# Interactive Learning: an Approach for Building DL Ontologies from Natural Language and Reasoning

Ryan Ribeiro de Azevedo[1,2], Fred Freitas[2], Rodrigo Rocha[1,2], José Antônio Alves Menezes[1], and Cleyton Mário de Oliveira Rodrigues[2]

[1] Computer Science, UAG/Federal Rural University of Pernambuco, Garanhuns, Brazil
{ryan,rodrigo,jam,lfap}@uag.ufrpe.br
[2] Center of Informatics, Federal University of Pernambuco, Recife, Brazil
{rra2, fred, rgcr,cmor}@cin.ufpe.br

**Abstract.** In this paper, we present an approach based on Reasoning and Natural Language Processing for Ontology Learning, specifically over Description Logic (DL) knowledge bases constituted by a TBox with ALC expressivity, from interactions with users in controled natural language text. The viability of our approach is demonstrated through the generation of descriptions of complex axioms from concepts defined by users. We evaluated our approach in an experiment with entry interactions enriched with hierarchy axioms, disjunction, conjunction, negation, as well as existential and universal quantification to impose restriction of properties. The obtained results prove that our model is an effective solution for reasoning, knowledge representation and automatic construction of expressive Ontologies. Thereby, it assists professionals involved in processes for obtain, construct and model knowledge domain.

**Keywords:** Description Logic (DL), Ontology Learning, Natural Language Processing, Knowledge Representation and Reasoning.

## 1 Introduction and Motivation

One of the subfields of Ontology that has been standing out along the last decade is Ontology Engineering. Its main purpose is to create, represent and model knowledge domains, most of which are not trivial, such as Bioinformatics and e-Business, among others. However, as pointed out by [14], the task of Ontology Engineering still consumes a big amount of resources even when principles, processes and methodologies are applied. Besides financially expensive, the ontology design is also an arduous and onerous task [6]. Thus, new technologies, methods and tools for overtaking these technical and economic challenges are necessary. This way, the need for highly specialized manual efforts required can be minimized.

For this purpose, a research line that is gaining importance through the past two decades is the extraction of domain models from text written in natural language, using Natural Language Processing (NLP) techniques. The process of modeling a knowledge domain from text and the automated design of ontologies, through an analysis of a set of texts using NLP techniques, for example, is known as Ontology

Learning and was first proposed by [11]. Even so, as mentioned by [17], despite of the increasing interest and efforts taken towards the improvement of Ontology Learning methods based in NLP techniques [16][1][2][3], the notable potential of the techniques and representations available to the learning process of expressive ontologies and complex axioms has not yet been completely exploited. In fact, there are still gaps and unanswered questions that need viable and effective solutions. Among them, the following stand out [13] [15] [16]:

- The lack of combining the knowledge of specialists of an arbitrary domain with the competencies and experience of ontology engineers in a single effort: there are scarce specialized resources and demand for professionals. This obstacle reduces the use of semantic ontologies by users and specialists in general.
- There is a considerable amount of tools and *frameworks* of *Ontology Learning* that have been developed aiming the automatic or semi-automatic construction of ontologies based on structured, semi-structured or unstructured data. Nonetheless, although useful, the majority of these tools used in Ontology Learning are only capable of creating informal or unexpressive ontologies.
- Evaluating the consistency of ontologies automatically by reasoning: it is necessary that the automatically created ontologies be assessed by the time of their development, minimizing the amount of errors committed by the ones involved in the development phase and verify whether or not the ontology is contradictory and free of inconsistencies.
- Knowledge-Based Systems which carry out large-scale reasoning for automatic decision-making in complex domains require and depend on expressive ontologies. These allow computational systems to infer precise conclusions. Thus, it is necessary that the tools used in Ontology Learning add axioms and perform subsumption reasoning, regarding the expressive DL-based ontologies.

All the questions and issues above mentioned justify the approach hereby proposed. It is based in a translator, which consists in the utilization of a hybrid method that combines syntactic and semantic text analysis both in superficial and in-depth approaches of NLP. Our approach shows that a translator for creating ontologies that formalizes and codifies knowledge in OWL DL *ALC* [7] from sentences provided by users is a viable and effective solution to the process of automatic construction of expressive ontologies and complete axioms.

## 2 The Approach and Example

One of the goals of this work consists in demonstrating that a translator, through the processing of sentences in natural language provided by users, is capable of creating automatically and, according to the discourse, interpreted $\mathcal{ALC}$ [7] ontologies with minimal expressivity. An overview of the translator's architecture and function flow diagram are depicted in Figure 1 and described as follows. The architecture of our approach is composed by 4 modules: the **Syntactic Parsing Module (1)**, **Semantic Parsing Module (2)**, **OWL DL Axioms Module (3)** and the **Reasoning Module (4)**.

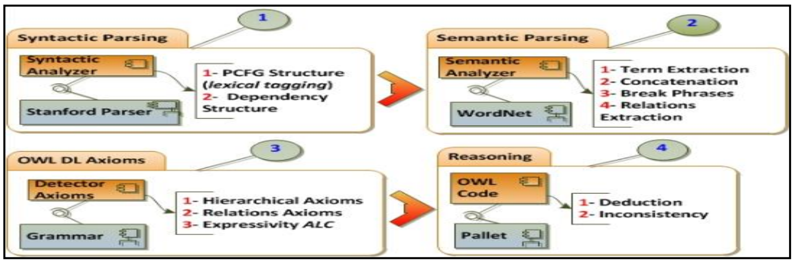


Fig. 1. Translator's architecture and function flow diagram

The activities executed in the respective modules and their functions are presented in the following sections.

