Ontology-Based Natural Language Parser for E-Marketplaces

S. Coppi[1], T. Di Noia[1], E. Di Sciascio[1], F. M. Donini[2], A. Pinto[1]

[1] Politecnico di Bari, Via Re David, 200, I-70125, Bari, Italy
{s.coppi,t.dinoia,disciascio,agnese.pinto}@poliba.it
[2] Università della Tuscia, via San Carlo, 32, I-01100, Viterbo, Italy
donini@unitus.it

Abstract. We propose an approach to Natural Language Processing exploiting knowledge domain in an e-commerce scenario. Based on such modeling an NLP parser is presented, aimed at translating demand/supply advertisements into structured Description Logic expressions, automatically mapping sentences with concept expressions related to a reference ontology.

1 Introduction

We focus on an approach specifically aimed at translating demand / supply descriptions expressed in Natural Language (NL) into structured Description Logic (DL) expressions, mapping in an automated way NL sentences with concepts and roles of a DL-based ontology. Motivation for this work comes from the observation that one of the major obstacles to the full exploitation of semantic-based e-marketplaces, particularly B2C and P2P ones, lies in the difficulties average users have in translating their advertisements into cumbersome expressions or in filling several form-based web pages. Yet constraining a user to completely fill in forms is in sharp contrast with the inherent Open World Assumption typical of Knowledge Representation systems. We report here how we faced this issue in the framework of MAMAS demand/supply semantic-matchmaking service [11]. Distinguishing characteristics of our NL parser include the direct use of DLs to express the semantic meaning, without intermediate stages in First Order Logic Form or Lambda calculus. This has been possible because of the strong contextualization of the approach, oriented to e-commerce advertisements, which possess an ontological pattern that expresses their semantics and affects grammar creation. Such pattern is reflected both in the structure of the ontologies we built for e-commerce tasks and in the creation of the grammars. Two separate lexical category sets are taken into account; the first one for goods, the second one for their description. This choice allows to embed the problem domain into the parser grammar. Furthermore we designed the grammar in two separate levels. In this way we achieve more flexibility: the first level only depends on the ontology terminology, while the second one only on the particular DL used. Finally, our parser performs automatic disambiguation of the parsed sentences, interacting with the reasoner.

