Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Fakultät für Informatik | Facoltà di Scienze e Tecnologie informatiche | Faculty of Computer Science

TECHNISCHE UNIVERSITÄT DRESDEN

EMCL Master Thesis

Reasoning with Text Annotations

by

Sudeep Ghimire

Supervisor : **Prof. Enrico Franconi**

Co-Supervisors: **Dr. Yue Ma**
Prof. François Lévy
Prof. Steffen Hölldobler

April, 2011

Acknowledgments

First, I would like to thank my supervisor Prof. Enrico Franconi at the KRDB Research Center for his kind supervision of this thesis. Without his support and advices, this research would have never been possible. My sincerer gratitude to my co-supervisors Yue Ma and Francois Levy, LIPN, Universite Paris XIII for great supervision, guidance, valuable advices and an opportunity to work at a well equipped research center.

I would make a special mention of the EMCL consortium for the two year Erasmus Mundus scholarship not only for an opportunity to study under an advance education system but also for many good things that I learnt about European research methodologies, cultures and lifestyles.

I want to thank all of my friends in Lisbon and Bolzano, who have made my these two years as one of the most beautiful periods in my life from all perspectives of academics, diverse culture exchanges and warm social integration. I wish to send my special thanks as well to my close friends round the world for being there whenever I was in need.

Last but not least, my deepest gratitude goes to my beloved family, my mother, father and brother for always encouraging me, taking care of me and understanding me.

Dedicated to my Parents

Contents

List of Figures

List of Tables

Abstract

With the emerging need for automation of business processes and the advent of semantic web it has become necessary that digital contents should be expressed not only in natural language, but also in a form that can be understood, interpreted and used by software agents, thus permitting them to find, share and integrate information more easily. Thus, Knowledge Representation and automated reasoning has been an interesting field of research for recent years, which is greatly supported by the availability of various logical languages underpinned with well defined syntax and semantics. Description logic (DL), which is also the foundation of standardized web ontology language OWL, has become a highly used de-facto standard for knowledge representation. DL reasoners can infer and detect logical contradictions in the ontologies specified in a certain web ontology language, such as OWL. Highly optimized DL reasoners like Pellet, Hermit, Fact++ can be used for reasoning over knowledge base represented using classical DL.

Annotated corpus with various annotations viz semantic, syntactic and pragmatic are important features for text-based applications, but are equally challenging and complex to be achieved, maintained and be used in an integrated way. So, a framework in which various linguistic annotations, produced by different ad hoc approaches and domain knowledge maintained by experts are integrated is useful for enrichment of the corpus. With standard representation formalism the annotated documents can be made suitable for automated reasoning. This formalism not only new annotations can be inferred from the existing ones but the same formalism and framework can be used for checking the correctness and completeness of annotations.

In this thesis the underlying knowledge representation formalism for representation of annotations is analyzed and solved by studying various logical languages including higher order logics, which accounts for cooperation of the different sorts of knowledge. The annotations are interpreted in various ways to handle the different needs to achieve varying useful results. At the same time different paradigms for reasoning with annotations are discussed along with necessary algorithms for dealing with such cases. An integrated framework is developed, the prototype version of which demonstrates the use of reasoning taking into account of both linguistic and semantic annotations, to detect conflicting annotations, new annotations and overall consistency of annotations. Thus, the research work that we have undertaken helped us to achieve an integrated knowledge representation and reasoning framework for dealing with annotated text corpus,supported by existing cutting age semantic web technologies. This thesis, also paves a path towards a metric to evaluate annotation i.e. detect logically inconsistent information, without depending on a golden standard, but by making use of the axioms expressed in ontologies.

Keywords: Description logic (DL), DL-reasoners, Natural Language Processing, Text Annotation, Knowledge Representation, Semantic Web Technology

Chapter 1

Introduction

"Logics is Everywhere"
EMCL students theme

With the emergence of standards for automation of business processes and the advent of semantic web now its necessary that digital contents should be expressed not only in natural language, but also in a form that can be understood, interpreted and used by software agents, thus permitting them to find, share and integrate information more easily. Thus, Knowledge Representation and automated reasoning has been an interesting field of research for recent years, which has been greatly supported by the availability of various logical languages underpinned with well defined syntax and semantics. Description logic (DL), which is also the foundation of standardized ontology language OWL, has become a highly used de-facto standard for knowledge representation. DL reasoners can infer and detect logical contradictions in the ontologies specified in a certain web ontology language, such as OWL. Highly optimized DL reasoners like Pellet [1], Hermit [2], Fact++ [3] can be used for reasoning over knowledge base represented using classical DL. Reasoning with Annotation of Text is an area that we are exploring on the application of various logics and existing standardized semantic web technologies.

Text corpus with various linguistic annotations, produced by different ad hoc approaches and domain knowledge maintained by experts play an important role in the advancement of Computational Linguistics (CL). At the same time, such annotations are equally challenging and complex to be achieved, maintained and checked for correctness and completeness. In this thesis, we have thus explored the possibility to integrate various annotations in a single framework and make use of existing technologies to extract additional annotations and to assert that the annotated corpus is logically consistent. This thesis is thus basically the study of how automated reasoning systems based on Description Logics (DLs) can be used for reasoning about text annotations. The ability of reasoning automatically on annotated documents makes it possible to provide computer aided support during text processing to automatically detect relevant properties, such as inconsistencies and redundancies.

In this chapter, we will basically explain the background of our work and basic logical foundation, related works, motivation behind the research work that we have undertaken and pave a foundation for understanding the rest of the chapters.

[1]http://clarkparsia.com/pellet/
[2]http://hermit-reasoner.com/
[3]http://owl.man.ac.uk/factplusplus/

1.1 Background

About a decade ago, Tim Berners-Lee, James Hendler and Ora Lassila published their seminal paper [3] describing the evolution of the current web from a human-processable environment to a machine-processable one. The basic idea incubated was to annotate resources and give them a machine-processable meaning. In the mean time many efforts have been placed to standardize both the language for representing the content and the production of annotations/metadata. One step closer towards machine understandable text processing is explained in [27] which is about annotations of documents with metadata also known as domain knowledge, which is in turn stored in the form of ontologies with semi-automatic approach.

In the last decades, logics have been applied to numerous areas of computer science, including knowledge representation, formal verification, database theory, distributed computing and, more recently, semantic web and ontologies. Various logical languages and allies have been proposed, studied and implemented to address the problem of automated reasoning. OWL has emerged as a standard for representing ontologies with expressivity to model most of the domain of consideration and is still evolving with new features in the latest version $OWL2$. At present we are equipped with highly optimized reasoners, which can be used for consistency checking, concept satisfiability, classification and realization over OWL based Knowledge.

Meanwhile, a trend grows toward platforms allowing to use different tools on the same text (see e.g. [13] for some descriptions, Stanford CoreNLP[4] suite for an implementation and the UIMA norm[5] for a framework) have been successfully implemented and tested. The interaction of different levels of analysis is well known by linguists, but the cooperation of different tools has been rarely studied in an integrated environment and are not equipped with formal description or reasoning mechanisms. Moreover, the outputs of each process are written in independent formats. Relying on a standard representation of different typed annotations attached to text fragments, we propose a logical framework that is able to cope with the results of different tools in a common formalism, to express logical relations linking them and, on this basis, to detect incoherences or to add inferred annotations.

The links between texts and its semantic markups are important for real usages (i.g. checking the correction, debugging, explaining, maintaining, repairing inconsistencies). Two direct benefits of structured semantic annotation are enhanced information retrieval and improved interoperability. Other applications include checking the conformance of procedures [11] and facilitating business rules writing and update pursued by the Ontorule project[6] which extends semantic data with typed rules according to a slightly extended SBVR classification [36].

1.2 Related Works

Knowledge acquisition or information extraction systems aim to acquire knowledge from texts, such as event extraction, named entity recognition, ontology construction. The acquired results are usually stored separately and used as semantic resources for other applications. Different from these systems, our work coincides with semantic annotation approaches (e.g. KIM[7] and *Semantic Turkey*[8]) with a focus on revealing the meaning of texts explicitly, by marking texts with suitable annotations from some ontological vocabularies. Naturally, to automate the semantic annotation process, I.E. techniques are beneficial and have been explored [46, 30]. Refer to [46] for a significant review of this field and interesting scenarios on using ontology for annotating texts. Note that most existing semantic annotation systems only consider a part of ontological elements: individuals and their conceptual categories [14, 26]. However, our application is to annotate domain specialized regulatory texts which differ from them in scale – each corpus is incomparably smaller – and in scope – the annotations have to cover a larger part of the content. In our case, annotation labels are from domain ontologies built by experts and thus extend far beyond general labels like *Person, Location, Organization*.

[4]`http://nlp.stanford.edu/software/corenlp.shtml`
[5]`http://uima.apache.org/`
[6]`http://ontorule-project.eu`
[7]`http://www.ontotext.com/kim/`
[8]`http://semanticturkey.uniroma2.it/`

Additionally, concept or role occurrences in these texts are much more frequent than that of individuals. This need for varied fine-grained annotations leads to knowledge modeling problems and to check, debug, and explain inconsistent annotations of texts with regarding to domain ontologies. Therefore, while many specific semantic annotation systems have been extensively studied, this work is to study the knowledge representation issue underlying ontology-based multi-layer annotation systems, and to provide a sound way to take advantage of different kinds of annotations. A problem highlighted in [46] and left for ongoing research is about keeping annotations consistent with evolving resources, particularly in combination with evolving ontologies. Our formalism provides a step in this direction by studying a way to detect conflicts in annotations with regard to domain ontologies, which is not supported by trivial representations of annotations by extending the usage of annotations of texts beyond semantic information retrieval by leveraging the relation among texts, annotations of texts, and domain ontologies.

Last, (logic-based) declarative I.E. systems [39, 42, 40, 45] have been developed to encode annotation process in logics, possibly with some predefined predicates for linguistic annotation patterns (e.g. *patternOcc, disambPrior, express*). Different from them, our framework aims to be part of an annotation platform on which users can uniformly analyze various linguistic and semantic annotations and their interactions.

1.3 Motivation

Modeling conceptual knowledge of a specific domain by ontologies and annotating texts with linguistic information (linguistic) are two critical problems for content management in the traditional Artificial Intelligence research. As seen from the previous section, there have been already much work on both tasks, but each of them usually misses the aspect of the other. In this thesis, we try to reduce the gap by studying a bridge between them, namely ontology-based semantic annotation of text. Basically the thesis addresses following two issues: 1. analyze and propose the knowledge representation formalism for handling different types of text annotations, 2. study and implement different reasoning paradigms to reason with text corpus annotated by different sources, making use of existing semantic web technology.

Let us try to make a thorough analyze of the problem statement by considering one sentence (it will be used in most of the places in the latter chapter, for giving examples) from regulatory texts provided by American Airlines, which is an eleven pages text on an air-travel loyalty program, taken as input corpus for our:

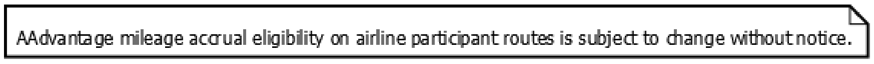
AAdvantage mileage accrual eligibility on airline participant routes is subject to change without notice.

Figure 1.1: Sample sentence

This sentence can be annotated with a number of concepts from domain ontology. For instance *credit* can be annotated as concept *Credit* from the ontology. At the same time, we can observe that *mileage* is the noun-form modifier for *credit*, so we can make linguistic annotation between these two with a relationship. Now, we know by intuitive linguistic studies that combination of head and modifier of the noun-form modifier relationship should mean the same as the head. Thus, we should be able to infer the semantic annotation to the combined word *mileage credit* as concept *Credit*.

With this example annotations, first questions that arise is how to link annotations to ontology, which we will call semantic annotations. We will address the problems related to this in section 3.2 and provide the logical formalism in section 4.1. Even though, annotations like noun-form modifier, which we call as linguistic annotations seem easier, but using them in collaboration with semantic annotations arises new paradigm for reasoning with rules, in addition to reasoning of semantic annotations with domain ontology. Both these cases are studied in section 4.2. At the same time we would like to mention that our goal is not to fully understand the text, but to modestly complement automatic annotation with some richer mechanisms, knowing that the usability of one or the other is domain dependent. We call an inference allowing to complete semantic knowledge by using other types of knowledge.

To achieve such an reasoning system taking into account of various forms of knowledge, our hypothesis is that this knowledge management and reasoning problem can be solved by using semantic web technologies. Ontologies can be used in order to provide formal grounding for representing the semantics of knowledge elements; they can guide creation of semantic annotations constituting a set of all meta-level characterizations easing knowledge source description, evaluation, and access. *DL*-Reasoners can be used to perform reasoning over the knowledge base formed from various sources, but represented in standardized OWL2 format. Rules languages can be used to write rules, that can be used to extract new annotations, based on the existing annotations and known information patterns. And one more important hypothesis is that Higher Order DL constructs can be translated into Classical DL constructs.

We are motivate to reason with annotations of texts because of it's importance to improve the results, and this is the subject of discussion of our thesis. It can be helpful at least for two tasks. The first is adding annotations to existing ones – e.g. proposing new annotations. The second is to detect and deal with inconsistent annotations, which is important when analyzing annotations made by different annotators or generated automatically without golden standards - or simply helping the expert remaining consistent. The first task is achieved by using hand written or learned logical rules, based on semantic web rule language and a shallow representation of the annotations of text. For the second task, we make use of the logical reasoning of annotations with respect to a domain ontology. The whole system follows the principle that semantic annotations can be analyzed automatically, supported by techniques developed in the Semantic Web field.

1.4 Contribution of the Thesis

This thesis is an extension and improvisation of the research work at the LIPN lab, Universit Paris 13 (accepted at the 24th FLAIRS conference [29]). The major contribution of this research work is to deal with various paradigm of reasoning with text annotations. Taking into account of the formalism proposed by the previous research the annotations are represented in a format suitable for knowledge representation and are interpreted in various ways(like strong satisfiability, weak satisfiability and classical interpretation) for various different reasoning tasks (inconsistency checking of semantic annotations with respect to domain ontology, entailment of semantic annotations or deducing new annotations by logical annotation rules). We also present the paradigm to reason with annotated corpus, by considering various forms of annotations viz semantic, syntactic, morphological in an single framework by integrating with rules developed by analysis of the linguistic patterns often missed out by other previous works. Reasoning with annotation rules opens up a wider spectrum to further research on various patters of annotations and relationship. The research work is accompanied by working prototype implementation, to demonstrate the importance of the proposed technology, which can provide useful and interesting outputs which can be inputs to other works related to annotated texts. Moreover, this work also points out the way for evaluation of semantic annotations, without using the golden standard, but making use of logical reasoning and axioms defined in ontology.

1.5 Organization of the Thesis

This thesis contains 6 chapters : Introduction, Preliminaries, Text Annotation, Knowledge Representation and Reasoning, Technology Application and Epilogue. It also has two appendix sections for proof of theorems, application results and worked out examples. Each chapter ends with a summary, which highlights the important readings of the chapter, necessary to be able to follow the next chapter. Summary of these chapters are following:

- **Introduction**: This chapter [1] gives readers a brief over-view of our research. The basic idea, problems and motivation behind the work is justified along with the overall contribution of the thesis. It contains a global summary of the thesis as well.
- **Preliminaries**: In this chapter [2] we will provide all background knowledge for the thesis. Here, we have introduced the different logical languages involved on the proposed solution such as: DL,

HDL, QCL, SWRL and justify their necessity. Semantic Web technologies used for prototyping the formalism such as OWL2, OWLAPI, JENA, PELLET are also discussed and justified.

- **Text Annotation**: This chapter [3] is the entry point for understanding the problem under consideration. We have introduces the various types of annotations, historical background, their relationships and our way of handling the text annotations. Various components involved on wider view of formalism are discussed.

- **Knowledge Representation and Reasoning**: This chapter [4] focuses on Knowledge representation issues to handle the different types of annotations. We discuss on the different paradigms of reasoning with annotations with necessary algorithms to handle cases with existing semantic web technologies. By the end of this chapter, reader will be familiar with how Knowledge base developed for the overall application along with reasoning methodologies.

- **Technology Application**: This chapter [5] explains the prototype model that has been developed to explore the proposed technology. It presents the system architecture, operation, optimization and discusses on the statistical analysis of the results. The chapter involves the necessary algorithms for our implementations supported by examples wherever necessary.

- **Epilogue**: As usual, this chapter sums up our work. It explains the current state of application along with further enhancements that can be achieved.

Chapter 2

Preliminaries

"Most of the mistakes in thinking are inadequacies of perception rather than mistakes of logic."
Edward de Bono

The problem that we are going to address involves a number of varying logical languages and This chapter builds up the foundation for the logical and technical description of the overall research. It introduces all the technologies being used for the formalism, along with the ones used for the implementation. The chapter divided into two sections discusses on the basic formalisms of the formal logical languages used for handling the knowledge representation and reasoning issues of the problem being addressed by the thesis. Since, thesis is supported by a working prototype, background of various technologies used is important to understand the system architecture and operations, which are discussed in the latter half of this chapter. At the same time, we have also included the definition of some of the terms, frequently used in the latter chapters to make it clear to the reader about the concepts being discussed.

2.1 Logics in Action

The work we had undertaken is a research in applicative logics. It's not only an effort to build up text processing application from the perspective of Computational Linguistics but demonstrate the application of Computational Logics and existing technologies in development of useful applications. The theories thus proposed involves various languages of logics, which are discussed briefly in this section.

2.1.1 Description Logics (DL)

Description logic (DL) is a well-known fragment of first order logic together with the development of knowledge representation system because it provides the logical foundation for ontologies and Semantic Web. In general, basic description logic is decidable and expensive *enough* for information system modeling. The DL describes a system throughout *concepts*, *roles*, *individuals* and their relationship in the system where an individual is an object in the system, a concept is a class of objects and a role is a relation between two objects.

The language of DL in general often contains two disjoint sets of symbols for atomic concepts and atomic roles which corresponding to symbols of unary predicates and binary predicates. It also contains a set of symbols for concept and role construction such as : negation($\neg$), conjunction ($\sqcap$), disjunction ($\sqcup$), existence ($\exists$), universal quantification ($\forall$), cardinality restrictions ($\geq_n, \leq_n$) as well as a set of symbols for the relation between concepts and between roles such as : subsumption ($\sqsubseteq$). Almost description languages also contain $\bot$ and $\top$ symbols as special concepts. For different variants of *DL*, naming conventions and detail we suggest [1]. The DL logics for our concern is $\mathcal{ALCH}$, a combination of:

- $\mathcal{AL}$ Basic DL with atomic negation, concept intersection, value restriction and limited existential quantification
- $\mathcal{C}$ Complex concept negation
- $\mathcal{H}$ Role hierarchy

We now provide the syntax and semantics of $DL - \mathcal{ALCH}$ with reference to [41].

Syntax

Let U_c be a set of concept names, U_r be a set of role names, $A \in U_c$ and $R \in U_r$. Then the concept expressions in $DL - \mathcal{ALCH}$ are built inductively as follows:

$C ::= A \mid \neg C \mid C_1 \sqcup C_2 \mid C_1 \sqcap C_2 \mid \exists R.C \mid \forall R.C$

And concept is a collection of concept inclusions such as:

$C_1 \sqsubseteq C_2$

The (acyclic) role hierarchy is a collection of role inclusions such as:

$R_1 \sqsubseteq R_2$

Following axioms are the part of $\mathcal{ALCH}\ A - Box$

concept assertion	$C(a)$
role assertion	$R(a, b)$
role inclusion	$R \sqsubseteq S$
individual inequality	$a \neq b$

Example 2.1 *For example, we present a concept expression with role hierarchy:*

$\exists R_1.C_1 \sqcap \forall R_2.(C_2 \sqcup \neg C_3)$ *with* $\{R_1 \sqsubseteq R_2, R_2 \sqsubseteq R_3,\}$

Semantics

The semantics of $DL - \mathcal{ALCH}$ concepts is defined relative to an interpretation $I = (\Delta^I, .^I)$, where Δ^I is a non-empty set, called domain of I, and $.^I$ is a valuation that maps every concept name to a subset of Δ^I and every role name to a subset of $\Delta^I \times \Delta^I$. The interpretation can be lifted to concept expressions and satisfied by role inclusion (role hierarchy) as follows:

1. $(\neg C)^I = \Delta^I - C^I$
2. $(C_1 \sqcup C_2)^I = C_1^I \cup C_2^I$
3. $(C_1 \sqcap C_2)^I = C_1^I \cap C_2^I$
4. $(\forall R.C)^I = \{x \in \Delta^I \mid \forall y \in \Delta^I\}$: if $(x, y) \in R^I$ then $y \in C^I$
5. $(\exists R.C)^I = \{x \in \Delta^I \mid \exists y \in \Delta^I\}$: $(x, y) \in R^I$ and $y \in C^I$
6. $(R_1 \sqsubseteq R_2)^I = R_1^I \subseteq R_2^I$
7. $(C_1 \sqsubseteq C_2)^I = C_1^I \subseteq C_2^I$

Similarly the semantics for $A - Box$ is as given below:

8. $(C(a))^I = a^I \in C^I$
9. $(R(a, b))^I = (a^I, b^I) \in R^I$
10. $(a \neq b)^I = a^I \neq b^I$

Here, we would like to make two important definitions viz satisfiability and consistency which will be one of the major task for us to check over the knowledge base we will develop.