## 2.1 Syntactic Parsing Module

The syntactic analysis of the sentences inserted by users takes place in the **Syntactic Parsing Module (1),** this module uses a Probabilistic Context-Free Grammars (PCFGs). For this purpose, we used the Stanford Parser 2.0.5[1]. Two activities are executed by this module, the lexical tagging and the dependence analysis (Using PCFG). The results obtained by this module are shown in Fig. 2 and 3**.** We used the sentence **(S1)**: "A self-propelled vehicle is a motor vehicle or road vehicle that does not operate on rails" to illustrate the results obtained by the translator's modules.

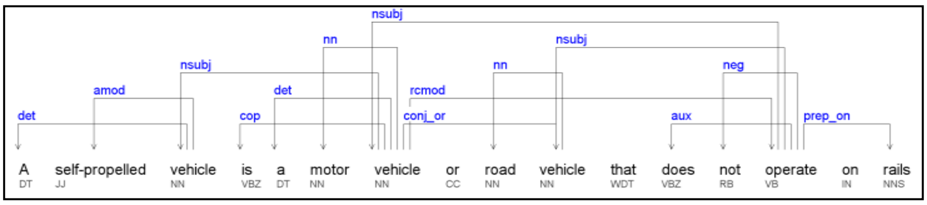


**Fig. 2.** Lexical tagging and dependence structure

Each word of the sentence (S1) above (Figure 2) is grammatically classified according to their lexical categories and the dependence between them is attributed. Syntagmatic categories are in red and, in black, the lexicon to which each category pertains: (**ROOT** (**S** (**NP** (DT A) (JJ self-propelled) (NN vehcle)) (**VP** (VBZ is) (NP (DT a) (NN motor) (NN vehicle)(CC or) (NN road) (NN vehicle))(**SBAR** (**WHNP** (WDT that)) (S (**VP** (VBZ does) (RB not) (**VP** (**VB** operate) (**PP** (IN on) (**NP** (NNS rails))

[1] http://nlp.stanford.edu/software/lex-parser.shtml

## 2.2 Semantic Parsing Module

The results of the activities carried out by the systems of the **Syntactic Parsing Module (1)** are used by the systems of the **Semantic Parsing Module (2)**, which carries out the activities shown in Figure 4 and are detailed as follows.

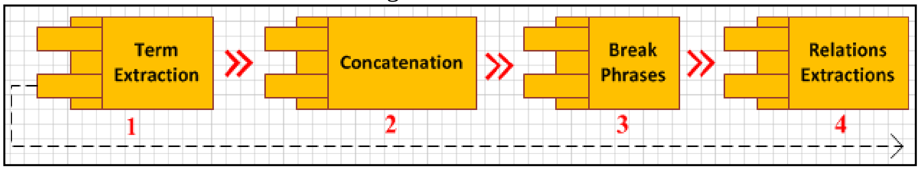


**Fig. 4.** Activities carried out in the Semantic Parsing Module

This module initiates its activities by assessing the entry sentence and the referred result of the syntactic analysis obtained in the previous module and then starts the extraction of terms (**Term Extraction**) that are fit to be concepts of the ontology (Activity (1)). Therefore, the terms extracted, which fit the concepts were: motor/NN, vehicle/NN, road/NN, vehicle/NN, self-propelled/JJ, vehicle/NN and rails/NNS.

The page must be prepared in two equal-width columns with 3/8 inch spacing between the columns. Text is to be single spaced. Left and right margins must be 1.25 inches, top margin 1 inch, and bottom margin 1.5 inches. The top margin of the first page only should be 1.5 inches instead of 1 inch. Use justified alignment with hyphenation. After the term extraction activity is done, the Activity (2), called **Concatenation** is enabled. This activity uses the results of the dependences between the terms (See Figures 2 e 3) and makes the junction of NPs composed by two or more nouns and/ or adjectives inside the analyzed sentence and which, in fact, are related. In the example sentence (*S*1), the concatenation results in the junction of the terms (self-propelled/JJ ↔ vehicle/NN), (motor/NN ↔ vehicle/NN) and (road/NN ↔ vehicle/NN) into an only term, because they are dependent of one another, resulting in just 3 terms: *self-propelled_vehicle*, *motor_vehicle* e *road_vehicle*, and no longer 7 terms, as in the initial phase of the **Term Extraction** activity.

In Activity (3), **Break Phrases,** every time terms or punctuation marks like comma (,), period (.), *and*, *or*, *that, who* or *which* (what we call sentence breakers) are found, the sentences are divided into subsentences and analyzed separately, the result for (S1) was: <u>self-propelled_vehicle is a motor_vehicle | or road_vehicle | that does not operate on rails</u> The last activity to take place in the **Semantic Parsing Module** is Activity (4), **Relations Extraction**. The relations between the terms are verified and validated through verbs found in the sentences and patterns observed in the translator's inner grammar. The verbs are separated and the terms dependent on verbs are extracted, resulting in: <u>self-propelled_vehicle **is a** motor_vehicle | self-propelled_vehicle **is a** road_vehicle | self-propelled_vehicle **operate** on rails</u>

This module detects the terms and the relations between them, both hierarchical and nonhierarchical. However, this module neither extracts disjunctions, conjunctions nor generates OWL code corresponding to the result obtained. The activity of this module is exclusively for detecting terms, their relations and validity. Some patterns/rules are used during the discovery and the learning of the **Semantic Parsing**