2 Description Logics and Natural Language Processing

To make the paper self-contained we begin by briefly revisiting fundamentals of DLs [3]. The basic syntax elements are *concept* names, such as, `CPU`, `device`; *role* names, such as `hasSoftware`, `hasDevice`; *individuals*, such as `HPworkstationXW`, `IBMThinkPad`. Concepts stand for sets of objects, and roles link objects in different concepts. Individuals are used for special named elements belonging to concepts. Formally, a semantic *interpretation* is a pair $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, which consists of the *domain* Δ and the *interpretation function* $\cdot^{\mathcal{I}}$, which maps every concept to a subset of Δ, every role to a subset of $\Delta \times \Delta$, and every individual to an element of Δ. The *Unique Name Assumption* (UNA) restriction is usually made, *i.e.*, different individuals are mapped to different elements of Δ, *i.e.*, $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for individuals $a \neq b$. Basic elements can be combined using *constructors* to form concept and role *expressions*, and each DL is identified by the operators set it is endowed with. Every DL allows one to form a *conjunction* of concepts, usually denoted as $\sqcap$; some DL include also disjunction $\sqcup$ and complement $\neg$ to close concept expressions under boolean operations. Expressive DLs [3] are built on the simple $\mathcal{AL}$ (Attributive Language) adding constructs in order to represent more expressive concepts. Allowed constructs in $\mathcal{AL}$ are: $\top$ *universal concept* (all the objects in the domain); $\bot$ *bottom concept* (the empty set); A *atomic concepts* (all the objects belonging to the set represented by A); $\neg A$ *atomic negation* (all the objects not belonging to the set represented by A); $C \sqcap D$ *intersection* (the objects belonging both to C and D); $\forall R.C$ *universal restriction* (all the objects participating to the R relation whose range are all the objects belonging to C); $\exists R$ *unqualified existential restriction* (there exists at least one object participating in the relation R). Expressions are given a semantics by defining the interpretation function over each construct. Concept conjunction is interpreted as set intersection: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and also the other boolean connectives $\sqcup$ and $\neg$, when present, are given the usual set-theoretic interpretation of union and complement. The interpretation of constructs involving quantification on roles needs to make domain elements explicit: for example, $(\forall R.C)^{\mathcal{I}} = \{d_1 \in \Delta \mid \forall d_2 \in \Delta : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow d_2 \in C^{\mathcal{I}}\}$. Concept expressions can be used in *inclusion assertions*, and *definitions*, which impose restrictions on possible interpretations according to the knowledge elicited for a given domain. The semantics of inclusions and definitions is based on set containment: an interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies a definition $C = D$ when $C^{\mathcal{I}} = D^{\mathcal{I}}$. A *model* of a TBox $\mathcal{T}$ is an interpretation satisfying all inclusions and definitions of $\mathcal{T}$. Adding new constructors to $\mathcal{AL}$ increases DL languages expressiveness, but may also make inference services intractable [5]. The allowed operators in a DL based an $\mathcal{AL}$ are indicated by a capital letter. For instance, $\mathcal{ALN}$ is a $\mathcal{AL}$ endowed with unqualified number restriction *i.e.*, $(\geq\ n\ R)$, $(\leq\ n\ R)$, $(=\ n\ R)$ (respectively the minimum, the maximum and the exact number of objects participating in the relation R); $\mathcal{ALC}$ allows full negation; in $\mathcal{ALE}$ there can be used the qualified existential restriction; in $\mathcal{ALEN}$ both existential and unqualified number restriction are defined and so on. Here we refer mainly to an $\mathcal{ALN}$ DL, which can be mapped in a subset of OWL-DL

[9]. Since the early days of terminological reasoners, DLs have been applied in semantic interpretation for natural language processing [12]. Semantic interpretation is the derivation process from the syntactic analysis of a sentence to its logical form – intended here as the representation of its context-dependent meaning. Typically, DLs have been used to encode in a knowledge base both syntactic and semantic elements needed to drive the semantic interpretation process. Several studies have been carried out aimed at building a good DL knowledge base for natural language processing [6,7]. A linguistically well motivated ontology ought to be partitioned into a language-dependent part (the *upper model*) and a domain-dependent part (the *domain model*), but it is well known this result is theoretically very hard to achieve. Implemented systems rely on the so-called *multilevel semantics architecture* [1]. For a recent survey of NLP projects using DLs, see Chapter 15 in [3].

3 A Grammar for Parsing E-commerce Advertisements

We started analyzing several advertisements related to different commerce domain *e.g.*, consumer electronics components, real estate services, job postings. As we expected, we noticed that advertisements present almost always, regardless of the domain, a characteristic structure and are strongly contextualized. Furthermore the lexicon often uses some jargon and is a finite and limited set of terms. With reference to the structure, there is always the good(s) to be bought/sold and related characteristics. Each good in the domain refers to a single *concept* in the knowledge domain but can be represented using different expressions, which are semantically equivalent. The same can be said for good characteristics. Hence, in each sentence there are at least two main lexical category: the good and its description. From a DL point of view, generic advertisement can be brought back to the following form:

$$C_1 \sqcap C_2 \sqcap ...C_n \sqcap \forall r_1.D_1 \sqcap \forall r_2.D_2 \sqcap ...\forall r_m.D_m$$

