DORIN LELEA
Digitally signed by DORIN LELEA
T=Director Scoala Doctorala, C=RO, S=TIMIS, L=TIMISOARA, O=UNIVERSITATEA POLITEHNICA DIN TIMISOARA, E=dorin.lelea@upt.ro, SERIALNUMBER=201104209 DL10, G=DORIN, SN=LELEA, CN=DORIN LELEA
Atest integritatea acestui document
Computerul meu
2022.04.14 13:12

CONTRIBUTIONS TO THE DEVELOPMENT AND STANDARDIZATION OF THE SEMANTIC WEB

Ph.D. ThePh.D. aimed at the obtention of the Scientific Title of Doctor Engineer
at
Politehnica University of Timişoara
in the domain
Computer Science and Information Technology
by

Zamfira Constantin-Andrei

Scientific Supervisor: prof.univ.dr.ing. Horia Ciocarlie

Referents: 1. *prof. univ. emerit dr. ing. Victor-Valeriu PATRICIU*, Academia de Tehnică Militară București
2. *prof. univ. emerit dr. Viorel NEGRU*, *Univ. de Vest Timișoara*
3. *prof. univ. dr. ing. Mircea POPA*, *Univ. Politehnica Timișoara*

Sustenation date: 25 March 2022

BUPT

Series „Phd Theses” in UPT are:

1. Automation
2. Chemistry
3. Energetics
4. Civil Engineering
5. Chemical Engineering
6. Electrical Engineering
7. Electronics and Telecommunications
8. Industrial Engineering
9. Mechanical Engineering
10. Computer Science
11. Materials Engineering
12. Systems Engineering
13. Energetics
14. Information Technology
15. Materials Engineering
16. Engineering and Management

Politehnica University of Timişoara initiated the above series in the scope of disseminating the knowledge and results of the research enterprise inside the university's Doctoral School. These series contain, according to H.B.Ex.S Nr. 14 / 14.07.2006, the Ph.D. theses defended inside the university starting with October 2006.

Copyright © Editura Politehnica – Timişoara, 2013

This publication is subject to provisions of the copyright law. The multiplication of this publication, either complete or partially, translation, printing, illustrations reuse, expose, broadcasting, reproduction on film, or in any other format is allowed only in compliance with the provisions of the romanian law for copyright and usage permissions obtained in writing from the Politehnica University of Timisoara. Breaching any of these rights will be punished according to the romanian law for copyright.

Romania, 300159 Timisoara, Bd. Republicii 9,
Tel./fax 0256 403823
e-mail: editura@edipol.upt.ro

BUPT

Foreword

This thesis was elaborated during the course of my activities within the Department of Computer Science and Information Technology of the Politehnica University of Timişoara.

Special thanks deserve the supervisor, prof.univ.dr. Horia Ciocârlie, that understood and sustained me in many aspects during the entire cycle of my Ph.D. studies, without the help of which I wouldn't be here today to defend this thesis.

Same manner consideration (or even bigger) deserve the Department of International Relations of UPT which offered me the privilege, through the Erasmus exchange program, to visit and live in one of the most beautiful places in the most beautiful European country, France, such as the Atlantic ocean (Bretagne) and the Mediterranean sea (Cote d'Azur), and to have the honor to study in ones of the best universities out there and some great academic environments.

Timişoara, Martie 2022 Zamfira Constantin-Andrei

BUPT

For the Department of International Relations of UPT,

Zamfira, Constantin-Andrei

Contributions to the Development and Standardization of Semantic Web

UPT Phd Theses, Series X, Nr. YY, Editura Politehnica 2022, 200 pages, 39 figures, 17 tables.

Abstract,

"Semantic Web" is the name of the third generation from the evolution of the World Wide Web. The first generation was static in nature and laid the foundation on static pages and content delivery while the second was dynamic with emphasize on the online communities and social interactions. The Semantic Web have as main objective the automation of processes that exist on the Classic Web, allowing machines to execute tasks that currently can be made only by humans, such are those of finding, analyzing, processing, combining and/or sharing of information.

Semantic markups are annotations of Web pages with data about their content that can be understood by the agents on the Web. In order to ensure that different agents have a common understanding of the terms ontologies are used in which are defined the terms and their relations from a domain application , thus establishing a common terminology among agents. In the context of the Semantic Web ontologies play an important role in the sustaining of automated processes (intelligent agents) accessing the information. Particularly, they are used to provide structured vocabularies that describe terms and relations between them, allowing agents to interpret them correcly and unambiguously. Due to the fact that the Semantic Web is in the infancy of its evolution, there is currently a great need for those vocabularies with semantically structured data that describe the significance of web resources, as well as intelligent agents capable to execute the desired operations onto the resources, the most important being those of reasoning.

In the first part of this thesis I set out to begin with an introduction to the domain of discussion, the Semantic Web (chapter 1), followed by a chapter 2 in which I have made a series of discussions, debates, analyses about the foundation stone of the Semantic Web:ontologies, and their representations using logical formalisms, especially Description Logics and First Order Logics. In Chapter 3 I created a reasoning system for the logical knowledge bases that, besides the usual types of knowledges also contained another one called 'negative constraints', stated in the form of some restrictions on the data from the base. This is the first of its kind system in the literature that realize intensive computations of rules unification on the set of facts with the final goal to find out all the implicit knowledge from the explicitly stated ones, process that in logics is known as 'saturation'.

