not permit comparison of languages which differ in expressive power, while the latter fixes the class of comparison $\mathcal{U}$ as the expressive intersection of the two languages. Later in this section we give an illustration of circumstances in which being explicit about the class of comparison is important.

Succinctness between max languages

When comparing the succinctness of any two max languages, notice that the available weights play no role in the outcome. If $\mathcal{L}(\Phi, W, \max)$ and $\mathcal{L}(\Psi, W', \max)$ are the languages under comparison, then for any utility function u representable in both languages, $\operatorname{ran} u \subseteq W \cap W'$. Due to this, any weighted formula (φ, w) where $w \notin W \cap W'$ will be superfluous when it occurs in a representation of u in either language. Since only minimal representations are relevant for succinctness, we can disregard all representations of u which use weights outside of $W \cap W'$:

Proposition 13. $\mathcal{L}(\Phi, W, \max) \odot \mathcal{L}(\Psi, W', \max)$ *iff* $\mathcal{L}(\Phi, W \cap W', \max) \odot \mathcal{L}(\Psi, W \cap W', \max)$*, for* $\odot \in \{\preceq, \prec, \sim, \perp\}$.

The same does not hold for arbitrary sum languages.

Proposition 14. *For* $k \in \mathbb{N}$, $\mathcal{L}(k\text{-}\mathit{pcubes}, W, \max) \sim \mathcal{L}(k\text{-}\mathit{cubes}, W', \max)$.

Recall that Proposition 4 shows that disjunctions as main connectives do not affect succinctness, because the translation required to eliminate disjunction does not affect the size of a goalbase. However, the same is not necessarily true for disjunctions that occur within the scope of other connectives. Therefore, for our analysis of expressivity we can safely ignore disjunction, but for succinctness we cannot.

Proposition 15. $\mathcal{L}(\mathit{pcubes}, \mathit{pos}, \max) \prec \mathcal{L}(\mathit{pos}, \mathit{pos}, \max)$.

Succinctness between max and sum languages

Here we examine the succinctness of some max languages with respect to some sum languages. These results indicate that each aggregator favors short representations for certain kinds of utility functions; whether max or sum is better for a particular application will depend on what utility functions agents are likely to have.

Proposition 16. $\mathcal{L}(\mathit{pcubes}, \mathit{pos}, \max) \perp \mathcal{L}(\mathit{pcubes}, \mathit{all}, \Sigma)$.

Proposition 17. $\mathcal{L}(\mathit{pcubes}, \mathit{pos}, \max) \perp \mathcal{L}(\mathit{pclauses}, \mathit{all}, \Sigma)$.

Proposition 18. $\mathcal{L}(\mathit{pcubes}, \mathit{all}, \Sigma) \prec \mathcal{L}(\mathit{atoms}, \mathit{all}, \max)$.

Proposition 19. $\mathcal{L}(\mathit{pclauses}, \mathit{all}, \Sigma) \sim \mathcal{L}(\mathit{atoms}, \mathit{all}, \max)$.

Proposition 20. $\mathcal{L}(\mathit{cubes}, \mathit{pos}, \max) \prec \mathcal{L}(\mathit{atoms}, \mathit{pos}, \Sigma)$.

We are now in a position to demonstrate why a more general definition of succinctness (subscripted by the comparison class) was needed: In Proposition 18 we showed that $\mathcal{L}(\mathit{pcubes}, \mathit{all}, \Sigma) \prec \mathcal{L}(\mathit{atoms}, \mathit{all}, \max)$, while in Proposition 20, we showed that $\mathcal{L}(\mathit{cubes}, \mathit{pos}, \max) \prec \mathcal{L}(\mathit{atoms}, \mathit{pos}, \Sigma)$. It is obvious that $\mathcal{L}(\mathit{pcubes}, \mathit{all}, \Sigma) \sim \mathcal{L}(\mathit{atoms}, \mathit{pos}, \Sigma)$, since the expressive intersection of the two languages is the whole of the latter, and in the latter every representation is small. Finally, $\mathcal{L}(\mathit{atoms}, \mathit{all}, \max) \sim \mathcal{L}(\mathit{cubes}, \mathit{pos}, \max)$ because their expressive intersection is the class of nonnegative unit-demand utility functions, which have small representations in both languages. We now have the following situation:

$$\begin{tikzcd}
\mathcal{L}(\mathit{pcubes}, \mathit{all}, \Sigma) \arrow[r, phantom, "\prec"] \arrow[d, phantom, "\wr"] & \mathcal{L}(\mathit{atoms}, \mathit{all}, \max) \arrow[d, phantom, "\wr"] \\
\mathcal{L}(\mathit{atoms}, \mathit{pos}, \Sigma) \arrow[r, phantom, "\succ"] & \mathcal{L}(\mathit{cubes}, \mathit{pos}, \max)
\end{tikzcd}$$

