Computing Utility from Weighted Description Logic Preference Formulas

Azzurra Ragone[1], Tommaso Di Noia[1], Francesco M. Donini[2], Eugenio Di Sciascio[1], Michael P. Wellman[3]

[1] SisInfLab, Politecnico di Bari, Bari, Italy
{a.ragone,t.dinoia,disciascio}@poliba.it
[2] Università della Tuscia , Viterbo, Italy
donini@unitus.it
[3] Artificial Intelligence Laboratory–University of Michigan, Ann Arbor, USA
wellman@umich.edu

Abstract. We propose a framework to compute the utility of a proposal w.r.t. a preference set in a negotiation process. In particular, we refer to preferences expressed as weighted formulas in a decidable fragment of First Order Logic (FOL). Although here we tailor our approach for Description Logics endowed with disjunction, all the results keep their validity in any decidable fragment of FOL. DLs offer expressivity advantages over propositional representations, and allow us to relax the often unrealistic assumption of additive independence among attributes. We provide suitable definitions of the problem and present algorithms to compute utility in our setting. We also study complexity issues of our approach and demonstrate its usefulness with a running example in a multiattribute negotiation scenario.

1 Introduction

Effective and expressive specification of preferences is a particularly challenging research problem in knowledge representation. Preference representation is essential, for example, to instruct a software agent to act on behalf of the objectives of a human being. One common approach to preference representation appeals to *multiattribute utility theory* [1], which concerns the construction of *utility functions* mapping vectors of *attributes* to real values. Given that the size of a multiattribute domain is exponential in the number of attributes, applications typically exploit independence relations among the attributes, in the most extreme case to assume that all attributes are additively independent, so that the multiattribute utility function is a weighted sum of single-attribute utility functions.

However, most real-world domains pose significant preferential dependencies, ruled out by the fully additive model. For example, referring to the desktop computer realm, we could not with an additive value function capture the fact that the value of some combination of attributes it is not simple the sum of the single attribute values. Indeed, some operating systems performs very poorly without a minimum amount of memory, therefore the value (utility) given to a specific operating system will depend on the amount of memory available.

Some recent approaches support relaxation of the fully additive assumption, for example by providing generalized versions [2] or exploiting graphical models of dependence structure [3–5], while remaining within the multiattribute framework.

Logical languages likewise provide a means to express interdependencies, but unlike multiattribute formulations they do not necessarily require that we explicitly decompose the domain into an orthogonal set of attributes. Furthermore, logical languages support integration of preference knowledge with domain knowledge modeled through an ontology. Using an ontology, indeed, it is possible to model relations among attributes in the domain (*e.g.*, *a Centrino is an Intel processor with a 32-bit CPU*), as well as the fact that some combination of features may be infeasible (therefore of minimal or undefined preference) due to constraints in the ontology itself (*e.g.*, *a Centrino processor is not compatible with a processor with a 64-bit architecture*).

In decision making problems, preferences are expressed over a set of possible alternatives, in order to rank them. In many cases, such as *e.g.*, bilateral negotiation, auctions, resource allocation, it is important to compute a utility value for, respectively, an agreement, an offer, an allocation w.r.t. the set of preferences expressed by the agent. If preferences are expressed using Propositional Logic, then the utility can be computed considering a particular propositional model (agreement, offer, allocation), taking into account formulas satisfied by that model.

While for Propositional Logic it is possible to refer directly to models (interpretations) in order to compute utility, this computation for First-order Logic (FOL) is less straightforward, as the number of possible models is infinite.

The main contribution of this paper is an approach that, given a set of preferences, represented as weighted DL formulas w.r.t. a shared ontology, computes the utility of a formula (agreement, offer, allocation, etc.) based on its possible models (interpretations). To our knowledge, the only prior method proposed in the literature for this problem is subsumption, which has some limitations, as we show in Section 4.

We point out that even though the results we show in this paper can be easily applied to whatever decidable logic with a model-theoretic semantics, we ground our approach on DLs because of their importance in the development of the Semantic Web.

The remainder of the paper proceeds as follows. First, we introduce Description Logics, then we give a brief overview of the problem of preference representation in the field of logic languages. In Section 4, we first introduce the problem of computing utility of a concept w.r.t. a preference set, showing how, sometime, subsumption leads to counterintuitive results. Then we analyze some complexity issues. In Section 5 we illustrate our framework for the computation of utility for a set of weighted DL formulas with the help of a running example. Finally, we discuss some considerations about the computational properties of the framework. Conclusion closes the paper.

2 Description Logic Basics

Description logics (DLs) are a family of formalisms well-established in the field of knowledge representation. Readers familiar with DLs may safely skim this section, attending mainly to the notation and examples. Those interested in learning more may refer to the *Description Logic Handbook* [6] for a much more comprehensive treatment.

The basic syntax elements of Description Logics are *concept names*, *properties*, and *individuals*. Concept names stand for sets of objects in the domain[4] (`Windows`, `Intel`, `LCDMonitor`), and properties link (sets of) such objects (`hasOS`, `hasCPU`, `hasMonitor`). Individuals correspond to special named elements belonging to concepts (`HP_Pavilion`, `Apple_iMac`). When we do not use proper names, we denote concepts by symbols $A, B, C, D, \ldots, \top, \bot$.

Description logics are usually endowed with a model-theoretic formal semantics. A semantic *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ represents the *domain* and $\cdot^{\mathcal{I}}$ is the *interpretation function*. This function maps every concept to a subset of $\Delta^{\mathcal{I}}$, and every property to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Then, given a concept name CN and a property name R we have: $CN^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The symbols $\top$ and $\bot$ are used to represent the most generic concept and the most specific concept respectively. Hence their formal semantics correspond to $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$.

Properties and concept names can be combined using *existential role quantification*. For example, $\texttt{PC} \sqcap \exists \texttt{netSupport}.\texttt{WiFi}$ describes the set of PCs supporting a wireless connection. Similarly, we can use *universal role quantification*, as in $\texttt{PC} \sqcap \forall \texttt{hasCPU}.\texttt{AMD}$, to describe the set of PCs having only AMD processors on board. The formal semantics of universal and existential quantification is as follows:

$$\exists R.C = \{x \in \Delta^{\mathcal{I}} | \exists y, (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$
$$\forall R.C = \{x \in \Delta^{\mathcal{I}} | \forall y, (x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$$

Concept expressions can be written using *constructors* to write concept and property *expressions*. Based on the set of allowed constructors we can distinguish different description logics. Essentially every DL allows one to form a *conjunction* of concepts, usually denoted as $\sqcap$; some DLs include also disjunction $\sqcup$ and complement $\neg$ to close concept expressions under boolean operations.

