Weighted Description Logics Preference Formulas for Multiattribute Negotiation

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 an agreement 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 and agreements expressed as a formula. We ground our framework in Description Logics (DL) endowed with disjunction, to be compliant with Semantic Web technologies. A logic based approach to preference representation allows, when a background knowledge base is exploited, 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 validate our approach through an experimental evaluation.

1 Introduction

The problem of user preference specification have always been a particularly challenging research problem in knowledge representation. Expressing preferences in an effective and expressive way is especially important in negotiation contests. Indeed, in negotiation processes preference representation is of capital importance in order to allow users to express preferences on issues characterizing the bargaining object. One common approach to represent preferences on multiattribute negotiation rely on *multi-attribute utility theory* [1], where utility functions assign a real value to different combinations of attributes. However, the number of possible combinations exponentially grow in the size of the attribute domain, therefore even with a small number of attributes, the number of possible combinations could be really high. For this reason, usually, independence relations among the attributes are exploited, and often attributes are assumed to be additively independent, so that the multiattribute utility function is a weighted sum of single-attribute utility functions. However, in real-world domains such an assumption is often not entirely true, as there are preferential dependencies among attributes that cannot be simple ruled out. For example, referring to a negotiation on the characteristics of a laptop, a user should be able to express a preference not only on a single attribute, *e.g.*, a particular operating systems, but on a combination of attributes: a particular operating system with a minimum amount of memory. This mean that a particular operating system does not have a value per se, so we cannot simple use an additive value function. 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.

On the other hand, logical languages seem to be the ideal candidates to express interdependencies among preferences. Moreover they can also model in an ontology the knowledge about the domain, in order to express interdependencies among attributes, (*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 due to constraints in the ontology itself (*e.g.*, *a Centrino processor is not compatible with a processor with a 64-bit architecture*). In a negotiation it is important to compute the utility value of different outcomes (agreements) w.r.t. the set of preferences expressed by the agent, in order to rank them.

The main contribution of this paper is an approach that, given a set of preferences, represented as weighted Description Logics formulas w.r.t. a shared ontology, computes the utility of a formula (agreement) 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 shown in [6].

The problem is particularly challenging as if preferences are expressed using Propositional Logic, then the utility can be computed considering a particular propositional model (representing the agreement), 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 results presented in the paper remain valid for whatever decidable logic with a model-theoretic semantics, but 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. Firstly we brief introduce the notation used in the paper and the problem of preference representation in the field of logic languages. Then we present the framework to compute the utility of a DL preference set w.r.t. a formula, illustrating a novel algorithm to compute a preference closure in our set. Finally we perform some experiments in order to evaluate the approach. Conclusion closes the paper.

2 Notation

Hereafter, we assume the reader be familiar with DLs syntax and semantics. We use symbols $A, B, C, D, \ldots, \top, \bot$ to denote concepts. Given a generic concept C, we use the notation $\mathcal{I} \models C$ to say that $C^{\mathcal{I}} \neq \emptyset$. Interpretation functions can also apply to concept expressions and axioms: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$, $(C \sqsubseteq D)^{\mathcal{I}} = C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Here we use $\mathcal{T}$ (for Terminology) to indicate a DL ontology, *i.e.*, a set of axioms of the form $C \sqsubseteq D$ (inclusion) and $C \equiv D$ (definition) with C and D being concepts. We say C is subsumed by D w.r.t. $\mathcal{T}$ when $\mathcal{T} \models C \sqsubseteq D$, or equivalently $C \sqsubseteq_{\mathcal{T}} D$; C is not satisfiable w.r.t. the ontology $\mathcal{T}$ when it is subsumed by the most specific concept $\mathcal{T} \models C \sqsubseteq \bot$, or equivalently $C \sqsubseteq_{\mathcal{T}} \bot$; C is not subsumed by D w.r.t. $\mathcal{T}$ when

$\mathcal{T} \not\models C \sqsubseteq D$, or equivalently $C \not\sqsubseteq_{\mathcal{T}} D$. We write $\mathcal{I} \models \mathcal{T}$ to denote that for each axiom $C \sqsubseteq D$ in $\mathcal{T}$ it results $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and for each axiom $C \equiv D$ in $\mathcal{T}$ it results $C^{\mathcal{I}} = D^{\mathcal{I}}$. 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 $\mathcal{I} \models C \sqsubseteq D$.