From this it can be seen that the *unsubscripted* succinctness relation is not transitive. This comes about because no two pairs of these four languages have the same expressive intersection. Because the expressive intersection of the languages can shift from comparison to comparison, we must make explicit the class of utility functions over which the comparison is being made if we wish to do more than pairwise comparison.

Complexity

Besides expressivity and succinctness, another important property of a preference representation language is the complexity of associated tasks. In this section we discuss the complexity of checking whether a certain level of utility is achievable for a given representation as well as the complexity of the *winner determination problem* of finding an allocation that maximizes overall utility for a group of agents.[5]

Maximal utility

A natural question to ask is which alternative will yield maximal utility for a given compact representation of a utility function. Here we formalize the related decision problem asking whether a certain level of utility is achievable.

Definition 7 (Maximal utility problem)**.** *The decision problem* $\textsc{Max-Util}(\mathcal{L}, F)$ *is defined as: Given a goalbase* $G \in \mathcal{L}$*, an aggregator* F*, and an integer* K*, check whether there is a model* $M \in 2^{\mathcal{PS}}$ *such that* $u_{G,F}(M) \geq K$*.*

$\textsc{Max-Util}$ is known to be NP-complete for most sum languages; only under severe restrictions on the syntax of formulas can we hope to get a polynomial decision problem (Chevaleyre, Endriss, & Lang 2006; Uckelman & Endriss 2007). In contrast to this, solving $\textsc{Max-Util}$ for *any* max language is trivial:

Fact 21. $\textsc{Max-Util}(\Phi, W, \max)$ *is linear in the size of the goalbase, for any* $\Phi \subseteq \mathcal{L}_{\mathcal{PS}}$ *and any* $W \subseteq \mathbb{Q}$.[6]

An algorithm solving $\textsc{Max-Util}$ simply has to iterate over the formulas in the goalbase, answer affirmatively as soon as it encounters a (φ, w) for which $w \geq K$, and answer negatively otherwise. (Recall that, by Definition 1, φ must be satisfiable, so any state $M \models \varphi$ has value $\geq w$.)

This complexity result requires some discussion. First, if we drop the condition that goalbases may only include satisfiable formulas, then $\textsc{Max-Util}$ becomes NP-complete, as we would then have to solve a satisfiability problem for each and every goal. Second (if we do not drop the satisfiability condition), we need to be careful about how we interpret the low complexity result for $\textsc{Max-Util}$. Note that our algorithm does not *compute* the actual model M yielding the desired level of utility; it only checks whether such an M *exists*. If we also require M itself, then we still need to extract a satisfying model M from the goal φ with the highest weight. In general, the problem of finding a satisfying assignment for a formula that is already known to be satisfiable is intractable. Hence, in general, the low complexity of $\textsc{Max-Util}$ for max languages does not imply low complexity of actually finding the best alternative. In contrast to this observation, for sum languages, we are not aware of any case where the complexity of checking existence of an alternative giving utility $\geq K$ and computing that alternative differ: For languages with an NP-complete $\textsc{Max-Util}$ this is a non-issue; for all sum languages with polynomial $\textsc{Max-Util}$ the proofs are constructive and directly show the computation of the top alternative to be polynomial (Uckelman & Endriss 2007). Finally, we stress that both our qualifications of Fact 21 (regarding the assumption of goals being satisfiable and the differences between $\textsc{Max-Util}$ and the problem of computing the best alternative) vanish for the restricted languages considered in this paper. For both cubes and clauses (and any of their sub-languages) satisfiability checking and extracting a model from a given (satisfiable) formula are trivial tasks.

[5]We shall assume basic familiarity computational complexity; see (Papadimitriou 1994) for one of several good introductions.

[6]$\textsc{Max-Util}$ has different parameters in Definition 7 and Fact 21: $\mathcal{L}$ in Definition 7 is the combination of Φ and W from Fact 21. We do not write Φ, W in Definition 7 because the components of $\mathcal{L}$ are not used in the definition.