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$
$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$$

Constructs involving number restriction enable us to define concepts in terms of the numbers of roles with specified properties. For example, $\texttt{Mac} \sqcap (\geq 4 \texttt{hasUSBport})$ describes a Macintosh PC with at least four USB ports. Some DLs integrate concrete domains into the language [7]. Formally a concrete domain $\mathcal{D}$ is a pair $\langle \Delta_D, pred(D) \rangle$ where Δ_D is the domain of $\mathcal{D}$ and $pred(D)$ is a set of predicates over Δ_D. An example of concrete domain restrictions appears in the expression $\texttt{PC} \sqcap \exists \texttt{hasRam}.(\geq_2 GB)$, which describes a PC with at least 2 GB of memory. Here the domain of $\mathcal{D}$ is represented by natural numbers while $\geq_2 \in pred(D)$ is a unary predicate for $\mathcal{D}$. Notice that while properties, such as `hasUSBport`, are mapped to a subset of of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, concrete

[4] We illustrate the main points here (and throughout the paper) using the domain of desktop computers.

properties, such as GB are mapped to a subset $\Delta^{\mathcal{I}} \times \Delta_D$.

$$(\geq\ n\ R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} |\ |\{b \in \Delta^{\mathcal{I}} | (a,b) \in R^{\mathcal{I}}\}| \geq n\}$$
$$(\leq\ m\ R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} |\ |\{b \in \Delta^{\mathcal{I}} | (a,b) \in R^{\mathcal{I}}\}| \leq m\}$$
$$(\leq_k\ R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x) \leq k\}$$
$$(\geq_h\ R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x) \geq h\}$$

In general, the expressiveness of a DL depends on the type of constructors allowed.

Given a generic concept C, we use the notation $\mathcal{I} \models C$ to say that $C^{\mathcal{I}} \neq \emptyset$.

In order to formally represent domain knowledge and constraints operating among elements of the domain, we employ a set of background axioms, that is, an *ontology*. Formally, ontology $\mathcal{T}$ (for Terminology) comprises axioms of the form $D \sqsubseteq C$, where D and C are well-formed formulas in the adopted DL, and $R \sqsubseteq S$, where both R and S are properties. $C \equiv D$ is a syntactic sugor for both $C \sqsubseteq D$ and $D \sqsubseteq C$. The formal semantics of such axioms is: $(C \sqsubseteq D)^{\mathcal{I}} = C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $(R \sqsubseteq S)^{\mathcal{I}} = R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$.

We write $\mathcal{I} \models \mathcal{T}$ to denote that for each axiom $C \sqsubseteq D$ in $\mathcal{T}$ it results $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Similarly $\mathcal{I} \models C \sqsubseteq_{\mathcal{T}} D$, with $C \sqsubseteq D \notin \mathcal{T}$, denotes that both $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models C \sqsubseteq D$.

In the rest of the paper we refer to the Ontology $\mathcal{T}$ depicted in Figure 1.

$\texttt{DesktopComputer} \sqsubseteq \exists \texttt{hasCPU} \sqcap \exists \texttt{hasRam}$

$\texttt{CPUArchitecture} \equiv \texttt{64BitCPU} \sqcup \texttt{32BitCPU}$

$\texttt{32BitCPU} \sqsubseteq \neg \texttt{64BitCPU}$

$\texttt{Intel} \sqcup \texttt{AMD} \sqsubseteq \texttt{CPU}$

$\texttt{Intel} \sqsubseteq \neg \texttt{AMD}$

$\texttt{hasUSBport} \sqsubseteq \texttt{hasPeripheralPort}$

$\texttt{Athlon64} \sqsubseteq \texttt{AMD} \sqcap \exists \texttt{arch} \sqcap \forall \texttt{arch.64BitCPU}$

$\texttt{Centrino} \sqsubseteq \texttt{Intel} \sqcap \exists \texttt{arch} \sqcap \forall \texttt{arch.32BitCPU}$

$\exists \texttt{hasCPU.Centrino} \sqsubseteq \exists \texttt{netSupport.WiFi}$

$\texttt{IntelDuo} \sqsubseteq \texttt{Intel} \sqcap \exists \texttt{arch} \sqcap \forall \texttt{arch.32BitCPU}$

$\texttt{Sempron} \sqsubseteq \texttt{AMD} \sqcap \exists \texttt{arch} \sqcap \forall \texttt{arch.64BitCPU}$

Fig. 1. Reference ontology.

Description Logics are considered to be highly expressive representation languages, corresponding to decidable fragments of first-order logic. Reasoning systems based on

DLs generally provide at least two basic inference services: *satisfiability* and *subsumption*.

Satisfiability: a concept expression C is satisfiable w.r.t. an ontology $\mathcal{T}$ when $\mathcal{T} \not\models C \sqsubseteq \bot$, or equivalently $C \not\sqsubseteq_{\mathcal{T}} \bot$;

Subsumption: a concept expression C is subsumed by a concept expression D w.r.t. $\mathcal{T}$ when $\mathcal{T} \models C \sqsubseteq D$, or equivalently $C \sqsubseteq_{\mathcal{T}} D$.

In our setting satisfiability is useful, for example, to catch inconsistencies among a buyer's expressed preferences or, analogously, among sellers' offered configurations. On the other hand, subsumption can be employed to verify if a particular seller's offer satisfies one or more buyer's preferences.

3 Preference Representation Using Description Logics

The problem of preference representation deals with the expression and evaluation of preferences over a set of different alternatives (outcomes). This problem can be challenging even for a small set of alternatives, involving a moderate number of features, as the user has to evaluate all possible configurations of feature values in the domain.

In this work, we deal with this problem by a combination of expressive language, to facilitate preference specification, and preference structure exploitation, justified by multiattribute utility theory.

Several approaches to negotiation have exploited logic languages in order to express preferences, most of them using propositional logic [8–10], however only few of the approaches proposed in the literature have explored the possibility to use also an ontology to model relations among attributes [11] or the use of more expressive logics as DLs [12, 13]. Lukasiewicz and Schellhase [14] propose a framework to model conditional preferences in DLs for matchmaking. In their framework they refer to set of concepts and the formal semantics of implication is defined in terms of set membership. Such a formulation well suits their target matchmaking task. Indeed, they are not interesed in computing a utility value for a concept, e.g. an agreement, but they focus on ranking a set of results w.r.t. a query.

