Alternating-Offers Protocol for Multi-issue Bilateral Negotiation in Semantic-Enabled Marketplaces

Azzurra Ragone[1], Tommaso Di Noia[1], Eugenio Di Sciascio[1], and Francesco M. Donini[2]

[1] SisInfLab, Politecnico di Bari, Bari, Italy
{a.ragone,t.dinoia,disciascio}@poliba.it
[2] Università della Tuscia , Viterbo, Italy
donini@unitus.it

Abstract. We present a semantic-based approach to multi-issue bilateral negotiation for e-commerce. We use Description Logics to model advertisements, and relations among issues as axioms in a TBox. We then introduce a logic-based alternating-offers protocol, able to handle conflicting information, that merges non-standard reasoning services in Description Logics with utility thoery to find the most suitable agreements. We illustrate and motivate the theoretical framework, the logical language, and the negotiation protocol.

1 Introduction

Fully automating negotiation mechanisms in e-marketplaces calls for adopting logical languages to model and reason on advertisements whenever the negotiation has to take place on complex descriptions, going beyond plain undifferentiated goods, where a single issue (usually price) is amenable to negotiation. Description Logics (DLs) are natural candidates languages for this purpose: they are the basis of Semantic Web Languages and they can be much more expressive than *e.g.*, Propositional Logic, yet they have decidable inference problems that can be useful in a number of negotiation scenarios. In this paper, in particular, we present a DL-based approach to multi-issue bilateral negotiation and introduce a novel logic-based alternating-offers protocol. The protocol merges DLs formalism and non-standard reasoning services with utility theory, to find the most suitable agreements. To this aim it takes into account existing logical relations between issues in requests and offers and related utilities of agents, expressed through logical formulas. The roadmap to the rest of this paper is as follows: next Section outlines the approach and assumptions we make. Then we move on to the logical formalism we adopt. Section 3 presents and motivates the protocol we devised. We discuss features of the protocol in Section 4. A brief analysis of relevant related work and a summary of the approach close the paper.

1.1 Scenario and Assumptions

We consider a marketplace of peer entities where users submit their semantically annotated descriptions, and negotiations among agents take place in a fully automated way. We start outlining: the *negotiation protocol*, *i.e.*, the set of rules that specifies how an

K. Aberer et al. (Eds.): ISWC/ASWC 2007, LNCS 4825, pp. 395–408, 2007.

agreement can be reached; the *negotiation strategy*, that specifies the action to take in each situation, given an explicit set of rules specified in the negotiation protocol [15]; the *utility function* of the agents, which is used to evaluate the possible outcomes of the negotiation [11]. The assumptions characterizing the proposed negotiation mechanism are:

one-to-many: the negotiation is a one-to-many negotiation, since the buyer's agent will negotiate simultaneously with other *m* different agents – each one representing a seller, whose offer has been previously stored in the e-marketplace.

rationality: agents are *rational*, they behave according to their preferences and seek to maximize their utilities [11, p.19] doing in each step the minimum possible concession, *i.e.*, the concession involving the minimum utility loss, see protocol Section 3.

incomplete information: each agent knows its utility function and ignores the opponent disagreement thresholds and utility functions.

conflict deal: disagreement is better than an agreement iff the agent's utility over such an agreement is smaller than disagreement thresholds[1] set by the agent before negotiation starts. Therefore when the agent's utility deriving from accepting an agreement (or going on with the negotiation) and opting out it is the same, it will prefer not to opt out [11].

Here we just give a quick outlook of the protocol we propose in the framework, and will thoroughly detail it in Section 3. The protocol is inspired to Rubinstein's alternating-offers one [16]. In that setting an agent starts making an offer to its opponent, who can either accept, make a counter-offer or exit the negotiation. If a counter-offer is made, the negotiation goes on until one of the agents accepts an offer or exits the negotiation. In some cases there is a negotiation *deadline*; if such deadline is reached before one agent has accepted an offer, the negotiation ends in a conflict deal. Our protocol anyway is different from that of Rubinstein; actually we consider: *multi-issue negotiation*: buyer and seller do not negotiate on a single item or on a single bundle of items, but on many issues, which are related with each other through an ontology; such issues may also characterize more complex items (*e.g.*, in a computer store domain a notebook equipped with Wi-Fi adapter and DVD recorder). Note also that at this stage of our work we do not consider a time deadline. The protocol is sorted out by a finite set of steps[2]: the negotiation always terminates because either the agreement has been reached or because one agent opts out. The agent who moves first is selected randomly for each negotiation. At each step the agent who moves has two choices: *concede* or *opt out*, while the other one *stands still*. Agents are forced to concede until a *logical compatibility* is reached between the initial request and the supply, *i.e.*, until the inconsistency sources are eliminated. At each step, amongst all the allowed concessions that satisfy the concession criteria enforced by the protocol, the agent should choose the concession that gives the highest utility to itself: the *minimal concession*. Therefore a concession should be *minimal* w.r.t. the utility loss paid by the agent who makes the

[1] Disagreement thresholds, also called disagreement payoffs, or reservation values, are the minimum utility that each agent requires to pursue a deal [15].

[2] In the following, for the sake of simplicity, we always describe an interaction between only two opposite agents; although we notice that multiple negotiations can be performed at the same time, among *one* agent and *many* candidate partners.

concession [9]. The negotiation ends either if a logical compatibility is reached (*the negotiation succeeds*) or if one agent opts out (*the negotiation ends in a conflict deal*). For what concerns **strategy**, the main target of the agent is to reach the compatibility, because only through compatibility it is possible to reach an agreement. If it is its turn to move, an agent can choose to concede or opt out: if the utility of the agent at that step is smaller than its disagreement threshold, then the agent opts out and the negotiation ends immediately. Otherwise, it will do a concession: the concession is the *minimum possible concession*, that is the concession less decreasing its utility.