Part II focuses on the application of Semantic Web technologies into a domain of a great importance for our lives, that is cybersecurity. After a preliminary introduction into the domain that was provided in Chapter 4 , it follows a chapter of discussions and analyses regarding the way are utilized the techniques from Artificial Intelligence and Semantic Web in the construction of intrusion detection systems in order to enhance their performance. Chapters 6 and 7 are the contributions of this second part in the domain in cause. The former proposes an ontology of cybersecurity that is intended as a large model, inside which are defined terms of the domain as well as their relations and their natural language explanations. The ontology is used by a firewall to detect the nature of the occurred situations. Chapter 7 proposes an intrusion detection system in networks of computers that relies on technologies presented throughout this thesis to enhance the detection performance and lower the error rate.

The thesis ends with a chapter in which are drawn the final conclusions upon the work presented, emphasizing the contributions brought to the domain compared to the state of the art from literature, as well as the future directions of my research.

BUPT

TABLE OF CONTENTS

BUPT

BUPT

Notations, Shorthands, Acronyms

Part I – The Semantic Web and Description Languages

W3C	=	World Wide Web Consortium
WWW	=	World Wide Web
SW	=	Web Semantic
CERN	=	Conseil Europeen pour Recherche Nucleaire
HTTP	=	Hypertext Transfer Protocol
HTTPS	=	Hypertext Transfer Protocol (Secured version)
HTML	=	Hypertext Markup Language
XML	=	Extensible Markup Language
URI	=	Uniform Resource Identifier
URL	=	Uniform Resource Locator
URN	=	Uniform Resource Name
IRI	=	Internationalized Resource Identifier (i.e. a generalization of URIs to allow the use of Unicode characters)
IRL	=	Internet Resource Locator (described in RFC 1736, conveys location and access information for resources)
UNC	=	Uniform Naming Convention (a syntax used by Microsoft to describe the location of network resources,)
DARPA	=	Defense Advanced Research Projects Agency
DAML	=	DARPA Agent Markup Language
IETF	=	Internet Engineering Task Force
RFC	=	Request for Comments
IANA	=	Internet Assigned Numbers Authority
IPv6	=	Internet Protocol (version 6)
DNS	=	Domain Name System
FOL	=	First Order Logic
ML	=	Modal Logic
RDF	=	Resource Description Framework
RDFS	=	RDF Schema
RDFa	=	Resource Description Framework in Attributes
RDFS(FA)	=	RDFS with Fixed metamodeling Architecture
OIL	=	Ontology Inference Layer
SPARQL	=	Simple Protocol for Access RDF Query Language
RIF	=	Rule Interchange Format
SWRL	=	Semantic Web Rule Language
TURTLE	=	Terse RDF Triple Language
N3	=	N-triples
KMS	=	Knowledge Management Systems
SW	=	Servicii Web
OWL	=	Web Ontology Language
SOAP	=	Simple Object Access Protocol
WSDL	=	Web Service Description Language
UDDI	=	Universal Description, Discovery and Integration
B2B, B2C	=	Business-to-Business, Business-to-Consumer
P2P	=	Peer-to-Peer
OGP	=	Open Graph Protocol
FOAF	=	Friend-of-a-Friend

SIOC	=	Semantically Interlinked Online Communities
R2RML	=	Relational to RDF Markup Language, limbaj pentru a descrie asocierile dintre datele relationale si RDF
D2RQ	=	Relational Data to RDF Query
R2D	=	RDF to Database
P2R	=	Prolog to RDF, a tool that allows access to Prolog knowledge bases
SPASQL	=	SPARQL Support for MySQL
GRDDL	=	Gleaning Resource Descriptions from Dialects of Language
CWM	=	Closed World Machine
KR	=	Knowledge Representation
DL	=	Description Logics
$\mathcal{AL}$	=	Attributive Language, a language that offers atomic, universal, bottom, negation, intersection, value restrictions, limited existential restrictions
$\mathcal{ALC}$	=	Attributive Language with Complements (concept negations)
$\mathcal{L}(\mathcal{G})$	=	a $\mathcal{G}$-combinable language, where $\mathcal{L}$ is a DL and $\mathcal{G}$ a group of conformed datatypes
$\mathcal{S}$	=	language $\mathcal{ALC}$ extended with axioms of transitive roles ($\mathcal{ALC}_{\mathcal{R}+}$)
$\mathcal{SI}$	=	language $\mathcal{S}$ extended with roles inverses
$\mathcal{SH}$	=	language $\mathcal{S}$ extended with roles hierarchies
$\mathcal{SHf}$	=	language $\mathcal{SH}$ extended with functional roles axioms
$\mathcal{SHI}$	=	language $\mathcal{SH}$ extended with roles inverses
$\mathcal{SHQ}$	=	language $\mathcal{SH}$ extended with qualified number restrictions
$\mathcal{SHIO}$	=	language $\mathcal{SH}$ extended with roles inverses and nominals
$\mathcal{SHIQ}$	=	language $\mathcal{SHI}$ extended with qualified number restrictions
$\mathcal{SHIN}$	=	limbajul $\mathcal{SHI}$ extended with unqualified number restrictions
$\mathcal{SHOQ}$	=	language $\mathcal{SH}$ extended with nominals and number restrictions
$\mathcal{SHOIQ}$	=	language $\mathcal{SHOI}$ extended with qualified number restrictions
$\mathcal{SHOIN}$	=	language $\mathcal{SHOI}$ extended with unqualified number restrictions
$\mathcal{SHOIN(D)}$	=	language $\mathcal{SHOIN}$ extended with a universal concrete domain
$\mathcal{SROIQ}$	=	language $\mathcal{S}$ extended with nominals $(\mathcal{O})$, qualified number restrictions $(\mathcal{Q})$, roles inverses $(\mathcal{I})$, roles inclusion axioms and disjunction $(\mathcal{R})$