We point out the importance to refer to a background knowledge, *i.e.*, having an ontology $\mathcal{T}$, in order to model not only interdependencies among attributes in preference statements, but also to model inner relations among attributes that cannot be disregarded *e.g.*, is-a, disjoint or equivalence relations. In order to lay out the importance of an ontology and why we cannot abstract from it, we use a simple example involving one perspective buyer and two sellers.

Example 1. Let us suppose the buyer has among her preferences:

$$P = \exists \texttt{hasCPU}.(\texttt{AMD} \sqcap \exists \texttt{arch} \sqcap \forall \texttt{arch}.\texttt{64BitCPU}) \sqcap \exists \texttt{hasRam}.(\geq_2\ GB)$$

(PC with a 64-bit AMD processor and at least 2 GB of memory)

and there are two sellers, A and B, that can provide the respective configurations:

$$A = \exists \texttt{hasCPU}.\texttt{Athlon64} \sqcap \texttt{hasRam}(=_2 GB)$$
$$B = \exists \texttt{hasCPU}.\texttt{Centrino} \sqcap \texttt{hasRam}(=_1 GB)$$

If we refer to the ontology $\mathcal{T}$ in Figure 1 we can state that seller A can satisfy the preference expressed by the buyer —from $\mathcal{T}$ we know that Athlon64 is a 64-bit AMD processor. Conversely, seller B cannot satisfy buyer's preference, because Centrino is not a 64-bit AMD processor. Δ

We extend the well-known approach of weighted propositional formulas [8–10], representing preferences as DL formulas, where at each formula we associate a value v representing the relative importance of that formula.

Definition 1. *Let $\mathcal{T}$ be an ontology in a DL. A **Preference** is a pair $\phi = \langle P, v \rangle$ where P is a concept such that $P \not\sqsubseteq_{\mathcal{T}} \bot$ and v is a real number assigning a worth to P. We call a finite set $\mathcal{P}$ of preferences a **Preference Set** iff for each pair of preferences $\phi' = \langle P', v' \rangle$ and $\phi'' = \langle P'', v'' \rangle$ such that $P' \equiv_{\mathcal{T}} P''$ then $v' = v''$ holds.*

From now on, whenever we mention a set of preference we refer to a Preference Set.

Example 2 (Desktop Computer Negotiation). Imagine a negotiation setting where buyer and seller are negotiating on the characteristics of a *desktop computer*. The buyer will have some preferences, while the seller will have some different configurations to offer to the buyer in order to satisfy her preferences. *Let us hence suppose the buyer is looking for a desktop PC endowed with an AMD CPU. Otherwise, if the desktop PC has an Intel CPU, it should only be a Centrino one. The buyer also wants a desktop PC supporting wireless connection.* Following Definition 1 the buyer's Preference set is $\mathcal{P} = \{\langle P_1, v_1 \rangle, \langle P_2, v_2 \rangle, \langle P_3, v_3 \rangle\}$, with:

$P_1 = \forall \texttt{hasCPU}.\texttt{Centrino} \sqcap \exists \texttt{hasCPU}$
$P_2 = \exists \texttt{hasCPU}.\texttt{AMD}$
$P_3 = \exists \texttt{netSupport}.\texttt{WiFi}$

On the other side, the seller could offer *a Desktop computer supporting either a wireless connection or an AMD CPU, specifically a Sempron one, and he does not have a desktop PC endowed with a Centrino CPU*.

$A = \texttt{DesktopComputer} \sqcap \neg \exists \texttt{hasCPU}.\texttt{Centrino} \sqcap (\exists \texttt{netSupport}.\texttt{WiFi} \sqcup \forall \texttt{hasCPU}.\texttt{Sempron})$

Therefore, given a preference set $\mathcal{P}$ and a proposal A, how to evaluate the utility of this proposal w.r.t. buyer's preferences? Intuitively, the utility value should be the sum of the value v_i of the preferences satisfied by the seller's proposal. Next sections will adress this problem, showing a computation method for weighted DL-formulas.

Δ

4 Utility

Weighted formulas have been introduced for propositional logic to assign a utility value to a propositional model representing *e.g.*, the final agreement. The computation of the model and its corresponding utility is quite easy since in propositional logic we deal with a finite set of models. Following Chevaleyre et al. [10] utility is computed as:

$$\sum \{v \mid \langle P, v \rangle \in \mathcal{P} \text{ and } m \models P\}$$

Where $\mathcal{P}$ is a propositional Preference Set, m is a propositional interpretation (model) *e.g.*, representing the final agreement. [5]. We call this approach *model-based*. Less straightforward is the case of more expressive logics. Some attempts in this direction have been made by Ragone et al. [12, 13] adapting the weighted formulas framework to Description Logics. There, they do not consider models as final agreements but formulas, and the utility value is computed as:

$$\sum\{v \mid \langle P, v\rangle \in \mathcal{P} \text{ and } A \sqsubseteq_{\mathcal{T}} P\}$$

Where $\mathcal{P}$ is a DL preference set, $\mathcal{T}$ is a DL ontology and A is a concept *e.g.*, representing a proposal in a negotiation process. We call this approach *implication-based*. Although very simple to compute and immediate, this basic approach may lead to counter-intuitive examples when dealing with logic languages allowing disjunction in the final agreement.

Example 3. Consider the following preference set $\mathcal{P}$ (here for the sake of clarity we do not consider the ontology $\mathcal{T}$)

$\phi_1 = \langle A_1, v_1\rangle$
$\phi_2 = \langle A_2, v_2\rangle$
$\phi_3 = \langle A_3, v_3\rangle$

and a concept A.

$A = (A_1 \sqcup A_3) \sqcap A_2$

Δ

In Example 3, following an *implication-based* approach the final utility is

$$u^{impl}(A) = v_2$$

Indeed, only $A \sqsubseteq P_2$ holds.

On the other hand, if we use a *model-based* approach we can say that the final utility value is:

$$u^{model}(A) \in \{v_1 + v_2, v_2 + v_3, v_1 + v_2 + v_3\}$$

If we consider interpretarions $\mathcal{I}$ of A we may have that only one of the conditions below holds:

$\mathcal{I} \models A_1 \sqcap \neg A_3 \sqcap A_2$
$\mathcal{I} \models \neg A_1 \sqcap A_3 \sqcap A_2$
$\mathcal{I} \models A_1 \sqcap A_3 \sqcap A_2$

Using a model-based approach, if we want to be as conservative as possible we may consider:

$$u^{model}(A) = min\{v_1 + v_2, v_2 + v_3, v_1 + v_2 + v_3\}$$

[5] $\sum\{\cdot\}$ indicates the summation over all the elements in the set $\{\cdot\}$.