Winner determination

When there are several agents, each with a utility function encoded using the same language, then the *winner determination problem* (WDP), the problem of finding a solution maximizing collective utility, is of interest. By "solution" we mean a partition of the set of propositional variables among the agents, thereby fixing a model for each of them. This definition is natural, for instance, if we think of variables as goods (other types of solutions, such as finding a single model that maximizes collective utility, are also of interest, but shall not be considered here).

There are a number of ways in which to define *collective utility* (Moulin 1988). For instance, the *utilitarian* collective utility of an alternative is the sum of the individual utilities. Optimizing with respect to utilitarian collective utility is equivalent to the winner determination problem in combinatorial auctions, where it is interpreted as finding an allocation of goods to bidders that would maximize the sum of the prices offered (Sandholm 2002). *Egalitarian* collective utility is defined as the utility of the agent worst off. Other options include maximizing the utility of the agent that is best off (*elitist collective utility*) and maximizing the product of individual utilities (*Nash product*). Formally, a collective utility function is a function $\sigma : \mathbb{R}^n \to \mathbb{R}$ mapping vectors of individual utilities to the reals.

Definition 8 (Winner determination problem)**.** *The decision problem* $\mathrm{WDP}(\mathcal{L}, F, \sigma)$ *for* n *agents is defined as: Given goalbases* $G_i \in \mathcal{L}$ *for* $i \in \{1..n\}$*, an aggregator* F*, a collective utility function* σ*, and an integer* K*, check whether there exists a partition* $\langle M_1, \ldots, M_n \rangle$ *of* $\mathcal{PS}$ *such that* $\sigma(u_{G_1,F}(M_1), \ldots, u_{G_n,F}(M_n)) \geq K$*.*

Clearly, the WDP is in NP for any of the collective utility functions mentioned before, because computing the collective utility for a given alternative can be done in polynomial time in all these cases. It is also easy to see that the $\mathrm{WDP}(\mathcal{L}, F, \sigma)$ is NP-hard whenever $\textsc{Max-Util}(\mathcal{L}, F)$ is. This follows from the fact that the two problems coincide for $n = 1$ (for any reasonable choice of σ). So the interesting cases to investigate are languages that give rise to an easy $\textsc{Max-Util}$ problem. (In the remainder of this section we focus again on the case of $F = \max$ only.)

For the *elitist* collective utility function $\sigma = \max$, winner determination is easy for the same reasons as those stated in support of Fact 21.

Fact 22. $\mathrm{WDP}(\Phi, W, \max, \max)$ *is linear in the combined size of the goalbases, for any* $\Phi \subseteq \mathcal{L}_{\mathcal{PS}}$ *and any* $W \subseteq \mathbb{Q}$*.*

For the *egalitarian* collective utility function $\sigma = \min$, NP-completeness follows immediately from results of Bouveret *et al.* (2005; 2007). As they do not consider negation, their result applies to positive formulas (NP-hardness transfers upwards to superlanguages). However, for atomic formulas the problem does become polynomial.

Fact 23. $\mathrm{WDP}(\textit{pos}, \mathbb{Q}, \max, \min)$ *is NP-complete.*

For the *utilitarian* collective utility function $\sigma = \Sigma$, we also get NP-completeness. The proof of NP-completeness of the WDP for combinatorial auctions (using, for instance, SET PACKING) is well-known. We state here a variant of the result, due to van Hoesel and Müller (2001), that directly corresponds to one of the simplest languages in our framework.

Fact 24. $\mathrm{WDP}(\textit{2-pcubes}, \{0, 1\}, \max, \Sigma)$ *is NP-complete.*

Finally, we consider the *Nash collective utility function* $\sigma = \Pi$ (product). For the Nash product to be a meaningful metric of social welfare we need to restrict ourselves to positive utilities. So in this context we assume that all weights are positive and that only goalbases specifying fully defined utility functions are used (e.g., by including $(\top, 0)$ in all goalbases). To the best of our knowledge, the complexity of this variant of the WDP has not been studied before. In the case of max languages, it is possible to give a simple reduction from the utilitarian case to this one.

Proposition 25. $\mathrm{WDP}(\textit{2-pcubes}, \mathbb{Q}^+, \max, \Pi)$ *is NP-complete.*

The simple reduction in the proof is possible because we are working with max languages. In this setting, the utility of any model M will always be equal to one of the weights in the goalbase. For a sum language, for instance, it is not immediately clear how to translate the weights, as utilities will end up being sums of several weights (so *this* proof would not go through, but the corresponding WDP can still be expected to be NP-complete).

