UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Compactly representing utility functions using weighted goals and the max aggregator

Uckelman, J.; Endriss, U.

DOI
10.1016/j.artint.2010.07.003

Publication date
2010

Document Version
Submitted manuscript

Published in
Artificial Intelligence

Link to publication

Citation for published version (APA):
Uckelman, J., & Endriss, U. (2010). Compactly representing utility functions using weighted goals and the max aggregator. *Artificial Intelligence*, *174*(15), 1222-1246. https://doi.org/10.1016/j.artint.2010.07.003

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:09 Mar 2023

Compactly Representing Utility Functions Using Weighted Goals and the Max Aggregator[☆]

Joel Uckelman[1], Ulle Endriss

Institute for Logic, Language and Computation, University of Amsterdam

Abstract

Weighted propositional formulas can be used to model preferences over combinatorial domains: each formula represents a goal we would like to see satisfied, the weight of a formula represents the importance of the goal in question, and to assess the desirability of a given alternative we aggregate the weights of the goals satisfied by that alternative. One of several options is to aggregate by using the maximum of the weights of the satisfied goals. This approach gives rise to a family of preference representation languages, one for each of a range of possible restrictions we can impose on either formulas or weights. We analyze the properties of these languages and establish results regarding their expressivity, and absolute and relative succinctness. We also study the computational complexity of the problem of finding the best and the worst alternative for a given set of weighted goals, and of finding an alternative that is optimal for a group of agents, for a range of different notions of collective optimality proposed in social choice theory and welfare economics.

Keywords: preference representation, preference aggregation

1. Introduction

1.1. Motivation & Background

Preference handling is a problem of central importance in Artificial Intelligence [3]. For example, recommender systems need to elicit and maintain a representation of the user's preferences and multiagent systems are often modeled as collections of decision-theoretic agents, each of which are guided by their own preferences. Possibly the most fundamental question in this context is how to best *represent* the preferences of an artificial agent or a human user. Designing suitable languages for representing preferences is particularly challenging when the alternatives over which an agent expresses preferences have a combinatorial structure. For example, in the context of combinatorial auctions [4] or other resource allocation problems [5], the number of bundles of goods an agent may obtain is exponential in the number of goods under discussion, so being able to express preferences over this exponentially large space of alternatives in a compact manner is crucial.

In this paper we study a particular family of languages for representing cardinal preferences over combinatorial domains that are Cartesian products of several binary domains. Expressing a *cardinal preference* means specifying a (utility or valuation) function mapping each alternative to a number reflecting the degree of preference for that alternative. For comparison, *ordinal preferences* are relations over pairs of alternatives specifying whether one is preferable to the other. The use of utility functions to model preferences is appropriate in some circumstances, and inappropriate in others. In the context of combinatorial auctions, for instance, modeling preferences as utility functions is appropriate: in such an auction each bidder needs to express how much they would be prepared to pay for a given

[☆]This paper is based on and extends work presented at the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR-2008) [1], and additionally presents material from [2].

Email addresses: j.d.uckelman@uva.nl (Joel Uckelman), ulle.endriss@uva.nl (Ulle Endriss)

[1]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).

bundle of goods [4]. From the representational point of view, this amounts to specifying a function from the space of bundles to numbers, i.e., to specifying a cardinal preference structure.

The family of preference representation languages we study is based on weighted propositional formulas, or *weighted goals*. For a given set of propositional variables, alternatives correspond to models for propositional formulas. A utility function mapping such alternatives to the reals can then be represented as a set of weighted goals, as follows: To compute the utility of a given alternative X, we first collect all the weights of the goals satisfied by X and then we aggregate those weights to obtain a single number, the utility of X. One natural aggregation function is Σ; in this case, the utility of X is the sum of the weights of the goals satisfied by X. An other option is to use the max operator; in this case the utility of X is the maximum of the weights of the goals satisfied by X.

Using weighted formulas for preference representation is an idea which originated in penalty logic [6]. Penalty logic addresses the problem of how to make inferences from an inconsistent knowledge base by augmenting the formulas in a knowledge base with weights, which indicate the cost of falsifying the associated formula; then the inference problem reduces to considering what is valid over only the minimal-cost consistent subsets of a given knowledge base. Here, costs are summed: That is, the cost of rejecting φ and ψ is the sum of their weights. Lafage and Lang [7] generalize penalty logic to permit aggregators other than sum, and also introduce a distance-based aggregator. Coste-Marquis et al. [8] take a different approach: Rather than using goalbases directly for representing cardinal preferences, they use goalbases to underpin various ordinal preference relations.

Chevaleyre et al. [9] shifted from considering the weights of unsatisfied formulas as penalties to considering the weights of satisfied formulas as utilities, and their work contains initial results on goalbases aggregated with sum; later work extends this to systematically cover most naturally-defined sum-aggregated goalbase languages [10]. Recently, this framework has been further extended from propositional logic to description logic [11, 12].

1.2. Our Contribution

As mentioned, most previous work has concentrated on the weights of unsatisfied goals or on *sum languages*, where weights are aggregated via Σ. In this paper we explore in depth the properties of the next most important family of languages that can be constructed in this framework, namely the *max languages*, in which the aggregation function used is max. The properties we study concern the expressivity, the succinctness, and the complexity of max languages.

Expressivity. We establish correspondence results which show what classes of utility functions can be represented by the most natural representatives of the family of the max languages. For instance, if the only logical connective allowed is conjunction, then we can represent the monotone utility functions, and only those.

Succinctness. We rank the most important max languages in terms of how compactly they can represent those utility functions that they are able to represent. Informally, language $\mathcal{L}$ is at least as succinct as $\mathcal{L}'$ if the increase in size when translating representations from $\mathcal{L}'$ to $\mathcal{L}$ is polynomially bounded. For instance, conjunctions of literals induce a strictly more succinct language than conjunctions of atoms if only positive weights are permitted, but an equally succinct language if both positive and negative weights can be used. We also provide results concerning bounds on the absolute succinctness of some languages, and we study whether languages have unique or multiple representations for the same utility function.

Complexity. We first address two natural problems that arise in the context of reasoning about the preferences of a single agent when these are represented using a max language. Here we study the computational complexity of (the decision variants of) the problem of finding the highest and the lowest utility that such an agent may experience. It turns out that the complexity strongly depends on the language chosen; some problems can be solved in linear time while others are coNP-complete. We also study the complexity of collective utility maximization, the problem of finding the "best" partitioning of the set of propositional variables amongst a group of agents (inducing a model for each of them). There are a number of different ways of aggregating the utilities of the members of a group to define what is best for that group. We analyze the complexity of the problem for several of the standard notions of collective utility, familiar from the social choice and welfare economics literature, such as utilitarian social welfare, egalitarian social welfare, and the Nash product [13].

1.3. Related Work

Similar questions have previously been addressed for other preference representation languages.

CP-nets [14] are a tool for representing conditional ordinal preferences over combinatorial domains (rather than utility functions, on which we focus in this paper). As with any individual preference representation formalism, we may wish to aggregate individual preferences into group preferences. Various methods for aggregating the CP-nets of multiple agents have been tried by, e.g., Rossi et al. [15] and Lang and Xia [16]; the analogous problem for us, aggregating the goalbases of multiple agents, we take up in Section 8. The complexity of answering ordering and dominance queries is also studied in the CP-nets literature [17]; we do not study these questions for goalbases because finding the utility of a given model for an agent is always computationally trivial.

Generalized additive (GA) decomposition is a way of representing utility functions as sums of "smaller" utility functions, each of which has as its domain some proper subset of the domain of the larger utility function. Originally proposed by Fishburn [18], GA-decomposition has more recently been used by Gonzales and Perny [19] for constructing GAI-nets, and by Brafman et al. [20] for eliciting preferences using GA-decomposable CP- and TCP-nets. Goalbase languages may be seen as particular way of GA-decomposing utility functions.

MC-nets [21] are a method of representing the utilities which accrue to coalitions of players in coalitional games. Ieong and Shoham [21] consider the expressive power and succinctness of MC-nets, as well as the complexity of answering questions about the core of the coalitional game being represented. MC-nets and the goalbase language $\mathcal{L}(\mathit{cubes}, \mathbb{R}, \Sigma)$ (the language of arbitrarily-weighted conjunctions of literals, aggregated by summing weights of satisfied formulas) are essentially notational variants of each other, applied to different problems. Elkind et al. [22] extend MC-nets to permit arbitrary formulas and prove some succinctness and complexity results for these languages.

Bidding languages—cardinal preference representation languages designed for expressing the values of bundles of goods—are a popular topic in the combinatorial auctions literature. The expressivity, succinctness, and complexity of the XOR and OR bidding languages have been studied extensively [23]. The XOR language is equivalent to our language $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max)$ (the language of positively-weighted conjunctions of atoms, aggregated by taking the maximum weight of all satisfied formulas). Boutilier and Hoos [24] have proposed two logic-based bidding languages (which are in some respects similar to our sum languages), and Boutilier [25] experimentally compared these two languages to see how their use affected runtime when solving the Winner Determination Problem for auctions.

Coste-Marquis et al. [8] use goalbases to induce various kinds of preference orderings, prove expressivity results about which preorders are expressible under which methods of constructing preorders from goalbases, and show whether it is possible to translate the underlying goalbases from one language to another without exponential blowup. These results are similar in concept to our succinctness results, though we view goalbases cardinally rather than ordinally.

1.4. Overview of This Paper

The remainder of this paper is structured as follows. Basic definitions concerning the framework of representing utility functions via weighted goals are provided in Section 2. Section 3 contains our expressivity results. Following some general remarks on succinctness in Section 4, Section 5 presents results on absolute succinctness and the uniqueness property and Section 6 is devoted to the issue of relative succinctness. Then, complexity results for reasoning about the preferences of individual agents are provided in Section 7, while our results on the complexity of collective utility maximization can be found in Section 8. Finally, Section 9 concludes with a brief discussion of some of the remaining open problems in the area.

2. Preliminaries

In this section we present definitions and notation used throughout this paper, as well as our basic framework for representing utility functions. First, we define the structures we need from propositional logic:

Definition 1 (Propositional Formulas)**.** The set $\mathcal{PS}$ is a fixed, finite set of propositional variables. We write $\mathcal{PS}_n$ to indicate that $|\mathcal{PS}| = n$. Let $\mathcal{L}_{\mathcal{PS}}$ be the language of propositional logic over $\mathcal{PS}$ and the logical constants $\top$ and $\bot$, closed under the Boolean connectives $\neg$, $\wedge$, and $\vee$.

The technical results found here apply to formulas that contain only the connectives $\neg$, $\wedge$, and $\vee$. We omit $\rightarrow$ (implication) as a Boolean connective because is it succinctly definable in terms of $\neg$ and $\vee$. Equivalence and XOR we also do not consider here, though their inclusion might result in more succinct languages.[2]

Definition 2 (Propositional Models)**.** A *model* is a set $M \subseteq \mathcal{PS}$. The satisfaction relation $\models$ for models and formulas is defined in the usual way for propositional logic:

$$M \models \top. \qquad M \models \neg\varphi \quad \text{iff} \quad M \not\models \varphi.$$
$$M \not\models \bot. \qquad M \models \varphi \wedge \psi \quad \text{iff} \quad M \models \varphi \text{ and } M \models \psi.$$
$$M \models p \quad \text{iff} \quad p \in M. \qquad M \models \varphi \vee \psi \quad \text{iff} \quad M \models \varphi \text{ or } M \models \psi.$$

We give names to some types of propositional formulas:

Definition 3 (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.

We abbreviate the class of general, unrestricted formulas to *forms*. 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.[3] Note that by convention $\bigwedge \emptyset = \top$ 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 $\top$).

Definition 4 (State Formulas)**.** If $X \subseteq \mathcal{PS}$, then define $\bar{X} = \mathcal{PS} \setminus X$, and $\neg X = \{\neg p \mid p \in X\}$. Then $\bigwedge(M \cup \neg\bar{M})$ is the *state formula* corresponding to the model M.

For example, if $\mathcal{PS} = \{a, b, c, d\}$, then the state formula for the model $\emptyset$ is $\neg a \wedge \neg b \wedge \neg c \wedge \neg d$ and for $\{a, b\}$ is $a \wedge b \wedge \neg c \wedge \neg d$. Notice that $M' \models \bigwedge(M \cup \neg\bar{M})$ iff $M = M'$.

We are interested in utility functions 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).

Utility functions are a typical method for specifying cardinal preferences:

Definition 5 (Utility Functions)**.** A *utility function* is a mapping $u\colon 2^{\mathcal{PS}} \rightarrow \mathbb{R}$.

Because the utility functions we consider have sets as their domain, and propositional models are sets, utility functions can be thought of as mapping models to their values.

We note here some properties of utility functions to which we make frequent reference:

Definition 6 (Properties of Utility Functions)**.** Suppose that u is a utility function. Then:

- u is *normalized* iff $u(\emptyset) = 0$.

[2] The XOR we refer to here is *logical* XOR, which is distinct from the so-called XOR in XOR bidding languages [23] and valuative XOR [24], both of which behave like max and operate on weights, rather than formulas.

[3] 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 have finite length.

- u is *nonnegative* iff $u(X) \geq 0$ for all X.
- u is *monotone* iff $u(X) \geq u(Y)$ for all $X \supseteq Y$.
- u is *modular* iff $u(X \cup Y) = u(X) + u(Y) - u(X \cap Y)$ for all X, Y.
- u is a *unit-demand* valuation iff $u(X) = \max_{a \in X} u(\{a\})$ and u is normalized.
- u is a *simple unit-demand* valuation iff $u(X) = 1$ for all $X \neq \emptyset$ and u is normalized.

Definition 7 (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. We will see in Section 3 that for the languages studied here this restriction does not affect expressive power, though the presence of unsatisfiable formulas can affect the computational complexity of some decision problems, as discussed in Section 7. When a particular goalbase is under consideration, we write w_φ to mean the weight of formula φ in that goalbase. For(G) is the set of formulas in G. Var(φ) is the set of propositional variables in the formula φ and Var$(G) = \bigcup_{\varphi \in \text{For}(G)} \text{Var}(\varphi)$. A formula $(\varphi, w_\varphi) \in G$ is *active* in a model M when $M \models \varphi$ and for all other $(\psi, w_\psi) \in G$ such that $M \models \psi$, $w_\psi \leq w_\varphi$.

Given a goalbase and a model, we may easily determine which weighted formulas the model makes true. Now, we would like to combine the weights of the satisfied formulas to find the value of the model. A function which combines weights must map a multiset of reals (the collected weights of the satisfied formulas) to a real (the value of the model). We call such functions *aggregation functions*, as they aggregate the many weights of satisfied formulas into the single value of the model.[4]

Definition 8 (Generated Utility Functions)**.** A goalbase G and an *aggregation function* $F \colon \mathbb{N}^{\mathbb{R}} \to \mathbb{R}$ *generate* a utility function $u_{G,F}$ mapping each model $M \subseteq \mathcal{PS}$ to $u_{G,F}(M) = F(w \mid (\varphi, w) \in G \text{ and } M \models \varphi)$.