Part II - Intrusion Detection Systems

NIST	=	National Institute of Standards and Technology
IDS	=	Intrusion Detection System
IPS	=	Intrusion Prevention System
VLAN	=	Virtual Local Area Network
VPN	=	Virtual Private Network
IoT	=	Internet of Things
NBA	=	Network Behaviour Analysis
NIDS	=	Network Intrusion Detection System

A-NIDS	=	Anomaly-based Network Intrusion Detection System
HIDS	=	Host Intrusion Detection System
SPA	=	Stateful Protocol Analysis
SSL	=	Secure Socket Layer
SSH	=	Secure Shell
TCP	=	Transport Control Protocol
IP	=	Internet Protocol
TelNet	=	Teletype Network
NIC	=	Network Interface Card
NTP	=	Network Time Protocol
UDP	=	User Datagram Protocol
ICMP	=	Internet Control Message Protocol
IGMP	=	Internet Group Management Protocol
DMZ	=	Demilitarized Zone
(D)DoS	=	(Distributed) Denial of Service
VoIP	=	Voice over IP
DHCP	=	Dynamic Host Configuration Protocol
IMAP	=	Internet Message Access Protocol
IRC	=	Internet Relay Chat
NFS	=	Netwook File System
POP	=	Post Office Protocol
Rlogin/Rsh	=	Remote Login/Shell
SMTP	=	Simple Mail Transfer Protocol
SNMP	=	Simple Network Management Protocol
TFTP	=	Trivial File Transfer Protocol
SYN/TCP	=	Synchronize packet in TCP communication
MAEC	=	Malware Attribute and Enumeration Characterization
CAPEC	=	Common Attack Pattern Enumeration and Classification
SCAP	=	Security Content Automation Protocol
OVAL	=	Open Vulnerability and Assessment Language
CPE	=	Common Platform Enumeration
CCE	=	Common Configuration Enumeration
CVE	=	Common Vulnerabilities and Exposures

List of Figures

List of Tables

Page intended left blank

1. WHAT IS THE SEMANTIC WEB?

In this chapter will be made an introduction to the principal domain of this research, which is the last generation of the World Wide Web recently occurred, called the ”Semantic Web”. Will be made a foreword about the Internet and the most important service that runs on top of it, the World Wide Web, showing its functionality based on its architecture and the generations of evolution through which it went starting from its inception.

The section dedicated to the Semantic Web will be given the definition of the concept and made a discussion regarding its design goals, as they have been stated by its creator himself, sir Timothy Berners-Lee, and will be done a series of analyses regarding how the project will change for better the processes on the Classical Web. After this introduction into the domain in cause will be discussed and analyzed the architecture with its stack of technologies and standards created until now, as well as the open problems with which it confronts and which led to the impossibility of standardization of certain layers. The chapter will end with a discussion about the domains of industry where these newly developed technologies find the most applications.

1.1. Classic Web vs. Semantic Web

1.1.1. What Is the World Wide Web?

The Internet saw a major evolution throughout its history, and today it can be said that it looks more like a collection of applications than one of interconnected resources. It is a known fact that resources accessed over the Internet are not a collection of passive multimedia elements anymore, such as text, images, audio, video, etc., but coagulated into well-defined applications and services that rival even the desktop-class ones. Even though they are visually similar, Web applications differ in the way they handle user input, requests, access to data, concurrency, inter-applications communications, etc.

Many people use the terms” Internet” and ”Web” interchangeably; it should be clear though that they are not synonymous. The internet is a global system of interconnected networks that communicate between them with the TCP/IP suite of protocols. Some of the most important services that run over the Internet are: electronic mail, phone dialing (VoIP), file transfer and sharing, real-time communications, TV and radio transmissions, peer-to-peer networks (P2P), mobile applications, etc.; the Web is ”yet-another” service that runs into it, altogether with the ones enumerated above.

The World Wide Web (abr. WWW, or, simply, Web), was invented at the beginning of ’90 by the English scholar Tim Berners-Lee. He proposed the use of hypertext as a method for linking and accessing information over the Internet, like a Web with nodes. It is a system of interlinked hypertext documents, called Web

BUPT

pages that consist of multimedia content (text, images, video) that can be accessed and visualized using a Web browser. It operates based on a client-server architecture type where documents are stored onto a Web server and are accessed and rendered by a Web browser through their resource identifiers (URI, URL, URN). This scenario is shown in fig.1.

The day of 6 August 1991 is considered the debut of the Web as a publicly available service over the Internet. It was executed on a NeXT computer at the European Center of Nuclear Research (CERN), in Geneva, Switzerland. On 30 April 1993 CERN announced that WWW is a freely available service for everyone [36].

The three technologies that lie at the heart of the Web are:

- a system of unique global identification for the resources: Uniform Resource Name (URN), Uniform Resource Identification (URI), Uniform Resource Locator (URL)
- publishing language Hypertext Markup Language (HTML)
- communication protocol Hypertext Transfer Protocol (HTTP)

In the space of Internet and the Web, there are two main actors: clients and servers. A Web client is generally a browser, like Internet Explorer, Mozilla Firefox, Safari, Opera. A Web server is an application that receives and serves clients' requests based on the HTTP protocol (generally Web pages). The logical link between the 2 main actors is done using resource identifiers, which have been earlier enumerated. A URI allows the unique identification of a resource on the Internet. The application-level ISO/OSI on top of which are created the applications is HTTP, altogether with its secured version HTTPS [40]

