KNOWLEDGE REPRESENTATION USING DESCRIPTION LOGICS FORMALISM:

KNOWLEDGE INTEGRATION AND REASONING PERFORMING

Yasser YAHIAOUI

Djillali Elyabes University of Sidi Belabbes

Department of informatics

Faculty of Engineering

Djillali Elyabes University of Sidi Belabbes

This dissertation is submitted for the degree of Doctor of science

December 2016

This project was directed by Pr. Ahmed LEHIRECHE

To my mother Luisa

My father Laid

Sisters & bothers

Future wife K,

My teachers from primary school until end of studies.

Yasser Y.

Knowledge Representation Using Description Logics Formalism: Knowledge Integration and Reasoning Performing- Yasser Yahiaoui – November 2016

ABSTRACT

The most familiar concept in Artificial intelligence is the knowledge representation. It aims to find explicit symbolization covering all semantic aspects of knowledge, and to make possible the use of this representation to produce an intelligent behavior like reasoning.

The most important constraint is the usability of the representation; it's why the structures used must be well defined to facilitate manipulation for reasoning algorithms which leads to facilitate their implementation.

In this thesis we propose a new approach based on the description logics formalism for the goal of simplification of description logics system implementation. This approach can reduce the complexity of reasoning Algorithm by the vectorisation of concept definition based on the subsumption hierarchy. This idea yields to create the so-called method SHAMS.

ACKNOWLEDGEMENTS

FIRST WE THANKS GOD FOR THE HELP AND STRENGTH THAT HE GAVE US TO ACHIEVE THIS WORK

I PRESENT MY FULL ACKNOWLEDGMENTS TO MY TEACHER PR. AHMED LEHIRECHE WHO IS A REAL INSPIRATION EXAMPLE AS TEACHER AND SEARCHER.

I THANK MY FRIENDS AND COLLEAGUES OF RESEARCH GROUP INFO-TEAM NAÂMA.

ALSO THANK YOU TO DR. REDA ADJOUJ, DR. REDA HAMOU, DR DJEMAL BEN SABER, DR. SOUFIANE BOUKLI AND AMINE ABDELMALEK FOR THE ACCEPTANCE OF EXAMINATION OF THIS THESIS.

ESPECIAL ACKNOWLEDGEMENT TO DR. DJELLOUL BOUCHIHA AND ALL COLLEAGUES OF UNIVERSITY CENTRE OF NAÂMA ALSO MY TEACHERS AND COLLEAGUES OF INFORMATICS DEPARTMENT AT UDL SIDI-BELABBES

CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Abbreviations and Acronyms

AL:	Attributed Language
DL:	Description Logics
DL-KRS:	Description Logics Knowledge Representation Systems
KR:	Knowledge Representation
SN:	Semantic Networks

List of Appendices

INTRODUCTION

Artificial intelligence has two different definitions each one focus on a point of view; the first one define AI like a field of research in which man try to give to the machine the ability of thinking as human, when the second speaks about acting as human. The two definitions aim the same goal which is a machine simulating the human behaviour, but the difference is on methods and approach.

The first point of view is based on the modelling of natural phenomenon with these processes and mechanisms. However, the second is based on modelling knowledge in logical computational forms to be manipulated with logical and mathematical models.

This logic based approaches have reviled a lot of concepts like semantics, knowledge representation, reasoning algorithms, expressiveness, and completeness, these concepts will be explained later with more details.

When speaking about knowledge representation, we are involving a higher level of automation and calculability, because we speak about representation which can carry with identification: the semantic (meaning), contextual (pragmatic), and relational aspect, these ones when they are well defined in a representation it will be qualified of strong and meaningful representation. The complexity of manipulation Algorithms using these representations must be simplified to be an input in deduction process of implicit knowledge from a minimal set of explicitly represented knowledge. That is called the reasoning process. Here we speak about a compromise to make a meaningful representation with simple representation structures. This aimed compromise is justified by the fact that more meaning integration entail more complexity of representation's structures and manipulation's Algorithms.

From the history of knowledge representation, we can find lot of formalisms of representation the most famous ones are those which gave good result in implementation when they are founded on heuristics ideas and method or theoretically proved when they are based on well formed mathematic or logic entailments, and they result from the eventual deduction from known and accepted axioms and theories.

One of these formalisms is Semantic networks were developed after the work of Quillian in 1967, with the goal of characterizing by means of network-shaped cognitive structures the knowledge and the reasoning of the system.

The second one is the frame systems which rise in works of Minski in 1981, it rely on the notion of a “frame” as a prototype and on the capability of expressing relationships between frames. This two formalisms are considered net work structures and they are motivating cognitive intuitions.

Description logics rise as successor of these previous formalisms to respond to limitations seen with semantic ambiguity for the SNs and low level of expressivity for the frames. DLs introduced a kind of semantic representation so-called logical based semantic which allowed expressing by terminological definition the three aimed aspects: contextual, semantic and relational.

In our work we consider the description logics the most important one. This formalism based on two kinds of relations: the subsumption and equivalence and define relations by the use of constructors as union (disjunction), intersection (conjunction), and negation (complement) , also it provides more detail with the quantification and restrictions possible using the existential quantifier, the universal quantifier and the number restriction, all this tools provides a strong and expressive formalism of knowledge representation and open researches on the best Algorithms for automated deduction and process design Looking for a complete reasoner and expressive knowledge base described in DLs.

Description logics are defined as a family of logic-based knowledge representation language that can be used to represent the terminological knowledge of an application domain in structured way.

To rich more expressivity of knowledge representation DLs languages have seen many augmentations, and developments.

The implementation the DLs systems lead to speak about problems that face researchers. As it is known the two major occupations of DLs systems developers are the expressivity and the effectiveness of the reasoner; " DLs systems need both expressive logics and fast reasoner procedures for deciding subsumption (or equivalently satisfiability) in such logics have discouragingly high worst-case complexities, normally exponential with respect to the problem size"[13] that means we have to evaluate this to aspect for each kind of representation derived from the DLs. It is what we are presenting here.

In this thesis we aim to create a representation having vectorial form. The idea treated is based on the possibility of simplification of the representation to make the design of reasoning process more simple and to find simple structures to carry the DLs based knowledge representation. Then the aimed compromise is between the complexity of reasoning and the level expressivity provided by the new representation. This compromise when reached yields to a new long work on new representations and new kind of reasoning Algorithms.

Then we present here a formal derivation from the interpretation of the DLs representation to find graphics forms where we can define the new attribute associated to the concept name to replace the definitions giving in axioms. This attribute can be easily manipulated and makes the reasoner more and simpler for implementation. Also we have to check if the new representation has no negative effects on the performances comparing to ordinary DLs knowledge systems.

This thesis is composed of two parts: the first one concerned with theoretical context and the background, when the second describes the proposed approach with illustrations of her strength and weaknesses.

The first part called Background contains three chapters; the first one describes the philosophical foundations of the artificial intelligence in general and especially the knowledge representations. The second speaks about Description logics systems beginning from definitions to the architecture and application domains. The third speaks about implementation and currents DLs systems.

The second part called proposed approach contains also three chapters beginning by the description of the new representation method and the generation process in the first chapter, and the reasoning adaptation to the new method of representation with evaluation of the Algorithm convergence. The third chapter contains a study case using a knowledge base created to be an add-on for Arabic language processing system.

PART I: BACKGROUND

CHAPTER 01: KNOWLEDGE REPRESENTATION FOUNDATIONS

1. INTRODUCTION

As beginning, we investigate knowledge representation (KR) and their techniques distinguished by their characteristics as the description logics (DLs)[1] which takes a part with formalisms that have grown out of the others knowledge representation techniques using frames and semantic networks.

The project interest on DLs systems performing, it's why we describe respectively KRs formalisms in general and especially the DLs systems, motivated by the fact that in one hand DLs offers complete and empirically traceable reasoning services [2] added to the varicosity of constructors which makes possible to build complex classes from simpler ones.

On the other hand they have wide range of applications. But their wider acceptance has hindered by their limited expressiveness and the intracabilty of their subsumption algorithms [1].

This section will be organized to make a background on the field. So we began with the philosophical foundations which introduce a sort of chronological and problematic development of formalisms and techniques in the field of knowledge representation with some historical notions.

The second section is interested on the knowledge representation formalisms. It contains terms definition and description of mechanisms before speaking about advantages and limits of each described formalism.

2. PHILOSOPHICAL FOUNDATION

We evolve heir some philosophical foundations to make a round on the present subject first we define our search domain which is the artificial intelligence (AI) that have a lot of definition according to the philosophical point of view. Very along tow main dimensions; the first dimension address the behavior what means that AI is searching to make a machine able to act as human, then the second one address the reasoning abilities that means that AI aims to make the machine able to think as human.

This tow points of view are distinguished each one to other by the methods used for measurement of the success. The first one measure against an ideal concept of intelligence, which is called rationality and it is defined as : «a system is rational if it does the right thing given what it knows» [3]. The second one «measure success in term of fidelity to human performance»[3]. Consequently we find in the literature some definitions that we will present classified like followed:

The rationality definitions look to the study of intelligent systems that think rationally as «the study of mental faculties through the use of computational models»[4] in the same mining «the study of the computations that makes it possible to perceive reason and act » [5] in the rational view the study of intelligent that act rationally is defined as «computational intelligence is the study of the design of intelligent agent»[6], and according to Nilson,1998 «AI … is concerned with intelligent behavior in artifacts»[7].

For the course of human performances this study is defined as «the existing new effort to make computers think … machines with minds, in the full and literature sense»[8]. Or «the automation of activities that we associate with human thinking, activities such as decision making, problem solving, learning… »[9]. And if we think about behavior o systems we find that the study of this kind of systems is «the art of creating machines that performs function that require intelligence when performed by people»[10] other ways «the study of how to make computers do things at which, at the moment, people are better»[11].

Consequently, in AI we can say that a classification like this is the result of the diffident points of view between the approach centred around the human like the Turing test approaches, by the other side the approaches centred around rationality as laws of thought approaches it's for what we arrive now to speak about the ability of computer to deal with the following tasks:

Natural language processing: to enable it to communicate successfully with humans by natural expression tools.

Machine learning: to adapt to new circumstances and to detect and extrapolate patterns

Computer vision: to perceive objects and distinguish then.

Robotics: to act and manipulate objects and move about.

Knowledge representation: to store what it knows with signification.

Automated reasoning: to use the stored information to answer question and new conclusion.

3. KNOWLEDGE REPRESENTATION

The knowledge representation is a research field concerned by formalisms and methods for providing a height level description of the world that can be effectively a main tool to the development of intelligent systems which means systems able to find implicit consequences of its explicit represented knowledge.

In 1970 s the knowledge representation field enjoyed great popularity [12] and the KR approaches were divided into two categories: logic-based formalisms which evolved out of the intuition that predicate calculus could be used unambiguously to capture facts about the world, and other non-logic-based approaches representation developed on more cognition notion as the network structures and rule based representations derived from experiments on recall from human memory and human execution of tasks like mathematical puzzle solving. Then these two categories have seen a development due to the use by applications developers. In on hand the non logical systems developed from very specific lines of thinking evolved to be treated a general purpose tools, expected to be applicable in different domains and on different type of problems, on the other hands the first order logic have seen a general and powerful machinery and consequently the logic-based approaches took a place in KR field, then the notion of reasoning was introduced.

3.1. NETWORKS BASED APPROACHES

The representation of knowledge in this approaches are done using ad-hoc data structure and reasoning is accomplished by similar ad-hoc procedure that can manipulate this structures [12]. These approaches are based on graphical interface. According to this lasts we specifies semantic networks and the frames as no-logical formalisms, the first ones were developed with the work of Quillian in 1967 that have as goals to represent by the net work-shaped the means of cognitive structure and facilitate the use of this structures of reasoning of the system.

a) Semantic Networks

KRs using SNs are based on the node and links, the nodes used to characterize concepts i.e. sets or classes of individuals objects, however the relationships are represented by links between node but some times when the relationships are characterized as complex links, they are represented by nodes at the place of links to rich more expressivity, these are carefully distinguished from nodes concepts. In addition concepts can have simple properties often called attributes which are typically attached to the corresponding nodes.[13]

Now days the SNs aims the representation of knowledge about concepts and their relationships defining for now treatment of knowledge about specific individuals contrary to the early networks that represents concepts and objects and the specific individual by the similarly using the nodes.

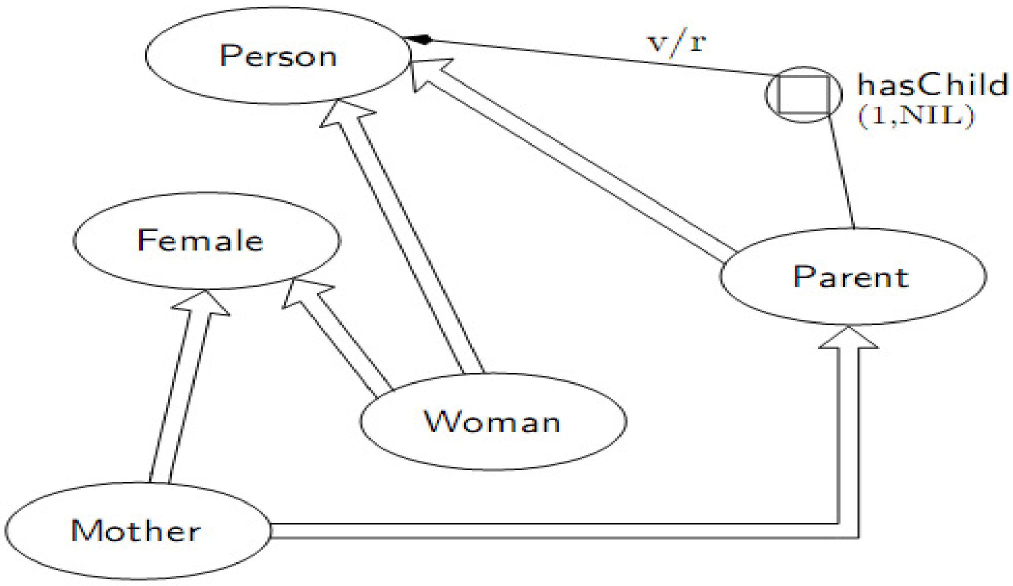


Figure 1 semantic network example

b) Frames

Are formalisms developed for th same goals of SN and similarity to. Them they are regarded as networks structures aiming to motivate cognitive intuition and in their features they have a strong common basis [12]

The frames are a data structure that typically used to represent a single object or a class of related object or a general concept (or predicate)[14]. There are some words which in literature were used synonymously for the words frame like memory unit and unit.

In general there is two or more type of frame such as class frames and instance frames. The former represent classes or sets of things and the latter represent particular instances of things [14]

A taxonomic hierarchy arranges the frames by links that much each frame to another, or some others parent frames which is a superset or a more general concept that include the set represented by the frame considered.

In the frame representations there are some notions that must be known as:

The slot: that means an attributes or properties of the thing represented by that frame and also it can describe a binary relation between two frames.

Inheritance: is the mechanism that causes a down propagation of the taxonomic hierarchy i.e. that represent a relation between two frames as one is considered more general than the other that means the second one represents a part of the set of individuals represented by the first one, then we accept that the second frame inherits slots from his parent add some others slots.

The subsumption is kind of relationships between class frames; it allows frames representation systems FRS to determine automatically the correct position of a class in taxonomic hierarchy [14] (data integration) we say that a concepts or a frames A subsumes a frames B if every instance of the concept B is an instance of A.

3.2. Logics and knowledge representation

In this section the first thing to define is logic which is a term attached to the philosophy "…it is where philosophy and logic, come together and become one… some species distinct from mathematical logic, symbolic logic or logic with any other familiar modifier"[15].

All this kinds of logics can be gathered in this definition: logic is the theory of consequence relations, of valid inferences such as it can be investigated and presented in many ways.

Logic offers structures and representations or formal systems to be used for analyzing concepts and augments that are central to philosophical inquiry. And it begins with languages that are qualified as constructed languages which will extend to natural languages or at less a portion of them.

Logic can be regarded as a second sense which is analysis and evaluation of propositions and arguments in the language used, in where we find well-formed formulas *WFF* that can stands in many relations, and these relations can be described in different ways, which reflect different ways of looking at logic and logical consequence. Some relations can be characterized syntactically terms of the grammatical structures of the expressions themselves [15]. And we speak here about logical syntax.

In logical approaches the meaning is achieve by defining a language such as the concepts will be represented using syntactic formulas that make it possible to define all the elements of the structures used on the above ideas. To be noted here: "that while the syntax may have different flavors in deferent settings the semantic is typically giving as a Tarski-style semantics" [13].

For this languages two kinds of alphabet are important and should be used which are unary predicate symbols to denote atomic concepts, and binary predicate symbols that are used to express relationships between concepts. And here constructers are used for building expressions as ∩ intersection, ∪ union but here it's between the set of individuals de note by the concepts name.

PART I: BACKGROUND

CHAPTER 02: DESCRIPTION LOGICS FORMALISM

1. THE DESCRIPTION LOGICS SYSTEMS

Description logics (DLs) [13, 16, 17] are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way. The name *description logics* is motivated by the fact that, on the one hand, the important notions of the domain are described by concept *descriptions*, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL; on the other hand, DLs differ from their predecessors, such as semantic networks and frames, in that they are equipped with a formal, *logic*-based semantics.

The definition of a concept is based on the two relations equivalence and subsumption, the first one is interpreted like sets equivalent because of the fact that every concept is viewed like a set of individuals. The second one represents a relation of inclusion i.e. we can say that a concept A is subsumed by B if all individuals that satisfy A definition satisfy B definition.

The expression of definitions is based on constructors (conjunction $\cap$, disjunction $\cup$), quantifiers ($\forall\, universel\ quantifier$, $\exists\, existential\ quantifier$) and numerical restrictions ($\leq n\ or \geq n$).

Description logics languages have seen lot of augmentations from the basic description logics languages till the actual languages, aiming more expressivity in knowledge description. These augmentations can be clearly seen when following the syntax after every augmentation.

To begin this section we will describe the basic foundations of DLs and connections to other models. We describe the structure and architecture of description logics systems, before coming to reasoning definition and characterization. The followers will be concerned by the compounds reasoning problems. The last section will describe the formal foundation of DLs systems and finally the DLs knowledge representation system DL-KRS.

2. BASIC FOUNDATION AND CONNECTIONS TO MODEL LOGICS

The DLs systems are initiated after the Tarski-style semantics based approaches, which are the semantique networks and frames. A brief view on history of description logics enables us to know the basic connections between DLs and other approaches. Then Research in Description Logics can be roughly classified into the following phases [18].

Phase 0 (1965–1980)

is the pre-DL phase, in which semantic networks [19] and frames [20] were introduced as specialized approaches for representing knowledge in a structured way, and then criticized because of their lack of a formal semantics [21,22, 23, 24]. An approach to overcome these problems was Brachman's structured inheritance networks [18], which were realized in the system KL-ONE, the first DL system.

Phase 1 (1980–1990)

was mainly concerned with implementation of systems, such as KL-ONE, K-REP, KRYPTON, BACK, and LOOM [25, 26, 27, 28, 29]. These systems employed so-called structural subsumption algorithms, which first normalize the concept descriptions, and then recursively compare the syntactic structure of the normalized descriptions [30]. These algorithms are usually relatively efficient (polynomial), but they have the disadvantage that they are complete only for very inexpressive DLs, i.e., for more expressive DLs they cannot detect all subsumption/instance relationships. During this phase, the first logic-based accounts of the semantics of the underlying representation formalisms were given [27, 31], which made formal investigations into the complexity of reasoning in DLs possible. For example, in [31] it was shown that seemingly small additions to the expressive power of the representation formalism can cause intractability of the subsumption problem. In [32] it was shown that subsumption in the representation language underlying KL-ONE is even undecidable, and in [33] it was shown that the use of a TBox formalism that allows the introduction of abbreviations for complex descriptions makes subsumption intractable if the underlying DL has the constructors conjunction and value restriction (these constructors were supported by all the DL systems available at that time). As a reaction to these negative complexity results, the implementors of the CLASSIC system (the first industrial-strength DL system) carefully restricted the expressive power of their DL [34, 35].

Phase 2 (1990–1995)

started with the introduction of a new algorithmic paradigm into DLs, so-called tableau based algorithms [36, 37, 38]. They work on propositionally closed DLs (i.e., DLs with all Boolean operators), and are complete also for expressive DLs. To decide the consistency of a knowledge base, a tableau based algorithm tries to construct a model of it by structurally decomposing the concepts in the knowledge base, thus inferring new constraints on the elements of this model. The algorithm either stops because all attempts to build a model failed with obvious contradictions, or it stops with a “canonical” model. Since, in propositionally closed DLs, the subsumption and the instance problem can be reduced to consistency, a consistency algorithm can solve all the inference problems mentioned above. The first systems employing such algorithms (KRIS and CRACK) demonstrated that optimized implementations of these algorithms led to an acceptable behavior of the system, even though the worst-case complexity of the corresponding reasoning problems is no longer in polynomial time [39, 40]. This phase also saw a thorough analysis of the complexity of reasoning in various DLs [37, 41, 42], and the important observation that DLs are very closely related to modal logics [43].