We define an agent's utility function over all possibile outcomes [11] as:

$$u^p : \mathcal{A} \cup \{Opt\} \rightarrow \Re \tag{1}$$

where $p \in \{\beta, \sigma\}$ —β and σ stand for buyer and seller respectively— $\mathcal{A}$ is the set of all possible agreements, *Opt* stands for Opt out.

2 Logical Formalism

In this paper we will refer to $\mathcal{AL}(D)$ [1], a fragment of OWL-DL, where besides `owl:Class` and `owl:ObjectProperty`, one is able to express `owl:DataTypeProperty` f (for **F**eatures) on objects such as year of building, length, weight and many others by means of *concrete domains*. Without loss of generality we assume that concrete domains we deal with are admissible[3]. In order to model the domain knowledge and represent relationships among elements, we use an ontology $\mathcal{O}$ containing Concept Inclusion axioms of the form $A \sqsubseteq C$ and $A \equiv C$, where the concept name A can appear only once on the left-had side of axioms. We restrict $\mathcal{O}$ to be acyclic, *i.e.*, the definition of A should not depend on A itself (see [1] for a precise definition). Using $\mathcal{AL}(D)$ it is possible to express subclass relations, disjointness relations involving concept names. As an example consider the following axioms (hereafter we will use DL syntax which results more compact than OWL one):

$$\texttt{CheapPC} \sqsubseteq \texttt{PC} \sqcap (\texttt{price} \leq 600) \quad (Subclass)$$
$$\texttt{WinX} \sqsubseteq \neg\texttt{Unix} \quad (DisjointClasses)$$

Formulas representing demands D and supplies S, are expressed as generic OWL-DL expressions. So, an example description can be the following one:
$\texttt{PC} \sqcap \neg\texttt{Notebook} \sqcap (\texttt{ram} \geq 1024) \sqcap (\texttt{hdd} \leq 160) \sqcap \exists \texttt{hasOS} \sqcap \forall \texttt{hasOS.Linux} \sqcap \exists \texttt{monitor} \sqcap \forall \texttt{monitor.}(\texttt{LCDmonitor} \sqcap \exists \texttt{characteristics} \sqcap \forall \texttt{characteristics.}(\texttt{inch} \geq 17))$
formally modeling this advertisement: "*personal computers, no notebooks, with a RAM of 1 Gbyte (with the possibility to extend), with an hard disk of at most 160 Gbytes and a 17" LCD monitor.*".

Notice that for what concerns numerical properties, also range expressions are allowed in the form $(f \geq n) \sqcap (f \leq m)$. In order to better explain the approach, in the rest of the paper we will refer to ontology $\mathcal{O}$ in Figure 1.

Even though subsumption and satisfiability are basic and useful reasoning tasks in a number of applications, there are typical problems related to negotiation that call for

[3] It is well known that $\Re$ is admissible [2].

$\texttt{HomePC} \sqsubseteq \texttt{PC} \sqcap \exists \texttt{hasOS} \sqcap \exists \texttt{hasProcessor};$	$\texttt{CheapPC} \sqsubseteq \texttt{PC} \sqcap (\texttt{price} \leq 600)$
$\texttt{Pentium} \sqsubseteq \texttt{Intel\_Processor} \sqcap (\texttt{mhz} \geq 3000)$	$\texttt{Centrino} \sqsubseteq \neg \texttt{Pentium}$
$\texttt{WinX} \sqsubseteq \texttt{OperatingSystem};$	$\texttt{Unix} \sqsubseteq \texttt{OperatingSystem}$
$\texttt{Linux} \sqsubseteq \texttt{Unix};$	$\texttt{Unix} \sqsubseteq \neg \texttt{WinX}$

Fig. 1. The ontology used in the examples

non-standard reasoning services. For instance, suppose you have the buyer's agent β with her **D**emand represented by the concept D and the seller's agent σ with his **S**upply represented by S. In case β's request D and σ's offer S are in conflict with each other with respect to the domain knowledge modeled in the ontology $\mathcal{O}$— in formulae $S \sqcap D \sqsubseteq_{\mathcal{O}} \bot$ — how to suggest to β which parts in D are in conflict with S and conversely to σ which parts in S are conflict with D? The above question is very common, among others, in negotiation scenarios where you need to know "what is wrong" between D and S and negotiate on it. In order to give an answer to the previous question and provide explanations, Concept Contraction [5,8] can be exploited.

Concept Contraction. Given two concepts C_1 and C_2 and an ontology $\mathcal{O}$, where $C_1 \sqcap C_2 \sqsubseteq_{\mathcal{O}} \bot$ holds, find two concepts K (for Keep) and G (for Give up) such that both $C_1 \equiv K \sqcap G$ and $K \sqcap C_2 \not\sqsubseteq_{\mathcal{O}} \bot$.

In other words K represents a contraction of C_1 which is satisfiable with C_2, whilst G represents some reason why C_1 and C_2 are not compatible with each other. With Concept Contraction, conflicting information both in β's request w.r.t. σ'supply can be computed and vice versa. Actually, for Concept Contraction minimality criteria have to be introduced. Following the Principle of Informational Economy [10], for G we have to give up as little information as possible. In [5,7] some minimality criteria were introduced and analyzed. In particular, if the adopted DL admits a normal form with conjunctions of concepts as $\mathcal{AL}(D)$, $G_{\exists}$ *minimal* irreducible solutions can be defined.