where C_i are the concepts related to the goods, and $\forall r_j.D_j$ to the goods description. This pattern can be also used as a guideline to model the task ontology for the specific marketplace. Atomic concepts representing a good are modeled as sub-concepts of a generic `Goods` concept. Notice that at least an $\mathcal{ALN}$ DL is needed to model a marketplace, in order to deal with concept taxonomy, disjoint groups, role restrictions ($\mathcal{AL}$), and particularly number restriction ($\mathcal{N}$) to represent quantity. The sentence structure led us to investigate techniques similar to *Semantic Grammars* [2] ones, where the lexical categories are based on the semantic meaning. We created two basic lexical category sets. One related to what we call Fundamental Nouns (FN), denoting nouns representing goods, the other one related to what we simply call Nouns (N), denoting nouns describing goods. The lexical categories built based on Ns can be identified because their names start with a capital D. For instance DP corrsponds to the *classical* NP but related to a noun phrase representing a good description. This distinction

is useful during grammar rules composition (see 3.1) because it allows to determine if a sentence is acceptable or not in our scenario. It must contain at least a constituent of category FN, otherwise it means there are no goods to look for. Since the idea was to bind the grammar to the reference DL ontology, we enforced the relationship using features identifying the role of lexical categories within the ontology itself. In a way inspired by the use of a **TYPE** feature in a *Template Matching* [2] approach, we created three different features, respectively for concept names (`concept`), role names (`role`), operators (`op`), whose value is strictly related to the terminology used in the ontology. Using such features it is possible both to align the lexicon with the terms in the ontology and to obtain a limited number of rules associating a semantic meaning to the constituents.

3.1 Lexicon and Grammars

With the aim of building reusable elements to be easily adapted for different marketplaces and ontologies, we separated information related to the terminology, the lexical category of the terms, and the expressiveness of the DL used to model the ontology. The idea is to minimize changes and possibly to reuse both the lexical and the semantic information. In fact the parsing process is conceived in two stages, each one using a different (kind of) grammar. Using the first grammar, terms in the NL sentence are strictly related both to the terminology used in the ontology –atomic concept names and role names– and to the logical operators. With the Level 1 Grammar a parser is able to bind set of words to the correspondent element in the ontology. The Level 2 grammar uses the intermediate result produced during the Level 1 phase to build the logical form of the sentence with respect to a good/description model. In this parsing phase logical operators and quantifiers allowed by the DL used to built the ontology are used to link the basic elements. This subdivision allows more flexibility. Adapting the grammar to a new ontology (based on the same DL) requires major changes only in the Level 1 grammar, in which concept and role names appear, in order to remap the new Lexicon to the terminology used in the ontology. On the other hand if the adopted DL is changed, *e.g.*, from a $\mathcal{ALN}$ DL to a $\mathcal{ALEN}$ DL [3], major changes are requested only for Level 2 rules.
In the following we show how the logical form of the sentence is built with the aid of some examples, conceived with reference to the toy ontology in Fig. 1 [3].

Lexicon First of all let us point out that, at the current stage of our work, we do not carry out any morphological analysis. In the lexicon, each term is endowed with the following features:

- `cat` represents the lexical category of the single word, *i.e.*, FN (noun indicating goods), N (noun describing goods), V (verb), ADJ (adjective), ADJN

[3] In the ontology, for the sake of clarity, we do not model also `Processor`, `Monitor`, `Storage_Device` as subconcept of `Goods`. Even if in a real computer marketplace scenario these can be modeled as `Goods` to be sold/bought.

AMD_Athlon_XP ⊑ Processor
Intel_Pentium4 ⊑ Processor
Intel_Celeron ⊑ Processor
CD_Reader ⊑ Storage_Device
CRT_monitor ⊑ Monitor
LCD_monitor ⊑ Monitor
CRT_monitor ⊑ ¬LCD_monitor
Computer ⊑ Goods
Desktop_Computer ⊑ Computer⊓(= 1 hasCPU)⊓∀hasCPU.Processor⊓∃hasComponent⊓
∀hasComponent.Monitor ⊓ ∃RAM
Notebook ⊑ Desktop_Computer ⊓ ∀hasComponent.LCD_monitor
Server ⊑ Computer ⊓ ∀hasCPU.Processor ⊓ (≥ 2 hasCPU) ⊓ ∀RAM.(≥ 1 0)00mb)
Monitor ⊑ ¬Processor
Monitor ⊑ ¬Storage_Device
Storage_Device ⊑ ¬Processor