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 [7–9], however only few of the approaches proposed in the literature have explored the possibility to use also an Ontology to model relations among attributes [10] or the use of more expressive logics as DLs [11].

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. We extend the well-known approach of weighted propositional formulas [7–9], 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 positive 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.*

Example 1 (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:

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

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

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. $\triangle$

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 Prefernce Set $\mathcal{P}$, a* ***Minimal Model*** *w.r.t. A, $\mathcal{P}$ and $\mathcal{T}$ 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 Computation of Minimal Utility Value

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

Definition 3 (Preference Clause). *Given a preference set $\mathcal{P} = \{\langle P_i, v_i\rangle\}, i = 1 \ldots n$, 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:*

$$A \sqsubseteq_{\mathcal{T}} \hat{P}_1 \sqcup \ldots \sqcup \hat{P}_n \qquad (1)$$

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.*

We may say that a *Preference Clause* contains the minimal set of preferences such that Equation (1) holds. Note that Equation (1) can be rewritten as $A \sqcap \neg\hat{P}_1 \sqcap \ldots \sqcap \neg\hat{P}_n \sqsubseteq_{\mathcal{T}} \bot$ (see [**?**] for more details).

Definition 4 (Preference Closure). *Given a Preference set $\mathcal{P} = \{\phi_i\}, i = 1 \ldots n$, 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*

$$\mathcal{I}^m \models \mathcal{CL} \qquad (2)$$

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. $\square$

4.1 Preference Clauses and Theory Prime Implicates

It is easy to see that the notion of Preference Closure shares many characteristics with the one of prime implicates [12] of a formula. Nevertheless the former extends, to some extent, the definition of prime implicates of a formula. For the sake of completeness we rewrite here the definition of theory prime implicates [13]. This formulation was originally proposed for a propositional setting but it remains consistent also if we move to First Order Logic or to whathever logical language for which we have the notion of: clause (the language allows the use of Boolean disjunction); logical entailment; logical equivalence w.r.t. entailment.

Definition 5 (Theory Prime Implicates [13]). *Let $\mathcal{T}$ be a propositional knowledge base and both A and CL be two propositional formulas. A* ***theory prime implicat****e of A w.r.t. $\mathcal{T}$ is a clause CL such that:*

- *$A \models_{\mathcal{T}} CL$ holds;*
- *for every clause CL', if both $A \models_{\mathcal{T}} CL'$ and $CL' \models_{\mathcal{T}} CL$ hold, then $\models_{\mathcal{T}} CL' \equiv CL$ holds.*

Noteworthy is that in the classical definition of theory of prime implicates there is no restriction of the symbols to be used in CL. In order to (re)define Definition 3 in terms similar to Theory Prime Implicates we introduce the notion of concept signature of a formula in DL. Given a DL formula C we call concept signature of C, denoted as $sign(C)$, the set of concept names occurring in C. It is straight to extend concept signature to axioms and set of axioms.

Definition 6 (Preference Clause - Preference Name). *Let $\mathcal{L}$ be a decidable Description Logic and let $\mathcal{T}$ and A be a consistent knowledge base in $\mathcal{L}$ and a concept in $\mathcal{L}$ such that $A \not\sqsubseteq_{\mathcal{T}} \bot$. Given a set of preferences $\mathcal{P} = \{\langle P_i, v_i \rangle\}$ we define a TBox $\mathcal{T}_{\mathcal{PN}} = \{PN_i \equiv P_i\}$ where PN_i is a concept name such that $PN_i \notin \bigcup_{\langle P_i, vi \rangle \in \mathcal{P}} sign(P_i) \cup sign(\mathcal{T}) \cup sign(A)$. We say that PN_i is a* ***preference name*** *for P_i and we use $\mathcal{PN}$ to denote the set of all preference names. A* ***preference clause*** *of A w.r.t. $\mathcal{T}$ is a clause CL such that:*