Definition 1. *Let C_1 and C_2 be two concepts such that $C_1 \sqcap C_2 \sqsubseteq_{\mathcal{O}} \bot$. For the corresponding concept contraction problem $\mathcal{Q}$, we say the solution $\langle G_{irr}, K_{irr} \rangle$ problem is* $G_{\exists}$-irreducible minimal *if the following conditions hold:*

1. *$G_{irr} = \bigsqcap_{i=1\ldots n} G_i$ where both $G_{irr} \sqsubseteq_{\mathcal{O}} \exists R$ and $K_{irr} \not\sqsubseteq_{\mathcal{O}} \exists R$ iff $C_2 \sqsubseteq \forall R.\bot$;*
2. *$K \sqcap G_i \sqcap C_2 \sqsubseteq_{\mathcal{O}} \bot$, for every $i = 1 \ldots n$;*
3. *if $\langle G', K' \rangle$ is a solution to $\mathcal{Q}$ satisfying Condition 1 then $K' \not\sqsubseteq_{\mathcal{O}} K_{irr}$ holds.*

The rationale behind the three conditions in above definition is the following:

Condition 1. This condition is needed in order to avoid solutions, *i.e.*, negotiation outcomes, which could not be useful in the user perspective. Consider the following example referring to the ontology in Figure 1.

$$D = \texttt{HomePC} \sqcap \forall \texttt{hasOS}.\texttt{Linux}$$
$$S = \texttt{PC} \sqcap \forall \texttt{hasOS}.\texttt{WinX}$$

Now, D and S are in conflict with each other w.r.t. $\mathcal{O}$; in order to regain the compatibility, β may contract her request. In this case two possible solutions are:

$$\langle G_1, K_1 \rangle = \langle \texttt{HomePC}, \texttt{PC} \sqcap \exists \texttt{hasProcessor} \sqcap \forall \texttt{hasOS.Linux} \rangle$$
$$\langle G_2, K_2 \rangle = \langle \forall \texttt{hasOS.Linux}, \texttt{HomePC} \rangle$$

The first solution does not satisfy Condition 1 because for the property hasOS both $\texttt{HomePC} \sqsubseteq_{\mathcal{O}} \exists \texttt{hasOS}$ and $\texttt{PC} \sqcap \exists \texttt{hasProcessor} \sqcap \forall \texttt{hasOS.Linux} \not\sqsubseteq_{\mathcal{O}} \exists \texttt{hasOS}$. If β decides to contract her request following this solution, then she keeps all the specifications of requested characteristics (also the one on the operating system) but her choice would lead to a vacuously true agreement on the operating system specification with σ. In fact, $K_1 \sqcap S$ implies $\forall \texttt{hasOS.}\bot$ *i.e.*, no operating system is admitted.

Condition 2. We desire to keep the number of conjuncts in G_{irr} as small as possible in order to avoid redundancies (minimality of G_{irr}). Turning back to the previous example, another solution to the same $\mathcal{Q}$ is $\langle G_3, K_3 \rangle = \langle \forall \texttt{hasOS.}(\texttt{Linux} \sqcap \texttt{Unix}), \texttt{HomePC} \rangle$.

Condition 3. Conversely, we want to keep as much information as possible in K_{irr}. In this example $\langle G_4, K_4 \rangle = \langle \forall \texttt{hasOS.Linux}, \texttt{HomePC} \sqcap \forall \texttt{hasOS.OperatingSystem} \rangle$.

With respect to the above example the only solution satisfying Definition 1 is $\langle G_4, K_4 \rangle$. To compute a $G_\exists$-*irreducible minimal* solution, the algorithm proposed in [6] can be simply adapted and used. Notice that even though within $\mathcal{O}$ in Figure 1 the disjunction relation is between WinX and Unix, the concept contraction procedure suggests to give up $\forall \texttt{hasOS.Linux}$ because Linux is a sub-class of Unix.

2.1 Dealing with Incomplete Information

Information about supply/demand descriptions can be, in our setting, incomplete. This may happen not only because some information may be unavailable, but also because some details have been considered irrelevant by either the seller or the buyer when they submitted their advertisements. For instance, some user may find tiresome to specify a lot of characteristics related to the brand or more technical characteristics of the product the user can be unaware of. Currently, the most common approach to this problem is avoiding incompleteness by forcing the user to fill long and tedious forms. There are several ways to deal with incomplete information and the choice may influence a negotiation. Suppose to have the following two entries in the e-marketplace:

D**:** *I am looking for a PC equipped with Wi-Fi and DVD recorder.*
S**:** *I offer a home PC with Intel Processor equipped with Linux Operating System.*

The above descriptions are then formalized as:

D**:** $\texttt{PC} \sqcap \forall \texttt{hasDevice.WiFi} \sqcap \forall \texttt{hasStorageDevice.DVDrecorder}$
S**:** $\texttt{HomePC} \sqcap \forall \texttt{hasProcessor.Intel} \sqcap \forall \texttt{hasOS.Linux}$

Under an ***open-world assumption*** we have two possible choices. First, we can keep incomplete information as *missing* information: we do not know *e.g.*, if the buyer is not interested in a particular characteristic or he simply has forgotten to specify it. In

this case the system has to contact to buyer/seller to further refine her/his description. Asking users to refine their descriptions before the negotiation process starts seems quite unrealistic, because of the amount of descriptions that can be stored in the system itself. It appears more feasible to leave this phase to a second negotiation stage related to a small subset of supplies/demands. For instance, the ones with the highest utility product. In fact these solutions are known to be Pareto-efficient and "fair", according to Nash [11]. Once buyer and seller have refined their descriptions it is possible to start a new negotiation (the so-called *post-negotiation* phase) where only the *updated* information is negotiated. On the other hand, still in the open-world assumption setting, a second possible choice can be to assume incomplete information as an *any-would-fit* assertion (don't care), so the system should cope with this incompleteness as is. Therefore also this information will be presented in the final agreement. Following the above example the final agreement would be:
$A = \texttt{HomePC} \sqcap \forall \texttt{hasProcessor.Intel} \sqcap \forall \texttt{hasOS.Linux} \sqcap \texttt{PC} \sqcap$
$\forall \texttt{hasDevice.WiFi} \sqcap \forall \texttt{hasStorageDevice.DVDrecorder}$
In the latter case no human intervention is needed. In this paper we take this approach, with agents not caring about missing information.