Fig. 1. The toy ontology used for examples

(numerical adjective), ADV (adverb), ART (article), CONJ (conjunction), PREP (preposition).
- `concept,role` represent, respectively, the corresponding atomic concept, role in the ontology.
- `op` represents the corresponding logical operator in DL.
- `sw`, is set `true` if the term is a *stopword*.
- `aux` is an auxiliary field for a further customization of the grammars.

Level 1 Grammar Actually, the mapping between the terms in the NL sentence and the ones in the ontology is not in a one to one relationship. There is the need to relate words set to the same concept or role within the ontology. In Fig. 2 a simple grammar is reported to deal with sentences related to our reference computer domain (see Fig. 1).

1) DPF[c,r,-] → N[c,r,-]
2) DP[-,r,x] → N[-,r,x]
3) DP[-,r,-] → N[-,r,-]
4) NP[c,-,-] → FN[c,-,-]
5) DP[-,r2,c1] → ADJN[c1,-,-] N[-,r2,-]
6) NP[concat(c1,c2),-,-] → N[c1=Desktop,-,-] FN[c2=Computer,-,-]
7) DP[-,hdd,-] → ADJ[-,r1=hard,-] N[-,r2=disk,-]
8) DPF[concat(c1,c2),r,-] → N[c1=LCD,r,-] N[c1=monitor,r,-]
9) DPF[concat(c1,c2),r,-] → V[-,r=hasStorageDevice,-] N[c1=CD,-,-] N[c1=Reader,-,-]

Fig. 2. Example Level 1 Grammar Rules

1) 2) 3) 4) map nouns N,FN to constituents NP,DP,DPF, which can contain more than one noun.

6) 7) 8) 9) deal with elements in the ontology represented by two or more words in the sentence. In particular, Rule 9) represents a role with its filler.
5) since number restriction are needed in e-commerce scenarios, as good descriptions, we allow to introduce them in this grammar. Role 5) creates a new DP constituent linking the role `mb` to its numerical restriction, *e.g.*, ($\geq$ 256 `mb`).

Level 2 Grammar This grammar binds the sentence to the expressiveness of the DL chosen to model the ontology. The purpose of Level 2 rules is to put together single concepts and roles of the ontology, to form an expression in DL representing the logical model of the sentence, reflecting the structure of the good/description ontological pattern. With respect to the rules in Fig. 3 we obtain:

1) 2) 3) introduce the DL operators $\geq$ and $\forall$. Rule 1) states that if there is a constituent DPF, *e.g.*, with `role`="`hasComponent`" and `concept`="`LCD_monitor`", a new DPA (a descriptive constituent) is created with `concept` containing the DL expression: $\forall$`hasComponent.LCD_monitor`. The distinction, inspired by the *Semantic Grammars* approach, is useful to reduce ambiguity in the resulting logical form. In a similar way rule 2) introduces the operator ($\geq$ n R) and the DPL category containing this operator. Rule 3) manages the case of an ($\geq$ n R) nested in a $\forall R.C$ expression such as $\forall$`RAM.`($\geq$ 256 `mb`).
4) 6) are useful to compose contiguous constituents of the same type.
5) 7) state that a sentence is composed by a constituent NP representing the good of the advertisement, followed by descriptive constituents DPA or DPC.

1) DPA[(all r c)] → DPF[c,r]
2) DPL[(atLeast x r)] → DP[-,r,x]
3) DPA[(all r2 c1)] → DPL[c1,-] DP[-,r2]
4) DPC[c1 c2] → DPA[c1,-] DPL[c2,-]
5) S[(And c1 c2 c3)] → DPC[c1,-] NP[c2,-] DPA[c3,-]
6) DPA[c1 c2] → DPA[c1,-] DPA[c2,-]
7) S[(And c1 c2)] → NP[c1,-] DPA[c2,-]

Fig. 3. Example Level 2 Grammar Rules

3.2 Ambiguity Resolution Through Filtering

After the parsing process, more then one DL logical expression –corresponding to the NL sentence– can be produced. Interacting with the DL reasoner, the parser is able to reduce the number of expression to just one, thanks to the domain knowledge. This is performed through the application of a sequence of post-processing filters.