Definition 2.1 ***Unsatisfiable Concept:*** *A named concept C in the ontology O is unsatisfiable iff, for each interpretation I of O, $C^I = \phi$.*

Definition 2.2 ***Incoherent Ontology:*** *An ontology O is incoherent iff there exists an unsatisfiable named concept in O.*

2.1.2 Higher Order Description logic

Higher Order Description logic (HDL) is an extension of DL with higher order capabilities. Adding higher-order constructs to a DL form a spectrum of increasing expressive power, including domain meta-modeling, i.e., using concepts and roles as predicate arguments. This becomes an obvious case for our problem (more discussion on this in section 3.2(. Higher-version of a DL $\mathcal{L}$ is written as *Hi($\mathcal{L}$)*. Here we present a thorough introduction to the syntax of HDL and the reader is advised to follow [15] and [9] for further discussions on HDL.

Syntax:

Let $OP(\mathcal{L})$ the set of *operators* and $MP(\mathcal{L})$ set of meta-predicates for description logics ($\mathcal{ALCH}$ for our case). Let us assume two disjoint countably infinite alphabets: $\mathcal{S}$, the set of names, and $\mathcal{V}$, the set of variables. The building blocks of a *Hi($\mathcal{L}$)* knowledge base are assertions, which in turn are based on expressions. The set of expressions denoted by $\mathcal{E}_{\mathcal{L}}(\mathcal{S})$ are defined over the alphabet $\mathcal{S}$ for *Hi($\mathcal{L}$)* inductively as follows:

if $\mathcal{E} \in \mathcal{S}$ then $\mathcal{E} \in \mathcal{E}_{\mathcal{L}}(\mathcal{S})$;

if $C/n \in OP(\mathcal{L})$ and $E_1, ..., E_n \in \mathcal{E}_{\mathcal{L}}(\mathcal{S})$ then $C(E_1, ..., E_n) \in \mathcal{E}_{\mathcal{L}}(\mathcal{S})$.

Now in order to define the ***Hi$\mathcal{L}$*** assertion, let us define a set of $MP(\mathcal{L})$ of meta-predicates as:

A ***Hi($\mathcal{L}$)*** assertion over $\mathcal{E}_{\mathcal{L}}(\mathcal{S})$ is a statement of the form $M(E_1, ..., E_n)$ where $M \in MP(L)$, $n \geq 0$ is the arity of M, and for every $1 \leq i \leq n, E_i \in \mathcal{E}_{\mathcal{L}}(\mathcal{S})$. A ***Hi($\mathcal{L}$)*** knowledge base (KB) is a set of assertions over $\mathcal{E}_{\mathcal{L}}(\mathcal{S})$.

Example 2.2 *$Inst_R(Mileage; tf1; sa$:concept) , where $Inst_R$ is in the set $MP(\mathcal{L})$ to denote a role instance. $Mileage$ is an ontological concept, tf is ontological individual and sa :concept is a role and all of them belong to the set $\mathcal{S}$.*

The intended meaning is that the text-fragment $tf1$ is a concept defined by the concept $Mileage$ in the pntology.

Semantics:

The semantics of Hi (L) is based on the notion of interpretation structure which is a triple $\sum = \langle \Delta, I_c, I_r \rangle$ where: (i) Δ is a non-empty(possibly countably infinite) set; (ii) I_c is a function that maps each $d \in \Delta$ into a subset of Δ; and (iii) I_r is a function that maps each $d \in \Delta$ into a subset of $\Delta \times \Delta$. In other words, $\sum$ treats every element of Δ simultaneously as: (i) an individual; (ii) a unary relation, i.e., a concept, through I_c; and (iii) a binary relation, i.e., a role, through I_r.

An interpretation over $\sum$ is a pair $I = \langle \sum, I_o \rangle$, where triple $\sum = \langle \Delta, I_c, I_r \rangle$ is an interpretation structure, and I_o is a function that maps: (i) each element of S to a single domain object of Δ and (ii) each element $C/n \in OP(\mathcal{L})$ to an n-ary function $C^{I_o} : \Delta^n \rightarrow \Delta$ that satisfies the conditions characterizing the operator C/n. I_o is extended to expressions in $\mathcal{E}_{\mathcal{L}}(\mathcal{S})$ inductively as follows: if $C/n \in OP(\mathcal{L})$, then $(C(E_1, ..., E_n))^{I_o} = C^{I_o}(E_1^{I_o}, ..., E_n^{I_o})$.

To interpret non-ground terms, we need assignments over interpretations. An assignment μ over $\langle \sum, I_o \rangle$ is a function $\mu : \gamma \rightarrow \Delta$.

Now we describe how to interpret terms in $Hi(\mathcal{L})$. Given an interpretation $I = \langle \sum, I_o \rangle$ and an assignment μ over I, we define the function $(.)^{I_o, \mu} : \tau(S, \gamma) \rightarrow \Delta$ as follows:

if $t \in S$ then $t^{I_o,\mu} = t^{I_o}$;
if $t \in V$ then $t^{I_o,\mu} = \mu(t)$;
if t is of the form $C(t_1, ..., t_n)$, then $t^{I_o,\mu} = C^{I_o}(t_1^{I_o,\mu}, ..., t_n^{I_o,\mu})$.

Satisfiability:

Satisfiability of an assertion with respect to an interpretation I and an assignment μ over I is defined based on the semantics of the meta-predicates in $MP(\mathcal{L})$. For the meta-predicates, satisfaction in I, μ is defined follows:

$I, \mu \models Inst_C(E_1, E_2)$ if $E_1^{I_o,\mu} \in (E_2^{I_o,\mu})^{I_c}$;
$I, \mu \models Inst_R(E_1, E_2, E_3)$ if $\left\langle E_1^{I_o,\mu}, E_2^{I_o,\mu} \right\rangle \in (E_3^{I_o,\mu})^{I_r}$;
$I, \mu \models Isa_C(E_1, E_2)$ if $(E_1^{I_o,\mu})^{I_c} \subseteq (E_2^{I_o,\mu})^{I_c}$;
$I, \mu \models Isa_R(E_1, E_2)$ if $(E_1^{I_o,\mu})^{I_r} \subseteq (E_2^{I_o,\mu})^{I_r}$;

Definition 2.3 *A $Hi(\mathcal{L})$ KB $\mathcal{H}$ is satisfied by I if all the assertions in $\mathcal{H}$ are satisfied by I. As usual, the interpretations I satisfying $\mathcal{H}$ are called the models of $\mathcal{H}$. A $Hi(\mathcal{L})$ KB $\mathcal{H}$ is satisfiable if it has at least one model.*

2.1.3 Quasi-Classical Logics (QCL)

Quasi-Classical Logics (QCL) first introduced in [4] is based on the idea of weakening the classical logics by adding restriction on proof theory. In this restriction, compositional proof rules (disjunction introduction) cannot be followed by decomposition rule (resolution). Even though this weakens the classical logics, this logical language is appealing for reasoning with inconsistencies arising in applications, more generally in information fusion [8].

The most important concept of interest for us introduced by QCL is the two kinds of semantics, namely, QC weak semantics written as $\models_w$ and QC strong semantics written as $\models_s$. Readers are suggested to follow [22] and [48] for further discussions on QCL. For our purpose, we will just need the concept of entailment relation from QCL, which we will discuss over here. **Entailment:**

Here we define the concept of entailment relation ($\models_Q$) as in [22, pg 9]. As, stated before QCL provides us with strong and weak semantics for satisfiability denoted as $\models_s$ and $\models_w$ respectively. The ($\models_Q$) is defined as:

Definition 2.4 *$\models_Q$ is an entailment relation $\alpha_1, ..., \alpha_n \models_Q \beta$ iff $\forall X(X \models_s \alpha_1$ and ... and $X \models_s \alpha_n$ implies $X \models_w \alpha_1$*

where X is the interpretation and $\models_s$ and $\models_w$ mean strong satisfaction and weak satisfaction respectively, whose semantics is explained at [22, pg 6] and [22, pg 8] respectively.

This concept of entailment, based on strong and weak satisfaction, is used in our formalism for defining entailment of semantic annotations. In section 4.2.2.2 we will demonstrate the use of this entailment relation, where we define the weak and strong semantics for our formalism. One important achievement we can make is for inferring new annotations on text-fragment based on sub-class relations of the domain ontology.

2.1.4 SWRL

In the rule-based OWL reasoning paradigm, the asserted knowledge, that is the knowledge stemming directly from the ontology definition, is mapped into an internal rule engine representation format, and inference rules are applied in order to deduce new knowledge. The inference rules are based on OWL entailments [25], that are rules which describe the information that should be inferred based on existing knowledge.

A Semantic Web Rule Language (SWRL[1]) is based on a combination of the OWL DL and OWL Lite sub-

[1]http://www.w3.org/Submission/SWRL/

languages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sub-languages of the Rule Markup Language[2]. The rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. Now we will define the syntax, semantics and satisfiability of SWRL rules.

Syntax

Atoms on $SWRL$ are defined as:

$Atom \leftarrow C(i) \mid D(v) \mid R(i,j) \mid U(i,v) \mid builtIn(p, v_1, ..., v_n) \mid i = j \mid i \neq j$, where

$C =$ Class	$D =$ Data type
$R =$ Object property	$U =$ Data type property

$i, j =$ Object variable names or Object individual names
$v_1, ..., v_n =$ Data type variable names or Data type value names
$p =$ Built-in names

Then a $SWRL$ rule is defined as:

$a \leftarrow b_1, ..., b_n$ where,

a : head (an atom) b_i: body (all atoms)

A SWRL knowledge base (P) is defined by a finite set of rules.

Semantics

Let I be an interpretation, $V_{IX} =$ Object Variables, $V_{DX} =$ Data Type Variables and $P =$ Power Set Operator. Then I is a quadruplet such that $I = (\Delta^I, \Delta^D, .^I, .^D)$, where

$\Delta^I =$ Object Interpretation Domain
$\Delta^D =$ Datatype Interpretation Domain
$.^I =$ Object Interpretation function
$.^D =$ Datatype Interpretation function and
$\Delta^I \cap \Delta^D = \phi$

such that

$V_{IX} \rightarrow P(\Delta^I)$; $V_{DX} \rightarrow P(\Delta^D)$

With this definition of interpretation I, the following table shows the Binding B(I) for the SWRL atoms:

SWRL Atoms	Condition on Interpretation
$C(i)$	$i^I \in C^I$
$R(i,j)$	$(i^I, j^I) \in R^I$
$U(i,v)$	$(i^I, v^D) \in U^I$
$D(v)$	$v^D \in D^D$
$builtIn(p, v_1, ..., v_n)$	$(v_1^D, ..., v_n^D \in p^D)$
$i = j$	$i^I = j^I$
$i \neq j$	$i^I \neq j^I$

Table 2.1: Binding $B(I)$ for the SWRL atoms

Satisfiability

In order to define the satisfiability of a SWRL rule R such that $R = H \leftarrow B$, where H and B are the head and body of the rule respectively, we first define the satisfiability of B and H.

[2]http://ruleml.org/

Definition 2.5 *SWRL atoms in the body(B) or antecedent are satisfied,if it is empty (trivially true)* ***or*** *every atom of it is satisfied.*
SWRL atom in the head(H) or consequent is satisfied,if it is not empty ***and*** *it is satisfied.*

Then the satisfiability of rule R is defined as follows:

Definition 2.6 *A rule(R) is satisfied by an interpretation of I if and only if for every binding $B(I)$ that satisfies the antecedent, $B(I)$ satisfies the consequent.*

In the context of reasoning with annotations, integration of rules can be beneficial for analyzing annotations together, including propagating new semantic annotations, reporting missing semantic annotations, detecting incoherence among semantic annotations or syntactic annotations and deducing axioms based on annotations to be annotated to a complete sentence (explained with examples in section 4.2.1.1). At the same time SWRL rules can be used fairly easily to define rules with various built in OWL constructs.

Even though SWRL is undecidable, adding the restriction of DL-safety makes it decidable, for details on reasoning with DL-safe rules we suggest [18]. DL Safety is a simple idea which is implicit in many rule systems: variables in DL Safe rules bind only to explicitly named individuals in your ontology. In the context of our reasoning with annotations, DL-safety is valid, thus making SWRL a wonderful choice.

2.2 Semantic Web Technologies

In order to demonstrate the realization of the formalism proposed, a prototype application has been developed and successfully tested. This section is a brief discussion on the number of latest semantic web technologies used for application development.

2.2.1 OWL2

OWL2 is the latest version of OWL i.e. Web Ontology Language standardized by W3C Web Ontology Working Group[3]. Ontologies are formalized vocabularies of terms, often covering a specific domain and shared by a community of users[38]. They specify the definitions of terms by describing their relationships with other terms in the ontology. OWL 2 is an extension and revision of the OWL Web Ontology Language developed by the W3C Web Ontology Working Group and published in 2004. OWL2 comes with new features and constructs with increased expressivity. Two important features of concern for us are *Annotations* and *punning*, follow [37] for discussion on other features.

OWL2 extends the annotations in OWL1 and OWL2 allows annotations (such as labels or comments) on ontologies, entities, anonymous individuals, axioms, and annotations themselves. With this added features OWL2 becomes more expressive, but this is not enough for the application that we are considering. Hence here froth the reader must be aware that by annotation assertion we mean the one given our formalism rather than the one provided by OWL2.

Punning is an useful feature introduced in OWL2 which relaxes the requirement of OWL1 which strictly separates between the names of, e.g., classes and individuals. OWL2 DL relaxes this separation somewhat to allow different uses of the same term,[37] e.g., Eagle, to be used for both a class, the class of all Eagles, and an individual, the individual representing the species Eagle belonging to the (meta)class of all plant and animal species. However, it still imposes certain restrictions: it requires that a name cannot be used for both a class and a data-type and that a name can only be used for one kind of property. The OWL 2 Direct Semantics treats the different uses of the same name as completely separate, as is required in DL reasoners. Even though this is an interesting feature, this is not enough for our case, which is discussed further in section 3.2.

[3]http://www.w3.org/

2.2.2 OWLAPI

OWLAPI[4] is a Java API and reference implementation for creating, manipulating and serializing OWL and is closely aligned with the OWL 2 specification ontologies. It includes first class change support, general purpose reasoner interfaces, validators for the various OWL 2 profiles, and support for parsing and serializing ontologies in a variety of syntaxes.[19] discusses the architecture, features and supported interfaces provided by the API. One important point to be noted is that in OWLAPI, representation of class expressions and axioms is not at the level of RDF triples. Indeed, the design of the OWL API is directly based on the OWL 2 Structural Specification [34] i.e. an ontology is simply viewed as a set of axioms and annotations[5].

During the implementation OWL is the basic representation specification for storing semantic annotations, syntactic annotations, linguistic annotations and rules for our system. So, most important implementation of OWLAPI is for rendering all the necessary OWL documents including serializing the rules in XML format, which are otherwise written in clausal form for better readability. Besides it provides us an easy way to parse and traverse an OWL document for Higher order to Classical translation of the Knowledge Base. For a detailed explanation on the design of the API and class structures reader is advised to follow[2].

2.2.3 JENA

JENA[6] is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. Jena provides various tools, including I/O modules for: RDF/XML with rich model API for manipulating RDF graphs. Basically in Jena, each arc in a RDF model is a statement, comprising of: subject[the resource from which the arc leaves], predicate [the property that labels the arc] and object [the resource or literal pointed to by the arc].[5] is suggested for further reading on the architecture of the API, different features and useful examples.

Our application scenario involves some inputs/outputs in the format RDF and XML, and Jena provided a much easier and cleaner way of manipulating such resources. One important feature of Jena, which made it necessary for us to use Jena is handling reification. Jena provides an API notion of a reifiedStatement[7] that encodes a statement in a model as a reification quad. This API-layer notion is reflected down into the Graph interface. Each Graph has an associated reifier that is responsible for storing reified triples compactly. Reified statements are used in our context to handle axioms, commonly used for composite annotations.

2.2.4 Pellet

Pellet[8] is an OWL 2 reasoner and provides standard cutting-edge reasoning services for OWL ontologies. OWL-DL[9] restricts OWL-Full ontologies in several different ways and Pellet relaxes most of the OWL-DL restrictions and handles OWL-Full ontologies[10]. This makes pellet a much easier option to handle ontologies. For an detail study on architecture, reasoning strategies,query engine and other details [43] is suggested.

Pellet supports two different incremental reasoning techniques: incremental consistency checking and incremental classification, which is necessary for our case in reasoning with rules. We can directly load a file that contains SWRL rules into Pellet and rules will be parsed and processed, and can be used for reasoning with OWLAPI interface.Pellet interprets SWRL using the DL-Safe rules [35] notion which means rules will be applied to only named individuals in the ontology and fits as per our requirement. In addition to reasoning with rules, we need to check the consistency and coherency of the overall document with annotations with

[4]http://owlapi.sourceforge.net/index.html

[5]Concept of Text annotation is completely different from the annotation in OWL2

[6]http://jena.sourceforge.net/

[7]For example on reification http://jena.sourceforge.net/how-to/reification.html

[8]http://clarkparsia.com/pellet/

[9]http://www.w3.org/TR/owl-ref/#OWLDL

[10]http://clarkparsia.com/pellet/faq/owl-full/

respect to domain ontology. Pellet is the first,and currently the only, complete OWL-DL consistency checker and has the most coverage of OWL as a whole of any reasoner [43]. It also provides us the explanation for the inconsistency and the axioms behind it.

2.3 Summary

This chapter, has thus built-up the necessary logical and technical foundation, necessary for the readers to understand the overall work. Understanding of the logical language and the various necessary formalisms of each of them is important to understand the formalism proposed in chapter 4 (Knowledge Representation and Reasoning). While, the technologies are important for the understanding of chapter 5 (Technology Application). The references provided at various points are suggested to be followed for deeper understanding of each subsections, since we have precisely discussed within the boundaries necessary for the understandability of this work. It is assumed that by the end of this chapter is, the reader is now familiar with the problem statement in brief and technologies supporting the research issues addressed by the work. The next chapter, is the entry point for description of the research work in details.

Chapter 3

Text Annotation

"Logic is one thing, the human animal another. You can quite easily propose a logical solution to something and at the same time hope in your heart of hearts it won't work out."
-Luigi Pirandello

Annotation is basically a descriptive text containing the identification, function, location, physical characteristics, and other information concerning a target textual unit in the text corpus. Thus, the annotations provide a meta-data for the further understanding of the content of the document. In this chapter we will study on the different types of text annotations and the mention various sources of obtaining annotation over a text corpus. We define a number of terms related to text-annotation, frequently used in the latter chapters to make it clear to the reader about the concepts being discussed. In the first section, we would analyze the over all perspective of text annotations and some links to historical works in relation to our current work. In section 3.2, we will discuss on some issues, approaches and the modeling issues so as to justify the new formalism necessary for representation of semantic annotations. 3.3 discusses the process of using various forms of annotation knowledge obtained from different sources, for an integrated reasoning framework. The different building blocks involved for overall reasoning process are discussed in section 3.4, where we will introduce the various annotation types used for semantic annotation, understanding of which is very important to follow the rest of the work.

3.1 Landscape of Annotations

Annotation is the process of attaching a label to a piece of text. This definition covers in fact different practices, some of which date back to long time before the web : the piece of texts, the accepted labels, and the meaning of the annotation can be very different. Librarians for instance annotate whole documents with list of keywords taken in a given thesaurus, meaning each keyword has to see with the topic of the document. Literature studies annotate fragments (possibly sentences or words) with free text, where the annotation can be any meta-comment of the fragment. Book indexes annotate fragments (paragraphs or pages or chapters) with keywords. These keywords are specific to the book and indicate particularly relevant fragments for somebody interested in the topic represented by this keyword ; the notion of relevance at hand depends on the purpose of the book.