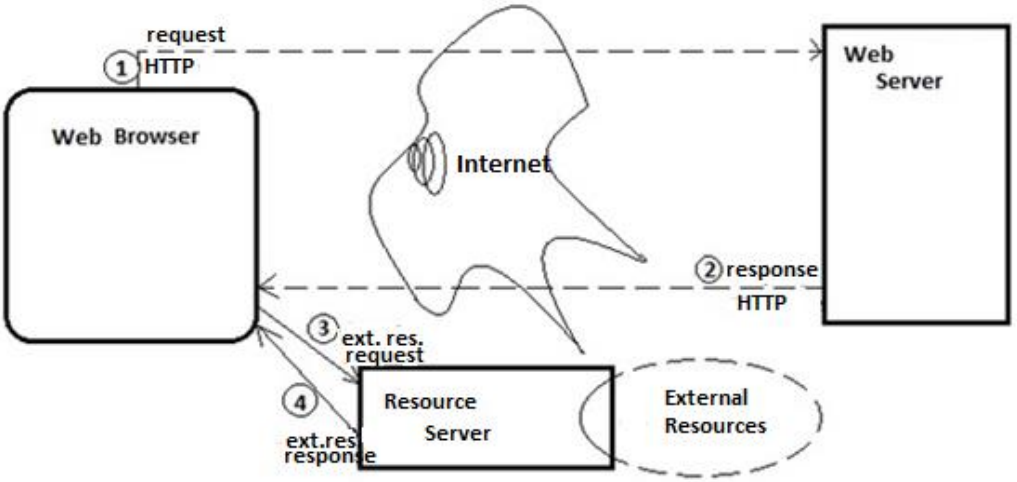

Fig. 1.1: The Client-Server type of architecture

The following scenario shows what happens between the moment of entering a URL into the browser's address bar and the rendering of the response Web page into the browser, as it was discussed in [36].

a) the Web browser translates the name of the server from the URL address into an IP address using a DNS

b) the browser asks for the resource by sending an HTTP Request message over the Internet to the computer located at that address
c) Request service from a TCP port number that is well known for the service HTTP for the destination host to be able to distinguish an HTTP request from another network protocol that he may be able to serve. HTTP in general uses port no.80.
d) the machine that receives the HTTP request forwards it to the Web server that listens for requests at port 80
e) if the server can fulfill the request then it sends an HTTP response message to the browser that indicates success and the page requested embedded within the message's body
f) the Web browser parses the HTML document from the response, interprets the markup code to render the text
g) if the page contains URIs of other resources (images, scripts) the browser will make additional HTTP requests to the server for each of these resources
h) after it receives its content the browser progressively renders the Web page by the way it is specified inside the HTML document

The fundamental property of the World Wide Web is its universality: a hypertext link is capable to relate ”anything” to ”anything”. The Web technologies are not capable to make difference between a raw sketch and sheer performance, commercial and academic information, or among various cultures, languages, or environments. Information varies in way too many aspects, such as that being created for consumption by humans or by machines. The former category enters TV shows, printed books, magazines, while the latter the software programs, databases, sensor outputs, etc. Until the present day, the World Wide Web was developed as an environment for humans and contains little information that can be automatically processed. The next generation, Web 3.0, known also as the ”Semantic Web” purposes to deal with this job [81].

Since its inception and until now the Web went through 3 generations of evolution. The first implementation was Web 1.0, which was characterized by static pages and content delivery, lacking any elements of user interaction and content sharing. The second generation, Web 2.0, focused on the possibility for humans to collaborate and share information online. Unlike the previous generation, this one was dynamic in the sense that serves clients' applications and provides open communication with an emphasis on online communities and social interactions. The Semantic Web is the generation that is currently in use today and a larger discussion will be made in the next section. For extra particularities regarding evolution and generations of WWW, I invite the reader to see the paper in [64] where he talks also about a fourth-generation that is seen as a Web of ultra-intelligence, electronic agents, and symbiotic and pervasive calculus. This generation is yet to emerge.

Next, I will offer the readers other introductory resources and materials on the domain. Maybe the most important of them is the book of its inventor, ”Weaving the Web” [41] in which the WWW is described broadly. Other papers with introductory purposes are written also by its author and have an equally high value are those in

BUPT

[36], [37], [38], [39], [132]. Others that talk about the constituent technologies, such as URI, HTTP, HTML, client-server architecture I invite the reader to see the works in [153], [177], [178].

1.1.2. What is the Semantic Web?

The Semantic Web is a term that was introduced at the beginning of the 2000s by no one else than the inventor of the World Wide Web and the founder of the consortium that deals with its development, W3C, the English scholar Tim Berners Lee. The main goal of this project is to drive the automation of processes from the classical Web, allowing machines to perform tasks that currently can be made only by humans, such as those related to finding, interpreting, analyzing, processing, combining, sharing information. That is not a separate entity from the classic Web but an extension of it, destined to add data and metadata to documents to extend them to data that has a semantic structure. This form of an extension to structured data allows the Web to be processed both automatically by machines and manually by humans. Is used as a synonym for the third generation of WWW, namely Web 3.0 [43], [64].

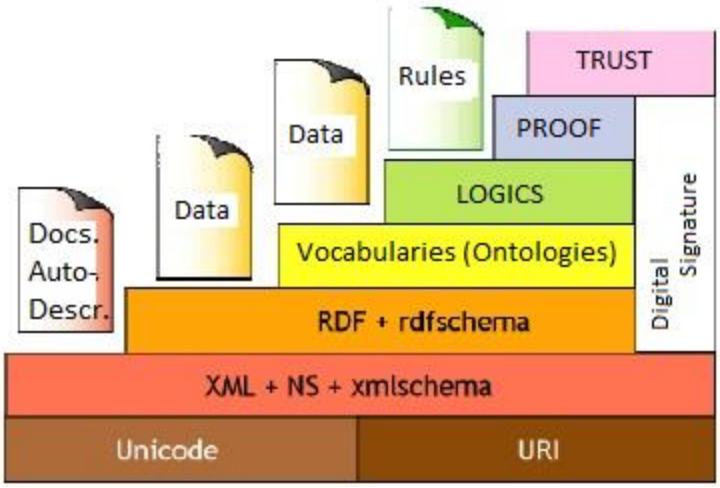

Fig.1.2: Semantic Web's stack of technologies

Figures 2 and 3 are shown the architecture with the component levels, as well as the technologies and standards that have been created for each level.