1. *Removing unsatisfiable descriptions.* Descriptions unsatisfiable with respect to the ontology are filtered out.

2. *Ontological pattern matching.* Checks whether the DL descriptions match a given ontological pattern. In the marketplace scenario it is verified if the concept expressions keep the good/description structure via a subsumption check with a DL expression representing such structure.
3. *Subsumption relationship.* Given D_1, D_2 two different translations of the same advertisement, if $D_1 \sqsubseteq D_2$, the filter removes the more general description D_2, which is less specific than D_1.
4. After the application of the previous filters, there could yet be more than one DL expression $D_1, D_2, ..., D_n$ associated to the sentence. In order both to avoid the same sentence being described with logical formulas inconsistent with each other and to put together all the information extracted from the NL sentence, we model the final translation as the conjunction of all the translations remaining after previous stages. In this way, if two resulting descriptions, D_i,D_j model information incompatible with each other, *i.e.*, $D_i \sqcap D_j \equiv \bot$, then an error message is returned, stating that the parser is not able to find a unique semantic model of the sentence. Furthermore, in this way we are able to catch all available information, even if it is not present in every candidate expression associated to the sentence.

4 System and Results

The NL parser presented here was designed with the aim of making the system as flexible and modular as possible. It is implemented in Java and all configurations, including grammars, are provided as XML files; a snapshot of the Graphical interface is in Fig. 4. The parser is is part of the MAMAS [4] framework, a semantic-based matchmaking service, which uses a largely modified version of the NeoClassic reasoner to provide both standard inference services (*e.g.*, subsumption and satisfiability) and novel non-standard services, in an $\mathcal{ALN}$DL, especially tailored for e-marketplaces. Given a supply/demand advertisement *potential ranking* [11] retrieves a sorted list of *satisfiable* matching advertisements, ranked accordng to their mismatch semantic distance from the query; *partial ranking* [11] retrieves a sorted list of *unsatisfiable* matching advertisements, ranked according to their dissimilarity semantic distance from the query (basically useful when nothing better exists); *abduce* [10] provides descriptions of what is missing in a description to completely fulfill the query, *i.e.*, it extends subsumption providing an explanation. To provide a flavor of the system behavior, in the following we report matchmaking results with respect to the marketplace descriptions shown in Table 1. Notice that in the table *demand0* is not consistent with the knowledge modeled in the ontology because of processors number specification [5]. Hence, the ranked list below is related only to *demand1* versus *supply1*, *supply2*, *supply3*.

[4] available at http://dee227.poliba.it:8080/MAMAS-devel/

[5] The ontology describes a desktop computer as a machine endowed with exactly 1 CPU (`Desktop_Computer` $\sqsubseteq$ ...(= 1 `hasCPU`) $\sqcap$ ...), then a notebook defined as a desktop computer (`Notebook` $\sqsubseteq$ `Desktop_Computer`...) cannot have two processors