Text annotation process that we are considering is similar to the librarians one, in that the goal is to retrieve documents. It differs from it because the notion of document is wider : it may be any object identifiable on the web - part of text, graphic, picture, movie. Another essential difference is that annotations have a logical structure, commonly embeddable in an ontology, so they allow wording-in dependent queries and the answer involves logically implied relations (subclasses, inverse relations, . . .). Uren and colleagues [47] give a significant review of the field and consider requirements for quality and maintenance of annotations. They analyze separately three types of data involved : ontologies, documents and annotations. They find that the quality of an annotation still can be improved, with respect to its degree of automation and its quality.

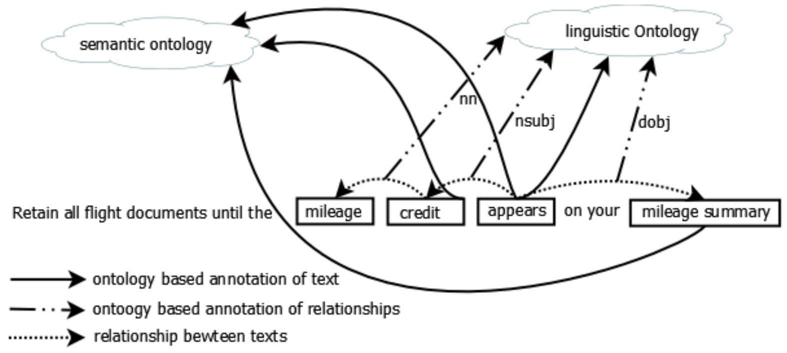
semantic ontology
linguistic Ontology
nn
nsubj
dobj
Retain all flight documents until the
mileage
credit
appears
on your
mileage summary
ontology based annotation of text
ontoogy based annotation of relationships
relationship bewteen texts

Figure 3.1: Ontology-based Annotation on texts

To further understand the landscape of the overall work that we are taking into consideration, consider the figure 3.1. The plain text *Retain all flight documents until the mileage credit appears on your mileage summary*, can be enriched with further information. Linguistic analysis can lead to the discovery number of syntactic and grammatical relationships and each word can be categoriezed into various parts of speech. At the same time, textual units can be provided with meanings, so that the user can understand the sentence. We distinguish these two cases of annotations as linguistic annotation and semantic annotation.

Linguistic annotations are the syntactic roles which are basically relations between two textual units, such as subject(‘appears’, ‘credit’) where the word ‘appears’ is of type verb and ‘credit’ is of type noun. Various formalisms have been used, which are generally translatable into graphs. Some efforts exist to accommodate the web norms (e.g. [24]. Where as semantic annotations, provide the further understanding of the textual unit, such as ‘credit’ can be linked to an ontological concept ‘Credit’ which has various properties and relationships with other concepts in the ontology.

In the scenario that we propose various types of ontologies are the backbones of the annotations which we term as Ontology-based Annotation. As shown in the figure above, two different types of ontologies provide meta information about the concepts being used to annotate the corpus so that we have an ontological interpretation such that the douments can be processed and understood by the machine. The above diagram shows the ontologies involved are semantic ontology and linguistic ontology, collectively referred as domain ontology henceforth, the structure of which is explained in section 3.4.3. Here we can observer that the word(s) of the sentences are annotated by various concepts defined in the ontology, e.g. words *mileage, credit, appears, mileage summary* are annotated from the domain ontology with various concepts and various annotation types [annotation types are discussed in section 3.4.1]. At the same time the relationships between words are considered by different types of relations like *nn, dobj, nsubj* etc. This diagram thus shows the overall picture of the landscape of ontology based text annotations.

It must be noted that, presently, most applications only use a part of ontological elements in annotations: either concepts, or individuals. Roles can be considered, with the help of patterns borrowed from information extraction(I.E.), but I.E. patterns are very specialized and costly to elaborate. Another need is to extend management of texts beyond information retrieval. For instance, [12] translates regulatory texts into a sequence of Abstract Syntax Trees, then into CTL formulas, and uses them to check the conformance of procedures. The Ontorule project tackles the annotation of business rules in policy texts to translate them into executable IT systems. They propose an architecture in the style of [**?**] but extend semantic data with typed rules according to a slightly extended SBVR classification [36]. Additionally, [30] extends the traditional pattern definition into context sensitive rules to add annotations.

3.2 Approaches and Problems

Much work has been done in the domain of Information Extraction for business intelligence, and good automated results are obtained for ontology classes (see the MUSING project and [32]). Annotation of regulatory and other specialized texts differ from information extraction in scale - each corpus is incomparably smaller - and in scope - the goal covers a larger part of the content. Figure 3.1 illustrates the idea of ontology based annotation where all the standard ontological elements are involved. We believe that the size of texts enables adding to standard tools and reasoning with the help of relations between heterogeneous knowledge which are generally not considered together. This is the deep motivation of our work : on a per application basis, some rules, depending on domain-specific language and knowledge, will improve the annotations, sufficiently to justify the design effort.

The most important problem to be noted is that a naive modeling could cause loss of information or representational mistakes. For instance, suppose we use a binary relation $hasSemAnno(tf, C)$ to represent that a text fragment tf is semantically annotated by concept C. Then two annotations, such as $hasSemAnno(tf, C_1)$ and $hasSemAnno(tf, C_2)$, are consistent even if disjointness of C_1 and C_2 results from a domain ontology. At the same time let us assume, that the same annotation be plainly represented as *tf is a C*, i.e. $C(tf)$. Although the previous inconsistency can be computed, due to multiple ontologies we may have to write both $City(tf)$ and $Noun(tf)$, which together entail that the concept $City$ and the grammatical category $Noun$ have overlaps — it is not intended either. Note that although the "punning" of OWL 2 relaxes the separation between the names of e.g., classes and individuals, it is not enough to solve the above representation problems.

To solve the problem, semantic annotations are considered apart, and *annotation assertions* are used to account of them. Globally, annotation assertions allow to perform various forms of reasoning which will be discussed in chapter 4, and, as a prototype implementation in chapter 5. An interesting approach that we have undertaken is to make use of rules to bridge inferences about constraints between semantic and linguistic.

3.3 Integration

As stated beforehand, the tool like Stanford CoreNLP suite are available, which can be used to process the same text corpus to obtain different types of annotations. It provides a set of natural language analysis tools which can take raw English language text input and give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, and mark up the structure of sentences in terms of phrases and word dependencies, and indicate which noun phrases refer to the same entities. It integrates tools for the English language, including the part-of-speech (POS) tagger, the named entity recognizer (NER), the parser, and the coreference resolution system. It thus enables us to quickly and painlessly get complete linguistic annotations of natural language texts. At the same time Stanford CoreNLP API is also available to use as per ones use[1].

[1] see http://nlp.stanford.edu/software/corenlp.shtml for details explanation

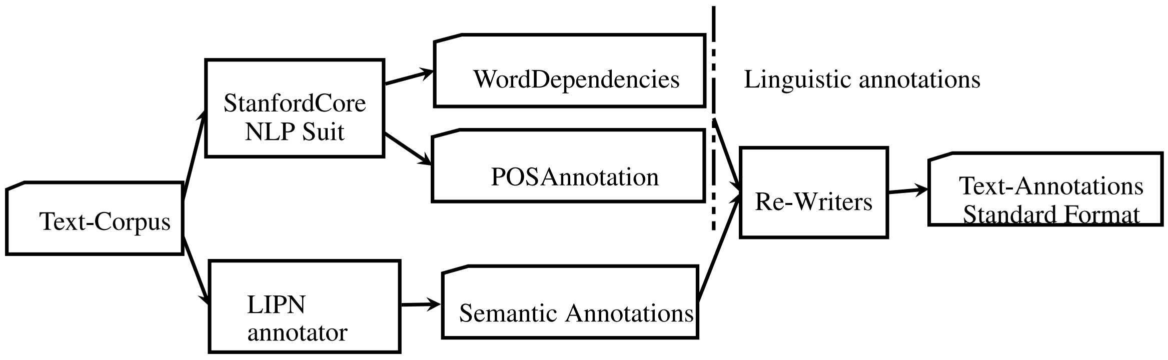

Figure 3.2: Integrated View of Text-Annotations generation and representation

Figure 3.2 shows the view of generation and representation of different types of annotations. As in figure from here on we will mention all forms of annotations, other than semantic annotations viz. syntactic annotations, morphological annotations or whatsoever is available as linguistic annotations. Semantic annotations are produced by *Semantic Annotator* produced at LIPN, University of Paris 13.

3.4 Building Blocks

An annotation notes a link between a text fragment and some knowledge, so it relies on several models: a text model on the one side to describe annotated fragments, a knowledge model on the other to describe annotations [28]. In order to be able to cope with several sorts of knowledge, several independent knowledge models are needed. In this section, we will explain the different different blocks and role played by each knowledge model on the overall application. It is to be noted that all forms of knowledge (semantic, syntactic, linguistic) are described as ontologies, which has a good compromise between expressive power and computability.

Before further discussion, we would like to define some terms to be used henceforth.

Definition 3.1 (Textual unit:) *Each word in the text-corpus, is defined as an textual unit. So, we can say that each textual unit or group of textual units can be annotated with entities from an ontology.*

Definition 3.2 (Text-Link:) *Textual unit(s) with unique semantic annotations refer to Text-Link. It is an representation used by the semantic annotator for storing the annotations. If one textual unit or group of textual units are annotated with **n** different entities, then we will have **n** text-links.*

Definition 3.3 (Text-Fragment:) *Text-Fragment makes the direct one-to-many relationship between textual-unit(s) and semantic annotation. If one text-link is annotated with **n** different entities, then we will have **1** text-fragment, with **n** relations to domain ontology.*

It is to be noted that each text-link/text-fragment is stored as RDF node/OWL individual and each node/individual contains enough information to identify the textual unit in text corpus, corresponding to the text-link/text-fragment.

Definition 3.4 (Domain Knowledge:) *Domain knowledge is the knowledge used to refer to field of consideration for processing the text-corpus. It's described as ontology and is either developed automatically or manually by specialists and experts in the particular field.*

3.4.1 Semantic Annotations

Semantic annotations map the link between the textual units with the domain knowledge. As justified in section 3.2, a direct modeling of semantic annotation leads to loss of information or lead to relational mistakes. So, we propose following five annotation types considered in our formalism, with their formal names given in brackets. *The semantics of each type is discussed in the next chapter.*

Concept Annotation (sa:Concept) Some text fragments denote ontological concepts by themselves (e.g. non-elite status member). Such text fragments are usually referred as elements of a domain terminology. This is the most frequent case in our working corpus.

Role Annotation (sa:Role) Similarly, text fragments may also denote conceptual roles if the underlying notions have been encoded as roles rather than as concepts in the ontology (e.g. applies to, be discontinued for or reservation).

Individual Annotation (sa:Individual) Some text fragments directly denote ontological individuals. They are traditionally referred to as named entities, such as X Airline which refers to a specific airline company, or the minimum mileage guarantee, a special policy name. In this case, the semantic annotation is the ontological individual itself.

Individual-Concept Annotation (sa:Ind-Con) Other text fragments refer to individuals but their annotations indicate the concept they belong to. This is the often case when using concept City as a label of Paris or when labeling the minimum mileage guarantee by concept Policy.

Note that when a concept from a reference ontology is used to annotate a text fragment, two cases are possible: either the text fragment talks about a special instance of this concept (sa:Ind-Con), or it is about the concept itself (sa:Concept) instead of any specific individual.

Axiom Annotation (sa:Axiom) Axioms cover all the complex annotations. For the semantic ones, an axiom can be a sumbsumption relation, as is justified as annotation of *"AAmembers earn mileage"*, which can be annotated as an axiom corresponding to the relation *earn(AAmember, mileage)*. Even though axioms can also be a unary fact, as partner(YY company) but role structure are important, e.g. has subject(textfragment1, textfragment2), where the role fillers are text fragments since their semantic value need not be known at the moment. All the same, at the discourse level, the main axioms met in our experience are completed roles, e.g. explanation(sentence1, sentence2).

With this last type of semantic annotation i.e. we *sa:Axiom*, we can introduce two types of semantic annotations, which we define now and the syntax and semantics of semantic will be discussed in the next chapter.

Definition 3.5 (Simple Annotation:) *An annotation type is called simple if annotations are only concerned with one single text fragment.*

Definition 3.6 (Composite Annotation:) *An annotation type is defined as the one under which annotations concerns more than one text fragment. For instance the relational instance and the subclass axiom of semantic annotation types.*

Example 3.1 *Suppose we have a sentence "Your summary includes flight and participant mileage earned" with eight textual units. The textual unit "summary" is semantically annotated of type* sa :*Concept from an ontology. But if the whole sentence which will form one text fragment of eight textual units is annotated as a subclass declaration (accounting for "includes") between the text fragments "participant mileage earned" and "summary", then we have a composite annotation.*

3.4.2 Linguistic Annotations

Linguistic Annotations provide linguistic information about the segments in the primary data, e.g., a morpho-syntactic annotation in which a part of speech and lemma are associated with each segment in the data. It also includes the identification of a segment as a word, sentence, noun phrase and various grammatical relations between textual units. Different types of linguistic annotations are produced by various tools. [23] is a good reference on understanding the concepts of linguistic annotations, while [31] and [6] provide readings on POS tagging and dependency grammar respectively.

In our context we take different annotations, generated by different tools, and the format in which the annotation is rendered, e.g. XML, LISP, etc. independent of its content. For example, a phrase structure syntactic annotation and a dependency-based annotation may both be represented using XML, even though the annotation information itself is very different. But the raw annotations, that we directly obtain by various tools are not useful enough for us to work with reasoning. So, each linguistic annotations are rewritten as OWL. the algorithm for doing this transformation is explained by the algorithm 2 in the chapter Technology Application, which is specific to our implementation.

3.4.3 Domain Ontologies

Domain ontologies under our consideration are semantic ontology and language ontology. We have made the distinction on these based on the fact that semantic ontology provides semantic labels for semantic annotation, and represents domain knowledge while language model embeds text model and linguistic knowledge. Both ontologies are described in OWL.

Semantic ontology is used for annotating the text-corpus. Our formalism, accepts any OWL ontology as the semantic one. The figure 3.3 shows a sub-ontology of the ontology taken into consideration.

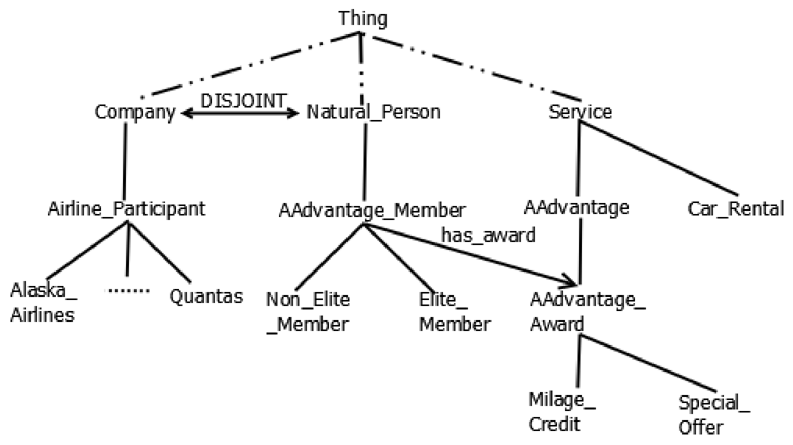

Figure 3.3: Sub-Ontology of Airline Services

This sample sub-ontology, clearly depicts the information that we want to exploit by making use of ontology to represent the domain knowledge.

The language ontology on the other hand is used to describes textual level information. As a basis, the concept name *TextFragment*, whose individuals are annotatable segments of texts, belongs to the ontology. Three role names *contains, contains2*, and *contain3* are also included to describe the containing relation between text fragments, satisfying the following conditions:

$$\begin{aligned}
&(1)\ \text{ObjectPropertyDomain}(R\ \ TextFragment),\\
&(2)\ \text{ObjectPropertyRange}(R\ \ TextFragment),\\
&\qquad \text{where } R \in \{contains, contains2, contains3\};\\
&(3)\ TextFragment \sqsubseteq (\leq 2)contains2.TextFragment;\\
&(4)\ TextFragment \sqsubseteq (\leq 3)contains3.\ TextFragment;\\
&(5)\ SubObjectProperty(contains2, contains);\\
&(6)\ SubObjectProperty(contains3, contains).
\end{aligned}$$

The first (resp. second) OWL axiom says that the Domain(resp. Range) of all three roles is TextFragment. The third (resp. fourth) puts a constraint on *contains2* (resp. *contains3*) : any given TextFragment instance can only be related with at most two (resp. three) TextFragment instances. The fifth and the sixth state that *contains2* and *contains3* are sub-roles of *contains*.

In this same language ontology, we can have various linguistic annotation types. We have taken into consideration of some of them like, the noun compound modifier (nn for short),direct object (dobj for short) and nominal subject (nsubj for short) as defined in [6] as an example. These are represented as any set of binary syntactic relations (with the help of as many roles). Basically they follow the following conditions:

$$\begin{aligned}
&(1)\ \text{ObjectPropertyDomain}(R\ \ TextFragment),\\
&(2)\ \text{ObjectPropertyRange}(R\ \ TextFragment),\\
&\qquad \text{where } R \in \{nn, dobj, nsubj, subj, obj\};\\
&(3)\ dobj \sqsubseteq obj;\\
&(4)\ nsubj \sqsubseteq subj.
\end{aligned}$$

The subset relations 3 and 4 are added, by exploiting the grammatical relation hierarchy defined in [6, fig 2].

Part of Speech (POS) tagging are also taken into consideration with the help of relation *hasPos* and POS constants, as in *hasPos(tf, Verb).*

3.5 Summary

In this chapter, we have built the theoretical background necessary for understanding the formalism, being discussed in the next chapter. The various definitions, examples and topics that have been discussed, are important to understand the overall perspective of text annotation, while at the same time provides necessary base for discussing our solution to handle reasoning with annotations. The points of discussion in section 3.4.1 on types of semantic annotations and the structure of domain ontologies in section 3.4.3 are important to follow up the logical formalism and rules being discussed in the next chapter. On the whole, by the end of this chapter, it is assumed that the reader is now familiar with the necessity, process, existing problems and types of text annotations along with the various components involved in our proposed solution for handling reasoning with text-annotations.

Chapter 4

Knowledge Representation and Reasoning

"Logic is the technique by which we add conviction to truth"
Jean de la Bruyere

In the previous chapter, we have made it clear regarding the different types of annotations (including related basic concepts), domain ontologies into considerations and the fields of logics that will be used for achieving the goal. This chapter presents the knowledge representation formalism (SAoT) proposed in order to achieve reasoning with annotations in compliance to the existing technologies. We also present the interpretation of SAoT by higher order description logics and meta modeling mechanism of ontology web language (OWL2[1]).

In general, text annotation is simply adding label to the textual unit(s), providing a lots of information for human user. But, it is still open the way on how precisely this attachment is to be interpreted. For machine processable applications, the method of storing the annotations and interpretations of annotations are to be well defined and standardized. In this chapter thus we analyze the issues and provide solutions to the knowledge representation issues of various types of annotations.

Reasoning with annotations can be performed for basically two purposes: Firstly to deduce new annotations and secondly to check the correctness of the annotations. From our studies, we have observed that we can integrate rule languages with our annotation base and can perform reasoning without considering domain ontology to obtain new annotations based on existing ones. While, checking the correctness of annotations must be performed with respect to domain ontology. These two forms of reasoning address different problems, yet we do not want to represent annotations in different formats for different purposes.

Thus in the section Knowledge Representation we propose a formalism to standardize the annotations obtained by other third parties. Then in the section of Reasoning we will address the two forms of reasoning separately and deal with algorithms proposed to use the annotations as per the need.

4.1 Knowledge Representation

As, already stated in preceding chapters and pin pointed in figure 3.2 the annotations on a corpus are obtained by various independent methods and procedures. Thus, obtained annotations are in different for-

[1]OWL2 also has the notion of annotation, which is different from the one we have used here. For details on OWL2 annotations follow http://www.w3.org/TR/owl2-syntax/#Ontology_Annotations

mats often unsuitable for reasoning. At the same time, since one of our motive is to address the modeling issues discussed in section 3.2, it opens up the issue for formalism for annotations representation. So, in this section, we propose the solutions to represent variant knowledge in an annotated corpus.