**Module (2).** The patterns/rules used for learning hierarchical axioms are showed in Table 1.

**Table 1.** Patterns/Rules for transforming the hierarchical axioms of terms

| **Construction Patterns of hierarchical axioms of terms** |
|---|
| (1) $SN_1$ (***is a* \| *is an* \| *is***) SN |
| (2) $SN_1$ (***are a* \| *are an* \| *are***) SN |
| (3) $SN_1$ ***and*** $SN_2$ ***and*** $SN_3$ ***and*** SNn (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN |
| (4) $SN_1$**,** $SN_2$ **,** $SN_3$, $SN_n$ (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN |
| (5) $SN_1$**,** $SN_2$ ***and*** $SN_3$ (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN |
| (6) $SN_1$**,** $SN_2$ ***and*** $SN_{3,}$ $SN_4$ (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN |
| (7) $SN_1$ (***is a* \| *is an* \| *is***) SN and/that/who/which (**is a \| is an \| is**) SN and (**is a \| is an \| is**) SN… |
| (8) $SN_1$ SNn (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN and SNn (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) $SN_1$ and $SN_n$… |
| (9) $SN_1$ ***and*** $SN_2$ ***and*** $SN_3$ ***and*** SNn (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN and (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) $SN_1$ and $SN_n$… |
| (10) $SN_1$**,** $SN_2$ ***and*** $SN_{3,}$ $SN_n$ (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN |
| (11) $SN_1$ (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) $SN_2$ ***or/and*** $SN_3$ |

The generated result for the above patters are the same, *i.e.*, $\{SN_1, SN_2, SN_{3,} SN_n\} \sqsubseteq SN$. Notice that the **and** connector for the mentioned patterns play the role of connector of concepts, establishing among them a dependency relation. The patterns above also are recognized as axioms OR, AND, and NOT and may be represented in the following way (See Table 2):

**Table 2.** Patterns/Rules for transforming the hierarchical axioms of relations using Inserction (and), Conjuction (or) and Nagation (not)

| **Construction Patterns using verbs, intersection (and), conjuction (or) and negation (not)** |
|---|
| (1) $SN_1$ (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN **o*r/and*** SN (***That/Who/Which***) (***has* not**) SN |
| (2) $SN_1$ (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN ***or/and*** SN (***That/Who/Which***) (***has***) SN |
| (3) $SN_1$ (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN ***or/and*** SN (***That/Who/Which***) (***Verb***) SN |
| (4) $SN_1$ (Verbo) SN (***or/and***) SN |
| (5) $SN_1$ (Verb) SN (***or/and***) (Verb) SN |
| (6) $SN_1$ ***has*** $SN_n$ ***and*** $SN_n$ |
| (7) $SN_1$ (***has not***) $SN_n$ |
| (8) $SN_1$ (***does not* \| *doesn't* \| *is not* \| *isn't***) $SN_n$ |
| (9) $SN_1$ (***does not* \| *doesn't* \| *is not* \| *isn't***) (Verb) $SN_n$ |
| (10) $SN_1$ ***has*** $SN_n$ ***and has not*** $SN_n$ |

| | |
|---|---|
| (11) | $SN_1$ (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN (***That/Who/Which***) (Verbo) (***or***) (Verbo) SN |
| (12) | $SN_1$ (***are a* \| *are an* \| *are* \| *is a* \| *is an* \| *is***) SN ***or/and*** SN (***That/Who/Which***) (Verbo) (***or/and***) (Verbo) SN |
| (13) | $SN_1$ (Verbo) SN (***or/and***) (Verbo) SN (***or/and***) (Verbo) SN… |

One should notice that new construction patterns of hierarchical axioms and relations may be inserted to the internal grammar of the described approach by human intervention. All the patterns presented in Table 2 composes disjunction (⊔), conjunction (⊓) and negation (¬) rules beyond the axioms with $\forall r.C$: universal restriction and $\exists r.C$: existential restriction. In the next section, the operation of the **OWL DL Axioms Module** is described and the obtained results using its processing are showed.

### 2.3 OWL DL Axioms Module

The function of the **OWL DL Axioms Module** is to symbolically find/learn axioms that prevent ambiguous interpretations and limit the possible interpretations of the discourse, enabling systems to verify and disregard inconsistent data. The process of discovering the axioms is the hardest part of the process of creating ontologies. Here, the axioms discovered correspond to DL $\mathcal{ALC}$ expressivity. The module recognizes coordinating conjunctions (*OR* and *AND*), labeled CC, indicating the union (disjunction) and intersection (conjunction) respectively for concepts and/or properties, recognizes linking verbs followed by negations, like *does not* (or *doesn't*), *has not* (or *hasn't*), and *is not* (or *isn't*) for negation axioms (¬), besides generating universal quantifiers (∀) and existential quantifiers (∃). In this module, we used Protégé-OWL API 3.5[2] and OWL API[3]. It also recognizes *is* and *are* as taxonomic relations (⊑ - hierarchical). The transformations occur in four steps and make use of the results obtained by the previous modules:

**Step (1):** construction of taxonomic/hierarchical relations. For our example, the results of Step (1) were: self-propelled_vehicle **is a** motor_vehicle → *self-propelled_vehicle ⊑ motor_vehicle* | self-propelled_vehicle **is a** road_vehicle → *self-propelled_vehicle ⊑ road_vehicle*. By subsumption reasoning the implicit hierarchical relations (motor-vehicle ⊑ vehicle) and (road-vehicle ⊑ vehicle) are discovered and created automatically.