Because multisets are unordered structures, any aggregation function will necessarily be associative and commutative over weights.[5] Natural aggregation functions to consider include Σ and max. In this paper, we restrict ourselves to the aggregation function max, the maximum function. When $F = \max$, the utility function generated from a goalbase G is

$$u_{G,\max}(M) = \max_{\substack{(\varphi,w) \in G \\ M \models \varphi}} w.$$

In other words, the value of a model is the same as the largest weight had by any formula which is true in that model. For example, if $\mathcal{PS} = \{p, q, r\}$, then the goalbase $G = \{(\top, 0), (p \vee q \vee r, 1)\}$ generates the utility function $u \colon X \mapsto \min(1, |X|)$. (This is easy to see by observing that $\top$ is the only true formula in the model $\emptyset$, so it sets the value of u_G there; both $\top$ and $p \vee q \vee r$ are true in every other model, but $p \vee q \vee r$ has a higher weight, so it sets the value of u_G in all models except $\emptyset$.)

Different aggregation functions may produce dramatically different utility functions from the same goalbase. E.g., if $G = \{(a, 1) \mid a \in \mathcal{PS}\}$, then $u_{G,\max}$ is the simple unit-demand utility function ($u(X) = 1$ if $X \neq \emptyset$, 0 otherwise) while $u_{G,\Sigma}$ is the simple additive utility function ($u(X) = |X|$). We assume that $\max(\emptyset) = -\infty$; it is often useful to include, say, $(\top, 0)$ in any goalbase intended for use with max so as to obtain utility functions with finite values for all models.

Definition 9 (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 $u_{G,F} = u_{G',F}$.

[4] The definition of aggregation function given here differs subtly from that given elsewhere [7, 10, 26], in that all of these write the aggregation function as $F \colon 2^{\mathbb{R}} \to \mathbb{R}$ when in practice the reader is meant to understand the aggregation function as operating on multisets. In particular, these authors often write sums of weights as $\sum\{w \mid \textit{stuff}\}$, when what is intended is $\sum_{\textit{stuff}} w$. We have striven to avoid this ambiguity in the present work; in the event that we have failed, please in all cases read sums of weights as sums of *multisets* (rather than sets) of weights. That is, $\sum\{1, 1\} = 2 \neq 1$.

[5] If the domain were arbitrary-length tuples of reals instead of multisets of reals ($\mathbb{R}^*$ instead of $\mathbb{N}^{\mathbb{R}}$), then there could be aggregators which are sensitive to the order in which formula weights are aggregated. Since goalbases are unordered structures, there is no compelling reason to be concerned with the order in which weights are aggregated, so we limit the domains of aggregators to multisets.

Goalbases provide a framework for defining 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 give 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.)

Definition 10 (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 all 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 $\mathcal{L}(\Phi, W, F)$.

Regarding weights, we study the restriction to the positive reals ($\mathbb{R}^+$) as well as the general case ($\mathbb{R}$). For complexity questions we will restrict our attention to the rationals ($\mathbb{Q}$). We restrict formulas by their structure, according to the types of formula defined in Definition 3. For example, the language $\mathcal{L}(\mathit{cubes}, \mathbb{R}^+, \max)$ consists of all goalbases which contain only positively-weighted cubes, and are aggregated using max. Many more examples of languages will be seen in Section 3, where we investigate language expressivity.

Finally, a note on non-binary domains: Due to the applications we have in mind for goalbase languages, resource allocation, auctions, voting—all binary in the sense that an agent has an item or does not, or a candidate is a winner or not—we have restricted our variables to have binary domains. Moreover, restriction to binary domains is a natural one when working with propositional logic. Nonetheless, it is possible to simulate in our framework variables which take on a larger (but still finite) set of values, by coding single many-valued variables into multiple binary-valued ones.

3. Expressivity

An important feature of any preference representation language is the range of preferences which can be represented in it. This available range is known as the *expressivity* of the language; this section is devoted to determining the expressivity of various max goalbase languages.

From the point of view of the theorist, we want to know how expressive goalbase languages are, both for their own sake and because these expressivity results are necessary for proving succinctness results in Section 6. As a user of goalbase languages, knowing the expressivity of goalbase languages can help us choose the language which is best suited for our application. 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 utility functions. A correspondence result shows that the utility functions representable in a particular language are exactly those comprising a particular class of utility functions. We provide correspondence results for eight max languages, and partial correspondence results for another four.

We begin by establishing some simple results regarding equivalences between languages, which allow us to narrow the range of languages to be considered in the remainder of the section. We then characterize the expressivity of the most important (distinct) languages.

3.1. Goalbase Equivalences

We are interested in comparing languages generated by different types of goalbases, in particular those generated from the following restrictions on formulas: *literals*, *cubes*, *clauses*, and *general* formulas using both conjunction and disjunction; as well as *positive* formulas and those including negation. Regarding weights, we want to consider *positive* and *general weights*. This gives rise to $4 \cdot 2 \cdot 2 = 16$ different languages. Here we establish several equivalences amongst languages and thereby show that we actually only need to consider a subset of all the languages that can be defined in this manner. Furthermore, if we are interested only in monotone utility functions, then we can further reduce the range of languages to consider.

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

Theorem 11. $G \cup \{(\varphi_1 \vee \cdots \vee \varphi_n, w)\} \equiv_{\max} G \cup \{(\varphi_i, w) \mid 1 \leq i \leq n\}$.

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

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

This tells us that with max as our aggregator, disjunctions as main connectives do not contribute to a language's expressivity. In fact, as any formula has an equivalent representation in disjunctive normal form, this tells us that disjunction can *never* increase the expressive power of a max language. In particular, from Theorem 11 we get the following equivalences between languages:

Corollary 12. *Fix $W \subseteq \mathbb{R}$. Then:*

1. $\mathcal{U}(\mathit{pclauses}, W, \max) = \mathcal{U}(\mathit{atoms}, W, \max)$,
2. $\mathcal{U}(\mathit{clauses}, W, \max) = \mathcal{U}(\mathit{literals}, W, \max)$,
3. $\mathcal{U}(\mathit{pforms}, W, \max) = \mathcal{U}(\mathit{pcubes}, W, \max)$,
4. $\mathcal{U}(\mathit{forms}, W, \max) = \mathcal{U}(\mathit{cubes}, W, \max)$.

Proof. For $\mathcal{U}(\mathit{forms}, W, \max) = \mathcal{U}(\mathit{cubes}, W, \max)$: Suppose that $(\varphi, w) \in G$. Without loss of generality, assume that $\varphi = \psi_1 \vee \cdots \vee \psi_n$ is in DNF. By Theorem 11, we may replace $(\psi_1 \vee \cdots \vee \psi_n, w)$ by $(\psi_1, w), \ldots, (\psi_n, w)$ and preserve goalbase equivalence. Repeating this for each original formula in G converts G to the language $\mathcal{L}(\mathit{cubes}, W, \max)$, since each ψ_i is a cube and the weights were left unchanged.

We use the same argument for each of the other cases, noting that the same transformation reduces a positive formula (in DNF) to a set of positive cubes, a clause to a set of literals, and a positive clause to a set of atoms. □

The same is *not* true for sum languages. E.g., under summation clauses are more expressive than literals: For $W = \mathbb{R}$, clauses can express *all* utility functions, while literals can express only modular functions [10, Corollary 3.8]. For each of the equivalences in Corollary 12, there is a set of weights W which violates it under summation.

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

Theorem 13. *Fix G. Let X^+ be a set of positive literals, and X^- a set of negative literals, such that no atom appears in both. If $u_{G \cup \{(\bigwedge X^+ \cup X^-, w)\}, \max}$ is monotone, then*

$$G \cup \left\{\left(\bigwedge X^+ \cup X^-, w\right)\right\} \equiv_{\max} G \cup \left\{\left(\bigwedge X^+, w\right)\right\}.$$

Proof. There are two cases to consider: models which are supersets of X^+, and models which are not.

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

Therefore, in all models M we have that $u_{G \cup \{(\bigwedge X^+ \cup X^-, w)\}, \max}(M) = u_{G \cup \{(\bigwedge X^+, w)\}, \max}(M)$. □

The following result shows that we can further reduce the range of languages to consider if we limit ourselves to monotone utility functions. It follows immediately from Theorem 13 and Corollary 12. (Recall that $\bigwedge \emptyset = \top$.)

Corollary 14. *Let* Mono *be the class of monotone utility functions. Fix $W \subseteq \mathbb{R}$. Then:*

1. $\mathcal{U}(\mathit{clauses}, W, \max) \cap \textsc{Mono} = \mathcal{U}(\mathit{atoms} + \top, W, \max) \cap \textsc{Mono}$.
2. $\mathcal{U}(\mathit{forms}, W, \max) \cap \textsc{Mono} = \mathcal{U}(\mathit{pcubes}, W, \max) \cap \textsc{Mono}$.

3.2. Correspondences

Corollary 12 tells us that the interesting languages, expressivity-wise, are those based on cubes, positive cubes, literals, and atoms. We prove that cubes are expressively complete for the full range of utility functions and that positive cubes correspond to the class of monotone functions:

Theorem 15. $\mathcal{U}(\mathit{cubes}, \mathbb{R}, \max)$ *is the class of all utility functions.*

Proof. Given a utility function u, define

$$G = \left\{ \left(\bigwedge X \cup \neg \bar{X}, u(X) \right) \;\middle|\; X \subseteq \mathcal{PS} \right\}$$

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

Theorem 16. $\mathcal{U}(\mathit{pcubes}, \mathbb{R}, \max)$ *is the class of monotone utility functions.*

Proof. ($\Rightarrow$) Suppose that $G \in \mathcal{U}(\mathit{pcubes}, \mathbb{R}, \max)$ but u_G is not monotone. So there are models $M \subset M'$ such that $u_G(M') < u_G(M)$. Then there is a $(\varphi, u_G(M)) \in G$ which is active in M such that $M \models \varphi$ but $M' \not\models \varphi$. Since $M' \supset M$, then there is some $a \in M' \setminus M$ for which $\varphi \models \neg a$. Therefore, φ is not a positive formula, which contradicts the hypothesis that φ is a pcube.

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

Theorem 17. $\mathcal{U}(\mathit{atoms}, \mathbb{R}, \max)$ *is the class of unit-demand utility functions.*

Proof. Suppose that u is a unit-demand valuation, which by definition means that $u(X) = \max_{a \in X} u(\{a\})$. Construct a $G \in \mathcal{U}(\mathit{atoms}, \mathbb{R}, \max)$ such that $G = \{(a, w) \mid a \in \mathcal{PS}, u(\{a\}) = w\}$. Then

$$u(X) = \max_{a \in X} u(\{a\}) = \max_{(a,w) \in G} w = u_{G,\max}(X).$$

Conversely, suppose that $G \in \mathcal{U}(\mathit{atoms}, \mathbb{R}, \max)$, and note that the same series of equivalences holds. □

We are not aware of a property of utility functions referred to in the literature which would characterize $\mathcal{U}(\mathit{literals}, \mathbb{R}, \max)$. The desired property is a generalization of the unit-demand valuation that also allows us to specify a value for *not* receiving a particular item.

By restricting the set of weights W we can capture classes of utility functions with a particular range. For instance, the class $\mathcal{U}(\mathit{pcubes}, \mathbb{R}^+, \max)$, is the class of nonnegative monotone functions. This class is known to be equal to $\mathcal{U}(\mathit{pforms}, \mathbb{R}^+, \Sigma)$ [10, Corollary 3.12].[6] This is a case where a syntactically simple language is more expressive with max than with sum.

On the other hand, some very simple classes of utility functions are hard to capture in structurally simple languages using max aggregation. For instance, recall that a utility function u is called *modular* iff $u(M \cup M') = u(M) + u(M') - u(M \cap M')$ for all $M, M' \in 2^{\mathcal{PS}}$. Modular functions are nicely captured by $\mathcal{U}(\mathit{literals}, \mathbb{R}, \Sigma)$ [10, Corollary 3.8]. However, there is no natural restriction to formulas that would allow us to characterize the modular functions under max aggregation. On the contrary, among the max languages considered here, only $\mathcal{L}(\mathit{cubes}, \mathbb{R}, \max)$ can express all modular functions, and this language is so powerful that it can actually express *all* utility functions.

We use the fact that $\mathcal{L}(k\text{-}\mathit{pcubes}, \mathbb{R}, \max)$ misses some modular utility functions when $k < |\mathcal{PS}|$ to show that the expressivity of $\mathcal{L}(k\text{-}\mathit{pcubes}, W, \max)$ and $\mathcal{L}(k\text{-}\mathit{cubes}, W, \max)$ strictly increases with k:

Theorem 18. *For all $k > j$,*

[6] To be precise, $\mathcal{U}(\mathit{pcubes}, \mathbb{R}^+, \max) = \mathcal{U}(\mathit{pforms}, \mathbb{R}^+, \Sigma)$ over total functions only. Because the max languages can also express partially defined functions returning $-\infty$ for some models while sum languages cannot, if we expand our consideration to partially-defined utility functions, then $\mathcal{U}(\mathit{pcubes}, \mathbb{R}^+, \max) \supset \mathcal{U}(\mathit{pforms}, \mathbb{R}^+, \Sigma)$.

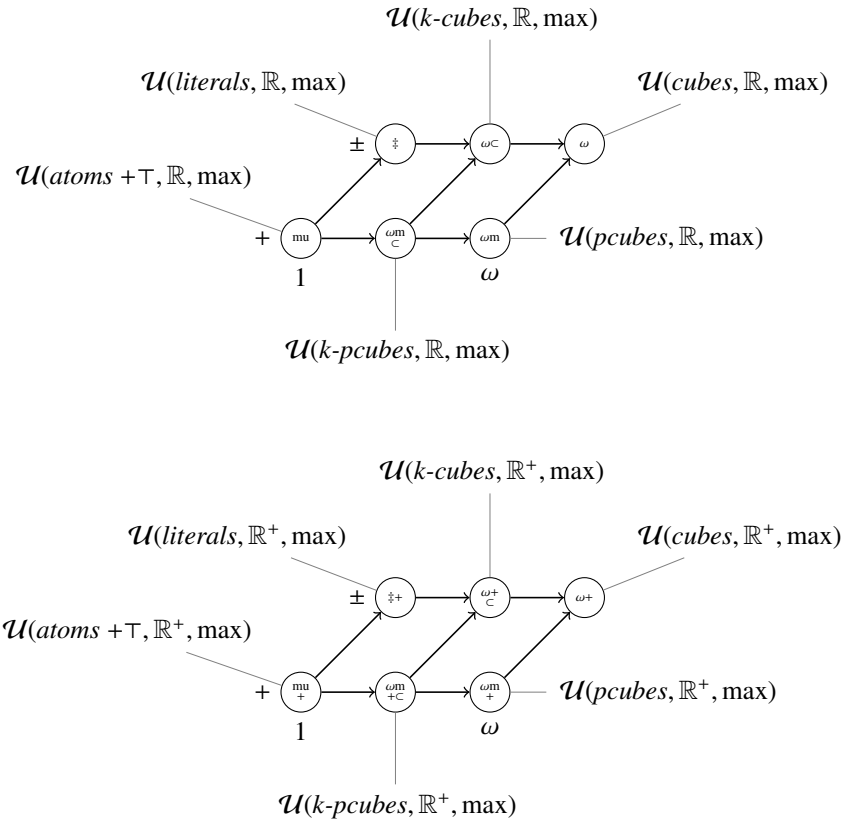

Figure 1: Summary of expressivity results.