4.1.1 Semantic Annotations

Semantic annotations are one of the major components of annotated corpus, thus needing to be handled carefully. In section 3.4.1 we have already discussed about different types of semantic annotation types viz. sa:Individual, sa:Ind-Con,sa:Concept, sa:Role. Let us consider that a text fragment TF is annotated by a concept/ a role (sa :*isConceptOcc* / sa :*isRoleOcc* annotation type), then the meaning of TF is covered by its annotation term, but not necessarily equal to it (the exact meaning can be a subtype of the annotation). In practice, it often happens that a concept which is not the exact meaning is chosen to annotate a text fragment. This happens either because sometimes the most refined annotations are not necessary for application scenario, or because they are hard to discover or may not occur in domain ontology because of the incompleteness of the ontology. Only for the case of annotation type sa:Individual, the meaning of the text fragment is supposed to be the instance in the ontology. Finally, for annotation type sa:Ind-Con, it states that the meaning of the text fragment is a special instance of the annotation concept, and the choice of the concept is a matter of relevance.

The following subsections give the necessary syntax and semantics used for representation of annotations, without loosing the actual meaning, that they are supposed to convey. The formalism that is used is taken directly from [29].

4.1.1.1 Syntax

Description Logics (DL), the logics underlying the OWL is the syntactic base for our formalism. It is assumed that reader is familiar with DL; [1] is suggested for comprehensive background reading.

Vocabulary *N* of each classical DL language *L* is composed of three pairwise disjoint parts: a set of concept names (or atomic concepts) N_c, a set of role names N_r, and a set of individuals N_i. Then, complex concepts are built from N by a set of concept construction operators of L. For our purpose we use the DL language $mathcalALCH$ whose complex concepts can be defined inductively as follows, where $A \in N_c$, $R \in N_r$:

$$C \rightarrow A \mid C_1 \sqcup C_2 \mid C_1 \sqcap C_2 \mid \neg C \mid \forall R.C \mid \exists R.C$$

Axioms in $\mathcal{ALCH}$ can be in the form of $C \sqsubseteq D, R \sqsubseteq S, A(a)$, and $R(a, b)$ with $a, b \in \mathrm{N}_i$. More expressive domain and language ontologies, not required for our current application scenario, can be treated similarly if necessary.

In the formalism, we are given three pairwise disjoint sets of names $\mathrm{N}_{type}, \mathrm{N}_{domn}, N_{lang}$, where $\mathrm{N}_{type} =$ sa:Concept, sa:Role, sa:Individual, sa:Ind-Con$\}$, N_{domn} and N_{lang} are vocabularies of a domain ontology O_{domn} and a language ontology O_{lang} [2]. The set of concepts (resp. roles, individuals) names of an ontology O is denoted as *Concept*(O) (resp. *Role*(O), *Individual*(O)), or, abusing the notation, *Concept*(N_O) (resp. *Role*(N_O), *Individual*(N_O)).

Definition 4.1 (Semantic annotation assertion) *A semantic annotation assertion is a triple in the form of* $\langle tf, ot, at\rangle$ *satisfying* $tf \in TextFragment$ *and the following conditions:*

- *If* $at \in \{$*sa:Concept*, *sa:Ind-Con*$\}$, *then* $ot \in$ *Concept*(O_{domn});
- *If* $at \in \{$*sa:Role*$\}$, *then* $ot \in$ *Role*(O_{domn});
- *If* $at \in \{$*sa:Individual*$\}$, *then* $ot \in$ *Individual*(O_{domn});

Finally, a text annotation knowledge base is defined below:

[2] N_{lang} contains at least $\{TextFragment, contains, contains2, contains3\}$. Other concepts in N_{lang} can be the concept *POS* with instances *Verb, Noun, Adj, Adv*, etc. Other roles can be nn and $hasPos$ as discussed in Section 3.4.3

Definition 4.2 *A text annotation knowledge base is $TaKb = O_{domn} \cup O_{lang} \cup AnnoAsserSet$, where $AnnoAsserSet$ is a set of semantic annotation assertions.*

4.1.1.2 Semantics

In this section, we will provide the formal semantics of the formalism that we have proposed. In order to do so we make use of ability of interpreting meta-data descriptions, inspired by the higher-order semantics of description logics [15], which is an extension of classical description logics. We have taken into consideration of only simple annotations due to the time constraint on completing the research on composite annotations.

Semantic concepts or roles need to be dealt with as individuals when annotating text fragments, and as concepts or roles to perform reasoning on annotations. All the same, different text fragments may be interpreted one as concept, another as role or individual. To address this problem and to allow for richer reasoning abilities over annotations of texts, we now extend classical model-theory semantics of description logics with the ability of interpreting annotation assertions, which is inspired by the higher-order semantics of description logics [16]. An interpretation I is first defined, which interprets the ontology term and the text fragment individuals in a higher-order way, leaving other elements of the language ontology and the four annotation types interpreted classically.

Definition 4.3 *For a text annotation knowledge base $TaKb$, an interpretation I is a 4-tuple (Δ, I_i, I_c, I_r), where the set Δ is called the domain of I, and where the mappings I_i, I_c, I_r satisfy:*

1. $I_i : N_{domn} \cup \mathit{Individual}(N_{lang}) \to \Delta$;
2. $I_c : \Delta \cup \mathit{Concept}(N_{lang}) \to 2^{\Delta}$;
3. $I_r : \Delta \cup \mathit{Role}(N_{lang}) \cup N_{type} \to 2^{\Delta \times \Delta}$;

Definition 4.4 *An interpretation $I = (\Delta, I_i, I_c, I_r)$ is called* extensible *if and only if I_i can be extended to a mapping of complex concepts of the domain ontology into Δ such that:*

- $I_c(I_i(\neg C)) = \Delta \setminus I_c(I_i(C))$;
- $I_c(I_i(C \wedge C')) = I_c(I_i(C)) \cap I_c(I_i(C'))$;
- $I_c(I_i(C \vee C')) = I_c(I_i(C)) \cup I_c(I_i(C'))$;
- $I_c(I_i(\forall R.C)) = \{x \mid \forall y.(x, y) \in I_r(I_i(R)) \text{ implies } y \in I_c(I_i(C))\}$;
- $I_c(I_i(\exists R.C)) = \{x \mid \exists y.(x, y) \in I_r(I_i(R)) \text{ and } y \in I_c(I_i(C))\}$.

Definition 4.5 *Given an extensible interpretation I,*

- *I satisfies $C \sqsubseteq D$ if and only if $I_c(I_i(C)) \subseteq I_c(I_i(D))$ for C, D in O_{domn}, $I_c(C) \subseteq I_c(D)$ otherwise;*
- *I satisfies $R \sqsubseteq S$ if and only if $I_r(I_i(R)) \subseteq I_r(I_i(S))$ for R, S in O_{domn}, $I_r(R) \subseteq I_r(S)$ otherwise;*
- *I satisfies $C(a)$ if and only if $I_i(a) \in I_c(C)$ for $C \in \mathit{Concept}(N_{lang})$ and $I_i(a) \in I_c(I_i(C))$ otherwise;*
- *I satisfies $S(a, b)$ if and only if $(I_i(a), I_i(b)) \in I_r(S)$ for $S \in \mathit{Role}(N_{lang}) \cup N_{type}$ and $(I_i(a), I_i(b)) \in I_r(I_i(S))$ otherwise.*

4.1.1.3 Satisfiability

An interpretation I satisfies a semantic annotation assertion $TA = \langle tf, ot, at \rangle$, written $I \models_s TA$, if and only if I is extensible and satisfies:

- $(I_i(tf), I_i(ot)) \in I_r(at)$
- $I_i(tf) = I_i(ot)$ if $at =$ sa:Individual;

  $I_c(I_i(tf)) \subseteq I_c(I_i(ot))$ if $at =$ sa:Concept;

  $I_i(tf) \in I_c(I_i(ot))$ if $at =$ sa:Ind-Con;

  $I_r(I_i(tf)) \subseteq I_r(I_i(ot))$ if $at =$ sa:Role.

Given a text annotation knowledge base $TaKb = O_{domn} \cup O_{lang} \cup AnnoSet$, an interpretation I satisfies (is a model of) $TaKb$, written $I \models_s TaKb$, if and only if I is extensible and satisfies O_{domn}, O_{lang}, and $AnnoSet$, as defined above. $TaKb$ is said satisfiable if it has a model. Note the use of the symbol $\models_s$ is also motivated because we take this semantics as the strong satisfiability of the annotations.

Since, we will also be considering weak satisfiability which will be used to compute the entailment of semantic annotations, let us define the weaker semantics. So, the weaker semantics is:

Definition 4.6 ***Weak Satisfiability of Semantic Annotation:*** *An interpretation I weakly satisfies a semantic annotation assertion $TA = \langle tf, A, sa{:}Concept \rangle$, written I $\models_w TA$, if and only if I is extensible and satisfies:*

- $I_i(tf) = I_i(ot)$ *if* $at =$ *sa:Individual;*
- $I_c(I_i(tf)) \subseteq I_c(I_i(ot))$ *if* $at =$ *sa:Concept;*
- $I_i(tf) \in I_c(I_i(ot))$ *if* $at =$ *sa:Ind-Con;*
- $I_r(I_i(tf)) \subseteq I_r(I_i(ot))$ *if* $at =$ *sa:Role.*

With the definition, of the satisfiability, we can now define the coherency of the knowledge base. Checking the consistency of all concept names mentioned in a TBox is called a TBox coherence check.

Definition 4.7 ***Incoherent TaKb:*** *The knowledge base $TaKb$ is incoherent iff there exists an unsatisfiable named concept C in $TaKb$.*

4.1.2 Representation of Linguistic Annotations

The different types of linguistic and syntactic relationships or tags on words are collectively called *linguistic annotations* in our domain of problem as also shown in figure 3.2. We have considered word form dependencies and part of speech annotations, obtained by using Stanford NLP Suit. The major motive to use linguistic annotations is to use them in integration with semantic annotations and analyzed linguistic patterns, to deduce new annotations. But, linguistic annotations do not play much role in reasoning with domain ontology, thus freeing us from complex representation issues of linguistic annotations. Our observations revealed that linguistic annotations can be represented as pure binary relations between text-fragments. Thus T-Box consists of various roles as defined by language ontology in section 3.4.3. And, A-Box for linguistic annotation ontology is $\mathcal{ALCH}$.

Definition 4.8 (Linguistic Annotation Assertion:) *An linguistic annotation assertional axiom is an expression of the form $r(tf1, tf2)$, where individual $tf1$ and $tf2$ are individuals corresponding to text-fragments and r is role name corresponding to the relationship between text-fragments. Roles r are as defined in the linguistic ontology as discussed in section 3.4.3.*

Example 4.1 *Let us consider the sentence we have considered in figure 3.1. The linguistic annotation nn between text-fragments corresponding to textual-units mileage and credit can be asserted by an $\mathcal{ALCH}$ axiom $nn(tf8, tf7)$, where $tf8$ and $tf7$ are the text-fragments representing the textual-units at position 8 and 7 respectively in the considered sentence.*

Since all the linguistic annotation assertions are $\mathcal{ALCH}$, the semantics for them are same as discussed in section 2.1.1.

4.2 Reasoning

After having represented the annotations in a formal way, under pinned with formal syntax and semantics, we can address the issues with reasoning with annotations. As, already stated, we are dealing with two forms of reasoning with annotations viz. with rules and with domain ontology, as depicted by the figure below.

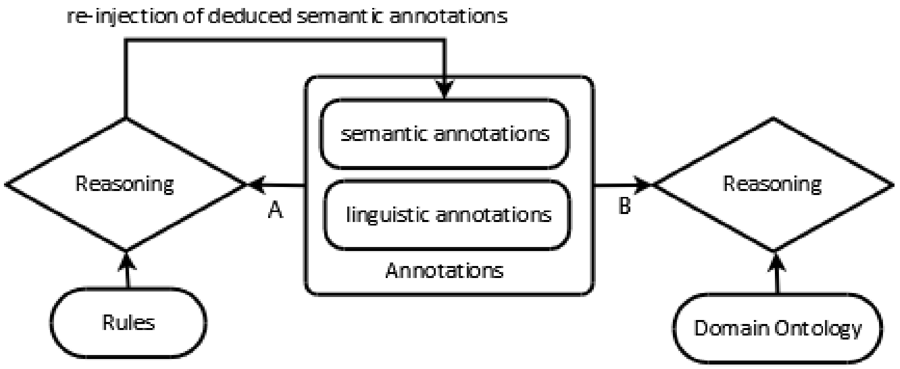

Figure 4.1: Reasoning Paradigm with text annotations

As seen from the above figure, two reasoning operations viz. A and B are defined, such that in case of A: rules are considered for reasoning, while in B: domain ontology is considered for consistency check of semantic annotations with respect to the domain ontology. In the over all reasoning paradigm A is performed first, so that semantic annotations deduced at this stage are re-injected to the knowledge base and are used for further reasoning with domain ontology in B. In the subsections that follows, we deal with each of these paradigms separately. Before we discuss further, we are to mention that we are motivated to perform the reasoning with existing highly optimized classical DL reasoners.

4.2.1 Reasoning with rules

In this section, we will discuss the reasoning with rules to show the benefit of providing a platform for reasoning with various kinds of annotations of texts. Here we will discuss only a set of rules which are proven to be working with the test corpora. This paradigm of reasoning over annotated corpus opens up a novel way of reasoning and leaves a wide space for other rules to be used with other types of semantic and linguistic annotations. One may argue that SPARQL might be enough to query the knowledge base to find information on the existing knowledge base, but our observation revealed that SPARQL is not enough to deduce new semantic annotations, with integration to linguistic annotations. We will explain further with example rules to clarify the need of rule based reasoning. The following subsection explains rules with examples, while the next section 4.2.1.2 deals with the transformation of semantic annotations formally represented as discussed in 4.1.1 into the representation enough for our reasoning need with rules.

4.2.1.1 Rules

To, begin with we extend the language ontology O_{lang} with an extra syntactic annotation type nn (noun compound modifier), which will be used in the set of rules under our considerations. In the following, a number of rules explained with examples are introduced to show the benefits of analyzing them together, including recognizing conflicting annotations 1, propagating new semantic annotations (Rules 2 and 3), reporting missing semantic annotations (Rule 4 and Rule 5), and detecting incoherence among semantic annotations or between syntactic and semantic annotations (Rule 6 and Rule 7, respectively). Generated semantic annotations are added in the knowledge base $TaKb$ and are checked for consistency with respect to domain ontology in our platform in step B as discussed in section 4.2.2, which is necessary to assure their soundness.

Rule 1 (Recognizing conflicting annotations)

$$\langle tf, R, \textit{sa:Role}\rangle \wedge hasPOS(tf, VI) \rightarrow wrongAnnoOn(tf, A^*)$$

This is a simple case of a rule used to detect wrong annotation on a text-fragment, which states that a text-fragment cannot be annotated as an semantic ontological role and an intransitive verb at the same time. This rule is a good example to demonstrate reasoning with rules, making use of both syntactic and linguistic annotations. The new concept name $wrongAnnoOn(tf, A^*)$ means that there is conflict between the annotations on text-fragment tf either one of them is to be deleted.

Rule 2 (Recognizing semantic annotations) $nn(tf_2, tf_1) \wedge contains2(tf_3, tf_1) \wedge contains2(tf_3, tf_2) \wedge \langle tf_2, A, \textit{sa:Concept}\rangle \rightarrow \langle tf_3, A, \textit{sa:Concept}\rangle$

That is, if the head of an nn annotation has some semantic annotation concept A, then the whole text fragment containing the head and modifier should have the same semantic annotation A.

For example, in sentence "Your summary includes participant mileage", we have syntactic and semantic annotation: $nn(tf_4, tf_5)$ and $\langle tf_5, \textit{AA\_Mileage}, sa\ \textit{:Concept}\rangle$, where $tf_4 =$ and tf_5 are the 4th and 5th word in the sentence, respectively. By Rule 2, we can recognize a new semantic annotation $\langle tf_{45}$, $AA\_Mileage, sa\ \textit{:Concept}\rangle$ with tf_{45} is the 2-word text fragment (participant-4 mileage-5).

Rule 3 (Recognizing semantic annotations) $nn(tf_3, tf_1) \wedge nn(tf_3, tf_2) \wedge \langle tf_{23}, B, \textit{sa:Concept}\rangle \wedge$
$\bigwedge_{i=2,3} contains2(tf_{23}, tf_i) \wedge \bigwedge_{i=1,2,3} contains3(tf_{123}, tf_i) \rightarrow \langle tf_{123}, B, \textit{sa:Concept}\rangle$.

This is a 3-word case extension of Rule 2. That is, for a text fragment $tf_{123} = tf_1 tf_2 tf_3$ with syntactical structure $nn(tf_3, tf_1)$ and $nn(tf_3, tf_2)$, if its sub-text fragment $tf_2 tf_3$ has a concept semantic annotation, then tf should have the same semantic annotation.

This is the case of the sentence: "AAdavantage *flight mileage credit* is determined on the basis of non-stop distances..." with tf_1 =flight, tf_2 =mileage, and tf_3 = credit and semantic annotation $\langle\ tf_2 tf_3$, $Mileage\_Credit$, sa:Concept$\rangle$. By this rule, we can get the text fragment "flight mileage credit" should have a semantic annotation $Mileage\_Credit$.

Rule 4 (Reporting missing semantic annotations)

$$nn(tf_2, tf_1) \wedge \langle tf_1, A, \textit{sa:Concept}\rangle \rightarrow \langle tf_2, U^*, \textit{sa:Concept}\rangle$$

where U^* is a new concept name.

That is, if the modifier of nn has some semantic annotation concept, then the head should be semantically annotated, but which concept is unknown. The underlying idea is that if the modifier carries some meaningful semantic information, so should the head.

An example sentence " AAdvantage flight awards may not be combined with other AAdvantage flight awards" where nn(AAdvantage,awards) and $\langle$ AAdvantage, $AA\_Program, sa\ \textit{:Concept}\rangle$ hold but leaving no semantic annotation for the text fragment "awards". By Rule 4, we can get a report that semantic annotations for "awards" is missing.

Rule 5 (Reporting missing semantic annotations)

$$nn(tf_3, tf_2) \wedge \langle tf_{12}, A, \textit{sa:Concept}\rangle \wedge contains2(tf_{12}, tf_1) \wedge contains2(tf_{12}, tf_2) \rightarrow \langle tf_3, U^*, \textit{sa:Concept}\rangle$$

where U^* is a new concept name.

That is, if the modifier of nn is a part of a 2-word textual unit and whole textual unit has some semantic annotation concept, then the head should be semantically annotated, but which concept is unknown. This is generalizing rule 4 with the same underlying idea is that if the modifier carries some meaningful semantic information, so should the head.

An example sentence " AAdvantage mileage accrual eligibility on airline participant routes is subject to change without notice." where nn(routes, participant) and $\langle$ airline participant, $Airline\_Participant, sa:Concept\rangle$ hold but leaving no semantic annotation for the text fragment "routes". By Rule 5, we can get a report that semantic annotations for "routes" is missing.

Rule 6 (Conflicting linguistic & semantic annotations) $nn(tf_3, tf_1) \wedge nn(tf_3, tf_2) \wedge \langle tf_{12}, A, \textit{sa:Concept}\rangle \wedge$

$$contains2(tf_{12}, tf_1) \wedge contains2(tf_{12}, tf_2) \wedge position(tf_1, p_1) \wedge position(tf_2, p_2) \wedge tf_1 \neq tf_2 \wedge p_1 < p_2$$

$$\rightarrow \textit{isWrongNN}(tf_3, tf_1) \wedge \textit{addNN}(tf_2, tf_1)$$

where isWrongNN and addNN are two new roles to report a wrong nn relation and to suggest a new nn relation between two text fragments, respectively.

That is, for a text fragment like $tf_1 tf_2 tf_3$ if tf_3 is the nn head of tf_1, tf_2, but $tf_{12} = tf_1 tf_2$ is a text fragment with some semantics, then it is highly potential that $nn(tf_3, tf_1)$ is wrong, which should be $nn(tf_2, tf_1)$.

An example is for sentence "Only individual persons eligible for AAdavantage program membership" with annotations $nn(tf_3, tf_1)$, $nn(tf_3, tf_2)$, and $\langle tf_{12}, AA_Program,$sa:Concept$\rangle$, where tf_1 is "AAdvantage", tf_2 is "program", tf_{12} is the text fragment "AAdavantage program" and tf_3 is "membership", we get that "AAdvantage" is modifier of "program" rather than that of "membership".