Conversely, in the most optimistic case we consider:

$$u^{model}(A) = MAX\{v_1 + v_2, v_2 + v_3, v_1 + v_2 + v_3\}$$

In the rest of the paper we will refer to the conservative situation but all the results can be easily adapted to the optimistic case. Consequently, we give a definition of Minimal Model and of its corresponding Minimal Utility Value.

Definition 2 (Minimal Models – Minimal Utility Value). *Given an ontology $\mathcal{T}$, a concept A, such that $\mathcal{T} \not\models A \sqsubseteq \bot$, and a Preference Set $\mathcal{P}$, a **Minimal Model** is an interpretation $\mathcal{I}$ such that:*

1. *both $\mathcal{I} \models A$ and $\mathcal{I} \models \mathcal{T}$*
2. *the value $u^c(A) = \sum\{v \mid \langle P, v\rangle \in \mathcal{P}$ and $\mathcal{I} \models P\}$ is minimal*

*We call $u^c(A)$ a **Minimal Utility Value** for A w.r.t. to $\mathcal{P}$.*

4.1 Complexity

In this section, we give some consequences on the complexity of computing the utility of a formula A, when utility is attached to models. In what follows, we abstract from the particular logic language $\mathcal{L}$, which gives our results the maximum attainable generality.

Lemma 1. *Given two concepts $A, P \in \mathcal{L}$, and an ontology $\mathcal{T}$, we have $A \sqsubseteq_{\mathcal{T}} P$ iff there exists a minimal model assigning value v to A when preferences are $\mathcal{P} = \{\langle P, v\rangle\}$.*

Proof. Given a singleton set of preferences $\mathcal{P} = \{\langle P, v\rangle\}$, then $u^c(A)$ can be equal to either 0, or v. Now if there exists a minimal model with value 0, then such a model is a model of $\mathcal{T}$ and A, and it must be not a model of P; hence, when the minimal model has utility 0, $\mathcal{T} \not\models A \sqsubseteq P$. On the other hand, there exists a minimal model assigning utility value v to A, then every model of $\mathcal{T}$ and A is also a model of P. But this is just the condition expressing semantically that $A \sqsubseteq_{\mathcal{T}} P$. □

A consequence of the above lemma is that computing a minimal model is at least as hard as computing subsumption in a DL $\mathcal{L}$; below we can even generalize this result to logical implication.

Theorem 1. *Given a language $\mathcal{L}$ in which deciding logical implication is $\mathcal{C}$-hard, deciding the existence of a minimal model with a given utility value is $\mathcal{C}$-hard, too.*

We observe that the result is the same (by definition) when utilities are directly assigned to *formulas*, as done *e.g.*, by Ragone et al. [12].

We now move to upper bounds on computing utilities over formulas by minimal models. We first assess the upper bound of the decision problem corresponding to computing the utility of a concept.

Theorem 2. *Let $\mathcal{L}$ be a language in which the satisfiability problem belongs to the complexity class $\mathcal{C}$, and such that* NP $\subseteq \mathcal{C}$*; moreover, let v be a positive real number, $\mathcal{P}$ be a set of preferences, $\mathcal{T}$ be a Terminology and A a concept, all expressed in $\mathcal{L}$. Then, deciding whether $u^c(A) < v$ is a problem in $\mathcal{C}$.*

Proof. Let $\mathcal{P} = \{\langle P_1, v_1\rangle, \ldots, \langle P_n, v_n\rangle\}$. Then, for each integer m between 0 and $2^n - 1$, let $(m)_2 = b_1 b_2 ... b_n$ be the binary representation of m, and let D_m be the concept defined as $A \sqcap B_1 \sqcap \cdots \sqcap B_n$, where for each $i = 1, ..., n$, if $b_i = 0$ then $B_i = \neg P_i$, else $B_i = P_i$. Intuitively, the i-th bit of $(m)_2$ decides whether P_i appears positively or negatively in D_m. Now let $\mathcal{S} = \{m \mid 0 \leq m \leq 2^{n-1}$ and $D_m \not\sqsubseteq_{\mathcal{T}} \bot\}$, *i.e.*, the set of all integers $m \in [0, 2^{n-1}]$ such that D_m be satisfiable in $\mathcal{T}$. Then, the utility of A can be expressed as

$$min \left\{ \sum_{i=1}^{n} b_i * v_i \;\middle|\; m \in \mathcal{S} \text{ and } (m)_2 = b_1 b_2 ... b_n \right\} \tag{1}$$

Intuitively, one searches the minimum of the objective function (1) over a *subset* $\mathcal{S}$ of the hypercube $\{0, 1\}^n$, where the vertices in $\mathcal{S}$ are only the ones which define a satisfiable combination of A and (possibly negated) preferences in $\mathcal{P}$. Clearly, for every satisfiable conjunction at least one model M exists, and when the utility computed by (1) is minimum, M is a minimal model.

Finally, observe that a "right" number m between 0 and 2^{n-1}—*i.e.*, a number individuating a minimal model—can be guessed nondeterministically in polynomial time. Hence, a nondeterministic Turing machine deciding $u^c(A) < v$ guesses a number m, checks the satisfiability of D_m (a problem in $\mathcal{C}$), computes $u = \sum_{i=1}^{n} b_i * v_i$, and halts with "yes" if $u < v$, otherwise "no". Therefore, when $\mathcal{C} =$ NP, the overall decision problem is in NP; when NP $\subset \mathcal{C}$, the satisfiability check in $\mathcal{C}$ dominates the overall complexity. □

For languages such that PSPACE $\subseteq \mathcal{C}$, the above theorem yields also an upper bound on the complexity of *computing* the utility of a concept. For instance, for $\mathcal{L} = \mathcal{ALC}$, and simple Terminologies, satisfiability is a problem PSPACE-complete [15]. Then computing (1) is a problem in PSPACE, too.