Phase 3 (1995–2000)

is characterized by the development of inference procedures for very expressive DLs, either based on the tableau approach [44, 45], or on a translation into modal logics [46, 47, 48, 49]. Highly optimized systems (FaCT, RACE, and DLP [50, 51, 52]) showed that tableau-based algorithms for expressive DLs led to a good practical behavior of the system even on (some) large knowledge bases. In this phase, the relationship to modal logics [46, 53] and to decidable fragments of first order logic [54, 55, 56, 57, 58] was also studied in more detail, and applications in databases (like schema reasoning, query optimization, and integration of databases) were investigated [59, 60, 61].

We are now in Phase 4, where the results from the previous phases are being used to develop industrial strength DL systems employing very expressive DLs, with applications like the Semantic Web or knowledge representation and integration in *Medicaland bio-informatics in mind*. On the academic side, the interest in less expressive DLs has been revived, with the goal of developing tools that can deal with very large terminological and/or assertional knowledge bases [62, 63, 64, 65].

The history of description logics illustrates the goals of DL-KRS implementors. We can find the expressiveness which means the power of a representation to characterize knowledge. The second one is the completeness of reasoning Algorithm. This last completeness represents the ability to deduce all true facts using the rules on system. The implementors must also preserve the consistency, efficiency and tractability.

3. STRUCTURES AND ARCHITECTURES

The core of Description Logics is the concept language, a formal language designed to describe classes and relationships between elements of the classes. DL based knowledge bases are built using concept languages expressions, and they are usually divided in two distinct parts: intentional and extensional. The intensional part describes the general schema of the classes and relationships, while the extensional part constitutes a (partial) instantiation of the schema, since it contains assertions about a set of individuals. Historically the intentional part takes the name of terminology or TBox, and the extensional part the name of assertional part or ABox [66].

A description logics system contains the knowledge part and processing part. The first is represented by the terminology description and the assertions description. The second represents the integration interface which allows the additions of definitions to the knowledge base and the reasoner who use the represented knowledge to deduce implicit correct facts.

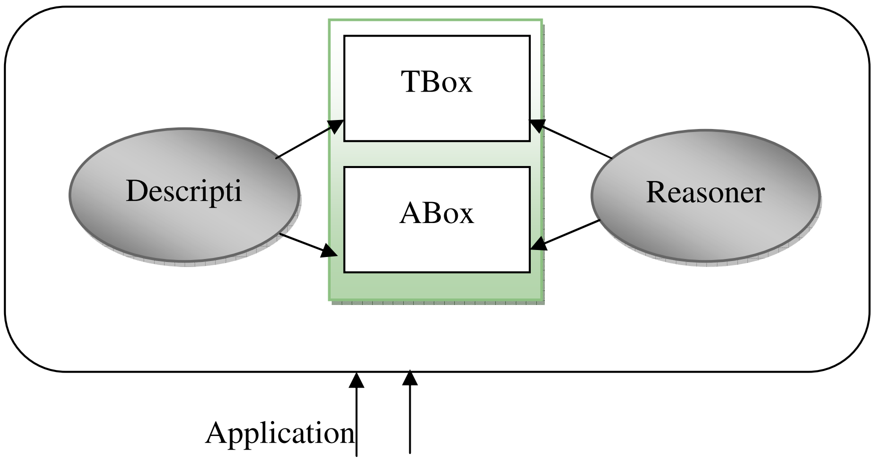


Figure 2 Architecture of DL-KRS

3.1. TBox and ABox

The knowledge base in DLs systems is composed by two parts terminological and assertional. The first define concepts and rules by the use of relations and constructors. And it s called TBox which is presented like a set of axioms that induce concepts definitions. The axiom $Gazelle \subset Mammal$ means that *Mammal* is generalisation of the concept Gazelle and here the term used is Gazelle is subsumed by *Mammal*. Other kind of axioms used to define roles which represent binary predicate. This lasts can be related with each other, and also used to make restrictions and relies between individuals satisfying concepts definitions. By this way is defined the hierarchy of concepts and the hierarchy of roles.

Description logics languages allow also assertions definition. And give information about individuals as instances of defined concepts and roles. For the roles it is a binary predicate instances. Other way, the concept assertion states that an instance satisfies a concept definition. For example *Mounir* is a *Father* and we note *Father*(Mounir). However, role assertion makes two individuals in relation like *Haschild* (*Mounir*, *Ryad*) which means that *Ryad* is a child of *Mounir* in natural language. Another kind of assertions is possible; it's called inequality assertion which allows making difference between two near individuals like *Meriem* and *Maria* or *M'hamed* and *Mohamed*. In this case the expression used is $M'hamed \neq Mohamed$.

Assembling assertions leads to construct the ABox (assertional box). When assembling concepts and roles definitions makes the TBox (terminological box) this two to gather make a DL's knowledge Base. This last can be defined like a minimal set of explicitly represented knowledge that allows expansions by computing the implicit knowledge.

3.2. DESCRIPTION LANGUAGE AND EVOLUTION

Description in DLs systems are based on a family of languages called concept languages. The fundamental elements are concepts and roles description. Each concept has a definition that can be satisfied by a set of individuals representing its assertions like in object model the concept represents the class and the assertions are objects of the class. The role is seen like relation between individuals or like attribute.

"The language is completely described by a formal syntax and a Tarsky-like semantics. A formal definition of the language is essential for knowledge bases characterisation, and for the definition of reasoning services"[13].

The expression in concept languages uses a set of constructors as operators and composed alphabet distincting concept names $\mathcal{CN}$ and role names $\mathcal{RN}$ and individual name $O.$ these elements are used to define a finite set of axiom with respect to a formal syntax.

Axioms are of the form

$$C \sqsubseteq D \text{ or } C \equiv D$$

Where C and D are concepts expressions.

Expressions figure in the form of

$$CN \mid \bot \mid \top \mid \neg\, C \mid C \cap D \mid C \cup D \mid \exists\, R.C \mid \forall\, R.C$$

DLs languages evolution is related to the goal of reaching more expressivity. It is why augmentations consist of some additional tools which yield to represent new aspects of knowledge. The table 01 shows *AL* languages evolution with specification of augmentations from generation to other.

Table 1 DLs language extensions and evolutions [68]

Generation	Augmentation
ALCN, ALCR and ALCNR	ALC augmented with the number restriction concept expressions (N) or/and role conjunction (R);
ALCF	ALC augmented with attributes (sometimes called features), attribute composition and attribute value map concept expressions;
ALCFN	ALCF augmented with the number restriction concept expressions;
ALCFNR	ALCFN augmented with role conjunction;
ALCN(_)	ALCN augmented with role composition in number restriction concept expressions;
ALC	ALC augmented with transitively closed primitive roles (axioms of the form RN 2 R);
ALC	ALC augmented with union, composition and transitive closure role expressions
ALC	ALC augmented with a restricted form of primitive role introduction axioms;
TSL	ALC augmented with union, composition, identity, transitive reflexive closure and inverse role expressions;
CIQ	TSL augmented with qualified number restriction concept expressions (inverse roles are the only form of role expression allowed inqualied number restrictions);

The table shows the augmentation from a generation to other. It is clear that the criteria's touched here gives more specification capabilities which mean more expressivity. As example from ALC to TSL the additional options are: the union, composition, identity, transitive reflexive closure and the inverse role. The same thing for the evolution from TSL to CIQ conceptors added inqualied restriction numbers to define an exact number of individuals participating in a relation denoted by a defined role name.

3.3. DLs reasoner

A knowledge base described in DL-KRS (DLs knowledge representation systems) is a set of concept definitions and assertions which define explicitly the minimal knowledge. To be correctly used for deduction process, it must satisfy some properties which aim the correctness and usability of the knowledge base by reasoner. These properties are formally defined as follows [13]. Let T be a terminology.

Satisfiability: a concept C is satisfiable with respect to $\mathcal{T}$ if there is an existing model I of $\mathcal{T}$ such that C^I is nonempty. In this case we say also that I is a model of C.

Subsumption: a concept C is subsumed by a concept D with respect to $\mathcal{T}$ if $C^I \sqsubseteq D^I$ for every model I of $\mathcal{T}$. In this case we write $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$.

Equivalence: two concepts C and D are equivalent with respect to $\mathcal{T}$ if $C^I = D^I$ for every model I of $\mathcal{T}$. In this case we write $C \equiv_{\mathcal{T}} D$ or $\mathcal{T} \models C \equiv D$.

Disjointness: two concepts C and D are disjoint with respect to $\mathcal{T}$ if $C^I \cap D^I = \emptyset$ for every model I of $\mathcal{T}$.

These properties are related by the possibilities of reduction to subsumption and to unsatisfiability. This is possible due to the following propositions.

Proposition 1 (Reduction to Subsumption)[13] For concepts C, D we have

C is unsatisfiable $\Leftrightarrow$ C is subsumed by $\bot$;

C and D are equivalent ⇔ C is subsumed by D and D is subsumed by C;

C and D are disjoint ⇔ C ∩ D is subsumed by ⊥.

Note that ⊥ represent a bottom concept which is interpreted like empty set of individuals. This means that when the unsatisfiability is defined by the non existence of individual satisfying the definition of the concept.

Tow concepts are equivalents if and only if every individual of a concept is an individual of the second one that means; every one of the two concepts is subsumed by the other one.

Disjointness is characterised by non existence of common individual that satisfy the two definitions of the two disjoints concept. What means; the conjunction is empty set with no individual who satisfy the definition of the conjunction expression.

Proposition 2. [13] (Reduction to Unsatisfiability) for concepts C, D we have

C is subsumed by D ⇔ C∩ ¬D is unsatisfiable;

C and D are equivalent ⇔ both (C ∩ ¬D) and (¬C ∩ D) are unsatisfiable;

C and D are disjoint ⇔ C ∩ D is unsatisfiable.

When C and the complement of D is unsatisfiable means there is no common individual between the negation of D and the subsumed concept C.

When the two concepts are equivalents means the double sense subsumption. Every intersection subsumer/subsumed is unsatisfiable.

The reasoning process is the main part of the DLKRS. It allows the entailment of logical consequences from the knowledge base. We can find two classes for reasoning services in DLs systems, the basic services and the complex services. The firsts consist on cheeking the truth value, which involve the satisfiability of knowledge base, the subsumption checking, the concept satisfiability and the instance checking.

The complex reasoning services are different from system to system. The most services provided are classification and retrieval. These services are additional to the top of the basic services. The classification has the goal of defining the taxonomy of concept. This last is presented like a graph in which nodes represent concepts and edges represent subsumption relation between them. The retrieval unable to answer queries about individuals occurring in defined concepts.

According to the subject which is represented by the knowledge base concerned, there is two kind of reasoning: terminological and assertional. The first use the TBox without considering the ABox. This is coming from the property of dependency of the terminology. due to this property the terminological reasoning is used for knowledge representation systems especially.

The second kind of reasoning is called Hybrid reasoning. Because it is concerned by the satisfiability checking and this last can replace each other property. By this way the hybrid reasoning is defined like in follows "Hybrid reasoning takes account of both the parts of a knowledge base. We consider algorithms for solving the problem of knowledge base satisfiability. In principle, this approach is general enough because all reasoning services can be reduced to knowledge base satisfiability" [66]. In the reality, the assertional reasoning treats the concept satisfiability but if we speak about hybrid reasoning we speak about KB satisfiability. Here there is an inclusion relationship because the concept satisfiability is a part of the KB satisfiability.

The reasoning algorithm, to check concept satisfiability begins with system constraints and applies completion rules while precondition are satisfied. And it stops when no rule is applicable. The result is that the system is completed with condition that there is no contradictory constraint. The completion rules are given like in table 02.

To be mentioned that the most of modern DLs systems use tableau decision procedure, to deal with more expressive DLs. This kind is called tableau Algorithms. It is based on KB satisfiability checker in order to decide subsumption problem.

Table 2 Completion rules for ALCN

If A contains ($C_1 \cap C_2$)(x), but not both C_1(x) and C_2(x) **then** $A' \equiv A\ U\ \{C_1(x), C_2(x)\}$
If A contains ($C_1 \cup C_2$)(x), but neither C_1(x) nor C_2(x) **then** $A' \equiv A\ U\ \{C_1(x)\}\ and\ A'' \equiv A\ U\ \{C_2(x)\}$
If *A* contains *(∃R.C)(x)* but there is no individual name *z* such that *C(z)* and *R(x,z)* are in *A* **then** $A' \equiv A\ U\ \{C(y), R(x,y)\}$
If *A* contains *(∀R.C)(x)* and *R(x,y)*, but it does not contains *C(y)* **then** $A' \equiv A \cup \{C(y)\}$.
If A contains $(\geq n\ R.C)(x)$ **and** there are no individual names z_1,.....z_2 such that R(x, z_i) (1 ≤ i ≤ n) **and** $z_i \neq z_j$ ($1 \leq i < j \leq n$) are contained in *A* **then** $A' = A \cup \{R(x,y_i) \mid 1 \leq i \leq n\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n\ \}$, where $y_1, \ldots, y_n$ are distinct indivi duals names not occurring in A.
If A contains distinct individual names $y_1, \ldots\ldots, y_{n+1}$ such that $(\leq n\ R)(x)\ and\ R(x, y_1), \ldots\ldots, R(x, y_{n+1})$ are in A, **and** $y_i \neq y_j$ is not in A for some $i \neq j$ **then** for each pair y_i, y_j such $i < j$ and $y_i \neq y_j$ is not in A, the Abox $A_{i,j} = [y_i/y_j]A$ is obtained from A by replacing each occurrence of y_i by y_j.

We note that the hybrid reasoning Algorithm is considered like a generalisation of the concept satisfiability. “For a large class of concept languages, including $\mathcal{ALCN}$, the algorithm can be a generalisation of that used for terminological reasoning. The notion of constraint system is extended, allowing the presence of constraints for individuals as well as variables”.[66]

A reasoning Algorithm can be qualified by correct if it is sound and complete. Soundness and completeness are defined as flow. “An algorithm that decides, given two concepts C and D, whether it holds that D subsumes C, is called complete if it is guaranteed that the algorithm returns “yes” whenever the subsumption relationship holds and it is called incomplete otherwise. The algorithm is called sound if it is guaranteed that, whenever the algorithm returns “yes” that D indeed subsumes C and it is called unsound otherwise. [67]

3.4. COMPLEXITY OF REASONING ALGORITHM

The reasoning Algorithm applies rules combined by iteration until a stop point. This last represents the case where no new inferences are given. The complexity is relative to how the Algorithm applies rules.

A modified algorithm is constructed to deal with non deterministic rules is presented like in follow. [13]

The Algorithm starts with $\{C_0(x_0)\}$

Applies the $\rightarrow\cap$ and $\rightarrow\cup$ as long as possible, and checks for clashes of form $A(x_0), \neg A(x_0)$ and $\perp(x_0)$;
Generates all the necessary direct successors of x_0 using the $\rightarrow \exists - and \geq -rule$;
Generates all the necessary identifications of these direct successors using the $\rightarrow\leq -$ rules and checks for clash caused by at-most restrictions;
Successively handles the successors in the same way.

Satisfiability of ALCN-concept descriptions is PSpace-complete.

The above argument shows that the problem is in PSpace. The hardness result follows from the fact that the satisfiability problem is already PSpace-hard for the sublanguage

ALC, which can be shown by a reduction from validity of Quantified Boolean Formulae. Since subsumption and satisfiability of ALCN-concept descriptions can be reduced to each other in linear time, this also shows that subsumption of ALCN-concept descriptions is PSpace complete.[13]

3.5. COMPOUND INFERENCE PROBLEMS

Some of the most important inference problems in DLs are of a compound nature [69] but heir it is clearly confirmed that this problem can be solved by the reduction into more basic inference problems mentioned above. At the same time if the target to achieve is the efficient implementation, it is vital to consider compound inference problem as first class citizens [70].

The compound inference problems take a part in DLs reasoners because they are used for more efficient and the important such problems are:

Classification: compute the restriction of the subsumption relation $\subseteq$ to the set of concept names used in T.

Realization: compute the set $R_{1,\mathcal{T}}$ (a) of those concept names A that are used in terminology T, satisfy A $\models_{\mathcal{T}}$ A(a) and are minimal with this property w.r.t the subsumption $\subseteq_{\mathcal{T}}$ [69].

Retrieval: try to give a set of assertions A and a set of terminology and a concept C for computing the set $I_{A,T}(C)$ of individuals names used in A satisfying A $\models$ C(a).

The compound inference problem offers a very important tools and services to DLs reasoners. The hierarchy of concept names constructed by the classification can facilitate the browsing and the structuring the Tbox by meaning that B is Abox A if and only if A $\subseteq_{\mathcal{T}}$ B. also the realization facilitates the browsing and understanding of the knowledge base. Add to that the realization offer tools for presuppositions of the concept memberships of individuals. [69] And the retrieval service the main use of this service is data base like querying of description logic's knowledge base: in some applications, it is natural to define ABox with huge number of individual names, and to query such ABox likes a data base with deductive capabilities [71].

The compound inference problems can be reduced into more basic inference problems. Obviously they becomes just use multiple invocations of instance checking and subsumption. However, basic inferences such as subsumption and instance checking are potentially vary costly, and thus it is vital for DLs reasoners to replace these "brute force" methods of compound inferences by more subtle approaches [69].

For more efficiency in implementation of compound inference services we have to think about reducing the number of subsumption tests because it's clear that as beginning the naïves approaches considers and performs n^2 tests of subsumption for a terminological box that's contains n concept names. The strategies of this optimization are described next.

The strategies that can be used for optimization of the reasoning services can be distinguished to two kinds, firstly non-DL context processing which is based on combinatory analysis for the classification problem that is regarded « as an abstract combinatorial problem on partial orders: compute a complete representation of a partial ordering by making as few as possible comparisons. This quite general problem is also considered in non-DL contexts » [72].

4. DESCRIPTION LOGICS SEMANTIC

The semantics of description logics leads to transform reflexions on represented knowledge from expressions and names to sets. This fact begins from the definition of I called the interpretation and figure as a no empty set, the second is the interpretation function unable to assign each atomic concept A to a set of individuals denoted by A^I containing as elements the individuals which satisfy the definition of the concept *,* by this why we define the domain name Δ^I . It represents a set including all concepts interpretation.

The interpretation function is extended to concept descriptions by the following inductive definitions [13]:

$$\mathrm{T}^{I} \equiv \Delta^{I}$$

$$\bot^{I} \equiv \Phi$$

$$\neg A^{I} \equiv \Delta^{I}/A^{I}$$

$$(C \sqcap D) \equiv (C^{I} \sqcap D^{I})$$

$$(\forall R.C)^{I} \equiv \{a \in \Delta^{\mathrm{I}} \mid \forall b.(a,b) \in \mathrm{R}^{\mathrm{I}} \rightarrow \mathrm{b} \in \mathrm{C}^{\mathrm{I}}\}$$

$$(\exists R.T)^{\mathrm{I}} \equiv \{a \in \Delta^{I} \mid \exists b.(a,b) \in R_{I}\}$$

The extensions of AL languages aiming more expressivity leads to express other aspects like: numerical restriction, roles construction, roles type functional and transitive Sf. The table 4 shows these extensions with their semantics.

The power of expressivity is coming from the operation allowed to describe complex relations existing between concepts and individuals. By corresponding to the first order logics in which we can find relation composition, inverse relations...Sf. these facilities can provide high level of expressivity. The semantic of composition and the inverse relation allowed computation of some complex interaction between predicates.

Some DLs language enable role composition, it is denoted by the symbol ○ and defined by the followings expressions.

$$R^{I} \circ S^{I} \equiv \{(a,c) \mid \exists b.(a,b) \in R^{I} \wedge (b,c) \in S^{I}\}.[13]$$

Iterated composition is denoted in the form $(\mathrm{R}^{\mathrm{I}})^{\mathrm{n}}$. To be more precise,

$$(R^{I})^{\circ} \equiv \{(d,d) \mid d \in \Delta^{I}\} \; and \; (R^{I})^{n+1} = (R^{I})^{n} \circ R^{I} \} \; [13]$$

Transitive and reflexive-transitive closures are the only constructors among the ones introduced so far that cannot be expressed in first-order predicate logic. [13] That means the composition is presented like an associative operation associating at least three Roles names or more.

Table 3 Some Description Logic concept constructors [01].