Conclusion

We have analyzed the expressivity, succinctness and complexity of languages for representing utility functions that are based on weighted propositional formulas. For this analysis, we have focused on the case where the utility of an alternative is given by the maximum weight among the weights of formulas that are satisfied by the alternative. Our results show that there are substantial differences in view of all three of these properties when comparing languages using summation for aggregation with those using max.

For expressivity, we were able to give a complete picture for the most important languages. This is due to the fact that including disjunction into the propositional language used to specify goals was found not to affect expressivity. More fine-grained results could probably still be obtained by considering languages generated by specific weight sets or by looking into formulas of limited length (although in some cases such results will be obvious consequences of what is known already). Concerning relative succinctness, we have established the most important relationships within the max languages and have also given several results that show how they relate to the sum languages. However, here a number of questions are still open and deserve attention in future work. Lastly, concerning complexity, we have observed that finding an alternative maximizing utility is computationally easy for max languages for all practical purposes. We have then discussed to what extent known results apply to our framework when analyzing the complexity of the problem of maximizing collective utility for a group of agents, and we have also proved a new result regarding the complexity of finding an allocation that maximizes the product of individual utilities (the Nash product).

References

Boutilier, C., and Hoos, H. H. 2001. Bidding languages for combinatorial auctions. In *Proc. Intl. Joint Conf. on Artificial Intelligence (IJCAI-2001)*, 1211–1217. Morgan Kaufmann.

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artifcial Intelligence Research* 21:135–191.

Bouveret, S.; Fargier, H.; Lang, J.; and Lemaître, M. 2005. Allocation of indivisible goods: A general model and some complexity results. In *Proc. 4th Intl. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, 1309–1310. ACM Press.

Bouveret, S. 2007. *Allocation et partage équitables de ressources indivisibles: modélisation, complexité et algorithmique*. Ph.D. Dissertation, Supaéro/University of Toulouse.

Chevaleyre, Y.; Endriss, U.; and Lang, J. 2006. Expressive power of weighted propositional formulas for cardinal preference modelling. In Doherty, P.; Mylopoulos, J.; and Welty, C., eds., *Proc. 10th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR-2006)*, 145–152. AAAI Press.

Coste-Marquis, S.; Lang, J.; Liberatore, P.; and Marquis, P. 2004. Expressive power and succinctness of propositional languages for preference representation. In Dubois, D.; Welty, C. A.; and Williams, M.-A., eds., *Proc. 9th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR-2004)*, 203–212. AAAI Press.

Gonzales, C., and Perny, P. 2004. GAI networks for utility elicitation. In Dubois, D.; Welty, C. A.; and Williams, M.-A., eds., *Proc. 9th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR-2004)*, 224–234. AAAI Press.

van Hoesel, S., and Müller, R. 2001. Optimization in electronic markets: examples in combinatorial auctions. *Netnomics* 3(1):23–33.

Lafage, C., and Lang, J. 2000. Logical representation of preferences for group decision making. In *Proc. 7th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR-2000)*, 457–468. Morgan Kaufmann.

Lang, J. 2004. Logical preference representation and combinatorial vote. *Annals of Mathematics and Artificial Intelligence* 42(1–3):37–71.

Moulin, H. 1988. *Axioms of Cooperative Decision Making*. Cambridge University Press.

Nisan, N. 2006. Bidding languages for combinatorial auctions. In Cramton, P.; Shoham, Y.; and Steinberg, R., eds., *Combinatorial Auctions*. MIT Press. 215–232.

Papadimitriou, C. 1994. *Computational Complexity*. Addison-Wesley.

Pinkas, G. 1995. Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence* 77(2):203–247.

Sandholm, T. W. 2002. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence* 135(1–2):1–54.

Uckelman, J., and Endriss, U. 2007. Preference representation with weighted goals: Expressivity, succinctness, complexity. In *Proc. AAAI Workshop on Preference Handling for Artificial Intelligence (AiPref-2007)*, 85–92.

