Representing Utility Functions via Weighted Goals[*]

Joel Uckelman[1], **Yann Chevaleyre**[2], **Ulle Endriss**[1], and **Jérôme Lang**[3]

[1] Institute for Logic, Language and Computation (ILLC)
University of Amsterdam
[2] Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision (LAMSADE)
Université Paris-Dauphine
[3] Institut de Recherche en Informatique de Toulouse (IRIT)
Université Paul Sabatier, Toulouse

Received 26 September 2008, revised 26 January 2009, accepted ???
Published online ???

Key words Preference representation, computational complexity, computational social choice.

We analyze the expressivity, succinctness, and complexity of a family of languages based on weighted propositional formulas for the representation of utility functions. The central idea underlying this form of preference modeling is to associate numerical weights with goals specified in terms of propositional formulas, and to compute the utility value of an alternative as the sum of the weights of the goals it satisfies. We define a large number of representation languages based on this idea, each characterized by a set of restrictions on the syntax of formulas and the range of weights. Our aims are threefold. First, for each language we try to identify the class of utility functions it can express. Second, when different languages can express the same class of utility functions, one may allow for a more succinct representation than another. Therefore, we analyze the relative succinctness of languages. Third, for each language we study the computational complexity of the problem of finding the most preferred alternative given a utility function expressed in that language.

1 Introduction

Preferences play an important role in many areas. Whenever someone needs to make a decision or choose between several alternatives, their choices will be guided by their preferences over the alternatives available to them. Similarly, when a group needs to make a decision, that decision should be informed by a suitable aggregation of the individual preferences of its members. Preferences are therefore at the core of *social choice theory* [1], which studies mechanisms for collective decision making; and computational aspects of preference representation are central to the field of *computational social choice* [2].

We can distinguish *ordinal* from *cardinal* preferences. An ordinal preference is a binary relation over the domain of alternatives, which is typically assumed to be transitive and complete, although this need not be the case. Such an ordinal preference relation allows us to check whether one alternative is "better" than another. Cardinal preferences are *utility functions* mapping alternatives to a suitable scale, often the reals. When the number of alternatives that we need to rank or over which we need to define a utility function is relatively small, then the choice of *language* for representing preferences is not crucial. However, the set of alternatives often has a combinatorial structure. For instance, if we need to decide on an allocation of (indivisible) goods to agents, then each agent will have preferences over the alternative subsets they may receive, the number of which is exponential in the number of goods. Similarly, when we elect the members of a committee, the number of alternative constellations is exponential in the number of seats available. In general, a *combinatorial domain* is the

[*] This paper is based on and extends work presented at the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-2006) and the AAAI-2007 Workshop on Preference Handling for Artificial Intelligence. The work of Joel Uckelman was supported by a GLoRiClass fellowship funded by the European Commission (Early Stage Research Training Mono-Host Fellowship MEST-CT-2005-020841). The work of Yann Chevaleyre and Jérôme Lang was partially supported by the ANR Project ANR-05-BLAN-0384 "Preference Handling and Aggregation in Combinatorial Domains". The work of Ulle Endriss was partially supported by the NWO Vidi Project 639.022.706 on "Collective Decision Making in Combinatorial Domains". The authors would like to thank the anonymous reviewers for their numerous helpful suggestions.

Cartesian product of several finite domains (often binary domains). Choosing an alternative from a combinatorial domain means choosing a value for each dimension of the domain. It is these kinds of situations in which we require languages that allow for a compact representation of preferences.

Several such languages have been developed and analyzed in the literature, both for the representation of utility functions and for the representation of ordinal preference relations. Languages for the succinct representation of ordinal preferences include formalisms for making *ceteris paribus* statements, which range from very expressive languages [3] to syntactical restrictions such as CP-nets [4], where a weaker expressivity is compensated for by the availability of efficient elicitation and optimization techniques. They also include languages based on conditional logics, prioritized logics, and prioritized constraint satisfaction problems (see e.g., Lang [5] for an overview). Languages for the succinct representation of utility functions include graphical models [6, 7, 8, 9], decision trees [10], valued constraint satisfaction problems [11], and bidding languages for combinatorial auctions [12, 13, 14].

In this paper we analyze languages for the representation of utility functions based on weighted propositional formulas, or *weighted goals* for short, an approach which originates in penalty logic [5, 15]. This family of languages is suitable for modeling preferences over combinatorial domains that are the Cartesian product of several binary domains, each represented by a propositional variable. The central idea is to associate numerical weights with goals specified in terms of propositional formulas, and to compute the utility value of an alternative as a function of the weights of the goals it satisfies. For example, the weighted goal $(p \wedge q, 4)$ expresses that our decision maker ascribes a value of 4 to making both p and q true. Similarly, $(\neg r, 7)$ means that making r false has value 7. There are several different ways in which we could aggregate the weights of the goals satisfied by an alternative to compute the utility of that alternative. In this paper we concentrate on what is arguably the most natural choice for such an aggregation operator, namely summation, though maximization is also a viable alternative [16, 17]. We note here that weighted goals can also be used to model ordinal preferences, for instance by interpreting weights as indicators for the relative importance of goals and by ranking alternatives in terms of the most important goal they violate [5, 18]. However, in the present paper we are only interested in modeling utility functions.

Weighted goals are a framework for defining an entire *family of languages*. The choice of aggregator is one choice that we need to make. A further choice concerns the range of formulas that we wish to admit as goals. For instance, we may not want to allow for negation or we may only permit formulas that do not exceed a certain length. Finally, we could impose restrictions on the range of weights to be used. For instance, we may allow weights to be drawn from the reals, the positive reals, or even just very small sets such as $\{0, 1\}$. Each choice of aggregator, restriction on formulas, and restriction on weights gives rise to a different language, and we can study and compare their properties.

In this paper we investigate three important properties of such languages: expressivity, succinctness, and complexity. We briefly introduce each of these here:

- *Expressivity:* Can our language of choice represent all functions belonging to a given class of utility functions which interests us? Not all languages are equally expressive and not all applications require full expressivity. Excess expressivity is often undesirable, because highly expressive languages tend to be computationally more demanding to reason about. We are interested in *correspondence results* between languages and classes of functions. For instance, a very simple result which we present shows that the language we obtain by restricting formulas to *literals* can express all *modular* utility functions, and only those. An interesting property closely related to expressivity is *uniqueness* of representation. A language has the uniqueness property with respect to a given class of utility functions if it has no more than one way of representing any function from that class. Syntactically rich languages often lack the uniqueness property, which may be considered wasteful.

- *Succinctness:* How much space do we require to encode a given utility function in a given language? If two languages can both express all functions from a given class of utility functions of interest, but one can do so using less space than the other, then the former language may be preferred. In fact, this definition is too restrictive. We will call language $\mathcal{L}$ at least as *succinct* as language $\mathcal{L}'$ if any utility function expressible in both languages can be expressed in $\mathcal{L}$ without a significant (that is, super-polynomial) increase in size over its representation in $\mathcal{L}'$. If there is at least one family of functions for which the best representation in $\mathcal{L}'$ is exponentially larger than it is in $\mathcal{L}$, then $\mathcal{L}$ is strictly more succinct than $\mathcal{L}'$. If $\mathcal{L}'$ has the uniqueness property, then this kind of result is particularly easy to prove, as it is sufficient to provide a concrete representation in $\mathcal{L}'$.

- *Complexity:* What is the computational complexity of certain tasks, such as comparing two alternatives or finding a most preferred alternative, when preferences are expressed in a given language? We will define the MAX-UTIL problem, the decision problem of checking whether there exists an alternative that exceeds a particular given utility level K (this problem is of course closely related to the problem of finding an alternative that maximizes utility, hence the name). As we shall see, the complexity of MAX-UTIL ranges from linear to NP-complete, depending on the language.

Other important properties of preference representation languages, which we shall not address in this paper, are *cognitive relevance* and *elicitation friendliness*. The former concerns the proximity of the formal representation language to the way in which humans represent their preferences. Elicitation is the process of constructing a representation of a preference structure by querying a decision maker, and some languages may fare better than others in this respect.

The remainder of this paper is organized as follows: Section 2 introduces the framework of weighted goals for the representation of utility functions. We also briefly discuss alternative languages, in particular the family of so-called OR/XOR bidding languages developed in the area of combinatorial auctions. Each of the subsequent sections is devoted to a property of representation languages. Section 3 takes up the question of which utility functions are representable in any given language. In Section 4, we compare languages as to their space efficiency, and in Section 5 we consider the complexity of answering some queries on goalbases.

2 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 briefly discuss related work. (More related work is reviewed in the conclusion.)

2.1 Basic Definitions and Notation

We are interested in utility functions declared over combinatorial domains that are the Cartesian product of several binary domains. A generic representation of this kind of domain is the set of all possible models for propositional formulas over a fixed language with a finite number of propositional variables (the dimensionality of the combinatorial domain).

Definition 2.1 (Utility Functions and Models) A utility function is a mapping $u : 2^{\mathcal{PS}} \to \mathbb{R}$, where $\mathcal{PS}$ is a fixed, finite set of propositional variables. A *model* is a set $M \in 2^{\mathcal{PS}}$. We write $\mathcal{PS}_n$ to indicate that $|\mathcal{PS}| = n$.

Let $\mathcal{L}_{\mathcal{PS}}$ be the language of propositional logic over $\mathcal{PS}$. In principle, we may allow any kind of propositional connective. The technical results in this paper apply to formulas that contain only the connectives $\neg$, $\wedge$, and $\vee$. We comment briefly on the use of additional connectives in the conclusion.

Definition 2.2 (Weighted Goals and Goalbases) A *weighted goal* is a pair (φ, w), where φ is a formula in the language $\mathcal{L}_{\mathcal{PS}}$ and $w \in \mathbb{R}$. A *goalbase* is a finite multiset $G = \{(\varphi_i, w_i)\}_i$ of weighted goals.

Goals are typically required to be satisfiable formulas, but for the languages studied in this paper this does not affect expressive power (whether unsatisfiable formulas are allowed could potentially affect complexity, and we comment on this issue in Section 5). When a particular goalbase is under consideration, we write w_φ to mean the weight of formula φ in that goalbase. $\mathrm{For}(G)$ is the set of formulas in G.

Definition 2.3 (Generated Utility Functions) A goalbase G and an aggregation function $F : 2^{\mathbb{R}} \to \mathbb{R}$ generate a utility function $u_{G,F}$ mapping each model $M \subseteq \mathcal{PS}$ to $u_{G,F}(M) = F(\{w : (\varphi, w) \in G \text{ and } M \models \varphi\})$.

In this paper, we restrict ourselves to $F = \Sigma$, the summation function, and often omit F hereafter. In particular, this means that

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

which is to say that the value of a model is the sum of weights of formulas made true in that model. For example, if $\mathcal{PS} = \{p, q, r\}$, then the goalbase $G_1 = \{(p \vee q \vee r, 2), (p \wedge q, 1), (p \wedge r, 1), (q \wedge r, 1), (p \wedge q \wedge r, -2)\}$ generates the utility function $u : X \mapsto \min(3, 2 \cdot |X|)$.

Definition 2.4 (Goalbase Equivalence) Two goalbases G and G' are equivalent with respect to an aggregation function F (written $G \equiv_F G'$) iff they define the same utility function. That is, $G \equiv_F G'$ iff $u_{G,F} = u_{G',F}$.

Goalbases provide a framework for defining different languages for representing utility functions. 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 the aggregator F gives rise to a different language. An interesting question then is whether there are natural goalbase languages (defined in terms of natural restrictions) such that the utility functions they generate enjoy simple structural properties. (This is indeed the case, as seen in Section 3.3.)

Definition 2.5 (Languages and Classes of Utility Functions) Let $\Phi \subseteq \mathcal{L}_{\mathcal{PS}}$ be a set of formulas, $W \subseteq \mathbb{R}$ a set of weights, and F an aggregation function. Then $\mathcal{L}(\Phi, W, F)$ is the set of goalbases formed by formulas in Φ with weights from W to be aggregated by F, and $\mathcal{U}(\Phi, W, F)$ is the class of utility functions generated by goalbases belonging to $\mathcal{L}(\Phi, W, F)$. More generally, we write $\mathcal{U}(\mathcal{L})$ to mean the class of utility functions generated by goalbases in the language $\mathcal{L}$.

As mentioned before, we treat the case where $F = \Sigma$ in this paper, and hereafter omit F from our notation where it causes no ambiguity.

Regarding weights, we will study the restriction to the positive reals ($\mathbb{R}^+$) as well the general case ($\mathbb{R}$). For complexity questions we will restrict attention to the rationals ($\mathbb{Q}$). The next definition summarizes the types of restrictions we consider for formulas.

Definition 2.6 (Types of Formulas) We define the following types of formulas:

- An *atom* is a member of $\mathcal{PS}$.
- A *literal* is an atom or its negation.
- A *clause* is a disjunction of literals.
- A *cube* is a conjunction of literals.
- A *positive* X is a satisfiable formula of type X that contains no negations.
- A *strictly positive* X is a non-tautologous positive X.
- A *k*-X is an X with at most k occurrences of atoms.