For languages such that $\mathcal{C} =$ NP, the above theorem yields an NPO upper bound on the complexity of *computing* the utility of a concept. We discuss this case in more detail, for $\mathcal{L}$ =Propositional Logic (PL). For this case, Theorem 1 implies that the decision problem is NP-hard; yet it does not tell us whether the computation of $u^c(A)$ admits some approximation schema, or not. We can show also NPO-hardness of computing (1), through a (not difficult) reduction from MAX-2-SAT, which is the problem of finding a model maximizing the number of clauses in a CNF of 2-literals (a.k.a. Krom) clauses. For convenience, we use the dual problem of finding a model that minimizes the number of *unsatisfied* conjunctions in a DNF of 2-literals conjunctions $D = D_1 \vee \cdots \vee D_n$. Then such a model minimizes also the utility of any (unused) literal C w.r.t. the set of preferences $\mathcal{P}_D = \{\langle D_1, 1\rangle, \ldots, \langle D_n, 1\rangle\}$. Since computing such a model is NPO-hard, our claim follows.

Observe that the above theorem does not hold for classes below NP, which need separate discussions. The case $\mathcal{C}$=PTIME covers the most simple logics, such as $\mathcal{L}$ =Conjunctions of Literals, or the DL $\mathcal{L} = \mathcal{FL}^-$ [6]. In fact, satisfiability of a conjunction amounts to check the absence of a literal and its negation in the conjunction. Yet, observe that if $\mathcal{P} \supseteq \{\langle A, v_1\rangle, \langle \neg A, v_2\rangle\}$, then for every formula C, $u^c(C) \geq min(v_1, v_2)$. In general, deciding if $u^c(C) \leq k$ is NP-complete, based on a simple reduction from 3-TAUT: given a DNF $D = D_1 \vee \cdots \vee D_n$, where each D_i is a conjunction, D is *not* a tautology iff

$u^c(A) \leq k$ (A being any literal) w.r.t. the preferences $\mathcal{P} = \{\langle D_1, 2k\rangle, \ldots, \langle D_n, 2k\rangle\}$. Less obviously, the same reduction holds for $\mathcal{L} = \mathcal{FL}^-$, using a role R_i for every propositional atom A_i, and letting the encoding γ be: $\gamma(\wedge) = \sqcap$, $\gamma(A_i) = \exists R_i$, and $\gamma(\neg A_i) = \forall R_i.B$ (for some concept name B). Then the previous DNF D is not a tautology iff $u^c(B) \leq k$ w.r.t. $\mathcal{P} = \{\langle \gamma(D_1), 2k\rangle, \ldots, \langle \gamma(D_n), 2k\rangle\}$. The fact that deciding the utility of an $\mathcal{FL}^-$concept w.r.t. a set of $\mathcal{FL}^-$preferences is NP-hard is remarkable, since satisfiability in $\mathcal{FL}^-$is trivial (every $\mathcal{FL}^-$concept is satisfiable).

5 Computation of Minimal Utility Value

In this section we show how the computation of the *minimal utility value* for a set of preferences $\mathcal{P}$ w.r.t. a concept A can be turned out in solving an optimization problem.

Given the set $\mathcal{P} = \{\phi_1, \phi_2, \phi_3\}$ of preferences and the concept A as in Example 3, we note that:

(a) A is more specific than the simple preference specification A_2;
(b) A is more specific than a disjunction of preference specification (that, in the most general case, may appear even negated).

On the other hand, due to constraints modeled within the ontology we may have some interrelations among elements of $\mathcal{P}$. For instance, it might result that:

(c) two preferences ϕ_1 and ϕ_2 cannot be satisfied at the same time;
(d) the conjunction of the former with the complement of the latter could be unsatisfiable;
(e) the combination of the complement of ϕ_1 and ϕ_2 is more specific than (*i.e.*, it implies) a third preference ϕ_3. In other words, **(c)** no model of A_1 can be also a model of A_2, **(d)** no model of A_1 can be also a model of $\neg A_2$, **(e)** all models of both $\neg A_1$ and A_2 are also models of A_3.

In term of interpretations it results that for all interpretations $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}$:

(a) $A^{\mathcal{I}} \subseteq (A_2)^{\mathcal{I}}$
(b) $A^{\mathcal{I}} \subseteq (A_1)^{\mathcal{I}} \cup (A_3)^{\mathcal{I}}$
(c) $(A_1)^{\mathcal{I}} \cap (A_2)^{\mathcal{I}} = \emptyset$
(d) $(A_1)^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \setminus (A_2)^{\mathcal{I}}) = \emptyset$
(e) $(\Delta^{\mathcal{I}} \setminus (A_1)^{\mathcal{I}}) \cap (A_2)^{\mathcal{I}} \subseteq (A_3)^{\mathcal{I}}$

Actually, if we consider also a concept A it is easy to see that whenever (c), (d) or (e) hold, then also the corresponding relations represented below are true (while the vice versa is false).

(f) $A \sqcap A_1 \sqcap A_2 \sqsubseteq_{\mathcal{T}} \bot \longrightarrow A^{\mathcal{I}} \cap (A_1)^{\mathcal{I}} \cap (A_2)^{\mathcal{I}} = \emptyset$
(g) $A \sqcap A_1 \sqcap \neg A_2 \sqsubseteq_{\mathcal{T}} \bot \longrightarrow \cap A^{\mathcal{I}} \cap (A_1)^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \setminus (A_2)^{\mathcal{I}}) = \emptyset$
(h) $A \sqcap \neg A_1 \sqcap A_2 \sqsubseteq_{\mathcal{T}} A_3 \longrightarrow A^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \setminus (A_1)^{\mathcal{I}}) \cap (A_2)^{\mathcal{I}} \subseteq (A_3)^{\mathcal{I}}$

In fact, in order to compute a *Minimal Utility Value*, if A represents *e.g.*, a final agreement, we are more interested in those models satisfying the latter equations rather than the ones satisfying (c), (d) and (e) because they are also models of A (as the *Minimal Model* is). Obviously, (a),(b),(f),(g),(h) can be generalized w.r.t. whatever *Preference Set*.

Noteworthy is that, since we use an ontology $\mathcal{T}$, the above observations apply to preference specifications represented as general concept expressions C and not only as concept names. We illustrate the above ideas with the help of an example.

Example 4 (Desktop Computer Negotiation cont'd). We again refer to the ontology $\mathcal{T}$ depicted in Figure 1.