In the introductory article from the Scientific American magazine [42] his inventor narrated the ideas behind the project. He made use of examples to show the expected capabilities of the Semantic Web. He pictured it as an environment where tasks created by humans are conducted intelligently by Web agents that act in a manner that is similar to human cognitive processes. It is stated there that the personal Web agent of a patient obtains data about his prescribed medicines from the doctor's agent even before he even gives him directly, searches the Web to create a list with the pharmacies that have those drugs in store and keeps only those that are in his insurance plan and are located on

BUPT

a radius lower than 20 miles from his house and that has feedback of at least „Good" at their profile on the service trust sites. The agent then tried to find matches among the available meeting times from the providers' agents and the patient's working schedule and presented him with this plan. This scenario seems taken from the Hollywood movies that present humanity in the next centuries in which the world is dominated by intelligent robots and AI technology that is human-alike.

The concept of "Semantic Network" was introduced at the beginning of the '60s by the scientists Alan Collins, Ross Quillian, and Elizabeth Lofthus resembling a way to represent semantically structured knowledge. Extends the concept of a network of Web pages interconnected by links that can be read by humans by inserting metadata about pages that can be interpreted by machines, allowing thus the automated agents to access the Web more intelligently and to perform various users' tasks. Many of the technologies proposed by W3C existed before being laid under the umbrella of the Semantic Web that has been used in various contexts, especially those that deal with information about a limited and well-defined domain and the necessity of data distribution is a common fact, such as the domain of scientific research. Other technologies with similar purposes had been created, such as microformats [178].

The main goal of the Semantic Web is to drive further the evolution of the classic Web by allowing users to find, combine, share information much more efficiently. Humans are capable to use the Web for tasks such as translating a word between 2 languages, buying a book, searching for the lowest price for a DVD. Computers cannot realize all of these tasks because Web pages are created for humans to read them, and not machines. The Semantic Web is a vision of information that can be automatically interpreted by machines, thus computers can do the labor for finding, combining and processing information on the Web. The Semantic Web, as it was originally envisioned by its creator, is a system that allows computers to understand and respond to users' requests based on their significance. This 'understanding' requires the sources of information to be semantically structured. Is considered an integrator between distinct contents, applications, and systems with applications in multiple domains [177].

The fundamental technologies of its architecture, as it is shown in fig.2, are RDF, OWL, and SPARQL. The first two represent the formats in which data on the Semantic Web are published, and also play the role of some ontology languages in which ontologies are constructed. The third is the standard interrogation language for data on the Semantic Web. The layers for which had not yet been created standard technologies are: Unifying Logic, Trust, and Proof. This is due to the existing problems that we confront and that lay under the umbrella of those three domains, the most important of these are vastness, imprecision, inconsistency, deceit, as had been confirmed by the sources from Wikipedia [230].

Next, I will be making an overview of the three fundamental Semantic Web technologies that I stated earlier.

BUPT

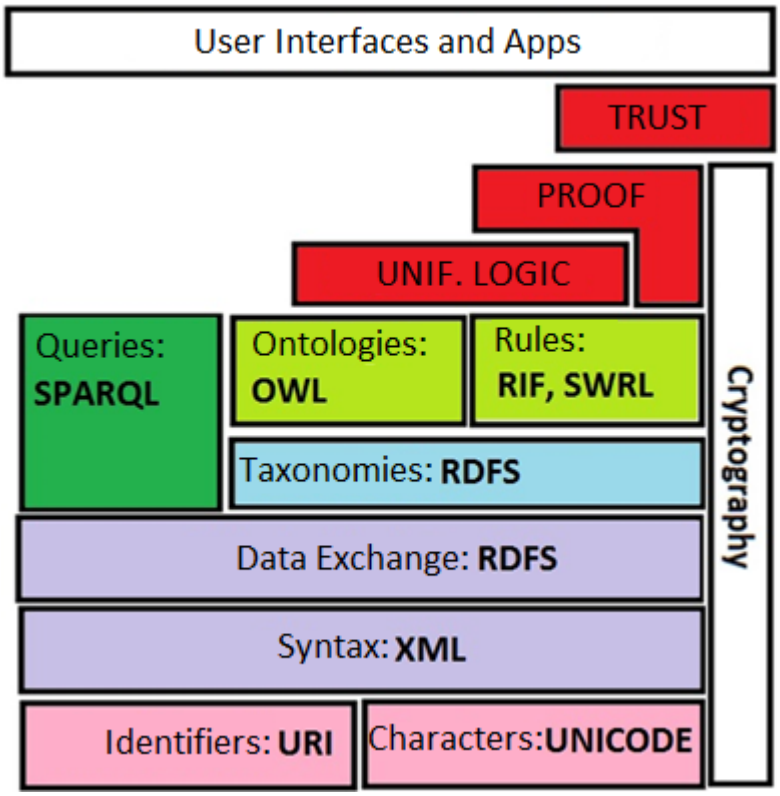

Fig.1.3: Architecture layers mapped to technologies

a) Uniform Resource Identifier (URI)

For identification of things on the Web, identifiers are used. Because a uniform system of identifiers is being used and because every identified thing is considered to be a resource, these are called Uniform Resource Identifiers (URI). A URI can be attributed to anything, and anything that has a URI is said to exist on the Web. They represent the fundamental notion of the Web.