When discussing positive clauses, positive cubes, and positive formulas, we frequently abbreviate these to *pclauses*, *pcubes*, and *pforms*, respectively. Additionally, we call strictly positive cubes and strictly positive formulas *spcubes* and *spforms*, respectively. (The term *spclauses* is redundant because every positive clause is falsifiable.) Atoms are 1-spclauses, 1-spcubes, and 1-spformulas (and also 1-pclauses, 1-pcubes, and 1-pformulas), while literals are 1-clauses, 1-cubes, and 1-formulas. Clauses, cubes, and formulas are ω-clauses, ω-cubes, and ω-formulas, respectively, which is to say that the formulas may be of any finite length.[1] Note that $\bigwedge \emptyset = \top$ and $\bigvee \emptyset = \bot$, from which follows that $\top$ is the unique 0-pcube and $\bot$ the unique 0-clause. The notation $X + \top$ indicates the set of formulas $X \cup \{\top\}$ (e.g., pclauses $+\top$ is the set containing all pclauses along with $\top$).

2.2 Related Languages

Goalbase languages and variations thereof have been considered in many places and used for a number of applications, sometimes under a different name [5, 13, 15, 16, 18, 19, 20, 21, 22].

Boutilier and Hoos [13], for instance, suggest a variant of $\mathcal{L}(\textit{pforms}, \mathbb{R}^+, \Sigma)$ as a means for communicating bids in a combinatorial auction. There are two differences between their language and $\mathcal{L}(\textit{pforms}, \mathbb{R}^+, \Sigma)$. First, they also allow for the logic connective XOR, which we do not consider here. Including additional connectives can improve succinctness, and 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 [13] also allow for weights to be assigned to subformulas of goals. The utility of a set of goods 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.

[1] Strictly speaking, we should write, e.g., $(< \omega)$-cubes instead of ω-cubes, but we abuse notation for the sake of brevity and because all formulas are assumed to be finite in length.

Another important group of languages for modeling cardinal preferences are the OR/XOR family of bidding languages for combinatorial auctions [12]. 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. The OR language is often the basis for bidding languages used in practice. A problematic feature of this language is that even the basic *evaluation problem*—the problem of computing the value assigned to a given set of goods according to the bid expression of a single bidder—is NP-hard. In contrast to this, the evaluation problem clearly is easy for any of our goalbase languages: given a model X and a goalbase G, computing $u_G(X)$ only requires one model checking operation for each goal in G.

In the XOR language [12, 14], 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 $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max)$ in our framework. A disadvantage of this language is that it is not concise for most interesting classes of valuation functions. Finally, combinations of XOR and OR (such as OR-of-XORs) have also been considered in the literature.

3 Expressivity

The language restrictions introduced in the previous section allow us to define a host of different languages. In this section we study the expressivity of these languages. We first state some simple equivalences between goalbases. We then define the notion of a language having unique representations and establish two uniqueness results. Finally, we prove a number of correspondence results that allow us to (almost) fully classify the languages identified with respect to the classes of utility functions they define.

3.1 Equivalences

Recall that two goalbases G and G' are *equivalent* ($G \equiv_\Sigma G'$) if they generate the same utility function (i.e., if $u_G = u_{G'}$). The following equivalences are, for convenience, stated as they are used later on, and not necessarily in their most general form.

Fact 3.1 *Given a goalbase G, formulas $\varphi, \varphi_1, \ldots, \varphi_k, \psi, \chi$, and weight $w \in \mathbb{R}$, the following equivalences hold:*

$$
\begin{aligned}
G \cup \{(\varphi \wedge \neg\psi, w)\} &\equiv G \cup \{(\varphi, w), (\varphi \wedge \psi, -w)\} && (1)\\
G \cup \{(\varphi \vee \neg\psi, w)\} &\equiv G \cup \{(\top, w), (\varphi, w), (\varphi \vee \psi, -w)\} && (2)\\
G \cup \{(\varphi \wedge (\psi \vee \chi), w)\} &\equiv G \cup \{(\varphi \wedge \psi, w), (\varphi \wedge \chi, w), (\varphi \wedge \psi \wedge \chi, -w)\} && (3)\\
G \cup \{(\varphi \vee (\psi \wedge \chi), w)\} &\equiv G \cup \{(\varphi \vee \psi, w), (\varphi \vee \chi, w), (\varphi \vee \psi \vee \chi, -w)\} && (4)\\
G \cup \{(\varphi_1 \wedge \cdots \wedge \varphi_k, w)\} &\equiv G \cup \{(\neg\varphi_1 \vee \cdots \vee \neg\varphi_k, -w), (\psi, w), (\neg\psi, w)\} && (5)\\
G \cup \{(\varphi_1 \vee \cdots \vee \varphi_k, w)\} &\equiv G \cup \{(\neg\varphi_1 \wedge \cdots \wedge \neg\varphi_k, -w), (\psi, w), (\neg\psi, w)\} && (6)\\
G \cup \{(\top, w)\} &\equiv G \cup \{(\varphi, w), (\neg\varphi, w)\} && (7)
\end{aligned}
$$

All of the above are easily verified by considering all possible combinations of truth values for φ, ψ, χ.

3.2 Uniqueness

Some languages have a unique way of expressing a given utility function, while others allow for several alternative representations. The next definition makes this notion of uniqueness precise.

Definition 3.1 A utility function u is *represented* in a language $\mathcal{L}$ if there exists a goalbase $G \in \mathcal{L}$ such that $u = u_G$. A utility function u is *uniquely represented* (modulo formula equivalence) in a language $\mathcal{L}$ if, given a set of formulas Φ containing one representative formula for each formula equivalence class in $\mathcal{L}$ (except $\bot$), there is a unique goalbase G such that $\mathrm{For}(G) = \Phi$ and $u_G = u$. (Note that some weights in G may be zero.) A language $\mathcal{L}$ is said to have *unique representations* if every u represented in $\mathcal{L}$ is uniquely represented.

$\mathcal{L}(\textit{cubes}, \mathbb{R})$, for instance, does not have unique representations: The two goalbases $\{(\top, 3), (p, 2)\}$ and $\{(p \wedge q, 5), (p \wedge \neg q, 5), (\neg p \wedge q, 3), (\neg p \wedge \neg q, 3)\}$ both define the same utility function (for $\mathcal{PS} = \{p, q\}$).

Theorem 3.2 *$\mathcal{L}(\textit{pcubes}, \mathbb{R})$ has unique representations.*

P r o o f. Given any utility function u, the weight of each positive cube is uniquely determined: We must have $w_\top = u(\emptyset)$, because $\top$ is the only positive cube satisfied by $\emptyset$, and furthermore $w_{\bigwedge X} = u(X) - \sum_{Y \subset X} w_{\bigwedge Y}$ for any nonempty set X. □

The recursive definition of the weights given in the proof of Theorem 3.2 can be turned into a direct rule for computing weights by using the so-called *Möbius inversion* [23, 24]:

$$w_{\bigwedge X} \quad = \quad \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} \cdot u(Y) \tag{8}$$

Theorem 3.3 *$\mathcal{L}(\textit{pclauses}, \mathbb{R})$ has unique representations.*

P r o o f. Let u be any utility function represented by positive clauses with weights $w_{\bigvee X}$ (with nonempty sets $X \subseteq \mathcal{PS}$). Then for any $Y \subseteq \mathcal{PS}$, $u(Y)$ must be equal to the sum of the weights $w_{\bigvee X}$ for which X and Y have a nonempty intersection:

$$u(Y) \quad = \sum_{X \cap Y \neq \emptyset} w_{\bigvee X} \quad = \sum_{\emptyset \subset X \subseteq \mathcal{PS}} w_{\bigvee X} - \sum_{\emptyset \subset X \subseteq \mathcal{PS} \setminus Y} w_{\bigvee X} \quad = \quad u(\mathcal{PS}) - \sum_{\emptyset \subset X \subseteq \mathcal{PS} \setminus Y} w_{\bigvee X}$$

This shows that each weight $w_{\bigvee X}$ is uniquely determined: For singletons $X = \{p\}$, by setting $Y = \mathcal{PS} \setminus \{p\}$ in above equation, we obtain $w_p = u(\mathcal{PS}) - u(\mathcal{PS} \setminus \{p\})$. For general sets X, using $Y = \mathcal{PS} \setminus X$, we then obtain $w_{\bigvee X} = u(\mathcal{PS}) - u(\mathcal{PS} \setminus X) - \sum_{\emptyset \subset X' \subset X} w_{\bigvee X'}$. □

Now, unraveling the recursive definition of the weights given in the proof above, we can also provide a direct rule for computing weights in $\mathcal{L}(\textit{pclauses}, \mathbb{R})$, similar to the Möbius inversion:

$$w_{\bigvee X} \quad = \quad \sum_{Y \subseteq X} (-1)^{|X \setminus Y|+1} \cdot u(\mathcal{PS} \setminus Y) \tag{9}$$

Furthermore, we have the following corollary because no positive clause is a tautology:

Corollary 3.4 *$\mathcal{L}(\textit{pclauses} + \top, \mathbb{R})$ has unique representations.*

3.3 Correspondences

We now address the following question: What class of utility functions can we model using a given language? As much as possible we will strive for exact characterization results that establish the correspondence between a natural goalbase language and a commonly used class of utility functions.

An important class of utility functions are the *k-additive functions* [24]. Let $\mathcal{PS}(k)$ be the set of all subsets of $\mathcal{PS}$ with at most k elements. A utility function u is k-additive if there exists a mapping $m : \mathcal{PS}(k) \to \mathbb{R}$ such that $u(X) = \sum\{m(Y) : Y \subseteq X \text{ and } Y \in \mathcal{PS}(k)\}$ for each set $X \subseteq \mathcal{PS}$. The concept of k-additivity has been applied in various places, including fuzzy measure theory [24], combinatorial auctions [25], and distributed multiagent resource allocation [26]. For example, if the variables in $\mathcal{PS}$ are used to model whether an agent owns certain resources, then k-additive utility functions naturally model situations where synergies among different resources are restricted to bundles of at most k elements.

For the next proof, and indeed much of the paper, we will make frequent use of the fact that whenever $\Phi \subseteq \Phi'$ and $W \subseteq W'$, then $\mathcal{U}(\Phi, W) \subseteq \mathcal{U}(\Phi', W')$.

Theorem 3.5 *$\mathcal{U}(k\text{-pcubes}, \mathbb{R})$, $\mathcal{U}(k\text{-cubes}, \mathbb{R})$, $\mathcal{U}(k\text{-pclauses} + \top, \mathbb{R})$, $\mathcal{U}(k\text{-clauses}, \mathbb{R})$, $\mathcal{U}(k\text{-pforms}, \mathbb{R})$, and $\mathcal{U}(k\text{-forms}, \mathbb{R})$ are equal to the class of all k-additive utility functions.*

P r o o f. Inspection of the definition of k-additivity shows that it is simply a notational variant of the language based on positive cubes of length $\leq k$. That is, $\mathcal{U}(k\text{-}pcubes, \mathbb{R})$ is the class of all k-additive utility functions. By language inclusion, we have the following:

$$\begin{array}{ccccc} \mathcal{U}(k\text{-}cubes, \mathbb{R}) & \subseteq & \mathcal{U}(k\text{-}forms, \mathbb{R}) & \supseteq & \mathcal{U}(k\text{-}clauses, \mathbb{R}) \\ \cup\!\mid & & \cup\!\mid & & \cup\!\mid \\ \mathcal{U}(k\text{-}pcubes, \mathbb{R}) & \subseteq & \mathcal{U}(k\text{-}pforms, \mathbb{R}) & \supseteq & \mathcal{U}(k\text{-}pclauses\ + \top, \mathbb{R}) \end{array}$$

Taken together, equivalences (1) and (3) from Fact 3.1 can be used to transform any goalbase in $\mathcal{L}(k\text{-}forms, \mathbb{R})$ into an equivalent goalbase in $\mathcal{L}(k\text{-}pcubes, \mathbb{R})$. (Use (3) to eliminate all disjunctions, then (1) to eliminate all negations. While these equivalences add more formulas, they never add *longer* formulas.) Thus, $\mathcal{U}(k\text{-}pcubes, \mathbb{R}) = \mathcal{U}(k\text{-}forms, \mathbb{R})$.

Equivalences (2) and (4) from Fact 3.1 can be used to transform any goalbase in $\mathcal{L}(k\text{-}forms, \mathbb{R})$ into an equivalent goalbase in $\mathcal{L}(k\text{-}pclauses\ + \top, \mathbb{R})$. (Use (4) to eliminate all conjunctions, then (2) to eliminate all negations.) Thus, $\mathcal{U}(k\text{-}pclauses, \mathbb{R}) = \mathcal{U}(k\text{-}forms, \mathbb{R})$. In summary, we see that each of the six classes must be equal to the class of k-additive functions. □

The next lemma clarifies the effect that the ability to express tautologies in a language has on the class of utility functions that can be defined. Roughly speaking, any utility function u expressible in a language based on strictly positive formulas must be *normalized*, i.e., will satisfy $u(\emptyset) = 0$.

The (affine) translation t_c of a utility function u is the map such that $t_c(u(X)) = u(X) + c$ for all $X \subseteq \mathcal{PS}$. A property P is *invariant under translation* if, for all utility functions u, $c \in \mathbb{R}$, and $X \subseteq \mathcal{PS}$, u has property P iff $t_c(u)$ has property P.