We recall that the buyer's Preference Set is $\mathcal{P} = \{\langle P_1, v_1\rangle, \langle P_2, v_2\rangle, \langle P_3, v_3\rangle\}$, with:

$P_1 = \forall \texttt{hasCPU.Centrino} \sqcap \exists \texttt{hasCPU}$
$P_2 = \exists \texttt{hasCPU.AMD}$
$P_3 = \exists \texttt{netSupport.WiFi}$

while the seller proposal is:

$A \equiv \texttt{DesktopComputer} \sqcap \neg \exists \texttt{hasCPU.Centrino} \sqcap (\exists \texttt{netSupport.WiFi} \sqcup \forall \texttt{hasCPU.Sempron})$

Notice that, because of the axioms in ontology $\mathcal{T}$:

$\texttt{Intel} \sqsubseteq \neg \texttt{AMD}$
$\texttt{Centrino} \sqsubseteq \texttt{Intel} \sqcap \exists \texttt{arch} \sqcap \forall \texttt{arch.32BitCPU}$

preferences P_1 and P_2 cannot be satisfied at the same time, and, moreover, due to the axiom

$$\exists \texttt{hasCPU.Centrino} \sqsubseteq \exists \texttt{netSupport.WiFi}$$

in the ontology $\mathcal{T}$, preference P_1 is more specific than preference P_3. Δ

Definition 3 (Preference Clause). *Given a set of preferences $\mathcal{P} = \{\langle P_i, v_i\rangle\}, i = 1 \ldots n$, an ontology $\mathcal{T}$ and a concept A such that $A \not\sqsubseteq_{\mathcal{T}} \bot$, we say that $\mathcal{P}$ is constrained if the following condition holds:*

$$A \sqsubseteq_{\mathcal{T}} \hat{P}_1 \sqcup \ldots \sqcup \hat{P}_n \tag{2}$$

*Where $\hat{P}_i \in \{P_i, \neg P_i\}$. We call $\hat{P}_1 \sqcup \ldots \sqcup \hat{P}_n$ a **Preference Clause** if there is no strict subset $\mathcal{Q} \subset \mathcal{P}$ such that $\mathcal{Q}$ is constrained.*

Note that with a *Preference Clause* one can represent not only relations (a) and (b) but also relations (f), (g) and (h) thanks to the well known equivalence:

$$C \sqsubseteq_{\mathcal{T}} D \Longleftrightarrow C \sqcap \neg D \sqsubseteq_{\mathcal{T}} \bot$$

In fact,

(f) $A \sqcap A_1 \sqcap A_2 \sqsubseteq_{\mathcal{T}} \bot \iff A \sqsubseteq_{\mathcal{T}} \neg A_1 \sqcup \neg A_2$
(g) $A \sqcap A_1 \sqcap \neg A_2 \sqsubseteq_{\mathcal{T}} \bot \iff A \sqsubseteq_{\mathcal{T}} \neg A_1 \sqcup A_2$

(h) $A \sqcap \neg A_1 \sqcap A_2 \sqsubseteq_{\mathcal{T}} A_3 \iff A \sqsubseteq_{\mathcal{T}} A_1 \sqcup \neg A_2 \sqcup A_3$

We may say that a *Preference Clause* contains the minimal set of preferences such that Equation (2) holds.

Definition 4 (Preference Closure). *Given a Preference set $\mathcal{P} = \{\phi_i\}, i = 1 \dots n$, an ontology $\mathcal{T}$ and a concept $A \not\sqsubseteq_{\mathcal{T}} \bot$, we call **Preference Closure**, denoted as $\mathcal{CL}$, the set of Preference Clauses built, if any, for each set in $2^{\mathcal{P}}$.*

In other words, a *Preference Closure* represents the set of all possible *Preference Clauses* over $\mathcal{P}$. It represents all possible (minimal) interrelations occurring between A and preference descriptions in $\mathcal{P}$ w.r.t. an ontology $\mathcal{T}$.

Proposition 1. *Given a concept A, a **Preference Closure** $\mathcal{CL}$ and an ontology $\mathcal{T}$, if $\mathcal{I}^m$ is a Minimal Model of A then*

$$\mathcal{I}^m \models \mathcal{CL} \tag{3}$$

Proof. By Definition 2 since $\mathcal{I}^m$ is a Minimal Model then $\mathcal{I}^m \models \mathcal{T}$ and $\mathcal{I}^m \models A$. As for all models $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}$, including $\mathcal{I}^m$, also $\mathcal{I} \models \mathcal{CL}$ is satisfied, then the proposition holds. □

In order to compute *Minimal Utility Value* $u^c(A)$ we reduce to an optimization problem (OP). Usually, in an OP we have a set of constrained numerical variables and a function to be maximized/minimized. In our case we will represent constraints as a set χ of linear inequalities over binary variables, *i.e.*, variables whose value is in $\{0, 1\}$, and the function to be minimized as a weighted combination of such variables. In order to represent χ we need some pre-processing steps.

1. Compute the *Preference Closure* $\mathcal{CL}$ for $\mathcal{P}$;
2. For each *Preference Clause* $A \sqcap \hat{P}_1 \sqcap \dots \hat{P}_n \sqsubseteq_{\mathcal{T}} \bot \in \mathcal{CL}$, compute a corresponding preference constraint set $\overline{\mathcal{CL}} = \{\neg\hat{P}_1, \dots, \neg\hat{P}_n\}$. We denote with $\overline{\mathcal{CL}}^c = \{\overline{\mathcal{CL}}\}$ the set of all preference constraint sets.

Observation 1 *The reason why we do not consider $\neg A$ when computing $\overline{\mathcal{CL}}$ is that in order to compute a* Minimal Utility Value *we are looking for Minimal Models,* i.e.*, models such that $A^{\mathcal{I}}$ (and $\mathcal{I} \models \mathcal{T}$) satisfying properties of Definition 2. Each Preference Clause can be rewritten as $\mathcal{T} \models \top \sqsubseteq \neg A \sqcup \neg\hat{P}_1 \sqcup \dots \neg\hat{P}_n$. If we rewrite the right hand side of the relation in terms of interpretation functions, from the semantics of $\sqcup$ operator, we have*

$$(\neg A)^{\mathcal{I}} \cup (\neg\hat{P}_1)^{\mathcal{I}} \cup \dots (\neg\hat{P}_n)^{\mathcal{I}} \neq \emptyset \tag{4}$$

Since a Minimal Model is an interpretation such that $A^{\mathcal{I}} \neq \emptyset$, then all the models we are looking for are such that $(\neg A)^{\mathcal{I}} = \emptyset$. As a consequence, for our computation, the term $(\neg A)^{\mathcal{I}}$ in Equation (4) is meaningless. We will clarify further this point in Observation 2 while discussing the OP we build to compute the Minimal Utility Value *in the following.*

Example 5 (Desktop Computer Negotiation cont'd). Consider again the Desktop Computer negotiation of Example 2. Given the set of preference $\mathcal{P} = \{\langle P_1, v_1\rangle, \langle P_2, v_2\rangle, \langle P_3, v_3\rangle\}$ and the proposal A, if we compute the **Preference Closure** $\mathcal{CL}$ we find:

$$\mathcal{CL} = \left\{ \begin{array}{l} A \sqcap P_1 \sqsubseteq_{\mathcal{T}} \bot; \\ A \sqcap \neg P_2 \sqcap \neg P_3 \sqsubseteq_{\mathcal{T}} \bot \end{array} \right\}$$

hence, the two corresponding preference constraint sets in $\overline{\mathcal{CL}}^c$ are:

$\overline{\mathcal{CL}}_1 = \{\neg P_1\}$
$\overline{\mathcal{CL}}_2 = \{P_2, P_3\}$

Δ

Based on well-known encoding of clauses into linear inequalities (*e.g.*, [16, p.314]) we transform each set $\overline{\mathcal{CL}} \in \overline{\mathcal{CL}}^c$ in a set of linear inequalities χ and then define a function to be minimized in order to solve an OP.