Appendix

Proof of Proposition 4

Proof. Fix a state X. If $w_{\varphi_1 \vee \cdots \vee \varphi_n}$ is not the maximum w_ψ such that $X \models \psi$, then some other $(\psi, w_\psi) \in G$ is. Since $w_{\varphi_i} = w_{\varphi_1 \vee \cdots \vee \varphi_n}$, then $w_\psi > w_{\varphi_i}$ for all i. In this case, G alone determines the value of the left and right goalbases.

If $w_{\varphi_1 \vee \cdots \vee \varphi_n}$ is the maximum w_ψ such that $X \models \psi$, then some φ_i are such that $X \models \varphi_i$. For each such φ_i, we have $w_{\varphi_i} = w_{\varphi_1 \vee \cdots \vee \varphi_n}$ in the goalbase on the right, and so both the left and right have the same maximum. □

Proof of Proposition 6

Proof. First, observe that goalbases contain only satisfiable formulas, so $a \notin X$—otherwise $\bigwedge X \wedge \neg a$ would not be satisfiable. There are two cases to consider: States which are supersets of X, and states which are not.

- Write u for $u_{G \cup \{(\bigwedge X \wedge \neg a, w)\},\max}$. In states $M \supseteq X$, we have that $M \models \bigwedge X$. It must be the case that $u(M) \geq w$ because $u(X) \geq w$ and u is monotone. Therefore, substituting $(\bigwedge X, w)$ for $(\bigwedge X \wedge \neg a, w)$ cannot change the value at M, since the value in M is already at least w.
- In states $M \not\supseteq X$, we have that both $\bigwedge X \wedge \neg a$ and $\bigwedge X$ are false, and so cannot be active. Thus substituting $(\bigwedge X, w)$ for $(\bigwedge X \wedge \neg a, w)$ cannot change the value at M, as inactive formulas do not affect the value of a utility function.

Therefore, in all states M we have that

$$u_{G \cup \{(\bigwedge X \wedge \neg a, w)\},\max}(M) = u_{G \cup \{(\bigwedge X, w)\},\max}(M).$$

□

Proof of Proposition 8

Proof. Given a utility function u, define

$$G = \big\{\big(\bigwedge X \wedge \bigwedge\{\neg p : p \in \mathcal{PS} \setminus X\}, u(X)\big) : X \subseteq \mathcal{PS}\big\}$$

Since the formulas are the states, and as such are mutually exclusive, exactly one weight will be active in each state, and so $u(X) = u_G(X)$. □

Proof of Proposition 9

Proof. ($\Longrightarrow$) If $u \in \mathcal{U}(pcubes, all, \max)$, then u is monotone because positive cubes are monotone formulas and $\max$ is a monotone function.

($\Longleftarrow$) If u is monotone, then let $G = \{(\bigwedge X, u(X)) : X \subseteq \mathcal{PS}\}$. Note that for $Y \subseteq X$, $u(Y) = w_{\bigwedge Y} \leq w_{\bigwedge X} = u(X)$ follows directly from the monotonicity of u. In state X, $u_G(X) = \max\{w_{\bigwedge Y} : Y \subseteq X\} = w_{\bigwedge X}$. Hence $u_G(X) = u(X)$. □

Proof of Proposition 10

Proof. $u_{G,\max}(X) = \max\{w_\varphi : X \models \varphi\}$, so for every state X there must be some $w_\varphi = u(X)$. □

Proof of Proposition 11

Proof. From left to right is obvious. From right to left: Suppose that $|G_n| = p(n)$ grows polynomially in n. The longest positive cube for each n is $\bigwedge \mathcal{PS}$, which contains n atoms. Therefore, the $p(n)$ longest pcubes contain no more than $n \cdot p(n) \geq \text{size}(G_n)$ atoms, which is also a polynomial. □

Proof of Proposition 12

Proof. Fix $u \in \mathcal{U}(pcubes, pos, \max)$. Let $G_0 = \emptyset$. While $u_{G_i,\max} \neq u$: Choose a least state X for which $u_{G_i,\max}(X) \neq u(X)$. Let $G_{i+1} = G_i \cup \{(\bigwedge X, u(X))\}$. Call G the G_i at which the algorithm terminates.

Correctness: $u_{G,\max} = u$ because each iteration ends with one more state correct than in the previous iteration, and there are finitely many states. Setting a weight for $\bigwedge X$ cannot disturb the value of any state $Y \subset X$, as X is the least state where $\bigwedge X$ is true, and cannot prevent us from correctly setting the value of any state $Z \supset X$ during subsequent iterations because u is monotone. Note also that the order of choice of cubes of the same size makes no difference in the outcome.