demands	**NL sentence/DL translation**
*demand*0	– *Looking for a Pentium4 biprocessor notebook with 256 mb RAM.* – `Request Incoherent w.r.t. the Ontology`
*demand*1	– *Desktop computer with 30 Gb hard disk, lcd monitor included.* – `Desktop_Computer` $\sqcap \forall$`hdd.`$(\geq$ 30 `gb`$) \sqcap$ `hasComponentLCD_monitor`
supplies	**NL sentence/DL translation**
*supply*1	–*Offering Notebook with 40 Gb hard disk and 256 Mb ram.* – `Notebook` $\sqcap \forall$`RAM.`$(=$ 256 `mb`$) \sqcap \forall$`hdd.`$(=$ 40 `gb`$)$
*supply*2	– *Offering Desktop computer with 80 Gb hard disk and 512 mb ram equipped with cd reader.* – `Desktop_Computer` $\sqcap \forall$`RAM.`$(=$ 512 `mb`$) \sqcap \forall$`hdd.`$(=$ 80 `gb`$)$ $\sqcap \forall$`hasStorageDevice.CD_Reader`
*supply*3	– *Offering Server with Pentium4 processors.* – `Server` $\sqcap \forall$`hasCPU.Intel_Pentium4`

Table 1. Marketplace example

Translated Advertisements	**89**
Completely translated	73
Incomplete translations	16
Wrong translation	**9**
Inconsistent translation w.r.t. the ontology	**2**

Table 2. Test Results

Potential matches ranking list ([potential ranking] – [abduce] results):
supply1 vs. *demand1* [0] – [$\top$]
supply2 vs. *demand1* [1] – [$\forall$`hasComponent.LCD_monitor`]
Analyzing the above results, we see that *supply1* completely satisfies *demand1*, in fact the mismatch distance is 0, as there is a subsumption relation between *demand1* and *supply1*, as can be also argued by the $\top$ result for the related Concept Abduction Problem. With reference to *supply2*, in order to make it completely satisfy *demand1*, information on $\forall$`hasComponent.LCD_monitor` should be specified, then the distance computed w.r.t. the ontology is 1 (instead of 2, due to the axiom in the ontology stating that `Desktop_Computer` $\sqsubseteq$ ... $\sqcap$ $\forall$`hasComponent.Monitor`...).
Partial matches ranking list ([partial ranking] result): *supply3* vs. *demand1* [1]

To carry out a test of the parser performances, without any claim of completeness, we selected the domain of real estate advertisements. The domain knowledge was provided examining advertisements from several English newspapers and websites. The ontology built for this marketplace is composed by 146 concepts and 33 roles. The Lexicon is of 553 words, and Level 1 and Level 2 Grammars respectively have 79 and 58 rules. We randomly selected 100 advertisements (all different from those originally used during the domain definition) from various British websites and used them as test set. Results are summarized in Table 2.