A well-known form of URIs is the URLs (Uniform Resource Locator). Unlike URIs, a URL both identifies and locates. Because the WWW is too huge to be controlled by a unique organization, URIs are decentralized, i.e. nobody oversees who makes them or how are used. If some of the schemes (like HTTP) depend on centralized systems (such as DNS), others (like Freenet) are completely decentralized, i.e. it is not needed anybody's permission to create one. URIs can be created for things that we do not owe. Their syntax is controlled by IETF which published RFC2396 as a general specification for URIs [40]. W3C keeps a list of all URI schemes.

Even though this flexibility makes out of the URIs a powerful technology, brings all together with it also a series of troubles. Because anybody can

create a URI will come to multiple URIs representing the same thing. Moreover, there is no means to see if two URIs refer to the same resource. This way we would not be able to certainly tell what exactly means a certain URI. The ability to tell things about URIs is among the greatest characteristics of the Semantic Web. The Semantic Web is built over syntaxes that use URIs to represent data, usually in structures of triples, that is multiple triples of URI data that are stored in databases or interchanged on the Web making use of a set of syntaxes especially for that, called RDF syntaxes.

b) Resource Description Framework (RDF)

RDF is the first of the 3 fundamental technologies of the Semantic Web, together with OWL and SPARQL. Particularly, RDF is the universal model for data on the Semantic Web, that is all data of its technologies are being represented in RDF format, including the schemes that describe them, RDF Schema. RDF format is not similar to the tabular model from relational databases nor like XML trees; it is a directed graph with labeled edges. We can think of RDF as a set of nodes that are interconnected by edges, both wearing labels.

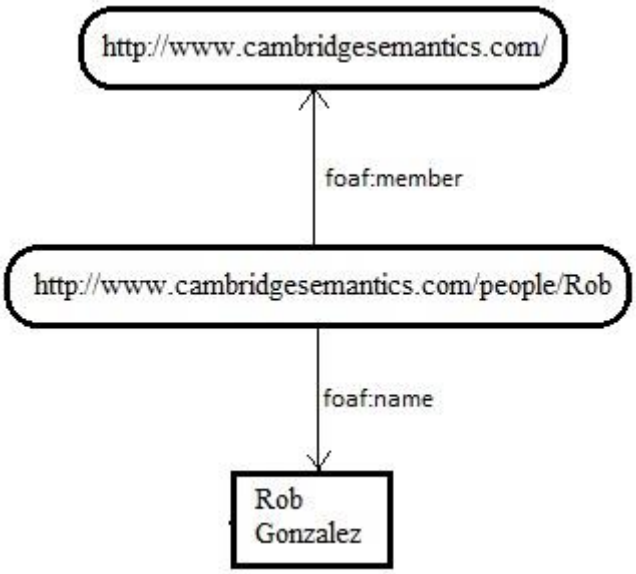

Fig.1.4: An example of an RDF graph

In an RDF graph can exist three types of nodes [177]:

- resources: a resource is anything about which something can be said, we can see it as a pair (resource, value); in the visual representation resources are indicated by ovals
- literals: here is with the sense of *value*; in the example, from fig.1.4 the resource is *http://www.cambridgesemantics.org/people/Rob,* and the value of the property foaf: name is the string of chars "Rob Gonzalez". In visual representation, literals are denoted by rectangles.
- void: a void node is a resource without a URI

In RDF resources and edges are URIs, literals are strings.
The most important formats for RDF data serialization are the following [177]:

- RDF/XML: XML style serialization of RDF, is the standard format for exchanging data on the Semantic Web
- Notation3: designed with the purpose of readability by humans and easiness of parsing
- N-Triples: a format simpler than Notation3
- Turtle (Terse RDF Triple Language)
- RDFa

c) Web Ontology Language (OWL)

OWL is the recommendation proposed by W3C in 2004 as a language for the representation of ontologies on the Semantic Web. It has been designed especially for the possibility to represent information about categories of objects and the relations among them (the type of knowledge that is usually known by the name of *ontology*), and also to be able to represent information about singleton objects (the type of information known as *data*) [15],[43].

Over its developing process had existed influences from multiple domains, such as representation languages (Description Logics, Frame Systems, Semantic Networks), and nonetheless RDF. Because it constitutes an effort from W3C from the development of the Semantic Web it should have fit into its stack of protocols and standards, altogether with XML, HTTP, RDF. Also, it should have kept as much as possible compatibility with other existing ontology languages, like SHOE, OIL, DAM+OIL. The many influences that wielded upon it led to the appearance of a number of problems, such are those related to its syntax, semantics, expressiveness and computational power, the tractability of reasoning, etc. The stalemates didn't come from each source in isolation but their combination and the constraints imposed on the language. The solution was found by the staff that dealt with its development that solves all the above-stated issues, which is the creation of 3 distinct profiles each fit for solving a certain set of those problems [112]:

- OWL-DL: created for the applications where the decidable inference of the reasoning problems and a friendly syntax are considered to be best suited
- OWL Lite: created for the applications where an even simpler syntax and an even more tractable inference
- OWL Full: created for the applications that need a great expressivity power and compatibility with other ontology languages (RDF, RDFS); it is completely expressive, a fact that makes it undecidable

A larger discussion around the main ontology languages of the Semantic Web, OWL and OWL2, will be made in Chapter 2, where I will talk about ontologies as the fundamental component of the functionality of the Semantic Web and the main representation languages will be presented.