Rule 7 (Problem between semantic annotations)

$$\langle tf_1, A, \textit{sa:Concept}\rangle \wedge \langle tf_2, B, \textit{sa:Concept}\rangle \wedge tf_1 \neq tf_2 \wedge \bigwedge_{i=1,2} contains(tf_i, tf) \rightarrow \textit{wrongSemPair}(tf_1, tf_2)$$

where wrongSemPair is a new role to report potentially wrong semantic annotation pairs.

Conflicts between two semantic annotations can be detected by this rule in the case that two different text fragments are semantically annotated and share some common sub-fragment. There is no such semantic annotations in our originally annotated corpus. However, consider the example sentence for Rule 4 again. By Rule 2 and the semantic annotation $\langle tf_3, AA\_Membership,$ sa:Concept$\rangle$, we have $\langle tf_{23}, AA\_Membership,$ sa:Concept$\rangle$ which forms a wrong semantic annotation pairs together with the original semantic annotation of tf_{12}.

4.2.1.2 Transformation of semantic annotations for A

SWRL knowledge base is a pair: $\langle Ontology, Rules\rangle$. Reasoning with Rules and Ontology is not decidable in general. To solve this problem SWRL+DL-safe is the approach adopted by Pellet. For simplicity, in our case[3] we do not consider the interaction of ontology and rules while reasoning with logical annotation rules as discussed in section 4.2.1.1. That is we have only a set of rules and facts. By observing the annotation rules we can see that they are plain horn clauses without negation. For each semantic annotation for

[3] To avoid the higher order semantics of SWRL, which is not considered in this thesis

computing satisfiability we perform the following translation on semantic annotations set $AnnoSet$.

For each semantic annotation of the form $\langle tf, ot, at\rangle \in AnnoSet$, we create a binary role such that, $\sigma(at)(\sigma(tf), \sigma(ot)) \in \sigma(AnnoSet)$; such that, the σ transformation satisfies:

$\sigma(\langle tf, ot, at\rangle) = at(tf, \sigma(ot))$ where,

$\sigma(ot)$ is the new individual satisfying $\sigma(ot_1) \neq \sigma(ot_2)$ if $ot_1 \neq ot_2$

With this representation, we will observe that all semantic annotations are binary relations. At the same time as already implicitly clear in the rules itself, all the linguistic annotations are binary relations over the text fragments [more correctly over $\sigma(tf)$]. Thus all the rules are in clausal form of binary relations and exclusion of domain ontology for reasoning with rules, avoids the higher order semantics. Thus, computation of satisfiability for knowledge base with rules same as for the computation of satisfiability with classical semantics following the satisfiability of rules as discussed in section 2.1.4.

Definition 4.9 *$\sigma(TaKb)$ is the σ translated fragment of $TaKb$, in which semantic annotations are obtained by the σ transformation explained above and by excluding the domain ontology. The semantics of $\sigma(TaKb)$ is thus classical. Thus, we can write $\sigma(TaKb) = O_{lang} \cup \sigma(AnnoSet) \cup Rules_{1-7}$, where $Rules_{1-7}$ is the ontology representing the set of rules and $\sigma(AnnoSet)$ =$\sigma(\langle tf, ot, at\rangle) \mid \langle tf, ot, at\rangle \in AnnoSet$.*

4.2.1.3 U^* Revisited

In the section where we discussed rules, in two rules viz. 4 and 5 we have introduced a new concept name U^*, which explicitly means UNKNOWN i.e semantic annotation is missing. From the semantics of SWRL discussed in section 2.1.4, we can observe that any text-fragment satisfying the bodies of rules 4 and 5 can be inferred as $\langle tf, U^*, \text{sa:Concept}\rangle$. But, provided that if the same text-fragments are annotated with some other annotation i.e. $\langle tf, ot, \text{sa:Concept}\rangle$, then such UNKNOWN annotations does not make any sense and must be deleted from the Knowledge Base. Moreover we can say that while inferring with the SWRL rules, we can get a logical closure which includes some redundant/unnecessary annotations, so we have to define an *deletion* operation to delete them. This deletion operation is explained by the following proposition:

Definition 4.10 *Let $\forall tf \in TFs$.If $\langle tf, U^*, \textit{sa:Concept}\rangle \in \beta_u$ then*

$$\langle tf, U^*, \textit{sa:Concept}\rangle \in \beta_u^d \textit{ iff } \exists\langle tf, A, \textit{sa:Concept}\rangle \in \beta_k.$$

where TFs is the set of all text-fragments, β_u are all annotations of the form $\langle tf, U^*, \textit{sa:Concept}\rangle$, β_k are all annotations of the form $\langle tf, A, \textit{sa:Concept}\rangle$.$AnnoSet$ is the set of all semantic annotations; $AnnoSet^{'}$ be the set of all semantic annotations that can be deduced from $AnnoSet \cup Rules_{1-7}$ under the SWRL semantics, where $Rules_{1-7}$ are the set of rules defined in section 4.2.1.1; β_u^d is the set of all annotations of the form $\langle tf, U^*, \textit{sa:Concept}\rangle$ to be deleted from $TaKb^{'}$. Then $\beta_k \subseteq Annoset'$, $\beta_u \subseteq Annoset'$ and $\beta_u^d \subseteq \beta_u$. Note that semantic annotations are obtained by transformation σ, as explained in section 4.2.1.2.

The deletion operation as defined above makes changes in the knowledge base, moreover on the set of semantic annotations i.e $AnnoSet$. Now, we will discuss on consequence of the deletion operation supported by a theorem to show the logical correctness of the operation.

Theorem 4.11 *If a semantic annotation $\alpha^{'}$ is deduced by reasoning over $AnnoSet \cup Rules_{1-7}$ and $\{\alpha_1, ..., \alpha_n\}$ is the proof path for the conclusion $\alpha^{'}$, then if any $\alpha_i \in \beta_u$ occurring in the path is deleted i.e $\alpha_i \in \beta_u^d$ then the conclusion $\alpha^{'}$ is also deleted i.e $\alpha^{'} \in \beta_u^d$.*

The proof path is an ordered set of all the semantic annotations used for generation of a conclusion i.e a new semantic annotation. The semantic annotations occurring in position $n-1$ is the cause for the conclusion at position n.

Example 4.2 *Let us consider the following:*
$TFs = \{tf_1, tf_2, tf_{12}\}$,
$AnnoSet = \{\langle tf_1, A, \textit{sa:Concept}\rangle, \langle tf_2, B, \textit{sa:Concept}\rangle, nn(tf2, tf1)\}$.
Then by reasoning with rules we get
$\{\langle tf_{12}, B, \textit{sa:Concept}\rangle, \langle tf_2, U^*, \textit{sa:Concept}\rangle, \langle tf_{12}, U^*, \textit{sa:Concept}\rangle\} \cup Annoset \in AnnoSet^{'}$
and
$\{\langle tf_2, U^*, \textit{sa:Concept}\rangle, \langle tf_{12}, U^*, \textit{sa:Concept}\rangle\} \in \beta_u$.
Now, let us consider
$\alpha^{'} = \langle tf_{12}, U^*, \textit{sa:Concept}\rangle$ *for which the proof path is* $\{\langle tf_1, A, \textit{sa:Concept}\rangle, \langle tf_2, U^*, \textit{sa:Concept}\rangle\}, \langle tf_{12}, U^*, \textit{sa:Concept}\rangle\}$. *Now by the deletion operation 4.10,* $\langle tf_2, U^*, \textit{sa:Concept}\rangle\} \in \beta_u^d$, *thus it should be true that* $\langle tf_{12}, U^*, \textit{sa:Concept}\rangle \in \beta_u^d$. *This is true, because of the same reason for* $tf2$ *as defined in the deletion operation 4.10.*

Deletion operation and it's consequence is interesting to study and understand. We must make sure that the stability of the model is preserved by the operation, so to handle the case lets analyze the overall system from the perspective of Datalog. Datalog is a declarative logic language in which each formula is a function-free Horn clause, and every variable in the head of a clause must appear in the body of the clause. The following presentation of the syntax and the semantics of disjunctive datalog is based on [33] and [10]. Let Σ be a first-order signature such that (i) the set of function signature $\mathcal{F}(\Sigma)$ contains only constants, and (ii) the set of predicate signature $\mathcal{P}(\Sigma)$ contains $\approx$ which is a special equality predicate with the arity of two. A datalog program with equality P is a finite set of rules of the form

$$A \leftarrow B_1, ..., B_m$$

where $m \geq 0$, and A and Bi are atoms defined over . Furthermore, each rule must be safe; that is, each variable occurring in a head literal must occur in a body literal as well. For a rule r, the set of atoms $head(r) = A$ is called the rule head, whereas the set of atoms $body(r) = \{B_i \mid 1 \leq i \leq m\}$ is called the rule body. A rule with an empty body is called a fact.

The ground instance of P over the Herbrand universe HU of P, written $ground(P, HU)$, is the set of ground rules obtained by replacing all variables in each rule of P with constants from HU in all possible ways. The Herbrand base HB of P is the set of all ground atoms defined over predicates in P. An interpretation M of P is a subset of HB. An interpretation M is a model of P if the following conditions are satisfied: (i) $body(r) \subseteq M$ implies $head(r) \cup M \neq \emptyset$, for each rule $r \in ground(P, HU)$; and (ii) all atoms from M with the $\approx$ predicate yield a congruence relation that is, a relation that is reflexive, symmetric, transitive, and $R(a_1, ..., a_i, ..., a_n) \in M$ and $ai \approx bi \in M$ imply $R(a_1, ..., b_i, ..., a_n) \in M$, for each predicate symbol $R \in P(\Sigma)$.

A model M of P is minimal if no subset of M is a model of P. The semantics of P is defined as the set of all minimal models of P, denoted by $\mathcal{MM}(P)$. Finally, the notion of query answering is defined as follows. A ground literal A is an answer of P, written $P \models A$, if $A \in M$ for $M \in \mathcal{MM}(P)$; First-order entailment coincides with the entailment for positive ground atoms on positive programs.

Definition 4.12 *Given a grounded Datalog* P *and* $M' \subseteq HB(P)$. *We define the restriction of* P *with respect to* M', *denoted* $P|_{M'} \subseteq P$, *by* $P|_{M'} = \{r \mid r \in P, head(r) \cap body(r) \subseteq M'\}$. *We call* M' *is a stable submodel of* P, *if* $M' = \mathcal{MM}(P|_{M'})$.

That is, $P|_{M'}$ is a subset of Datalog rules of P whose body and head atoms are all in M'.

Example 4.3 *Given* $P = \{a, a \rightarrow b, b \rightarrow c\}$, *if we choose* $M1 = \{a, b\}$, $P|_{M1} = \{a, a \rightarrow b\}$. *But if* $M2 = \{a, c\}$, $P|_{M2} = \{a\}$. *We can see that* $M1$ *is a stable submodel of* P *but* $M2$ *is not.*

Note that the set of semantic annotations and linguistic annotations, together with the set of annotation rules defined in section 4.2.1.1 form a Datalog, denoted P_{anno}. Moreover, they are safe with the Herbrand Universe $HU = N_{domn} \cup Individual(N_{lang})$. The deletion operation defined in definition 4.10 makes sense in terms of having the the following theorem hold.

Theorem 4.13 *Suppose M is the minimal model of the annotation Datalog P_{anno}, that is $M = \mathcal{MM}(P_{anno})$. Let M' be the set of ground atoms obtained after the deletion operation and $P' = P_{anno}|_{M'}$. We have $M' = \mathcal{MM}(P')$.*

That is, ours deletion operator can get a M' which is a stable sub-model of the given set of P_{anno}. The proof is provided in appendix A

Example 4.4 *Let $M = \{\langle tf_1, A, \textit{sa:Concept}\rangle$, $s\langle tf_2, B, \textit{sa:Concept}\rangle$, $\langle tf_12, B, \textit{sa:Concept}\rangle$,*
$\langle tf_2, U^*, \textit{sa:Concept}\rangle$, $\langle tf_{12}, U^*, \textit{sa:Concept}\rangle\}$
and
$P = \{\langle tf_1, A, \textit{sa:Concept}\rangle$, $\langle tf_2, B, \textit{sa:Concept}\rangle$,
$nn(tf2, tf1) \wedge \langle tf_1, A, \textit{sa:Concept}\rangle \rightarrow \langle tf_2, U^*, \textit{sa:Concept}\rangle$,
$nn(tf2, tf1) \wedge \langle tf_2, B, \textit{sa:Concept}\rangle \rightarrow \langle tf_{12}, B, \textit{sa:Concept}\rangle$,
$nn(tf2, tf1) \wedge \langle tf_2, U^*, \textit{sa:Concept}\rangle \rightarrow \langle tf_{12}, U^*, \textit{sa:Concept}\rangle\}$.
Now after the deletion operation
$M' = \{\langle tf_1, A, \textit{sa:Concept}\rangle$, $\langle tf_2, B, \textit{sa:Concept}\rangle$ $\langle tf_{12}, B, \textit{sa:Concept}\rangle\}$
and
$P^{'} = \{\langle tf_1, A, \textit{sa:Concept}\rangle$, $\langle tf_2, B, \textit{sa:Concept}\rangle$,
$nn(tf2, tf1) \wedge \langle tf_2, B, \textit{sa:Concept}\rangle \rightarrow \langle tf_{12}, B, \textit{sa:Concept}\rangle$.
Clearly, $M' = \mathcal{MM}(P')$. Hence we can say that the deletion operation, the consequence preserves the stability of the model.

4.2.2 Reasoning with Domain Ontology

Reasoning with Domain ontology describing the domain of consideration is an important aspect of reasoning with annotations. It basically covers reasoning with semantic annotations, because semantic annotations are used to link up the meaning of the textual units with the knowledge in the ontology. One of the important advantage of this reasoning (B in figure 4.1) is checking the correctness of semantic annotations and assert that the knowledge base of annotations together with domain ontology is logically consistent.

As seen from the semantics of our formalism of semantic annotations, its trivial that the union of the ontologies that form the knowledge base $TaKb$ is Higher Order. Since, we are motivated to make use of classical reasoning technique, one obvious challenge is to find a satisfiable logical transformation to classical DL. In the following sections, we deal with such a transformation axioms and discuss on various logical properties of transformed knowledge base.

4.2.2.1 Reduction to Classical DL

In order to compute satisfiability and to reuse with classical reasoners, we propose the transformation of $TaKb$ into classical form, hence-forth denoted as $\phi(TaKb)$. Thus, in order to compute the satisfiability of a text annotation knowledge base TaKb, we will show a reduction from the satisfiability checking of $TaKb$ to the classical satisfiability checking of the transformed DL ontology $\phi(TaKb)$.

Firstly, we will define three injective functions from $N_{domn} \cup Individual(Textfragment)$ to three pairwisely disjoint sets, each disjoint from $N_{domn} \cup N_{lang} \cup N_{type}$ are defined below, where $Individual(Textfragment)$ is the set of instances of concept $Textfragment$ in O_{lang}:

$\mu_o : N_{domn} \cup Individual(Textfragment) \rightarrow S_i^o$;
$\mu_c : N_{domn} \cup Individual(Textfragment) \rightarrow S_i^c$;
$\mu_r : N_{domn} \cup Individual(Textfragment) \rightarrow S_i^r$:

Based on μ_o; μ_c; μ_r, a function λ from $Concept(O_{domn})$ to a set of new DL concepts is inductively defined as follows:

$\lambda(A) = \mu_c(A) for A \in N_{domn}$; $\lambda(\neg C) = \neg\lambda(C)$;
$\lambda(C \sqcup D) = \lambda(C) \sqcup \lambda(D)$; $\lambda(C \sqcap D) = \lambda(C) \sqcap \lambda(D)$;
$\lambda(\forall R.C) = \forall\mu_r(R).\lambda(C)$; $\lambda(\exists R.C) = \exists\mu_r(R).\lambda(C)$;

The underlying idea of the reduction is to rename the ontological elements which are interpretated higher-orderly, according to their positions in ontological axioms. For $TaKb = O_{domn} \cup O_{lang} \cup AnnoSet$, its reduction $\phi(TaKb)$ is defined for each of them as following :

- Reduction of the domain ontology O_{domn}:
  If $C \sqsubseteq D \in O_{domn}, \mu_c(C) \sqsubseteq \mu_c(D) \in \phi(O_{domn})$;
  If $C(a) \in O_{domn}, \mu_c(C)(\mu_o(a)) \in \phi(O_{domn})$;
  If $R(a,b) \in O_{domn}, \mu_r(R)(\mu_o(a), \mu_o(b)) \in \phi(O_{domn})$;
  $\forall C \in Concept(O_{domn}), \mu_c(ot) = \lambda(ot) \in \phi(O_{domn})$.
- Reduction of the domain ontology O_{lang}:
  If $R \sqsubseteq S \in O_{lang}, R \sqsubseteq S \in \phi(O_{lang}$;
  If $C(a), R(a,b) \in O_{lang}$ and $a, b \in Textfragment$
  -then $C(\mu_o(a)) \in \phi(O_{lang}), R(\mu_o(a), \mu_o(b)) \in \phi(O_{lang})$;
  -otherwise, $C(a), R(a,b) \in \phi(O_{lang})$
- Reduction of the $AnnoSet$:
  For an assertion $\langle tf, ot, at\rangle$ of $AnnoSet$
  $\mu_r(at)(tf, \mu_o(ot)) \in \phi(AnnoSet)$ and
  $\mu_o(ot) = \mu_o(tf) \in \phi(AnnoSet)$ if $at = sa$:$Individual$;
  $\mu_c(ot)(\mu_o(tf)) \in \phi(AnnoSet)$ if $at = sa$:$Ind - Con$;
  $\mu_c(tf) \subseteq \mu_c(ot) \in \phi(AnnoSet)$ if $at = sa$:$Concept$;
  $\mu_r(tf) \subseteq \mu_r(ot) \in \phi(AnnoSet)$ if $at = sa$:$Role$.

Proposition 4.14 *Given a text annotation knowledge base $TaKb$,$\phi(TaKb)$ is a $\mathcal{DLH}$ ontology if the domain ontologies are in $\mathcal{DL}$ and the size of $\phi(TaKb)$ is linear in that of $TaKb$, where $\mathcal{DLH}$ is the language containing constructors from $\mathcal{DL}$ plus role hierarchy.*

Theorem 4.15 *Given a text annotation knowledge base $TaKb$, $TaKb$ is satisfiable if and only if $\phi(TaKb)$ is satisfiable under classical DL semantics.*

Theorem 4.16 *Given a text annotation knowledge base $TaKb$, $TaKb$ is coherent if and only if $\phi(TaKb)$ is coherent under classical DL semantics.*

Theorem 4.17 *Given a text annotation knowledge base $TaKb$ and the domain ontology is in $\mathcal{ALC}$, then checking the satisfiability of $TaKb$ is EXPTIME-hard.*

4.2.2.2 Entailment of semantic annotations

Entailment is an interesting feature to be considered. Let us assume that the domain ontology has the subset relationship between two classes i.e. $C \sqsubseteq D$. Now if a text-fragment tf is annotated by C, then we must be able to imply that tf is annotated by D.

For the purpose of deducing the entailment we will make use of the QCL introduced in [22] and briefly discussed in section 2.1.3. Based on the weak satisfiability semantics of semantic annotations as explained in definition 4.6, here we propose the definition of entailment of semantic annotations.

Definition 4.18 *An semantic annotation $TA = \langle tf, A, sa{:}Concept\rangle$ is entailed by $TaKb$, written as $TaKb \models_e TA$ iff for any model I such that $I \models_s TaKb$ implies $I \models_w TA$.*

Now we give the translation for semantic annotations based on the weaker semantics to compute the entailment.

- Reduction of the $AnnoSet$:
  For an assertion $\langle tf, ot, at\rangle$ of $AnnoSet$
  $\mu_o(ot) = \mu_o(tf) \in \phi(AnnoSet)$ if $at = sa$:$Individual$;

$\mu_c(ot)(\mu_o(tf)) \in \phi(AnnoSet)$ if $at = sa$:$Ind - Con$;
$\mu_c(tf) \subseteq \mu_c(ot) \in \phi(AnnoSet)$ if $at = sa$:$Concept$;
$\mu_r(tf) \subseteq \mu_r(ot) \in \phi(AnnoSet)$ if $at = sa$:$Role$.