1. $\mathcal{U}(k\text{-}pcubes, W, \max) \supset \mathcal{U}(j\text{-}pcubes, W, \max)$*, and*
2. $\mathcal{U}(k\text{-}cubes, W, \max) \supset \mathcal{U}(j\text{-}cubes, W, \max)$.

Proof. (1) The nonstrict part follows by inclusion. For the strict part, we show that $\mathcal{U}((k+1)\text{-}pcubes, W, \max) \setminus \mathcal{U}(k\text{-}pcubes, W, \max)$ is nonempty because it includes a family of modular utility functions: Suppose that u is modular and nonnegative, and at least $k+1$ singleton models $\{p_1\}, \ldots, \{p_{k+1}\}$ have nonzero value. Then there is no $G \in \mathcal{L}(k\text{-}pcubes, W, \max)$ such that $u(\{p_1, \ldots, p_{k+1}\}) = u_G(\{p_1, \ldots, p_{k+1}\})$. This is the case because with max as our aggregator $u_G(\{p_1, \ldots, p_{k+1}\})$ would necessarily be the weight of some k-pcube and by assumption if $X \subset \{p_1, \ldots, p_{k+1}\}$ then $u(X) < u(\{p_1, \ldots, p_{k+1}\})$. However, if u gives exactly $k+1$ singleton models nonzero value, then u is representable in $\mathcal{L}((k+1)\text{-}pcubes, W, \max)$ by the goalbase

$$\left\{\left(\bigwedge X, u(X)\right) \,\middle|\, X \subseteq \mathcal{PS} \text{ and } |X| \leq k+1\right\}.$$

(2) The same modular utility functions missing from $\mathcal{L}(k\text{-}pcubes, W, \max)$ are missing from $\mathcal{L}(k\text{-}cubes, W, \max)$, due to the fact that the addition of negation to the language is not helpful for representing monotone utility functions (see Theorem 13). □

Our correspondence results are summarized in Figure 1. In the figure, each node represents one language we examined, 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:

ω	ω-additive (general)	m	monotone
+	nonnegative	u	unit-demand
‡	general unit-demand	$\subset$	proper subset of

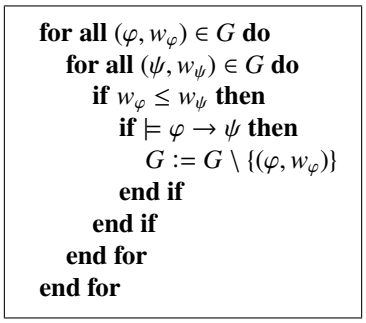

Figure 2: An algorithm for removing superfluous formulas from a goalbase G.

Where $\subset$ 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 (decreasing into the page) is the *positivity* axis and has two steps: 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. Note that several languages do not appear in the figure due to being expressively equivalent to some language which does appear there. For max languages, this is due to the fact expressed in Corollary 12, namely that disjunction does not contribute expressivity to max languages.

4. Succinctness: Terminological Interlude

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

For max languages, some weighted goals in a goalbase may never contribute to the utility of an alternative. Recall that a formula $(\varphi, w_\varphi) \in G$ is *active* in a model M when $M \models \varphi$ and for all other $(\psi, w_\psi) \in G$ such that $M \models \psi$, $w_\psi \leq w_\varphi$. Goalbases may sometimes contain formulas which fail to contribute value to the goalbase in any model. We call these formulas *superfluous*. Equivalence is preserved if superfluous formulas are removed from a goalbase one at a time: If $(\varphi, w) \in G$ is a superfluous formula, then $G \equiv G \setminus \{(\varphi, w)\}$. However, removal of all superfluous formulas simultaneously is not necessarily equivalence-preserving, and which formulas are superfluous must be reevaluated after any single formula is removed from a goalbase. For example, all formulas in $\{(a, 1), (\neg\neg a, 1)\}$ are superfluous under max; however, removing both produces the empty goalbase, which is not equivalent to the original. For max languages, contradictions are always superfluous, but for any other weighted formula (even formulas with zero weight) there is some max goalbase where it is not superfluous.

Fact 19. *Fix $(\varphi, w_\varphi) \in G$, and the aggregator as* max. *Then, if (φ, w_φ) is superfluous, for every model M where (φ, w_φ) is active, the set $\{(\psi, w_\psi) \in G \mid M \models \psi$ and w_ψ is maximal$\}$ is not a singleton.*

Since superfluous formulas are useless, they may be removed without harm. We can remove superfluous formulas from any $G \in \mathcal{L}(forms, \mathbb{R}, \max)$ as in Figure 2. Two aspects of this algorithm are significant: First, we remove superfluous formulas one at a time. Second, this algorithm is not efficient, as it requires a quadratic number of calls to an UNSAT oracle; we would like to do better than a coNP algorithm. In the previous section, we saw that the only max languages which are (expressively) interesting are based on cubes. This fact greatly reduces the complexity of deciding whether a formula is superfluous, since deciding whether one cube implies another is polynomial (see the proof of Theorem 60 for the algorithm).

Next, we note a useful fact relating superfluity to activity:

Fact 20. *Under the max aggregator, if G contains no superfluous formulas then every $(\varphi, w) \in G$ has a model in which it is uniquely active.*

Some goalbase languages have a unique way of expressing a given utility function, while others allow for several alternative representations. Here we make this notion of uniqueness precise:

Definition 21 (Unique Representations)**.** 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 $\mathrm{For}(G) \subseteq \Phi$, $u_G = u$, and G contains no superfluous formulas. A language $\mathcal{L}$ is said to have *unique representations* if every u represented in $\mathcal{L}$ is uniquely represented.

Any language which has unique representations can be thought of as minimal in the sense that any further restriction on permissible weighted formulas will lead to a reduction in expressivity. Effectively this means that the set of representatives of formula equivalence classes forms a minimal basis for the vector space in which the goalbases live.

Definition 22 (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, $\lceil \log w \rceil$ bits). The *size of a goalbase* G, written as $\mathrm{size}(G)$, is the sum of the sizes of the weighted goals in G.

Observe that the size of a goalbase may differ from its cardinality: If $G = \{(a \wedge b, 1)\}$, then $\mathrm{size}(G) = 2$ (or 3, if we are not neglecting the bit used for storing the weight) while $|G| = 1$.

Definition 23. A goalbase $G \in \mathcal{L}(\Phi, W, F)$ is *minimal* if for all $G' \in \mathcal{L}(\Phi, W, F)$ such that $G' \equiv_F G$, $\mathrm{size}(G) \leq \mathrm{size}(G')$.

The difficulty of recognizing whether G is minimal, or of finding a G which is a minimal representative of some utility function u, is strongly dependent on what $\mathcal{L}$ and u are. In the general case, formulas in a minimal goalbase will be pairwise nonequivalent, but this is not a sufficient condition for minimality. For some languages (such as $\mathcal{L}(\mathit{atoms}, \mathbb{R}, \max)$) and some classes of utility functions (e.g., modular) we can detect minimality and generate minimal representations easily, while for many richer languages how to do this is not obvious or unknown.

For max languages, in a minimal goalbase no formula is implied by any formula with a smaller or equal weight; for sum languages, formulas in a minimal goalbase will be pairwise non-equivalent.

5. Absolute Succinctness and Uniqueness

In this section we find (absolute) bounds on the sizes of goalbases for the two most expressive and useful max languages, $\mathcal{L}(\mathit{pcubes}, \mathbb{R}, \max)$ and $\mathcal{L}(\mathit{cubes}, \mathbb{R}, \max)$. The bounds we derive here imply bounds for sublanguages of these as well, including all max languages we consider in this paper. For $\mathcal{L}(\mathit{pcubes}, \mathbb{R}, \max)$ and its sublanguages, we calculate exact sizes of goalbases; for $\mathcal{L}(\mathit{cubes}, \mathbb{R}, \max)$, we give upper and lower bounds which are sometimes tight—though for sublanguages there is the possibility of tighter bounds. Along the way, we prove that $\mathcal{L}(\mathit{pcubes}, \mathbb{R}, \max)$ has unique representations.

The range of a utility function u ($\mathrm{ran}\, u$)—that is, the set of values it may take on—strongly influences the size of representations that utility function may have in any max language:

Theorem 24. *For any goalbase G, $|G| \geq |\mathrm{ran}\, u_{G,\max}|$.*

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

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

By Theorem 24 we have that if $|G_n|$ is polynomial then $|\mathrm{ran}\, u_{G_n}|$ is polynomial. However, the converse does not hold: Let

$$G_n = \left\{ \left(\bigwedge X, 1 \right) \;\middle|\; |X| = \frac{n}{2} \right\} \cup \{(\top, 0)\}.$$

Here, $|\mathrm{ran}\, u_{G_n}| = 2$ but $|G_n| = \binom{n}{n/2} + 1$ is superpolynomial and G_n is minimal in $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max)$.

There is a clear connection between superfluity and goalbase minimality, namely that if G is a minimal goalbase for a utility function $u \in \mathcal{U}(\Phi, W, \max)$, then G contains no superfluous formulas. However, the converse does not hold for $\mathcal{L}(forms, \mathbb{R}, \max)$: $\{(p, 1), (\neg p, 1)\}$ and $\{(\top, 1)\}$ represent the same utility function $u(X) = 1$, yet neither formula in the larger goalbase is superfluous.

Recall from Definition 21 that a language can have unique representations, meaning that it is sufficiently restrictive as to have exactly one minimal representation of each representable utility function. Several sum languages have unique representations, as discussed in [2, Section 3.4.2] and [10, Section 3.2]. This also occurs for at least one max language:

Theorem 25. *$\mathcal{L}(pcubes, \mathbb{R}, \max)$ has unique representations.*

Proof. Fix $u \in \mathcal{U}(pcubes, \mathbb{R}, \max)$. Let $G_0 = \emptyset$. While $u_{G_i,\max} \neq u$: Choose a least model X for which $u_{G_i,\max}(X) \neq u(X)$. (By *least*, we mean that $|X|$ is minimal.) Let $G_{i+1} = G_i \cup \{(\bigwedge X, u(X))\}$. Call G the G_i at which this algorithm terminates. (An example of a run of this algorithm appears after this proof.)

Correctness: $u_{G,\max} = u$ because each iteration ends with one more model correct than in the previous iteration, and there are finitely many models. Setting a weight for $\bigwedge X$ cannot disturb the value of any model $Y \subset X$, as X is the least model where $\bigwedge X$ is true, and cannot prevent us from correctly setting the value of any model $Z \supset X$ during subsequent iterations, because by Theorem 16 u is monotone. Note also that the order of choice of cubes of the same size makes no difference in the outcome.

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

We give an example of run of the algorithm used in the proof above: Given the utility function

$$\begin{array}{llll} u(\emptyset) = 0 & u(\{b\}) = 0 & u(\{a, b\}) = 2 & u(\{b, c\}) = 3 \\ u(\{a\}) = 1 & u(\{c\}) = 2 & u(\{a, c\}) = 2 & u(\{a, b, c\}) = 10 \end{array}$$

the algorithm constructs the corresponding goalbase G as follows: $G_0 = \emptyset$ by definition. $u_{G_0,\max}(\emptyset) = -\infty \neq u(\emptyset) = 0$, and $\emptyset$ is the unique smallest incorrect model, so we let $G_1 = G_0 \cup \{(\top, 0)\}$. Now $\{a\}$ and $\{c\}$ are the least models made incorrect by $u_{G_1,\max}$; we choose to correct $\{a\}$ first. (Recall that the order in which models of the same size are corrected does not affect the result.) Let $G_2 = G_1 \cup \{(a, 1)\}$. Continuing in this fashion, we will correct the models $\{c\}$, $\{a, b\}$, $\{b, c\}$, and $\{a, b, c\}$:

$$\begin{array}{llll} G_0 = \emptyset & G_2 = G_1 \cup \{(a, 1)\} & G_4 = G_3 \cup \{(a \wedge b, 2)\} & G_6 = G_5 \cup \{(a \wedge b \wedge c, 10)\} \\ G_1 = G_0 \cup \{(\top, 0)\} & G_3 = G_2 \cup \{(c, 2)\} & G_5 = G_4 \cup \{(b \wedge c, 3)\} & \end{array}$$

After constructing G_6, we see that there are no further incorrect models, i.e., $u_{G_6,\max}(X) = u(X)$ for all $X \subseteq \mathcal{PS}$, so $G = G_6$ is the unique representation of u in $\mathcal{L}(pcubes, \mathbb{R}, \max)$.

Note that while this algorithm does show how to construct the minimal representation for any representable utility function in $\mathcal{L}(pcubes, W, \max)$, it is not an efficient algorithm for finding representations, as it requires us to check exponentially many models in order to set weights for them.

Next, we derive upper and lower bounds for the size of representations in $\mathcal{L}(cubes, W, \max)$, but first we prove several technical lemmas which we will need. For the remainder of this section, we assume that all $u_{G,\max}$ are total. First, recall that *active* formulas in max-aggregated goalbases are those which have a weight equal to the value of some model where they are true.

Lemma 26. *Fix a goalbase $G \in \mathcal{L}(cubes, W, \max)$ and a $(\varphi, w) \in G$. If a model X has an extension $Y \supset X$ such that $u_{G,\max}(Y) < u_{G,\max}(X)$ and (φ, w) is active in X, then φ is not a positive cube.*

Proof. Suppose otherwise. Let (φ, w) be such that $X \models \varphi$, $u_{G,\max}(X) = w$, φ a pcube, and $u_{G,\max}(Y) < u_{G,\max}(X)$. Then for all $Y \supset X$ it follows that $Y \models \varphi$ because φ is a monotone formula. So $u_{G,\max}(Y) \geq w = u_{G,\max}(X)$, contrary to hypothesis. □

In words: If a cube is active in a model which can decline in value when extended, then there must be a negative literal in that cube.

Here we define the $X\uparrow$ notation for denoting the set of extensions of X, which is used throughout the remainder of this section:

Definition 27 (Upsets)**.** If X is a model, then $X\uparrow = \{Y \mid X \subseteq Y \subseteq \mathcal{PS}\}$.