Lemma 3.6 *Fix Φ as a strictly positive set of formulas and P a property of utility functions which is invariant under translation. Then $\mathcal{U}(\Phi, W)$ is the class of normalized utility functions with property P iff $\mathcal{U}(\Phi \cup \{\top\}, W)$ is the class of utility functions with property P.*

P r o o f. ($\Rightarrow$) Suppose that $\mathcal{U}(\Phi, W)$ is the class of normalized utility functions with property P. First, we show that every $u_G \in \mathcal{U}(\Phi \cup \{\top\}, W)$ has property P: Fix $u_G \in \mathcal{U}(\Phi \cup \{\top\}, W)$. $G \setminus \{(\top, w) : w \in W\} \in \mathcal{L}(\Phi, W)$ and so $u_{G \setminus \{(\top, w): w \in W\}}$ has property P by hypothesis. $u_G = t_w(u_{G \setminus \{(\top, w): w \in W\}})$ and so by invariance u_G has property P.

Next, we show that every u with property P is in $\mathcal{U}(\Phi \cup \{\top\}, W)$: Fix u with property P. $t_{u(\emptyset)}(u)$ is normalized and has property P by invariance, so $t_{u(\emptyset)}(u) \in \mathcal{U}(\Phi, W)$ by hypothesis. Let G represent $t_{u(\emptyset)}(u)$ in $\mathcal{L}(\Phi, W)$. Then $G \cup \{(\top, u(\emptyset))\} \in \mathcal{L}(\Phi \cup \{\top\}, W)$ and $u_{G \cup \{(\top, u(\emptyset))\}} = u$.

($\Leftarrow$) Suppose that $\mathcal{U}(\Phi \cup \{\top\}, W)$ is the class of utility functions with property P. First, we show that every $u_G \in \mathcal{U}(\Phi, W)$ is normalized and has property P: Fix $u_G \in \mathcal{U}(\Phi, W)$. Normalization follows due to Φ being strictly positive. $G \cup \{(\top, w)\} \in \mathcal{L}(\Phi \cup \{\top\}, W)$ for any $w \in W$, and by hypothesis $u_{G \cup \{(\top, w)\}}$ has property P. $u_{G \cup \{(\top, w)\}} = t_w(u_G)$, and so by invariance has property P.

Next, we show that every normalized u with property P is in $\mathcal{U}(\Phi, W)$: Fix u normalized and with property P. For any $w \in W$, $t_w(u)$ has property P by invariance and so is in $\mathcal{U}(\Phi \cup \{\top\}, W)$. Let G represent $t_w(u)$ in $\mathcal{L}(\Phi \cup \{\top\}, W)$. Since Φ is strictly positive, $(\top, w) \in G$. Then $u_{G \setminus \{(\top, w)\}} = t_w^{-1}(t_w(u)) = u$, and so $u \in \mathcal{U}(\Phi \cup \{\top\}, W)$. □

Next we explore the class of k-additive utility functions for specific values of k. It is a well-known fact that *any* utility function is k-additive for some $k \in \mathbb{N}$ (certainly for $k = |\mathcal{PS}|$). (This is why we refer to general functions as ω-additive.) Our next result is therefore an immediate corollary of Theorem 3.5.

Corollary 3.7 *$\mathcal{U}(pcubes, \mathbb{R})$, $\mathcal{U}(cubes, \mathbb{R})$, $\mathcal{U}(pclauses\ + \top, \mathbb{R})$, $\mathcal{U}(clauses, \mathbb{R})$, $\mathcal{U}(pforms, \mathbb{R})$, and $\mathcal{U}(form, \mathbb{R})$ are equal to the class of all utility functions.*

Another special case of interest is the class of 1-additive utility functions, better known as the *modular* utility functions. An alternative way of characterizing this class is to define a utility function u as modular if $u(X \cup Y) = u(X) + u(Y) - u(X \cap Y)$ for all sets $X, Y \subseteq \mathcal{PS}$. Modular utility functions are very simple and have limited expressive power. Nevertheless they are frequently used in applications, e.g., in work on modeling negotiation between autonomous software agents [27].

Corollary 3.8 *$\mathcal{U}(literals, \mathbb{R})$ and $\mathcal{U}(atoms + \top, \mathbb{R})$ are the class of all modular utility functions, and $\mathcal{U}(atoms, \mathbb{R})$ is the class of all normalized modular utility functions.*

Proof. The set of 1-pcubes is equal to the set of atoms together with $\top$. Therefore, by Theorem 3.5, $\mathcal{U}(atoms + \top, \mathbb{R})$ is the class of all modular functions. The class of 1-cubes is equal to the class of literals together with $\top$. But by equivalence (7) of Fact 3.1, literals alone have the same expressive power as literals together with $\top$. Hence, again by Theorem 3.5, $\mathcal{U}(literals, \mathbb{R})$ is the class of all modular functions. The fact that $\mathcal{U}(atoms, \mathbb{R})$ is the class of all normalized modular utility functions follows from Lemma 3.6. □

For the remainder of this section we consider languages where the set of weights is restricted to the positive reals. Clearly, the utility functions that can be so expressed will be *nonnegative*, i.e., $u(X) \geq 0$ for all $X \subseteq \mathcal{PS}$. The question is whether we can express *all* nonnegative utility functions in this manner.

Theorem 3.9 *$\mathcal{U}(cubes, \mathbb{R}^+)$ and $\mathcal{U}(form, \mathbb{R}^+)$ are the class of all nonnegative utility functions.*

Proof. Clearly every $u \in \mathcal{U}(form, \mathbb{R}^+)$ (and hence also every $u \in \mathcal{U}(cubes, \mathbb{R}^+)$) is nonnegative. For the converse, suppose that u is nonnegative. Then define

$$G = \Big\{ \Big(\bigwedge M \wedge \bigwedge \{\neg p : p \in \mathcal{PS} \setminus M\}, u(M) \Big) : M \subseteq \mathcal{PS} \text{ and } u(M) \neq 0 \Big\}$$

and observe that $u_G = u$ and that G contains only positively-weighted cubes. □

That is, general formulas as well as cubes are fully expressive over nonnegative utility functions when weights are required to be positive. As we shall see next, the same is not true for clauses.

Theorem 3.10 *$\mathcal{U}(clauses, \mathbb{R}^+)$ is a proper subset of all nonnegative utility functions.*

Proof. $\mathcal{L}(clauses, \mathbb{R}^+) \subset \mathcal{L}(form, \mathbb{R}^+)$, so by Theorem 3.9, $\mathcal{U}(clauses, \mathbb{R}^+)$ contains only nonnegative utility functions. The utility function u over $\mathcal{PS} = \{p, q\}$ with $u(\{p, q\}) = 1$ and $u(X) = 0$ for any $X \neq \{p, q\}$ demonstrates that the inclusion is strict. The following five constraints must be satisfied for there to be a $G \in \mathcal{L}(clauses, \mathbb{R}^+)$ which represents u:

$$w_p + w_q + w_{p\vee q} + w_{\neg p\vee q} + w_{p\vee\neg q} + w_{p\vee\neg p} = 1 \tag{10}$$
$$w_p + w_{\neg q} + w_{p\vee q} + w_{p\vee\neg q} + w_{\neg p\vee\neg q} + w_{p\vee\neg p} = 0 \tag{11}$$
$$w_{\neg p} + w_q + w_{p\vee q} + w_{\neg p\vee q} + w_{\neg p\vee\neg q} + w_{p\vee\neg p} = 0 \tag{12}$$
$$w_{\neg p} + w_{\neg q} + w_{\neg p\vee q} + w_{p\vee\neg q} + w_{\neg p\vee\neg q} + w_{p\vee\neg p} = 0 \tag{13}$$
$$w_\varphi \geq 0 \text{ for all clauses } \varphi \tag{14}$$

Together, constraints (11), (12), (13), and (14) force $w_\varphi = 0$ for every clause φ, contradicting (10). □

$\mathcal{L}(clauses, \mathbb{R}^+)$ seems not to characterize a natural class of functions. For (strictly) positive formulas with positive weights, on the other hand, we do obtain nice correspondences. A utility function u is *monotone* if, for all $X, Y \subseteq \mathcal{PS}$, $u(X) \leq u(Y)$ whenever $X \subseteq Y$.

Theorem 3.11 *$\mathcal{U}(spforms, \mathbb{R}^+)$ is the class of all normalized monotone utility functions.*

Proof. No strictly positive formula is a tautology, so every $u \in \mathcal{U}(spforms, \mathbb{R}^+)$ is normalized, and because all weights and formulas are positive, it is also monotone.

For the converse: Let u be an arbitrary normalized monotone utility function. We construct a $G \in \mathcal{L}(spforms, \mathbb{R}^+)$ for which $u_G = u$ as follows. Define a sequence of utility functions $u_1, \ldots, u_n$ so that

$$u_k(X) = \max\{u(X') : X' \subseteq X \text{ and } |X'| \leq k\}.$$

In this way, $u_1 = \max_{a \in X} u(\{a\})$ and $u_n = u$. Additionally, we define $u_0(X) = 0$ for all X, for convenience. Observe that we can use the these u_i to decompose u such that $u = \sum_{k=1}^{n}(u_k - u_{k-1})$, and so if we can construct a goalbase for each $u_k - u_{k-1}$, then the union of those goalbases will be a goalbase for u. Hereafter, we will abbreviate $u_k - u_{k-1}$ to u^*_k. To construct G_k, a goalbase for u^*_k, let $X_0 = \emptyset$ and $\langle X_1, \dots, X_{\binom{n}{k}} \rangle$ be the set of size-k subsets of $\mathcal{PS}$, ordered so that $u^*_k(X_i) \le u^*_k(X_j)$ for $i < j$. Then let

$$G_k = \left\{ \Big(\bigvee_{j=i}^{\binom{n}{k}} \bigwedge X_j, u^*_k(X_i) - u^*_k(X_{i-1}) \Big) : 1 \le i \le \tbinom{n}{k} \right\}$$

from which it can easily, though tediously, be checked that $u_{G_k} = u^*_k$. (For example, if $\mathcal{PS} = \{a, b, c\}$ and $u(a) \le u(b) \le u(c)$, then $G_1 = \{(a \vee b \vee c, u(a)), (b \vee c, u(b) - u(a)), (c, u(c) - u(b))\}$. View items a, b, and c as substitutes, but with b conferring a bonus over a, and c a further bonus over b. This is the structure which can be seen in G_1. Higher-order G_is capture this same idea, but for sets of items larger than singletons.)

Finally, let $G = \bigcup_{k=1}^n G_k$. Now $u_G = u$, since for each k, $u_{G_k} = u^*_k$ and $\sum_{k=1}^n u^*_k = u_n = u$. Finally, observe that every formula in G is strictly positive; and all the weights $u^*_k(X_i) - u^*_k(X_{i-1})$ are nonnegative by virtue of the ordering declared over the X_i. □

Note that in the preceding theorem, we could also add *nonnegative* as a property, because normalization and monotonicity together imply nonnegativity. An application of Lemma 3.6 yields the following corollary.

Corollary 3.12 *$\mathcal{U}$(pforms, $\mathbb{R}^+$) is the class of all nonnegative monotone utility functions.*

A utility function u is *supermodular* if $u(X \cup Y) \ge u(X) + u(Y) - u(X \cap Y)$ for all $X, Y \subseteq \mathcal{PS}$. Supermodularity (and its counterpart, submodularity, defined below) are widely used concepts in the economics literature [28]. Supermodularity seems not to correspond directly to a natural goalbase language, but we can characterize a large subclass.

Theorem 3.13 *$\mathcal{U}$(pcubes, $\mathbb{R}^+$) is the class of all nonnegative utility functions satisfying the constraint $\sum_{Y \subseteq X} (-1)^{|X \setminus Y|} \cdot u(Y) \ge 0$ for all $X \subseteq \mathcal{PS}$.*

Proof. That $\mathcal{U}$(*pcubes*, $\mathbb{R}^+$) is the class of all nonnegative utility functions satisfying $\sum_{Y \subseteq X} (-1)^{|X \setminus Y|} \cdot u(Y) \ge 0$ immediately follows from the fact that the weight of any positive cube is determined by the Möbius inversion as stated in equation (8). □

Note that the property corresponding to $\mathcal{U}$(*pcubes*, $\mathbb{R}^+$) implies nonnegativity, monotonicity, and supermodularity. Nonnegativity and monotonicity follow from Corollary 3.12. For supermodularity, suppose that $G \in \mathcal{L}$(*pcubes*, $\mathbb{R}^+$). Then

$$\begin{aligned} u_G(X \cup Y) &= \sum_{Z \subseteq X \cup Y} w_{\bigwedge Z} = \sum_{Z \subseteq X} w_{\bigwedge Z} + \sum_{Z \subseteq Y} w_{\bigwedge Z} - \sum_{Z \subseteq X \cap Y} w_{\bigwedge Z} + \sum_{\substack{Z \subseteq X \cup Y \\ Z \not\subseteq X, Y}} w_{\bigwedge Z} \\ &\ge \sum_{Z \subseteq X} w_{\bigwedge X} + \sum_{Z \subseteq Y} w_{\bigwedge Z} - \sum_{Z \subseteq X \cap Y} w_{\bigwedge Z} = u_G(X) + u_G(Y) - u_G(X \cap Y), \end{aligned}$$

which is equivalent to the supermodularity condition.