- *$sign(CL) \subseteq \mathcal{PN}$;*
- *$A \sqsubseteq_{\mathcal{T} \cup \mathcal{T}_{\mathcal{P}}} CL$ holds;*
- *for every clause CL', such that $sign(CL') \subseteq \mathcal{PN}$, if both $A \models_{\mathcal{T} \cup \mathcal{T}_{\mathcal{P}}} CL'$ and $CL' \models_{\mathcal{T} \cup \mathcal{T}_{\mathcal{P}}} CL$ hold, then $\models_{\mathcal{T}} CL' \equiv CL$ holds.*

In other words, a preference clause can be seen as a theory prime implicate where CL has to be rewritten in terms of Preference Names. In fact, prime implicates for Propositional Logic are more similar to preference clauses than the ones proposed for for Description Logics [14].

4.2 From Preference Closure to Minimal Utility Value

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* $A \sqcap \hat{P}_1 \sqcap \ldots \hat{P}_n \sqsubseteq_{\mathcal{T}} \bot \in \mathcal{CL}$, compute a corresponding preference constraint set $\overline{\mathcal{CL}} = \{\neg\hat{P}_1, \ldots, \neg\hat{P}_n\}$. We denote with $\overline{\mathcal{CL}}^c = \{\overline{\mathcal{CL}}\}$ the set of all preference constraint sets.

Example 2 (Desktop Computer Negotiation cont'd). Consider again the Desktop Computer negotiation of Example 1. 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: $\overline{\mathcal{CL}}_1 = \{\neg P_1\}$, $\overline{\mathcal{CL}}_2 = \{P_2, P_3\}$ $\triangle$

Based on well-known encoding of clauses into linear inequalities (*e.g.*, [15, 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 7 (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 $p_i \in \{0, 1\}$ and define the corresponding array $\mathbf{p} = (p_1, \ldots, p_n)$ (see Example 3);*
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*

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

Observation 1 *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 3 (Desktop Computer Negotiation cont'd). Back to the Desktop Computer negotiation of Example 1, 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. $\triangle$

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[4]:

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

Theorem 1. *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

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

while case (b) means that the corresponding preference is satisfied by A^m. Both cases 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. □

4.3 An Algorithm to compute a Preference Closure

Although the above shown similarities, it is quite hard to adapt the algorithm already proposed in the literature for the theory of prime implicates to compute a Preference Clousure. Indeed, algorithms for Propositional Logic rely on the propositional structure of formulas and axioms in the theory $\mathcal{T}$. In [16], Ngair needs the knowledge base to be a conjunction of DNF formulas while Dechter and Rish [17] uses a knowledge base in CNF. Marquis and Sadaoui [13] rewrite propositional formulas using a Shannon expansion which, to our knowledge, is not applicable to non-propositional languages. Moreover known algorithms for Description Logics [14] do not take into accounts the knowledge base (theory). In this section we propose an algorithm to compute a Preference Closure for a Description Logic $\mathcal{L}$. The main idea behind the algorithm we are going to describe bases on the following two simple observations:

1. since Boolean disjunction enjoys the commutativity property, if $A \sqsubseteq_{\mathcal{T}} P_i \sqcup P_j$ then $A \sqsubseteq_{\mathcal{T}} P_j \sqcup P_i$. Hence, once we know that $A \sqsubseteq_{\mathcal{T}} P_i \sqcup P_j$ holds we do not check $A \sqsubseteq_{\mathcal{T}} P_j \sqcup P_i$ also.
2. given a clause CL with $sign(CL) \subseteq \mathcal{PN}$, if $A \sqsubseteq_{\mathcal{T}} CL$ holds then also $A \sqsubseteq_{\mathcal{T}} CL \sqcup PN'$, with $PN' \in \mathcal{PN}$, holds (the same may be for $\neg PN'$).

To compute all the clauses belonging to the Preference Closure, we build a tree where each node is labeled with a preference name or a negated preference name and edges represent a Boolean disjunction between the two preferences (positive or negated) represented by the two nodes connected by the edge. We denote with $L(n)$ the label of a node n in the tree and with N the set containing only the leaf nodes of the tree. In order to build the tree, we impose a total order $<$ between preference names. We call PN_{last} the preference name such that for each preference name PN_i the relation $PN_i < PN_{last}$ always holds.