The set of models $2^{\mathcal{PS}}$ may be thought of as a Boolean lattice; then $X\uparrow$ is the sublattice rooted at X. Alternatively, $X\uparrow$ may be thought of as all of the ways of extending X.

Lemma 28. *Suppose that w is the minimum value of any model in $X\uparrow$. Let $(\varphi_1, w), \ldots, (\varphi_k, w) \in G \in \mathcal{L}(cubes, W, \max)$ be the formulas which are true in at least one model in $X\uparrow$, false outside of $X\uparrow$, and have weight w. Let $G' = G \setminus \{(\varphi_i, w)\}_{1 \leq i \leq k} \cup \{(\bigwedge X, w)\}$. Then:*

1. $G' \equiv_{\max} G$.
2. $\text{size}(G') \leq \text{size}(G)$.

Proof. For 1, we must show that the changes made to G to get G' result in no models being disturbed from their original values. The formula $\bigwedge X$ is true exactly in $X\uparrow$ and nowhere else, so it disturbs no models outside of $X\uparrow$. Adding $(\bigwedge X, w)$ disturbs no models in $X\uparrow$, since $(\bigwedge X, w)$ is inactive in any Y where $u(Y) > w$, and provides the correct value in the remaining models in $X\uparrow$ since w is minimal there. Every $\varphi_i \models \bigwedge X$, so if $M \models \varphi_i$ then $M \models \bigwedge X$ also, which covers all models where a φ_i was active.

For 2: $\bigwedge X$ is not longer than $\varphi_1, \ldots, \varphi_k$: $\bigwedge X$ is the shortest formula which is true only in $X\uparrow$. Each φ_i is true only in $X\uparrow$ also, so $\text{size}(\varphi_i) \geq \text{size}(\bigwedge X)$, and therefore $\sum_{i=1}^{k} \text{size}(\varphi_i) \geq \text{size}(\bigwedge X)$ (strictly larger if $k > 1$). □

This lemma permits us to reduce iteratively any goalbase in $\mathcal{L}(cubes, W, \max)$: Let $G = G_0$. Apply the reduction to the smallest sublattice $X\uparrow$ of G_i to which it has not yet been applied, and let the result be G_{i+1}. (Starting from the smallest sublattice means starting with $\mathcal{PS}$ as the root and working our way downwards to $\emptyset$.) At some stage i, we will reach a fixed point—that is, $G_i = G_\infty$—where no further applications of the reduction will have any effect. (The upper bound for reaching a fixed point happens to be $i = 2^{|\mathcal{PS}|}$, though all we require here is that it happens after finitely many applications of the reduction.) Let $G' = G_\infty$, and call such a G' *pcubes-minimal*.

Since this reduction is never size-increasing, we may make use of it to observe a useful fact about the formulas in minimal goalbases in $\mathcal{L}(cubes, W, \max)$:

Lemma 29. *For every $G \in \mathcal{L}(cubes, W, \max)$, there is a minimal $G' \equiv_{\max} G$ such that $(\bigwedge X, w) \in G'$ iff $w = \min_{Z \in X\uparrow} u(Z)$ and $u(Y) < w$ for every $Y \subset X$.*

Proof. Suppose that $G'' \equiv_{\max} G$ and G'' is minimal. Let G' be the result of exhaustively applying the reduction in Lemma 28 to G''. Since the reduction is equivalence-preserving and size-reducing, $G' \equiv_{\max} G''$ and $\text{size}(G') \leq \text{size}(G'')$. Since G'' was already minimal, G' cannot be smaller, so $\text{size}(G') = \text{size}(G'')$ and G' is also minimal.

($\Rightarrow$) Suppose that $(\bigwedge X, w) \in G'$. Then $(\bigwedge X, w)$ was not eliminated by the reduction. Since $\bigwedge X$ is a positive cube, it is true exactly in $X\uparrow$ and nowhere else. If there were some model $Y \subset X$ for which $u(Y) \geq w$, then the reduction would have eliminated $(\bigwedge X, w)$ and replaced it with $(\bigwedge Y, w)$ instead, so it must be the case that $u(Y) < w$ for all $Y \subset X$. For the other condition, suppose that there is a model $Z \in X\uparrow$ such that $u(Z) < w$. Since $Z \models \bigwedge X$ and max is our aggregator, it follows that $u(Z) \geq w$, which is a contradiction.

($\Leftarrow$) Suppose that $w = \min_{Z \in X\uparrow} u(Z)$ and $u(Y) < w$ for every $Y \subset X$. Since w is the minimal model value in $X\uparrow$, it follows that there is a model $Z \supseteq X$ for which $u(Z) = w$. In order for $u(Z) = w$, we need a formula $(\varphi, w) \in G'$ such that $Z \models \varphi$. Because $u(Y) < w$ for all $Y \subset X$, it must also be the case that $Y \not\models \varphi$ for all $Y \subset X$. The only formula which meets both of these requirements which could have survived the reduction is $(\bigwedge X, w)$, since any longer formula would have been replaced by $(\bigwedge X, w)$ and any shorter formula would either be true in some model $Y \subset X$ or fail to be true in Z. □

Note that w is not necessarily equal to $u(X)$ here—in fact, if u_G is nonmonotone, then w will frequently *not* be $u(X)$. For example, if

$$u(X) = \begin{cases} 2 & \text{if } X = \emptyset \\ 1 & \text{otherwise,} \end{cases}$$

then over $\mathcal{PS} = \{p, q\}$ the goalbase $\{(\top, 1), (\neg p \wedge \neg q, 2)\}$ represents u in the language $\mathcal{L}(\mathit{cubes}, \mathbb{R}, \max)$ and is pcubes-minimal, yet the minimal weight in $\emptyset\uparrow$, which is 1, does not equal $u(\emptyset) = 2$.

Lemma 29 is crucial for the bounds we will derive below, as it tells us exactly which positive cubes we will find in a minimal goalbase which is also pcubes-minimal.

Furthermore, from Lemma 29, we have the following special case:

Lemma 30. *For every $G \in \mathcal{L}(\mathit{cubes}, W, \max)$, there is a minimal $G' \equiv_{\max} G$ such that $(\top, \min_{X \in 2^{\mathcal{PS}}} u(X)) \in G'$.*

Proof. $\min_{X \in 2^{\mathcal{PS}}} u(X) = \min_{Z \in \emptyset\uparrow} u(Z)$, and $\emptyset$ has no proper subsets. □

It is not always that case that a G' which results from the reduction under discussion is uniquely minimal. E.g.,

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

can be represented as either $\{(\top, 0), (a, 1)\}$ or $\{(\neg a, 0), (a, 1)\}$, which are the same size.

Now we attempt to calculate bounds on size(G) when $G \in \mathcal{L}(\mathit{cubes}, W, \max)$.

Lemma 31. *Let $G \in \mathcal{L}(\mathit{cubes}, W, \max)$ be minimal. Let $G^+ \subseteq G$ be the set of pcubes in G. Then*

$$\text{size}(G^+) = \sum \Big\{ \max(1, |X|) \;\Big|\; \neg\exists Y \subset X \textit{ s.t. } u_{G,\max}(Y) \geq \min_{Z \in X\uparrow} u_{G,\max}(Z) \Big\}.$$

Proof. By Lemma 29 we may, without loss of generality, assume that

$$G^+ = \Big\{ \Big(\bigwedge X, w \Big) \;\Big|\; \neg\exists Y \subset X \text{ s.t. } u(Y) \geq \min_{Z \in X\uparrow} u(Z) \Big\}.$$

Therefore

$$\text{size}(G^+) = \sum \Big\{ \max(1, |X|) \;\Big|\; \neg\exists Y \subset X \text{ s.t. } u_{G,\max}(Y) \geq \min_{Z \in X\uparrow} u_{G,\max}(Z) \Big\}.$$

Note that we must have $\max(1, |X|)$ instead of simply $|X|$, due to the fact that $|\emptyset| = 0$ but $\text{size}(\bigwedge \emptyset) = \text{size}(\top) = 1$. □

Using this and a result from Section 3, we can derive the exact size for every minimal G in the monotone portion of $\mathcal{L}(\mathit{cubes}, W, \max)$:

Theorem 32. *If $G \in \mathcal{L}(\mathit{cubes}, W, \max)$, G is minimal, and $u_{G,\max}$ is monotone, then*

$$\text{size}(G) = \sum \Big\{ \max(1, |X|) \;\Big|\; \neg\exists Y \subset X \textit{ s.t. } u_{G,\max}(Y) \geq u_{G,\max}(X) \Big\}.$$

Proof. Since $u_{G,\max}$ is monotone, we know by Theorem 13 that removal of negative literals is equivalence-preserving. Since G is minimal, it follows that the negative literals are already gone, so every formula in G is a positive cube. Hence $G = G^+$, so we have from Lemma 31 that

$$\text{size}(G) = \sum \Big\{ \max(1, |X|) \;\Big|\; \neg\exists Y \subset X \text{ s.t. } u_{G,\max}(Y) \geq \min_{Z \in X\uparrow} u_{G,\max}(Z) \Big\}.$$

Finally, notice that because $u_{G,\max}$ is monotone, $\min_{Z \in X\uparrow} u_{G,\max}(Z) = u_{G,\max}(X)$, which permits us to simplify the condition. □

Now we turn to the case where $u_{G,\max}$ is nonmonotone. For any such minimal G, we know by Lemma 31 the precise size of G^+, the positive subset of G, so all that remains is to derive bounds for G^-, the subset of G containing negative literals. First, we derive bounds for the size of G^-.

Lemma 33. *Let $G \in \mathcal{L}(cubes, W, \max)$ be minimal. Let G^- be the set of cubes in G containing negative literals. Then*

$$\left|\{p \in \mathcal{PS} \mid u_{G,\max}(X) > u_{G,\max}(X \cup \{p\})\}\right| \leq \text{size}(G^-) \leq |\mathcal{PS}| \cdot \left|\left\{X \;\middle|\; \exists Y \subset X \text{ s.t. } u(Y) \geq \min_{Z \in X\uparrow} u(Z)\right\}\right|.$$

Proof. By Lemma 26, if there are models X and $X \cup \{p\}$ where $u_{G,\max}(X \cup \{p\}) < u_{G,\max}(X)$, then there must be a formula (φ, w) active in model X that contains $\neg p$. These are the formulas which comprise G^-.

The best case is that every such $\neg p$ appears in G^- exactly once, which gives us the lower bound. The worst case is to write a state formula $\bigwedge X \cup \neg\bar{X}$ to cover each pair of model $X, X \cup \{p\}$ over which there is a decline in value; every such state formula has length $|\mathcal{PS}|$. □

Now we have all of the pieces necessary for exhibiting upper and lower bounds on the size of goalbases in $\mathcal{L}(cubes, W, \max)$ which represent nonmonotone utility functions.

Theorem 34. *If $G \in \mathcal{L}(cubes, W, \max)$, G is minimal, and $u_{G,\max}$ is nonmonotone, then*

$$\text{size}(G) \geq \sum\left\{\max(1, |X|) \;\middle|\; \neg\exists Y \subset X \text{ s.t. } u_{G,\max}(Y) \geq \min_{Z \in X\uparrow} u_{G,\max}(Z)\right\} + \left|\{a \in \mathcal{PS} \mid u_{G,\max}(X) > u_{G,\max}(X \cup \{a\})\}\right|.$$

Proof. Recall that $G = G^+ \cup G^-$. Lemma 31 gives us the exact size of G^+, while Lemma 33 gives us a lower bound for the size of G^-. □

Theorem 35. *If $G \in \mathcal{L}(cubes, W, \max)$, G is minimal, and $u_{G,\max}$ is nonmonotone, then*

$$\text{size}(G) \leq \sum\left\{\max(1, |X|) \;\middle|\; \neg\exists Y \subset X \text{ s.t. } u_{G,\max}(Y) \geq \min_{Z \in X\uparrow} u_{G,\max}(Z)\right\} + |\mathcal{PS}| \cdot \left|\left\{X \;\middle|\; \exists Y \subset X \text{ s.t. } u_{G,\max}(Y) \geq \min_{Z \in X\uparrow} u_{G,\max}(Z)\right\}\right|$$

Proof. Recall that $G = G^+ \cup G^-$. Lemma 31 gives us the exact size of G^+, while Lemma 33 gives us an upper bound for the size of G^-. □

Note that neither the upper nor lower bounds given here are tight in general, though for each bound we do have an example of a goalbase for which one of the bounds is tight. For the upper bound, consider the utility function

$$u(X) = |X| \bmod 2,$$

the parity function on $\mathcal{PS}$. When $\mathcal{PS} = \{a, b, c\}$, the goalbase

$$\{(\top, 0), (a \wedge b \wedge c, 1), (a \wedge \neg b \wedge \neg c, 1), (\neg a \wedge b \wedge \neg c, 1), (\neg a \wedge \neg b \wedge c, 1)\}$$

is minimal for $u(X)$. The lower bound here is 7, the upper bound is 13, and the actual size is also 13. For the lower bound, consider the utility function

$$u(X) = \begin{cases} 1 & \text{if } a \notin X \\ 0 & \text{otherwise,} \end{cases}$$

for which the goalbase $\{(\top, 0), (\neg a, 1)\}$ is minimal over $\mathcal{PS} = \{a, b, c\}$. In this case, both the actual size and lower bound are 2, while the upper bound is again 13. Finally, the utility function $u(X) = 3 - |X|$ is represented minimally over $\mathcal{PS} = \{a, b, c\}$ by the goalbase

$$\left\{\begin{array}{c} (\top, 0), \\ (\neg a, 1), (\neg b, 1), (\neg c, 1), \\ (\neg a \wedge \neg b \wedge c, 2), (\neg a \wedge b \wedge \neg c, 2), (a \wedge \neg b \wedge \neg c, 2), \\ (\neg a \wedge \neg b \wedge \neg c, 3) \end{array}\right\}$$

which has size 16, but hits neither the lower nor the upper bound, which are 4 and 22, respectively.

Clearly these bounds could be refined by further analyzing the composition of G^- for various nonmonotone utility functions, but at present what exactly the differences are among the three examples—what causes one to hit the lower bound, one to hit the upper bound, and one to hit neither—is not apparent to us. Furthermore, the bounds themselves are not easy to compute; here again, some additional insight into the structure of such functions might be of use. We leave these issues for future work.

6. Relative Succinctness

Frequently one language contains shorter representations of some utility functions than another language does. Here we give a definition of relative succinctness (used also in [10] and similar to those given in [8, 27]) to make this notion precise. 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 counterintuitive 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 36 (Relative 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 $f\colon \mathcal{L}(\Phi, W, F) \to \mathcal{L}(\Psi, W', F')$ 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 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 $\mathcal{U} = \{u_{G,F} \mid G \in \mathcal{L}\} \cap \{u_{G',F'} \mid G' \in \mathcal{L}'\}$, which is the *expressive intersection* of $\mathcal{L}$ and $\mathcal{L}'$. Later, in Section 6.2, we illustrate some circumstances in which being explicit about the class of comparison is important.