The utility function $u : X \mapsto \max(1, |X|)$ shows that there are supermodular utility functions that are not in $\mathcal{U}$(*pcubes*, $\mathbb{R}^+$). As can easily be checked, if $\mathcal{PS} = \{p, q, r\}$, then expressing u in terms of positive cubes requires the use of a negative weight: $w_{p \wedge q \wedge r} = -1$. The previous theorem holds also for $\mathcal{U}$(*spcubes*, $\mathbb{R}^+$) if "nonnegative" is replaced with "normalized".

The dual of the supermodularity property is submodularity. A utility function u is *submodular* if $u(X \cup Y) \le u(X) + u(Y) - u(X \cap Y)$ for all $X, Y \subseteq \mathcal{PS}$.

Theorem 3.14 *$\mathcal{U}$(pclauses, $\mathbb{R}^+$) is the class of all nonnegative utility functions satisfying the constraint $\sum_{Y \subseteq X} (-1)^{|X \setminus Y|+1} \cdot u(\mathcal{PS} \setminus Y) \ge 0$ for all $X \subseteq \mathcal{PS}$.*

Proof. Follows from the fact that weights of positive clauses are determined by equation (9). □

Note that the property corresponding to $\mathcal{U}(\mathit{pclauses}, \mathbb{R}^+)$ implies monotonicity, normalization, and submodularity. Normalization and monotonicity follow from Theorem 3.11. To show submodularity, let $G \in \mathcal{L}(\mathit{pclauses}, \mathbb{R}^+)$ and let $X, Y \subseteq \mathcal{PS}$. For positive clauses φ, $X \cup Y \models \varphi$ together with $X \not\models \varphi$ implies $Y \models \varphi$. Furthermore, $X \not\models \varphi$ implies $X \cap Y \not\models \varphi$. Therefore:

$$\{(\varphi, w) \in G : X \cup Y \models \varphi \text{ and } X \not\models \varphi\} \quad \subseteq \quad \{(\varphi, w) \in G : Y \models \varphi \text{ and } X \cap Y \not\models \varphi\}$$

As all the weights w are positive, we immediately obtain the required inequality characterizing submodularity, namely $u_G(X \cup Y) - u_G(X) \leq u_G(Y) - u_G(X \cap Y)$.

An example which confirms that not all submodular utility functions belong to $\mathcal{U}(\mathit{pclauses}, \mathbb{R}^+)$ is the function $u : X \mapsto \min(2, |X|)$ for $\mathcal{PS} = \{p, q, r\}$. On the one hand, u is submodular, on the other we must have $w_{p \vee q \vee r} = -1$ if we are to express u using positive clauses. The previous theorem holds also for $\mathcal{U}(\mathit{pclauses} + \top, \mathbb{R}^+)$ if "normalized" is replaced with "nonnegative".

The kind of functions characterized by Theorem 3.13 are also known as *belief functions*, while those characterized by Theorem 3.14 are known as *plausibility functions* (when the functions are restricted to $[0, 1]$) [29, 30].

3.4 Summary

Our correspondence results are summarized in Table 1. We have not analyzed the interplay of bounding the length of formulas and restricting weights to positive reals in detail. By Theorem 3.5, any language restricting the length of formulas to at most k atoms can only generate k-additive utility functions. The opposite direction is less clear. While inspection of the proofs of Theorems 3.13 and 3.14 show that these results extend to the k-additive case in the expected manner, this is not so for Theorems 3.9 and 3.11. For instance, we do not know whether $\mathcal{U}(k\text{-}\mathit{cubes}, \mathbb{R}^+)$ is the class of all nonnegative k-additive functions or only a subclass thereof.

4 Succinctness

In this section, we consider how space efficient languages are relative to one another. In order to do so, we first provide a definition of goalbase size so that we have grounds for comparison.

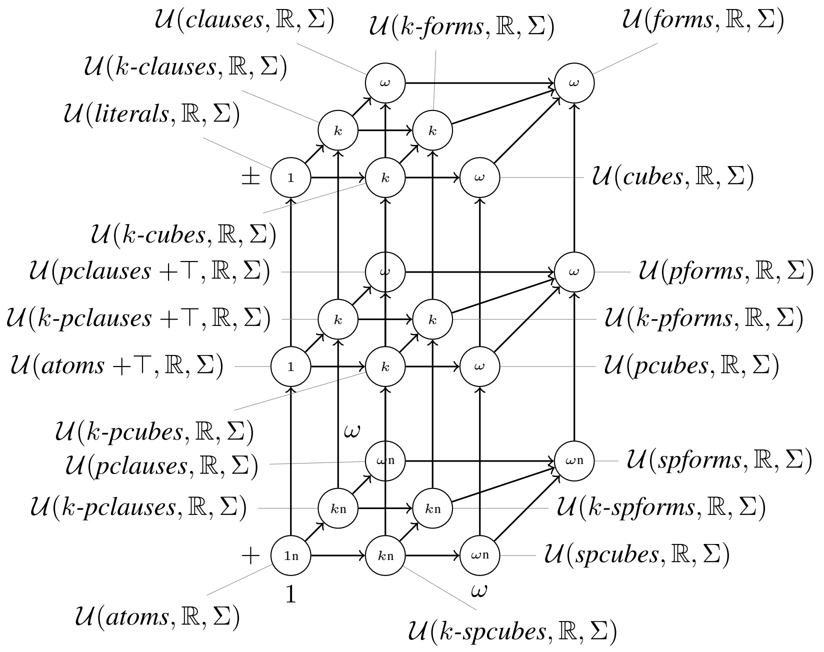
Definition 4.1 (Formula Length and Goalbase Size) The *length of a formula* φ is the number of occurrences of atoms it contains. The *size of a weighted goal* (φ, w) is the length of φ plus the number of bits needed to store w (that is, $\log w$ bits). The *size of a goalbase* G, written as $\text{size}(G)$, 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 $n = |\mathcal{PS}_n|$. 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. (Superexponential growth in weights affects all languages equally.)

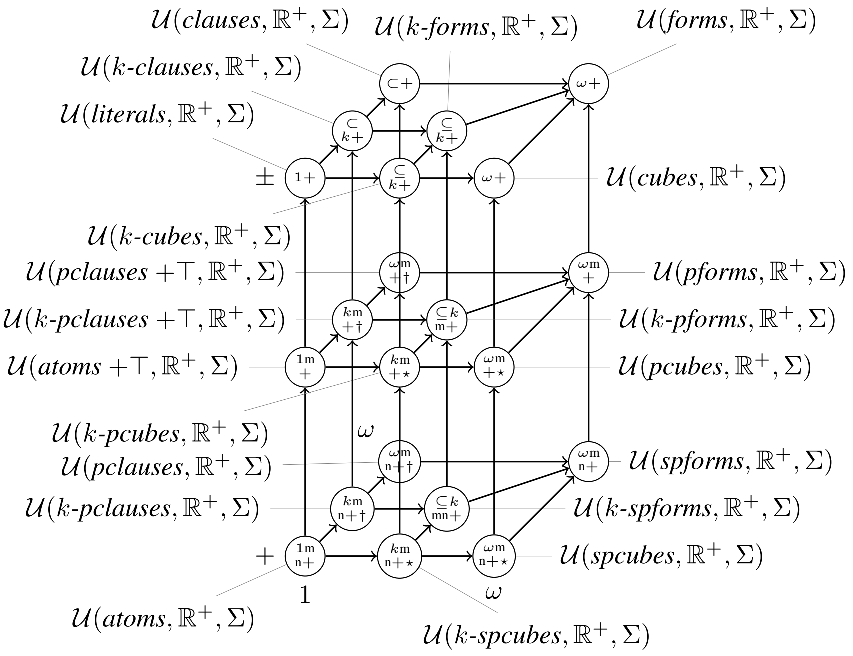
Frequently one language contains shorter representations of some utility functions than does another language. Here we offer a definition of relative succinctness to make this notion precise. This definition is similar to ones given by Cadoli et al. [31] and Coste-Marquis et al. [18]. Because we wish to compare languages which differ in expressive power, we define succinctness over only the expressive overlap of the languages being compared. This leads to some counterintiutive results for languages with little expressive overlap and makes the comparative succinctness relation intransitive, but it also permits us to make comparisons where the expressive overlap is substantial, though not total.

Definition 4.2 (Succinctness) Let $\mathcal{L}(\Phi, W, F)$ and $\mathcal{L}(\Psi, W', F')$ be goalbase languages and $\mathcal{U}$ a class of utility functions for which every member is expressible in both languages. Then $\mathcal{L}(\Phi, W, F) \preceq_{\mathcal{U}} \mathcal{L}(\Psi, W', F')$ iff there exists a function $f : \mathcal{L}(\Phi, W, F) \to \mathcal{L}(\Psi, W', F')$ and a polynomial p such that for all $G \in \mathcal{L}(\Phi, W, F)$, if $u_{G,F} \in \mathcal{U}$ then $u_{G,F} = u_{f(G),F'}$ and $\text{size}(f(G)) \leq p(\text{size}(G))$.

Read $\mathcal{L} \preceq_{\mathcal{U}} \mathcal{L}'$ as: $\mathcal{L}'$ is at least as succinct as $\mathcal{L}$ over the class $\mathcal{U}$. When $\mathcal{L}'$ is strictly more succinct than $\mathcal{L}$—that is, in no case are representations more than polynomially worse, and in at least one case, they are



$\mathcal{U}(clauses, \mathbb{R}, \Sigma)$
$\mathcal{U}(k\text{-}forms, \mathbb{R}, \Sigma)$
$\mathcal{U}(forms, \mathbb{R}, \Sigma)$
$\mathcal{U}(k\text{-}clauses, \mathbb{R}, \Sigma)$
$\mathcal{U}(literals, \mathbb{R}, \Sigma)$
$\pm$
$\mathcal{U}(cubes, \mathbb{R}, \Sigma)$
$\mathcal{U}(k\text{-}cubes, \mathbb{R}, \Sigma)$
$\mathcal{U}(pclauses + \top, \mathbb{R}, \Sigma)$
$\mathcal{U}(pforms, \mathbb{R}, \Sigma)$
$\mathcal{U}(k\text{-}pclauses + \top, \mathbb{R}, \Sigma)$
$\mathcal{U}(k\text{-}pforms, \mathbb{R}, \Sigma)$
$\mathcal{U}(atoms + \top, \mathbb{R}, \Sigma)$
$\mathcal{U}(pcubes, \mathbb{R}, \Sigma)$
$\mathcal{U}(k\text{-}pcubes, \mathbb{R}, \Sigma)$
ω
$\mathcal{U}(pclauses, \mathbb{R}, \Sigma)$
$\mathcal{U}(spforms, \mathbb{R}, \Sigma)$
$\mathcal{U}(k\text{-}pclauses, \mathbb{R}, \Sigma)$
$\mathcal{U}(k\text{-}spforms, \mathbb{R}, \Sigma)$
$+$
$\mathcal{U}(spcubes, \mathbb{R}, \Sigma)$
1
ω
$\mathcal{U}(atoms, \mathbb{R}, \Sigma)$
$\mathcal{U}(k\text{-}spcubes, \mathbb{R}, \Sigma)$



$\mathcal{U}(clauses, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(k\text{-}forms, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(forms, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(k\text{-}clauses, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(literals, \mathbb{R}^+, \Sigma)$
$\pm$
$\mathcal{U}(cubes, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(k\text{-}cubes, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(pclauses + \top, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(pforms, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(k\text{-}pclauses + \top, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(k\text{-}pforms, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(atoms + \top, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(pcubes, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(k\text{-}pcubes, \mathbb{R}^+, \Sigma)$
ω
$\mathcal{U}(pclauses, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(spforms, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(k\text{-}pclauses, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(k\text{-}spforms, \mathbb{R}^+, \Sigma)$
$+$
$\mathcal{U}(spcubes, \mathbb{R}^+, \Sigma)$
1
ω
$\mathcal{U}(atoms, \mathbb{R}^+, \Sigma)$
$\mathcal{U}(k\text{-}spcubes, \mathbb{R}^+, \Sigma)$

Expressivity

Each node represents one language, and an arrow from one node to another indicates that the tail language is included in the head language. Within each node, the expressivity of the language is given, according to the key below:

1	1-additive (modular)
k	k-additive
ω	ω-additive (general)
n	normalized
$+$	nonnegative
m	monotone
$\star$	plausibility function
$\dagger$	belief function
$\subset$	proper subset of
$\subseteq$	subset of

Where $\subseteq$ (or $\subset$) is indicated, the language represents a (proper) subset of the class of utility functions with the given properties. In all other cases, the language represents exactly the class of utility functions with the given properties. The x-axis (increasing to the right) is the *cubes* axis, along which allowable cubes grow from length 1 up to ω; the y-axis (increasing into the page) is the *clauses* axis, also running from 1 to ω. The z-axis (decreasing upward) is the *positivity* axis and has three steps: strictly positive, positive, and general. Each language in the lower graph is a sublanguage of the corresponding language with general weights in the upper graph, but we have omitted these arrows for clarity.

Table 1 Summary of Expressivity Results

super-polynomially better in $\mathcal{L}'$—we write $\mathcal{L} \prec_{\mathcal{U}} \mathcal{L}'$. When we have nonstrict succinctness in both directions, we write $\mathcal{L} \sim_{\mathcal{U}} \mathcal{L}'$; when we have nonstrict succinctness in neither direction, i.e., incomparability, we write $\mathcal{L} \perp_{\mathcal{U}} \mathcal{L}'$. 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 $\mathcal{L}$ and $\mathcal{L}'$.