Definition 4.19 *An $\mathcal{ALCH}$ axiom A is entailed by $TaKb$, written as $TaKb \models_e A$ iff $TaKb \models_s A$*

Based on the definition of entailment and the translation function we propose the following theorem (proof is provided in appendix A)

Theorem 4.20 *Suppose $TaKb$ is the knowledge base satisfying $TaKb \models_e C \sqsubseteq D$ and TaKb $\models_e \langle tf, C, sa{:}Concept\rangle$, then we have $TaKb \models_e \langle tf, D, sa{:}Concept\rangle$.*

4.3 Summary

In this chapter, we have presented the formal syntax of the solution for knowledge representation of the text annotations: syntax, semantics, reasoning tasks and computations algorithms. The semantic annotations are expressed as triples, but semantically interpreted in different way (strong satisfiability, weak satisfiability and classical interpretation) for various different reasoning tasks (inconsistency checking of semantic annotations with respect to domain ontology, entailment of semantic annotations or deducing new annotations by logical annotation rules). Two important paradigms for reasoning with annotations are discussed, both of which provide interesting reasoning service for text annotation systems. Since, the annotations of texts are important information to access the content of the texts, our knowledge representation system will facilitate the semantic access to the texts.Understanding of this chapter is important for the next chapter, which provides the technical description of the prototype model. Thus, by the end of this chapter its assumed that the user is familiar with syntax and semantics of semantic annotations, linguistic annotations, rules under considerations and the algorithms used for interpreting the annotations for both the paradigm of the reasoning with annotations.

Chapter 5

Technology Application

"Logic takes care of itself; all we have to do is to look and see how it does it"
Ludwig Wittgenstein

This chapter describes the prototype model, that is developed in compliance to the theoretical background that has been developed in the previous chapters. The implementation of the prototype is supported by the availability of various semantic web implementations, APIs, tools to deal with annotations, OWL and DL-reasoners. The prototype handles all the issues related to knowledge representation, reasoning paradigms and integration of various knowledge sources. During development, we have followed as closely as possible the software development processes and principles. In the section that follows we will start to analyze the application from the software engineering perspective and then provide solution to each of the requirement. Next sections are the broader explanation of the system developed. The chapter concludes with the rough statistical analysis of the achievements made on the improvement of the annotation knowledge, when operated with text-corpus provided by American Airlines. The synopsis provided a summary of our experience of implementing the system and some useful and reusable achievements as the product of the implementation.

5.1 Requirements

As mentioned before our major goal is to develop a system, where reasoning is done over text corpus, annotated with different linguistic and semantic knowledge. We were motivated to achieve this with existing semantic web technologies and already proven highly optimized classical reasoners. So, the prototype application developed must fulfill following requirements:

5.1.1 System features

- *Integration*: Given a text corpus with linguistic and semantic annotations, the system must consider both of them for reasoning. At the same time provide an interface to use other form of annotations provided they are given in some standard formats. This feature accomplishes the theoretical foundation explained in section 3.3.

- *Semantic Annotation*: Represent and use semantic annotations in a format usable for reasoning without loosing the meaning of different types of semantic annotations. Its to implement the knowledge representation formalism discussed in section 4.1.1 and to implementations of algorithms discussed in 4.2 for reasoning with different paradigms.
- *Rule language integration*: The system should be integrated with some rule language *eg.* SWRL to allow the user to write the relationship between text-fragments based on some semantic/syntactic pattern. We have integrated the rules discussed in section 4.2.1.1, for reasoning paradigm as explained in section 4.2.1.
- *Consistency and Coherence Check*: Given a knowledge base, the system must report and explain to the user regarding inconsistencies and incoherences. This will allow the user to detect the consistency of the knowledge base. It is to accomplish the reasoning with respect to domain ontology as discussed in section 4.2.2.

5.1.2 External interface requirements

- *User interface*: Though the system does not require an implicit user interface, but still a way to view the annotated corpus would be handy. With this the user can make correction based on the results of reasoning services.
- *Reporting tool*: This will help the user to make statistical analysis based on the input annotations and inferred annotations after reasoning. The usefulness of different syntactic patterns can be compared.
- *Suggestions and Explanations Interface*: The causes of inconsistencies and incoherences must be explained, so that the user can either make correction on the annotations or on domain ontology, whichever is suitable. At the same time in addition to new inferred annotations, mismatching and missing semantic/syntactic annotations are to be suggested.

5.2 System Description

In this section we will discuss about the system architecture, application flow and the different individual components of the overall system. The system is developed using Java programming languages making use of a number of APIs for various semantic web technologies as discusses in section 2.2. We have followed to our best to make a implicit separation between the data layer (corpus and annotations) and the reasoning layer, which will make system flexible to changes in data sources, reasoners and specific implementations.

5.2.1 Architecture

The figure 5.1 shows the layered architecture of the application, where three distinct layers viz inputs, processing and outputs are depicted with various components of each layer. Each of the different modules of the processing layer are loosely coupled with each other, depicted as independent components. In software: coupling, is the degree to which each program module relies on each other module, for details see [44], which discusses more on loose coupling and it's advantages. Data Transfer Objects (DTOs) are used to pass information between each modules, which will eventually make the application more flexible to change in data sources. In this section we will discuss the technical aspect of each of the components. We will illustrate each of the sections with algorithms and examples as necessary and it is to be noted that some of the algorithms are specific to our implementations and requirements, which can be modified as necessary. The algorithms thus give insight to understanding the prototype model developed.

5.2.1.1 Inputs

Since, the overall system involves a various forms of knowledge, obtained from different sources and in different formats, a thorough understanding of input parameters is important. In this section, we will discuss about the inputs to the system.

Text corpus: It provides the collection of sentences in the documents in XML format. Each XML node has the information regarding the document number where the sentence occurs and its position in that document, along with the contents of the sentence.

Example 5.1

```
<SENTENCE ID="sent0#30" docoffset="4695">
AAdvantage mileage accrual eligibility on airline
participant routes is subject to change without notice.
</SENTENCE>
```

Each sentence is parsed using Stanford Natural Language Parser Wrapper [1] to generate the text-fragments with their defining properties like offsets.

Semantic Annotations: These are the collection of RDF triplets, where each annotation is assigned to text-link. The semantic annotations taken as input are generated by LIPN annotator developed at LIPN[2].

Example 5.2

```
<TextLink rdf:ID="lc122">
<contentInSentence rdf:resource="#sent0-30"/>
<start_offset>0</start_offset>
<end_offset>18</end_offset>
<isSourceConcept rdf:resource="#AAdvantageMileage"/>
</TextLink>
```

Hench-forth we will call the collection of these text-links with semantic annotations as *formatA*.

Linguistic Annotations: Linguistic annotations are provided in XML format and represents the syntactic relationships between words in sentences. Such annotations which are inputs for our system are outputs of Stanford CoreNLP tools.

Example 5.3

```
<s id="31">
<dependencies style="typed">
  ..................................
  <dep type="nn">
    <governor idx="4">eligibility</governor>
    <dependent idx="2">mileage</dependent>
  </dep>
  ...................................
</dependencies>
</s>
```

[1] http://nlp.stanford.edu/software/lex-parser.shtml
[2] http://www-lipn.univ-paris13.fr/recherche/?lang=uk

Rules: Rules that are developed by analysis of the corpus and linguistic studies are written in Semantic Web Rule Language (SWRL) [3]. The rules are stored as OWL file, written following the RDF Concrete Syntax and is illustrated in Appendix B.

Example 5.4

$$contains2(?y, ?w), contains2(?y, ?x), isSourceConcept(?x, ?z), nn(?x, ?w) \rightarrow isSourceConcept(?y, ?z)$$

Domain Ontology: This is one of the most important input for reasoning with respect to the domain information concerning the text corpus over which reasoning is performed. Domain ontology describes the concepts and their relationships, based on which the semantic annotations are made. The ontology we have taken into consideration is an ontology developed at LIPN, based on the regulatory texts provided by American Airlines, on air-travel loyalty program.

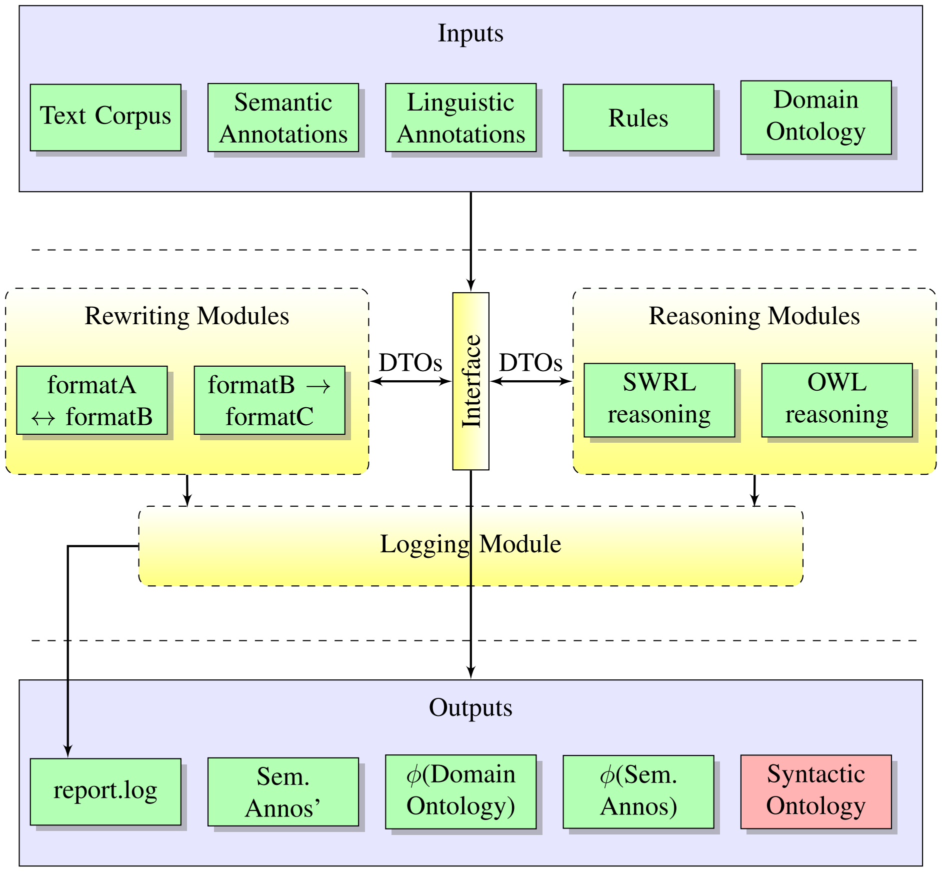

Figure 5.1: System Architecrure

5.2.1.2 Interface

This module acts as the bridge between all the other modules and is the point of communication between each other. Inputs are feed to the system through this point and outputs are generated by this module. This

[3]http://www.w3.org/Submission/SWRL/

makes the system de-coupled with reasoning and re-writing modules. At the same time each of the reasoning and re-writing modules are de-coupled with each other, thus avoiding the changes in each other other, if some implementation is changed. For instance if some other useful reasoning paradigm is discovered, then it can be integrated to the system, with only minor changes. At, the same time if further analysis of the work, reveals that different reasoners can work better with our identified paradigm of reasoning, then any of the module can be re-implemented, without effecting the others. For now both the reasoning paradigms are implemented with Pellet.

5.2.1.3 Re-Writing Module

Re-writing from one format to another is an important aspect of the application. The inputs for the system are in XML/RDF formats, which are processed by the rewriting modules and re-written into OWL in a format proposed by our formalism as discussed in section 4.1. The re-writing modules are implemented using different APIs like JENA, OWLAPI as discussed in 2.2 and utility functions provided by StanfordNLP API[4] and JAXP [5].

$formatA \leftrightarrow formatB$ module is used to write back and forth the semantic annotations in the text-link notations (RDF format obtained from LIPN annotator) and text-fragment notations as three valued tuples proposed in the formalism which is defined in definition 4.1. This module also implements methods to parse the input documents and generates text-fragments, corresponding to single or multi-word textual units.

An important rewriting is for the representation of semantic annotations, where we rewrite Text-Links as Text-Fragments. This procedure depicted by the process $formatA \rightarrow formatB$ in our notation is achieved with the algorithm 1.

[4]http://nlp.stanford.edu/software/corenlp.shtml
[5]http://java.sun.com/xml/downloads/jaxp.html

Algorithm 1: Algorithm to convert Text Links to Text Fragments ($formatA \rightarrow formatB$)

```
Input: Text Links $TL\_RDF$ //in RDF
Input: Text Links $sentence$
// the sentence being processed Output: Text Fragments $TF\_OWL$ //in OWL
Collection⟨$TextFragment$⟩ $TFs$ ;
Collection⟨$TextFragment$⟩ $TF\_all$ ;
foreach Node $ND \in TL\_RDF$ do
    create new Text Fragment $TF$;
    $TF$.offsets=$ND$.offsets ;
    $TF$.contentInSentence=$sentence$.number;
    TextFragment $tempTF$=getSameTF($TFs$,$TF$) ;
    if $tempTF$== NULL then
        $TF$.setSemanticAnnotation($ND$.getSemanticAnnotation()) ;
        $TF$.setID(RANDOM_ID) ;
        $TFs$.add($TF$);
    end
    else
        $tempTF$.addSemanticAnnotation($ND$.getSemanticAnnotation()) ;
    end
end
$TF\_all$=generateTextFragments($S$) ; // utility function as described below  foreach $TF \in TFs$ do
    TextFragment $tempTF$=getSameTF($TF\_all$,$TF$) ;
    if $tempTF \neq$ NULL then
        $TF$.ID=$tempTF$.ID ;
        $TS\_all$.remove($tempTF$);
        $TS\_all$.add($TF$);
    end
end
foreach $TF \in TF\_all$ do
    Write TextFragment ($TF$) as OWL Individual, with respective data properties ;
end
```

In the above algorithm, we have seen a function $generateTextFragments(S)$, which is an utility implementation, used to generate text-fragments form a given sentence. Our implementation considers 1-word, 2-word and 3-word text-fragments. To illustrate this, let us consider a fragment of the sample sentence *AAdvantage mileage accrual.* If this sentence is taken as input then three 1-word text-fragments corresponding to *AAdvantage*, *mileage*, *accrual* and two 2-word text-fragments corresponding to *AAdvantage mileage*, *mileage accrual* and one 3-word text-fragment corresponding to *AAdvantage mileage accrual* are created. The offsets for each words are obtained by using Stanford NLP API to maintain standard used by other tools, rather than providing our own implementation. The unique id for each text-fragment is generated by following this expression *TFF+position+_+sentenceNumber* for 1-word text-fragments, while *TFF+position+_+sentenceNumber+_+TO+_+position+_+sentenceNumber* for multi-word text-fragments. So, for the sentence that we have been considering and the text-link of the example 5.2, after the $formatA \rightarrow formatB$ translation we get

$$\langle TFF1\_0-30\_TO\_2\_0-30, AAdvantageMileage, \text{sa:Concept}\rangle$$

and a number of text-fragments generated as just explained above.

A part of the same module, writes linguistic annotations obtained in XML format where relationships are between words into OWL with relationships between text-fragments. Linguistic annotations are used for reasoning with rules, thus need to be rewritten in OWL format. Syntactic Ontology [red box, in o/p block] is the temporary file generated by this algorithm. OWL file representing linguistic annotations in each sentences are generated at run time, and are deleted once reasoning with rules is completed, because we do not use linguistic annotations for reasoning with domain ontology.

```
Algorithm 2: Algorithm to write Linguistic annotations as OWL
  Input: Stanford dependencies (grammatical relations between words) $Dep\_XML$ //in XML
  Input: Selected Sentence Number $S\_No$
  Output: Linguistic Annotation $LA\_OWL$ //in OWL
  foreach Node $ND \in Dep\_XML$ st $ND$.type='S' and $ND.id = S\_No$ do
      Nodes $nds$= all Nodes of type 'Dep' ;
      foreach Node $nd \in nds$ do
          Individual indv1= OWLIndividual($nd$.childnode s.t. type='governor') ;
          Individual indv2= OWLIndividual($nd$.childnode s.t. type='dependent') ;
          ObjectProperty role=OWLObjectProperty($nd$.getValueOfAttribute('type')) ;
          Relation rel= role(indv1,indv2) ;
          //Write individuals, role and relation to OWL
      end
  end
```

After re-writing the linguistic annotations by the above algorithm for the example 5.3, we get an entry in the OWL file, storing linguistic annotations as:

Example 5.5

```
<owl:NamedIndividual rdf:about="#TFF4_0-30">
<nn rdf:resource="#TFF4_0-30"/>
</owl:NamedIndividual>
```

Note: *All the rewritten outputs of $A \rightarrow B$ are used for reasoning using pellet with integration of SWRL rules.*

$B \rightarrow C$ module implements the higher order to classical logic rewriting, to in-cooperate the necessity for representation of semantic annotations in classically reasonable format. This module implements the algorithm stated in section 4.2.2.1 to rewrite the tuple representation of semantic annotations and domain ontology (the combination of both of which is the Knowledge Base for us), such that union of which is neither higher order nor misses any semantic information.

5.2.1.4 Reasoning Module

This module performs all the reasoning tasks, by making use of Pellet. One of the most important task to be performed by this module is to check the consistency and coherency of the knowledge base of the system. Pellet provided direct interface to check the consistency and explain the axioms necessary for inconsistency. The algorithm below shows the implementation to check the coherency. This algorithm is based on the definition 4.7 and supported by theorem 4.16.

```
Algorithm 3: Algorithm for checking incoherency
  Input: domain knowledge base $\phi(TaKb)$
  Output: Inconsistent Classes with Explanation
  foreach $C \in \phi(TaKb)$ do
      boolean sat: satisfiable(C);
      if !sat then
          Get explanation
      end
  end
```

This algorithm is used because Pellet API does not provide an implementation for checking in-coherency. This algorithm thus makes use of the implementation of Pellet API to check the satisfiability of a class in the knowledge base to check the coherency. An example of the case when we get the in-coherency is explained with example 3.1.

Reasoning with SWRL rules is an important aspect of our application. Pellet provided interface to reason with DL-safe SWRL rules as discussed in section 4.2.1. While reasoning with rules, a number of new semantic annotations are generated, which need to be re-injected into the knowledge base as depicted in figure 4.1, so that such new semantic annotations are also used for reasoning with respect to domain ontology. But, for reasoning with rules, the semantic annotation are treated as binary relations, with the annotation type as the name of the role. So, thus obtained semantic annotations are to be re-interpreted as the tuple notation, so that they can be re-injected into the $AnnoSet$. The algorithm below explains how such semantic annotations represented as binary relations are re-interpreted as semantic annotation tuple as introduced by our knowledge representation formalism.

Algorithm 4: Algorithm to interpret inferred relations as semantic annotations

```
Input: Inferred OWL axioms $Axioms$
Output: Set of semantic annotations $Set < SemAnnoTuple > semAnnos$
SemAnnoTuple temp ;
foreach $Axiom \in Axioms$ do
    if Axiom is of type ObjectPropertyAssertion then
        if ObjecProperty $\notin specialObjectProperties$ then
            temp= $\langle getTFIndividual(Axiom), getOntologyTerm(Axiom), ObjectPRoperty \rangle$ ;
            semAnnos.add (temp);
        end
    end
end

getTFIndividual(Axiom){ foreach Individual $\in Axiom$ do
    if Individual.contains("TFF_") then
        return Individual ;
    end
end
}
getOntologyTerm(Axiom) { foreach Individual $\in Axiom$ do
    if !Individual.contains("TFF_") then
        return Individual ;
    end
end
}
```

The algorithm above is thus used to interpret the result of reasoning with SWRL, where we get the inferred semantic annotations as relations as already explained in section 4.2.1.2. Note that by *specialObjectProperties* we mean all the specifically defined object-properties in writing SWRL rules for our implementation like *isWrongNN,addNN, wrongSemPair etc*. The function to obtain individuals representing text-fragment and OntologyTerm are *Very Specific* to our implementation.