The following functions will be useful to build the Preference Closure starting from the tree.

$$abs : N \rightarrow \mathcal{PN}$$
$$abs(n) = \begin{cases} PN, if \;\; L(n) = PN \\ PN, if \; L(n) = \neg PN \end{cases}$$

$$branch\_clause : N \rightarrow \mathcal{L}$$
$$branch\_clause(n) = \bigsqcup\{L(m) \mid m \text{ is a node in the branch containing } n\}$$

Given a leaf node n, the tree is expanded according to Algorithm $expand(n)$.

```
**Algorithm:**$expand(n)$
**foreach** *preference name* $PN$ *such that* $abs(L(n)) < PN$ **do**
    **create** two nodes $m'$ and $m''$ such that $L(m') = PN$ and $L(m'') = \neg PN$;
    **add** the edges $(n, m')$ and $(n, m'')$ to the tree ;
**end**
```

We say a branch is **open** if it is possible to further expand the branch starting from its leaf node. A branch is closed if it is not open. We impose the root node of the tree, denoted as ϵ, has no label. The tree for Preference Closure computation is computed according to Algorithm 1. Note that the subsumption check in Line 9 of Algorithm 1, both does not involve the TBox and it compares only clauses. Then it can be easily reduced to set comparison.

In Figure 1 is depicted the tree built to compute the Preference Closure in Example 2. The number below the leaf nodes represents the reason why the corresponding branch closed w.r.t. lines in Algorithm 1. As an example, number 16 (respectively 9 and 12) says taht the brach closed because of line 16 (respectively 9 and 12) in Algorithm 1.

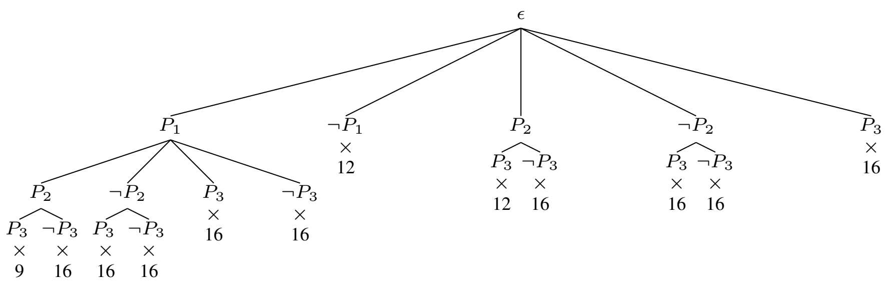

Fig. 1. The tree built using $preference\_closure$ to compute the Preference Closure of Ex. 2.

The complexity of Algorithm 1 is dominated by the number of open nodes in the tree. Although one may optimize the search by expanding first nodes with shortest and longest paths, in the worst case a constant fraction of the powerset of the preferences must be taken into account. Hence the subsumption test of Line 12 is repeated $2^{|\mathcal{P}|}$ times. Let $\|x\|$ be a measure of the size of x. Denoting by $s(\|\mathcal{T}\|, \|C\|, \|D\|)$ a function bounding the time complexity of a subsumption test $C \sqsubseteq_{\mathcal{T}} D$ in the chosen DL, the overall complexity is bounded by $2^{|\mathcal{P}|} \cdot s(\|\mathcal{T} \cup \mathcal{T}_{\mathcal{PN}}\|, \|A\|, |\mathcal{P}|)$. Observe that the third argument of s is just $|\mathcal{P}|$, since the size of preferences is hidden in $\mathcal{T}_{\mathcal{PN}}$. It is reasonable to assume that the number of preferences is bounded; in this case, the overall complexity of Algorithm 1 is dominated by the subsumption test in the chosen DL.

```
Algorithm: $preference\_closure(\epsilon, A, \mathcal{T}, \mathcal{P}, \mathcal{PN}, \mathcal{T}_{\mathcal{PN}})$
$\mathcal{CL} = \emptyset$;
foreach preference name $PN_i \in \mathcal{PN}$ do
    create two nodes $m'_i$ and $m''_i$ such that $L(m'_i) = PN$ and $L(m''_i) = \neg PN$;
    add the edges $(\epsilon, m'_i)$ and $(\epsilon, m''_i)$ to the tree;
end
while there is an open branch in the tree do
    foreach open branch $\beta_j$ with $n_j$ being its leaf node do
        if $\mathcal{CL} \models A \sqsubseteq branch\_clause(n_j)$ then
            close $\beta_j$;
        end
        else if $A \sqsubseteq_{\mathcal{T} \cup \mathcal{T}_{\mathcal{PN}}} branch\_clause(n_j)$ then
            $\mathcal{CL} = \mathcal{CL} \cup \{A \sqsubseteq branch\_clause(n_j)\}$ ;
            close $\beta_j$;
        end
        if $label(n_j) = PN_{last}$ then
            close $\beta_j$;
        end
        else
            $expand(n_j)$;
        end
    end
end
```
Algorithm 1: An algorithm to compute a Preference Closure