**Step (2):** construction of nonhierarchical relations. For our example, the results of Step (2) were: *self-propelled_vehicle* ***operate*** *on rails* → *self-propelled_vehicle ⊑ ∃operate_on.rails*. **Step (3):** verification of conjunctions and disjunctions. The conjunctions OR and AND are verified and analyzed. They can be associated with concepts and/or properties. The results of Step (3) were:: *A self-propelled_vehicle is a motor_vehicle* ***or*** *road_vehicle* → *self-propelled_vehicle ⊑ (motor_vehicle ⊔ road_vehicle)*.

---

[2] http://protegewiki.stanford.edu/wiki/Main_Page
[3] http://owlapi.sourceforge.net/

**Step (4):** detection of negations. The fourth analysis detects the negations, its dependences and classifies the sentence to apply the patterns. Two negations are possible: negations and disjunctions of concepts and negations of properties. For (S1), the following result was obtained: *self-propelled_vehicle that does not operate on rails* → *self-propelled_vehicle* $\sqcap \neg\exists$*operate_on.rails.*

The **final result**, after the integration of the partial results obtained by the three modules, for (S1) in OWL 2 code, was: (S1): *"A self-propelled vehicle is a motor vehicle or road vehicle that does not operate on rails"* → *self-propelled_vehicle* $\sqsubseteq$ *(motor-vehicle* $\sqcup$ *road-vehicle)* $\sqcap$ $\neg\exists$*operate_on.rails* | *(motor_vehicle* $\sqcap$ *road_vehicle)* $\sqsubseteq$ *vehicle.*

## 2.4 Reasoning Module

In knowledge-based systems, the notion of reasoning is associated with the process of reaching conclusions. Besides the knowledge itself, an arbitrary KB has explicit axioms, useful to turn implicit knowledge evident, through formal logic reasoning. Two important aspects of logic-based reasoning with Semantic Web Ontologies, fully addressed by our approach, are: (i) verification of an ontology's specification, giver some user propositions ($\varphi$) ,the approach finds out inconsistencies and contradictions in the OWL DL automatically generated code (e.g.: (e.g.: $\varphi_{1,}\ \varphi_{2,}\ ...,\ \varphi_{n} \rightarrow C \sqsubseteq \bot$), and (ii) the deduction of new axioms, based on the propositions already informed (e.g.: $\varphi_{1,}\ \varphi_{2,}...,\varphi_{n} \rightarrow \varphi_{n}$). This statement can be verified by observing the results and discussion, at the following section. The activity of reasoning performed by our approach happens in parallel to the activity processing of sentences in natural language to OWL DL $\mathcal{ALC}$. In our reasoning learning algorithm (See Figure 5).

```
Algorithm 1: Reasoning Learning Algorithm.
Input : S : a set with the given sentences (φi)
Output: RD : a set with (DL ALC Representation) Decisions
Begin
        Boolean hasUndefinedClasses;
        Let RD ← { }
        Let Ontology ← null
        // run Pellet inference in background
        Call PelletInferenceProcedure();
        Foreach sentence φx ∈ S do
              Let isUniversalResctriction, hasUndefinedClasses ← false;
              // a procedure converts to OWL DL and adds the concepts
              Call ConvertToOWLDLCode(Ontology, φx , ϑ);
              // after the concepts, the generic instances are added
              Call AssertIndividuals(Ontology, ϑ);
              // the user fix the property restriction (only, some, not or a combination among these)
              Call FixPropertyRestriction(ϑ);
              // check for undefined new concepts
              hasUndefinedClasses = VerifyUndefinedClasses(ϑ);
              If hasUndefinedClasses == true then
                    Call AddRemainingConcepts(Ontology);
              End
              // investigates inconsistencies and appends the result into RD set
              Call SubsumptionReasoningProcedure(Ontology, RD);
        End
End
```

Figure 5: Reasoning Learning Algorithm.

Implemented in Java, we make use of the De Morgan's laws and the internal algorithms of the Pellet inference engine like a auxiliary algorithms, performs activities of subsumption, deduction and verification of the contradictory facts propositions by the users.

# 3 Results and Discussions

## 3.1 Data Set

In order to validate the translator, sentences from various knowledge domains were used. The data set utilized in the experiments contains a total of 120 sentences and, in all of them, there were negative axioms and besides these, conjunctive or disjunctive axioms and/or two and/or three types of axioms in the same sentence, as well as axioms with definition of terms hierarchy. We opted for sentences found in *Wikipedia* glossaries because they offered a controlled language without syntactic and semantic errors, besides providing a great opportunity for automatic learning. Some examples of sentences used in the experiments and the respective results generated by the translator, along with a discussion on these results are shown as follows.