Finding the succinctness relation between some pairs of goalbase languages is trivial, as when one language in a pair is a sublanguage of the other, or when the two languages have no expressive overlap. These cases can be dismissed without argument. Recall from Definition 21 that a goalbase language may have unique representations: If a utility function is representable in the language, then there is exactly one representation of it in the language. If $\mathcal{L}$ has unique representations and $\mathcal{L} \not\succ \mathcal{L}'$ is true, then we can show that $\mathcal{L} \not\succ \mathcal{L}'$ as follows:

Proof strategy. We present a family of utility functions $\mathcal{U}$, and construct a (smallish, but not necessarily optimal) representation $G'_u \in \mathcal{L}'$ for each $u \in \mathcal{U}$. Then, we construct a representation $G_u \in \mathcal{L}$ for each $u \in \mathcal{U}$ where at least one G_u is exponentially larger than its corresponding G'_u. Because we know that $\mathcal{L}$ has unique representations, we know that we can't find a smaller (or any other!) representation of u in $\mathcal{L}$, so we have shown that $\mathcal{L} \not\succ \mathcal{L}'$. □

This is a handy proof strategy, one which we shall make use of in the proofs of Theorems 42, 50, and 51.

Often we consider families of utility functions $\{u_n\}_{n\in\mathbb{N}}$, where for each n we have that $|\mathcal{PS}| = n$. 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. Every family of utility functions considered here has weights which are independent of n, so we disregard the size of the weights in our succinctness results. (Superexponential growth in weights affects all languages equally.)

Here we state some basic properties of the succinctness relation, which we use frequently in our proofs, often without reference.

Fact 37. *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 $\mathcal{L}_1 \prec \mathcal{L}_3$.*

Note that Fact 37.2 is useful contrapositively also, for deriving $\not\preceq$ results for superlanguages. For Fact 37.3, 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 $\mathcal{L}_2 \prec \mathcal{L}_3$.

For certain languages, due to the limited length of formulas which may appear in minimal goalbases, it will be the case that we need not distinguish between growth of $\text{size}(G)$ and growth of $|G|$.

Lemma 38. *Let $\mathcal{L}$ be a goalbase language. For each $n \in \mathbb{N}$, let φ_n be the longest formula in $\mathcal{L}$ in n variables with no shorter equivalent. Then if there is a polynomial p such that* $\text{size}(\varphi_n) \in O(p(n))$*, it follows that for all families of minimal goalbases $\{G_n\}_{n\in\mathbb{N}} \subseteq \mathcal{L}$,* $\text{size}(G_n)$ *is polynomial in n iff $|G_n|$ is polynomial in n.*

Proof. ($\Rightarrow$) This direction is immediate, since it is impossible to form exponentially many formulas from polynomially-many atom instances.

($\Leftarrow$) Suppose that $|G_n| = p(n)$ is a polynomial. The longest formula φ_n in $\mathcal{L}$ in n variables has $\text{size}(\varphi_n) = q(n)$, for some polynomial q. The worst case is that each of the $p(n)$ formulas in G_n has size $q(n)$, for a total size of $p(n) \cdot q(n)$, which is still polynomial in n. □

In particular, this means that $\text{size}(G)$ and $|G|$ are interchangeable in terms of growth in $|\mathcal{PS}|$ for languages such as $\mathcal{L}(\textit{cubes}, W, F)$ and $\mathcal{L}(\textit{clauses}, W, F)$, but not necessarily for languages such as $\mathcal{L}(\textit{forms}, W, F)$. We use this lemma implicitly in several of our relative succinctness results.

Now we derive a simple succinctness result which applies to all languages which permit formulas of no more than a fixed, finite length.

Theorem 39. *For any fixed $k \in \mathbb{N}$, arbitrary set of formulas Ψ, and arbitrary sets of weights W, W' and aggregator F: If $\Phi \subseteq$ k-forms, then $\mathcal{L}(\Phi, W, F) \geq \mathcal{L}(\Psi, W', F)$.*

Proof. There are only $O(n^k)$ formulas of length k or less, and so any utility function u representable in $\mathcal{L}(\Phi, W, F)$ cannot have a representation more than polynomially larger than the best one in $\mathcal{L}(\textit{forms}, \mathbb{R}, F)$. Hence, $\mathcal{L}(\Phi, W, F) \geq \mathcal{L}(\textit{forms}, \mathbb{R}, F)$. Furthermore, $\mathcal{L}(\textit{forms}, \mathbb{R}, F) \supseteq \mathcal{L}(\Psi, W', F)$, so by Fact 37.2 we have that $\mathcal{L}(\Phi, W, F) \geq \mathcal{L}(\Psi, W', F)$. □

As a consequence, any two languages with bounded-length formulas and the same aggregator are equally succinct over their expressive intersection. Put another way, there are no interesting questions involving relative succinctness between pairs of languages in which both impose a fixed bound on formula length.

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

Fact 40. *For succinctness relations $\odot \in \{\preceq, \prec, \sim, \perp\}$:*

$$\mathcal{L}(\Phi, W, \max) \odot \mathcal{L}(\Psi, W', \max) \iff \mathcal{L}(\Phi, W \cap W', \max) \odot \mathcal{L}(\Psi, W \cap W', \max).$$

6.1. Succinctness Between Max Languages

Next, we show that all pcubes and cubes languages are equally succinct when using max for our aggregator.

Theorem 41. *For all $j, k \in \mathbb{N} \cup \{\omega\}$ and $W, W' \subseteq \mathbb{R}$:*

1. $\mathcal{L}(j\textit{-pcubes}, W, \max) \sim \mathcal{L}(k\textit{-pcubes}, W', \max)$.
2. $\mathcal{L}(j\textit{-pcubes}, W, \max) \sim \mathcal{L}(k\textit{-cubes}, W', \max)$.
3. $\mathcal{L}(j\textit{-cubes}, W, \max) \sim \mathcal{L}(k\textit{-cubes}, W', \max)$.

Proof. When at least one of $j, k \in \mathbb{N}$, all three cases follow immediately from Theorem 39. We must give a proof when $j = k = \omega$, but here the first and third cases trivialize, so all that remains is to show that $\mathcal{L}(\textit{pcubes}, W, \max) \sim \mathcal{L}(\textit{cubes}, W', \max)$. By Fact 40, we may assume without loss of generality that $W = W'$. $\mathcal{L}(\textit{pcubes}, W, \max)$ expresses only monotone utility functions, and Theorem 13 ensures that any representation containing negative literals may be reduced to a shorter, equivalent one by simply deleting the negative literals; the result of such a deletion is in $\mathcal{L}(\textit{pcubes}, W, \max)$. Hence the minimal representations for any u representable in both $\mathcal{L}(\textit{pcubes}, W, \max)$ and $\mathcal{L}(\textit{cubes}, W', \max)$ will be the same, which proves that $\mathcal{L}(\textit{pcubes}, W, \max) \sim \mathcal{L}(\textit{cubes}, W', \max)$. □

Recall that Theorem 11 shows that disjunctions as main connectives do not affect succinctness, because the translation required to eliminate disjunction does not affect the size of a goalbase. However, the same is not necessarily true for disjunctions which occur within the scope of other connectives. Therefore, for our analysis of expressivity of max languages in Section 3 we could safely ignore disjunction, but for succinctness we cannot, as the next result demonstrates.

Theorem 42. $\mathcal{L}(pcubes, \mathbb{R}^+, \max) \prec \mathcal{L}(pforms, \mathbb{R}^+, \max)$.

Proof. We have that $\mathcal{L}(pcubes, \mathbb{R}^+, \max) \preceq \mathcal{L}(pforms, \mathbb{R}^+, \max)$ because every pcube is a positive formula. For strict succinctness: The family of utility functions represented by

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

in $\mathcal{L}(pforms, \mathbb{R}^+, \max)$ grows linearly with n. The same family may be represented in $\mathcal{L}(pcubes, \mathbb{R}^+, \max)$ by

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

which has size 2^{n-1} for any (even) n. By Theorem 25, $\mathcal{L}(pcubes, \mathbb{R}^+, \max)$ has unique representations, which establishes the minimality of this representation. □

More generally, the same argument shows that for any intersecting sets of weights W, W', $\mathcal{L}(pcubes, W, \max) \prec \mathcal{L}(pforms, W', \max)$—so long as there is some $w \in W \cap W'$, we may use that for the formula weights instead of using 1 as we do in the proof—and also that $\mathcal{L}(pcubes, W, \max) \prec \mathcal{L}(forms, W, \max)$ and $\mathcal{L}(cubes, W, \max) \prec \mathcal{L}(pforms, W, \max)$, by virtue of Theorem 13.

When dealing with negation-containing goalbases for monotone utility functions, we might wish to put all nonpositive formulas in a standard form in order to simplify working with them. Negation normal form is a way of doing this. A formula φ is in *negation normal form* (NNF) if all occurrences of negation apply to atoms only. Any formula may be rewritten to an equivalent formula in NNF without an increase in size, by recursive application of De Morgan's Laws and elimination of double negation.

Before proceeding, we define a notion of uniform substitution for formulas and goalbases:

Definition 43 (Uniform Substitution)**.** If $\varphi, \psi_1, \ldots, \psi_k$ are formulas and $p_1, \ldots, p_k \in \mathcal{PS}$, then $\varphi[\psi_1/p_1, \ldots, \psi_k/p_k]$ is the result of (simultaneously) substituting ψ_i for every occurrence of p_i in φ. If G is a goalbase, then $G[\psi_1/p_1, \ldots, \psi_k/p_k]$ is the result of applying the substitution to each $(\varphi, w) \in G$.

In the following, we will abuse notation by writing $\varphi[\top/\neg p]$ for the less perspicuous $\varphi[\bot/p]$. Once we have all formulas in a monotone max goalbase translated to NNF, we may apply the following equivalence in order to remove the negative literals altogether:

Lemma 44. *If $u_{G,\max}$ is monotone and every formula in G is in NNF, then $G[\top/\neg p_1, \ldots, \top/\neg p_n] \equiv_{\max} G$.*

Proof. To show that $G[\top/\neg p_1, \ldots, \top/\neg p_n] \equiv_{\max} G$, it suffices to show for a single $(\varphi, w) \in G$ that $(G \setminus \{(\varphi, w)\}) \cup \{(\varphi[\top/\neg p], w)\} \equiv_{\max} G$, as we can then repeat the process for every other formula and every other $p \in \mathcal{PS}$.

Fix a $(\varphi, w_\varphi) \in G$ which has $\neg p$ as a subformula. Suppose that no model M where $p \notin M$ is such that $M \models \varphi$ but $M \cup \{p\} \not\models \varphi$. In this case, switching p from false to true will never make φ false if had been true, so we may safely replace all occurrences of $\neg p$ by $\top$. That is, $\models \varphi[\top/\neg p] \leftrightarrow \varphi$ and we are done.

Otherwise, let M be such a model. Since φ is in NNF, $\varphi[\top/\neg p]$ will remain true in every model where φ was true; but there is additionally the possibility that $M \cup \{p\} \models \varphi[\top/\neg p]$ when $M \not\models \varphi$. However, since $u_{G,\max}$ is monotone and $M \models \varphi$, we know that $u_{G,\max}(M \cup \{p\}) \geq w_\varphi$, and so there must already be some $(\psi, w_\psi) \in G$ such that $M \cup \{p\} \models \psi$ and $w_\psi = u_{G,\max}(M \cup \{p\})$. Therefore, making it so that $M \cup \{p\} \models \varphi[\top/\neg p]$ will not disturb the value of the utility function there (or in any other model). Hence, $(G \setminus \{(\varphi, w)\}) \cup \{(\varphi[\top/\neg p], w)\} \equiv_{\max} G$. □

With this lemma in hand, we now improve on Theorem 13 and show that there is no succinctness gain from using negation in arbitrary formulas, not just cubes, when representing monotone utility functions in max languages.

	$\mathcal{L}(pcubes, \mathbb{R}^+, \max)$	$\mathcal{L}(atoms + \top, \mathbb{R}^+, \max)$	$\mathcal{L}(pforms, \mathbb{R}^+, \max)$	$\mathcal{L}(cubes, \mathbb{R}^+, \max)$	$\mathcal{L}(literals, \mathbb{R}^+, \max)$	$\mathcal{L}(forms, \mathbb{R}^+, \max)$	$\mathcal{L}(pcubes, \mathbb{R}, \max)$	$\mathcal{L}(atoms + \top, \mathbb{R}, \max)$	$\mathcal{L}(pforms, \mathbb{R}, \max)$	$\mathcal{L}(cubes, \mathbb{R}, \max)$	$\mathcal{L}(literals, \mathbb{R}, \max)$	$\mathcal{L}(forms, \mathbb{R}, \max)$
$\mathcal{L}(forms, \mathbb{R}, \max)$	$\succ$	$\sim$	$\sim$	$\succ$	$\sim$	$\sim$	$\succ$	$\sim$	$\sim$	$\succ$	$\sim$	$\sim$
$\mathcal{L}(literals, \mathbb{R}, \max)$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	
$\mathcal{L}(cubes, \mathbb{R}, \max)$	$\sim$	$\sim$	$\prec$	$\sim$	$\sim$	$\prec$	$\sim$	$\sim$	$\prec$	$\sim$		
$\mathcal{L}(pforms, \mathbb{R}, \max)$	$\succ$	$\sim$	$\sim$	$\succ$	$\sim$	$\sim$	$\succ$	$\sim$	$\sim$			
$\mathcal{L}(atoms + \top, \mathbb{R}, \max)$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$				
$\mathcal{L}(pcubes, \mathbb{R}, \max)$	$\sim$	$\sim$	$\prec$	$\sim$	$\sim$	$\prec$	$\sim$					
$\mathcal{L}(forms, \mathbb{R}^+, \max)$	$\succ$	$\sim$	$\sim$	$\succ$	$\sim$	$\sim$						
$\mathcal{L}(literals, \mathbb{R}^+, \max)$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$							
$\mathcal{L}(cubes, \mathbb{R}^+, \max)$	$\sim$	$\sim$	$\prec$	$\sim$								
$\mathcal{L}(pforms, \mathbb{R}^+, \max)$	$\succ$	$\sim$	$\sim$									
$\mathcal{L}(atoms + \top, \mathbb{R}^+, \max)$	$\sim$	$\sim$										
$\mathcal{L}(pcubes, \mathbb{R}^+, \max)$	$\sim$											

Table 1: Summary of relative succinctness results. Entries to be read row first.