Definition 5 (Minimal Utility Value OP). *Let $\mathcal{P}$ be a set of preferences and $\overline{\mathcal{CL}}^c$ be the set of all preference constraint sets. We define a* Minimal Utility Value OP*, represented as $\langle \chi, u(\mathbf{p})\rangle$, the optimization problem built as follows:*

1. ***numerical variables*** *– for each preference $\langle P_i, v_i\rangle \in \mathcal{P}$, with $i = 1, \ldots, n$ introduce a binary variable p_i and define the corresponding array $\mathbf{p} = (p_1, \ldots, p_n)$ (see Example 6);*
2. ***set χ of linear inequalities*** *– pick up each set $\overline{\mathcal{CL}} \in \overline{\mathcal{CL}}^c$ and build the linear inequalities*

$$\sum\{(1-p) \mid \neg P \in \overline{\mathcal{CL}}\} + \sum\{p \mid P \in \overline{\mathcal{CL}}\} \geq 1$$

3. ***function to be minimized*** *– given the array $\mathbf{p}$ of binary variables*

$$u(\mathbf{p}) = \sum\{v \cdot p \mid p \text{ is the variable mapping } \langle P, v\rangle\}$$

Observation 2 *If we considered also $\neg A$ when computing the sets $\overline{\mathcal{CL}} \in \overline{\mathcal{CL}}^c$ we would have had inequalities in the form:*

$$(1-a) + \sum\{(1-p) \mid \neg P \in \overline{\mathcal{CL}}\} + \sum\{p \mid P \in \overline{\mathcal{CL}}\} \geq 1$$

Since we are interested in models where $A^{\mathcal{I}}$ is interpreted as nonempty, then variable a has to be equal to 1. Hence the first element of the above summation is always equal to 0. In other words, we can omit $\neg A$ when computing a preference constraint set $\overline{\mathcal{CL}}$.

The solution to a *Minimal Utility Value OP* will be an assignment $\mathbf{p}_s$ for $\mathbf{p}$, *i.e.*, an array of $\{0, 1\}$-values, minimizing $u(\mathbf{p})$.

Example 6 (Desktop Computer Negotiation cont'd). Back to the Desktop Computer negotiation of Example 2, after the computation of Preference Closures and set $\overline{\mathcal{CL}}^c$, we build the corresponding optimization problem in order to find the model with the minimal utility value:

$$\begin{aligned} \mathbf{p} &= (p_1, p_2, p_3) \\ \chi &= \begin{cases} 1 - p_1 \ \geq 1 \\ p_2 + p_3 \geq 1 \end{cases} \\ u(\mathbf{p}) &= v_1 \cdot p_1 + v_2 \cdot p_2 + v_3 \cdot p_3 \end{aligned}$$

Possible solutions are:

$$\begin{aligned} \mathbf{p}'_s &= (0, 1, 0) \ , \ u(\mathbf{p}'_s) = v_2 \\ \mathbf{p}''_s &= (0, 0, 1) \ , \ u(\mathbf{p}''_s) = v_3 \\ \mathbf{p}'''_s &= (0, 1, 1) \ , \ u(\mathbf{p}'''_s) = v_2 + v_3 \end{aligned}$$

The minimal solution will be either $\mathbf{p}'_s$ or $\mathbf{p}''_s$, depending of the value of v_2 and v_3. $\bigtriangleup$

Given a a solution $\mathbf{p}_s$ to a *Minimal Utility Value OP* $\langle \chi, u(\mathbf{p}) \rangle$, we call *Minimal Preference Set* $\overline{\mathcal{P}}^m$ and *Minimal Assignment* A^m, respectively, the set and the formula built as in the following[6]:

$$\begin{aligned} \overline{\mathcal{P}}^m &= \{\langle P_i, v_i \rangle \mid p_i = 1 \textit{ in the solution } \mathbf{p}_s\} \\ A^m &= \bigsqcap \{P_i \mid p_i = 1 \textit{ in the solution } \mathbf{p}_s\} \sqcap \bigsqcap \{\neg P_i \mid p_i = 0 \textit{ in the solution } \mathbf{p}_s\} \end{aligned}$$

Theorem 3. *Given a solution* $\mathbf{p}_s$ *to a* Minimal Utility Value OP $\langle \chi, u(\mathcal{P}) \rangle$ *and a* Minimal Assignment A^m*:*

1. *if* $\mathcal{I}^m$ *is a* Minimal Model *then* $\mathcal{I}^m \models A^m$*;*
2. $u(\mathbf{p}_s)$ *is a* Minimal Utility Value.

Proof. First we show that there exists at least one model $\mathcal{I}^m \models \mathcal{T}$ *such that both* $\mathcal{I}^m \models A$ *and* $\mathcal{I}^m \models A^m$*. If* $\mathcal{I}^m$ *did not exist, then* $A^{\mathcal{I}^m} \cap (A^m)^{\mathcal{I}^m} = \emptyset$*. We can easily rewrite the latter relation as* $A \sqcap A^m \sqsubseteq_{\mathcal{T}} \bot$ *which is equivalent to* $A \sqsubseteq_{\mathcal{T}} \neg A^m$*. But this is not possible. Indeed, if* A *and* A^m *were inconsistent with each other w.r.t.* $\mathcal{T}$ *then, by Proposition 1 we should have the corresponding Preference Clause in* $\mathcal{CL}$ *and the related inequality in* χ*:*

$$\sum \{(1 - p) \mid P \textit{ appears in } A^m\} + \sum \{p \mid \neg P \textit{ appears in } A^m\} \geq 1$$