## 3.2 Results and Discussion Part 1 – Generating OWL DL $\mathcal{ALC}$

**Processed Sentence (1):** juvenile is a young fish or animal that has not reached sexual maturity.
**Result**: → *juvenile* ⊑ (*young_fish* ⊔ *young_animal*) ⊓ ¬∃*hasReached.Sexual-maturity* | → *young_fish* ⊑ *fish* | → *young_animal* ⊑ *animal*

**Discussion**: the result of the analysis of the sentence is different from the results of the processing performed by the LExO system [16]: Juvenile ≡ (young ⊓ (Fish ⊔ Animal) ⊓ ¬∃reached.(Sexual ⊓ Maturity). The compared system (LExO) classifies *young*, *fish* and *animal* as distinct terms, however, by the interpretation in natural language of the sentence, the word *young* is an adjective of the *fish* concept, thus, our approach classifies and represents '*young fish'* as a composite noun, that is, composing a single concept (*young_fish*). The same occurs for s*exual maturity,* these two terms are classified as a single concept in the same way as the classification of the previous concept (*young_fish*). We can also observe the creation of two axioms, one of union of concepts and one of negation of property. By subsumption reasoning the hierarchical axioms: young_fish ⊑ fish and young_animal ⊑ animal were discovered and automatic created by our approach.**).**

**Processed Sentence (2):** vector is an organism which carries or transmits a pathogen.
**Result**: → *vector* ⊑ (*organism* ⊓ (∃*carries.Pathogen* ⊔ ∃*transmits.Pathogen*))
**Discussion**: the result obtained in (2) was also compared with results generated by LExO [16]: Vector ≡ (Organism ⊓ (*carries* ⊔ ∃*transmit.Pathogen*)). The verb *to carry* was not correctly classified as an existential restriction when analyzed by LExO, whereas in our approach, the sentence was coherently classified, the existential

quantifier was created and the disjunction of the relations created was performed, where carriesPathogen and transmitsPathogen are disjoint (⊔), which evidences the accurate interpretation of the sentence in natural language.

**Processed Sentence (3):** fish is an aquatic animal and isn't a mammal.
**Result:** → fish ⊑ (aquatic_animal ⊓ ¬mammal ) | → *aquaic_animal* ⊑ *animal*
**Discussion**: the expected result was generated, the *fish* concept is subclass of *aquatic animal* concept and disjoint of *mammal* concept. Furthermore, the hierarchical relation *aquatic_animal* ⊑ *animal* was discovered by subsunction reasoning. This complement alone is useless, however when combined with an intersection, it can be really useful in reasoning tasks. Figure 8 shows part of the resultant OWL DL axiom constructed automatically for sentence **(3).**

**Processed Sentence (4):** whale is an aquatic animal and marine mammal.
**Result:** → whale ⊑ (aquatic_animal ⊓ marine_mammal) | → *aquatic_animal* ⊑ *animal* | → marine_mammal ⊑ mamal | → marine_mammal ⊑ mammal
**Discussion**: in this sentence, the concept *whale* forms a hierarchy with the other two concepts (*aquatic animal* and *aquatic mammal*). Besides, it generates an intersection of both terms, meaning that individuals pertaining to the concept *whale* pertain to the set of individuals of both concepts at the same time. By subsumption reasoning the hierarchical axioms: aquatic_animal ⊑ animal and marine_mammal ⊑ mammal were discovered and created automatically by our approach. Futhermore, our approach discovered, by deduction, the following axioms: whale ⊑ animal and whale ⊑ mammal, in this case, distinctly of sentence in (2), and as expected, the OWL 2 code associated to this deduction is not generated. The user is only informed that the translator was able to deduce according to the sentence processing and it is showed that it performs a reasoning although there are limitations.

### 3.3 Results and Discussion Part 2 – Reasoning over the DL $\mathcal{ALC}$ Generated

The following examples, shows the reasoning using concepts and definitions that have a knowledge base initially empty $\mathcal{K} = \emptyset$. The goal of this phase was verify the ability of our approach in performing subsumption reasoning, and also in detecting inconsistencies on the ontologies created from real time dialogues between users and the system. Example 1 shows the propositions ($\varphi$) given by users in natural language, the generated representation in DL $\mathcal{ALC}$ and the obtained results after the reasoning performed:

**Example 1** – **Example ontology for deduction**

- ***User***: cow eats grass → cow ⊑ ∃eats.grass
- ***Approach***: eats only grass or another thing?
- ***User***: only grass
- ***Approach***: Ok… What is a grass? → cow ⊑ ∀eats.grass | ~~cow ⊑ ∃eats.grass~~
- ***User***: grass is a kind of plant → grass ⊑ plant
- ***Approach***: Ok… Is this enough to model the concept grass?
- ***User***: yes
- ***Approach***: Ok… Tell me more about this domain we are modeling!

- ***User***: herbivore eats only plant
- ***Approach***: herbivore $\sqsubseteq$ ∀eats.plant
- ***Approach***: eats only plant or another thing?
- ***User***: only plant
- ***Approach***: (**Deduction**) Hummm, Is cow subclass of herbivore?
- ***User***: yes
- ***Approach***: cow $\sqsubseteq$ herbivore → The axiom $\propto_{\Delta}$ (*cow*, *herbivore*), for example, states that any *cow* is also a *herbivore* ($[\forall x : E1(x) \rightarrow E2(x)]$, $E1, E2 \in C \cup P$). This way, the knowledge base of our approached improved by the knowledge extracted in the dialogues with the users is: $\mathcal{K}$ = {cow $\sqsubseteq$ ∀eats.grass; herbivore $\sqsubseteq$ ∀eats.plant; cow $\sqsubseteq$ herbivore}.