Name	Syntax	Semantic	Symbol
Top	T	Δ^I	AL
Bottom	$\bot$	Φ	AL
Intersection	$C \cap D$	$(C^I \cap D^I)$	AL
Union	$C \cup D$	$(C^I \cup D^I)$	U
Negation	$\neg C$	Δ^I / A^I	C
Value restriction	$\forall R.C$	$\{a \in \Delta^I \mid \forall b.(a,b) \in R^I \rightarrow b \in C^I\}$	AL
Existential quantifier	$\exists R.C$	$\{a \in \Delta^I \mid \exists b.(a,b) \in R^I\}$	$\mathcal{E}$
Unqualified number restriction	$\geq n R$ $\leq n R$ $= n R$	$\{a \in \Delta^I \| \|b \in \Delta^I \|(a,b) \in R^I\}\| \geq n$ $\{a \in \Delta^I \| \|b \in \Delta^I \|(a,b) \in R^I\} \leq n$ $\{a \in \Delta^I \| \|b \in \Delta^I \|(a,b) \in R^I\} = n$	N
Qualified number restriction	$\geq n R.C$ $\leq n R.C$ $= n R.C$	$\{a \in \Delta^I \| \|b \in \Delta^I \|(a,b) \in R^I \wedge b \in C^I\}\| \geq n$ $\{a \in \Delta^I \| \|b \in \Delta^I \|(a,b) \in R^I \wedge b \in C^I\} \leq n$ $\{a \in \Delta^I \| \|b \in \Delta^I \|(a,b) \in R^I \wedge b \in C^I\} = n$	$\mathcal{Q}$
Role value map	$R \subseteq S$ $R = S$	$\{a \in \Delta^I \|\forall b.(a,b) \in R^I \longrightarrow (a,b) \in S^I\}$ $\{a \in \Delta^I \|\forall b.(a,b) \in R^I \leftrightarrow (a,b) \in S^I\}$	
Agreement and disagreement	$u_1 = u_2$ $u_1 \neq u_2$	$\{a \in \Delta^I \| \exists b \in \Delta^I . u_1^I(a) = b = u_2^I(a)\}$ $\{a \in \Delta^I \| \exists b_1, b_2 \in \Delta^I . u_1^I(a) = b_1 \neq b_2 = u_2^I(a)\}$	$\mathcal{F}$
Nominal	I	$I^I \in \Delta^I$ with $\|I^I\| = 1$	$\mathcal{O}$

As mentioned before the description logics implementations use simplified syntax. This aim to facilitate the manipulation of constructors and to avoid non ASCII chars used to denote constructors and relations. In next table a concrete syntax is defined. In where some words like are used to express every possible definition in respect to relatively new AL generations.

The semantic do not change when using simplified syntax, because it is clear that every symbol (some of them are non ASCII chars) are replaced by a word which indicate the same meaning. But here it s to be noted that it s difficult to differentiate if this same word is used to express a concept in a knowledge base. It's why the notation is like polish notation.

Table 4 Concrete syntax of concept constructors [01].

Name	Abstract syntax	Concrete syntax
Top	T	TOP
Bottom	$\perp$	BOTTOM
Intersection	$C \cap \dots \cap D$	(and C ... D)
Union	$C \cup \dots \cup D$	(or C ... D)
Negation	$\neg C$	(non C)
Value restriction	$\forall R.C$	(all R C)
Existential quantifier	$\exists R.$T	(Some R)
Limited existential quantification	$\exists R.C$	(some R C)
Unqualified number restriction	$\geq n\, R$	(at-least n R)
	$\leq n\, R$	(at-most n R)
	$= n\, R$	(exactly n R)
Qualified number restriction	$\geq n\, R.C$	(at-least n R C)
	$\leq n\, R.C$	(at-most n R C)
	$= n\, R.C$	(exactly n R C)
Role value map	$R_1 \subseteq R_2$	(subset R_1 R_2)
Same as agreement	$R_1 = R_2$	(same-as R_1 R_1)
Role fillers	$\exists R.I_1 \cap \dots \cap R.I_n$	(fillers R_1 $I_1 \dots I_n$)
One-of	$\exists I_1 \cup \dots \cup.I_n$	(one-of $I_1 \dots I_n$)

The syntax provides possibilities for roles constructors. The role composition is denoted by (compose R_1 ... R_n), the role inverse is (inverse R), the complement is (not R), the transitive closer (transitive-closer R) and the role restriction (restrict R C). Like that is defined the terminological box using the noun of the constructor.

The assertions are also giving with respect to this concrete syntax like (instance a C) for indicating that a is an instance of C or a satisfy the definition of C. The instantiations of role is denoted (related a b R). some additional concept can be considered in more recent syntaxes.

5. APPLICATION DOMAINS

Description logics is known as a powerful formalism of KR. It provides tools that allow modelling several parts of the real world. These capabilities makes that DLs are well known by searchers in deferent domains like: Conceptual information modelling, natural language processing, software engineering, configuration, medical informatics, databases, digital libraries and web based information systems. Next we describe briefly how DLs can be useful in some fields.

5.1. Conceptual information modelling

This field is concerned by the expression of information in computable form. Other ways, symbolizing a part of the real word with abstraction of some details to have computable forms. Conceptual model creation is very helpful in applications modelling. It takes a big importance for many fields. In follows: a summarization of this fields as presented by *Mylopoulos* in 1998 [74].

Artificial intelligence programs turned out to require the representation of a great deal of human knowledge in order to act “intelligently”. As a result, they relied on conceptual models built up using knowledge representation languages, such as semantic networks directed graphs labelled with natural language identifiers. DLs are the historical descendants of attempts to formalize semantic networks.
The design of database systems was seen to have as an important initial phase the construction of a “conceptual level schema”, which determined the information needs of

the users, and which was eventually converted to a physical implementation schema. Chen's Entity–Relationship model, and later semantic data models, were the result of efforts in this direction.

"More generally, the development of all software has an initial requirements acquisition stage, which nowadays is seen to consist of a requirements model that describes the relationship of the proposed system and its environment. The environment in this case is likely to be a conceptual model.

Independently, the object-oriented software community has also proposed viewing software components (classes/objects) as models of real-world entities. This was evident in the features of Simula, the first object-oriented programming language, and became a cornerstone of most object-oriented techniques, including the current leader, UML.

An important claim regarding the benefits of abstraction in conceptual modelling is that it results in a structured information model, which is easier to build and maintain. Interestingly, Description Logics further this goal by supporting the automatic classification of concepts with respect to others, thereby revealing generalizations that may not have been recognized by the modeller" [13].

DLs offer tools to express conceptual models by the roles and concepts definitions also it provides representation for particular situations using assertions expression. For A. Borgida and R. J. Brachman the steps that lead to describe models in using DLs formalism are [13]:

- Identify the individuals one can encounter in the U of D. Revisit this later considering issues such as materialization and values.
- Enumerate concepts that group these values.
- Distinguish independent concepts from relationship-roles.
- Develop taxonomy of concepts. Revisit this later considering issues such as Disjointness and covering for subconcepts.
- Identify any individuals (usually enumerated values) that are of interest in all states of the world in this U of D.

- Systematically search for part-whole relationships between objects, creating roles for them. Later, make them sub-roles of the categories of roles.
- Identify other “properties” of objects, and then general relationships in which objects participate.
- Determine local constraints involving roles such as cardinality limits and value restrictions.
- Elaborate any concepts introduced as value restrictions.
- Determine more general constraints on relationships, such as those that can be modelled by sub-roles or same-as. (The latter often correspond to “inheritance” across some relationship other than IS-A.)
- Distinguish essential from incidental properties of concepts, as well as primitive from defined concepts.
- Consider properties of concepts such as rigidity, identifiers, etc., and use the techniques of to simplify and realign the taxonomy of primitive concepts.

5.2. Natural language processing

Description logics with the definition of a terminology is very adapted to be used for natural processing systems, especially in constructing the lexical knowledge bases called exactly the lexical semantics. The term in this case represents concepts of the knowledge base described in AL.

Description logics offer a logical form of represented knowledge and by this way it allows to drive the semantic interpretation process. Other ways, the DLs represents the concepts based on properties and relation with other concepts which integrate at the same time the contextual and the syntactic knowledge. This property makes the power of usability DLs in natural language processing (NLP). For this lasts, developing KBs based on DLs has been a subject of several works from the 1980s and beginning of 1990s.

To speak about the lexical semantic part of the knowledge base, it’s very simple to see that concepts definitions. But it is not same case when interesting to the syntactic integration, because it’s not directly made, the constructor must define roles able to make in relation two words of different nature like in verb phrases a verb like “write” is make in relation to the writer by a role defined that can be called “Subject” and by this same way we can define a relation for complements which leads to implementing syntax of verb phrases using roles definition.

It is known that the example cited bellow is for a simple form of verb phrases. In this case subject or complement figures like a single word. The definition becomes more complex when syntactic elements figure as combined words for example “Ahmed write his report” here the definition is possible But with more complex relation that can be roles combinations.

The semantic interpretation requires a knowledge base partitioned in two parts represented by an *Upper-Model* and a *Domain-Model* this like a theoretical purpose is very hard to be achieved. But the idea focus on the fact that “if selectional restrictions are too specific, disambiguation is achieved, but probably many correct sentences will be rejected (e.g., the sentences involving some form of metaphor, type shifting, or metonymy); if selectional restrictions are too general, the opposite problem may appear. In principle, a good linguistically motivated ontology should be abstract, large-scale, reusable. However, these goals are very hard to achieve since they conflict with the practical need to implement effective and discriminating Ontologies in specialized domains.” [13]

In practice, a number of works aiming the semantic interpretation are found. But the results are not satisfiable in the field of natural language processing. DLs provide a description formalism seen like powerful for the knowledge bases (called also Ontologies) construction but the manipulation and processing Algorithms are always far to respond to the requirements.

5.3. Description logics and semantic web

The earliest applications of description logics was concerned by digital library, we can find as example *Untangle* and *FindUR* which represent a web based information systems using DLs to express knowledge. The rise of semantic web caused the need of expressive formalisms; this last is justified by the need to create meaningful and usable resources.

Description logics are well adapted to describe elements of knowledge especially the complex relation between them.

For the goal of reasoning on resources, Web languages describe knowledge using frame-like syntax. But some extensions are needed to achieve height level of expressivity like description logics. These extensions can be seen in OIL language like in follows (see [13]):

- Arbitrary Boolean combinations of classes (called class expressions) can be formed, and used anywhere that a class name can be used. In particular, class expressions can be used as slot fillers, whereas in typical frame languages slot fillers are restricted to being class (or individual) names.
- A slot-filler pair (called a slot constraint) can itself be treated as a class: it can be used anywhere that a class name can be used, and can be combined with other classes in class expressions.
- Class definitions (frames) have an (optional) additional field that specifies whether the class definition is primitive (a subsumption axiom) or non-primitive (an equivalence axiom). If omitted, this defaults to primitive.
- Different types of slot constraint are provided, specifying value restriction, existential quantification and various kinds of cardinality constraint. Global slot definitions are extended to allow the specification of supers lots (subsuming slots) and of properties such as transitive and symmetrical.
- Unlike many frame languages, OIL has no restriction on the ordering of class and slot definitions, so classes and slots can be used before they are "defined". This means that OIL Ontologies can contain cycles.
- In addition to standard class definitions (frames), which can be seen as DL axioms of the form $CN \sqsubseteq C$ and $CN \equiv C \equiv$ where CN is a concept name, OIL also provides axioms for asserting Disjointness, equivalence and coverings with respect to class expressions. This is equivalent to providing general inclusion (or equivalence) axioms, i.e., axioms of the form $CN \sqsubseteq C$ ($C \equiv D$), where both C and D may be non-atomic concepts.

Here an example of description using OIL language can make a highlight on the formulation of description. The syntax of OIL use the term "*slot-def*" to define role name and "*inverse*" for the inversed role. The concept are viewed like classes it's why every concept definition is placed after "*class-def*"..Sf.

```
name "Family"
documentation "Example ontology describing family relationships"
definitions
slot-def hasChild
inverse isChildOf
class-def defined Woman
subclass-of Person Female
class-def defined Man
subclass-of Person not Woman
class-def defined Mother
subclass-of Woman
slot-constraint hasChild
```

Figure 3 family ontology described in OIL[13]

The DLs contribution in web based systems is provided by many languages either than OIL like DALM or DALM+OIL.

5.4. Medical Informatics

There are more than one application in the field of medical informatics that can be concerned by the use of description logics like representation formalism “terminology, intelligent user interfaces, decision support and semantic indexing, language technology, and systems integration.” [13]

For every kind of application knowledge are used. That means; powerful and expressive formulation is required for effective system. It must respond to lot of constraints coming from requirements like size, complexity, connectivity, and the wide range of granularity.

The two best known efforts – *OpenGalen* and *Snomed-rt* – both use idiosyncratic Description Logics with generally limited expressivity but specialized extensions to cope with issues around part–whole and other transitive relations. There is also a conflict between the needs for re-use and the requirement for easy understandability by domain expert authors. *OpenGalen* has coped with this conflict by introducing a layered architecture with a high level “Intermediate Representation” which insulates authors from the details of the Description Logic, which is treated as an “assembly language” rather than the primary medium for expressing the ontology.[13]

5.5. Data bases

A knowledge base or a database booth of them existed for a goal of saving related data of a model created by abstraction from the real word in coherent form. These resources should be well structured for facilitating the manipulation. Each one of them is created using specific tools with respect to integrity constraint, facilities of access, coherence and significance.

But the thing that makes difference between KB and DB is that in the first the formulation must enable the deduction of consequent knowledge from the explicitly represented ones, where in the second we need just that “the explicitly represented knowledge” be efficiently retrievable. From that point of view, it is clear that the DB need only a query language. However, for a KB there is more than a querying data because of the integration of the semantic aspect. This last enable more possibilities such as question answering, schema constraint checking, inference, .SF.

The main aimed goal for the research on databases is how to make DB knowledgeable, it is why is required some kind of height level languages called *Semantic Modelling Languages*. In these languages the Universe of discourse (reality to be modelled) is viewed like population of entities which are enter-related by many relations and each one is described by some atomic value called the attributes.

These transformations are based on the known Relational Model used such as beginning point. From this description, the semantic phase defines the Logical Schema which provides knowledge about data structure, relations and constraints that must be hold. But it's to be noted that there are works based on the object oriented formalization.

PART I: BACKGROUND

CHAPTER 03: DLS IMPLEMENTATIONS AND CURRENTS SYSTEMS

1. DESCRIPTION LOGICS SYSTEMS IMPLEMENTATION

The earliest description logics systems were based on two kinds of operation called *tell & ask*. The first one is for telling knowledge to the system where the second is used for asking about what was previously told to the system or there logical consequences.

The implemented DLs knowledge representation systems can be categorized in tow generations. This is coming from the additional represented aspect (augmentations aiming more expressivity) like the inverse Roles and composition of roles.

The first generation are much related to the view inspired from the cognitive behaviour on what focus the predecessor of DLs systems (SNs, Frame systems). These systems had the finality of finding a simple formalization for knowledge being represented and possibility of inferring implicit knowledge intuitively. In 1975 the beginning of development of the named system KL-one which was not in first steps seen like logical based approach.

"KL-One started the era of logic based representation systems which can be used to formalize application problems as inference problems over the constructs supported by the representation language." [13]

For that, the KL-one based the representation on primitives which are concepts and the roles, where the meaning of the concept is defined from the super concepts and restrictions that associate it with others concepts. These restrictions are built using defined roles.

The first Algorithms developed were called classifiers, because they were interested especially to the detection of the subsumption hierarchy (so called parent-child relation). An additional reasoning component was the "realize". This last check for each individual in the ABox the most specific concept in which the individual is an instance.

Another developed system called KRYPTON. Based initially on the first order theorem prover and developed to respond to other specific purposes related to terminological and assertional reasoning. "Krypton can be regarded as one of the first efforts in combining knowledge representation and theorem-proving techniques but was not used for industrial applications"[13].

Thing to be noted that KRYPTON is deferent because "the focus of Krypton was not on the structures to be maintained by the system but was centred around the question about what the system should do for the user, i.e., what services should be made available".[13]

For the inference this system proposes for the TBox services such as consistency checking, subsumption and Disjointness and the most specific concept is computable. And for the ABox reasoning consistency, realization, instance checking and instance retrieval. The KRYPTON there is a specific characteristic which is "the idea that the user should only know, at some level not dependent on implementation details, what questions the system is capable of answering and what operations are permitted that allow new information to be provided to it".[13]

Others systems were developed at the same time with KRYPTON such as: Nikl, Penni, Kl-Two. The first was developed to success KL-one with augmentations enabling the roles classification by subsumption, with particularity that "the algorithms in the Nikl classifier were faster in the average case because "obvious" information was exploited to a larger degree" and that the provided Algorithm for subsumption was incomplete. This second disadvantage has been omitted by the assertional reasoning add with the Penni system. But with more augmentation concerning the quantificational reasoning components a system rise named KL-two.

Another important system developed to omit disadvantages seen in precedent systems is called KANDOR. "Basically, KANDOR supported conjunction, value restriction and number restrictions as concept-forming operators. In minimum number restrictions, range restricted roles could be used (hence, qualified minimum number restrictions were allowed"[13]

The second generation of DLs systems was created for specific purpose which is being usable in application domains. The first discussed system is called CLASSIC which is basically "CLASSIC system supported the logic ALNFh−1 with TBoxes and ABoxes plus facilities for dealing with numbers..... We use the lowercase letter h to indicate that Classic supports role inclusion but not role conjunction, i.e., Classic supports "single-inheritance" role hierarchies. Classic is available for research purposes. Implementation languages for Classic are Common-Lisp." [13]

CLASSIC provided complete subsumption reasoner after some augmentations from the basic CLASSIC to the full version. “Furthermore, CLASSIC provides simple support for closed-world reasoning”

LOOM is another known system, implemented with respect of the DLs $\mathcal{ALCQRIFO}$ augmented by the addition of role conjunction. “The current version of Loom is implemented in Common-Lisp and is available for research purposes. A new system (called Power-Loom) for Common-Lisp as well as C and Java-based platforms can be licensed as well.”[13]

There are two goals that motivated LOOM development. The first one is to incorporate an expressive query language for querying the ABox, and the second is to support the rule-based programming.

This system is characterized that is more expressive than the CLASSIC but the Algorithms proposed for concept consistency and subsumption checking are incompletes. but “The arguments for the Loom approach can be summarized as follows: The intractability of the representation language can hardly be avoided if the requirements of users are to be fulfilled. Therefore, the idea is to support the features in one system rather than as a set of application-specific ad hoc supplements]...[Obviously, incompleteness is no problem as long as the answers of the inference system are interpreted in the right way (i.e., “no” answers should not be trusted).”[13]

Others proposed DLs environments are called BACK and FLEX developed for the same initial goals of the Loom system. BACK is abbreviated from Berlin Advanced Computational Knowledge, When FLEX is known for his flexibility of reasoning process.

“The Description Logic of the initial Back system can be called $\mathcal{ALQR}^{-1}$. There was also support for reasoning with numbers and attribute sets. Research on the inference algorithms for the basic Back language stimulated the development of theoretical results on the complexity of concept consistency reasoning.” [75]

“In order to provide a hybrid representation language, BACK was one of the first systems in which TBox concept terms could also be used in an ABox to assert, e.g., disjunctive information about individuals. In addition, distinct individuals were assumed to denote distinct objects."[13]

FLEX uses the description logics $\mathcal{ALCQRIFO}$ unequalled and equalled number restrictions. The first version of FLEX was developed on PROLOG but successors called FLEX++ was implemented in C++. This implementation was faster but not enough expressive and used incompletes Algorithms.

"Experiences with system implementations indicated that either limited expressiveness or incompleteness of reasoning could possibly lead to problems in applications. Therefore, other researchers investigated the implementation of systems based on sound and complete algorithms." [13] Note here that soundness means that reasoner prove just valid formulas with respect to its semantic.

With the tableau Algorithms new DLs system architecture has been constructed. This leads to implemented systems such as KRIS and CRACK.

KRIS is presented like DL system that guaranties completeness and soundness in reasoning process with expressive DLs which is $\mathcal{ALCNF}$, in where is allowed the numbers restriction and role conjunction. The KRIS was implemented in Common-Lisp and represents first system providing complete and sound Algorithm for reasoning on both TBox and ABox for expressive description logics. Add to that KRIS act on a concrete domain.

"One of the main results of the KRIS project was that sound and complete inference algorithms are an important starting point for research on optimized sound and complete algorithms for practical system development."[13]

The system CRACK which is a description logics system typed ALCRIFO has as main characteristic: that the inference Algorithm provided should be able to deal with individuals in concept terms. It represents a better than its predecessors by the goals achieved "but the inference techniques employed, which had been developed for (manually) deriving decidability results, e.g., with tableau algorithms, were not suited for direct implementation."[13] The decidability problem roses as a beginning new horizon for performing new systems to achieve the aimed systems.