Theorem 45. *If Φ is closed under transformation to NNF and for every $G \in \mathcal{L}(\Phi, W, \max)$ which is in NNF, there exists a $G[\top/\neg p \mid p \in \mathcal{PS}] \in \mathcal{L}(\Psi, W', \max)$, then $\mathcal{L}(\Phi, W, \max) \preceq \mathcal{L}(\Psi, W', \max)$.*

Proof. By Lemma 44, if G is in NNF, then $G \equiv_{\max} G[\top/\neg p_1, \ldots, \top/\neg p_n]$. Since transformation to NNF and substitution of $\top$ for all negative literals are both size-preserving, $\text{size}(G) = \text{size}(G[\top/\neg p_1, \ldots, \top/\neg p_n])$, so there is always a space-efficient translation from $\mathcal{L}(\Phi, W, \max)$ to $\mathcal{L}(\Psi, W', \max)$. □

Note that W' cannot be completely arbitrary here: The condition requiring that $G[\top/\neg p \mid p \in \mathcal{PS}] \in \mathcal{L}(\Psi, W', \max)$ implies that $W' \supseteq W$, though it is not stated in the theorem.

Corollary 46. *$\mathcal{L}(forms, W, \max) \sim \mathcal{L}(pforms, W', \max)$, for all $W, W' \subseteq \mathbb{R}$.*

Proof. By Fact 40, we may assume without loss of generality that $W = W'$. Then it follows by inclusion that $\mathcal{L}(forms, W, \max) \geq \mathcal{L}(pforms, W', \max)$, and $\mathcal{L}(forms, W, \max) \preceq \mathcal{L}(pforms, W', \max)$ follows from Theorem 45. □

Our relative succinctness results are summarized in Table 1. 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 here. The most common result by an overwhelming margin is for two max languages to be equally succinct, a consequence of Theorem 41, which gives us equal succinctness between any pair of cubes and pcubes languages. (For example, $\mathcal{L}(cubes, \mathbb{R}^+, \max) \sim \mathcal{L}(pcubes, \mathbb{R}, \max)$.) All differences in succinctness shown in the table are due to the combination of Theorem 42 with Fact 40 or Theorem 13. (See the text immediately following Theorem 42 for the required argument.) Furthermore, we list no clauses languages here, because Corollary 12 reduces all clauses languages to simpler cubes languages. Because our succinctness results for max languages are much more powerful than comparable results for sum languages [10, Table 2], the table for max languages is complete.

6.2. Cross-Aggregator Succinctness

Here we examine the succinctness of selected max languages with respect to selected sum languages. For any pair of the 10 max languages and 18 sum languages considered in [10, Section 4], there is a succinctness result to be shown. We do not present all 180 potential results here, however, both for brevity and because for some sum languages without

unique representations we lack the tools to easily prove succinctness results involving them. The results we do present are representative of results of this kind.

The first two results, Theorems 48 and 49, indicate that each aggregator favors short representations for certain kinds of utility functions; whether max or sum is better for a particular application will depend on what utility functions agents are likely to have. Additionally, the final three results in this section highlight why it was necessary to index our succinctness relation to a particular class of comparison; a discussion of this appears at the end of the section.

We define two families of utility functions, $u_n^\forall$ and $u_n^\exists$, which are useful for showing succinctness differences between languages.

Definition 47. Let $u_n^\forall$ and $u_n^\exists$ be the utility functions over $\mathcal{PS}_n$ where

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

First, we use $u_n^\exists$ and a utility function with maximal range to compare $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max)$ with $\mathcal{L}(\mathit{pcubes}, \mathbb{R}, \Sigma)$. This result highlights the difference between max and sum as aggregators.

Theorem 48. $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max) \perp \mathcal{L}(\mathit{pcubes}, \mathbb{R}, \Sigma)$.

Proof. ($\not\preceq$) The family $u_n^\exists$ was shown in the proof of [10, Theorem 4.11] to have exponential representations in $\mathcal{L}(\mathit{pcubes}, \mathbb{R}, \Sigma)$. In $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max)$, $u_n^\exists$ is represented by $G = \{(p, 1) \mid p \in \mathcal{PS}\}$. Therefore $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max) \not\preceq \mathcal{L}(\mathit{pcubes}, \mathbb{R}, \Sigma)$.

($\not\succeq$) Let $u_n(X) = \sum_{a_i \in X} 2^i$. Then $|\operatorname{ran} u_n| = 2^n$, and so if $G_{\max}$ represents u_n, then by Theorem 24, $|G_{\max}| \geq 2^n$. In $\mathcal{L}(\mathit{atoms}, \mathbb{R}^+, \Sigma)$, we have $G_\Sigma = \{(a_i, 2^i) \mid 0 \leq i < n\}$ which represents u_n. Hence $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max) \not\succeq \mathcal{L}(\mathit{pcubes}, \mathbb{R}, \Sigma)$. □

We use $u_n^\forall$ to arrive at a similar result for $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max)$ and $\mathcal{L}(\mathit{pclauses}, \mathbb{R}, \Sigma)$:

Theorem 49. $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max) \perp \mathcal{L}(\mathit{pclauses}, \mathbb{R}, \Sigma)$.

Proof. ($\not\preceq$) The family $u_n^\forall$ was shown in the proof of [10, Theorem 4.10] to have exponential representations in the language $\mathcal{L}(\mathit{pclauses}, \mathbb{R}, \Sigma)$. In $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max)$, $u_n^\forall$ is represented by $\{(\bigwedge \mathcal{PS}, 1)\}$.

($\not\succeq$) Let $u_n(X) = \sum_{a_i \in X} 2^i$. Then $|\operatorname{ran} u_n| = 2^n$, and so if $G_{\max}$ represents u_n, then by Theorem 24, $|G_{\max}| \geq 2^n$. In $\mathcal{L}(\mathit{atoms}, \mathbb{R}^+, \Sigma)$, we have $G_\Sigma = \{(a_i, 2^i) \mid 0 \leq i < n\}$ which represents u_n. Hence $\mathcal{L}(\mathit{pcubes}, \mathbb{R}^+, \max) \not\preceq \mathcal{L}(\mathit{pclauses}, \mathbb{R}, \Sigma)$. □

The next two results are examples of how cross-aggregator succinctness may be surprising. The languages in question have very little expressive overlap, which is the underlying reason for these results. First, we show that the austere language $\mathcal{L}(\mathit{atoms}, \mathbb{R}, \max)$ is strictly more succinct than the seemingly-richer $\mathcal{L}(\mathit{pcubes}, \mathbb{R}, \Sigma)$:

Theorem 50. $\mathcal{L}(\mathit{pcubes}, \mathbb{R}, \Sigma) \prec \mathcal{L}(\mathit{atoms}, \mathbb{R}, \max)$.

Proof. $\mathcal{U}(\mathit{atoms}, \mathbb{R}, \max)$ corresponds to the class of unit-demand utility functions. Every unit-demand utility function u is expressible linearly in the language $\mathcal{L}(\mathit{atoms}, \mathbb{R}, \max)$ as $\{(a, u(a))\}_{a \in \mathcal{PS}}$. In $\mathcal{L}(\mathit{pcubes}, \mathbb{R}, \Sigma)$ (which is fully expressive and has uniqueness, *cf.* [10, Theorem 3.2 and Corollary 3.7]), u is represented by $\{(\bigwedge X, w_{\wedge X}) \mid X \subseteq \mathcal{PS}\}$ where

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

For any unit-demand u which is also single-minded (i.e., exactly one $a \in \mathcal{PS}$ is such that $u(a) \neq 0$), it is the case that $G = \{(a, u(a))\}$, the same as in $\mathcal{L}(\mathit{atoms}, \mathbb{R}, \max)$. For non-single-minded u, let $X \subseteq \mathcal{PS}$ be the items which u assigns nonzero value. Then G will contain a nonzero weight for $w_{\wedge Y}$ whenever $Y \subseteq X$. So, for the family $u_n^\exists$, which is the family of simple unit-demand utility functions, we have that representations in $\mathcal{L}(\mathit{pcubes}, \mathbb{R}, \Sigma)$ are exponential in $|\mathcal{PS}|$. □

Next, we present a result similar to Theorem 50, but with the aggregators reversed. It may at first seem surprising to find that there is a language with a natural definition which is strictly less succinct than the extremely limited $\mathcal{L}(\mathit{atoms}, \mathbb{R}^+, \Sigma)$, but we get this result because max languages are poor at representing modular utility functions.

Theorem 51. $\mathcal{L}(cubes, \mathbb{R}^+, \max) \prec \mathcal{L}(atoms, \mathbb{R}^+, \Sigma)$.

Proof. ($\preceq$) From [10, Corollary 3.8] we have that $\mathcal{U}(atoms, \mathbb{R}^+, \Sigma)$ is the class of normalized nonnegative modular utility functions. From Theorem 16, $\mathcal{U}(cubes, \mathbb{R}^+, \max)$ is the class of nonnegative monotone utility functions, so $\mathcal{L}(atoms, \mathbb{R}^+, \Sigma)$ is the expressive intersection of the two languages. Every representation in $\mathcal{L}(atoms, \mathbb{R}^+, \Sigma)$ is linear in $|\mathcal{PS}|$. If $u(\{a\}) \geq 0$, then the atom a will appear somewhere in any representation of u in $\mathcal{L}(cubes, \mathbb{R}^+, \max)$, so no representations which grow logarithmically in $|\mathcal{PS}|$ are possible there.

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

Finally, we compare $\mathcal{L}(pclauses, \mathbb{R}, \Sigma)$ and $\mathcal{L}(atoms, \mathbb{R}, \max)$. A feature of interest here is that we establish a succinctness gap which falls below the discriminating power of our succinctness relation, since complex unit-demand valuations are linearly representable in $\mathcal{L}(atoms, \mathbb{R}, \max)$, but the best representations of the same are quadratic in $\mathcal{L}(pclauses, \mathbb{R}, \Sigma)$.

Theorem 52. $\mathcal{L}(pclauses, \mathbb{R}, \Sigma) \sim \mathcal{L}(atoms, \mathbb{R}, \max)$.

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

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

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

We are now in position to demonstrate why the more general definition of succinctness (subscripted by the comparison class) is needed: In Theorem 50 we showed that $\mathcal{L}(pcubes, \mathbb{R}, \Sigma) \prec \mathcal{L}(atoms, \mathbb{R}, \max)$, while in Theorem 51, we showed that $\mathcal{L}(cubes, \mathbb{R}^+, \max) \prec \mathcal{L}(atoms, \mathbb{R}^+, \Sigma)$. Furthermore, $\mathcal{L}(pcubes, \mathbb{R}, \Sigma) \sim \mathcal{L}(atoms, \mathbb{R}^+, \Sigma)$, since the expressive intersection of the two languages is the whole of the latter, and in the latter every representation is small. Finally, $\mathcal{L}(atoms, \mathbb{R}, \max) \sim \mathcal{L}(cubes, \mathbb{R}^+, \max)$ because their expressive intersection is the class of nonnegative unit-demand utility functions, which have small representations in both languages. We now have the following situation:

$$\begin{array}{ccc} \mathcal{L}(pcubes, \mathbb{R}, \Sigma) & \prec & \mathcal{L}(atoms, \mathbb{R}, \max) \\ \wr & & \wr \\ \mathcal{L}(atoms, \mathbb{R}^+, \Sigma) & \succ & \mathcal{L}(cubes, \mathbb{R}^+, \max) \end{array}$$

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

7. The Complexity of Individual Utility Maximization and Minimization

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 decision problems MAX-UTIL and MIN-UTIL. The problem MAX-UTIL asks whether there is any model producing at least a specified amount of utility. The problem MIN-UTIL is the pessimal version of MAX-UTIL, which asks whether an agent will always obtain at least some specified amount of utility, no matter what model he finds himself in.

7.1. The Complexity of MAX-UTIL

The decision problem MAX-UTIL is the problem of determining whether an agent, given his preferences, can attain at least some specified amount of utility.

Definition 53 (The Decision Problem 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, is there a model $M \in 2^{\mathcal{PS}}$ where $u_{G,F}(M) \geq K$?

Note that we consider only cases where the set of weights W is a subset of $\mathbb{Q}$, in order to avoid issues of how to represent irrational weights.

Who (if anyone) needs to solve MAX-UTIL depends on the context in which our preference representation languages are being applied. Take auctions, for example: The Winner Determination Problem in combinatorial auctions [4] is the problem of allocating goods to bidders while maximizing revenue. Whether MAX-UTIL needs to be solved by the auctioneer in order to determine which bidders win which goods depends on the concrete algorithm the auctioneer uses. Even in cases where it is not necessary to solve MAX-UTIL in order to solve the Winner Determination Problem, the complexity of MAX-UTIL provides a lower bound on how complex the Winner Determination Problem can be: Observe that in the (degenerate) single-bidder case, the two problems coincide. If only one bidder shows up to the auction, then determining which items he wins is precisely the same as finding his optimal model. Therefore, the Winner Determination Problem can never be easier than MAX-UTIL, as it contains MAX-UTIL as a subproblem. Finally, for an agent himself it is useful to solve MAX-UTIL if he builds his bids not directly from an explicitly represented utility function, but instead from constraints or through elicitation. In that case, the agent may only be able to find his optimal model by first solving MAX-UTIL.

In contrast to the hardness results we have for MAX-UTIL for most sum languages [2, Section 5.5.1; 10], solving MAX-UTIL for *any* max language is trivial:

Theorem 54. MAX-UTIL$(\mathit{forms}, \mathbb{Q}, \max) \in \mathsf{TIME}(n)$*, when restricted to goalbases containing only satisfiable formulas.*

Proof. An algorithm solving MAX-UTIL for any max language simply has to iterate over the formulas in the goalbase, answer affirmatively as soon as it encounters a (φ, w) for which $w \geq K$, and answer negatively otherwise. □

This complexity result requires some discussion. First, without the restriction to satisfiable formulas, MAX-UTIL$(\mathit{forms}, \mathbb{Q}, \max)$ is NP-complete, as lifting this restriction imposes the additional requirement of checking whether φ is satisfiable whenever $w \geq K$.[7] Second (assuming that we retain the satisfiability condition), we must be careful about how we interpret the low complexity of MAX-UTIL. Note that our algorithm does not *compute* the actual model M yielding the desired level of utility; it only checks whether such an M *exists*. If we also require M itself, then we still need to extract a satisfying model M from some goal (φ, w) where $w \geq K$.