**Example 2** – **Example of a contradictory (verification) ontology and ontology for deduction**

- ***User***: herbivore eats leaves
- ***Approach***: eats only leaves or another thing?
- ***Approach***: herbivore $\sqsubseteq$ ∃eats. leaves
- ***User***: another thing
- ***Approach***: Ok… What is a leaves?
- ***User***: herbivore eats meat → After adding the proposition ($\varphi$: *herbivore eats meat*), the approach would check for inconsistencies, however the knowledge in $\mathcal{K}$, the only available, is not enough for detecting a possible inconsistency. The proposed approach cannot learn that herbivore does not eat meat and derivatives, and that meat is not a plant or a type of plant. The dialogue continues as follows:
- ***Approach***: eats only meat or another thing?
- ***User***: another thing
- ***Approach***: herbivore $\sqsubseteq$ ∃eats.meat
- ***Approach***: Tell me more about this domain we are modeling! What is a meat? → herbivore $\sqsubseteq$ ∃eats.meat
- ***User***: meat is kind of food and is not a plant.→ meat $\sqsubseteq$ food $\sqcap$ ($\sqsubseteq$ ¬plant)
- ***Approach***: Ok… Is this enough to model the concept meat?
- ***User***: yes
- ***Approach***: (**Deduction and subsumption reasoning**) → grass $\sqsubseteq$ ¬ meat → The axiom grass $\sqsubseteq$ ¬ meat, for example, states the concepts for *grass* and *meat* to be disjoint ($[\forall x : C1(x) \wedge C2(x) \rightarrow \perp]$, $C1, C2 \in C$).
- ***Approach***: (**Inconsistency**) → cow $\sqsubseteq \perp$ and herbivore $\sqsubseteq \perp$ → cow and herbivore are inconsistent. The explanation given to the users for inconsistency's checking is: if cow $\sqsubseteq$ herbivore, herbivore $\sqsubseteq$ ∀eats.plant, herbivore $\sqsubseteq$ ∃eats.meat and $\propto_{\otimes}$ (*plant*, *meat*) then we have an inconsistency in the *cow* concept. If herbivore $\sqsubseteq$ ∀eats.plant, herbivore $\sqsubseteq$ ∃eats.meat and meat $\sqsubseteq$ ¬plant then we also have an inconsistency in *herbivore* concept. If the user modifies the axiom herbivore $\sqsubseteq$ ∃eats.meat, once we know that *herbivore* do not eat meat, $\mathcal{K}$ may be free of inconsistencies. The dialogue continues as follows:
- ***User***: herbivores don't eat meat → herbivore $\sqsubseteq$ ¬∀eats.meat. By inserting this axiom, $\mathcal{K}$ became free of inconsistencies and it assumed the following structure: $\mathcal{K}$

= {cow ⊑ ∀eats.grass; herbivore ⊑ ∀eats.plant; cow ⊑ herbivore; meat ⊑ food ⊓ (⊑ ¬plant); grass ⊑ ¬ meat; herbivore ⊑ ¬∀eats.meat}.

**Example 3** – **Example of a contradictory (verification) ontology and ontology for deduction.**

The following example shows a long dialog containing contradictions, inconsistencies and verifications.

- ***User***: The aquatic animals live in water
- ***Approach***: live in only water or another thing?
- ***User***: only water
- ***Approach***: aquatic_animals ⊑ ∀live_in.water
- ***Approach***: (**Deduction and subsumption reasoning**): aquatic_animals ⊑ animals
- ***Approach***: What is a water?
- ***User***: marine mammals are aquatic animals
- ***Approach***: marine_mammals ⊑ aquatic_animals
- ***Approach***: (**Deduction and by subsumption reasoning**): marine_mammals ⊑ mammals | aquatic_animals ⊑ animals | marine_mammals ⊑ animals
- ***Approach***: Tell me more about this domain we are modeling! Whats is animals?
- ***User:*** aquatic animals are not mammals → aquatic_animals ⊑ ¬mammals
- ***Approach***: (**Inconsistency**): marine_mammals is inconsistent → marine_mammals ⊑ ⊥. → The approach highlights the inconsistency by exposing: if aquatic_animals ⊑ ¬mammals and marine_mammals ⊑ mammals then we have an inconsistency in marine_mammals concept, because marine_mammals cannot be subclass of aquatic_animals.
- ***Approach***: Tell me more about this domain we are modeling!
- ***User***: marine mammals aren't aquatic animals → marine_mammals ⊑ ¬aquatic_animals. → After the above iteration (marine_mammals ⊑ ¬aquatic_animals) $\mathcal{K}$, became free of inconsistencies. The dialogue continue as the following:
- ***User***: cetacean are marine mammals who live only in the sea
- ***Approach***: only in the sea?
- ***User***: yes
- ***Approach***: cetacean ⊑ ∀live_in. sea
- ***Approach***: (**Deduction and by subsumption reasoning**) cetacean ⊑ ¬ aquatic_animals
- ***Approach***: (**Deduction and by subsumption reasoning**) cetacean ⊑ marine_mammals → The approach discovers an implicit knowledge in the dialogue domain.
- ***Approach***: What is a sea?
- ***User***: sea is a water → sea ⊑ water
- ***Approach***: Ok… Is this enough to model the concept sea?
- ***User***: no
- ***Approach***: please provide a better definition!
- ***User***: sea is a ocean → sea ⊑ ocean
- ***Approach***: Ok, please continue.
- ***User***: shark live only in the sea

- ***Approach***: only in the sea?
- ***User***: yes
- ***Approach***: Ok, please continue → shark ⊑ ∀live.sea
- ***User***: sharks are aquatic animals
- ***Approach***: shark ⊑ aquatic_animals.
- ***Approach***: (**Deduction and subsumption reasoning**) cetacean ⊑ ¬shark | mammals ⊑ ¬shark | marine_mammals ⊑ ¬shark | shark ⊑ animal | → For every discover the approach shows an explanation to the user. Due space limitations in this work, we omitted the explanations created by the approach for the generated results in this interaction.