2. CURRENT DESCRIPTION LOGICS REASONER

A number of changes are applied on reasoner for the aim of performing and optimizing. As result a number of existing reasoner are characterized in a list established by searchers in Manchester University [73] with links to source as verification this list is presented in alphabetic order like in follow.

BaseVISor, hosted at VIStology, Inc., published under BaseVISor is licensed for academic and research use free of charge; all other uses require a commercial license. BaseVISor is a versatile, highly efficient forward-chaining inference engine, based on a Rete network, specialized to handle facts in the form of RDF triples with support for OWL 2 RL and XML Schema Data types. It is written in Java and can be run as a standalone application or be embedded in existing applications. BaseVISor is optimized for ontological and rule-based reasoning. Users can develop procedural attachments that can be invoked from within a rule to perform a complex calculation, access a Web service or execute new processes, complementing a large set of BaseVISor's built-in functions. BaseVISor is licensed for academic and research use free of charge. *Supported interfaces:* Command Line, Other. *Supported reasoning services*: realisation, classification, satisfiability, conjunctive query answering, entailment, consistency. *Supported syntaxes*: RDF/XML, OWL/XML, All OWL API.

BUNDLE, hosted at University of Ferrara, published under AGPL license Version 3. BUNDLE ("Binary decision diagrams for Uncertain reasoNing on Description Logic thEories") is a probabilistic reasoner based on Pellet. It extends Pellet in order to allow the computation of the probability of queries from a probabilistic knowledge base that follows the DISPONTE probabilistic semantics. *Supported interfaces*: Jena, Command Line, OWL API. *Supported reasoning services*: satisfiability, entailment, consistency, explanation. *Supported syntaxes*: Turtle, RDF/XML, Krss2, Latex, OWL/XML, Functional, All OWL API, Manchester.

CEL, hosted at Technische Universität Dresden, published under Apache License 2.0 (CEL) / GNU Lesser General Public License 3.0 (CEL Plug-in). CEL is a free (for non-commercial use) Lisp-based reasoner for EL+. It implements a refined version of a polynomial-time classification algorithm and supports new features like module extraction and axiom pinpointing. It is distributed with a Java-base adapter to use the OWL API and to be used as a Protégé plug-in. *Supported interfaces:* Protege, Command Line, OWL API. *Supported reasoning services:* classification, satisfiability, consistency. *Supported syntaxes:* Turtle, RDF/XML, Krss2, OBO, OWL/XML, Functional, All OWL API, Manchester.

Chainsaw, hosted at The University of Manchester, published under LGPL. Chainsaw is a free (LGPL) OWL 2 DL reasoner for very large ontologies. It uses a modular decomposition to tackle the high complexity of the reasoning. Uses delegate reasoner(s) to perform single reasoning tasks. *Supported interfaces:* Protege, OWL API. *Supported reasoning services:* realisation, classification, satisfiability, entailment, consistency. *Supported syntaxes:* All OWL API.

Clipper, hosted at Vienna University of Technology, published under Apache 2.0. Clipper is a Reasoner for conjunctive query answering over Horn-SHIQ ontology via query rewriting. *Supported interfaces:* OWL API. *Supported reasoning services:* conjunctive query answering. *Supported syntaxes:* All OWL API Download, Core publication.

DBOWL, hosted at University of Malaga, published under GNU General Public License. DBOWL is a scalable reasoner for OWL ontologies with very large Aboxes. DBOWL stores ontologies and classifies instances in Named Classes and Properties using relational database technology. It combines relational algebra expressions and fixed-point iterations in order to compute the closure of the ontology, called the knowledge base creation. It also supports SPARQL queries. *Supported interfaces:* Other. *Supported reasoning services:* classification, satisfiability, conjunctive query answering, consistency. *Supported syntaxes:* RDF/XML.

DeLorean, hosted at Not given, published under NA. DeLorean is a fuzzy rough Description Logic reasoner. It supports a fuzzy rough extension of OWL 2. *Supported interfaces:* Command Line, Other. *Supported reasoning services:* realisation, classification, satisfiability, entailment, consistency, Service Fuzzy. *Supported syntaxes:* NA.

DistEL, hosted at Wright State University, published under NA. DistEL is a distributed reasoner that runs on a cluster of machines. It has support for a major part of OWL 2 EL profile. As of now, it has support for only classification task. *Supported interfaces:* Command Line. *Supported reasoning services:* realisation, classification. *Supported syntaxes:* NADownload, Core publication.

DRAOn, hosted at University of Paris 8, IUT of Montreuil, published under LGPL. DRAOn is an OWL reasoner that supports distributed reasoning over a networked ontologies. It is based on local reasoners that implement an algorithm building compressed models. *Supported interfaces:* OWL API. *Supported reasoning services:* entailment, consistency. *Supported syntaxes:* All OWL API.

DReW, hosted at Vienna University of Technology, published under Apache 2.0. DReW is a reasoner for DL-Programs over Datalog-rewritable Description Logics. The algorithm is based on rewriting to Datalog. *Supported interfaces:* Command Line, OWL API. *Supported reasoning services:* conjunctive query answering. *Supported syntaxes:* All OWL API.

ELepHant, hosted at Not given, published under Apache License, Version 2.0. ELepHant is a consequence-based reasoner that currently supports part of the OWL 2 EL fragment for the reasoning tasks classification, consistency and realization. Its aim is to provide lightweight and performant reasoning for the full OWL 2 EL fragment. *Supported interfaces:* Command Line. *Supported reasoning services:* realisation, classification, consistency. *Supported syntaxes:* Functional.

ELK, hosted at University of Ulm, Germany, published under Apache 2. ELK is a reasoner for OWL 2 ontologies that currently supports a part of the OWL EL ontology language. *Supported interfaces:* Protege, Command Line, OWL API. *Supported reasoning services:* realisation, classification, satisfiability, entailment, consistency. *Supported syntaxes:* Functional.

ELOG, hosted at Not given, published under GNU GPL v3. ELOG is a reasoner for log-linear description logics, a probabilistic logical formalisms that combines description logics and log-linear models. *Supported interfaces:* OWL API. *Supported reasoning services:* realisation, entailment. *Supported syntaxes:* RDF/XML, OWL/XML, All OWL API.

FaCT++, hosted at The University of Manchester, published under LGPL. FaCT++ is a free (LGPL) highly optimised open-source C++-based tableaux reasoner for OWL 2 DL. *Supported interfaces:* Protege, Command Line, OWL API, Other. *Supported reasoning services:* realisation, classification, satisfiability, entailment, consistency. *Supported syntaxes:* NA.

fuzzyDL, hosted at ISTI – CNR, published under NA. fuzzyDL is a free Java/C++ based reasoner for fuzzy SHIF with concrete fuzzy concepts (explicit definition of fuzzy sets + modifiers). It implements a tableau + Mixed Integer Linear Programming optimization decision procedure to compute the maximal degree of subsumption and instance checking w.r.t. a general TBox and Abox. It supports Zadeh semantics, Lukasiewicz semantics and is backward compatible with classical description logic reasoning. *Supported interfaces:* Protege. *Supported reasoning services:* satisfiability, entailment, consistency, Service Fuzzy. *Supported syntaxes:* OWL/XML.

HermiT, hosted at University of Oxford, published under LGPL. HermiT is an OWL 2 DL reasoner — to my knowledge, one of the few such systems that attempts (modulo bugs) to fully and correctly support the OWL 2 DL specification. *Supported interfaces:* Protege, Command Line, OWL API. *Supported reasoning services:* realisation, classification, satisfiability, entailment, consistency. *Supported syntaxes:* All OWL API.

jcel, hosted at Technische Universität Dresden, published under Apache License 2.0 and GNU Lesser General Public License 3.0. jcel is a free open-source Java-based reasoner for EL+ and supports parts of the OWL 2 EL profile. It implements a polynomial-time modular consequence-based algorithm for general TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval). It supports the OWL API and can be used as a Protégé plug-in. *Supported interfaces:* Protege, Command Line, OWL API. *Supported reasoning services:* classification, satisfiability, entailment, consistency. *Supported* *syntaxes:* Turtle, RDF/XML, Krss2, OBO, OWL/XML, Functional, All OWL API, Manchester.

JFact, hosted at The University of Manchester, published under LGPL. JFact is a pure Java port of FaCT++, with versions for Owlapi 3.x and 4.x. It is kept in step with FaCT++ and updated regularly. It has been used on Android devices with Owlapi 3.5. It is available packaged as a Protégé plugin, for Protégé 4.3 and 5. *Supported interfaces:* OWL API. *Supported reasoning services:* realisation, classification, satisfiability, entailment, consistency. *Supported syntaxes:* NA.

Konclude, hosted at University of Ulm, derivo GmbH, published under LGPL 2.1. Konclude is a parallel, high-performance reasoner for the Description Logic SROIQV(D). It is implemented in C++ and uses a reasoning technique that is based on a highly optimized tableau algorithm assisted by a completion-based saturation procedure. At the moment, it supports many standard reasoning tasks such as consistency checking, classification, realisation and can be used via command line or OWLlink. *Supported interfaces:* OWLLink, Command Line. *Supported reasoning services:* realisation, classification, satisfiability, consistency. *Supported syntaxes:* OWL/XML, Functional.

LiFR, hosted at Centre for Research and Technology Hellas (CERTH), published under GNU LGPL. LiFR is a Lightweight Fuzzy DL Reasoner, capable of performing in resource-constrained devices. It supports f-DLP (Zadeh fuzzy operators) by translating DL axioms to first order clauses and by using the hyper-tableaux calculus. It accepts as input a variant of the KRSS syntax. *Supported interfaces:* Other. *Supported reasoning services:* satisfiability, entailment, consistency, Service Fuzzy. *Supported syntaxes:* NA.

Mastro, hosted at Sapienza University of Rome, published under Executable available for research purposes. For other needs, please contact the developers.. Mastro is a free (for non-commercial use) Java-based reasoner for OWL 2 QL and ontology languages of the DL-Lite family. The main reasoning services provided by Mastro are: conjunctive query answering, consistency checking, instance checking, and TBox reasoning (logical implication, classification, and satisfiability). Mastro works in two settings: the classical setting with ontologies composed by a TBox and an ABox, and the Ontology-based Data Access setting in which the ontology is connected to external data management systems through semantic mappings that associate SQL queries over the external data sources (typically relational DBs) to the elements of the ontology. It supports the OWL-API and comes with its own Java-based interface. *Supported interfaces:* Command Line, OWL API. *Supported reasoning services:* classification, satisfiability, conjunctive query answering, entailment, consistency. *Supported syntaxes:* RDF/XML, OWL/XML, Functional, All OWL API, Manchester.

MORe, hosted at University of Oxford, published under GNU Lesser GPL. MORe uses module extraction techniques to classify ontologies combining reasoners especially optimised for different OWL 2 profiles. *Supported interfaces:* Protege, Command Line, OWL API. *Supported reasoning services*: classification, satisfiability. *Supported syntaxes:* All OWL API.

ontop, hosted at Free University of Bozen-Bolzano, published under Apache 2.0. Ontop is a platform to query databases as Virtual RDF Graphs using SPARQL. It's extremely fast and is packed with features. *Supported interfaces:* Protege, OWL API, Other. *Supported reasoning services:* realisation, conjunctive query answering. *Supported syntaxes:* All OWL API.

Pellet, hosted at Clark & Parsia, LLC, published under AGPL v3. Pellet is a free open-source Java-based reasoner for OWL 2 and SWRL. It supports the full expressivity of SROIQ Description Logic, user-defined datatypes and DL-safe rules. Pellet uses a tableau-based decision procedure to provide many reasoning services (subsumption, satisfiability, classification, instance retrieval, conjunctive query answering) along with the capability to generate explanations for the inferences it computes. It has bindings for both the OWL-API and the Jena libraries. *Supported interfaces:* Jena, Protege, Command Line, OWL API. *Supported reasoning services:* realisation, classification, satisfiability, conjunctive query answering, entailment, consistency, explanation. *Supported syntaxes:* Turtle, RDF/XML, Krss2, OWL/XML, Functional, Manchester.

Racer, hosted at Concordia University, Montreal, Canada; University of Lübeck, Germany;, published under BSD-3. Racer (Renamed ABox And Concept Expression Reasoner) is a knowledge representation system that implements a highly optimized tableau calculus for the description logic SRIQ(D). Racer provides implementations of standard reasoning problems for T-boxes and A-boxes. In addition, some non-standard inference services are provided, such as, e.g., logical abduction. Racer also provides the powerful and semantically well-defined conjunctive query language nRQL (new Racer Query Language, to be pronounced as niracle and heard as miracle), which also supports negation as failure, numeric constraints w.r.t. attribute values of different individuals, substring properties between string attributes, etc. It has convenient APIs for accessing its reasoning services from within Common Lisp and Java. Racer is distributed under the BSD 3-clause license. *Supported interfaces:* OWLLink, Protege, Command Line, OWL API, Other. *Supported reasoning services:* realisation, classification, satisfiability, conjunctive query answering, consistency, explanation. *Supported syntaxes:* RDF/XML, OWL/XML, Functional, All OWL API.

RDFox, hosted at University of Oxford, published under Oxford Academic Licence. RDFox is a RDF triple store and parallel datalog/SWRL reasoner. It supports most SPARQL builtins (used in BIND and FILTER). The data is separately supplied in Turtle or N-Triple format. *Supported interfaces:* Other. *Supported reasoning services:* conjunctive query answering, entailment.*Supported syntaxes:* Turtle, All OWL API.

RuQAR, hosted at Poznan University of Technology, published under NA. RuQAR is a free (for non-commercial use) tool that provides the ABox reasoning and conjunctive query answering with OWL 2 RL ontologies executed by forward chaining rule reasoners. The tool implements also a method of transforming OWL 2 ontologies into Jess and Drools engines. *Supported interfaces:* Other. *Supported reasoning services:* conjunctive query answering. *Supported syntaxes:* RDF/XML, OWL/XML.

Snorocket, hosted at CSIRO, published under Apache 2.0. Snorocket is an open source, high-performance ontology reasoner that supports a subset of the OWL EL profile. *Supported interfaces:* Protege, OWL API. *Supported reasoning services:* classification, consistency. *Supported syntaxes:* All OWL API.

TReasoner, hosted at Tyumen State University, published under GNU GPL 2. TReasoner is a free tableau algorithm based reasoner. It is written on Java, uses OWL API and supports SROIQ(D) logics. It was created for developing and new optimization techniques for tableau algorithm. This reasoner used for many applied tasks: from database validation to automated timetabling. *Supported interfaces:* Command Line, OWL API. *Supported reasoning services:* classification, satisfiability, consistency. *Supported syntaxes:* All OWL API.

TRILL, hosted at University of Ferrara, published under The Artistic License 2.0. TRILL ("Tableau Reasoner for descrIption Logics in Prolog") is a probabilistic reasoner which implements a tableau algorithm in Prolog to find the set of all the explanations and compute the probability of a query. TRILL is available for both Yap Prolog and SWI-Prolog. *Supported interfaces:* Other. *Supported reasoning services:* satisfiability, entailment, consistency, explanation. *Supported syntaxes:* NA Download, Core publication.

TRILLP, hosted at University of Ferrara, published under The Artistic License 2.0. TRILLP is based on the reasoner TRILL written in Prolog. TRILLP is a tableau probabilistic reasoner which computes a Boolean formula that represents the set of the explanations of the query. From this formula the probability of the query is then computed. *Supported interfaces:* Other. *Supported reasoning services:* satisfiability, entailment, consistency. *Supported syntaxes:* NA.

TrOWL, hosted at University of Aberdeen, published under AGPL for open source applications. TrOWL is a Tractable reasoning infrastructure for OWL 2. TrOWL supports not only standard TBox and ABox reasoning, but also conjunctive query answering in SPARQL. *Supported interfaces:* Jena, Protege, Command Line, OWL API. *Supported reasoning services:* realisation, classification, satisfiability, conjunctive query answering, entailment, consistency. *Supported syntaxes:* NA.

WSClassifier, hosted at University of New Brunswick, Canada, published under MIT. WSClassifier is a free prototypical (under MIT license) Java reasoner for classifying DL ALCHOI(D-) with limited datatype support, using a hybrid of the consequence-based reasoner ConDOR and the hypertableau-based reasoner HermiT. It supports OWL-API. *Supported interfaces:*OWL API. *Supported reasoning services:* classification. *Supported syntax*es: All OWL API.

Due to the same reference (official web site of university of Manchester) there are other existing reasoners we arrange them in the table:

Table 5 Some Description Logic reasoners[01].

COROR	**RacerPro**	***SAT**
BACK	CB	Cerebra Engine
CICLOP	CLASSIC	Condor
CRACK	DLP	Fact
FLEX	HAM-ALC	K-REP
Bossam	KRIS	LOOM
MSPASS	QuOnto	SHER
YAK	OWLGres	Pronto
DLEJena	F-OWL	Fresg
OWLer	OntoMinD	Screech
REQUIEM	YARR	Kaon2
Elly	SoftFacts	

PART II: PROPSED APPROACH

CHAPTER 01: DERIVATION OF REPRESENTATION METHOD

1. THE SUBSUMPTION HIERARCHICAL ATTRIBUTE

The subsumption is a relationship that associates two sets of individuals satisfying concepts. It means that one of these two sets is a sub set of the other and we say the first one subsumes the second which is called the subsumed. The extension of this structure gives a sort of a hierarchy of concepts. This last represents a dependency graph based on the subsumption. This graph allowed the detection of the transitive closer of subsumption relationship.

$D \sqsubset C$ $B \sqsubset C$

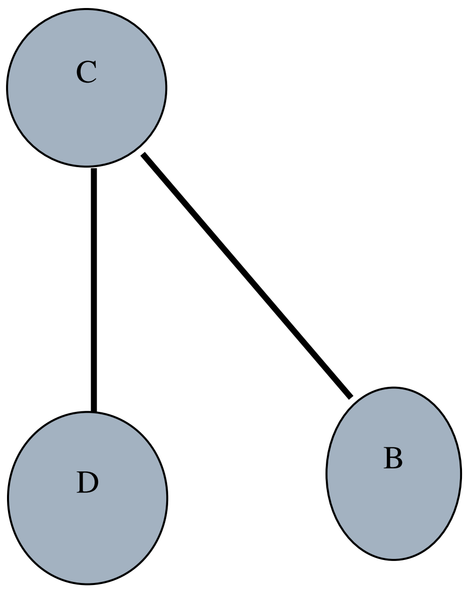


Figure 4 The subsumption hierarchical structure

The resulted graph can be used to transform the representation to Vectorial form based on two major ideas: a sequential codification of nodes in one level (integration order). and the second one represent gives the level in which concept (represented by node). These structures (called *SHA* for *subsumption hierarchical attribute*) are created like in following (see figure 4).

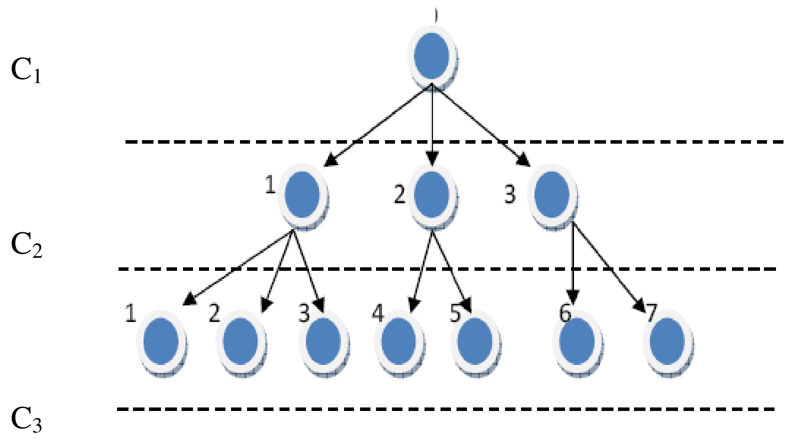


Figure 5 the subsumption hierarchical attribute creation

The vector which represents the path from the root (more general concept) to the concerned node is called a *SHA* of the concerned concept. Every concept is associated to one or more vector relatively to existence of paths. The meaning of this representation can reflect all direct or indirect subsumption relationships between represented concepts. Because of this property the use of *SHA* enable the accomplishment the checking of KB coherence. Also it allows the inference of implicit knowledge. These lasts are confirmed by the property reducing all axioms (concepts definitions) to subsumption.