4.4 Performance evaluation

In order to evaluate what is the impact, from a performance perspective, of the computation of a Preference Clause in a real world application, we built some testbeds and evaluated execution time w.r.t. the parameters highlighted in the previous section. We selected some OWL DL ontologies with different expressivity (considering the corresponding DL language) from the publicly available repository of TONES project[5]. Based on these ontologies we randomly generated concept expressions which resulted consistent w.r.t. the source ontology. From these sets of concepts, one for each ontology, we selected 150 different subsets of 11, 10, 9, 8, 7, 6, 5, 4, 3, 2 concepts. Hence, we generated 150 tests with 1 concept representing A and 10 concepts representing preferences $P_1, \ldots, P_{10}$ and similarly for the other subsets. In Figure 2 in Appendix we report results for three of the ontologies we tested, i.e Pizza, Movie and BuildingsAndPlaces (see Table 1). For Pizza ontology we removed inconsistent classes. For the sake of presentation we report only average execution time w.r.t. number of axioms in the original ontology (*i.e.*, without $\mathcal{T}_{\mathcal{PN}}$) as the number of preferences changes. Complete testbeds and test results are available at `http://sisinflab.poliba.it/dinoia/preference_tests.zip`. The reference architecture is a Intel Core Duo Quad CPU @ 2.66 Ghz processor, 3.2 Gigabytes of RAM, Ubuntu Linux 9.04, Kernel 2.6.28, Sun Java Runtime Environment 1.6.0.

Ontology Name	**DL Expressivity**	**Number of Axioms**
Pizza	$\mathcal{SHOIN}$	712
Movie	$\mathcal{ALCN}$	140
BuildingsAndPlaces	$\mathcal{ALCHOQ}$	1204

Table 1. Reference Ontologies

5 Conclusion

Logic languages have been proposed here as a natural and powerful way to express preferences in negotiation contests. We presented a framework to compute the utility value of a formula (agreement), when preference are expressed as weighted DL formulas w.r.t. a shared ontology. We remark that, even if we use DLs, the framework is completely general and suitable for whatever decidable fragment of FOL. We propose a novel algorithm to compute a Preference closure and discuss also its complexity. An experimental evaluation showed the applicability and suitability of our framework. Currently, we are studying how to combine this approach with graphical models, and in particular GAI (Generalized Additive Independence) [18, 3], in order to model multiattribute auctions.

[5] `http://owl.cs.manchester.ac.uk/repository/browser`

Acknowledgment

We are very grateful to Michelantonio Trizio for developing the system used to test and evaluate the approach. We wish to thank anonymous reviewers for useful comments and suggestions. This research is partially sponsored by *Distributed Production as Innovative System (DIPIS)* and *Telecommunication Facilities and Wireless Sensor Networks in Emergency Management projects*.