3 Logic-Based Alternating-Offers Protocol

In this section we model an alternating-offer protocol taking into account the semantics of request and offers as well as the domain knowledge modeled within an ontology in the OWL DL fragment we identified in Section 2, exploiting Concept Contraction. For the sake of clarity and without loss of generality, from now on we consider that the agent entering the marketplace is the buyer β and her potential partners are the sellers' agents σ. The first step of the protocol is the normalization of both β's demand D and σ's supply S. The normalization step substitutes A with $A \sqcap C$ everywhere in a concept, if either $A \sqsubseteq C$ or $A \equiv C$ appears in $\mathcal{O}$, then considers the equivalence $\forall R.(A \sqcap B) \equiv \forall R.A \sqcap \forall R.B$ as a recursive rewrite rule from left to right. After the normalization stage, D is a conjunction of elements in the form:

$$D = \prod_i C_i \tag{2}$$

where $C_i \in \texttt{Norm}$ with

$$\texttt{Norm} = \begin{cases} \left.\begin{array}{r} CN \\ \neg CN \end{array}\right\rangle - CN \text{ is a Class Name} \\ \exists R - R \text{ is an Object Property} \\ \left.\begin{array}{l} (f \leq n) \\ (f \geq n) \\ (f = n) \end{array}\right\rangle - \text{Constraints on numerical features} \\ \forall R.C - \text{with } C \in \texttt{Norm} \end{cases}$$

As an example consider the concept in Section 2. After normalization it is rewritten as:
$\texttt{PC} \sqcap \neg \texttt{Notebook} \sqcap (\texttt{ram} \geq 1024) \sqcap (\texttt{hdd} \leq 160) \sqcap \exists \texttt{hasOS} \sqcap \forall \texttt{hasOS.Linux} \sqcap \exists \texttt{monitor} \sqcap$
$\forall \texttt{monitor.LCDmonitor} \sqcap \forall \texttt{monitor.} \exists \texttt{characteristics} \sqcap$
$\forall \texttt{monitor.} \forall \texttt{characteristics.} (\texttt{inch} \geq 17)$.

In the normalized form C_i represents issues the user is willing to negotiate on. The buyer is able to express her utilities on single issues or on bundles of them. For instance, w.r.t. the previous request the buyer may set utility values on `PC` (single issue) as well as on the whole formula $(\texttt{ram} \geq 1024) \sqcap \forall \texttt{monitor.LCDmonitor}$ (bundle of issue). We indicate these concepts with P_k — for **P**references.

Now, for each P_k the buyer β expresses a utility value $u^\beta(P_k)$ such that $\sum_i u^\beta(P_k) = 1$. As usual, both agents' utilities are normalized to 1 to eliminate outliers, and make them comparable. Since we assume that utilities are additive, the global utility is just a suitable sum of the utilities for preferences entailed by the final agreement. In particular, given a concept expression A representing a final agreement, we define the final utility associated to the agent p, with $p \in \{\beta, \sigma\}$, see (1) as:

$$u^p(A) = \sum_k \{u^p(P_k) \mid A \sqsubseteq P_k\} \qquad u^p(Opt) = t_p \tag{3}$$

where t_p is the **disagreement threshold** of agent p (see Section 1.1).

3.1 The Protocol

Summing up, before the real negotiation starts (step 0) we have a demand D and a supply S such that:

$$D = \bigsqcap_i C_i \qquad\qquad S = \bigsqcap_j C_j$$

Based on C_i and C_j, the buyer and seller, respectively, formulate their preferences P_k (for the buyer) and P_h (for the seller) and for each of them set a utility value such that:

$$\sum_k u^\beta(P_k) = 1 \qquad\qquad \sum_h u^\sigma(P_h) = 1$$

Finally, both for β and σ we have the corresponding **disagreement thresholds** and utility functions t_β, u^β and t_σ, u^σ.
If $D \sqcap S \sqsubseteq_\mathcal{O} \bot$ then demand and supply descriptions are in conflict with each other and β and σ need to negotiate on conflicting information if they want to reach an agreement. The negotiation will follow an alternating offers pattern: at each step, either β or σ gives up a portion of its conflicting information choosing the item with the minimum utility. At the beginning, both β and σ need to know what are the conflicting information. Notice that both agents β and σ know D and S, but they have no information neither on counterpart utilities nor preferences. Both β and σ will solve two Concept Contraction problems, computing a $G_\exists$ *minimal irreducible* solution, and rewrite D and S as:

$$D = G_0^\beta \sqcap K_0^\beta \qquad\qquad S = G_0^\sigma \sqcap K_0^\sigma$$

In the above rewriting G_0^β and G_0^σ represent, respectively, some of the causes that make D in conflict with S and the reason why S is in conflict with D. At a first glance it would seem β needs only $\langle G_0^\beta, K_0^\beta \rangle$ and σ needs $\langle G_0^\sigma, K_0^\sigma \rangle$. We will see later that β

needs also information on σ in order to check its fairness during negotiation steps. Since we compute *G-irreducible* solutions we can normalize G_0^β and G_0^σ, following the same procedure for D and S, as:

$$G_0^\beta = G_{(0,1)}^\beta \sqcap G_{(0,2)}^\beta \sqcap \ldots \sqcap G_{(0,n)}^\beta = \prod_{i=1}^{n} G_{(0,i)}^\beta$$
$$G_0^\sigma = G_{(0,1)}^\sigma \sqcap G_{(0,2)}^\sigma \sqcap \ldots \sqcap G_{(0,m)}^\sigma = \prod_{j=1}^{m} G_{(0,j)}^\sigma$$

In the previous formulas, indexes $(0,i)$ and $(0,j)$ represent the i-th and j-th conjunctive element in G^β and G^σ at round 0. Due to the structure of D, S and $\mathcal{O}$ we have that: *for each $G_{(0,i)}^\beta$ there always exists a C_i in the normalized version of D— as represented in Equation (2) — such that $G_{(0,i)}^\beta = C_i$*. The same relation holds between each $G_{(0,j)}^\sigma$ and C_j in the normalized form of S. Hence, some of P_k and P_h can be partially rewritten in terms of $G_{(0,i)}^\beta$ and $G_{(0,j)}^\sigma$ respectively. Since the information in G_0^β and G_0^σ are the reason why an agreement is not possible, then either β or σ will start conceding one of $G_{(0,i)}^\beta$ or $G_{(0,j)}^\sigma$ reducing their global utility of $u(G_{(0,i)}^\beta)$ or $u(G_{(0,j)}^\sigma)$, respectively.