2. COHERENCE CHECKING

The definition of domain uses terminology noted Γ, which amount the system to define concepts possibly in term of others already defined, this introduction of means concepts must verify coherence constraints between defined concepts. From a logical point of view, a concept makes a sense for us if there is some interpretation that satisfies the axioms of Γ (that is, a model of Γ) such that the concept denotes a nonempty set in that interpretation. [13]

Satisfiability: in a terminology of a domain called T if there is a model Γ of T such that C Γ is nonempty model of C. Which means that the concept is satisfied in T if C defined in the terminological box contains at least an under concepts or an assertion if it is a bottom concepts. With the integration of *SHA* we can check the satisfiability by looking for the right side of his *SHA* from the beginning to the first extension component (first 0 in left side). If this part exists in at least one vector of another concept, or if there is at least one individual defined as assertion of the concerned concept.

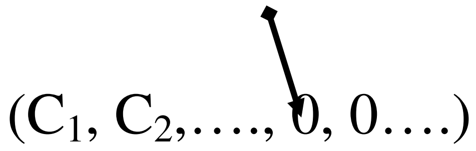


Figure 6 SHA for checking the satisfiability

Subsumption: as defined in precedent the subsumption represent a kind of inclusion relation between two sets of individuals –concepts such as we can say that a concept A is subsumed by a concept D if and only if $\forall I \subset A \Rightarrow I \subset C$ and we write in this case $C \subseteq D$ with respect to a domain T we write $T|=C \subseteq D$. That means that D is subset of C.

Using the *SHA* the right side of this attribute for the D must be equal the right side of C, but for C in the place of the first extension component ("0") it must have the number of the integration order of C.

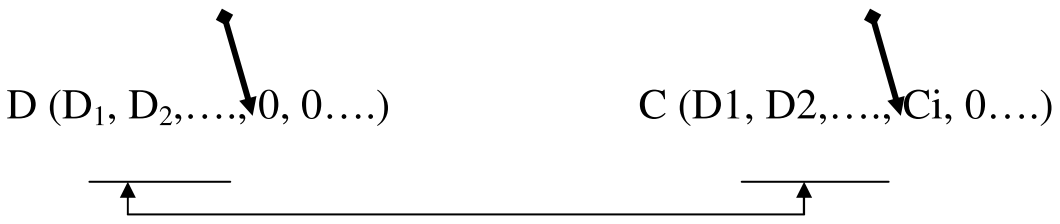


Figure 7 SHA for checking the Subsumption

Equivalence: tow concept can compared and we say that they are C1, C2, …. equivalent if and only if their extensions are equal with respect to every model [24]. And represent formally in propositional way by: $A \equiv C \Leftrightarrow (A \subseteq C) \wedge (C \subseteq A)$ which means that A is subsumed by C and C is subsumed by A at the same time.

Using SHA, for every SHA having the components of C in the left side there is another SHA having in the left side component of D with same right side.

D (D$_1$, D$_2$, A$_1$, A$_2$….)
C (C1, C2, A1, A2….)

Figure 8 SHA for checking the Equivalence

Disjointness: two concepts are disjoint if there are comment individuals which are elements of the two concepts, we represent formally C, D are disjoints $\Rightarrow C \cap D = \phi$. For the SHA, all SHAs representing C are totally different from those representing D ie. No common components in right sides (see figure 9 with $A_i \neq C_i$).

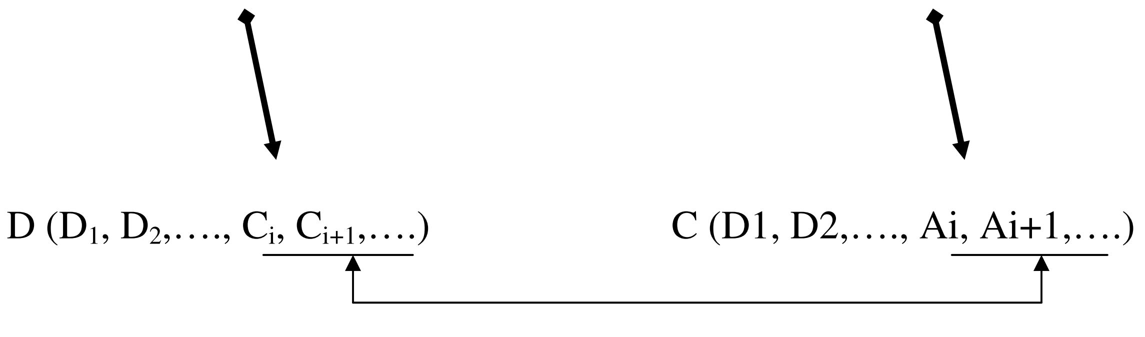

D (D$_1$, D$_2$,…., C$_i$, C$_{i+1}$,….)
C (D1, D2,…., Ai, Ai+1,….)

Figure 9 SHA for checking the Disjointness

All reasoning services can be reduced to subsumption based representation and we admit that for two concepts C, D we have:

C is unsatisfiability $\Leftrightarrow$ C is subsumed by $\perp$ (bottom concepts)

C and D are equivalent $\Leftrightarrow$ C is subsumed by D and D is subsumed by C

C and D are disjoint $\Leftrightarrow$ C∩D is subsumed by $\perp$

This reduction facilitate understanding checking ideas, because it makes them in sets form which allowed obtaining decision procedures about this four constraints noted above. And of course it's evident to confirm that the $\mathcal{SHA}$ is useful in the reasoner implementation because it is derived from subsumption.

3. $\mathcal{SHA}$ GENERATION PROCESS [76]

The first step of the $\mathcal{SHA}$ generation process is the direct subsumption detection which consists in representing the axioms defined by equivalence relation in subsumption based form. This representation is based on implications existing between the two kinds of representation. Like sets treatment each definition using constructors can be reduced to a group of subsumption (inclusion) relation as follows.

$$\mathrm{C}^{\mathrm{I}} \equiv \mathrm{D}^{\mathrm{I}} \cap \mathrm{B}^{\mathrm{I}} \Rightarrow (C^{\mathrm{I}} \subset D^{\mathrm{I}}) \wedge (C^{\mathrm{I}} \subset B^{\mathrm{I}})$$

$$\mathrm{C}^{\mathrm{I}} \equiv D^{\mathrm{I}} \cup \mathrm{B}^{\mathrm{I}} \Rightarrow (D^{\mathrm{I}} \subset \mathrm{C}^{\mathrm{I}}) \wedge (\mathrm{B}^{\mathrm{I}} \subset \mathrm{C}^{\mathrm{I}})$$

$$\mathrm{C}^{\mathrm{I}} \equiv \bar{\mathrm{B}}^{\mathrm{I}} \Rightarrow \mathrm{B}^{\mathrm{I}} \cap \mathrm{C}^{\mathrm{I}} \equiv \Phi$$

Reducing The complex expressions to simple subsumption expressions needs some iteration based on other rules after the application of this specified below as initial treatments. For reasons of expressiveness, indicators, to represent the infinite and the finite sub-domain, are needed. The infinite sub-domain comes from the relations of subsumption between a concept and an expression like.

$$\mathrm{C}^{\mathrm{I}} \subset (D^{\mathrm{I}} \cup \mathrm{B}^{\mathrm{I}})$$

Which means that there is a part of a set which can be a small part or big part or the whole of the super set. This probable forms cause less expressive representation and make useful the use of the infinity indicators for each sub-domain.

$$(D^{\mathrm{I}} \cap \mathrm{B}^{\mathrm{I}}) \subset \mathrm{C}^{\mathrm{I}} \Rightarrow \exists A, (D^{\mathrm{I}} \cap \mathrm{B}^{\mathrm{I}}) \cap \mathrm{C}^{\mathrm{I}} \equiv A \Rightarrow (A \subset D^{\mathrm{I}}) \wedge (A \subset \mathrm{B}^{\mathrm{I}}) \wedge (A \subset \mathrm{C}^{\mathrm{I}})$$

$$(D^{I} \cup B^{I}) \subset C^{I} \Rightarrow (D^{I} \subset C^{I}) \wedge (B^{I} \subset C^{I})$$

$$C^{I} \subset (D^{I} \cap B^{I}) \Rightarrow (C^{I} \subset D^{I}) \wedge (C^{I} \subset B^{I})$$

$$C^{I} \subset (D^{I} \cup B^{I}) \Rightarrow (C^{I} \subset D^{I}) \vee (C^{I} \subset B^{I}) \vee (C^{I} \subset (D^{I} \cap B^{I}))$$

For the roles level, the role is seen as binary relation between two sets of individuals. The expression R.C corresponds to the individuals in relation with C by R. For example, the sentence "individuals having a female child" is presented ∃hasChild.Female, and the sentence "individuals all of whose children are female" is presented ∀hasChild.Females[13]. By the same way the number restriction is represented to integrate cardinalities. (See figure10).

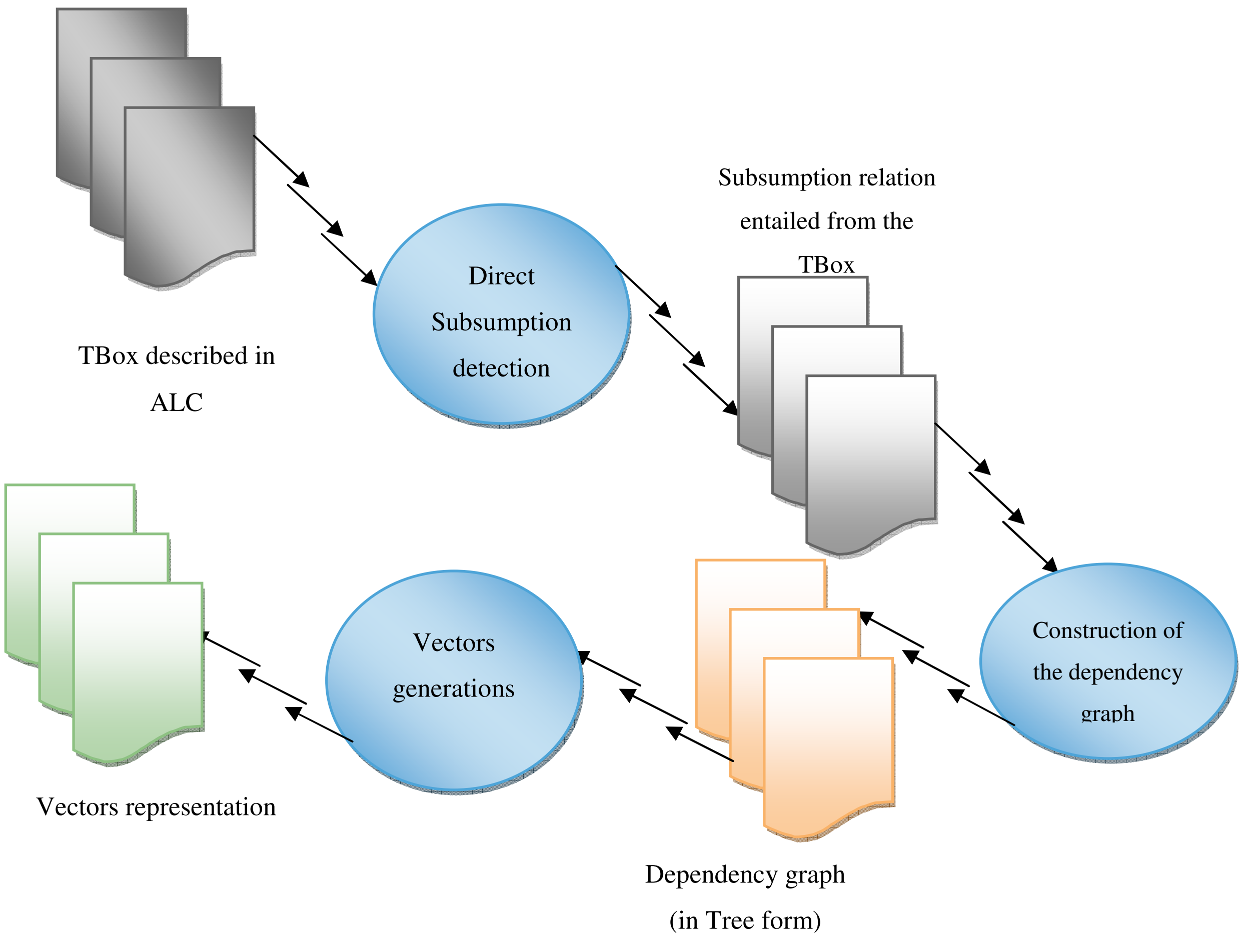

TBox described in ALC
Direct Subsumption detection
Subsumption relation entailed from the TBox
Construction of the dependency graph
Dependency graph (in Tree form)
Vectors generations
Vectors representation

Figure 10 SHA generation process

This proposed process is completely computable with high effectiveness degree. But it's to be mentioned that the implementation like all implementations of DLs systems use for the integration interface a simplified syntax. In next we present the syntax used based polish notation for expressing definition axioms.

S → % OP | C

OP → CON(S,S)| Neg (S)| T R.C|

T → EX | UN |<=Number | >=Number

The terminal alphabet is giving like: Conj for conjunction, Disj for Disjunction,

CON→ Conj | Disj

C → <concept name>

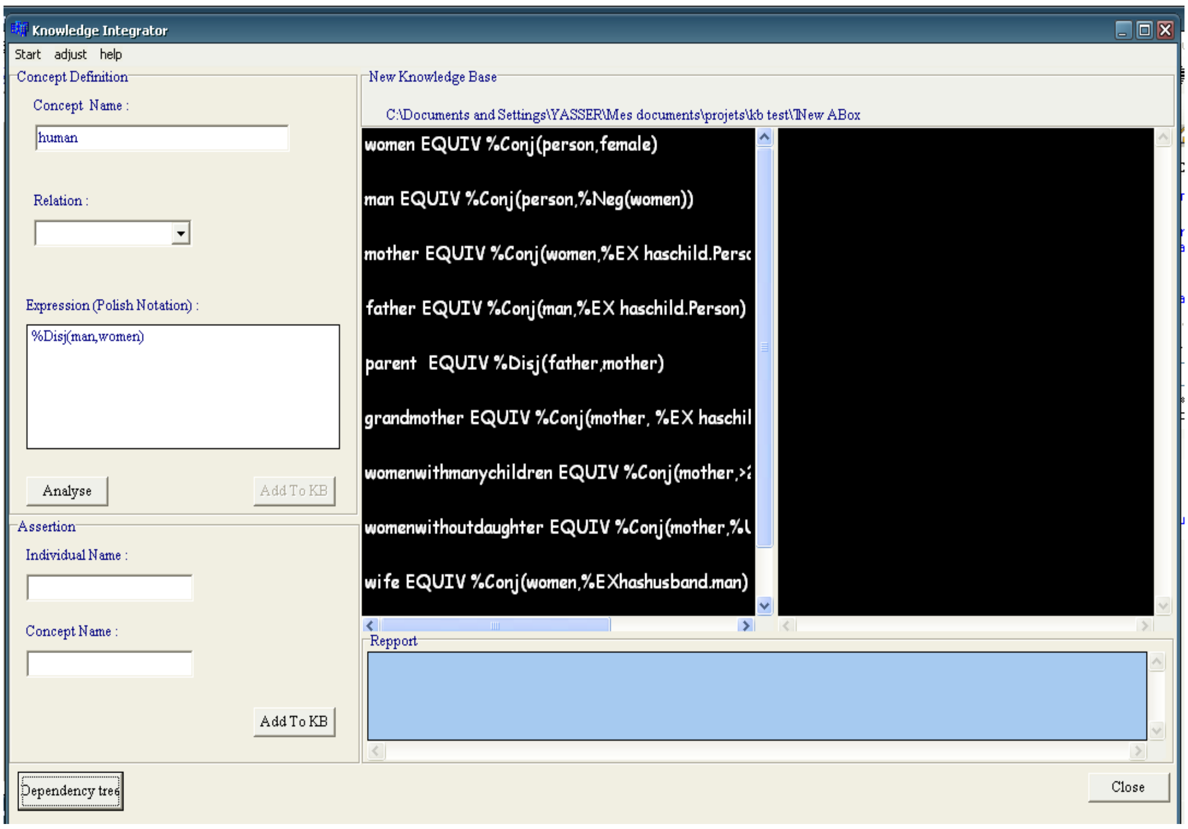
R → <role name>

Figure 11 simplified concrete syntax

4. AUTOMATED PROCESS

To accomplish this task, a simplified syntax has been used. The implemented interface analyse the introduced text respecting the polish notation generated by the proposed reduced syntax and generate *SHA* vectors after inspecting the direct subsumption relation between concepts.[76]

Using this rule the application is made to provide an interface for integration and generation of *SHA* like in figures.



C:\Documents and Settings\YASSER\Mes documents\projets\kb test\New ABox

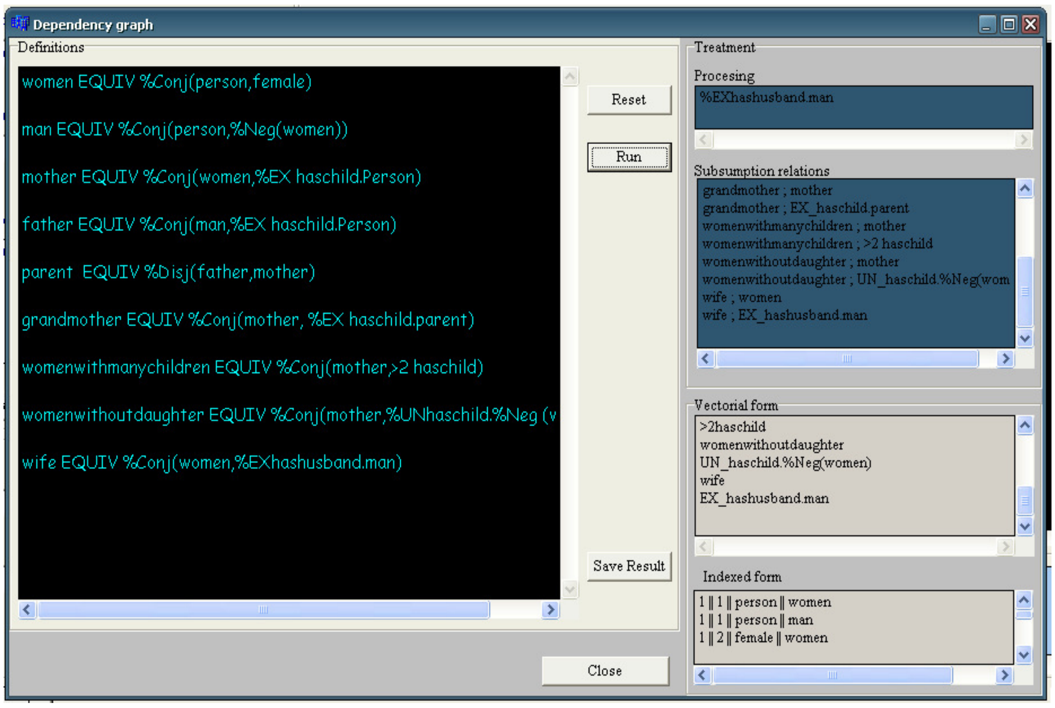
```
women EQUIV %Conj(person,female)
man EQUIV %Conj(person,%Neg(women))
mother EQUIV %Conj(women,%EX haschild.Perso
father EQUIV %Conj(man,%EX haschild.Person)
parent  EQUIV %Disj(father,mother)
grandmother EQUIV %Conj(mother, %EX haschil
womenwithmanychildren EQUIV %Conj(mother,>2
womenwithoutdaughter EQUIV %Conj(mother,%\
wife EQUIV %Conj(women,%EXhashusband.man)
```

Figure 12 The main frame of the SHA generator soft ware.

After being analyzed and add line by line the resulted text representing a part of the knowledge base is treated to inspect the direct subsumption relation between concepts. This task is necessary to generate SHAs. (See figure 13)

While this manipulation concerned the codification of knowledge after being expressed with respect to language, The possible choices were to create in interface between an existing language compiler and SHA generator, or create an analyser for simplified syntax to manipulate easily the expressed knowledge.