Fact 4.1 *For all languages $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$:*

1. *If $\mathcal{L}_1 \subseteq \mathcal{L}_2$, then $\mathcal{L}_1 \preceq \mathcal{L}_2$.*
2. *If $\mathcal{L}_1 \preceq \mathcal{L}_2$ and $\mathcal{L}_3 \subseteq \mathcal{L}_1$, then $\mathcal{L}_3 \preceq \mathcal{L}_2$.*
3. *If $\mathcal{L}_1 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_3$ and $\mathcal{L}_1 \prec \mathcal{L}_2 \preceq \mathcal{L}_3$, then $\mathcal{L}_1 \prec \mathcal{L}_3$.*
4. *If $\mathcal{L}_1 \perp \mathcal{L}_2$ and $\mathcal{L}_1 \cup \mathcal{L}_2 \subseteq \mathcal{L}_3$, then $\mathcal{L}_1, \mathcal{L}_2 \prec \mathcal{L}_3$.*
5. *If $\mathcal{L}_1 \sim \mathcal{L}_2$ and $\mathcal{U}(\mathcal{L}_1) = \mathcal{U}(\mathcal{L}_2)$, then $\mathcal{L}_1 \odot \mathcal{L}_3$ iff $\mathcal{L}_2 \odot \mathcal{L}_3$, where $\odot \in \{\sim, \succeq, \succ, \preceq, \prec, \perp\}$.*

Note that Fact 4.1.2 is useful contrapositively also, for deriving $\not\preceq$ results for superlanguages. For Fact 4.1.3, it would be inadequate to require that $\mathcal{L}_1 \preceq \mathcal{L}_2 \prec \mathcal{L}_3$ instead, since it could happen that $\mathcal{L}_1$ is too small to represent the utility functions which cause $\mathcal{L}_2 \prec \mathcal{L}_3$. Fact 4.1.5 expresses the notion that if two languages are equal in succinctness and expressivity, then they stand in the same succinctness relation with any third language.

In the rest of this section, we prove many pairwise succinctness results in order to have as full a picture as possible of the qualities of the languages introduced in Section 3. All known succinctness results among these languages are summarized in Table 2 at the end of this section.

4.1 Some Basic Succinctness and Equivalence Results

Many succinctness and equivalence results can be arrived at merely by knowing the expressivity of the languages being compared and the basic properties of the succinctness relation contained in Fact 4.1.

Theorem 4.3 *For any fixed k, arbitrary set of formulas Ψ, and arbitrary sets of weights W and W': If $\Phi \subseteq k$-forms, then $\mathcal{L}(\Phi, W) \succeq \mathcal{L}(\Psi, W')$.*

Proof. There are only $O(n^k)$ formulas of length k or less, and so any utility function u representable in $\mathcal{L}(\Phi, W)$ cannot have a representation more than polynomially larger than the best one in $\mathcal{L}(\mathit{forms}, \mathbb{R})$. Hence, $\mathcal{L}(\Phi, W) \succeq \mathcal{L}(\mathit{forms}, \mathbb{R})$. Furthermore, $\mathcal{L}(\mathit{forms}, \mathbb{R}) \supseteq \mathcal{L}(\Psi, W')$, and so by Fact 4.1.2 we have that $\mathcal{L}(\Phi, W) \succeq \mathcal{L}(\Psi, W')$. □

From Theorem 4.3, it follows that all k-languages are pairwise equally succinct. For example, $\mathcal{L}(k\text{-}\mathit{spcubes}, \mathbb{R}) \sim \mathcal{L}(k\text{-}\mathit{forms}, \mathbb{R}^+)$.

Next, we establish the relationships between positive and strictly positive languages:

Lemma 4.4 *If Φ is a strictly positive set of formulas, then $\mathcal{L}(\Phi, W) \sim \mathcal{L}(\Phi \cup \{\top\}, W)$.*

Proof. $\mathcal{L}(\Phi, W) \preceq \mathcal{L}(\Phi \cup \{\top\}, W)$ by inclusion. For the converse: Fix $G \in \mathcal{L}(\Phi \cup \{\top\}, W)$. If G does not contain $\top$, then $G \in \mathcal{L}(\Phi, W)$ also. If G contains $\top$, combine all occurrences $(\top, w_1), \ldots, (\top, w_k)$ into a single $(\top, \sum_{i=1}^{k} w_i)$. If $\sum_{i=1}^{k} w_i = 0$, then remove $\top$ to again produce a goalbase in both languages. If instead $\top$ now has nonzero weight, then u_G is not representable in $\mathcal{L}(\Phi, W)$, since u_G is not normalized and by Lemma 3.6 only normalized utility functions can be represented using strictly positive formulas (let P be the null property). Therefore, any u representable in both languages has exactly the same representations in both. □

Theorem 4.5 $\mathcal{L}(\mathit{pforms}, W) \sim \mathcal{L}(\mathit{spforms}, W)$, $\mathcal{L}(\mathit{pcubes}, W) \sim \mathcal{L}(\mathit{spcubes}, W')$, *and* $\mathcal{L}(\mathit{pclauses} + \top, W) \sim \mathcal{L}(\mathit{pclauses}, W')$.

Proof. When $W = W'$, the result is a direct consequence of Lemma 4.4, giving us the first equivalence. By Theorem 3.2, $\mathcal{L}(\mathit{pcubes}, \mathbb{R})$ has unique representations, from which follows that its sublanguages do also, and so any utility function representable in both $\mathcal{L}(\mathit{pcubes}, W)$ and $\mathcal{L}(\mathit{spcubes}, W')$ has the same representation in both, yielding the second equivalence. By Corollary 3.4, the same holds for $\mathcal{L}(\mathit{pclauses} + \top, W)$ and $\mathcal{L}(\mathit{pclauses}, W')$, giving the third equivalence. □

Theorem 4.6 $\mathcal{L}(\mathit{spcubes}, \mathbb{R}^+) \sim \mathcal{L}(\mathit{pclauses}, \mathbb{R}^+)$.

Proof. Every utility function expressible in $\mathcal{L}(\mathit{spcubes}, \mathbb{R}^+)$ is supermodular, while every utility function expressible in $\mathcal{L}(\mathit{pclauses}, \mathbb{R}^+)$ is submodular. Let u be such a utility function. The only nonnegative utility functions which are both supermodular and submodular are modular, and so the spcubes representation of u is in 1-spcubes and the pclauses representation is in 1-pclauses. Since 1-spcubes and 1-pclauses are just atoms, u has the same representation in both $\mathcal{L}(\mathit{spcubes}, \mathbb{R}^+)$ and $\mathcal{L}(\mathit{pclauses}, \mathbb{R}^+)$. □

4.2 Equivalence via Goalbase Translation

It is sometimes possible to show that two languages are equally succinct by applying a size-preserving translation to the goalbases in both directions.

Lemma 4.7 *Let Φ and Ψ be sets of formulas. If $\Phi \supseteq$ cubes or $\Phi \supseteq$ clauses, $\Psi \supseteq$ cubes or $\Psi \supseteq$ clauses, and $\Phi \cup \Psi \subseteq$ cubes $\cup$ clauses, then $\mathcal{L}(\Phi, \mathbb{R}) \sim \mathcal{L}(\Psi, \mathbb{R})$.*

P r o o f. Suppose that $G \in \mathcal{L}(\Phi, \mathbb{R})$. Enumerate $(\varphi_i, w_i) \in G$. We construct an equivalent goalbase G'. Let

$$G_0 = G$$

$$G_{i+1} = \begin{cases} (G_i \setminus \{(\varphi_i, w_i)\}) \cup \{(\neg\varphi_i, -w_i), (\top, w_i)\} & \text{if } \varphi_i \notin \Psi \\ G_i & \text{otherwise} \end{cases}$$

and let $G' = G_{|G|}$.

The transformation produces an equivalent goalbase: By equivalences (5) and (6) from Fact 3.1, $G_i \equiv G_{i+1}$ for all i, so $G = G_1 \equiv G_2 \equiv \cdots \equiv G_{|G|-1} \equiv G_{|G|} = G'$.

The transformation produces a goalbase in the appropriate language: Suppose that $\varphi \in \Phi$. The set Ψ contains at least every clause or every cube. If φ is a clause, then $\neg\varphi$ is (equivalent to) a cube, and vice versa. Hence at least one of φ and $\neg\varphi$ are in Ψ. $\top$ is both a cube ($\bigwedge \emptyset$) and a clause ($p \vee \neg p$), so $\top \in \Psi$ regardless. Thus $G' \in \mathcal{L}(\Psi, \mathbb{R})$.

The transformation produces a goalbase as succinct as the original: If φ is a cube, then φ requires the same number of atoms and binary connectives as as $\neg\varphi$ (written as a clause); similarly, if φ is a clause. The only increase in size between G and G' can come from the addition of $\top$, so we have that $|G'| \leq |G| + 1$.

Therefore, $\mathcal{L}(\Phi, \mathbb{R}) \succeq \mathcal{L}(\Psi, \mathbb{R})$. By the same argument $\mathcal{L}(\Phi, \mathbb{R}) \preceq \mathcal{L}(\Psi, \mathbb{R})$. So $\mathcal{L}(\Phi, \mathbb{R}) \sim \mathcal{L}(\Psi, \mathbb{R})$. $\square$

Theorem 4.8 $\mathcal{L}(\textit{cubes}, \mathbb{R}) \sim \mathcal{L}(\textit{clauses}, \mathbb{R})$.

P r o o f. Follows immediately from Lemma 4.7. $\square$

4.3 Strict Succinctness and Incomparability by Counterexample

The most straightforward method for showing that one language is not more succinct than another is to produce a family of utility functions whose representations grow exponentially in one but merely polynomially in the other. Here we define two families of utility functions which will be used repeatedly for demonstrating strict succinctness and incomparability results.

Definition 4.9 Let $u_n^\forall$ and $u_n^\exists$ be the utility functions over $\mathcal{PS}_n$ where

$$u_n^\forall(X) = \begin{cases} 1 & \text{if } X = \mathcal{PS} \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad u_n^\exists(X) = \begin{cases} 1 & \text{if } X \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 4.10

$$\mathcal{L}(\textit{pclauses}, \mathbb{R}), \mathcal{L}(\textit{pclauses} + \top, \mathbb{R}) \prec \mathcal{L}(\textit{clauses}, \mathbb{R})$$
$$\mathcal{L}(\textit{spcubes}, \mathbb{R}), \mathcal{L}(\textit{pcubes}, \mathbb{R}) \prec \mathcal{L}(\textit{cubes}, \mathbb{R})$$

P r o o f. $\mathcal{L}(\textit{pcubes}, \mathbb{R}) \preceq \mathcal{L}(\textit{cubes}, \mathbb{R})$ since every pcube is a cube. Consider the family of utility functions $u_n^\exists$, which may be represented in cubes as

$$\Big\{(\top, 1), \Big(\bigwedge\{\neg p : p \in \mathcal{PS}\}, -1\Big)\Big\}$$

the length of which increases linearly with n. $u_n^\exists$ may be represented in pcubes as

$$\Big\{\Big(\bigwedge X, w_{\bigwedge X}\Big) : \emptyset \subset X \subseteq \mathcal{PS}\Big\} \quad \text{where} \quad w_{\bigwedge X} = \begin{cases} 1 & \text{if } |X| \text{ is odd} \\ -1 & \text{if } |X| \text{ is even.} \end{cases}$$

Every pcube except $\top$ receives a nonzero weight, and and by Theorem 3.2 this representation is unique. For any n, $2^n - 1$ pcubes are weighted, so the size of the representation increases exponentially with n.

For $\mathcal{L}(\mathit{pclauses} + \top, \mathbb{R}) \prec \mathcal{L}(\mathit{clauses}, \mathbb{R})$, replace "$\exists$", "$\wedge$", "pcubes", "cubes", and "Theorem 3.2" in the above proof with "$\forall$", "$\vee$", "pclauses $+\top$", "clauses", and "Theorem 3.3", respectively. □

Theorem 4.11 $\mathcal{L}(\mathit{pcubes}, \mathbb{R}), \mathcal{L}(\mathit{spcubes}, \mathbb{R}) \perp \mathcal{L}(\mathit{pclauses}, \mathbb{R}), \mathcal{L}(\mathit{pclauses} + \top, \mathbb{R})$.

P r o o f. ($\not\prec$) The family of utility functions $u_n^{\exists}$ is represented uniquely and linearly as $\{(\bigvee \mathcal{PS}, 1)\}$ in pclauses. When representing $u_n^{\exists}$ in pcubes, the weights $w_{\bigwedge X} = (-1)^{|X|+1}$, so the unique representation there assigns nonzero weights to $2^n - 1$ distinct pcubes.