Suppose β starts the negotiation and gives up $G_{(0,2)}^\beta = C_5$ with $P_3 \sqsubseteq_\mathcal{O} G_{(0,2)}^\beta$. Then she reformulates her request as

$$D_1 = \prod_{i=1..4,6..} C_i$$

and sends it to σ. Notice that since $P_3 \sqsubseteq_\mathcal{O} G_{(0,2)}^\beta$, the global utility of β decreases to

$$u_1^\beta = \sum_{k=1..2,4..} u(P_k)$$

Now, σ is able to validate if β really changed her request to reach an agreement and did not lie. To do so, σ computes $\langle G_1^\beta, K_1^\beta \rangle$ solving a concept contraction problem w.r.t. the new demand D_1 and checks if G_1^β is more general than G_0^β. In formulas, σ checks if $G_0^\beta \sqsubseteq_\mathcal{O} G_1^\beta$ holds, in case of positive answer, then σ knows that β did not lie and he continues the negotiation process. Otherwise he may decide to leave the negotiation (conflict deal) or ask β to reformulate her counteroffer. If the negotiation continues, σ computes his conflicting information w.r.t. to D_1 and rewrites S as:

$$S = G_1^\sigma \sqcap K_1^\sigma \qquad \textit{where} \qquad G_1^\sigma = \prod_{j=1}^{m} G_{(1,j)}^\sigma$$

Again, for each $G_{(1,j)}$ there exists a C_j in the normalized version of S. Hence, if σ decides to concede $G_{(1,j)}$, his global utility decreases proportionally to the utility of P_h to which $G_{(1,j)}$ belongs to. Once σ sends his counteroffer to β, this latter is able to check if

σ lied. Similarly to σ in step 0, β computes $\langle G_1^\sigma, K_1^\beta \rangle$ and checks if $G_0^\sigma \sqsubseteq_\mathcal{O} G_1^\sigma$. The process ends when one of the following two conditions holds:

1. the global utility of an agent is lower than its **disagreement threshold**. In this case the negotiation terminates with a conflict deal.
2. there is nothing more to negotiate on and the global utility of each agent is greater than its disagreement threshold. In this case the negotiation terminates with an agreement. **The agreement** A **is computed** simply as $A = D_{last} \sqcap S_{last}$, where D_{last} and S_{last} are the request and the offer in the last step.

3.2 Minimum Concession

Since users can express an utility value also on bundles, whenever they concede an issue as the **minimum concession** (in term of minimum global utility decrease), the set of all the bundles in which the issue is present has to be taken into account. They choose based on the utility of the whole set. For instance, suppose the buyer sets as preferences the following ones:

$$P_1 = \forall \texttt{monitor.LCDmonitor}$$
$$P_2 = (\texttt{hdd} \leq 200)$$
$$P_3 = \forall \texttt{monitor.LCDmonitor} \sqcap \forall \texttt{monitor.}\forall \texttt{characteristics.}(\texttt{inch} = 17)$$

with the following utilities: $u^\beta(P_1) = 0.1$, $u^\beta(P_2) = 0.4$ and $u^\beta(P_3) = 0.5$.

At the n-th step the conflicting information is:

$$G_n^\beta = \forall \texttt{monitor.LCDmonitor} \sqcap (\texttt{hdd} \leq 200)$$

Hence, β can concede whether $\forall \texttt{monitor.LCDmonitor}$ or $(\texttt{hdd} \leq 200)$. If she concedes $\forall \texttt{monitor.LCDmonitor}$ then her global utility decreases of $u^\beta(P_1) + u^\beta(P_3) = 0.6$, while conceding $(\texttt{hdd} \leq 200)$ her utility decreases of $u^\beta(P_2) = 0.4$ only. In this case the **minimum concession** is $(\texttt{hdd} \leq 200)$.

3.3 The Algorithm

Here we define the behavior of agents during a generic n-th round of the negotiation process. We present only the algorithm related to β's behavior since the behavior of σ is dual w.r.t. β's one.

1-4 If there is nothing in conflict between the old D_{n-1} and just-arrived S_n, then there is nothing more to negotiate on: the agreement is reached and returned. Notice that while computing the final agreement we use the "any-would-fit" approach to deal with *incomplete information* (see Section 2.1).

5-11 If β discovers that σ lied on his concession, then β decides to exit the negotiation and terminates with a conflict deal. If we want β ask σ to concede again it is straightforward to change the protocol to deal with such a behavior.

13-15 If after the minimum concession, the utility of β is less than her **disagreement threshold**, then the negotiation ends with a conflict deal.

if $D_{n-1} \sqcap S_n \not\sqsubseteq_{\mathcal{O}} \bot$ **then**
  agreement A reached;
  return $A = D_{n-1} \sqcap S_n$;
end
if $n > 0$ **then**
  compute $\langle G_n^{\sigma}, K_n^{\sigma} \rangle$ from D_{n-1} and S_n;
  if $G_{n-1}^{\sigma} \not\sqsubseteq_{\mathcal{O}} G_n^{\sigma}$ **then**
    σ lied;
    conflict deal: **exit**;
  end
end
compute minimum concession $G^{\beta}_{(n-1,i)}$;
if $u^{\beta}_{n-1} < t^{\beta}$ **then**
  conflict deal: **exit**;
end
formulate D_n deleting $G^{\beta}_{(n-1,i)}$ from D_{n-1};
send D_n to σ ;