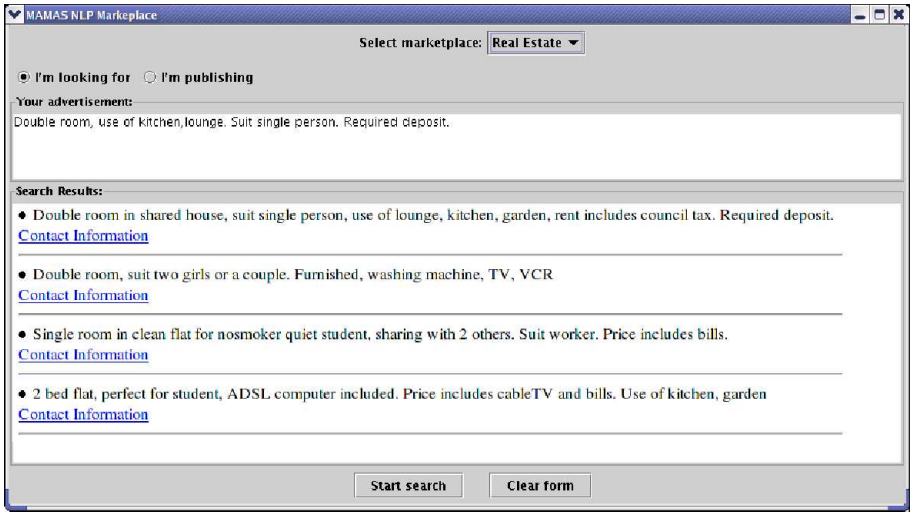

Fig. 4. Parser graphical user interface

5 Discussion and Conclusion

The Semantic Web intiative, which envisions ontology-based semantic markup both for interoperability between automated agents and to support human users in using semantic information, has provided a renovated interest towards NL based systems and approaches. Relevant recent works include *Aqualog* [8], which uses the GATE (http://gate.ac.uk) infrastructure and resources, extended by use of Jape grammars that add relations and question indicators to annotations returned by GATE. The input query in natural language is mapped to a triple-based data model, of the form <subject, predicate, object>. These then are further processed by a dedicated module to produce ontology-compliant queries. If multiple relations are possible candidates for interpreting the query, they revert to string matching is used to determine the most likely candidate, using the relation name, eventual aliases, or synonyms provided by lexical resources such as WordNet. Swift et al. [13] proposed a semi-automatic method for corpus annotation using a broad-coverage deep parser to generate syntactic structure, semantic representation and discourse information for task-oriented dialogs. The parser, like the one we propose, is based on a bottom-up algorithm and an augmented context-free grammar with hierarchical features, but generates a semantic representation that is a flat unscoped logical form with events and labeled semantic arguments. This method builds linguistically annotated corpora semi-automatically by generating syntactic, semantic and discourse information with the parser, but the best parse has to be selected by hand from a set of alternatives. Our system, instead, uses a post-processing module that refers to an ontology and a reasoner to automatically select the final translated sentence. Semantic interpretation in our system is performed using a semantic gram-

mar, which allows to produce constituents with both syntactic and semantic meanings; a similar approach is used by Bos et al. [4]; they apply *Combinatory Categorial Grammar* (CCG) to generate semantic representations starting from CCG parser. The tool they use to build semantic representations is based on the lambda calculus and constructs first-order representations from CCG derivations. In this work we exploited use of knowledge domain, to model task ontologies and grammars, making them both highly re-usable. We are currently working on the introduction of a morphological analysis in conjunction with WordNet for lexicon modeling, and on an extension of the approach to more expressive DLs.

Acknowledgments
The authors acknowledge partial support of projects PON CNOSSO, PITAGORA, and MS3DI.

References

1. A.Lavelli, B.Magnini, and C.Strapparava. An approach to multilevel semantics for applied systems. In *Proc. ANLP'92*, pages 17–24, 1992.
2. James Allen. *Natural Language Understanding (2nd ed.)*. The Benjamin Cummings Publishing Company Inc., 1999.
3. F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2002.
4. J. Bos, S. Clark, and M. Steedman. Wide-coverage semantic representations from a ccg parser. In *Proc. COLING-04*, 2004.
5. R.J. Brachman and H.J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. AAAI-84*, pages 34–37, 1984.
6. J.A.Bateman. Upper modeling: Organizing knowledge for natural language processing. In *Proc. 5th Int. Workshop on Natural Language Generation*, pages 54–61, 1990.
7. K.Knight and S.Luk. Building a large knowledge base for machine translation. In *Proc. AAAI'94*, 1994.
8. V. Lopez and E.Motta. Ontology-driven question answering in aqualog. In *Proc. NLDB-04*, 2004.
9. T. Di Noia, E. Di Sciascio, and F.M. Donini. Extending Semantic-Based Matchmaking via Concept Abduction and Contraction. In *Proc. EKAW 2004*, pages 307–320. 2004.
10. T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Abductive matchmaking using description logics. In *Proc. IJCAI-03*, pages 337–342, 2003.
11. T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. A system for principled Matchmaking in an electronic marketplace. *International Journal of Electronic Commerce*, 8(4):9–37, 2004.
12. R.Brachman, R.Bobrow, P.Cohen, J.Klovstad, B.Webber, and W.Woods. Research in natural language understanding, annual report. Technical Report 4274, Bolt Beranek and Newman, 1979.
13. M. D. Swift, M. O. Dzikovska, J. R. Tetreault, and J. F. Allen. Semi-automatic syntactic and semantic corpus annotation with a deep parser. In *Proc. LREC 04*, 2004.