The pertinent choice was the second for two reasons: first to be well informed about the respected syntax. The second is that the goal of the study is not the creation of language but to check the utility and effectiveness of the SHA in KRS (Knowledge Representation System).



```
women EQUIV %Conj(person,female)

man EQUIV %Conj(person,%Neg(women))

mother EQUIV %Conj(women,%EX haschild.Person)

father EQUIV %Conj(man,%EX haschild.Person)

parent  EQUIV %Disj(father,mother)

grandmother EQUIV %Conj(mother, %EX haschild.parent)

womenwithmanychildren EQUIV %Conj(mother,>2 haschild)

womenwithoutdaughter EQUIV %Conj(mother,%UNhaschild.%Neg (v

wife EQUIV %Conj(women,%EXhashusband.man)
```

```
%EXhashusband.man
```

```
grandmother ; mother
grandmother ; EX_haschild.parent
womenwithmanychildren ; mother
womenwithmanychildren ; >2 haschild
womenwithoutdaughter ; mother
womenwithoutdaughter ; UN_haschild.%Neg(wom
wife ; women
wife ; EX_hashusband.man
```

```
>2haschild
womenwithoutdaughter
UN_haschild.%Neg(women)
wife
EX_hashusband.man
```

```
1 || 1 || person || women
1 || 1 || person || man
1 || 2 || female || women
```

Figure 13 indexed hierarchy generator

This second form shows steps of the process clearly. It begins by the TBox used which is shown in the left side. When calling “Run” the first thing is to make torques of the form (subsumer, subsumed), after that the system check concept never been in the right side like concept of the same level and eliminate them from the list. This process will be repeated until elimination of all concepts (list empty). For every level the program give sequential value for each concept detected, finally the $\mathcal{SHA}$ is constructed from the result presented in right bottom corner.

To illustrate let’s take as example the known human being knowledge base, for which the TBox is presented in ALC like following:

$$\begin{array}{rcl}
\mathsf{Woman} & \equiv & \mathsf{Person} \sqcap \mathsf{Female} \\
\mathsf{Man} & \equiv & \mathsf{Person} \sqcap \neg\mathsf{Woman} \\
\mathsf{Mother} & \equiv & \mathsf{Woman} \sqcap \exists\mathsf{hasChild.Person} \\
\mathsf{Father} & \equiv & \mathsf{Man} \sqcap \exists\mathsf{hasChild.Person} \\
\mathsf{Parent} & \equiv & \mathsf{Father} \sqcup \mathsf{Mother} \\
\mathsf{Grandmother} & \equiv & \mathsf{Mother} \sqcap \exists\mathsf{hasChild.Parent} \\
\mathsf{MotherWithManyChildren} & \equiv & \mathsf{Mother} \sqcap \geqslant 3\ \mathsf{hasChild} \\
\mathsf{MotherWithoutDaughter} & \equiv & \mathsf{Mother} \sqcap \forall\mathsf{hasChild}.\neg\mathsf{Woman} \\
\mathsf{Wife} & \equiv & \mathsf{Woman} \sqcap \exists\mathsf{hasHusband.Man}
\end{array}$$

Figure 14 human being TBox

The application of the process result of the graph of the figure 15. the codification in SHA is realised after level detection and association of sequential code and concepts of the same level.

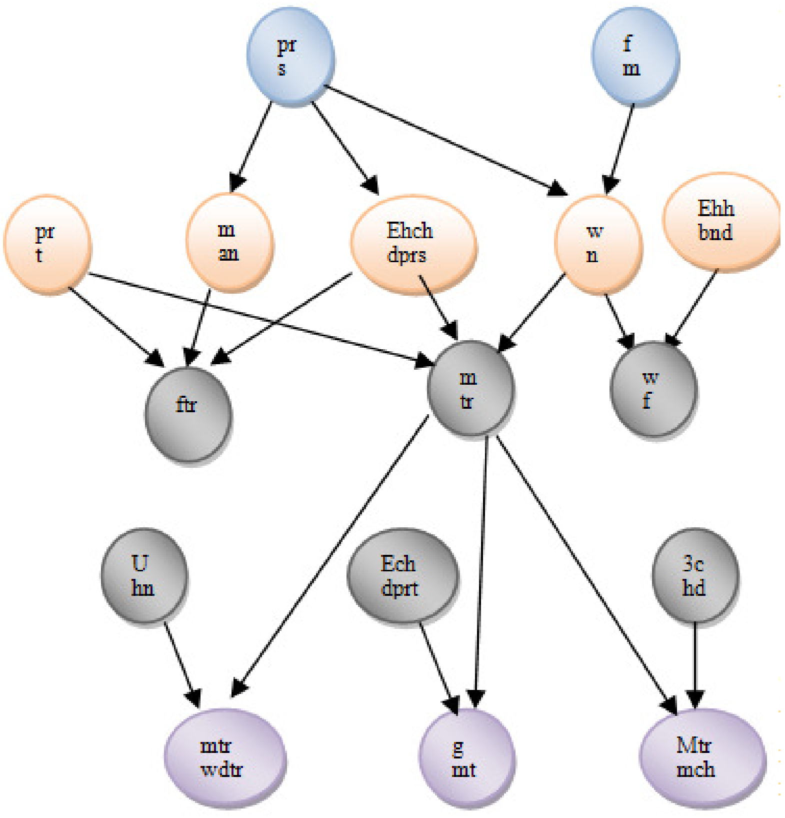


Figure 15 automated process for the subsumption dependency tree deduction

As it is previously indicated to create the vectorial representation we must deal with a valuation of the dependency graph, this is realized by attributing numbers sequentially to nodes of the same level (see the figure 15). The vector can be defined using numbers of nodes that construct the path from Root to the node corresponding to the defined concept.

For example the concept Father can be defined by two vectors (1, 1, 1, 0) and (1, 2, 1, 0), we can deduce that Father is the intersection of (1, 1, 0, 0) and (1, 2, 0, 0) which are Man and Parent we write Father≡ Parent ∩ Man. All vectors for the human being TBox are presented in the table 5

Table 6 vectorial representation

Concept	Vectors
Person	(1, 0, 0, 0)
Female	(2,0,0,0)
Woman	(1,4,0,0); (2,4,0,0)
Man	(1,2,0,0)
Parent	(-1,1,0,0)
Exist has child.Person	(1,3,0,0)
Exist has husband	(-1,5,0,0)
Univ. has child. not woman	(-1,-1,1,0)
Exist has child parent	(-1,-1,3,0)
>=3child	(-1,-1,5,0)
Wife	(1,4,6,0)
Father	(1,2,2,0); (1,3,2,0)
Mother	(1,3,4,0); (2,4,4,0); (1,4,4,0)
MotherWithManychildren	(1,3,4,3); (1,4,4,3); (2,4,4,3); (-1,-1,5,3)
MotherWithoutDaughter	(1,3,4,1); (1,4,4,1);(2,4,4,1)
GrandMother	(1,3,4,2); (2,4,4,2); (-1,-1,3,2)

A simple attempt to use these SHAs can show that the meanings of conjunction and disjunction are clearly represented. The SHAs attribution is an application which associates concept with at least one vector. Mathematically saying, it is presented like follows:

$$f: D \longrightarrow V^k$$

$$C\ (concept) \longrightarrow \{sha_1, \ldots\ldots, sha_n\}$$

Where “k” represents the maximum of dependency graph profound, “n” number of SHAs for one concept (note that n≠0), D represents the domain a set of vectors of “k” components. The function of attribution must be injective. This is evident because of the sequential codification under every level.

What we can deduce from the SHAs in the table05?

Conjunctions

Let’s imagine a concept C represented by a node in level “t”. If in SHAs representing C there before the “t^th” component different left-sides then C represent the intersection between concepts represented by these left-sides. By this why we obtain the following axioms.

$$\text{Father} \equiv \text{man} \cap \exists\ \text{haschild.Person}$$

$$\text{Woman} \equiv \text{Person} \cap \text{Female}$$

$$\text{Mother} \equiv \text{women} \cap \text{Parent} \cap \exists\ \text{haschild.Person}$$

$$Motherwithmanycildren \equiv Mother \cap > 2\ h\text{aschild}$$

$$MotherWithoutDaughter \equiv Mother \cap \forall\ h\text{aschild.} \neg\ \text{Woman}$$

$$\text{Grand}Mother \equiv Mother \cap \text{haschild.Parent}$$

Disjunctions

Concept that never exists in definitions of SHAs of the same concepts are disjoint concept as *man* and *woman* because there is no concept in which there are two SHAs having (1, 2... and [(1, 4, ... or (2, 4, ...] to gather. The same thing is available with *female* and *man.*

For the union of concept comparing SHAs allowed the detection of sub-concepts of a super-concept by the checking for same left side that figure in SHAs of different concepts, in this case we say that the union of these lasts is equivalent to the concept represented by this SHA containing the common left side and “0” in all right component. From this point of view we can find:

$$\text{Person} \equiv \text{Woman} \cup \text{Man}$$

$$\text{Parent} \equiv \text{Father} \cup \text{Mother}$$

$$\text{Mother} \equiv Motherwithmanycildren \cup MotherWithoutDaughter \cup \text{Grand}Mother$$

For the subsumption the detection is simpler, because the structure is constructed from the sub direct subsumption relationship. That means that the transitive closer is allowed by comparing vectors.

PART II: PROPSED APPROACH

CHAPTER 02: REASONING WITH $\mathcal{SHA}$

1. ADAPTED TABLEAU $\mathcal{ALC}$ RULES

In this section we present how to adapt the reasoning rules on the new representation structures. Each component in SHA has a signification thing that makes the deduction process based on vectors comparing.

In follows the Tableau $\mathcal{ALC}$ rules can be transformed to deal with the vectorial representation with graphical illustrates.

1.1. The ∩-rule

If $\mathcal{A}$ contains $(C_1 \cap C_2)(x)$, but not both $C_1(x)$ and $C_2(x)$ **then**

$$A' \equiv A\ U\ \{C_1(x), C_2(x)\}$$

Let's suppose $\mathcal{A}$, C_1 and C_2 concepts represented by V$(a1,a2,...ap)$, V$_1(c_1,c_2,.....c_p)$, and V$_2(b_1,b_2,....b_p)$

P is the constructed by the conjunction between C1 & C2 which yields to two vectors $(c_1,.......,c_k,0,......,0)$ and $(b_1,..........,b_k,0,.......,0)$ with $c_k= b_k$.

Condition

If we find another concept $\mathcal{A}$ with vector V of the form $(a_1,.........., a_k,0,.......,0)$ where $a_k= c_k= b_k$

Action

- Create a node $\mathcal{A}'$ with representative vector of the form $(a_1,-1,.......,-1, t,0,.......,0)$ where t-1 is number of concepts in level k-1.
- Add new vectors to define $\mathcal{A}$, C_1 and C_2 like follows

$$\textbf{A}\Rightarrow(a_1,-1,.......,-1, t, a_k,.......,0)$$

$$\textbf{C}_1\Rightarrow(a_1,-1,.......,-1, t, c_k,.......,0)$$

$$\textbf{C}_2\Rightarrow(a_1,-1,.......,-1, t, b_k,.......,0)$$

And by the way each concept subsumed by A, C_1 or C_2 must have new vector defined by taking the left side of this three vectors.

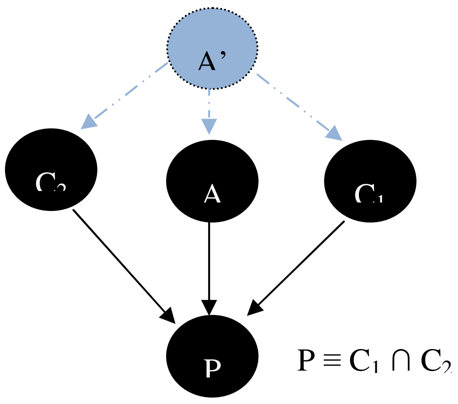


Figure 16 Representation of the rule of conjunction applied to the vectorial form

1.2. The ∪-rule

If $\mathcal{A}$ contains $(C_1 \cap C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$ **then**

$$A' \equiv A \cup \{C_1(x)\} \text{ and } A'' \equiv A \cup \{C_2(x)\}$$

Let's suppose $\mathcal{A}$, C_1 and C_2 concepts represented by $V(a1,a2,...ap)$, $V_1(c_1,c_2,.....c_p)$ and $V_2(b_1,b_2,....b_p)$

$\mathcal{P}$ is the constructed by the disjunction between C_1 & C_2 which yields to two vectors $(a_1,.......,a_{k-},a_k,c_{k+1},.......,0)$ and $(a_1,.........,a_{k-1},a_k,b_{k+1},.......,0)$ representing C1 and C2 respectively

Condition

If $\mathcal{A}$ is represented by a vector in the form $(a_1,.......,a_{k-1},0,...,0)$ & $\mathcal{P}$ is represented by a vector of the form $(a_1,........,a_{k-1},a_k,0,.....,0)$ and we find C_1 and C_2 having respectively vectors of the form

$(a_1,.......,a_{k-1},a_k,c_{k+1},......,0)$, $(a_1,.........,a_{k-1},a_k,b_{k+1},.......,0)$.

Action

- Create two new nodes A' and A" with vectors $(a_1,-1,.......,-1, t,0,.......,0)$, $(a_1,-1,.......,-1, t+1,0,.......,0)$ while t-1 is number of concept existing in level k-2.

- Add new vectors to define $\mathcal{A}$, C_1 and C_2 like follows

A ➔ $(a_1,-1,\ldots\ldots,-1, t, a_{k-1},\ldots\ldots,0)$, $(a_1,-1,\ldots\ldots,-1, t+1, a_{k-1},\ldots\ldots,0)$

P ➔ $(a_1,-1,\ldots\ldots,-1, t, a_{k-1},a_k,0,\ldots\ldots,0)$, $(a_1,-1,\ldots\ldots,-1, t+1, a_{k-1},a_k,0,\ldots\ldots,0)$

C_1 ➔ $(a_1,-1,\ldots\ldots,-1, t, -1, -1, c_{k+1},\ldots\ldots,0)$, $(a_1,-1,\ldots\ldots,-1, t+1, -1,-1, c_{k+1},\ldots\ldots,0)$, $(a_1,-1,\ldots\ldots,-1, t, a_{k-1},a_k, c_{k+1}, 0,\ldots\ldots,0)$, $(a_1,-1,\ldots\ldots,-1, t, a_{k-1},a_k, c_{k+1}, 0,\ldots\ldots,0)$

C_2 ➔ $(a_1,-1,\ldots\ldots,-1, t+1,-1, -1, b_{k+1},\ldots\ldots,0)$, $(a_1,-1,\ldots\ldots,-1, t+1, -1, -1, b_{k+1},\ldots\ldots,0)$, $(a_1,-1,\ldots\ldots,-1, t+1, a_{k-1},a_k, c_{k+1}, b_{k+1}, 0,\ldots\ldots,0)$, $(a_1,-1,\ldots\ldots,-1, t, a_{k-1},a_k, b_{k+1}, 0,\ldots\ldots,0)$

By this same method we define new vectors for every subsumed concepts derived from this concepts

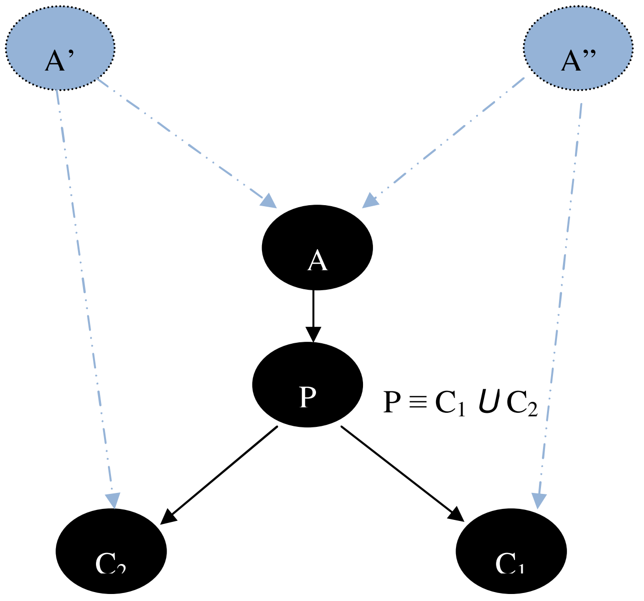


Figure 17 Representation of the rule of disjunction applied to the vectorial form

1.3. The ∃-rule

If $\mathcal{A}$ contains $(\exists\mathcal{R}.C)(x)$ but there is no individual name z such that $C(z)$ and $\mathcal{R}(x,z)$ are in $\mathcal{A}$ **then**

$$A' \equiv A \cup \{C(y), R(x,y)\}$$

Where y is an individual name not occurring in $\mathcal{A}$

In this rule the condition specified the individuals z such as $\mathcal{R}(x,z)$ occurring in C but not in $\mathcal{A}$ which contains the concept defined by $(\exists \mathcal{R}.C)(x)$. This leads to a vectorial definition like in follows.

Note that when the condition is verified the actions need to create some new nodes to represent the sub concepts of C and $\mathcal{A}$.

Lets define $A_1 = \{a, A(a) \wedge R(x,a)$ with $C(x)\}$ and $A_2 = \{y, C(y) \wedge R(x,y)$ with $c(x) \wedge \neg A(y)\}$ then we imagine when A and C are represented by $(a_1, \ldots\ldots, a_k, 0, \ldots\ldots,)$ and $(a_1, a_2' \ldots\ldots a_k', 0, \ldots\ldots, 0)$

Condition

If there is no individual z occurring in C and not in $\mathcal{A}$ having relation with x such as $\mathcal{R}(x,z)$, which means an individual occurring in $\mathcal{A}_2$.

Action

- Create $\mathcal{A}'(a_0, -1, \ldots\ldots, -1, t, 0, \ldots\ldots 0)$ where (t-1) is the number of concept in level (k-1) and $\mathcal{A}' \equiv \mathcal{A} \cup \mathcal{A}_2$, add new vectors in definitions of $\mathcal{A}$, $\mathcal{A}_2$ and there sub concepts are like in precedent.

A ➔ $(a_1, -1, \ldots\ldots, -1, t, a_k, 0, \ldots\ldots, 0)$

A_2 ➔ $(a_1, -1, \ldots\ldots, -1, t, a_k, -1, -1, a_k, 0, \ldots\ldots, 0)$

With this same method we define new vectors for every concept subsumed by all concepts having new representatives' vectors.

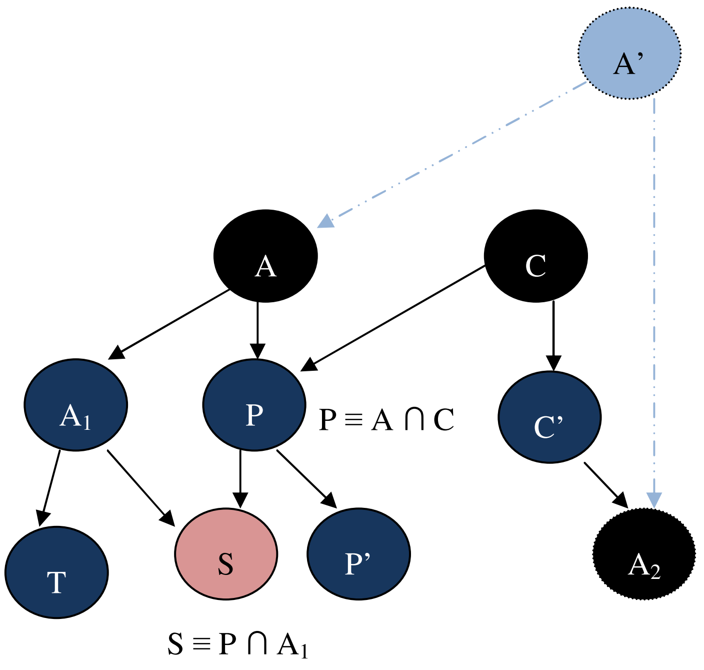


Figure 18 Representation of the rule of existential quantification applied to the vectorial form

1.4. The ∀ rule

If $\mathcal{A}$ contains $(\forall \mathcal{R}.C)(x)$ and $\mathcal{R}(x,y)$, but it does not contains $C(y)$ **then**

$$A' \equiv A\ U\ \{C(y)\}$$

We suppose that $A(a_1, \ldots, a_k, 0, \ldots 0)$ and $C(a_0, a_1', \ldots, a_k', 0 \ldots 0)$

The concept $A_1 = \{x, A(x) \wedge R(x,y) \text{ with } C(y)\}$

Condition

If $\mathcal{A}$ contains $\mathcal{A}_1$ and not $C(y)$

Action

- Create $\mathcal{A}'(a_1, -1, \ldots, t, 0, \ldots, 0)$ while (t-1) is number of concept in level (k-1)
- Add vectors for $\mathcal{A}$ and C and all of their subsumed concepts

A ➔ $(a_1, -1, \ldots, -1, t, a_k, 0, \ldots, 0)$

C ➔ $(a_1, -1,\ldots\ldots, -1, t, a'_k, 0,\ldots\ldots,0)$

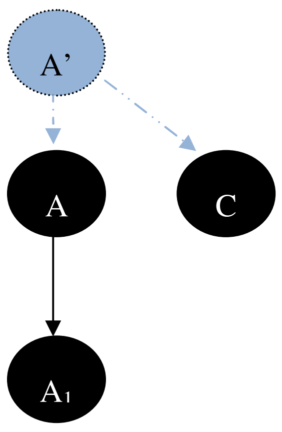


Figure 19 Representation of the rule of universal quantification applied to the vectorial form.