5.2.1.5 Logging Module

The application generates various types of information at different points of executions, which is necessary to be logged for the purpose of usability. Reasoning module and Rewriting module make use of a central logging module implementing log4j [6] to write into log files. One basic log are all the text-links with mismatching offset values, which the user can use to make the necessary corrections, an important finding of such a log is the presence of semantic annotation on more than 3-word text-links, which are ignored for reasoning with rules in our current implementation. At the same time, system logs the statistics regarding the linguistic/semantic annotations based on inputs and outputs, to be used for analyzing the system and detecting more useful syntactic patterns. Reasoning module makes use of logging module to log all the generated semantic annotations including un-necessary annotations (which are eventually deleted) and warn the user about the wrong/suggested linguistic annotations. So, this log is a useful part of the output to be used by user to see the output annotations. Also, note that logging is also useful, because the prototype lacks an interactive User Interface. One example of such a log after reasoning with rules is:

Example 5.6

```
<TFF6_0-30_TO_8_0-30,isSourceConcept,UNKNOWN>
<TFF8_0-30,isSourceConcept,UNKNOWN>
<TFF3_0-30_TO_4_0-30,isSourceConcept,UNKNOWN>
<TFF2_0-30_TO_4_0-30,isSourceConcept,Eligibility>
<TFF3_0-30_TO_4_0-30,isSourceConcept,Eligibility>
--Removed----<TFF3_0-30_TO_4_0-30,isSourceConcept,UNKNOWN>
<TFF7_0-30_TO_8_0-30,isSourceConcept,UNKNOWN>
****************WRONG ANNOTATIONS WARNINGS****************
< TFF6_0-30_TO_7_0-30 , Airline_Participant> OR < TFF7_0-30_TO_8_0-30 , UNKNOWN>
***************WRONG NN WARNINGS****************
Remove nn(TFF8_0-30,TFF6_0-30) and Add nn(TFF7_0-30,TFF6_0-30)
```

5.2.1.6 Outputs

All the modules write the outputs for the system through interface (except logs). Syntactic Annotation ontology is temporary output, generated for reasoning with rules and is deleted once the SWRL reasoning is completed. For each of the sentences temporary OWL files for both syntactic and semantic annotations are generated, which are deleted after SWRL reasoning is complete. *Sem. Annos'* is the RDF file (obtained by $formatB \rightarrow formatA$ re-writing operation) with all the semantic annotations (original + re-injected annotations obtained by reasoning with rules) re-written text-link format, which will be used by RDFa processing tool [7] to display original document in HTML with semantic annotations. $\phi(sem.annos')$ and $\phi(domainontology)$ are the outputs of $formatB \rightarrow formatC$ modules, which are eventually used by the OWL reasoning module, for our case of reasoning with respect to domain ontology.

5.2.2 Operation

As seen from the preceding section, prototype application is composed of number of modules, each of them are to achieve different functionalities. In this section, we will explain the operation of the overall application and sequence of executions, along with different outputs generated at different points. The figure 5.2 shows the application flow and different types of annotations taken into consideration in the prototype. Green blocks are RDFs, black blocks are XMLs, blue blocks are OWL files (among which some are generated at runtime) and red blocks are the processing modules. The system operates in the sequence as per to meet the requirements better explained by the figures 3.2 and 4.1 in the previous theoretical chapters.

First step is to process the input documents, with the Annotation Rewriters, which is the implementation of Rewriter Module as described in 5.2.1. Re-writing is an important phase, because this initiates the task of integrating various forms of knowledge, by writing them in the format suitable for reasoning. The outputs of this phase are FormatB: are the tuple representation of semantic annotations, with probable Text-Fragments produces by implementing the algorithms 1 and all text fragments generated as described after the agorithm,

[6] http://logging.apache.org/log4j/1.2/
[7] developed by other group in LIPN

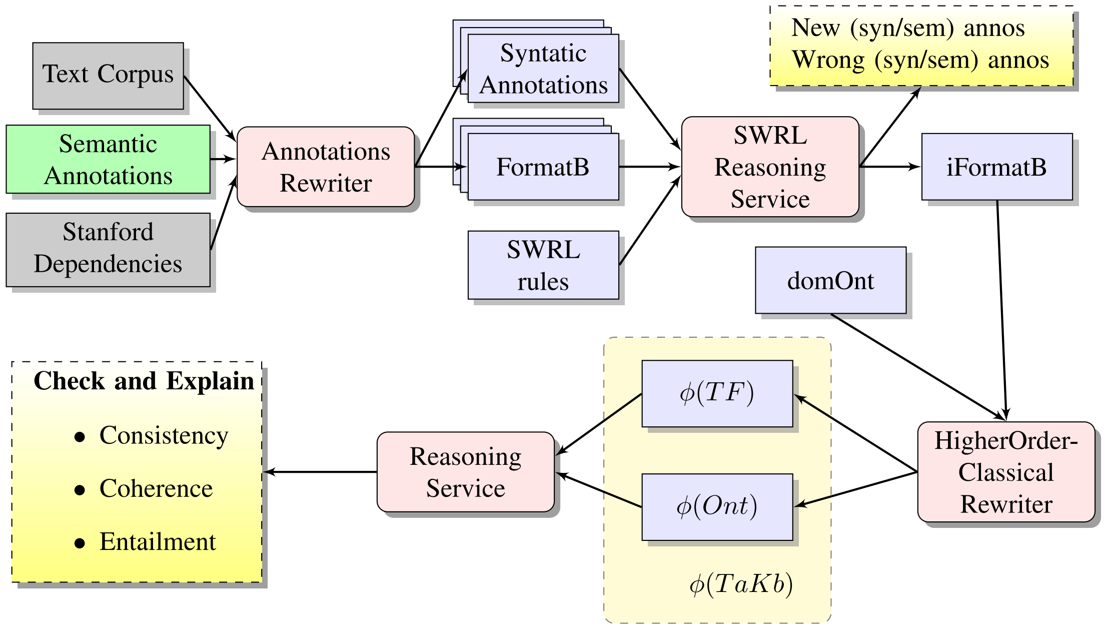

Figure 5.2: System Operation

Syntatic Annotations: are the OWL representation of linguistic annotations obtained by implementation of the algorithm 2. These outputs are now in the standard format usable for reasoning with rules.

SWRL reasoning service makes use of the outputs of the preceding process in combination of SWRL rules as inputs. Before the semantic annotations are used for reasoning, σ transformation (not shown in figure, because of space) as described in section 4.2.1.2 is performed over *formatB*, so that semantic annotations are re-written in OWL format. This module implements all the reasoning and interpreting algorithms described in subsection reasoning module of section 5.2.1. Note that while reasoning on this stage we don't consider domain ontology and also observe that SWRL reasoning precedes the reasoning with respect to domain ontology. The, major output of this section, iFormatB is the tuple representation of semantic annotations, which contains all the Text-Fragments, with their semantic annotation(s) including the ones that have been generated by reasoning. While, re-injecting semantic annotations we make use of the algorithm 4, to make sure to not to miss out the actual intent of the inferred annotations. At the same time, at this stage, we log all the linguistic annotations and the conflicts between semantic/linguistic annotations as warning message to the user, which can be used for correction of annotations, with human innervation.

The next process that follows is one important implementation of the formalism that we have proposed. The 'Higher Order-Classical Rewriter' implements the formalism described in section 4.2.2.1. At this section we take care of the higher order semantics of the knowledge base caused by the inclusion of domain ontology and translate the knowledge base into classically reasonable format[8]. The output of this process is used by the reasoning service to perform classical reasoning to check the consistency and coherency (described in algorithm 3) of the semantic annotations in the documents over the defined domain ontology. The service also provides the explanations for the causes, whenever necessary. Note that the final annotation are checked for correctness without making use of the golden standard. The outputs for some of the cases are discussed in Appendix B, for further understanding.

[8]knowledge base is as $\phi(TaKB)$ as defined in 4.2

5.2.3 Optimization

While existing techniques for TBox reasoning (i.e., reasoning about concepts) seem able to cope with real world ontologies [21], but often existing techniques for ABox reasoning (i.e., reasoning about individuals) cannot cope with realistic sets of instance data. This difficulty arises not so much from the computational complexity of ABox reasoning, but from the fact that the number of individuals (e.g., annotations) might be extremely large [20]. Even though Pellet reasoner, that we have used for reasoning, makes use of various optimizations techniques, as described in [43], we faced issues regarding space when using real time corpus.

Our original corpus consists of 245 sentences, 727 semantic annotations and 843 syntactic annotations and the domain ontology is moderately big and complex. On considering the whole of corpus at the same time about 50k text fragments are generated. For each sentence with *n* words, we get *n+(n-1)+(n-2)* text-fragments in account for 1-word,2-word and 3-word text-fragments respectively. On the first run of application, in an environment of Intel Core 2 Duo 2GHZ processor, 3GB main memory and 1GB JVM heap size, the system threw java.lang.OutOfMemoryError: Java heap space [9] error. So, to handle this case for reasoning with large ABox, we have optimized the system without altering the expected results.

The Syntactic annotations that we have considered, depict the relationship between textual units in the same sentence. The SWRL rules are used to discover new semantic and syntactic annotations, based on such relationships and existing semantic annotations. It is thus obvious that processing each sentence at one time and reasoning with annotations on only one sentence at a time does not make any change in the final results. The one optimization that we obtained is by reasoning with annotations within chunks, each chunk corresponding to a sentence, rather than the whole corpus at the same time. This solution not only solved the memory issue, but also helped in generating organized outputs, which can be easily analyzed.

5.3 The Results

For the prototype evaluation, we have taken into consideration one syntactic relation *nn* (noun compound modifier). The justification of choosing this type is as explained in [29], which states among 727 semantic annotations we have on the corpus, 53% contain text fragments with at least one *nn* annotation. Whilst, among 843 syntactic annotation nn on the corpus, 64% have at least one semantic annotation associated to their text fragments. Considering that there are 48 syntactic dependency types defined in [7], these numbers mean that the overlap between semantic annotations and nn annotations is remarkable. Thus the evaluation of the system is performed based on the SWRL rules explained in section 4.2.1.1, the cause of such rule pattern is also clear by the table 5.1 taken from [29].

Type of nn	Number
head	186
modifiers only but no head	134
two modifiers together	49
head and one modifier together	10
none	235

Table 5.1: Coverage of Rules on the Annotated Corpus, where *type of nn* is divided based on whether its head or its modifiers are semantically annotated

[9]http://java.sun.com/developer/technicalArticles/J2SE/monitoring/

With the considered corpus with semantic annotations and syntactic annotations (of type *nn*) and 6 SWRL rules, we found the following results:

Total Sentences	245
I/P Semantic Annotations	727
I/P Syntactic Annotations	843
I/P rules	6
Total Execution Time	11mins,06secs
Inferred Semantic Annotations	717
Inferred missing semantic annotations(UNKNOWNS)	544
Unused missing semantic annotations(Deleted UNKNOWNS)	51
Suggested wrong semantic annotations	38
Suggested syntactic annotations to be removed and added	32

Table 5.2: Statistical analysis of outputs

At the same time, we made some experiments with checking the correctness of the semantic annotations. The domain ontology that we have taken into considerations does not have any $disjoint$ properties. Thus, we added two dummy classes A and B and defined that $disjoint(A, B)$. Then by annotating the textual units in the corpus with these new classes, we checked for the working of application to find the consistency and coherency of the corpus. If we have the annotation such that $\langle tf, A, \text{sa:Concept}\rangle$ and $\langle tf, B, \text{sa:Concept}\rangle$ then we will observe that the system will warn us with the message that the knowledge-base is in-coherent. In case if we have $\langle tf, A, \text{sa:Ind-Con}\rangle$ and $\langle tf, B, \text{sa:Ind-Con}\rangle$ we will be informed that the knowledge base is in-consistent. This result is as per required, which can not be detected in the original semantic annotations, taken as input. Both these cases are explained further with example in appendix B.

5.4 Synopsis

This chapter, thus explains the problem form the perspective of the implementation. The prototype model, clearly demonstrates that the proposed formalism is well supported by the existing technology. But, development of the application was not a straight-forward task to be achieved. Designing of re-writing models, required extensive studies of the different independent formats obtained as input from various sources. Due, to the difference in the input source, different APIs, utility packages had to be used along with writing our own utility packages (distributable as independent packages), accessible for further work. Pellet and OWLAPI provide different interfaces to reason with different paradigm and each have their own cons and pros. We, had to use both the interfaces, mainly because our Knowledge Base is union of various ontologies. Pellet interface does not provide explanation service for inconsistency and unsatisfiability, when reasoning involves more than on ontologies, thus we had to use OWLAPI for reasoning with domain ontology. While, interface provided by Pellet API, is optimized to work with SWRL rules, because of which we have used this for this paradigm. Working with ontology with complex structures, specially when we need to consider pretty small details, like implementing translation discussed in section 4.2.2.1 with OWLAPI was an interesting challenge. A number of utility classes developed for traversing ontology in an easier way are useful distributable obtained during the implementation.

By the end of this chapter, we have presented our research from both the theoretical and implementation perspective. We have addressed all the problems and issues behind the motivation to achieve an useful technology to work with annotated corpus. In the last chapter that follows, we will conclude the overall work by summarizing the current state of the work and some suggestions for the research work that can be undertaken to work further in this field.

Chapter 6

Epilogue

"Logic is the beginning of wisdom, not the end."
Leonard Nimoy

As we arrive at this point, starting from the motivation for undertaking the research, we have explained the solution that we have proposed to solve the issues regarding reasoning with text annotations. Even though, we have worked on wider perspective of reasoning with text annotations, there are still issues to be addressed and further researches to be undertaken. Before we wrap up everything, in this chapter we precisely mention the state of the mechanism proposed, the doors opened by this work for further interesting work that opens up new challenges and conclude with our observations.

6.1 Current State

We have formally defined an approach for reasoning with text annotations obtained from various kinds of background knowledge. Moreover, semantics preserving formalism for representation of semantic annotations is motivated, proposed and usability of which is demonstrated with a working prototype. The current version of the prototype works well only with the case when the input semantic annotations are of simple types. It is because, the LIPN semantic annotator, that is used to generate semantic annotations, produces semantic annotations of simple types only. Even though the implementation for composite annotations is provided, there is no standard representation of the composite annotations in the input format, which made it harder for us to develop test cases to check the working with composite annotation types. But it works fine with the composite annotations which are generated by SWRL rules, because at this point composite annotations are represented in tuple format as provided by our formalism.

In the current state of the research we have thus provided two scenarios to illustrate the reasoning mechanism over annotated texts: One is aimed to reason the annotations with rule language by excluding the domain ontology to deduce extra annotations, missing annotations and in-correct annotations; The other is an effort to make use of classical DL formalism to provide the a metric to evaluate the annotation correctness without golden-standards. The formalism is accompanied by the algorithms to translate the knowledge base in higher order into classical satisfiability checking, so that the state of art ontology reasoners can be reused for reasoning. All the proposed knowledge representation formalism, algorithms and knowledge integration approach is demonstrated with the fully functional prototype model, whose output can be usefully used as input for other applications that work with annotated text corpus providing an exciting platform for interactions between textual and ontological information.

6.2 Future Works

The current state of work, provides a strong foundation for reasoning with text annotations, obtained from various sources. With the help of the prototype model, we have demonstrated that the proposed formalism is well supported by the existing technologies. In this section, we will discuss on some major tasks that can be under taken as further research in this field.

6.2.1 Composite Annotations

Composite annotations are important types of annotations, which can enrich the information contained in an annotated corpus. We have defined composite annotations in section 3.4.1 and have explained with examples. Annotating sentences or large group of words by axioms can be really useful for further understanding and interpretation of the text-corpus. But, we have excluded the formalism related to composite annotations, due to the time constraint and incompleteness of the input annotated corpus. This can be one challenging task to be undertaken. During the research we also came up with various types of rules, that make use of composite annotations, which can help us to deduce new annotations. Moreover introduction of composite annotations can help us to compose the rules as adopted in SSBVR through the distinction between fact type and fact type form. One of the major issues on composite annotations is the representation format to be used by the annotators, which opens up a new task for LIPN annotator to consider annotations of composite annotations.

6.2.2 Re-injection Architectures

We have used a naive approach to re-inject only semantic annotations discovered by reasoning with rules, and use them for reasoning with respect to domain ontology, which is depicted in figure 4.1. Further study on re-usability of the added annotations to improve results of other tools can be interesting field to work on. Richer treatment architectures become possible if new or problematic annotations can be re-injected in some tools, possibly interactive ones. So, definition and experimenting with various re-injection architecture can be interesting field to further work on. Importantly re-injection, deletion or updating the linguistic annotations can be logically challenging and can lead to really useful work on reasoning with annotated corpus.

6.2.3 Others

Even though in the current implementation we have used the optimization technique to solve the issue of space (discussed in section 5.2.3), the way to optimize for faster execution has not been analyzed. At some points number of useless text-fragments are generated, which uselessly increase the writing and searching time, specially when the sentence is longer and have few annotations. This issue can be studied to some extent to improve the prototype. Further more statistical analysis of the performance of different types of DL reasoners can be studied to observe their compliance with reasoning with large A-box and SWRL rules, where incremental reasoning plays an important role.

Thus this research work basically opens up two paths for further research. Firstly detail study on more linguistic and semantic annotation types to discover new annotation types, new rules and gain more knowledge. Secondly further work on optimization of processing and integration of the model with other tools working with annotated corpus.

6.3 Conclusion

This research work has thus addressed the problem of reasoning with different types of annotations obtained from various sources in different paradigms each of which has it's own advantage. The formalism proposed for representation of semantic annotation provides a wider range for representation of semantic annotations. Prototype model developed using the existing de-facto semantic web technologies generated large number of new annotations which can be very useful for various tools. Not only did this research enhanced the importance of annotated corpus, but also motivates the linguists for further research on linguistic patters, so that they can be re-written in rule languages to automatically extract more annotations. It is also to be noted that the new/conflicting linguistic annotations, which are now not considered for further reasoning, open a challenge to handle/update such information in the knowledge base.

Above all this research work was an interesting field to work on, which gave an insight of various logical languages that can be used to solve different issues in an integrated environment. It was also an great opportunity to study the applicability of various semantic web technologies for the development of an interesting application based on logical reasoning.

Appendix A

Theorem Proofs

Theorem 4.15
Given a text annotation knowledge base $TaKb$, $TaKb$ is satisfiable if and only if $\phi(TaKb)$ is satisfiable under classical DL semantics.
Proof Lets assume that $TaKb$ is satisfiable under higher order semantics. Now we need to prove that $\phi(TaKb)$ is satisfiable under classical semantics.

Let us consider a interpretation $I = (\Delta, I_i, I_c, I_r)$ such that:
$Delta = \{tf, A\}$
$I_i = \{tf\}$
$I_c = \{A\}$
$I_r = \{(tf, A)\}$
and a semantic annotation $SA = \langle tf, A, sa : Concept \rangle$, which is satisfiable in I, under the higher order semantics as given in 4.1.1.2.

Now we get a interpretation $I^{'}$ after translation by applying ϕ function, which is as $I' = (\Delta^I, .^I)$ such that:
$\Delta = \{tf_{indv}, A_{indv}, tf_{concept}, A_{concept}\}$
$(\mu_o(A))^I = \{A_{indv}\}$
$(tf)^I = \{tf_{indv}\}$
$(\mu_c(tf))^I = \{tf_{concept}\}$
$(\mu_c(A))^I = \{A_{concept}, tf_{concept}\}$
$(\mu_r(sa : Concept))^I = \{(tf_{indv}, A_{indv})\}$
where the new names are obtained by the injective functions μ_* defined in section 4.2.2.1

Also, the ϕ translation on SA gives us $\phi(SA) = \mu_r(sa : concept)(tf, \mu_o(A))$ and $\mu_c(tf) \subseteq \mu_c(ot)$. Now we are to prove that $phi(SA)$ is satisfiable under classical semantics in the interpretation $I^{'}$.

$\mu_r(sa : Concept)(tf, \mu_o(A))$ is satisfiable if $(tf^I, (\mu_o(A))^I) \in (\mu_r(sa : Concept))^I$. Which is true in $I^{'}$

$\mu_c(tf) \subseteq \mu_c(A)$ is satisfiable if $(\mu_c(tf))^I \subseteq (\mu_c(A))^I$. which is true in $I^{'}$. Similar results will follow for other cases of annotation types. Thus, we can observe that ϕ is a satisfiability preserving transformation. This allows to say that we can compute the satisfiability of $TaKb$ by computing the satisfiability of $\phi(TaKb)$ under the classical semantics.