The problem of finding a satisfying assignment for an arbitrary formula that is already known to be satisfiable is probably still intractable: FSAT, which is the function problem version of SAT, is complete for FNP, the extension of NP to function problems. Given a formula φ, FSAT will return either a satisfying model M, or "no" if there is no satisfying model. If we somehow know already that φ is satisfiable, then we know that FSAT will always give us a model instead of answering "no". Call this subproblem of FSAT where the input formulas are guaranteed to be satisfiable TFSAT (for "total" FSAT). TFSAT is a member of the class TFNP, which is the subset of FNP where all problems are total—that is to say, these problems never return "no" as an answer. Clearly, $\mathsf{FP} \subseteq \mathsf{TFNP} \subseteq \mathsf{FNP}$, but no more beyond that is known. If $\mathsf{FP} = \mathsf{TFNP}$, this would imply that $\mathsf{P} = \mathsf{NP} \cap \mathsf{coNP}$, which is considered unlikely [29]. Hence, it is likely that there is no polynomial algorithm for finding a satisfying assignment for an arbitrary known-satisfiable formula, so in general, the low complexity of MAX-UTIL for max languages does not imply low complexity of the corresponding function problem which finds a witness.

In contrast to this observation, for sum languages, we are not aware of any case where the complexity of checking existence of an alternative giving at least K utility and computing that alternative differ, so long as we restrict ourselves to languages closed under substitution of logical constants.[8] For languages with an NP-complete MAX-UTIL this is a non-issue; for all sum languages with polynomial MAX-UTIL the proofs are constructive and directly show the computation of the top alternative to be polynomial.

[7] By taking satisfiability as a precondition, we make MAX-UTIL$(\mathit{forms}, \mathbb{Q}, \max)$ into a *promise problem*, as discussed by Even et al. [28].

[8] For languages which are *not* closed under substitution of logical constants, it is not always the case that the decision problem can be used to solve the function problem. For a discussion of this, see [30] and [2, Section 5.7.1].

Finally, we stress that both limitations of Theorem 54—the assumption that all goals are satisfiable, and the difference for $\mathcal{L}(forms, W, \max)$ between solving MAX-UTIL and actually computing the best alternative—vanish for certain, more restricted max languages. For both cubes and clauses (and any of their sublanguages) determining the satisfiability of single formulas is trivial; as a result, MAX-UTIL($cubes, W, \max$) and MAX-UTIL($clauses, W, \max$) are unconditionally in P. Finding a model for a single satisfiable cube or clause is also trivial, which makes computing the best alternative simple for these languages as well.

7.2. The Complexity of MIN-UTIL

So far in this section, we have considered optimal models, but what of pessimal models? Just as an agent may wish to know how well he can do, he may wish to know how poorly, as well. MIN-UTIL can be seen as the pessimistic dual of the optimistic MAX-UTIL, in the sense that it checks lower bounds instead of upper bounds.

Definition 55 (The Decision Problem MIN-UTIL)**.** The decision problem MIN-UTIL(Φ, W, F) is defined as: Given a goalbase $G \in \mathcal{L}(\Phi, W, F)$ and an integer K, are all models $M \in 2^{\mathcal{PS}}$ such that $u_{G,F}(M) \geq K$?

(Note that MIN-UTIL is *not* the complement of MAX-UTIL: This can easily be seen from the problem instance $\langle\{(\top, 1)\}, 1\rangle$, which is a member of both decision problems, for many different languages and choices of aggregators.)

We have seen in [2, Section 5.5] that for sum languages, MIN-UTIL behaves similarly to MAX-UTIL. However, this is not the case for max languages:

Theorem 56. MIN-UTIL($forms, \mathbb{Q}, \max$) *is* coNP*-complete.*

Proof. For coNP membership: Any purported counterexample model M is polynomially checkable, simply evaluating $u_{G,\max}(M)$ to see if it is less than K.

For coNP-hardness: Let φ be an instance of the well-known coNP-hard problem UNSAT, and $\langle\{(\neg\varphi, 1)\}, 1\rangle$ an instance of MIN-UTIL($forms, \mathbb{Q}, \max$). It is easy to see that if φ is not satisfiable, then $u_{\{(\neg\varphi,1)\}}(M) = 1$ for all models M, and vice versa. Hence UNSAT reduces to MIN-UTIL($forms, \mathbb{Q}, \max$). □

If we restrict the goalbases in our inputs to those containing no superfluous formulas, however, we get a more favorable result for MIN-UTIL($forms, \mathbb{Q}, \max$):

Theorem 57. MIN-UTIL($forms, \mathbb{Q}, \max$) $\in$ TIME(n)*, when restricted to goalbases containing no superfluous formulas.*

Proof. Since no $(\varphi, w) \in G$ is superfluous, any such φ will determine the value of $u_G(M)$ for some model M. Hence, the value of the worst model may be found simply by identifying the (φ, w) with the least w, which can be done by iterating a single time over G. If that $w \geq K$, the MIN-UTIL instance is positive, and negative otherwise. □

As with the sum languages over the same sets of formulas, there are some max languages for which MIN-UTIL remains polynomial in the absence of any further restrictions. This may be seen in the following three theorems.

Theorem 58. MIN-UTIL($pforms, \mathbb{Q}, \max$) $\in$ TIME(n).

Proof. For any instance $\langle G, K\rangle$, we know that u_G is monotone. Hence, $\emptyset$ is a minimally-valued model. Therefore, $\langle G, K\rangle \in$ MIN-UTIL($pforms, \mathbb{Q}, \max$) iff $u_G(\emptyset) \geq K$, so all that is required to decide the instance is reading G once and comparing two rationals. □

Theorem 59. MIN-UTIL($literals, \mathbb{Q}, \max$) $\in$ TIME(n).

Proof. Let $\langle G, K\rangle$ be an instance. We present in Figure 3 a linear-time algorithm for building the maximal optimal model M (maximal in the sense that, among all optimal models, it has the most true atoms). For each $p_i \in \mathcal{PS}$, $\hat{w}_i$ tracks the value of the best weighted literal containing p_i seen so far, while $\hat{p}_i$ tracks whether that literal was positive or negative. Because M is the maximal optimal model, it follows that $\mathcal{PS} \setminus M$ is the minimal pessimal model, since the values of items are mutually independent in u_G. Finally, we check whether $u_G(\mathcal{PS} \setminus M) \geq K$. □

Theorem 60. MIN-UTIL($cubes, \mathbb{Q}, \max$) $\in$ TIME(n^2).

$\hat{p}_1, \ldots, \hat{p}_n := 1, \hat{w}_1, \ldots, \hat{w}_n := -\infty$
for all $(\ell, w) \in G$ **do**
  if $\ell = p_i$ and $w \geq \hat{w}_i$ **then**
    $\hat{p}_i := 1, \hat{w}_i := w$
  else if $\ell = \neg p_i$ and $w > \hat{w}_i$ **then**
    $\hat{p}_i := 0, \hat{w}_i := w$
  end if
end for
$M := \{p_i \in \mathcal{PS} \mid \hat{p}_i = 1\}$

Figure 3: An algorithm for finding maximal optimal models for $G \in \mathcal{L}(\mathit{literals}, \mathbb{Q}, \max)$.

Decision Problem				Complexity
MAX-UTIL	forms	$\mathbb{Q}$	max	TIME(n)
MIN-UTIL	pforms	$\mathbb{Q}$	max	TIME(n)
MIN-UTIL	literals	$\mathbb{Q}$	max	TIME(n)
MIN-UTIL	cubes	$\mathbb{Q}$	max	TIME(n^2)
MIN-UTIL	forms	$\mathbb{Q}$	max	coNP-complete

Table 2: Summary of complexity results for MAX-UTIL and MIN-UTIL.

Proof. We argue that it is quadratic to identify and remove superfluous formulas from goalbases in $\mathcal{L}(\mathit{cubes}, \mathbb{Q}, \max)$; once G contains no superfluous formulas, we may invoke Theorem 57 to finish deciding the reduced instance.

First, observe that when $X, Y, X', Y' \subseteq \mathcal{PS}$,

$$\models \left(\bigwedge X \wedge \bigwedge \neg Y\right) \rightarrow \left(\bigwedge X' \wedge \bigwedge \neg Y'\right) \iff X' \subseteq X \text{ and } Y' \subseteq Y, \text{ or } X \cap Y \neq \emptyset.$$

That is to say, testing whether one cube implies another involves only checking whether some sets intersect or are supersets of some other sets, all of which are $O(n \log n)$ operations. This means we can find and remove superfluous cubes from any $G \in \mathcal{L}(\mathit{cubes}, \mathbb{Q}, \max)$ as follows:

For each pair of cubes $(\bigwedge X \wedge \bigwedge \neg Y, w), (\bigwedge X' \wedge \bigwedge \neg Y', w') \in G$, if $w' > w$ and either $X' \subseteq X$ and $Y' \subseteq Y$ or $X \cap Y \neq \emptyset$, then $(\bigwedge X \wedge \bigwedge \neg Y, w)$ is superfluous; remove it from G.

This algorithm is quadratic in $|G|$. Once G contains no superfluous formulas, the least remaining weight w may be found and checked for whether $w \geq K$. □

It is worth noting the dramatic difference the choice of aggregator makes for MIN-UTIL over cubes languages: From [2, Theorem 5.5.21], we have that MIN-UTIL is already coNP-complete for positively-weighted 2-cubes using summation, while here we have shown that MIN-UTIL for arbitrarily-weighted cubes of any length remains polynomial when aggregating with max.

Theorem 54 shows that MAX-UTIL is linear for all max languages (when only satisfiable formulas are given as input). The general case of MIN-UTIL is surprisingly hard, being coNP-complete. The full language $\mathcal{L}(\mathit{forms}, \mathbb{Q}, \max)$ is perhaps more suitable for optimists interested in how much utility they may hope to achieve, rather than pessimists interested in how much utility they are guaranteed. On the other hand, as there is no difference in expressivity between $\mathcal{L}(\mathit{cubes}, \mathbb{Q}, \max)$ and $\mathcal{L}(\mathit{forms}, \mathbb{Q}, \max)$ (see Corollary 12), nothing compels us to use the additional formulas we gain by permitting disjunction; and in fact it seems that we are punished with additional complexity for using disjunction in this case. See Table 2 for a summary of results for MAX-UTIL and MIN-UTIL.

8. The Complexity of Collective Utility Maximization

Collective utility maximization is the central problem we face when attempting to allocate indivisible goods among a group of agents. The goal may range from maximizing revenue, as with auctions (or minimizing cost, as with reverse

auctions), to maximizing overall satisfaction, as with task allocation. Determining how to use the limited capabilities of a satellite to satisfy each space agency which contributed to its cost [31, 32]; selling flowers or bonds to bidders at auction; assigning radio spectra to communications operators, routes to bus operators [33], and take-off and landing slots to airlines [34]—all of these are resource allocation problems where we want to maximize collective utility according to some measure.

When there are several agents, each with a utility function encoded using the same language, then the *collective utility maximization problem* (MAX-CUF), the problem of finding a solution maximizing collective utility, is of interest. By "solution" we mean a partition of the set of propositional variables among the agents, thereby fixing a model for each of them. This definition is natural, for instance, if we think of variables as goods.[9] (For example, if there are three goods a, b, c and two agents $1, 2$, then one possible allocation is to assign goods a and b to agent 1 and good c to agent 2.) In this section, we focus on the complexity of MAX-CUF for several max languages and notions of collective utility.

Now, we introduce the definitions we need for discussing collective utility maximization. A collective utility function (CUF) [13] fulfills the same role for groups of agents as the aggregation function does for individuals, *viz.*, mapping multiple utilities to a single, aggregated value:

Definition 61 (Collective Utility Functions)**.** A *collective utility function* (CUF) is a mapping $\sigma\colon \mathbb{R}^* \to \mathbb{R}$.

Since any function from tuples of reals to reals is a CUF, there are a great diversity of CUFs from which to choose [13]. In practice, however the four CUFs most frequently encountered are the egalitarian, utilitarian, elitist, and Nash product collective utility functions:

Definition 62 (Common Collective Utility Functions)**.**

- $\sigma = \max$ is the *elitist* collective utility function.
- $\sigma = \min$ is the *egalitarian* collective utility function.
- $\sigma = \Sigma$ is the *utilitarian* collective utility function.
- $\sigma = \Pi$ is the *Nash product* collective utility function.

The *utilitarian* collective utility of an alternative is the sum of the individual utilities. Optimizing with respect to utilitarian collective utility is equivalent to the Winner Determination Problem in combinatorial auctions, where it is interpreted as finding an allocation of goods to bidders that would maximize the sum of the prices offered [35]. The elitist and egalitarian CUFs are similar, but focus on different individuals as salient for collective utility: The *egalitarian* collective utility is the utility of the agent worst off, while the elitist collective utility is the utility of the agent best off. (A finer-grained version of egalitarian collective utility, the *leximin ordering* was advocated by Rawls [36]; another possibility is to focus on the utility of some agent other than the best or worst off, for example, the median agent, as the *median-rank dictator* CUF does.) Finally, the *Nash product*, the product of individual utilities, attempts to strike a balance between fairness and total utility.[10]

Now we give a formal definition of MAX-CUF, which is similar to the definition of MAX-UTIL, but lifted from an individual agent to a group of agents:

Definition 63 (The Decison Problem MAX-CUF)**.** The decision problem MAX-CUF(Φ, W, F, σ) for n agents is defined as: Given goalbases $G_1, \ldots, G_n \in \mathcal{L}(\Phi, W, F)$, a collective utility function σ, and an integer K, is there a partition $\langle M_1, \ldots, M_n\rangle$ of $\mathcal{PS}$ such that $\sigma(u_{G_1,F}(M_1), \ldots, u_{G_n,F}(M_n)) \geq K$?

First, we state two lemmas bounding the complexity of MAX-CUF:

[9] Other types of solutions, such as *finding* a single model which maximizes collective utility, are also of interest, but shall not be considered here. The combinatorial vote problem of Lang [26] is exactly this problem, in the context of voting.

[10] Note that all of the collective utility functions we consider are associative and commutative, so aggregating a tuple of individual utilities is the same as aggregating a multiset of individual utilities. Functions which are nonassociative or noncommutative tend to be less interesting as CUFs, because they fail to treat all agents equally.

Lemma 64. MAX-CUF$(\Phi, W, F, \sigma) \in$ NP *whenever F and σ are polynomially-computable functions.*

This holds because whenever F and σ are polynomially-computable functions, we can in all cases easily check whether a given allocation does in fact produce at least K utility.

Before proceeding to our next lemma, we need to define a reasonableness notion for individual and collective utility functions.

Definition 65 (Singleton Consistency)**.** A function $f\colon \mathbb{R}^* \to \mathbb{R}$ without fixed arity is *singleton consistent* if $f(\alpha) = \alpha$ for all $\alpha \in \mathbb{R}$.

Any reasonable individual aggregator or collective utility function will be singleton consistent. For individual aggregators, singleton consistency means that an agent having exactly one satisfied weighted formula (φ, w) has utility w. For collective utility functions, singleton consistency means that a single-agent society has the same utility as its sole member. Singleton consistent functions give the intuitively right answers for the utility of single agents stranded on desert islands. All of the functions we consider here—min, max, sum, and product—are singleton consistent.