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: KR. (2004) 224–234
3. Bacchus, F., Grove, A.: Graphical models for preference and utility. In: UAI. (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. JAIR **21** (2004) 135–191
5. Engel, Y., Wellman, M.P.: CUI networks: A graphical representation for conditional utility independence. JAIR **31** (2008) 83–112
6. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.M., Wellman, M.: Computing utility from weighted description logic preference formulas. In: Declarative Agent Languages and Technologies VII Workshop Notes. `http://www.di.unito.it/~baldoni/DALT-2009/DALT-WorkshopNotes.pdf`.
7. Pinkas, G.: Propositional non-monotonic reasoning and inconsistency in symmetric neural networks. In: IJCAI. (1991) 525–531
8. Lafage, C., Lang, J.: Logical representation of preferences for group decision making. In: KR. (2000) 457–468
9. Chevaleyre, Y., Endriss, U., Lang, J.: Expressive power of weighted propositional formulas for cardinal preference modelling. In: KR. (2006) 145–152
10. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.: A logic-based framework to compute Pareto agreements in one-shot bilateral negotiation. In: ECAI. (2006) 230–234
11. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.M.: Description logics for multi-issue bilateral negotiation with incomplete information. In: AAAI. (2007) 477–482
12. Cadoli, M., Donini, F.M.: A survey on knowledge compilation. AI Comm. **10**(3-4) (1997) 137–150
13. Marquis, P., Sadaoui, S.: A new algorithm for computing theory prime implicates compilations. In: AAAI. (1996) 504–509
14. Bienvenu, M.: Prime implicate normal form for $\mathcal{ALC}$ concepts. In: AAAI. (2008) 412–417
15. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall (1982)
16. Ngair, T.H.: A new algorithm for incremental prime implicate generation. In: IJCAI. (1993) 46–51
17. Dechter, R., Rish, I.: Directional resolution: The Davis-Putnam procedure, revisited. In: KR. (1994) 134–145
18. Fishburn, P.C.: Interdependence and additivity in multivariate, unidimensional expected utility theory. International Economic Review **8** (1967) 335–342

Appendix

N. of prefrences	Movie	Pizza	BuildingsAndPlaces
1	6,87	5,07	10,27
2	5,3	5,78	9,58
3	9,87	7,4	11,16
4	13,32	9,74	15,7
5	23,96	14,74	21,59
6	42,75	26,04	39,73
7	102,08	64,76	95,78
8	241,05	155,17	294,86
9	597,52	398,22	1055,96
10	1484,68	1317,39	2319,38

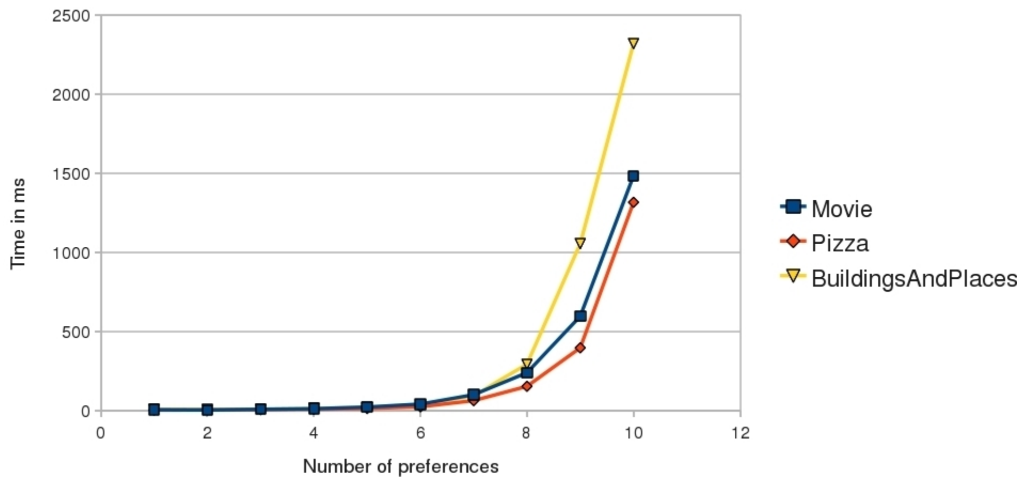

Fig. 2. Execution times (in ms) of $preference\_closure$ using generated testbeds for Pizza, Movie and BuildingAndPlaces ontologies