Algorithm 1. The behavior of β at step n

3.4 An Illustrative Example

Keeping the computer equipment as reference domain, consider the following example: a buyer β looking for a "*personal computer equipped with an Intel processor whose clock frequency is at least 3000 MHz, 2 Giga bytes of RAM. The computer must have Linux operating system pre-installed. A 19" LCD monitor is also requested.*". On the other side, a seller offers a "*personal computer for domestic use with a 2500 MHz Pentium on board. The computer is provided with a Windows operating system, a 17" LCD monitor and a WiFi adapter.*". Both the request D and the offer S can be formalized as:

$D =$ $\texttt{PC} \sqcap \exists \texttt{hasProcessor} \sqcap \forall \texttt{hasProcessor}.(\texttt{Intel\_Processor} \sqcap (\texttt{mhz} \geq 3000)) \sqcap \exists \texttt{hasOS} \sqcap \forall \texttt{hasOS}.\texttt{Linux} \sqcap (\texttt{hdd} \geq 100) \sqcap \exists \texttt{monitor} \sqcap \forall \texttt{monitor}.(\texttt{LCDmonitor} \sqcap \exists \texttt{characteristics} \sqcap \forall \texttt{characteristics}.(\texttt{inch} = 19)) \sqcap (\texttt{ram} = 2048)$

$S =$ $\texttt{HomePC} \sqcap \forall \texttt{hasProcessor}.(\texttt{Pentium} \sqcap (\texttt{mhz} = 2500)) \sqcap \forall \texttt{hasOS}.\texttt{WinX} \sqcap (\texttt{hdd} = 80) \sqcap \exists \texttt{monitor} \sqcap \forall \texttt{monitor}.(\texttt{LCDmonitor} \sqcap \exists \texttt{characteristics} \sqcap \forall \texttt{characteristics}.(\texttt{inch} = 17)) \sqcap \exists \texttt{hasWiFi}$

Their normalized forms[4] are respectively:

$D =$ $\texttt{PC} \sqcap \exists \texttt{hasProcessor} \sqcap \forall \texttt{hasProcessor}.\texttt{Intel\_Processor} \sqcap \forall \texttt{hasProcessor}.(\texttt{mhz} \geq 3000) \sqcap \exists \texttt{hasOS} \sqcap \forall \texttt{hasOS}.\texttt{Linux} \sqcap (\texttt{hdd} \geq 100) \sqcap \exists \texttt{monitor} \sqcap \forall \texttt{monitor}.\texttt{LCDmonitor} \sqcap \forall \texttt{monitor}.\exists \texttt{characteristics} \sqcap \forall \texttt{monitor}.\forall \texttt{characteristics}.(\texttt{inch} = 19) \sqcap (\texttt{ram} = 2048)$

[4] In order to keep the example compact, we do not consider here the normalization step $(f = n) \equiv (f \geq n) \sqcap (f \leq n)$, which should be taken into account.

S = $\texttt{HomePC} \sqcap \forall \texttt{hasProcessor}.\texttt{Pentium} \sqcap \forall \texttt{hasProcessor}.(\texttt{mhz} = 2500) \sqcap \forall \texttt{hasOS}.\texttt{WinX} \sqcap (\texttt{hdd} = 80) \sqcap \exists \texttt{monitor} \sqcap \forall \texttt{monitor}.\texttt{LCDmonitor} \sqcap \forall \texttt{monitor}.\exists \texttt{characteristics} \sqcap \forall \texttt{monitor}.\forall \texttt{characteristics}.(\texttt{inch} = 17) \sqcap \exists \texttt{hasWiFi}$

Now both agents β and σ set their preferences with corresponding utilities and utility threshold.

$$
\begin{aligned}
P_1^\beta &= \forall \texttt{monitor}.\texttt{LCDmonitor} && u^\beta(P_1^\beta) = 0.4 \\
P_2^\beta &= (\texttt{hdd} \geq 100) && u^\beta(P_2^\beta) = 0.2 \\
P_3^\beta &= \forall \texttt{hasOS}.\texttt{Linux} && u^\beta(P_3^\beta) = 0.2 \\
P_4^\beta &= \forall \texttt{monitor}.\texttt{LCDmonitor} \sqcap \\
&\quad \forall \texttt{monitor}.\forall \texttt{characteristics}.(\texttt{inch} = 19) && u^\beta(P_4^\beta) = 0.1 \\
P_5^\beta &= (\texttt{ram} = 2048) && u^\beta(P_4^\beta) = 0.1 \\
t^\beta &= 0.6
\end{aligned}
$$

$$
\begin{aligned}
P_1^\sigma &= \forall \texttt{monitor}.\forall \texttt{characteristics}.(\texttt{inch} = 17) && u^\sigma(P_1^\sigma) = 0.4 \\
P_2^\sigma &= \forall \texttt{hasProcessor}.(\texttt{mhz} = 2500) && u^\sigma(P_2^\sigma) = 0.3 \\
P_3^\sigma &= (\texttt{hdd} = 80) && u^\sigma(P_3^\sigma) = 0.2 \\
P_4^\sigma &= \forall \texttt{hasOS}.\texttt{WinX} && u^\sigma(P_4^\sigma) = 0.1 \\
t^\sigma &= 0.5
\end{aligned}
$$

K and G are computed for both β and σ.

K_0^β = $\texttt{PC} \sqcap \exists \texttt{hasProcessor} \sqcap \forall \texttt{hasProcessor}.\texttt{Intel\_Processor} \sqcap \exists \texttt{hasOS} \sqcap \forall \texttt{hasOS}.\texttt{OperatingSystem} \sqcap \exists \texttt{monitor} \sqcap \forall \texttt{monitor}.\exists \texttt{characteristics} \sqcap \forall \texttt{monitor}.\texttt{LCDmonitor} \sqcap (\texttt{ram} = 2048)$

G_0^β = $\forall \texttt{monitor}.\forall \texttt{characteristics}.(\texttt{inch} = 19) \sqcap \forall \texttt{hasProcessor}.(\texttt{mhz} \geq 3000) \sqcap \forall \texttt{hasOS}.\texttt{Linux} \sqcap (\texttt{hdd} \geq 100)$