The application of this basic Algorithm can result some augmentation of nodes number from iteration to another until stability which represent case where there is no new additional nodes (see figure 20).

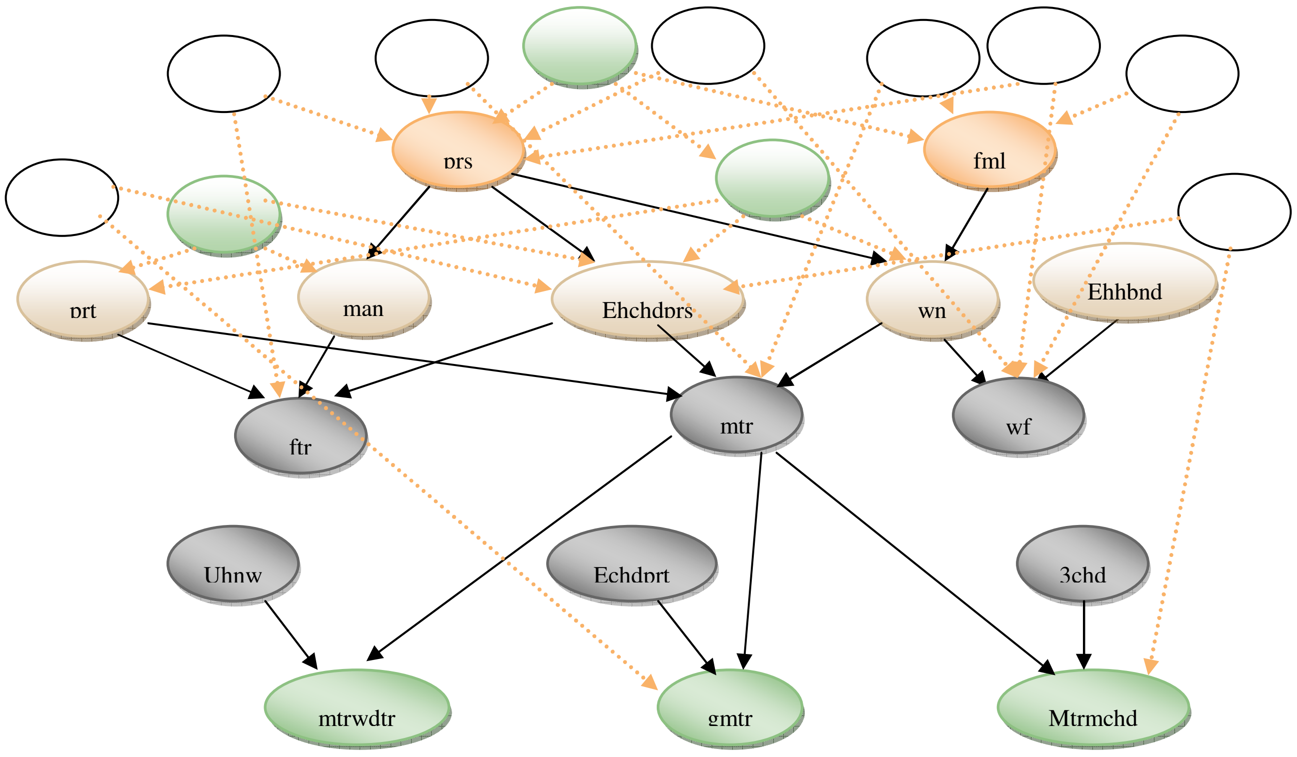


Figure 20 Human Being KB augmented after one execution of reasoning rules

Note at this stage that the roles not being applied here concern the assertional level and based on numbers restriction equalled, unequalled, qualified and unqualified. We consider that there is no influence on the proposed representation approach. The numerical restrictions means numbers of individuals which means that it can be computed by the ordinary methods or at least using the SHAs of the more specific defined concept for which individuals are instances.

This rules like presented in here:

1.5. The →≥rule

Condition A contains $(\geq n\ R)(x)$, and there are no individual names $z_1, \ldots, z_n$ such that $R(x, z_i)$ $(1 \leq i \leq n)$ and $z_i \neq z_j$ $(1 \leq i < j \leq n)$ are contained in A.

Action $A' = A \cup \{R(x, y_i) \mid 1 \leq i \leq n\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n\}$, where $y_1, \ldots, y_n$ are distinct individual names not occurring in A.

1.6. The →≤ rule

Condition: A contains distinct individual names $y_1, \ldots, y_{n+1}$ such that $(\leq n\ R)(x)$ and R(x, y1), . . . , $R(x, y_{n+1})$ are in A, and $y_i \neq y_j$ is not in A for some $i \neq j$.

Action: For each pair y_i, y_j such that $i > j$ and $y_i \neq y_j$ is not in A, the ABox $A_{i,j} = [y_i/y_j]A$ is obtained from A by replacing each occurrence of y_i by y_j.

That is confirmed by “The algorithm uses value restrictions in interaction with already defined role relationships to impose new constraints on individuals.”[13]

2. CONVERGENCE OF THE ALGORITHM

Every one of the four rules is conditioned with binary condition (true /false). The rule Figure in form of “if condition then action” every action makes new SHA to represent new nodes. This means that the probability of creating new SHAs is a Bernoulli distribution. And by iteration it becomes Binomial distribution. That means that we can calculate the number of additional SHAs by the function flike in follows.

$$f(x_1, x_2, x_3, x_4) = N_1.T_1(x_1) + N_2.T_2(x_2) + N_3.T_3(x_3) + N_4.T_4(x_4)$$

&

$$p = 1 - q$$

Where

$T_i(x) = \mathrm{C}_{\mathrm{x}}^{\mathrm{n}} . \mathrm{P}^{\mathrm{x}} . (1 - \mathrm{P})^{\mathrm{n-x}}$ With 0<x<n,

x_iSuccess number; it represents the case when conditions of rule is satisfied,

n for number of iteration (experiments).

N_i Represents number of new SHAs created by the rule i when condition is satisfied. $\mathrm{N_i}$ is relative to the number of nodes created.

$$C_x^n = \frac{n!}{x!\,(n-x)!}$$

Then

$$T_i(x) = \frac{n!}{x!\,(n-x)!} . \mathrm{P}^{\mathrm{x}} . (1 - \mathrm{P})^{\mathrm{n-x}}$$

To prove that the Algorithm will attend a stop point means that we must find the limit of the function f.when n is bigger. This limit must be equal to 0 ie. Not more additional SHAs.

Now we accept that if all $T_i(x)$ converge f converge also. And the limit of the augmentation function must be equal to “0” to riche stability case in which the inference stops.

$$\lim_{\substack{x\to+\infty \\ n\geq x}} T_i(x) = \lim_{\substack{x\to+\infty \\ n\geq x}} \left(\frac{n!}{x!\,(n-x)!}\,.\,\mathrm{P}^{\mathrm{x}}\,.\,(1-\mathrm{P})^{\mathrm{n-x}}\right)$$

To prove the convergence, we know that $0 \leq x \leq n$ that means that if $x \to +\infty$, $n \to +\infty$ because if we like to have a big x we must have a bigger n.

Then

$$T_i(x) = \frac{n!}{x!\,(n-x)!}\,.\,\mathrm{P}^{\mathrm{x}}\,.\,(1-\mathrm{P})^{\mathrm{n-x}}$$

$$= \frac{n!}{(n-x)!\,n^{\mathrm{x}}}\,.\,\frac{(n\mathrm{P})^{\mathrm{x}}}{x!}\,.\,\frac{(1-\mathrm{P})^{n}}{(1-\mathrm{P})^{\mathrm{x}}}$$

We accept these three additional constraints

$$\lim_{n\to\infty}\frac{n!}{(n-x)!\,n^{\mathrm{x}}} = 1$$

$$\lim_{P\to 0}(1-\mathrm{P})^{\mathrm{x}} = 1$$

$log(1-\mathrm{P})^{\mathrm{n}} = n\,log(1-P)$ when $n \to \infty$ and $P \to 0.$

We deduce that

$$\lim_{\substack{n\to\infty, \\ P\to 0, \\ nP\to\lambda}} (1-\mathrm{P})^{\mathrm{n}} = e^{-\lambda}$$

As conclusion

When $n \to +\infty, p \to 0$ and $nP \to \lambda$ where

$\frac{n!}{x!(n-x)!} \cdot \mathrm{P}^{\mathrm{x}} \cdot \left(1 - \mathrm{P}\right)^{\mathrm{n-x}}$ Can be approximate to $e^{-\lambda} \frac{(\lambda)^{\mathrm{x}}}{x!}$

Now, $x \to +\infty$, $(0 \leq x \leq n\,)$

$$\lim_{\substack{x\to+\infty \\ n\geq x}} \boldsymbol{T_i(x)} = \lim_{\substack{x\to+\infty \\ n\geq x}} \boldsymbol{e^{-\lambda}} \frac{(\boldsymbol{\lambda})^{\mathrm{x}}}{\boldsymbol{x}!} = \mathbf{0}$$

That means

$$\lim_{\substack{X\to+\infty \\ n\geq x_i}} \boldsymbol{f(x_1, x_2, x_3, x_4)} = \lim_{\substack{x\to+\infty \\ n\geq x}} \boldsymbol{e^{-\lambda}} \frac{(\boldsymbol{\lambda})^{\mathrm{x_1}}}{\boldsymbol{x_1}!} + \lim_{\substack{x\to+\infty \\ n\geq x}} \boldsymbol{e^{-\lambda}} \frac{(\boldsymbol{\lambda})^{\mathrm{x_2}}}{\boldsymbol{x_2}!} + \lim_{\substack{x\to+\infty \\ n\geq x}} \boldsymbol{e^{-\lambda}} \frac{(\boldsymbol{\lambda})^{\mathrm{x_3}}}{\boldsymbol{x_3}!} + \lim_{\substack{x\to+\infty \\ n\geq x}} \boldsymbol{e^{-\lambda}} \frac{(\boldsymbol{\lambda})^{\mathrm{x_4}}}{\boldsymbol{x_4}!}$$

$$\lim_{\substack{X\to+\infty \\ n\geq x_i}} \boldsymbol{f(x_1, x_2, x_3, x_4)} = \mathbf{0}$$

We know that if every $T_i(x) \to 0$ means that the Algorithm achieves stability. Other way the limit “0” means that there is no inference, no augmentation or no new deductions which means the case of stability.

PART II: PROPSED APPROACH

CHAPTER 03: STUDY CASE ARABIC LANGUAGE RESSOURCE CREATION

1. STADY CASE DESCRIPTION

this study case, the KB taken as example is developed like a Meta knowledge base describing a higher level of knowledge for lexical Ontologies. The goal of creating this KB was the integration of “syntactic role” which means the function of the word in a sentence especially for the Arabic language.

The system is designed to allow the reuse of existing resources and consolidate them with the Meta KB as an add-on. The first idea was represented using AL but for first implementation it was translated to vectorial attribute associated to each node.

2. THE ALP SYSTEM ARCHITECTURE [77]

In the traditional DLs systems “A knowledge base (KB) comprises two components, the TBox and the ABox. The TBox introduces the terminology, i.e., the vocabulary of an application domain, while the ABox contains assertions about named individuals in terms of this vocabulary”

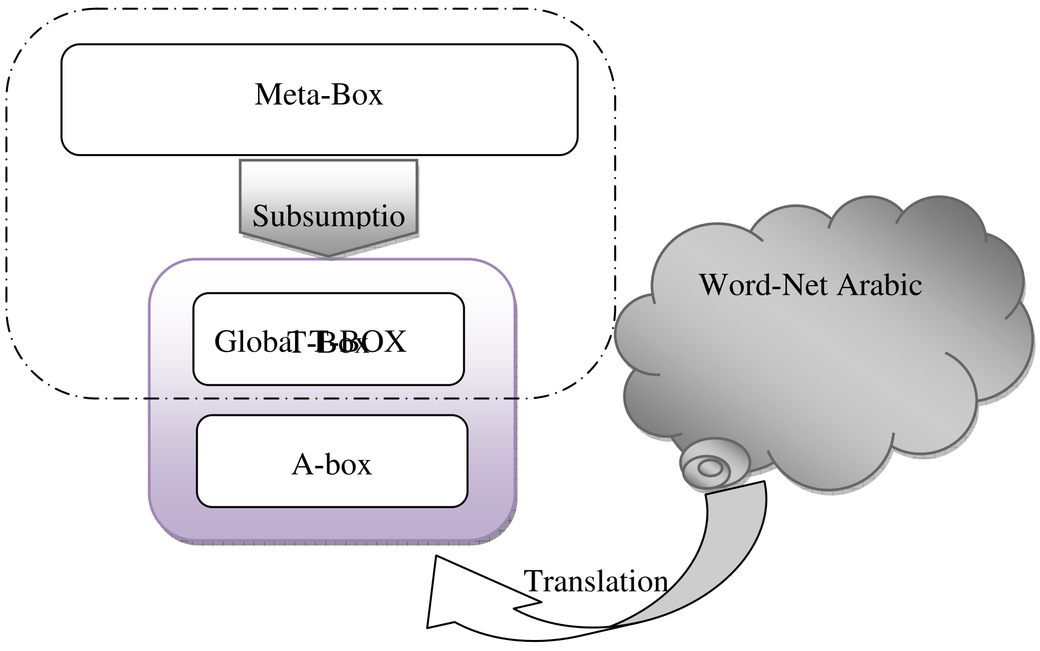


Figure 21 architecture of the ALP system

However; the addition of the Meta-Box in the present system (see figure 21) can be considered as a consolidation for the traditional system. This fact makes the introduction of the syntactic role of the term (concept) with its definition possible and it is implicitly defined in the class of the word which is integrated by establishment of the link between the Meta-Box and the T-Box using the relation of the subsumption. Since all the terms existing in Word-Net Arabic ontology can be a subsumed by a concept among the Meta-Box concepts either directly or indirectly.

The T-Box / A-Box are traditionally constructed by the translation of definitions in the Ontology of concepts and their relations to DLs knowledge base defined using the ALC description language.

3. THE WORD IN THE ARABIC LANGUAGE

The word in Arabic language can be viewed as an occurrence of a node of a dependency graph representing a hierarchical organization of the classes existing for the syntactic roles as following : the word is called in Arabic "Kalima", this model includes the three nodes which are "Fiil" (verb) , "Ism" (noun) , and "harf" (propositions, conjunctions and so on). And all of these terms have dependency with others to create the hierarchy of concepts. (see Figure 22)

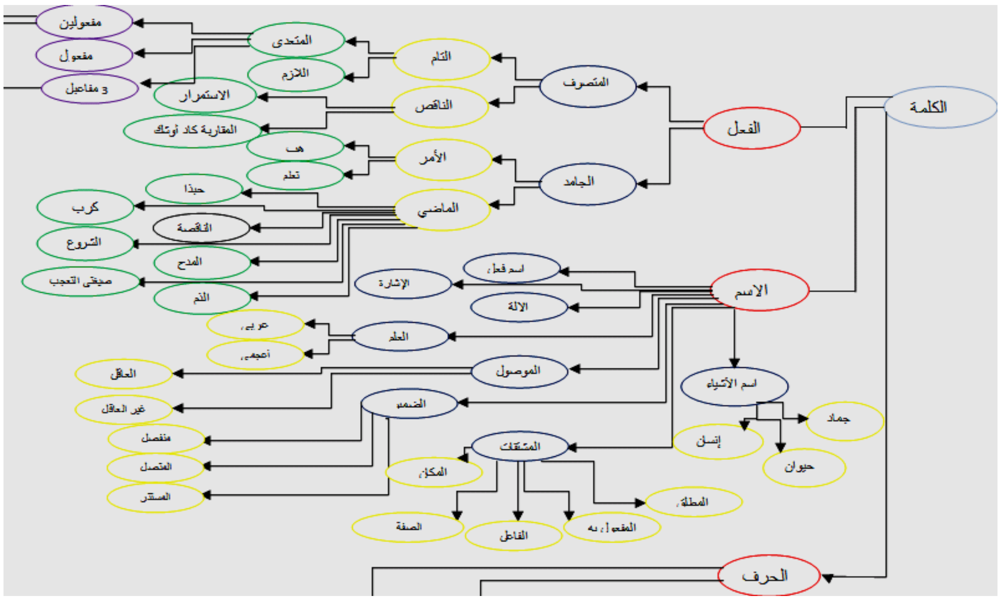


Figure 22 hierarchy of Syntactic Roles of Words in Arabic Language

This hierarchy have been inspired from a lecture in the Arabic documentations especially the poems of el ADJROUMIA and ELFIAT IBN MALEK; but for the sake of semantic implementation we have focused on the classes which influence the meaning and the use of the word in sentences. In order to realize this graph in fig 02 we use the same techniques of the dependency graph construction known as a graphic representation which is “a sample dependency graph in which word nodes are given in bold face and dependency relations are indicated by labelled edges”[77].

In this proposition the relation represented by edges is “IS A” which is interpreted as subsumption in DLs terminology or subset in the interpretation of DLs in set theory seeing the concepts as a set of individuals.

4. THE REPRESENTATION IN AL LANGUAGE

The graph in the figure 22 here can reflect the concepts of the Meta-Box and the relation between them, the thing that helps in defining the axioms and constructing the Meta-Box which is the essential part because of the meaningful representation given implicitly with the specification of the syntactic roles of each word defined in the second part. We present here the description in the KL-One language written and constructed for the definition of this first level part of the global knowledge base.

$$Kalima \equiv T$$

$$F\hat{\imath}il \subset \text{kalima}$$

$$Ism \subset \text{kalima}$$

$$harf \subset \text{kalima}$$

$$F\hat{\imath}il \equiv \text{Motassarrif} \cup \text{Djamid}$$

$$\text{Motassarrif} \equiv \text{Tam} \cup \text{Nakis}$$

$$\text{Tam} \equiv \text{Motaâddi} \cup \text{Ellazim}$$

$$\text{Motaâddi} \equiv \text{Mmafôul} \cup \text{Mmafôul2} \cup \text{Mmafôul3}$$

$$\text{Nakis} \equiv \text{Istimrar} \cup \text{Elmokaraba}$$

$$\text{Djamid} \equiv \text{Amr} \cup \text{madhi}$$

$$\text{Nakis(m)} \subset \text{Madhi}$$

$$Madhe \subset \text{Madhi}$$

$$DHem \subset \text{Madhi}$$

$$chorou\hat{e} \subset \text{Madhi}$$

$$Etta\hat{a}djoub \subset \text{Madhi}$$

$$Fi\hat{i}l \equiv \neg (Ism \cup Harf) \equiv \neg Ism \cap \neg harf$$

$$Ism \equiv \neg (Fi\hat{i}l \cup Harf) \equiv \neg Fi\hat{i}l \cap \neg harf$$

$$harf \equiv \neg (Fi\hat{i}l \cup Ism) \equiv \neg Fi\hat{i}l \cap \neg Ism$$

$$Ism \equiv IsmFi\hat{i}l \cup Ichara \cup \hat{A}lam \cup Maoussoul \cup Dhamir \cup Achyae \cup Mouchtak$$

$$\hat{A}lam \equiv \hat{A}rabi \cup A\hat{e}djami$$

$$Maoussoul \equiv \hat{A}akil \cup Ghir\hat{a}akil$$

$$Dhamir \equiv Mouttassil \cup Mounifassil \cup Moustatir$$

$$Mouchtak \equiv IsmMakan \cup Sifa \cup IsmFa\hat{i}il \cup IsmMaf\hat{o}ul \cup Mf\hat{o}ulMoutlak$$

$$Achyae \equiv Djamad \cup Hayaouane \cup Insane$$

$$harf \equiv ouhadi \cup thounaei \cup thoulathi \cup roba\hat{e}i \cup khomassi$$

This part contains concepts definitions. It uses the conjunction and the disjunction to make the axioms. The Meta knowledge base allowed the integration of syntax by the use of roles.

The noun can be *Faîil* (subject) or *MafôulBih* (complement) in verbal sentences, and it can be *Mobtadae* (the first noun in nominale sentence) or *khabar* (2nd name in NS) in nominal sentences. It can be also *Maoussouf* (qualified) for a *Sifa* (adjective). There are other links between words in Arabic syntax that can be integrated like roles. By this way we can add some axioms to the Meta-Box. (See following examples)

$$\text{Faîil} \equiv \text{Ism} \cap \exists\, \text{ISFaîil}.\text{Fiîl}$$

$$\text{Mafôul} \equiv \text{Ism} \cap \exists\, \text{ISMafôu}.\text{Motaâddi}$$

$$\text{Moubtadae} \equiv \text{Ism} \cap \exists\, \text{ISMobtada}.\text{Ism}$$

$$\text{Sifa} \equiv \text{Ism} \cap \exists\, \text{ISSifa}.\text{Ism}$$

5. THE REPRESENTATION USING SHAS

The SHA of these concepts can be deduced from the hierarchy (see figure 22) directly because it is represents the dependency graph of the subsumption relationship. But the restrictions will not be considered. It's why the work must be completed before dealing with SHAs creation.