In order to be satisfied, the latter inequality must have either (a) at least one variable assigned to 0 *in the first summation or (b) at least one variable assigned to* 1 *in the second one. Case (a) means that the corresponding preference is not satisfied by* A^m *while case (b) means that the corresponding preference is satisfied by* A^m*. Both cases*

[6] with $\bigsqcap\{\cdot\}$ we denote the conjunction of all the concepts in the set $\{\cdot\}$

are conflicting with the definition of A^m.
By construction of χ, we have that if $\mathcal{I}^m \models A^m$ then $\mathcal{I}^m \models A$ (see Observation 2). Since A^m comes from the minimization of $u(\mathbf{p}_s)$ then $\mathcal{I}^m \models A^m$ represents a model of A^m (and then of A) such that

$$\sum\{v \mid \langle P, v\rangle \in \mathcal{P} \text{ and } \mathcal{I}^m \models P\}$$

is minimal.
It is straightforward to show that $u(\mathbf{p}_s)$ is a Minimal Utility Value. □

5.1 Computational properties of the method

We now relate the computation method proposed in this section with the computational complexity results of the previous section. First of all, we observe that the size of the Preference Closure $|\mathcal{CL}|$ can be—in the worst case—exponential in n, the size of the preference set. Since Linear Integer Programming is an NPO-complete problem[16], overall our problem can be solved nondeterministically in exponential time and space.

However, we observe that $|\mathcal{CL}|$ *does not depend* on the size of the ontology $\mathcal{T}$, which typically is much larger than the size of $\mathcal{P}$. In some sense, $\mathcal{CL}$ *compiles out* all the complexity due to the satisfiability problem in the chosen DL, leaving the OP of the combinatorics related to compatibility of preferences among each other, and with the formula C whose minimal utility value has to be computed. This is perfectly reasonable when Satisfiability in the chosen DL is a PSPACE-complete problem, or harder, since the best known procedures for solving PSPACE-complete problems use exponential time anyway, and the space used is exponential only in the number of preferences, not in the size of $\mathcal{T}$.

For the cases in which the language for preferences has a low-complexity satisfiability problem, say, NP, or PTIME , though, preprocessing into $\mathcal{CL}$ the complete structure of preference compatibilities may be an overshoot. In such cases, it would seem more reasonable to devise specialized procedures that compute on demand the satisfiability of a conjunction of preferences.

An orthogonal analysis can be done on the *scalability* of the method when the utilities of several offers $C_1, \ldots, C_m$ must be compared. Here it seems that one has to solve m separate OPs of size exponential in $|\mathcal{P}|$. While this is the worst case, some optimization based on the logic for offers is possible. In fact, observe that $C_i \sqsubseteq_{\mathcal{T}} C_j$ implies $u^c(C_j) \leq u^c(C_i)$ (a model of C_i is also a model of C_j). Hence, when searching for the offer with the maximum least utility, C_j can be safely disregarded. Intuitively, more specific offers are preferred over more generic ones, with the intuition that a generic offer C_j has a worst-case utility $u^c(C_j)$ which is less than the worst-case utility $u^c(C_i)$ of a more specific offer C_i.

6 Conclusion

Logic languages have been proposed here as a natural and powerful preference representation tool for automated negotiation purposes. We have shown how it is possible

to compute a utility value for a concept (agreement, proposal, allocation), when preferences are expressed as weighted DL formulas w.r.t. a shared ontology $\mathcal{T}$. Although we ground our framework in the DLs realm, we point out that the framework itself is completely general and suitable for whatever decidable fragment of FOL. We also reported complexity results and showed the applicability and benefits of our approach with the help of a meaningful example. Currently, we are studying how to combine this approach with graphical models, and in particular GAI (Generalized Additive Independence) [17, 3], in order to model multiattribute auctions.

References

1. Keeney, R.L., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. John Wiley & Sons, New York (1976)
2. Gonzales, C., Perny, P.: GAI networks for utility elicitation. In: Ninth Intl. Conf. on Principles of Knowledge Representation and Reasoning, Whistler, BC, Canada (2004) 224–234
3. Bacchus, F., Grove, A.: Graphical models for preference and utility. In: Eleventh Conf. on Uncertainty in Artificial Intelligence, Montreal (1995) 3–10
4. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: A tool for representing and reasoning about conditional ceteris paribus preference statements. Journal of Artificial Intelligence Research **21** (2004) 135–191
5. Engel, Y., Wellman, M.P.: CUI networks: A graphical representation for conditional utility independence. Journal of Artificial Intelligence Research **31** (2008) 83–112
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge Univ. Press (2002)
7. Baader, F., Hanschke, P.: A Scheme for Integrating Concrete Domains into Concept Languages. Technical Report Tech Rep. RR-91-10 (1991)
8. Pinkas, G.: Propositional non-monotonic reasoning and inconsistency in symmetric neural networks. In: Twelfth Intl. Joint Conf. on Artificial Intelligence. (1991) 525–531
9. Lafage, C., Lang, J.: Logical representation of preferences for group decision making. In: Seventh Intl. Conf. on Principles of Knowledge Representation and Reasoning. (2000) 457–468
10. Chevaleyre, Y., Endriss, U., Lang, J.: Expressive power of weighted propositional formulas for cardinal preference modelling. In: Tenth International Conference on Principles of Knowledge Representation and Reasoning. (2006) 145–152
11. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.: A logic-based framework to compute Pareto agreements in one-shot bilateral negotiation. In: Seventeenth European Conference on Artificial Intelligence. (2006) 230–234
12. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.M.: Description logics for multi-issue bilateral negotiation with incomplete information. In: Twenty-Second AAAI Conference on Artificial Intelligence. (2007) 477–482
13. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.M.: Alternating-offers protocol for multi-issue bilateral negotiation in semantic-enabled marketplaces. In: 6th International Semantic Web Conference (ISWC'07). Volume 4825 of Lecture Notes in Computer Science., Springer-Verlag (2007) 395–408
14. Lukasiewicz, T., Schellhase, J.: Variable-strength conditional preferences for matchmaking in description logics. In: KR. (2006) 164–174
15. Baader, F., Hollunder, B.: $\mathcal{KRIS}$: $\mathcal{K}$nowledge $\mathcal{R}$epresentation and $\mathcal{I}$nference $\mathcal{S}$ystem. SIGART Bulletin **2**(3) (1991) 8–14

16. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall (1982)
17. Fishburn, P.C.: Interdependence and additivity in multivariate, unidimensional expected utility theory. International Economic Review **8** (1967) 335–342