Minimality: For any state X, either $\bigwedge X$ receives a weight or not. If $\bigwedge X$ receives a weight, then there is no state $Y \subset X$ for which $(\bigwedge Y, u(Y))$ dominates $(\bigwedge X, u(X))$. Furthermore, there is no state $Z \supset X$ for which $X \models \bigwedge Z$. Hence, if the algorithm assigns a weight to $\bigwedge X$, then this is the sole way in which we can make $u_{G,F}(X) = u(X)$. If, on the other hand, the algorithm produces a G where $\bigwedge X$ receives no weight, then at some step i in the construction $u_{G_i,F}(X)$ became correct before we reached state X. If we were to set a weight for $\bigwedge X$, it would be superfluous and so G would not be minimal. In summary: Any smaller G will give an incorrect value for some state, and any different, yet still correct, G will necessarily contain a superfluous formula.

Note that this is not an efficient algorithm for finding representations, as it requires us to check exponentially many states in order to set weights for them. □

Proof of Proposition 14

Proof. $\mathcal{L}(k\text{-}pcubes, W, \max) \preceq \mathcal{L}(k\text{-}cubes, W', \max)$ since the formulas of latter language are a superset of the formulas of the former. Because k-pcubes are monotone, by

Proposition 6, we have that any $G \in \mathcal{L}(k\text{-}cubes, W', \max)$ expressible in both languages is equivalent to some $G' \in \mathcal{L}(k\text{-}pcubes, W, \max)$ such that G' is formed from G by removing all of the negative literals from G. Thus, $\text{size}(G') \leq \text{size}(G)$, so it follows that $\mathcal{L}(k\text{-}pcubes, W, \max) \succeq \mathcal{L}(k\text{-}cubes, W', \max)$. □

Proof of Proposition 15

Proof. $\mathcal{L}(pcubes, pos, \max) \preceq \mathcal{L}(pos, pos, \max)$ because every pcube is a positive formula. For strict succinctness: The family of utility functions represented by

$$\{((p_1 \vee p_2) \wedge (p_3 \vee p_4) \wedge \cdots \wedge (p_{n-1} \vee p_n), 1)\}$$

in $\mathcal{L}(pos, pos, \max)$ grows linearly with n, while the minimal representation in $\mathcal{L}(pcubes, pos, \max)$ is

$$\{(\bigwedge_{k=1}^{n/2} p_{i_k}, 1) : i_1, \ldots, i_{n/2} \in \{1,2\} \times \{3,4\} \times \cdots \times \{n-1, n\}\}$$

which has size 2^{n-1} for any (even) n. □

Proof of Proposition 16

Proof. [$\not\preceq$] Let $u_n(X) = \sum_{a_i \in X} 2^i$. Then $|\text{ran}\, u_n| = 2^n$, and so if $G_{\max}$ represents u_n, then by Proposition 10, $|G_{\max}| \geq 2^n$. In $\mathcal{U}(atoms, pos, \Sigma)$, we have $G_\Sigma = \{(a_i, 2^i) : 0 \leq i < n\}$ which represents u_n. Hence $\mathcal{U}(pcubes, pos, \max) \not\preceq \mathcal{U}(pcubes, all, \Sigma)$.

[$\not\succeq$] Let

$$u(X) = \begin{cases} 1 & \text{if } X \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

which has a large representation in $\mathcal{U}(pcubes, all, \Sigma$. In $\mathcal{U}(pcubes, pos, \max)$, u is represented by $G = \{(p, 1) : p \in \mathcal{PS}\}$. Therefore $\mathcal{U}(pcubes, pos, \max) \not\succeq \mathcal{U}(pcubes, all, \Sigma)$. □

Proof of Proposition 17

Proof. [$\not\preceq$] Let

$$u(X) = \begin{cases} 1 & \text{if } X = \mathcal{PS} \\ 0 & \text{otherwise} \end{cases}$$

which is large in the language $\mathcal{L}(pclauses, all, \Sigma)$. In $\mathcal{L}(pcubes, pos, \max)$, u is represented by $\{(\bigwedge \mathcal{PS}, 1)\}$.