($\not\succ$) The family of utility functions $u_n^{\forall}$ is represented uniquely and linearly as $\{(\bigwedge \mathcal{PS}, 1)\}$ in pcubes, but the representation in pclauses is exponential, as shown in the proof of Theorem 4.10. □

Theorem 4.12

$$\mathcal{L}(\mathit{pclauses}, \mathbb{R}), \mathcal{L}(\mathit{pclauses} + \top, \mathbb{R}) \not\preceq \mathcal{L}(\mathit{spcubes}, \mathbb{R}^+), \mathcal{L}(\mathit{pcubes}, \mathbb{R}^+)$$
$$\mathcal{L}(\mathit{spcubes}, \mathbb{R}), \mathcal{L}(\mathit{pcubes}, \mathbb{R}) \not\preceq \mathcal{L}(\mathit{pclauses}, \mathbb{R}^+), \mathcal{L}(\mathit{pclauses} + \top, \mathbb{R}^+)$$

P r o o f. The first part is demonstrated by the $u_n^{\forall}$ family of functions, the second part by the $u_n^{\exists}$ family. □

Corollary 4.13

$$\mathcal{L}(\mathit{spcubes}, \mathbb{R}) \prec \mathcal{L}(\mathit{forms}, \mathbb{R}) \qquad \mathcal{L}(\mathit{pclauses}, \mathbb{R}) \prec \mathcal{L}(\mathit{forms}, \mathbb{R})$$
$$\mathcal{L}(\mathit{pcubes}, \mathbb{R}) \prec \mathcal{L}(\mathit{forms}, \mathbb{R}) \qquad \mathcal{L}(\mathit{pclauses} + \top, \mathbb{R}) \prec \mathcal{L}(\mathit{forms}, \mathbb{R})$$

P r o o f. Immediately from Fact 4.1.3 and Theorem 4.10. □

Corollary 4.14

$$\mathcal{L}(\mathit{spcubes}, \mathbb{R}) \prec \mathcal{L}(\mathit{pforms}, \mathbb{R}) \qquad \mathcal{L}(\mathit{pclauses}, \mathbb{R}) \prec \mathcal{L}(\mathit{pforms}, \mathbb{R})$$
$$\mathcal{L}(\mathit{pcubes}, \mathbb{R}) \prec \mathcal{L}(\mathit{pforms}, \mathbb{R}) \qquad \mathcal{L}(\mathit{pclauses} + \top, \mathbb{R}) \prec \mathcal{L}(\mathit{pforms}, \mathbb{R})$$

P r o o f. Immediately from Fact 4.1.4 and Theorem 4.11. □

4.4 Strict Succinctness, Nonconstructively

It is difficult to demonstrate that a language which lacks unique representations is less succinct than another language, because the exhibition of a single exponentially growing family of utility functions (as above) does not preclude the existence of better representations in the same language. Here, we take a nonconstructive approach to produce the following strict succinctness result:

Theorem 4.15 $\mathcal{L}(\mathit{cubes}, \mathbb{R}) \prec \mathcal{L}(\mathit{forms}, \mathbb{R})$.

To prove this theorem, we will introduce the Fourier transform on Boolean domains, using the same notation as in [32]. Then, to apply the Fourier transform on cubes, we will need two lemmas. The first one will show how the size of cubes relates to their degree. The second lemma will show that a function which approximates parity accurately necessarily has a high degree.

For each $S \subseteq \mathcal{PS}$, the *parity function* $\chi_S : 2^{\mathcal{PS}} \to \{-1, 1\}$ is defined as $\chi_S(X) = (-1)^{|S \cap X|}$. Because these functions form an orthonormal basis for the space of real functions on $2^{\mathcal{PS}}$, any function $f : 2^{\mathcal{PS}} \to \mathbb{R}$ can be represented as a linear combination with respect to this basis. This is known as the *Fourier-Walsh expansion*:

$$f(X) = \sum_{S \subseteq \mathcal{PS}} \hat{f}(S)\chi_S(X),$$

where the $\hat{f}(S) \in \mathbb{R}$ are the Fourier coefficients, which are computed as follows:

$$\hat{f}(S) = \frac{1}{2^n} \sum_{X \subseteq \mathcal{PS}} f(X)\chi_S(X)$$

for all $S \subseteq \mathcal{PS}$. The degree of a function f is the cardinality of the largest subset of S with a nonzero Fourier coefficient: $\deg(f) = \max\{|S| : \hat{f}(S) \neq 0\}$.

Lemma 4.16 *If $G \in \mathcal{L}(k\text{-cubes}, \mathbb{R})$, then the degree of u_G will be at most k.*

Proof. Let us first show the lemma under the condition that G contains a single cube of at most k literals. Let $y \in \mathcal{PS}$ be any variable not present in that cube. For all $S \subseteq \mathcal{PS}$ such that $y \in S$, the Fourier coefficients $\hat{u}_G(S)$ are the following:

$$\begin{aligned}\hat{u}_G(S) &= \frac{1}{2^n} \sum_{X \subseteq \mathcal{PS},\, y \notin X} u_G(X)\chi_S(X) + \frac{1}{2^n} \sum_{X \subseteq \mathcal{PS},\, y \in X} u_G(X)\chi_S(X) \\ &= \frac{1}{2^n} \sum_{X \subseteq \mathcal{PS} \setminus \{y\}} \chi_S(X)\big(u_G(X) - u_G(X \cup \{y\})\big) = 0.\end{aligned}$$

Therefore, if $\hat{u}_G(S) \neq 0$ then S contains only variables present in the cube, thus $|S| \leq k$. Thus, the degree of u_G is at most k. Suppose now that G contains more than one cube. Then, u_G can be seen as a linear combination of utilities each generated by single cubes. Because the Fourier transform is linear (in other words, if $f = g + h$ then $\hat{f} = \hat{g} + \hat{h}$), the degree of u_G is also bounded by k. □

The next lemma is familiar from the literature on bounding the complexity of Boolean circuits. The proof is inspired by the lecture notes of L. Trevisian [33, Lemma 4].

Lemma 4.17 *There are some constants $c > 0$ and $n_0 > 0$ (such constants are completely independent from n) such that, if $n \geq n_0$, then given any function $g : 2^{\mathcal{PS}} \to \mathbb{R}$ that agrees with the parity function $\chi_{\mathcal{PS}}$ on at least $\frac{3}{4}$ of $2^{\mathcal{PS}}$, the degree of g will be at least $c\sqrt{n}$.*

Proof. Let $g : 2^{\mathcal{PS}} \to \mathbb{R}$ be a function that agrees with $\chi_{\mathcal{PS}}$ on at least a $\frac{3}{4}$ fraction of $2^{\mathcal{PS}}$. Let t be the degree of g. Let $A = \{X \subseteq \mathcal{PS} \mid g(X) = \chi_{\mathcal{PS}}(X)\}$, which by definition has the property $|A| \geq \frac{3}{4}2^n$ where $n = |\mathcal{PS}|$. Clearly, for any $S \subseteq \mathcal{PS}$ and $X \in A$, we have $\chi_S(X) = \chi_{\mathcal{PS}}(X)\chi_{\mathcal{PS}\setminus S}(X) = g(X)\chi_{\mathcal{PS}\setminus S}(X)$. Note that the function $\chi_S(X)$ has a degree equal to $|S|$, but can be replaced over A by $g(X)\chi_{\mathcal{PS}\setminus S}(X)$, which has a degree of at most $t+n-|S|$. Consequently, any function χ_S over A with $|S| \geq \frac{n}{2}$ can be replaced by its Fourier-Walsh expansion, which is a linear combination over the set of functions $\mathcal{F} = \{\chi_{S'} \mid S' \subseteq \mathcal{PS}, |S'| \leq t + \frac{n}{2}\}$.

The Fourier transform guarantees that any function $f : A \to \mathbb{R}$ can be written as a linear combination over $\{\chi_S \mid S \subseteq \mathcal{PS}\}$. But because each of these functions χ_S over A can itself be decomposed over $\mathcal{F}$, $f : A \to \mathbb{R}$ can be written as a linear combination over $\mathcal{F}$ as follows: $f(X) = \sum_{S \subseteq \mathcal{PS},\, |S| \leq t + \frac{n}{2}} \alpha_S \cdot \chi_S(X)$, with $\alpha_S \in \mathbb{R}$. The number of α_S coefficients is $\sum_{k=0}^{t+\frac{n}{2}} \binom{n}{k}$. However, because the set of functions $f : A \to \mathbb{R}$ forms a vector space over the reals of dimension $|A|$, the number of α_S coefficients must be at least $\frac{3}{4}2^n$. This leads to the inequality $\sum_{k=\frac{n}{2}}^{t+\frac{n}{2}} \binom{n}{k} \geq \frac{2^n}{4}$ which, after applying Stirling's approximation and some basic formula manipulation, becomes $t = \Omega(\sqrt{n})$. □

We are now in position to prove Theorem 4.15.

Proof. (Theorem 4.15.) In the first part of the proof, we will show that the function $\chi_{\mathcal{PS}}$ can be polynomially represented in $\mathcal{L}(\textit{forms}, \mathbb{R})$, and in the second part, we will show that this is not the case for $\mathcal{L}(\textit{cubes}, \mathbb{R})$. Let us prove the first part. It is known that the parity function can be written as a Boolean AND/OR formula φ_{parity} containing at most n^2 literals [34, p. 100]. We can then build the goalbase $G = \{(\top, -1), (\varphi_{\text{parity}}, 2)\}$ which generates $\chi_{\mathcal{PS}}$ with a polynomial number of literals.

Let us now prove the second part. More precisely, we will show that in order to represent $\chi_{\mathcal{PS}}$ in $\mathcal{L}(\textit{cubes}, \mathbb{R})$, at least $2^{\Omega(\sqrt{n})}$ cubes are required. Consider a goalbase $G = \{(\varphi_i, \alpha_i)\}_i$ where φ_i are cubes (possibly containing negative literals). Let G_{low} be all the pairs (φ_i, α_i) of G such that the number of literals in φ_i is strictly lower than $c\sqrt{n}$, where the constant c is being chosen as in Lemma 4.17. Let $G_{\text{high}} = G \setminus G_{\text{low}}$. Let $u_{G_{\text{low}}}$ be the utility function generated by G_{low}. Together with Lemma 4.16, we can now apply Lemma 4.17 which implies that $u_{G_{\text{low}}}$ disagrees with $\chi_{\mathcal{PS}}$ on at least a $\frac{1}{4}$ fraction of $2^{\mathcal{PS}}$. In order for u_G to compute the parity function, the cubes of G_{high} must compensate the errors made by those of G_{low} on this $\frac{1}{4}$ fraction of $2^{\mathcal{PS}}$, but we will show that this compensation requires a very large number of cubes. Let us thus evaluate the fraction of $2^{\mathcal{PS}}$ which can be affected by the cubes of G_{high}. Because each cube has at least $c\sqrt{n}$ literals, at most $2^{n-c\sqrt{n}}$ interpretations will be affected by each of these cubes. Thus, to affect $\frac{1}{4}$ fraction of $2^{\mathcal{PS}}$, G_{high} will need to have at least $\frac{\frac{2^n}{4}}{2^{n-c\sqrt{n}}} = 2^{c\sqrt{n}-2}$ cubes. □

Corollary 4.18 $\mathcal{L}(\textit{clauses}, \mathbb{R}) \prec \mathcal{L}(\textit{forms}, \mathbb{R})$.

Proof. Follows immediately from Theorems 4.8 and 4.15, Fact 4.1.5, and Corollary 3.7. □

Elkind et al. [22] give a succinctness result in their Example 1 and Theorem 1 which compares the basic marginal contribution nets (MC-nets) of Ieong and Shoham [21] with general MC-nets. Because basic MC-nets are effectively goalbases in $\mathcal{L}(\textit{cubes}, \mathbb{R})$ and general MC-nets are goalbases in $\mathcal{L}(\textit{forms}, \mathbb{R})$, we can adapt their proof to arrive at another nonconstructive succinctness result:

Theorem 4.19 $\mathcal{L}(\textit{cubes}, \mathbb{R}^+) \prec \mathcal{L}(\textit{forms}, \mathbb{R}^+)$.

Proof. $\mathcal{L}(\textit{cubes}, \mathbb{R}^+) \preceq \mathcal{L}(\textit{forms}, \mathbb{R}^+)$ by inclusion. For strict succinctness: Enumerate $\mathcal{PS}_{2n} = \{x_1, x_2, \dots, x_{2n-1}, x_{2n}\}$ and define the family of utility functions

$$u_{2n}(X) = \begin{cases} 1 & \text{if } x_{2i-1} \in X \text{ or } x_{2i} \in X, \text{ for all } 1 \le i \le n \\ 0 & \text{otherwise.} \end{cases}$$

The goalbase $\{((x_1 \vee x_2) \wedge \cdots \wedge (x_{2n-1} \vee x_{2n}), 1)\}$ represents $u_{2n} \in \mathcal{L}(\textit{spforms}, \mathbb{R}^+)$ linearly.