Note at this step that we can define a specification for the role *ISMafôul;* coming from the meaning of verbs that needs one, two and three complements. Here it is clear that if we define three roles, the three roles be subsumed by the role *ISMafôul*.

This result a number of SHAs presented in Table 06 like follows.

Table 7 the Meta-base with SHAs

(1)Mafôul مفعول	**(1)Moutaâddi** المتعدي	**(1) Tam** التام	**Motassarif** المتصرف **(1)**	فعل **Verb Fiîl (1)**	كلمة **Word Kalima (1)**
(2) 2Mafôul مفعولين					
(3) 3Mafôul ثلاث مفاعيل					
	(2)Allazim اللازم				
	(3)Istimrar الاستمرار	(2) Nakis الناقص			
كاد	(4)Mokaraba المقاربة				
أوشك					
	هب	(3) Amr الأمر	الجامد Djamid (2)		
	تعلم				
	حبذا	(4) Madhi الماضي			
	كرب				
	الناقصة				
	الشروع				
	المدح				
	الذم				
	التعجب				
			(3)Âlam علم	اسم Noun Ism (2)	
			(4) Ala آلة		
		(5) Moutlak م المطلق	Mouchtak مشتق (5)		
		(6)IsmMafôulالمفعول به			
		(7) Ism Faîil الفاعل			
		(8) Sifa الصفة			
		(9) Elmakan المكان			
			(6)Ichara اشارة		
		(10) Âakil عاقل	(7) Maoussol موصول		
		(11)GhirÂakil غير عاقل			
		(12)Mounfassil منفصل	(8) Dhamir ضمير		
		(13)Mouttassilمتصل			
		(14)Mousstatir مستتر			
			(9) IsmFiîl اسم فعل		
		(15)Djamad جماد	(10) Achyae الشيئ		
		(16)Hayaouane حيوان			
		(17)Insane انسان			
			أحادي(11)	حرف Particles harf (3)	
			ثنائي(12)		
			ثلاثي(13)		
			رباعي(14)		
			خماسي(15)		

The restriction defined using roles can be the source of new nodes in the dependency graph. The new nodes are indexed sequentially by the SHAs generation process. Like a new modification of the TBox and they will be added by redefinition of the KB using the automated generation application seen previously.

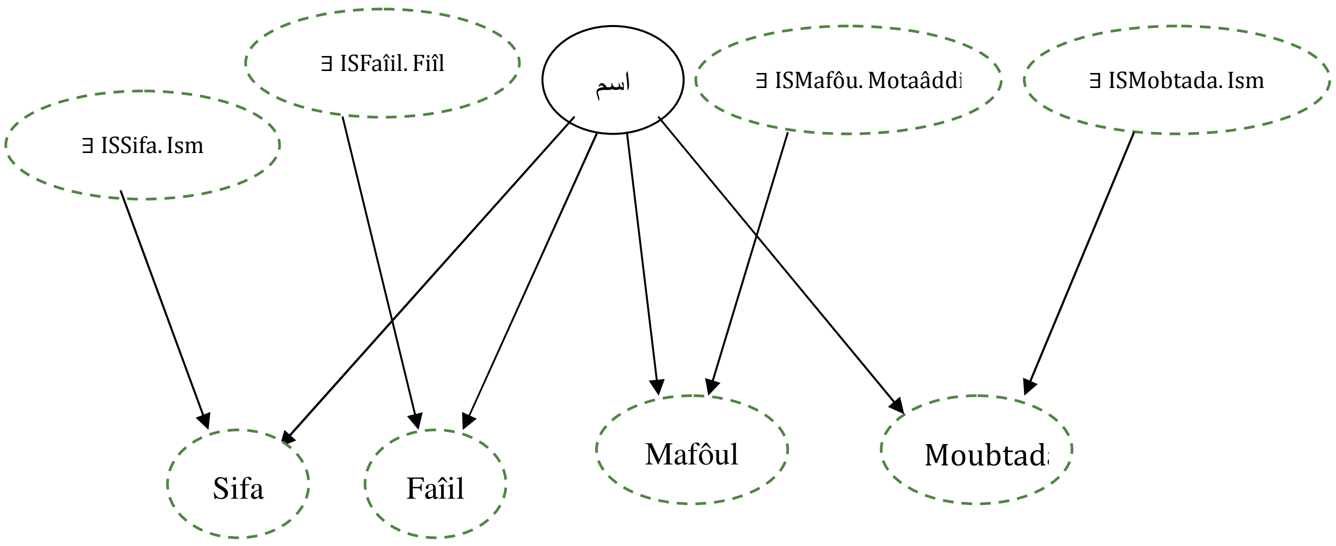


Figure 23 example of the additional restriction to hierarchy of words in Arabic

All Arabic words are seen like assertions for this Meta-Box and each word is an assertion of one of the more specific concepts defined. To make the links between these two levels, it is simple to associate every word with the SHA of the more specific concept where it is classified.

As seen in example the SHA generation process is perfectly reversible. This makes that we can return to the defined axioms from the existing SHAs using a correspondence checker which compare the vectors and deduce relation before regenerating axioms with conjunction and disjunction composition.

DISCUSSIONS

DISCUSSIONS ABOUT SHA

The advantages of this approach is that; in one hand, by structuring knowledge in significant and easily computational way. This provides some facilities such as:

It enables an easier implementation, the choice to make between making an interface between existing DLs language analysers and vectorial representation automated generators or creating a new simple languages that result after analysis to the KB with SHAs like concepts definitions.

It makes the reasoning Algorithm a simple process manipulating significant vectors. The difference is clear between manipulating axioms and comparing vectors component. The same deductions are allowed with leas complex Algorithms.

The generation of the SHAs is an easily automated process. We have experiment this implementation and it’s clear that the process is based on the direct subsumption relation detection which results of torques of the form (subsumer, subsumed). These last unable the generation of the dependency graph and by consequence the SHAs.

In the other hand, comparing the performance of ordinary systems the changes have no negative effects on abilities and requirements needed from KRSs such as:

Intuitively we can say that effectiveness will not decrease. The part touched in this approach is the codification of DLs existing representation. That means if the structured code is enough meaningful to carry all represented aspects. The result required from the DLs systems using SHAs will be the same comparing to ordinary systems.

Efficiency will increase because of the use of vectors comparing which makes the algorithm leas complex. We imagine at this stage that the Algorithm which deals with vectors by comparing their components is simplest than an algorithm dealing with defined axiom using the AL syntax.

The possibility of making feed-back to the AL original representation confirms that the vectorial approach represents a considered power ability to express the semantic in comparable level with logical-based semantic approaches because basically this vectorial representation is a translation of DLs representations.

The critical point is that at this step we can't affirm that the SHAs are full expressive because it is known that reducing equivalences to subsumption relationships decreases the level expressivity. But also we can't affirm in the other hand that the representation with SHAs is leas expressive thing which is let for future study. But the feed-back that we have experiment from the SHA representation to AL description yields to confirm the strength of SHA like representation approach.

Also the number of SHAs in KB is not known from the beginning. It represents number of paths found after the computation of the dependency graph.

CONCLUSION

CONCLUSION

In this work we have investigate fundamentals domains such as logics, knowledge representation and description logics. These fields related to the computational semantic which is a big challenge of modern Artificial Intelligence.

The major constraint that faces the development of intelligent systems is that when we achieve a high level of semantic integration we use complex procedure to manipulate the representation structure.

The Subsumption Hierarchical Attribute is simple and meaningful representation structure. It represents in simple form of vector carrying semantics, context and relational knowledge. The power of this representation comes from being derived from DLs representation formalism especially the meanings of subsumption relation between concepts. It's known that a reduction of any other definition in DLs to subsumption is possible using an automated generation of SHAs. This last is proved by the implantation.

The reasoning Algorithm readapted to the vectorial representation is a simple vectors comparator and generator, which shows that SHAM (Subsumption hierarchical Attribute) is an efficient method of knowledge representation. It yields to simplified reasoning Algorithm without any negative influence on knowledge system achieved performances.

In this work we presented the first step that consists on proposed approach. This work is based on DLs languages simplification. This is a real beginning of long studies and researches to respond to all currents problematic of DLs system when using SHAs representation.

The future works, will have a beginning by implementation of SHAMs system as first step which will be followed by the description of resources designed essentially for ALP systems as generic application field. These steps and others lead to the consolidation of our idea about the SHA representation. Also it's to be noted clearly that we have to proceed to integrate the augmentations of DLs languages to achieve more expressivity and study the opportunities to deal with complete and sound reasoning Algorithms.

Finally, this study enabled us to touch very interesting domains such as knowledge engineering, logics and semantics. It enabled us also to make links with some previously learned fields like cognition and cognitive psychology. This is a big motivation to continue our works on artificial intelligence and especially the knowledge representation domain.

REFERENCES

[1] Ian Horrocks, "optimising tableau decision procedures for description logics", PhD thesis university of Manchester, 1997.

[2] Uwe Keller, "Towards Novel Techniques for Reasoning in Expressive Description Logics based on Binary Decision Diagrams", Knowledge Web PhD Symposium 2007, Innsbruck, Austria, June 6, 2007.

[3] Stuart J. Russell and Peter Norveg. " artificial intelligence a modern approach", second edition, (Prentice Hall) Pearson Education international.2003.

[4] Chamak E. And McDermott D, "introduction to artificial intelligence" Addison-Wesley reading Massachusetts, 1985.

[5] Winston P.H., "artificial intelligence Addison-Wesley" reading Massachusetts, third edition, 1992.

[6] Pool D., Mackworth A. K., And Goeble R., "A computational intelligence: A logical approach", Oxford university Press, Oxford UK, 1998.

[7] Nilson N. J. "Artificial intelligence: A new synthesis", San Mateo, California 1998.

[8] Haugeland J., "artificial intelligence: the very Idea" MIT Press Cambredge, Massachusetts, 1985.

[9] Bellman R. E., "An Introduction to Artificial intelligence: can the computer think?" Boyd & Faster Publication Company, San Francisco, 1978.

[10] Kurzweil R., " the age of intelligent machine", MIT Press Cambredge, Massachusetts, 1990.

[11] Rich, E., and Knight, K., Artificial Intelligence (second edition), New York: McGraw-Hill, 1991.

[12] Nardi, D., Brachman, R.J.: An introduction to description logics. In: Baader et al. [13], pp. 5–44.

[13] Baader F., Calvanese D., McGuinness D., Nardi D., and. Patel-Schneider P.F, editors. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.

[14] Peter D. Karp, "The design space of frame knowledge representation systems", Technical Report 520, SRJ International Al Centre, 1992

[15] Lou GOBLE , The Blackwell Guide to Philosophical Logic, Blackwell: Malden, Mass., und Oxford 2001

[16] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. Studia Logica, 69(1):5–40, October 2001.

[17] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in expressive description logics. In A. Robinson and A. Voronkov, editors. Handbook of Automated Reasoning, pages 1581–1634. Elsevier Science Publishers (North-Holland), Amsterdam, 2001 (Chapter 23).

[18] Porter P., Lifschitz V. and HarmelenF. V., “Handbook of knowledge representation”, Oxford : Elsevier, 2007.

[19] M.R. Quillian. Word concepts: A theory and simulation of some basic capabilities. Behavioral Science, 12:410–430, 1967.

[20] M. Minsky. A framework for representing knowledge. In J. Haugeland, editor. Mind Design. MIT Press, 1981. A longer version appeared in The Psychology of Computer Vision (1975).

[21] W.A. Woods. What’s in a link: Foundations for semantic networks. In D.G. Bobrow and A.M. Collins, editors. Representation and Understanding: Studies in Cognitive Science, pages 35–82. Academic Press, 1975.

[22] R.J. Brachman. What's in a concept: Structural foundations for semantic networks. Int. Journal of Man-Machine Studies, 9(2):127–152, 1977.

[23] P.J. Hayes. "In defence of logic." In Proc. of the 5th Int. Joint Conference on Artificial Intelligence (IJCAI'77), pages 559–565, 1977. A longer version appeared in The Psychology of Computer Vision (1975).

[24] P.J. Hayes. The logic of frames. In D. Metzing, editor. "Frame Conceptions and Text Understanding", pages 46–61. Walter de Gruyter and Co., 1979.

[25] R.J. Brachman and J.G. Schmolze. "An overview of the KL-ONE knowledge representation system", Cognitive Science, 9(2):171–216, 1985.

[26] E. Mays, R. Dionne, and R. Weida. K-Rep system overview. SIGART Bull., 2(3):93–97, 1991.

[27] R.J. Brachman, R.E. Fikes, and H.J. Levesque. KRYPTON: A functional approach to knowledge representation. IEEE Computer:67–73, October 1983.

[28] Peltason C. The BACK system—an overview. SIGART Bull., 2(3):114–119, 1991.

[29] R. MacGregor. "The evolving technology of classification-based knowledge representation systems". In J.F. Sowa, editor. Principles of Semantic Networks, pages 385–400. Morgan Kaufmann, Los Altos, CA, 1991.

[30] B. Nebel. Reasoning and Revision in Hybrid Representation Systems. Lecture Notes in Artificial Intelligence, vol. 422. Springer, 1990.

[31] R.J. Brachman and H.J. Levesque, editors. Readings in Knowledge Representation. Morgan Kaufmann, Los Altos, CA, 1985.

[32] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. Artificial Intelligence, 48(1):1–26, 1991.

[33] L. Pacholski, W. Szwast, and L. Tendera. Complexity of two-variable logic with counting. In Proc. of the 12th IEEE Symp. on Logic in Computer Science (LICS'97), pages 318–327. IEEE Computer Society Press, 1997.

[34] P.F. Patel-Schneider, D.L. McGuinness, R.J. Brachman, L.A. Resnick, and A. Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. SIGART Bull., 2(3):108–113, 1991.

[35] R.J. Brachman. "Reducing" CLASSIC to practice: Knowledge representation meets reality. In Proc. of the 3rd Int. Conf. on the Principles of Knowledge Rep resentation and Reasoning (KR'92), pages 247–258. Morgan Kaufmann, Los Altos, CA, 1992.

[36] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. Artificial Intelligence, 48(1):1–26, 1991.

[37] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91), pages 151–162, 1991.

[38] B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption algorithms for concept description languages. In Proc. of the 9th Eur. Conf. on Artificial Intelligence (ECAI'90), pages 348–353, London (United Kingdom), 1990. Pitman.

[39] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In Proc. of the Workshop on Processing Declarative Knowledge (PDK'91), Lecture Notes in Artificial Intelligence, vol. 567, pages 67–86. Springer, 1991.

[40] P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: Preliminary report. In Proc. of the 1995 Description Logic Workshop (DL'95), pages 131–139, 1995.

[41] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representationand Reasoning (KR'91), pages 151–162, 1991.

[42] F.M. Donini,M. Lenzerini, D. Nardi, andW. Nutt. Tractable concept languages. In Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91), pages 458–463, 1991.

[43] K. Schild. A correspondence theory for terminological logics: Preliminary report. In Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91), pages 466–471, 1991.

[44] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. J. of Logic and Computation, 9(3):385–410, 1999.

[45] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors. Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99), Lecture Notes in Artificial Intelligence, vol. 1705, pages 161–180. Springer, 1999.

[46] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI'94), pages 205–212, 1994.

[47] G. De Giacomo and M. Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In Proc. of the 11th Eur. Conf. on Artificial Intelligence (ECAI'94), pages 411–415, 1994.

[48] G. De Giacomo. Decidability of class-based knowledge representation formalisms. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1995.

[49] G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96), pages 316–327, 1996.

[50] I. Horrocks. Using an expressive description logic: FaCT or fiction? In Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98), pages 636–647, 1998.

[51] V. Haarslev and R. Möller. RACE system description. In Proc. of the 1999 Description Logic Workshop (DL'99), volume 22 of CEUR Electronic Workshop Proceedings, 1999.

[52] P.F. Patel-Schneider. DLP. In Proc. of the 1999 Description Logic Workshop (DL'99), volume 22 of CEUR Electronic Workshop Proceedings, 1999.

[53] K. Schild. Querying knowledge and data bases by a universal description logic with recursion. PhD thesis, Universität des Saarlandes, Germany, 1995.

[54] A. Borgida. On the relative expressiveness of description logics and predicate logics. Artificial Intelligence, 82(1–2):353–367, 1996.

[55] L. Pacholski, W. Szwast, and L. Tendera. Complexity of two-variable logic with counting. In Proc. of the 12th IEEE Symp. on Logic in Computer Science (LICS'97), pages 318–327. IEEE Computer Society Press, 1997.

[56] E. Grädel, P.G. Kolaitis, and M.Y. Vardi. On the decision problem for two variable first-order logic. Bulletin of Symbolic Logic, 3(1):53–69, 1997.

[57] E. Grädel. Guarded fragments of first-order logic: A perspective for new description logics? In Proc. of the 1998 Description Logic Workshop (DL'98), volume 11 of CEUR Electronic Workshop Proceedings, 1998.

[58] E. Grädel. On the restraining power of guards. J. of Symbolic Logic, 64:1719–1742, 1999.

[59] M. Buchheit, F.M. Donini, W. Nutt, and A. Schaerf. A refined architecture for terminological systems: Terminology = schema+views. Artificial Intelligence, 99(2):209–260, 1998.

[60] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98), pages 149–158, 1998.

[61] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In Proc. of the 6th Int. Conf. On Principles of Knowledge Representation and Reasoning (KR'98), pages 2–13, 1998.

[62] F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), pages 364–369, 2005.

[63] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors. Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006), Lecture Notes in Artificial Intelligence, vol. 4130, pages 287–291. Springer-Verlag, 2006.

[64] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In M.M. Veloso and S. Kambhampati, editors. Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI-05), pages 602–607. AAAI Press/The MIT Press, 2005.

[65] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. Quonto: Querying ontologies. In M.M. Veloso and S. Kambhampati, editors. Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI-05), pages 1670–1671. AAAI Press/The MIT Press, 2005.

[66] Sergio Tessaris : Questions And Answers:Reasoning And Querying In Description Logic. a thesis submitted to the university of Manchester for PHD degree. April 2001.

[67] Birte Glimm : Querying Description Logic Knowledge Bases. a thesis submitted to the University of Manchester for the degree of PHD in the Faculty of Engineering and Physical Sciences, 2007.

[68] Yasser YAHIAOUI, Ahmed LEHIRECHE, Djelloul Bouchiha "A Proposed Reasoning Algorithm Using DLs in Vectorial Form" ACM digital library in proceeding of IPAC2015, Batna- Algeria- November 2015

[69] Woods W. A. & Schmolze J. G. "the KL-one family computers and mathematics with applications", Special Issue on Artificial Intelligence 23(2-5) 133-177, 1992.

[70] Baader F. Franconi E. Hollunder B. Nebel B. Profitlich H.J. “An Empirical Analysis of Optimization Techniques for Terminological Representation Systems” Applied Intelligence, 4(2):109-132, 1994

[71] Volker Haarslev & Ralf MMöller “High Performance Reasoning with Very Large Knowledge Bases: {A} PracticalCase Study” Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, {IJCAI} 2001 Seattle, Washington, USA, August 4-10, 2001

[72] Ulrich Faigle , György Turán “Sorting and recognition problems for ordered sets” Lecture Notes in Computer Science, Volume 182 pp 109-118, Date: 18 June 2005

[73] http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/ seen 10/01/2016

[74] John Mylopoulos. Information modelling in the time of the revolution. Information Systems, 23(3–4):127–155, 1998.

[75] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, Principles of Semantic Networks, pages331–361. Morgan Kaufmann, LosAltos , 1991.

[76] Yasser Y. Ahmed L. Djelloul B. “proposed representation approach based on description logics formalism” IJISA volume 8 (05) 2016.

[77] Yasser Y. Ahmed L “a Meta description logics knowledge base for Arabic language processing” The Third International Conference on Digital Information Processing and Communications(ICDIPC2013)- United Arab Emirates, 4/02/2013

APPENDICES

EXAMPLE

The human being TBox described in the simplified syntax.

```
women EQUIV %Conj(person,female)

man EQUIV %Conj(person,%Neg(women))

mother EQUIV %Conj(women,%EX haschild.Person)

father EQUIV %Conj(man,%EX haschild.Person)

parent  EQUIV %Disj(father,mother)

grandmother EQUIV %Conj(mother, %EX haschild.parent)

womenwithmanychildren EQUIV %Conj(mother,>2 haschild)

womenwithoutdaughter EQUIV %Conj(mother,%UNhaschild.%Neg (women))

wife EQUIV %Conj(women,%EXhashusband.man)
```