K_0^σ = $\texttt{HomePC} \sqcap \forall \texttt{hasProcessor}.\texttt{Pentium} \sqcap \exists \texttt{monitor} \sqcap \forall \texttt{monitor}.\texttt{LCDmonitor} \sqcap \forall \texttt{monitor}.\exists \texttt{characteristics} \sqcap \exists \texttt{hasWiFi}$

G_0^σ = $\forall \texttt{monitor}.\forall \texttt{characteristics}.(\texttt{inch} = 17) \sqcap \forall \texttt{hasProcessor}.(\texttt{mhz} = 2500) \sqcap (\texttt{hdd} = 80) \sqcap \forall \texttt{hasOS}.\texttt{WinX}$

Now suppose that by coin tossing, β moves first. She starts giving up the constraint on processor clock frequency, which is her minimum concession since a utility not assigned to a characteristic is usually equivalent to a utility equal to zero. Then she computes her utility and, since it is greater than the threshold value, decides to go on with the negotiation process. In the following step we have:

K_1^β = $\texttt{PC} \sqcap \exists \texttt{hasProcessor} \sqcap \forall \texttt{hasProcessor}.\texttt{Intel\_Processor} \sqcap \exists \texttt{hasOS} \sqcap \forall \texttt{hasOS}.\texttt{OperatingSystem} \sqcap \exists \texttt{monitor} \sqcap \forall \texttt{monitor}.\exists \texttt{characteristics} \sqcap \forall \texttt{monitor}.\texttt{LCDmonitor} \sqcap (\texttt{ram} = 2048)$

G_1^β = $\forall \texttt{monitor}.\forall \texttt{characteristics}.(\texttt{inch} = 19) \sqcap \texttt{hasOSLinux} \sqcap (\texttt{hdd} \geq 100)$

K_1^σ = $\texttt{HomePC} \sqcap \forall \texttt{hasProcessor}.\texttt{Pentium} \sqcap \exists \texttt{monitor} \sqcap \forall \texttt{monitor}.\texttt{LCDmonitor} \sqcap \forall \texttt{monitor}.\exists \texttt{characteristics} \sqcap \exists \texttt{hasWiFi} \sqcap \forall \texttt{hasProcessor}.(\texttt{mhz} = 2500)$

G_1^σ = $\forall$monitor.$\forall$characteristics.(inch $= 17) \sqcap$ (hdd $= 80) \sqcap \forall$hasOS.WinX

At this point, σ gives up $\forall$hasOS.WinX which is the preference with the minimum utility. The protocol continues until agents reach logical compatibility. A final agreement could then be:

A = HomePC$\sqcap\exists$hasOS$\sqcap\forall$hasOS.Linux$\sqcap\exists$hasProcessor$\sqcap\forall$hasProcessor.(mhz $=$ 2500) $\sqcap\forall$hasProcessor.Pentium $\sqcap$ (hdd $= 80) \sqcap$ (ram $= 2048)\sqcap$ $\forall$monitor.LCDmonitor $\sqcap\forall$monitor.$\exists$characteristics$\sqcap$ $\forall$monitor.$\forall$characteristics.(inch $= 19) \sqcap \exists$hasWiFi,

with corresponding utilities $u^\beta = u^\beta(P_1^\beta) + u^\beta(P_3^\beta) + u^\beta(P_4^\beta) + u^\beta(P_5^\beta) = 0.8$ for β and $u^\sigma = u^\sigma(P_2^\sigma) + u^\sigma(P_3^\sigma)$ = 0.5 for σ.

4 Features of the Negotiation Mechanism

We briefly point out some characteristics of the proposed negotiation protocol [15]. **Semantics**. In the proposed protocol a formal language is exploited. Using a domain ontology it is possible to discover implicit conflicting information between a demand D and a supply S. Furthermore, a general logic-based technique, *i.e.*, Concept Contraction, grounded in well-known belief revision theory [10], is adopted to compute concessions. **Bundles**. Users (both β and σ) can set utility values not only on single issues but also on a bundle of issues. **Strong Agreement**. During the negotiation process the agents negotiate on conflicting issues. Hence, if the negotiation does end with an outcome, no conflicting information are present in the final agreement. **Simplicity**. Interaction among agents requires low communication costs: agents do not have to guess either the preferences of their opponent or their utility function. Moreover it is possible to compute strategy in a reasonable amount of time [11]. **Distribution**. The negotiation mechanism does not require a mediator helping parties to reach an agreement during the negotiation process or managing the entire process. Agents negotiate on their own with no need of an external help. **Efficiency**. If the protocol ends with an agreement, this is on the Pareto frontier. Due to the structure of formulas β and σ and *G-irreducible* solutions to concept contraction problems, during round n for each $G_{(n,i)}^\beta \in G_n^\beta$, representing the source of information in D_n, there is always at least one (in the most general case, more than one) $G_{(n,j)}^\sigma \in G_n^\sigma$ representing the corresponding source of information in S_n. Hence, if β concedes $G_{(n,i)}^\beta$, losing the utility of bundles containing $G_{(n,i)}^\beta$, then σ is sure to maintain the utility on bundles involving $G_{(n,j)}^\sigma$. In other words, if an agent concedes something decreasing its utility then the opponent's utility does not decrease. Notice that, in order to reach a surely Pareto-efficient agreement and always find a Nash bargain solution, while maintaining the same logic approach and the use of bundles, a protocol similar to the one proposed in [9] can be adopted. Nevertheless, if we want to consider the *willingness to risk conflict* of both agents, in order to decide for each round which agent will concede, it becomes hard to maintain the approach fully distributed. If the approach remains distributed then we have to hypothesize agents estimate the willingness to risk conflict of the opponent. In this case we would probably lose in simplicity.