[$\not\succeq$] Same argument as for $\mathcal{U}(pcubes, all, \Sigma)$. □

Proof of Proposition 18

Proof. $\mathcal{U}(atoms, all, \max)$ corresponds to the class of unit-demand utility functions. Every unit-demand utility function is expressible linearly in $\mathcal{L}(atoms, all, \max)$ as $\{(a, u(a))\}_{a \in \mathcal{PS}}$. In $\mathcal{L}(pcubes, all, \Sigma)$ (which is fully expressive and has uniqueness (Chevaleyre, Endriss, & Lang 2006, proof of Prop. 13)), u is represented by $\{(\bigwedge X, w_{\bigwedge X}) : X \subseteq \mathcal{PS}\}$ where

$$w_{\bigwedge X} = (-1)^{|X|+1} \min_{a \in X} u(a).$$

For any unit-demand u which is also single-minded (i.e., exactly one $a \in \mathcal{PS}$ is such that $u(a) \neq 0$), $G = \{(a, u(a))\}$, the same as in $\mathcal{L}(atoms, all, \max)$. For non-single-minded u, let $X \subseteq \mathcal{PS}$ be the items which u assigns nonzero value. Then G will contain a nonzero weight for $w_{\bigwedge Y}$ whenever $Y \subseteq X$. So, for the family of simple unit-demand utility functions

$$u(X) = \begin{cases} 1 & \text{if } X \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

we have that representations in $\mathcal{L}(pcubes, all, \Sigma)$ are exponential in $|\mathcal{PS}|$. □

Proof of Proposition 19

Proof. Recall that $\mathcal{U}(atoms, all, \max)$ is the class of unit-demand utility functions. Simple unit-demand utility functions are expressible linearly in $\mathcal{L}(pclauses, all, \Sigma)$ as $\{(\bigvee \mathcal{PS}, 1)\}$. Consider complex unit-demand utility functions u (where items may differ in value but the value of a bundle is the value of the best item contained in it) expressed in $\mathcal{L}(pclauses, all, \Sigma)$. Without loss of generality, suppose that the items are ordered $a_1 \leq \cdots \leq a_n$ in value. Then

$$\{(\bigvee(\mathcal{PS} \setminus \{a_1, \ldots, a_{i-1}\}), u(a_i) - u(a_{i-1})\}_{1 \leq i \leq n}$$

is the unique minimal representative of u in $\mathcal{L}(pclauses, all, \Sigma)$, containing $n(n-1)/2$ atoms. □

Proof of Proposition 20

Proof. [$\preceq$] From (Chevaleyre, Endriss, & Lang 2006, Prop. 8) we have that $\mathcal{U}(atoms, pos, \Sigma)$ is the class of normalized nonnegative modular utility functions. From Proposition 9, $\mathcal{L}(cubes, pos, \max)$ is the class of nonnegative monotone utility functions, so $\mathcal{U}(atoms, pos, \Sigma)$ is the expressive intersection of the two languages. Every representation in $\mathcal{L}(atoms, pos, \Sigma)$ is linear in $|\mathcal{PS}|$. If $u(\{a\}) \geq 0$, then the atom a will appear somewhere in any representation of u in $\mathcal{L}(cubes, pos, \max)$, so no representations which grow logarithmically in $|\mathcal{PS}|$ are possible there.

[$\not\succeq$] Consider the family of utility functions $u(X) = |X|$. In $\mathcal{L}(atoms, pos, \Sigma)$, u is represented by $\{(a, 1) : X \in \mathcal{PS}\}$, which is linear in $|PS|$, while the unique representation in $\mathcal{L}(cubes, pos, \max)$ is $\{(\bigwedge X, |X|) : X \subseteq \mathcal{PS}\}$, which is exponential in $|PS|$. □

Proof of Proposition 25

Proof. We show NP-hardness by reduction from Fact 24. Suppose we are given an instance of the problem WDP($2\text{-}pcubes, \{0,1\}, \max, \Sigma$), with goalbases G_i and bound K. We construct new goalbases G'_i by replacing each weight w in G with 2^w. Now consider the instance of WDP($2\text{-}pcubes, \mathbb{Q}^+, \max, \Pi$) with the new goalbases G'_i and bound 2^K. Note that $w_1 + \cdots + w_n \geq K$ iff $2^{w_1} \times \cdots \times 2^{w_n} \geq 2^K$. Hence, a model M achieves utilitarian collective utility $\geq K$ with respect to goalbases G_i iff M achieves Nash collective utility $\geq 2^K$ with respect to goalbases G'_i. So the Nash WDP must be at least as hard as the utilitarian WDP. □