Because u_{2n} is nonnegative, it has a representation $G \in \mathcal{L}(\textit{cubes}, \mathbb{R}^+)$. If $(\varphi, w) \in G$ and $w > 0$, then for each $1 \le i \le n$, φ contains at least one of x_{2i-1} and x_{2i} as a literal: Suppose otherwise, and let X be a state where $X \models \varphi$ but x_{2i-1} and x_{2i} are false. Then $u_{2n}(X) \ge w$ since G contains no negative weights; but $u_{2n}(X) = 0$, and so $w = 0$, contrary to assumption. Next, consider the states X where $u_{2n}(X) = 1$ and for any state $Z \subset X$, $u_{2n}(Z) = 0$. For any two such minimal nonzero states X, Y, they must differ on at least two atoms p, q. (If X and Y differed on only one atom, then $X \subset Y$ or $Y \subset X$, contradicting minimality.) Therefore, every $(\varphi, w) \in G$ such that $X \models \varphi$ contains a literal p but not q and vice versa for every $(\psi, w') \in G$ such that $Y \models \psi$. Since each minimal state has at least one (φ, w) which is true there but in no other minimal state, and there are 2^n such minimal states, $|G| \ge 2^n$, and so $\text{size}(G) \in O(2^{|\mathcal{PS}|})$. □

Note that the bulk of this proof shows that $\mathcal{L}(\textit{spforms}, \mathbb{R}^+) \not\preceq \mathcal{L}(\textit{cubes}, \mathbb{R}^+)$. This, combined with the contrapositive of Fact 4.1.2, produces many of the $\not\preceq$ results seen in Table 2.

4.5 Summary

Our succinctness results are summarized in Table 2. The table contains many more results than are proved in the text, but in all cases these are straightforward consequences of results which do appear in the text. (E.g., $\mathcal{L}(\textit{spcubes}, \mathbb{R}) \prec \mathcal{L}(\textit{clauses}, \mathbb{R})$.) There are many open questions (any cell which contains neither $\prec$, $\succ$, nor $\sim$ has something yet to be resolved). All open questions involve at least one language which lacks unique representations. Most cases in which nothing is known involve a language which uses positive formulas or general formulas. We suspect that resolving these questions will require difficult proofs, as the one for Theorem 4.15 which shows that $\mathcal{L}(\textit{cubes}, \mathbb{R}) \prec \mathcal{L}(\textit{forms}, \mathbb{R})$.

Finally, it is worth noting that $\sim$ is intransitive, due to the succinctness relation being defined over languages which may differ in expressivity. E.g., $\mathcal{L}(\textit{atoms}, \mathbb{R})$ is equally succinct as any other language, so $\mathcal{L}(\textit{atoms}, \mathbb{R}) \sim \mathcal{L}_1$ and $\mathcal{L}(\textit{atoms}, \mathbb{R}) \sim \mathcal{L}_2$, but it is still possible that $\mathcal{L}_1 \not\sim \mathcal{L}_2$.

	$\mathcal{L}(spcubes, \mathbb{R}^+, \Sigma)$	$\mathcal{L}(pclauses, \mathbb{R}^+, \Sigma)$	$\mathcal{L}(spforms, \mathbb{R}^+, \Sigma)$	$\mathcal{L}(pcubes, \mathbb{R}^+, \Sigma)$	$\mathcal{L}(pclauses + \top, \mathbb{R}^+, \Sigma)$	$\mathcal{L}(pforms, \mathbb{R}^+, \Sigma)$	$\mathcal{L}(cubes, \mathbb{R}^+, \Sigma)$	$\mathcal{L}(clauses, \mathbb{R}^+, \Sigma)$	$\mathcal{L}(forms, \mathbb{R}^+, \Sigma)$	$\mathcal{L}(spcubes, \mathbb{R}, \Sigma)$	$\mathcal{L}(pclauses, \mathbb{R}, \Sigma)$	$\mathcal{L}(spforms, \mathbb{R}, \Sigma)$	$\mathcal{L}(pcubes, \mathbb{R}, \Sigma)$	$\mathcal{L}(pclauses + \top, \mathbb{R}, \Sigma)$	$\mathcal{L}(pforms, \mathbb{R}, \Sigma)$	$\mathcal{L}(cubes, \mathbb{R}, \Sigma)$	$\mathcal{L}(clauses, \mathbb{R}, \Sigma)$	$\mathcal{L}(forms, \mathbb{R}, \Sigma)$
$\mathcal{L}(forms, \mathbb{R}, \Sigma)$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\succ$	$\succeq$	$\succeq$	$\succ$	$\succ$	$\succeq$	$\succ$	$\succ$	$\succeq$	$\succ$	$\succ$	$\sim$
$\mathcal{L}(clauses, \mathbb{R}, \Sigma)$	$\succeq$	$\succeq$		$\succeq$	$\succeq$		$\succeq$	$\succeq$		$\succ$	$\succ$		$\succ$	$\succ$		$\sim$	$\sim$	
$\mathcal{L}(cubes, \mathbb{R}, \Sigma)$	$\succeq$	$\succeq$		$\succeq$	$\succeq$		$\succeq$	$\succeq$		$\succ$	$\succ$		$\succ$	$\succ$		$\sim$		
$\mathcal{L}(pforms, \mathbb{R}, \Sigma)$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\not\succeq$			$\succ$	$\succ$	$\sim$	$\succ$	$\succ$	$\sim$			
$\mathcal{L}(pclauses + \top, \mathbb{R}, \Sigma)$	$\not\succeq$	$\sim$	$\not\succeq$	$\not\succeq$	$\sim$	$\not\succeq$	$\not\succeq$		$\not\succeq$	$\perp$	$\sim$	$\prec$	$\perp$	$\sim$				
$\mathcal{L}(pcubes, \mathbb{R}, \Sigma)$	$\sim$	$\not\succeq$	$\not\succeq$	$\sim$	$\not\succeq$	$\not\succeq$		$\not\succeq$	$\not\succeq$	$\sim$	$\perp$	$\prec$	$\sim$					
$\mathcal{L}(spforms, \mathbb{R}, \Sigma)$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\not\succeq$			$\succ$	$\succ$	$\sim$						
$\mathcal{L}(pclauses, \mathbb{R}, \Sigma)$	$\not\succeq$	$\sim$	$\not\succeq$	$\not\succeq$	$\sim$	$\not\succeq$	$\not\succeq$		$\not\succeq$	$\perp$	$\sim$							
$\mathcal{L}(spcubes, \mathbb{R}, \Sigma)$	$\sim$	$\not\succeq$	$\not\succeq$	$\sim$	$\not\succeq$	$\not\succeq$		$\not\succeq$	$\not\succeq$	$\sim$								
$\mathcal{L}(forms, \mathbb{R}^+, \Sigma)$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\succeq$	$\succ$	$\succeq$	$\sim$									
$\mathcal{L}(clauses, \mathbb{R}^+, \Sigma)$	$\succeq$	$\succeq$		$\succeq$	$\succeq$			$\sim$										
$\mathcal{L}(cubes, \mathbb{R}^+, \Sigma)$	$\succeq$	$\succeq$	$\not\succeq$	$\succeq$	$\succeq$	$\not\succeq$	$\sim$											
$\mathcal{L}(pforms, \mathbb{R}^+, \Sigma)$	$\succeq$	$\succeq$	$\sim$	$\succeq$	$\succeq$	$\sim$												
$\mathcal{L}(pclauses + \top, \mathbb{R}^+, \Sigma)$	$\sim$	$\sim$	$\preceq$	$\sim$	$\sim$													
$\mathcal{L}(pcubes, \mathbb{R}^+, \Sigma)$	$\sim$	$\sim$	$\preceq$	$\sim$														
$\mathcal{L}(spforms, \mathbb{R}^+, \Sigma)$	$\succeq$	$\succeq$	$\sim$															
$\mathcal{L}(pclauses, \mathbb{R}^+, \Sigma)$	$\sim$	$\sim$																
$\mathcal{L}(spcubes, \mathbb{R}^+, \Sigma)$	$\sim$																	

Table 2 Summary of Succinctness Results. Entries to be read row first. Empty cells are open questions.

5 Complexity

In this section, we analyze the effect that restrictions on goalbases have on the complexity of answering questions about the utility functions they represent, focusing specifically on the problem MAX-UTIL—finding a model which produces maximal utility, expressed as a decision problem.[2]

Definition 5.1 (MAX-UTIL) The decision problem MAX-UTIL(Φ, W, F) is defined as: Given a goalbase $G \in \mathcal{L}(\Phi, W, F)$ and an integer K, check whether there is a model $M \in 2^{\mathcal{PS}}$ where $u_G(M) \geq K$.

MAX-UTIL is clearly in NP for the unrestricted language, since whether $u_G(M) \geq K$ is polynomially checkable, and is NP-hard via a reduction from MAXSAT [36]. In this section, we consider the difficulty of MAX-UTIL for more restrictive languages. Note that if we permit goalbases to contain unsatisfiable formulas, then MAX-UTIL trivializes to SAT, since deciding the problem in the general case will involve determining what (if any) models make a formula true. Therefore, in the cases where we show NP-completeness, we do so *even in the case where goalbases contain only satisfiable formulas*. In contrast, in the cases where we show that MAX-UTIL $\in$ P, we do so without this restriction. As above, we deal only with $F = \Sigma$ as our aggregator, and so omit F in what follows.

[2] For all of the languages we consider, solving the decision problem MAX-UTIL will involve constructing a satisfying allocation, if one exists. It is possible to construct classes of goalbases where MAX-UTIL can be solved trivially, and yet it is not trivial to find a satisfying allocation. Uckelman and Witzel [35] give an alternate formulation of MAX-UTIL in which the decision and function problems do not come apart in this way.

5.1 Hardness Results

We first present results on languages for which MAX-UTIL is still NP-hard, despite the restrictions imposed.

Theorem 5.2 MAX-UTIL$(k\textit{-cubes}, \mathbb{Q}^+)$ *is NP-complete for $k \geq 2$, even if goalbases contain only satisfiable formulas.*

Proof. The decision problem MAX k-CONSTRAINT SAT is defined as: Given a set C of k-cubes in $\mathcal{PS}$ and an integer K, check whether there is a model $M \in 2^{\mathcal{PS}}$ which satisfies at least K of the k-cubes in C. MAX-UTIL$(k\textit{-cubes}, \mathbb{Q}^+)$ is a weighted version of MAX k-CONSTRAINT SAT, which is NP-complete for $k \geq 2$ [37, LO12, Appendix B]. □

In the remaining NP-completeness results, we do not state that formulas need be satisfiable, as these languages do not contain unsatisfiable formulas.

Theorem 5.3 MAX-UTIL$(k\textit{-clauses}, \mathbb{Q}^+)$ *is NP-complete for $k \geq 2$.*

Proof. MAX-UTIL$(2\textit{-clauses}, \mathbb{Q}^+)$ is a weighted version of the well-known NP-complete problem MAX 2-SAT [36], and hence is contained in MAX-UTIL$(k\textit{-clauses}, \mathbb{Q}^+)$ for $k \geq 2$. □

Theorem 5.4 MAX-UTIL$(k\textit{-spcubes}, \mathbb{Q})$ *is NP-complete for $k \geq 2$.*

Proof. We show NP-hardness for $k = 2$ by reduction from MAX 2-SAT [36], using a construction previously employed to show NP-hardness of the winner determination problem in combinatorial auctions when bids are encoded using k-additive functions [26]. Let S be a set of 2-clauses and let $K \leq |S|$. MAX 2-SAT asks whether there exists a subset S' of S with $|S'| = K$ that is satisfiable. We construct a goalbase G as follows:

- For any positive clause $(p \vee q) \in S$, add the weighted goals $(p, 1)$, $(q, 1)$, and $(p \wedge q, -1)$ to G.
- For any clause $(p \vee \neg q) \in S$ with one negative literal, add $(\top, 1)$, $(q, -1)$, and $(p \wedge q, 1)$ to G.
- For any clause $(\neg p \vee \neg q)$ with two negative literals, add $(\top, 1)$ and $(p \wedge q, -1)$ to G.

Clearly, there exists a satisfiable $S' \subseteq S$ with $|S'| = K$ iff there exists a model M such that $u_G(M) \geq K$. We are not yet done, because G is not a goalbase in *strictly* positive cubes. Let G' be the result of removing all occurrences of $(\top, 1)$ from G. If d is the number of nonpositive clauses in S, then $u_{G'}(M) = u_G(M) - d$ for any model M. Hence, MAX 2-SAT for S will succeed iff there exists a model M such that $u_{G'}(M) \geq K - d$. Therefore, MAX-UTIL$(2\textit{-spcubes}, \mathbb{Q})$ must be at least as hard as MAX 2-SAT. □

Theorem 5.5 MAX-UTIL$(k\textit{-pclauses}, \mathbb{Q})$ *is NP-complete for $k \geq 2$.*

Proof. The proof works by reduction from MAX 2-SAT, just as for Theorem 5.4, except that now we construct G as follows:

- For any clause of the form $p \vee q$, we simply add $(p \vee q, 1)$ to G.
- For any clause of the form $p \vee \neg q$, we add $(\top, 1)$, $(p, 1)$, and $(p \vee q, -1)$ to G.
- For any clause of the form $\neg p \vee \neg q$, we add $(\top, 1)$, $(p, -1)$, $(q, -1)$, and $(p \vee q, 1)$ to G.

As $\top$ is not a positive clause, we must eliminate all occurrences of $(\top, 1)$ in the same way as we did in the proof of Theorem 5.4. □

5.2 Easiness Results

For two languages (and all their sublanguages) we can obtain easiness results.

Theorem 5.6 $\text{MAX-UTIL}(\textit{pforms}, \mathbb{Q}^+) \in \text{P}$.