5 Related Work and Summary

Several recent logic-based approaches to negotiation are based on propositional logic. In [3], Weighted Propositional Formulas (WPF) are used to express agents preferences in the allocation of indivisible goods, but no common knowledge (as our TBox) is present. Utility functions expressed through WPF are classified in [4] according to the properties of the utility function (sub/super-additive, monotone, etc.). We used the most expressive functions according to that classification, namely, weights over unrestricted formulas, but for the fact that our formulas are DL concepts—*i.e.*, non-propositional. In [17] an agreement is defined as a model for a set of formulas from both agents, but agents preferences are not taken into account. In [12] a propositional logic framework endowed of an ontology $\mathcal{T}$ is proposed, where a one-shot protocol is exploited to reach Pareto-efficient agreements. In order to reach a Pareto-efficient agreement a trusted mediator is needed, to whom agents reveal their preferences, and which suggests to the agents the most suitable agreement. The framework in [12] was further improved in [14], extending propositional logic with concrete domains in order to handle also numerical features as price, weight, time etc. In this work, instead, no mediator is needed. The work presented in [18] adopts a kind of propositional knowledge base arbitration to choose a fair negotiation outcome. However, *common knowledge* is considered as just more entrenched preferences, that could be even dropped in some deals. Instead, the logical constraints in our ontology $\mathcal{T}$ of formulas must *always* be enforced in the negotiation outcomes. Finally we devised a *protocol* which the agents should adhere to while negotiating; in contrast, in [18] a game-theoretic approach is taken, presenting no protocol at all, since communication between agents is not considered. Although we used a rather inexpressive DL, our approach can be easily extended up to $\mathcal{ALEH}(D)$, which can express qualified existential concepts and sub-properties. Summarizing, we have motivated and illustrated a logic-based approach to bilateral negotiation in P2P e-marketplaces; we proposed a semantic-based alternating-offers protocol exploiting Description Logics, non-standard inference services, and utility theory to find the most suitable agreements. To the best of our knowledge there is only another work exploiting DLs in negotiation scenarios [13]. In that work the more expressive *SHOIN(D)* is used to model the logic-based negotiation protocol, and only standard inference services, such a satisfiability, are exploited in order to catch inconsistency sources between D and S. Instead, the use of a non-standard inference service, such a Concept Contraction, can be useful to provide also an explanation of "what is wrong" between D and S, *i.e.*, the reason why β and σ can not reach an agreement and *what* has to be given up in order to reach that. However, in this paper we model a scenario with *partial* incomplete information (agents know opponent preferences, but not utilities of such preferences), while in [13] a scenario with *fully* incomplete information is studied, where agents do not know anything about the opponent one (neither preferences nor utilities). Moreover, differently from the approach presented in [13], the negotiation mechanism also works without agents knowing their *exact* utilities. It is enough that each agent knows, at every round, which issue to concede next; so only partial orders on issues are needed. Work is ongoing on various directions, namely: extending the DL adopted, finding a "cheap" way to ensure that the reached agreement is Pareto-efficient, and carry out large-scale experiments with real advertisements. We also plan to apply our approach to

Semantic Web Services (SWS) contracting, for negotiating Service Level Agreements (SLA) between users and SWS providers.

Acknowledgments

We acknowledge support of project EU-FP-6-IST-026896 "TOWL". We are grateful to one anonymous reviewer for useful suggestions.

References

1. Baader, F., Calvanese, D., Mc Guinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook. Cambridge University Press, Cambridge (2002)
2. Baader, F., Hanschke, P.: A schema for integrating concrete domains into concept languages. In: proc. of IJCAI 1991, pp. 452–457 (1991)
3. Bouveret, S., Lemaitre, M., Fargier, H., Lang, J.: Allocation of indivisible goods: a general model and some complexity results. In: Proc. of AAMAS 2005, pp. 1309–1310 (2005)
4. Chevaleyre, Y., Endriss, U., Lang, J.: Expressive power of weighted propositional formulas for cardinal preference modeling. In: Proc. of KR 2006, pp. 145–152 (2006)
5. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., Mongiello, M.: Concept Abduction and Contraction in Description Logics. In: Proc. of DL 2003 (2003)
6. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., Mongiello, M.: A uniform tableaux-based method for concept abduction and contraction in description logics. In: Proc. of ECAI 2004, pp. 975–976 (2004)
7. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., Mongiello, M.: Concept Abduction and Contraction for Semantic-based Discovery of Matches and Negotiation Spaces in an E-Marketplace. Electronic Commerce Research and Applications 4(4), 345–361 (2005)
8. Di Noia, T., Di Sciascio, E., Donini, F.: Semantic Matchmaking as Non-Monotonic Reasoning: A Description Logic Approach. Journal of Artificial Intelligence Research 29, 269–307 (2007)
9. Endriss, U.: Monotonic concession protocols for multilateral negotiation. In: Proc. of AAMAS 2006, pp. 392–399 (2006)
10. Gärdenfors, P.: Knowledge in Flux: Modeling the Dynamics of Epistemic States, Bradford Books. MIT Press, Cambridge (1988)
11. Kraus, S.: Strategic Negotiation in Multiagent Environments. MIT Press, Cambridge (2001)
12. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.: A logic-based framework to compute pareto agreements in one-shot bilateral negotiation. In: Proc. of ECAI 2006, pp. 230–234 (2006)
13. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.: Description logics for multi-issue bilateral negotiation with incomplete information. In: proc. of AAAI 2007, pp. 477–482 (2007)
14. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.: When price is not enough: Combining logical and numerical issues in bilateral negotiation. In: Proc. of AAMAS 2007, pp. 97–99 (2007)
15. Rosenschein, J.S., Zlotkin, G.: Rules of Encounter. MIT Press, Cambridge (1994)
16. Rubinstein, A.: Perfect equilibrium in a bargaining model. Econometrica 50, 97–109 (1982)
17. Wooldridge, M., Parsons, S.: Languages for negotiation. In: Proc of ECAI 2004, pp. 393–400 (2000)
18. Zhang, D., Zhang, Y.: A computational model of logic-based negotiation. In: Proc. of AAAI 2006, pp. 728–733. AAAI Press, Stanford, California, USA (2006)