The other way round of the proof can be proved the similar way, by taking the inverse of the μ relation.
■

Theorem 4.16
Given a text annotation knowledge base $TaKb$, $TaKb$ is coherent if and only if $\phi(TaKb)$ is coherent under classical DL semantics.
Proof The proof follows from the proof A, because checking coherency is the result of checking satisfiability. ■

Theorem 4.17
Given a text annotation knowledge base $TaKb$ and the domain ontology is in $\mathcal{ALC}$, then checking the satisfiability of $TaKb$ is EXPTIME-hard.
Proof Trivial, from theorem 4.15. Because, checking satisfiability of $TaKb$ is same as checking satisfiability of $\phi(TaKb)$ under classical semantics. ■

Theorem 4.20
Suppose $TaKb$ is the knowledge base satisfying $TaKb \models_e C \sqsubseteq D$ and TaKb $\models_e \langle tf, C, \text{sa:Concept}\rangle$, then we have $TaKb \models_e \langle tf, D, \text{sa:Concept}\rangle$.
Proof To solve this problem, we will make use of the definition 4.19 and the reductions used in 4.2.2.1. By stronger semantics we get. Let I be an interpretation for $TaKb$, then I
$\models_s \mu_c(C) \sqsubseteq \mu_c(D)$ ——-(1)
$\models_s \mu_c(tf) \sqsubseteq \mu_c(C)$——-(2)
then we get
$\models_s \mu_c(tf) \sqsubseteq \mu_c(D)$ from 1 and 2

Now taking the weaker semantics we can write $I \models_w \langle tf, D, \text{sa:Concept}\rangle$. By which we can claim $I \models_s \langle tf, C, \text{sa:Concept}\rangle$ and $I \models_s C \sqsubseteq D$ implies $I \models_w \langle tf, D, \text{sa:Concept}\rangle$.

Hence by definition of entailment 4.19 we get $TaKb \models_e \langle tf, D, \text{sa:Concept}\rangle$ ■

Theorem 4.11
If a semantic annotation $\alpha^{'}$ is deduced by reasoning over $AnnoSet \cup Rules_{1-7}$ and $\{\alpha_1, ..., \alpha_n\}$ is the proof path for the conclusion $\alpha^{'}$, then if any $\alpha_i \in \beta_u$ occurring in the path is deleted i.e $\alpha_i \in \beta_u^d$ then the conclusion $\alpha^{'}$ is also deleted i.e $\alpha^{'} \in \beta_u^d$.
Proof Let us consider the two words case, then we get the following text-fragments tf_1, tf_2, tf_{12} and $AnnoSet = \{\langle tf_1, A, \text{sa:Concept}\rangle, nn(tf2, tf1)\}$. By reasoning we get $AnnoSet^{'} = AnnoSet \cup \{$ $\langle tf_2, U^*, \text{sa:Concept}\rangle$, $\langle tf_{12}, U^*, \text{sa:Concept}\rangle$ $\}$. For this theorem since, we deal with U^*, so the conclusion involving U^* in it's proof path is $\langle tf_{12}, U^*, \text{sa:Concept}\rangle$. The proof path is $\{\langle tf_2, U^*, \text{sa:Concept}\rangle\}$. Now, we get the following cases:
Case 1:
If, $\langle tf_2, U^*, \text{sa:Concept}\rangle \notin \beta_u^d$, then $\langle tf_{12}, U^*, \text{sa:Concept}\rangle \notin \beta_u^d$. Which is TRUE.
Case 2:
If, $\langle tf_2, U^*, \text{sa:Concept}\rangle \in \beta_u^d$ then its because $\exists\langle tf_2, A, \text{sa:Concept}\rangle$. But, if this is the case, then by reasoning with rules, we also get that $\langle tf_{12}, A, \text{sa:Concept}\rangle$ by rule 2, which will trigger the deletion operation. Thus, $\langle tf_{12}, U^*, \text{sa:Concept}\rangle \in \beta_u^d$

This case can be generalized with the case of three word text-fragments. ■

Theorem 4.13
Suppose M is the minimal model of the annotation Datalog P_{anno}, that is $M = \mathcal{MM}(P_{anno})$. Let M' be the set of ground atoms obtained after the deletion operation and $P^{'} = P_{anno}|_{M'}$. We have $M^{'} = \mathcal{MM}(P^{'})$.

That is, ours deletion operator can get a $M^{'}$ which is a stable sub-model of the given set of P_{anno}.

Proof It is obvious that $\mathcal{MM}(P^{'}) \subseteq M^{'}$ because $HB(P^{'}) \subseteq M^{'}$ by the definition of $P^{'}$ (i.e. all atoms of $P^{'}$ are in $M^{'}$) and $\mathcal{MM}(P^{'}) \subseteq HB(P^{'})$ by the definition of minimal model and Herbrand base.

To prove that $M^{'} \subseteq \mathcal{MM}(P^{'})$, for any $x \in M^{'}$, we have $x \in M$ since $M^{'} \subseteq M$. We claim that there exists at least one proof path of x from P, denoted $\{x_1, ..., x_n, x\}$ such that $x_i \in M'$. Otherwise, by the definition 4.11, if $x_i(1 \leq i \leq n)$ is deleted, that is $x_i \notin M^{'}$, then x is also delete ($x \notin M^{'}$) which is contrary with the hypothesis that $x \in M^{'}$. Therefore, $\{x_1, ..., x_n, x\}$ is a proof path of x in $P^{'}$ (since $x_i \in M^{'}$), so $x \in \mathcal{MM}(P^{'})$. Hence we get that for every x if $x \in M^{'}$ then $x \in \mathcal{MM}(P^{'})$ and if $x \notin M^{'}$ then $x \notin \mathcal{MM}(P^{'})$. Hence $M^{'} = \mathcal{MM}(P^{'})$ ■

Appendix B

Workout examples

Example: 1

To, demonstrate the use of reasoning on semantic and linguistic annotations with SWRL rules, let us consider the figure below, which is the sentence 30 in the corpus that we have taken into consideration. The text below marks the input semantic annotations corresponding to the textual units covered by the box. The arrows are for the linguistic annotations and the label marks the type of linguistic annotation.

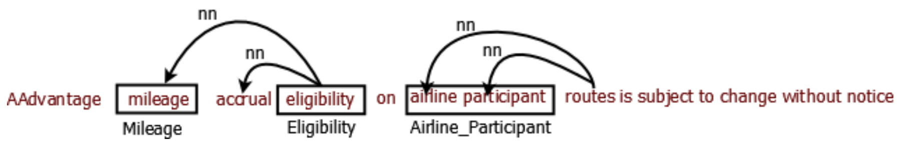

Figure B.1: Ontology-based Annotation on texts

After reasoning with SWRL rules, for this sentence, we get the following output for that sentence.

```
<TFF8_0-30,isSourceConcept,UNKNOWN>
<TFF6_0-30_TO_8_0-30,isSourceConcept,UNKNOWN>
<TFF7_0-30_TO_8_0-30,isSourceConcept,UNKNOWN>
<TFF3_0-30_TO_4_0-30,isSourceConcept,UNKNOWN>
<TFF2_0-30_TO_4_0-30,isSourceConcept,Eligibility>
<TFF3_0-30_TO_4_0-30,isSourceConcept,Eligibility>
--Removed----<TFF3_0-30_TO_4_0-30,isSourceConcept,UNKNOWN>
****************WRONG ANNOTATIONS WARNINGS****************
< TFF6_0-30_TO_7_0-30 , Airline_Participant> OR < TFF7_0-30_TO_8_0-30 , UNKNOWN>
****************WRONG NN WARNINGS****************
Remove nn(TFF8_0-30,TFF6_0-30) and Add nn(TFF7_0-30,TFF6_0-30)
```

Regarding the semantic annotations, TFF8_0-30 i.e textual unit *routes* is inferred *UNKNOWN* by rule 5. TFF7_0-30_TO_8_0-30 and TFF6_0-30_TO_8_0-30 corresponding to textual units *participant routes* and *airline participant routes* respectively are inferred *UNKNOWN* by the rules 2 and 3 respectively. TFF_4_0-30 is inferred *UNKNOWN* by rule 4 and hence TFF_3_0-30_TO_4_0-30 (accrual eligibility) is *UNKNOWN* by rule 2. Similarly, TFF_3_0-30_TO_4_0-30 (mileage accrual eligibility) and TFF_2_0-30_TO_4_0-30 is inferred *Eligibility* by rules 2 and 3 respectively. The deletion operation is performed as defined by proposition 4.10. Note, that all these inferred annotations are added in our knowledge base $TaKb$ to be used for checking the consistency and coherency after the ϕ transformation.

Now we will talk about the warnings generated which are for the information to the user and don't change our knowledge base. Regarding the linguistic annotations warning, it suggests the noun form modifiers to be removed between TFF8_0-30 and TFF6_0-30 i.e. *nn(routes, airline)* and instead add between TFF7_0-30 and TFF6_0-30 i.e. *nn(participant, airline)*, which is inferred by the rule 6. Similarly the semantic annotations warning stating annotation corresponding to one of the textual units *airline participant* OR *participant routes* is inferred by the rule 7.

***Note:** This example also demonstrates how incremental reasoning is used to infer new information, based on the originally asserted and the inferred assertion during reasoning with rules.*

Example: 2

To demonstrate the pellet reasoning to check inconsistency let us consider the following sentence. In addition it is to be noted that in the domain ontology we have defined the *Disjoint* relation between the concepts *A* and *B*.

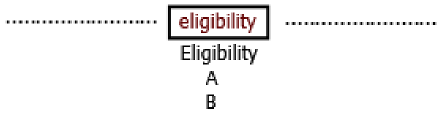

Figure B.2: Use-Case 2 for inconsistency check

For the considered fragment of the sentence 30, in addition to the annotations in the figure B.1, we have two more annotations viz textual-unit *eligibility* is annotated as *A* and *B*, linked with annotation type sa :$isSourceInstanceOcc$. With this scenario we will get the following output:

```
Inconsistent: because:
DisjointClasses(<http://lipn.univ-paris13.fr/RCLN/semAnno#A>
<http://lipn.univ-paris13.fr/RCLN/semAnno#B>)
ClassAssertion(<http://lipn.univ-paris13.fr/RCLN/semAnno#B>
 <http://lipn.univ-paris13.fr/RCLN/semAnnoTF#TFF4_0-30>)
ClassAssertion(<http://lipn.univ-paris13.fr/RCLN/semAnno#A>
 <http://lipn.univ-paris13.fr/RCLN/semAnnoTF#TFF4_0-30>)
```

This shows the application of reasoning service to check the in-consistency.

Example: 3

To demonstrate the pellet reasoning to check incoherency let us consider the following figure taken as fragment of sentence in fig B.1. The added information in domain ontology as in example above remains itact.

mileage
Mileage
A
B

Figure B.3: Use-Case 3 for Coherency check

For this considered fragment, two more annotations viz textual-unit *mileage* is annotated as *A* and *B*, linked with annotation type sa :$isSourceInstanceOcc$. With this scenario we will get the following output. In this case we get the following output:

```
Not Coherent: because TFF2_0-30 is not Satisfiable
```

Example: 4 Here we will show the example of SWRL rule written in the OWL format, which is the format to be used by the prototype model. All the rules written before in other format are used for readability purpose.
$contains2(?y, ?w), contains2(?y, ?x), isSourceConcept(?x, ?z), nn(?x, ?w) \rightarrow isSourceConcept(?y, ?z)$

This rule is represented in this format. The name-spaces are defined as required.

```
<swrl:Imp>
<swrl:head>
   <swrl:AtomList>
     <rdf:rest rdf:resource="nil"/>
      <rdf:first>
       <swrl:IndividualPropertyAtom>
         <swrl:propertyPredicate rdf:resource="isSourceConcept"/>
          <swrl:argument1 rdf:resource="#y"/>
          <swrl:argument2 rdf:resource="#z"/>
        </swrl:IndividualPropertyAtom>
      </rdf:first>
     </swrl:AtomList>
    </swrl:head>
    <swrl:body>
     <swrl:AtomList>
       <rdf:first>
         <swrl:IndividualPropertyAtom>
           <swrl:propertyPredicate rdf:resource="contains2"/>
            <swrl:argument2 rdf:resource="#w"/>
            <swrl:argument1 rdf:resource="#y"/>
          </swrl:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest>
         <swrl:AtomList>
           <rdf:rest>
             <swrl:AtomList>
               <rdf:first>
```

```
                 <swrl:IndividualPropertyAtom>
                   <swrl:propertyPredicate rdf:resource="isSourceConcept"/>
                    <swrl:argument1 rdf:resource="#x"/>
                    <swrl:argument2 rdf:resource="#z"/>
                   </swrl:IndividualPropertyAtom>
                </rdf:first>
                <rdf:rest>
                 <swrl:AtomList>
                  <rdf:rest rdf:resource="nil"/>
                  <rdf:first>
                    <swrl:IndividualPropertyAtom>
                      <swrl:propertyPredicate rdf:resource="nn"/>
                      <swrl:argument2 rdf:resource="#w"/>
                    <swrl:argument1 rdf:resource="#x"/>
                   </swrl:IndividualPropertyAtom>
                 </rdf:first>
               </swrl:AtomList>
             </rdf:rest>
           </swrl:AtomList>
          </rdf:rest>
          <rdf:first>
            <swrl:IndividualPropertyAtom>
               <swrl:propertyPredicate rdf:resource="contains2"/>
               <swrl:argument2 rdf:resource="#x"/>
             <swrl:argument1 rdf:resource="#y"/>
           </swrl:IndividualPropertyAtom>
         </rdf:first>
       </swrl:AtomList>
     </rdf:rest>
   </swrl:AtomList>
</swrl:body>
</swrl:Imp>
```

Bibliography

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and Patel F. P. Schneider. *The description logic handbook*. Cambridge University Press New York, NY, USA, 2007.

[2] Sean Bechhofer, Raphael Volz, and Phillip W. Lord. Cooking the semantic web with the owl api. In *International Semantic Web Conference*, pages 659–675, 2003.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[4] Philippe Besnard and Anthony Hunter. Quasi-classical logic: Non-trivializable classical reasoning from inconsistent information. In Christine Froidevaux and Jrg Kohlas, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, volume 946 of *Lecture Notes in Computer Science*, pages 44–51. Springer Berlin / Heidelberg, 1995.

[5] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, WWW Alt. '04, pages 74–83, New York, NY, USA, 2004. ACM.

[6] Marie catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *In LREC 2006*, 2006.

[7] Marie catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *In LREC 2006*, 2006.

[8] Laurence Cholvy and Anthony Hunter. Information fusion in logic: A brief overview. In *Proceedings of the First International Joint Conference on Qualitative and Quantitative Practical Reasoning*, pages 86–95, London, UK, 1997. Springer-Verlag.

[9] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Azzurra Ragone. Second-order description logics: Semantics, motivation, and a calculus. In Volker Haarslev, David Toman, and Grant E. Weddell, editors, *Description Logics*, volume 573 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[10] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming, 1997.

[11] Nikhil Dinesh, Aravind Joshi, Insup Lee, and Bonnie Webber. Extracting formal specifications from natural language regulatory documents. In *ICoS-5*, Buxton, England, 2006.

[12] Nikhil Dinesh, Aravind Joshi, Insup Lee, and Bonnie Webber. Extracting formal specifications from natural language regulatory documents. In *In: Proceedings of the Fifth International Workshop on Inference in Computational Semantics*, 2006.

[13] Patrice Enjalbert, Benoˆ Habert, and Kalina Bontcheva., editors. *Platforms for Natural Language Processing*, volume 49-2 of *TAL*. Atala, 2008.

[14] Francesca Fallucchi, Maria Teresa Pazienza, Noemi Scarpato, and Armando Stellato. Semantic turkey - a new web experience in between ontology editing and semantic annotation. In José Cordeiro, Joaquim Filipe, and Slimane Hammoudi, editors, *WEBIST (2)*, pages 90–97, 2008.

[15] Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. On higher-order description logics. In Grau et al. [17].

[16] Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. On higher-order description logics. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motikand, and Ulrike Sattler, editors, *Description Logics*, volume Vol. 477, Oxford, July 2730 2009. CEUR Workshop Proceedings.

[17] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors. *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30, 2009*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[18] Peter Haase, Pascal Hitzler, Markus Krtzsch, Jrgen Angele, and Rudi Studer. Practical reasoning with owl and dl-safe rules. 2006.

[19] Matthew Horridge and Sean Bechhofer. The owl api: A java api for working with owl 2 ontologies, 2009.

[20] Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The instance store: Dl reasoning with large numbers of individuals. In *Proc. of the 2004 Description Logic Workshop*, 2004.

[21] Ian R. Horrocks. Using an expressive description logic: Fact or fction? In *Proc. of KR 98*, pages 636–647, 1998.

[22] Anthony Hunter. Reasoning with contradictory information using quasi-classical logic. *Journal of Logic and Computation*, 10:677–703, 1999.

[23] Nancy Ide and Laurent Romary. Outline of the international standard linguistic annotation framework. In *Proceedings of ACL'03 Workshop on Linguistic Annotation: Getting the Model*, pages 1–5, 2003.

[24] Nancy Ide and Laurent Romary. Representing linguistic corpora and their annotations. In *Proceedings of the Fifth Language Resources and Evaluation Conference (LREC*, 2006.

[25] Atanas Kiryakov, Damyan Ognyanov, and Dimitar Manov. OWLIM A Pragmatic Semantic Repository for OWL. pages 182–192. 2005.

[26] Atanas Kiryakov, Borislav Popov, Damyan Ognyanoff, Dimitar Manov, and Kirilov Miroslav Goranov. Semantic annotation, indexing, and retrieval. *Journal of Web Semantics*, 2:49–79, 2004.

[27] Michal Laclavik, Martin Seleng, Emil Gatial, Zoltan Balogh, and Ladislav Hluchy. Ontology based text annotation - ontea, 2006.

[28] F. Levy, A. Guisse, A. Nazarenko, N. Omrane, and S. Szulman. An environment for the joint management of written policies and business rules. *Tools with Artificial Intelligence, IEEE International Conference on*, 2:142–149, 2010.

[29] Yue Ma, Francois Levy, and Sudeep Ghimire. Reasoning with text annotations. In *Proc. of the 24th Florida Artificial Intelligence Research Society Conference (FLAIRS-24)*, 2011.

[30] Yue Ma, Adeline Nazarenko, and Laurent Audibert. Formal description of resources for ontology-based semantic annotation. In *LREC 2010*, 2010.

[31] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *COMPUTATIONAL LINGUISTICS*, 19(2), 1993.

[32] Diana Maynard, Horacio Saggion, Milena Yankova, and Wim Peters. natural language technology for information integration in business intelligence. In *10th International Conference on Business Information Systems*, pages 25–27, 2007.

[33] Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universitt Karlsruhe (TH), Karlsruhe, Germany, January 2006.

[34] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax.
urlhttp://www.w3.org/TR/owl2-syntax/, 2008.

[35] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for owl-dl with rules. In *Journal of Web Semantics*, pages 549–563. Springer, 2004.

[36] OMG. Sbvr, http://www.omg.org/spec/sbvr/current (2008). `http://www.omg.org/spec/SBVR/Current`, 2006.

[37] OMG. Owl 2 web ontology language new features and rationale. `http://www.w3.org/TR/owl2-new-features/`, 2009.

[38] OMG. Owl2, http://www.w3.org/tr/owl2-overview/introduction. `http://www.w3.org/TR/owl2-overview`, 2009.

[39] Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *AAAI 2007*, pages 913–918. AAAI Press, 2007.

[40] Frederick Reiss, Sriram Raghavan, Rajasekar Krishnamurthy, Huaiyu Zhu, and Shivakumar Vaithyanathan. An algebraic approach to rule-based information extraction. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 933–942, Washington, DC, USA, 2008. IEEE Computer Society.

[41] Satya S. Sahoo and Krishnaprasad Thirunarayan. Tableau algorithm for concept satisfiability in description logic alch, 2009.

[42] Warren Shen, AnHai Doan, Jeffrey F. Naughton, and Raghu Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB 2007, pages 1033–1044. VLDB Endowment, 2007.

[43] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner, 2007.

[44] Refresh Software. Applying loosely-coupled architectures to the content driven enterprise. `http://www.refreshsoftware.com/SiteObjects/031230514FB35BB8010DAA48FA4F1C1C/Applying%20Loosely%20Coupled%20Architecture%20To%20The%20CDE.pdf`, 2006.

[45] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. Sofie: A self-organizing framework for information extraction. Technical Report 5-004, Max Planck Institute, Saarbrcken, Nov 2008.

[46] Victoria Uren, Philipp Cimiano, José Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Journal of Web Semantics*, 4(1):14–28, 2006.

[47] Victoria Uren, Philipp Cimiano, José Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semant.*, 4:14–28, January 2006.

[48] Xiaowang Zhang, Guilin Qi, Yue Ma, and Zuoquan Lin. Quasi-classical semantics for expressive description logics. In Grau et al. [17].