Since this research's scope is not to provide an exhaustive description of the Semantic Web and its technologies, I will give the reader a few references to resources from literature where he can find multiple knowledge from the domain. I will begin with a few books that have been written by its authors, Berners-Lee and colleagues: [43], [81], as well as others written by other authors but of the same undoubtedly value: [15]. Among the papers with a review content about the domain, I mention firstly the ones of its creators [42], [44], [183]; some of the other authors read by me in the creation of this thesis are those in [16], [177], [178], [109], [64], [101]. Many others can be found in the references section at the end of the current document.

The next section will present the principal application domains of the technologies of the Semantic Web, as it was found by the author in the domain literature read for the creation of the current thesis.

1.2. Application Domains

1.2.1. Knowledge Management Systems

Knowledge Management Systems (KMS) describe techniques for representation, access, sharing, communication of knowledge through the collaboration and management of the knowledge as an organizations' asset. The Semantic Web is important for the KMS because it has the capacity to automate the aggregation and analysis of information, which has as result in growth in the processing speed. Has been designed with the goal to provide the missing elements of the classical Web, the main are: structure, metadata, and relations in order to correlate data and offer relations and descriptions. Ontologies are the fundamental component for the realization of the full potential of the Semantic Web. These offer background information for describing data and makes explicit the context of information. Ontologies are shared descriptions and can be used for the annotation of certain data sources, such as Web pages, collections of documents, databases, etc. This makes possible the interoperability process between different data sources but doesn't completely solve the integration problem because it is not possible for all the organizations on the Semantic Web to use a common terminology or ontology. It is most probable that will occur in different ontologies and, in order to allow interoperation mediation is necessary among these ontologies. Mediation is necessary in the semantic knowledge management process in order to realize the sharing of data among heterogeneous knowledge bases and, also to allow applications to reuse data from different bases. Another utility of the ontology mediation occurs in the Semantic Web services. Generally, it is not necessary that the client and the provider of a service to utilize the same terminology in their communication, and mediation is used especially to facilitate this goal.

1.2.2. Business Process Management

A business process that relies on automated information systems is known by the name E-Business. E-business software solutions allow the integration of processes of both inter- and intra-companies that are run over Web, Intenet, extranets etc. The application of Semantic Web technologies in the field of business

BUPT

process management has a significant role in the exchange of information between business groups for corporate purposes. It found itself a prominent role in the search for relevant data, information exchanges among agents, data filtering used at finding business sites, market trends analysis, integration and composition of complex systems, machine dialogs exchanges over different domains, multimedia collections, virtual community, flexibility and standardization of vocabularies, etc.

Information exchange among the business processes is a process that takes place by means of Web Services. These are software components having characteristics such as self-descriptiveness, self-contained, that expose their functionalities to consumers in order to provide interoperable machine-to-machine interactions. The specifications of the Web Services of an organization are exposed within public registers, such as a books catalog. The three main operations of a system of Web services are: registering, discovery and bounding. In order to extract services as optimal as possible registers (WS deposits) use ontologies developed in standard languages, such as SOAP, WSDL, UDDI sau OWL. Ontologies represent the building blocks in the development of e-business applications.

1.2.3. Electronic Commerce

The Web changed radically the availability of online data and the volume of information exchanged electronically, revolutionizing the access to personal information and knowledge management from the large corpora. Also, it has started to change commercial relations between clients and providers. Around 1% of the total amount of B2C transactions from the USA are made electronically, that is a small portion though but the growth perspective is very probable due to the continuous growth of Internet users. Electronic commerce is a field with a huge economical impact. Internet electronic commerce offers a much higher level of flexibility and openness that will support the optimization of business relations. Instead of providing a single link to each provider, it connects him with a big number of potential clients. Bringing e-commerce to its full potential requires a peer-to-peer (P2P) method. Anyone should be able to trade and negotiate with anyone, but unfortunately, this type of architecture meets a big number of obstacles before becoming reality. Some of these hindrances are: mechanized support for finding and comparing providers and their offers, for the work with multiple and heterogeneous data formats, and the work with multiple and heterogeneous business logics. The Semantic Web hold technologies and services that have the potential to solve the key problems from e-commerce in an efficient manner. Thus, the use of Semantic Web technologies to bring e-commerce to its full potential is a promising activity. It is also the natural means to link Semantic Web technologies and Web services. The transformation of latters from the status of advertising to a functional technology requires solutions to the problems enumerated above.

1.2.4. e-Science Knowledge Grids

E-science is the use of electronic resources by scientists that work in teams for big-scale and distributed projects. Science at a large scale, as has been stated by the Human Genome Project, will be in a continuous growing mode done by collaborations that are distributed at a global level activated by means of the Internet that will require access to some very large collections of data, computing resources and visualization tools of high performance. Biology already advanced to big inter-

BUPT

disciplinary teams distributed around the entire globe that works together on specific problems. Post-genomic experimentation and the data-intensive one expect to overwhelm the community with an avalanche of data that needs to be culled and organized. These data are often complex, generated by different environments, varying in quality, stored in multiple places, difficult to analyze, often fluctuating, and, mostly consisting of incomplete sets of data. Analysis methods for the processing of different types of data emerge and evolve constantly. The questions that we ask ourselves about data, and computational analyses are much more complicated: singleton or multiple species, the entire genome or a single gene, the entire metabolism, or a single biological process. The computing power necessary to model the metabolism path is huge, consequently, the traditional experimentation methods are augmented with ones "in silico", i.e. prediction of genes and metabolic paths that these encode from the DNA of an organism.