Lemma 66. MAX-CUF(Φ, W, F, σ) *is at least as hard as* MAX-UTIL(Φ, W, F) *for any singleton-consistent σ.*

Here, singleton consistency ensures that σ behaves as the identity function when only a single agent is involved, and hence for such σ MAX-CUF and MAX-UTIL coincide.

MAX-CUF is easy for max languages when using the elitist collective utility function, for the same reasons as those stated in support of Theorem 54.

Fact 67. MAX-CUF$(\mathit{forms}, \mathbb{Q}, \max, \max) \in$ TIME(n) *so long as goalbases contain only satisfiable formulas.*

However, when we switch to other common CUFs, MAX-CUF becomes hard even for severely restricted languages:

Theorem 68. MAX-CUF$(2\text{-}\mathit{pcubes}, \{0, 1\}, \max, \sigma)$ *is* NP*-complete for $\sigma \in \{\Sigma, \Pi, \min\}$.*

Proof. For the first case, $\sigma = \Sigma$, we follow van Hoesel and Müller [37, Theorem 2] by reducing the known NP-complete problem TRIPARTITE MATCHING [38] to MAX-CUF$(2\text{-}\mathit{pcubes}, \{0, 1\}, \max, \Sigma)$. Instances of TRIPARTITE MATCHING are $\langle X, Y, Z, T\rangle$, where the sets X, Y, Z are such that $|X| = |Y| = |Z|$ and $T \subseteq X \times Y \times Z$. An instance $\langle X, Y, Z, T\rangle$ is a member iff there is an $M \subseteq T$ which is a perfect matching (i.e., each $x \in X$, $y \in Y$, and $z \in Z$ appears in exactly one triple in M).

For the reduction, we interpret the set X as agents and the sets Y and Z as items appearing in 2-pcubes. For each $(x, y, z) \in T$, put $(y \wedge z, 1) \in G_x$. Also put $(\top, 0) \in G_x$. Let $K = |X|$. The only way to achieve K utility by allocating $Y \cup Z$ to the agents in X is to ensure that at least one (non-$\top$) goal is satisfied from each G_x; conversely, satisfying more than one (non-$\top$) goal in any G_x does not increase collective utility, since the individual aggregator is max. Hence $\langle X, Y, Z, T\rangle \in$ TRIPARTITE MATCHING iff $\langle \{G_x\}_{x\in X}, |X|\rangle \in$ MAX-CUF$(2\text{-}\mathit{spcubes}, \{0, 1\}, \max, \Sigma)$.

For the other cases, use the same reduction from TRIPARTITE MATCHING but let $K = 1$. □

This result subsumes parts of the NP-completeness results of Bouveret et al. [39, Figure 1] and Bouveret [40, Proposition 4.22] for $\sigma = \min$. As they do not consider negation, their results (by a reduction from SET PACKING) apply to MAX-CUF$(\mathit{pforms}, \mathbb{Q}, \max, \min)$, which contains the problem MAX-CUF$(2\text{-}\mathit{spcubes}, \{0, 1\}, \max, \min)$ that we have just proved to be NP-complete. Notice also that the presence of $(\top, 0)$ in each goalbase is necessary only in the $\sigma = \Pi$ case.[11] For the other two cases including $(\top, 0)$ is safe but unnecessary; for Σ and min we could state a slightly stronger result using only 2-spcubes and $W = \{1\}$, leaving the proof otherwise unchanged.

MAX-CUF is significantly harder than MAX-UTIL. There are numerous languages for which MAX-UTIL is polynomial, but where MAX-CUF is NP-complete for some collective utility function. For example, Theorem 68 shows that a tiny fragment of a language with positive formulas and positive weights already has an NP-complete MAX-CUF problem, despite that MAX-UTIL is trivial over the same languages. See Table 3 for a summary of results for MAX-CUF.

[11] Suppose that $\top$ is left unweighted. Because $\max \emptyset = -\infty$, an agent who has no satisfied formulas in his goalbase will have a utility of $-\infty$. If there are an even number of such agents and $\sigma = \Pi$, then the collective utility will be $-\infty \times \cdots \times -\infty = \infty$, which isn't the desired result.

Decision Problem					Complexity
MAX-CUF	satisfiable φ	$\mathbb{Q}$	max	max	TIME(n)
MAX-CUF	2-spcubes	$\{1\}$	max	Σ	NP-complete
MAX-CUF	2-spcubes	$\{1\}$	max	min	NP-complete
MAX-CUF	2-pcubes	$\{0, 1\}$	max	Π	NP-complete

Table 3: Summary of complexity results for MAX-CUF.

9. Conclusions and Future Work

We have analyzed the expressivity, succinctness and complexity of languages for representing utility functions that are based on weighted propositional formulas aggregated by the max operator. Our results show that there are substantial differences in all three categories in comparison with languages using sum for aggregating weights [10].

For expressivity, we were able to give a complete picture for the most important languages, in part because the inclusion of disjunction in max languages has no effect on expressivity, which cuts down the number of languages to consider. More fine-grained results could probably still be obtained by considering languages generated by specific weight sets or by looking into formulas of limited length (although in some cases such results will be direct consequences of what is known already). Concerning relative succinctness, we have established all relationships between the max languages we consider, and have also given several results that show how they relate to selected sum languages. Additionally, we have exhibited bounds on the absolute succinctness of max-aggregated goalbases formed using cubes and positive cubes. Lastly, concerning complexity: We have observed that finding an alternative maximizing individual utility (MAX-UTIL) is computationally easy for max languages for all practical purposes, while finding an alternative minimizing individual utility (MIN-UTIL) ranges from linear to coNP-complete, depending on which formulas the language allows. We have then discussed to what extent known results apply to our framework when analyzing the complexity of the problem of maximizing collective utility for a group of agents (MAX-CUF), and proved NP-completeness results for collective utility maximization in several restrictive languages.

Some problems remain open. As mentioned before, we do not know the exact expressivity of the max languages with bounded formula length ($\mathcal{L}(k\text{-}cubes, \mathbb{R}, \max)$ and its positive sublanguages, for $1 < k < \omega$), and many cross-aggregator succinctness results are still unresolved. The bounds on goalbase size for $\mathcal{L}(cubes, \mathbb{R}, \max)$ are not tight in all cases, and could be improved. Also of interest is sharpening the boundary between the languages for which MAX-CUF is NP-complete and those for which it is only polynomial—for example, when using the Nash product as the collective utility function, how little expressive power may we remove from $\mathcal{L}(2\text{-}spcubes, \{0, 1\}, \max)$ before its MAX-CUF problem becomes easier, or how much may we add to $\mathcal{L}(atoms, \{0, 1\}, \max)$ before its MAX-CUF becomes harder?

Finally, there looms the larger question of whether the properties goalbase languages have by virtue of their aggregators may be tackled in a more general way, rather than by doing once for each aggregator the painstaking classification work done here for max and in [10] for Σ. Unfortunately, max and Σ differ enough that we do not yet see how to generalize over them; perhaps it would be easier to generalize over max and min, or Σ and Π, as these pairs of aggregators have more in common than max and Σ do.

References

[1] J. Uckelman, U. Endriss, Preference Modeling by Weighted Goals with Max Aggregation, in: G. Brewka, J. Lang (Eds.), Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR-2008), 579–587, 2008.

[2] J. Uckelman, More Than the Sum of Its Parts: Compact Preference Representation Over Combinatorial Domains, Ph.D. thesis, University of Amsterdam, ILLC Publication DS-2009-12, 2009.

[3] J. Goldsmith, U. Junker, Preference Handling for Artificial Intelligence (editorial), AI Magazine 29 (4) (2008) 9–12.

[4] P. Cramton, Y. Shoham, R. Steinberg (Eds.), Combinatorial Auctions, MIT Press, 2006.

[5] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J. A. Rodríguez-Aguilar, P. Sousa, Issues in Multiagent Resource Allocation, Informatica 30 (2006) 3–31.

[6] G. Pinkas, Propositional nonmonotonic reasoning and inconsistency in symmetric neural networks, in: J. Mylopoulos, R. Reiter (Eds.), Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-1991), Morgan Kaufmann, 525–531, 1991.

[7] C. Lafage, J. Lang, Logical representation of preferences for group decision making, in: A. G. Cohn, F. Giunchiglia, B. Selman (Eds.), Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000), Morgan Kaufmann, 457–468, 2000.

[8] S. Coste-Marquis, J. Lang, P. Liberatore, P. Marquis, Expressive power and succinctness of propositional languages for preference representation, in: [41], 203–212, 2004.
[9] Y. Chevaleyre, U. Endriss, J. Lang, Expressive Power of Weighted Propositional Formulas for Cardinal Preference Modelling, in: P. Doherty, J. Mylopoulos, C. A. Welty (Eds.), Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR-2006), AAAI Press, 145–152, 2006.
[10] J. Uckelman, Y. Chevaleyre, U. Endriss, J. Lang, Representing Utility Functions via Weighted Goals, Mathematical Logic Quarterly 55 (4) (2009) 341–361.
[11] A. Ragone, T. D. Noia, F. M. Donini, E. D. Sciascio, M. P. Wellman, Computing Utility from Weighted Description Logic Preference Formulas, in: Declarative Agent Languages and Technologies VII (DALT-2009), Lecture Notes in Computer Science, Springer, to appear, 2009.
[12] A. Ragone, T. D. Noia, F. M. Donini, E. D. Sciascio, M. P. Wellman, Weighted Description Logics Preference Formulas for Multiattribute Negotiation, in: Scalable Uncertainty Management. Third International Conference, SUM 2009, Washington, DC, USA, September 28–30, 2009. Proceedings, vol. 5785 of *Lecture Notes in Computer Science*, Springer, 193–205, 2009.
[13] H. Moulin, Axioms of Cooperative Decision Making, vol. 15 of *Econometric Society Monographs*, Cambridge University Press, 1988.
[14] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, D. Poole, CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements, Journal of Artifcial Intelligence Research (JAIR) 21 (2004) 135–191.
[15] F. Rossi, K. B. Venable, T. Walsh, mCP Nets: Representing and Reasoning with Preferences of Multiple Agents, in: D. L. McGuinness, G. Ferguson (Eds.), Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25–29, 2004, San Jose, California, USA, AAAI Press / The MIT Press, 729–734, 2004.
[16] J. Lang, L. Xia, Sequential composition of voting rules in multi-issue domains, Mathematical Social Sciences 57 (3) (2009) 304–324.
[17] J. Goldsmith, J. Lang, M. Truszczyński, N. Wilson, The computational complexity of dominance and consistency in CP-nets, Journal of Artificial Intelligence Research (JAIR) 33 (2008) 403–432.
[18] P. C. Fishburn, Utility Theory for Decision Making, John Wiley & Sons, 1970.
[19] C. Gonzales, P. Perny, GAI Networks for Utility Elicitation, in: [41], 224–234, 2004.
[20] R. I. Brafman, C. Domshlak, T. Kogan, Compact Value-Function Representations for Qualitative Preferences, in: D. M. Chickering, J. Y. Halpern (Eds.), UAI '04, Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence, July 7–11 2004, Banff, Canada, AUAI Press, 51–59, 2004.
[21] S. Ieong, Y. Shoham, Marginal Contribution Nets: A Compact Representation Scheme for Coalitional Games, in: J. Riedl, M. J. Kearns, M. K. Reiter (Eds.), Proceedings, 6th ACM Conference on Electronic Commerce (EC-2005), ACM Press, 193–202, 2005.
[22] E. Elkind, L. A. Goldberg, P. W. Goldberg, M. Wooldridge, A Tractable and Expressive Class of Marginal Contribution Nets and Its Applications, Mathematical Logic Quarterly 55 (4) (2009) 362–376.
[23] N. Nisan, Bidding Languages for Combinatorial Auctions, in: [4], 215–232, 2006.
[24] C. Boutilier, H. H. Hoos, Bidding Languages for Combinatorial Auctions, in: B. Nebel (Ed.), Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4–10, 2001, Morgan Kaufmann, 1211–1217, 2001.
[25] C. Boutilier, Solving Concisely Expressed Combinatorial Auction Problems, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI 2002), AAAI Press, 2002.
[26] J. Lang, Logical preference representation and combinatorial vote, Annals of Mathematics and Artificial Intelligence 42 (1–3) (2004) 37–71.
[27] M. Cadoli, F. M. Donini, P. Liberatore, M. Schaerf, Space Efficiency of Propositional Knowledge Representation Formalisms, Journal of Artificial Intelligence Research (JAIR) 13 (2000) 1–31.
[28] S. Even, A. L. Selman, Y. Yacobi, The Complexity of Promise Problems with Applications to Public-Key Cryptography, Information and Control 61 (1984) 159–173.
[29] C. H. Papadimitriou, On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence, Journal of Computer and System Sciences 48 (3) (1994) 498–532.
[30] J. Uckelman, A. Witzel, Logic-Based Preference Languages with Intermediate Complexity, in: J. Chomicki, V. Conitzer, U. Junker, P. Perny (Eds.), Proceedings of the 4th Multidisciplinary Workshop on Advances in Preference Handling (MPREF-2008), AAAI Press, Chicago, 123–127, 2008.
[31] M. Lemaître, G. Verfaillie, N. Bataille, Exploiting a Common Property Resource under a Fairness Constraint: a Case Study, in: T. Dean (Ed.), Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 99), Morgan Kaufmann, 206–211, 1999.
[32] S. Bouveret, J. Lang, Efficiency and Envy-freeness in Fair Division of Indivisible Goods: Logical Representation and Complexity, Journal of Artifcial Intelligence Research (JAIR) 32 (2008) 525–564.
[33] E. Cantillon, M. Pesendorfer, Auctioning Bus Routes: The London Experience, in: [4], 573–591, 2006.
[34] M. O. Ball, G. L. Donohue, K. Hoffman, Auctions for the Safe, Efficient, and Equitable Allocation of Airspace System Resources, in: [4], 507–538, 2006.
[35] D. Lehmann, R. Müller, T. Sandholm, The Winner Determination Problem, in: [4], 288–317, 2006.
[36] J. Rawls, A Theory of Justice, Harvard University Press, 1971.
[37] S. van Hoesel, R. Müller, Optimization in electronic markets: examples in combinatorial auctions, Netnomics 3 (1) (2001) 23–33.
[38] R. M. Karp, Reducibility Among Combinatorial Problems, in: R. E. Miller, J. W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, 1972.
[39] S. Bouveret, H. Fargier, J. Lang, M. Lemaître, Allocation of Indivisible Goods: A General Model and some Complexity Results, in: F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, M. Wooldridge (Eds.), 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25–29, 2005, Utrecht, The Netherlands, ACM, 1309–1310, 2005.
[40] S. Bouveret, Allocation et partage équitables de ressources indivisibles: modélisation, complexité et algorithmique, Ph.D. thesis, Supaéro/University of Toulouse, 2007.
[41] D. Dubois, C. A. Welty, M.-A. Williams (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR-2004), Whistler, Canada, June 2–5, 2004, AAAI Press, 2004.