Proof. Since all weights are positive, whichever state makes the most formulas true is optimal. Because all formulas in the language are positive, we are guaranteed that every formula we encounter is satisfiable. In particular, the state $\mathcal{PS}$, in which all atoms are true, makes every positive formula true, and hence $\mathcal{PS}$ is always an optimal state. (In fact, $\mathcal{PS}$ is the maximal optimal state. There might also be optimal states making fewer atoms true.) This means that the algorithm which checks whether $u(\mathcal{PS}) \geq K$ decides every instance of $\text{MAX-UTIL}(\textit{pforms}, \mathbb{Q}^+)$; furthermore, finding the value of any single state is linear. □

Theorem 5.7 $\text{MAX-UTIL}(\textit{literals}, \mathbb{Q}) \in \text{P}$.

Proof. Fix a goalbase $G \in \mathcal{L}(\textit{literals}, \mathbb{Q})$. Keep for each atom p a number Δp, the difference between the sum of p's positive occurrences and sum of p's negative occurrences seen so far. (Initially $\Delta p = 0$.) Iterate over the formulas in G, updating the deltas as we go. (Thus, on seeing $(\neg p, 5)$, we subtract 5 from Δp.) On reaching the end of the goalbase, define a model $M = \{p : \Delta p > 0\}$. M will be the minimal optimal model. (The maximal optimal model is $\{p : \Delta p \geq 0\}$.) This algorithm is linear in the sum of the size of the goalbase and $\mathcal{PS}$. □

5.3 Summary

To summarize the results in this section, Theorems 5.2, 5.3, 5.4, and 5.5 show that MAX-UTIL is NP-complete for any language which contains $\mathcal{L}(2\text{-}\textit{pclauses}, \mathbb{Q})$, $\mathcal{L}(2\text{-}\textit{spcubes}, \mathbb{Q})$, $\mathcal{L}(2\text{-}\textit{cubes}, \mathbb{Q}^+)$, or $\mathcal{L}(2\text{-}\textit{clauses}, \mathbb{Q}^+)$. This covers every language mentioned in this paper except $\mathcal{L}(\textit{pforms}, \mathbb{Q}^+)$ and $\mathcal{L}(\textit{literals}, \mathbb{Q})$ and their sublanguages, which are all in P.

6 Conclusion

We have examined the properties of various goalbase languages for representing utility functions, concentrating on (1) the expressivity of different languages by characterizing the classes of utility functions they can represent; (2) the relative succinctness of pairs of languages over their expressive intersection; and (3) the computational complexity of finding the most preferred alternative when a utility function is encoded using a goalbase language (the MAX-UTIL problem). Our results can provide useful guidelines for application designers to help them select a preference representation language with the appropriate characteristics.

Concerning the complexity of the MAX-UTIL problem, we have been able to offer a complete picture by classifying *all* languages that can be defined in our framework as rendering MAX-UTIL either NP-complete or trivial. For expressivity our results are close to being complete: Many goalbase languages directly correspond to natural classes of utility functions, independently used in many different fields. Only for a handful of languages our results are either not complete characterizations (e.g., Theorem 3.10) or the characterizing property is somewhat artificial (e.g., Theorem 3.13). Finally, we have been able to establish a good number of relative succinctness results and we have presented different techniques for deriving such results. Here, however, there still remain some open questions. Maybe the most interesting of these concerns the relative effect, in terms of succinctness, of dropping either negation or one of the two binary connectives. For instance, it would be particularly interesting to establish the relative succinctness of, say, positive formulas and general cubes. Answering one or two of the remaining open questions would most likely allow us to resolve most of the others as well.

Our results cover languages that can be constructed by restricting the range of logical connectives permitted to a subset of $\{\neg, \wedge, \vee\}$. Other connectives one might want to consider are implication ($\rightarrow$), equivalence ($\leftrightarrow$), and exclusive disjunction (XOR). Interestingly, none of our results would be affected if we were (1) to enrich the syntax of positive (k-)formulas to permit also $\leftrightarrow$ and (2) to enrich the syntax of general (k-)formulas to involve any of these three connectives. For expressivity and complexity results, this is immediately clear. For succinctness results, this is also immediately clear as far as $\rightarrow$ is concerned, because translating implications into clauses does not change formula size. This is not the case for $\leftrightarrow$ and XOR. Still, also here our succinctness results are not affected, because all results involving either positive or general formulas state that these are at least as succinct as the language they are being compared to, which will still be the case after we have enriched the syntax.

We have mentioned several other frameworks for defining utility functions (and more generally, preference structures) in the introduction and in Section 2.2. A similar analysis as given in this paper (also covering expressivity, succinctness and complexity) of goalbase languages based on the aggregator $F = \max$, where the utility of an alternative is given by the weight of the most "important" goal it satisfies, is available elsewhere [17]. Lafage and Lang [16] discuss the definition of utility functions in terms of weighted goals from an axiomatic point of view, for different aggregators F. Coste-Marquis et al. [18] address expressivity, succinctness and complexity questions for ordinal preference structures generated by weighted goals. Nisan [12] gives a number of expressivity and succinctness results for OR/XOR bidding languages. Ieong and Shoham [21] use what is effectively $\mathcal{L}(\mathit{cubes}, \mathbb{R}, \Sigma)$ for representing the value of a coalition in coalitional games (a bundle is a coalition and the atoms are the coalition members), and provide some algorithms for computing core membership, core nonemptiness, and Shapley value. Elkind et al. [22] refine this work by showing that the Shapley value can be efficiently calculated for the sublanguage of $\mathcal{L}(\mathit{cubes}, \mathbb{R}, \Sigma)$ where all formulas are read-once.

There are several promising directions in which to develop this work further. One of these concerns resolving the remaining open succinctness questions, as mentioned above. Another interesting question concerns the complexity of choosing an alternative that is optimal for a *group* of agents. There are several variants of this problem. One would be to study the complexity of applying different voting rules to a profile of individual preferences, expressed in a compact representation language. Some initial research in this direction has been reported elsewhere, under the heading of *combinatorial vote* [5]. A second variant of the problem would be to partition the set of propositional variables amongst the agents. This induces one model for each of them (the variables "given" to an agent are set to *true*, all others to *false*). We can then investigate the complexity of finding a partition that maximizes *collective utility* for a group of agents. There are different interpretations of the notion of collective utility [28]. For instance, we may seek to find a partition that maximizes the sum of individual utilities (*utilitarianism*) or we may wish to choose a solution that leaves the weakest agent as well off as possible (*egalitarianism*). It is not difficult to see that most natural questions of this kind would give rise to problems for which the corresponding decision problem is certainly within NP. If the MAX-UTIL problem is already NP-hard, then hardness will naturally transfer to the collective optimization problem as well. So the most interesting questions here concern maximizing collective utility with respect to preference representation languages for which the individual optimization problem is easy. This question has been investigated elsewhere [17] for goalbase languages with the aggregator $F = \max$. Finally, it would be interesting to explore further the use of goalbase languages for the representation of coalitional games. Existing work [21, 22] has demonstrated their applicability in this context, but so far only a small number of languages have actually been used there.

References

[1] K. J. Arrow, A. K. Sen, and K. Suzumura (eds.), Handbook of Social Choice and Welfare (North-Holland, 2002).

[2] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet, A short introduction to computational social choice, in: Proc. 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM-2007), edited by J. van Leeuwen, G. F. Italiano, W. van der Hoek, C. Meinel, H. Sack, and F. Plasil, LNCS Vol. 4362 (Springer-Verlag, 2007), pp. 51–69.

[3] J. Doyle and M. P. Wellman, Preferential semantics for goals, in: Proc. 9th National Conference on Artificial Intelligence (AAAI-1991), (AAAI Press, 1991), pp. 698–703.

[4] 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 Artifcial Intelligence Research (JAIR) **21**, 135–191 (2004).

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

[6] F. Bacchus and A. J. Grove, Utility independence in a qualitative decision theory, in: Proc. 5th International Conference on Principles of Knowledge Representation and Reasoning (KR-1996), (Morgan Kaufmann Publishers, 1996), pp. 542–552.

[7] P. La Mura and Y. Shoham, Expected utility networks, in: Proc. 15th Conference on Uncertainty in Artificial Intelligence (UAI-1999), (Morgan Kaufmann, 1999), pp. 366–373.

[8] C. Boutilier, F. Bacchus, and R. Brafman, UCP-networks: A directed graphical representation of conditional utilities, in: Proc. 17th Conference on Uncertainty in Artificial Intelligence (UAI-2001), (Morgan Kaufmann, 2001), pp. 56–64.
[9] C. Gonzales and P. Perny, GAI networks for utility elicitation, in: Proc. 9th International Conference on Principles of Knowledge Representation and Reasoning (KR-2004), (AAAI Press, 2004), pp. 224–234.
[10] C. Boutilier, R. Dearden, and M. Goldszmidt, Exploiting structure in policy construction, in: Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI-1995), (Morgan Kaufmann, 1995), pp. 1104–1113.
[11] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie, Semiring-based CSPs and valued CSPs: Frameworks, properties and comparison, Constraints **4**(3), 199–240 (1999).
[12] N. Nisan, Bidding languages for combinatorial auctions, in: Combinatorial Auctions, edited by P. Cramton, Y. Shoham, and R. Steinberg (MIT Press, 2006).
[13] C. Boutilier and H. H. Hoos, Bidding languages for combinatorial auctions, in: Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-2001), (Morgan Kaufmann, 2001), pp. 1211–1217.
[14] T. W. Sandholm, Algorithm for optimal winner determination in combinatorial auctions, Artificial Intelligence **135**(1–2), 1–54 (2002).
[15] G. Pinkas, Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge, Artificial Intelligence **77**(2), 203–247 (1995).
[16] C. Lafage and J. Lang, Logical representation of preferences for group decision making, in: Proc. 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000), (Morgan Kaufmann Publishers, 2000), pp. 457–468.
[17] J. Uckelman and U. Endriss, Preference modeling by weighted goals with max aggregation, in: Proc. 11th International Conference on Principles of Knowledge Representation and Reasoning (KR-2008), (AAAI Press, 2008), pp. 579–587.
[18] S. Coste-Marquis, J. Lang, P. Liberatore, and P. Marquis, Expressive power and succinctness of propositional languages for preference representation, in: Proc. 9th International Conference on Principles of Knowledge Representation and Reasoning (KR-2004), (AAAI Press, 2004), pp. 203–212.
[19] P. Haddawy and S. Hanks, Representations for decision-theoretic planning: Utility functions for deadline goals, in: Proc. 4th International Conference on Principles of Knowledge Representation and Reasoning (KR-1994), (Morgan Kaufmann Publishers, 1992), pp. 71–82.
[20] F. Dupin de Saint-Cyr, J. Lang, and T. Schiex, Penalty logic and its link with Dempster-Shafer theory, in: Proc. 10th Conference on Uncertainty in Artificial Intelligence (UAI-1994), (Morgan Kaufmann Publishers, 1994), pp. 204–211.
[21] S. Ieong and Y. Shoham, Marginal contribution nets: A compact representation scheme for coalitional games, in: Proc. 6th ACM Conference on Electronic Commerce (EC-2005), (ACM Press, 2005), pp. 193–202.
[22] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge, A tractable and expressive class of marginal contribution nets and its applications, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), (IFAAMAS, 2008), pp. 1007–1014.
[23] G. C. Rota, On the foundations of combinatorial theory I: Theory of Möbius functions, Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete **2**(4), 340–368 (1964).
[24] M. Grabisch, k-order additive discrete fuzzy measures and their representation, Fuzzy Sets and Systems **92**(2), 167–189 (1997).
[25] V. Conitzer, T. W. Sandholm, and P. Santi, Combinatorial auctions with k-wise dependent valuations, in: Proc. 20th National Conference on Artificial Intelligence (AAAI-05), (AAAI Press, 2005), pp. 248–254.
[26] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet, Multiagent resource allocation in k-additive domains: Preference representation and complexity, Annals of Operations Research **163**(1), 49–62 (2008).
[27] J. S. Rosenschein and G. Zlotkin, Rules of Encounter (MIT Press, 1994).
[28] H. Moulin, Axioms of Cooperative Decision Making (Cambridge University Press, 1988).
[29] A. P. Dempster, Upper and lower probabilities induced by a multivaluated mapping, Annals of Mathematical Statistics **38**(2), 325–339 (1967).
[30] G. Shafer, A Mathematical Theory of Evidence (Princeton University Press, Princeton, NJ, 1976).
[31] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf, Space efficiency of propositional knowledge representation formalisms, Journal of Artificial Intelligence Research (JAIR) **13**, 1–31 (2000).
[32] Y. Mansour, Learning Boolean functions via the Fourier transform, in: Theoretical Advances in Neural Computation and Learning, (Kluwer, 1994).
[33] L. Trevisan, Lecture notes 25, CS278: Computational complexity, UC-Berkeley, December 1 2004, `http://www.cs.berkeley.edu/~luca/cs278/notes/lecture25.pdf`.
[34] T. Lee, Kolmogorov Complexity and Formula Size Lower Bounds, PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, 2006.
[35] J. Uckelman and A. Witzel, Logic-based preference languages with intermediate complexity, in: Proceedings of the 4th Multidisciplinary Workshop on Advances in Preference Handling (MPREF-2008), (AAAI Press, Chicago, 2008), pp. 123–127.
[36] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness (W.H. Freeman and Co., 1979).
[37] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, Complexity and Approximation (Springer-Verlag, 1999).