Now, $\mathcal{K}$ assumed the following structure: $\mathcal{K}$ = {cow ⊑ ∀eats.grass; herbivore ⊑ ∀eats.plant; cow ⊑ herbivore; meat ⊑ food ⊓ (⊑ ¬plant); grass ⊑ ¬ meat; herbivore ⊑ ¬∀eats.meat; aquatic_animals ⊑ ∀live_in.water; marine_mammals ⊑aquatic_animals; marine_mammals ⊑ mammals; aquatic_animals ⊑ animals; marine_mammals ⊑ animals; marine_mammals ⊑ ¬aquatic_animals; cetacean ⊑ ∀live_in. sea; cetacean ⊑ ¬ aquatic_animals; cetacean ⊑ mammals; sea ⊑ water; sea ⊑ ocean; shark ⊑ ∀live.sea; shark ⊑ aquatic_animals; cetacean ⊑ ¬shark; mammals ⊑ ¬shark; marine_mammals ⊑ ¬shark; shark ⊑ animal}. Summarizing the obtained results in the dialogue iterations (examples (1), (2) and (3)), we obtained the following data: 64 iterations, among which 27 iterations from users and 37 from the approach. In the 63 iterations, the approach was in doubt only time ((**Deduction →** "Hummm, Is cow subclass of herbivore?"). 2 inconsistencies was verified in $\mathcal{K}$, and it deducted 12 axioms by deduction and subsumption reasoning, among each 6 hierarchical axioms → $\propto_{\Delta}$ ($E_1$, $E_2$), and 6 disjoint axioms → $\propto_{\otimes}$ ($C_1$, $C_2$).

### 3.4 Validation

The objective of planning and performing the experiments was to verify the model constructed and test it in order to answer the following research questions: is the translator capable of constructing minimal-expressivity $\mathcal{ALC}$ ontologies and represent knowledge? And, is the translator capable of identifying rules and axioms inherent to $\mathcal{ALC}$ expressivity based on the texts produced from dialogues with users? In order to answer these questions, the following hypothesis was formulated to assess the translator's performance during the experiment: $\boldsymbol{H_{a,1}}$: The approach constructs the expressive ontology correctly in more than half of the cases; $\boldsymbol{H_{0,1}}$: The approach does not construct the expressive ontology in more than half of the cases. Analyzing all the 120 sentences, we obtained the following results: in 75% of the sentences analyzed (90 sentences), the translator detected and created coherently the axioms, whereas in 30 sentences (25%, also including the ones the translator could not possibly solve in any way). The translator did not detect the axioms coherently and committed errors; however, in 24 out of those 30 sentences, the translator created axioms and made possible the recreation of the ontologies through the proceeding of inserting new definitions. Figure 10 and Table 3 shows the results of three interactions.

**Table 3**: Rate – Hits and Errors

| | | |
|---|---|---|
| **1st - Round** | | |
| | Rate Percentage | Number of Sentences |
| Total | 100% | 120 |
| Hit | 75% | **90** |
| Error | 25% | **30** |
| **2nd - Round** | | |
| | Rate Percentage | Number of Sentences |
| Total | 100% | **30** |
| Hit | 80% | **24** |
| Error | 20% | 6 |
| **Final Total** | | |
| | Rate Percentage | Number of Sentences |
| Total | 100% | 120 |
| Hit | 95% | **114** |
| Error | 5% | **6** |

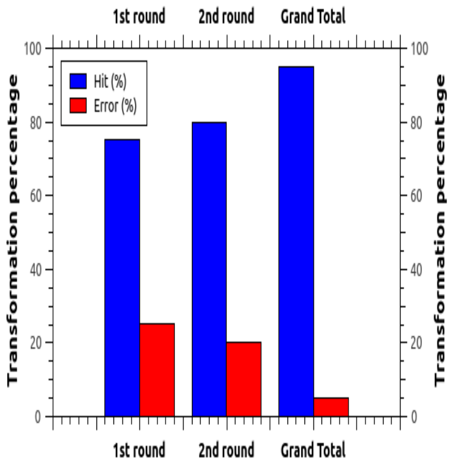


**Fig 10**: Transformation Percentage.

The right unilateral binomial test was applied and the following results obtained. Limiar → 50% | Level of significance α (5%) | p-value = 1.886e-08.

### 3.5 Sustention

As p-value is lower than the significance of the null hypothesis ($H_{0,1}$: The translator does not construct the ontology in more than half of the cases), it is not accepted, that is, the translator statistically constructs the ontology correctly in more than half of the cases. Using the same binomial test we confirm that this success ratio is statistically superior to 67% (0.67 < success ratio < 1; p-value=0.03636). Therefore, we conclude that the success ratio presented by our approach, 75%, is statistically higher than 67% and, in fact, significant.

## 4 Related Works

Our designed system was based in previous works proposed in the literature, despite some of them have distinct focus and detailing levels, all of them deal with the problem of automatically generate expressive ontologies and the assistance to ontology engineering activities. Among the researches that contributed to maturing and development of our proposed approach, we should highlight the following works. The LExO system [16][15] had published satisfactory results in state-of-the-art, it is also carries out a comparative analysis for other systems which propose automatic generators of expressive ontologies from natural language (NL) texts. The LExO[4] does not use the Stanford Parser at syntactic analysis phase, but the MINIPAR [12] does. Some published works pointed a slighted advantage to Stanford Parser in relation to others. This way, we would conclude that systems which use better parsers obtain more accurate results. Despite LExO transforms and represents accordingly the processed text in OWL DL code, it does not performs subsumption reasoning and does not discovers implicit relations in the analyzed propositions. Another excellent system related to our approach is the ACE[5] (Attemp to Controlled English) [8][9][10]. It transforms the given definitions in OWL DL code. However, it uses a specific syntax for its input propositions, then users unfamiliar with this syntax in NL will face problems in using it. We highlight that ACE represents accordingly in OWL DL the propositions given by the user, however it do not present subsumption reasoning and it fails checking inconsistencies in the classes defined and transformed in OWL DL.

The ACE works with ABox and TBox which may cause problems in DL representation, once it models individuals as concepts and concepts as individuals. The system does not perform deduction and subsumption reasoning. Finally, we present the TextOntoEx [4] system. The authors point that the system discovers relations between terms, however it does not generate OWL DL code, and seems to create only RDF codes. Despite it has been validated, fragments of code automatically generated are not shown and PLN techniques used are not presented. Furthermore, the system does not perform subsumption reasoning, does not discover other relations, does not deduce from a given set of propositions and does not check for inconsistencies. Our approach uses the Stanford Parser, performs subsumption reasoning and checks inconsistencies in the created classes during the development. This way it is an advantageous system in relation to others previously mentioned before. It processes texts in NL without properly syntax; however it requires a formal language in the conception of propositions given to the system. We only work with Tbox to eliminate confusions between individuals and concepts in definition representations. Nevertheless, when a concept is classified as an individual, the user is alerted to fix it.

## 5 Conclusions and Future Works

In this paper, we describe an approach to automatic development of expressive ontologies from definitions provided by users. The results obtained through the experiments suggest the need for automatic creation of expressive axioms, sufficient to creating ontologies with $\mathcal{ALC}$ expressivity, besides the success in the identification of rules and axioms pertaining to $\mathcal{ALC}$ expressivity. We also conclude that our approach can aid both experienced ontology engineers and developers and inexperienced users just starting to create ontologies. As future works, we include the integration of our approach with other existing approaches in the literature, the creation of a module for

[4] https://code.google.com/p/lexo/
[5] http://attempto.ifi.uzh.ch/site/description/

automatic inclusion of unprecedented patterns in the translator and one module for automatic insertion of individuals for terms of ontologies created by the translator.

## References


1. Buitelaar, P and Cimiano, P.: Ontology learning and population: Bridging the gap between text and knowledge. In: Frontiers in Artificial Intelligence and Applications Series (Vol. 167). IOS Press, (2008).
2. Buitelaar, P., Cimiano, P., & Magnini, B.(Eds.), Ontology learning from text: Methods, applications and evaluation , pp. 3–12. IOS Press,(2005).
3. Cimiano, P and Völker, J.: Text2Onto. NLDB, 227–238. (2005).
4. Dahab,M. Y., Hassan, H. A., Rafea, A .TextOntoEx: Automatic Ontology Construction from Natural English Text. In: Expert Systems with Applications. Volume 34, Issue 2, Pages 1474–1480. (2008).
5. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL Reasoner, J. Web Sem. 5 (2) 51–53, (2007).
6. Gómez-Pérez, A, Fernández-López, M and Corcho, O.: Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. In: Advanced information and knowledge processing, Springer, London, (2004).
7. Horrocks, I. et al.: OWL: a Description-Logic-Based Ontology Language for the Semantic Web. In: The Description Logic Handbook: Theory, Implementation and Applications. P458-486. Cambridge University Press. 2ed. (2007).
8. Kuhn, Tobias. A Survey and Classification of Controlled Natural Languages. In: Computational Linguistics - Association For Computational Linguistics. (2014).
9. Kuhn, Tobias. A Principled Approach to Grammars for Controlled Natural Languages and Predictive Editors. In Journal of Logic, Language and Information. 22(1). (2013).
10. Kuhn, Tobias.: Controlled English for knowledge representation. Ph.D. thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich. (2010).
11. Madche, A. and Staab, S.: Ontology learning for the semantic web. In: IEEE Intelligent Systems, 16(2):72-79, (2001).
12. Lin, Dekang Dependency-based evaluation of MINIPAR. In: Proceedings of the workshop on the Evalution of Parsing Systems at the 1st International Conference on Language Resources and Evaluation (LREC). (1998).
13. Pease, A.: Ontology: A Practical Guide. Published by Articulate Software Press. USA. 2011.
14. Simperl, E and VTempich, C. Exploring the Economical Aspects of Ontology Engineering synthetic. In: Handbook on Ontologies, International Handbooks on Information Systems. Springer, Berlin, Germany, pp. 337–358, (2009).
15. Völker, J.: Learning Expressive Ontologies. No. 002. In: Studies on the Semantic Web, AKA Verlag / IOS Press, ISBN: 978-3-89838-621-0. (2009).
16. Völker, J, Hitzler, P., Cimiano, P. Acquisition of OWL DL Axioms from Lexical Resources. In Enrico Franconi, Michael Kifer, Wolfgang May, Proceedings of the 4th European Semantic Web Conference (ESWC'07), volume 4519 of Lecture Notes in Computer Science, pp. 670-685. Springer, June 2007.
17. Zouaq, A. : An Overview of Shallow and Deep Natural Language Processing for Ontology Learning. Chapter 2. Pg 16-37. In: Ontology Learning and Knowledge Discovery Using the Web: Challenges and Recent Advances. IGI Global. EUA . ISBN 978-1-60960-625-1 (hardcover). (2011).