The Grid is a vision of some virtual organizations that means a distribution of resources that is flexible, secured, and coordinated. It is now seen as a platform for sustaining the above-mentioned operations at a global scale for data and computations-intensive applications. The major difference between the Grid and the classical Web consists of the big computing power, the large amount of data that can be processed, and the speed with which the data are transferred between the Grid's nodes. The Grid will provide equally large storing facilities and data extraction from a variety of sources and will present the data in the same format without counting any source.

1.2.5. Social Networks

Social Networks have become an important part of modern society and lay a huge impact on personal, social, political, educational, professional, and business life. They mediate people from the whole world by means of the social networking websites, such as Facebook, Twitter, MySpace, LinkedIn, Orkut, etc. Allow sharing of information on Twitter, messaging on Yahoo Messenger, Google Talk, distribution of the content and ideas on blogs and forums, media uploading and downloading, folksonomies by means of wikis and podcasts, and many others. Socialization on networks attracted millions of users around the globe and became the most popular, convenient, and cheap method of communication. A multitude of such sites arises in business due to the opportunity of large profits generated by these in the whole world. A large number of projects have been initiated and various tools developed in the field. General-purpose search engines assimilate more and more the newly developed technologies of the Semantic Web. Tumbup is a new search engine connected to Facebook with the purpose of analyzing the activities of members, such as recommendations of places, products, etc. to produce more satisfying results. Other search engines, such as Wolfram Alpha, True Know, or Zoom are also connected to Intranet sites and corporate blogs. Facebook introduced Open Graph Protocol, a technology of the Web Semantic that allows third-party sites to interact with the ones for social networks. This new protocol integrates Web pages in the social graph and sustains and supports the interactions between the visited sites and the profile of a user on Facebook. The Semantic Web has been used by many researchers and a lot of projects won momentum in the recent period, some of the most prominent include Friend-of-a-Friend (FOAF), Semantically Interlinked Online Communities (SIOC).

BUPT

Other domains that found applicability for the Semantic Web worth mentioning are biological information, medical and clinical data linking, manufacturing systems, constructions, transports, infrastructure. For additional material from this domain, I invite the reader to see a few works created for this purpose: [95],[155],[127], [80].

1.3. Current State of the Research

It was needed many years in order to put together the pieces of the puzzle that constitute the Semantic Wen, such as the standardization of the universal data language RDF, creation of the ontology language OWL, standardization of SPARQL that adds interrogation capabilities to RDF data, and many others. With the languages and standards 'in place', Semantic Web technologies started to be used. These technologies are popular, especially in domains such as research and life sciences, where they help researchers combine informa about different diseases and drugs that have different namings in different places around the globe.

In this section will be presented one of the most important applications and tools that have been constructed to support the development of the Semantic Web. These fall into the following categories: development environments, Web browsers, inference engines/reasoners, triple stores, transformers from the classical data models to semantic ones, search engines, etc.

1.3.1 Development Environments

Protégé is an open-source ontology editor and knowledge management system. Provides a graphical interface for the creation of ontologies, contains deductive classifiers for validating the models and inferring new knowledge based on the ontology analysis. Similar to Eclipse, it is a framework for which a lot of other projects create plugins. The application is written in Java programming language and relies massively on the Swing component in the construction of the graphical interface. Protege editor has been developed at Stanford University in California and was made available under the license of BSD2. The early versions of the tool had been developed in collaboration with the University of Manchester.

(Apache) Jena is a framework that was developed in the Java programming language and is used for the creation of Semantic Web applications. Has initially been developed by researchers at HP Labs from Bristol, Great Britain in 2000. Provides an exhaustive set of Java libraries in order to support developers in creating RDF, RDFS, RDFa, OWL, or SPARQL code conforming to W3C recommendations. Includes a rule-based inference engine for reasoning onto the RDF or OWL ontologies and a variety of strategies for storing RDF triples in memory. Jena is similar to Sesame but unlike that one, it provides also support for the creation of OWL ontologies. The framework contains many intern reasoner systems, and also Pellet, an open-source OWL reasoner written in Java can be set for working with Jena.

Sesame is an open-source framework for the interrogation and analysis of RDF data. Was created by the Dutch software company Aduna as part of the Semantic Web project "On-To-Knowledge". Contains the implementation of a triple store in memory and one on disk, altogether with 2 servlet packages that can be used to manage and provide access to stores from a server. The RIO package (RDF Input/Output) contains an API for the RDF parsers and writers from Java. The

parsers and writers for popular RDF serializations are distributed with the framework, and the users can extend their list by putting their parsers inside their Java applications' Classpath. Supports two interrogation languages: SPARQL si SeRQL and has another component, Alibaba which is an API for the mapping of Java classes to ontologies, and vice-versa. This makes possible the use of ontologies directly in Java code.

Ontotext: an ontology development environment written in Java at Stanford University, California.

TopBraid Composer: an ontology development environment written in Java at Stanford University, California.

1.3.2. Inference Engines (reasoners)

FaCT++ is an enhanced version of FaCT and is implemented also in C++. Uses tableaux algorithms of the description logic SHIQ (D) in the reasoning process and relies on various strategies such as absorption, models combination, cycles elimination, synonym replacement, heuristics for ordering, and classification of taxonomies. Its main disadvantages are the limited user interface and reasoning services compared to other existing reasoners.

Pellet uses reasoning in description logics SHIN and SHON, being implemented also in Java. The reasoning strategies implied are: Tbox partitioning, nominals support, absorption, semantic branching, slow unveiling, dependency-oriented jumps. Reasoning on data types, individuals, ABox queries optimization makes it very suitable for robust Semantic Web applications. Offers standard reasoning services for OWL ontologies including various optimization techniques such as nominals, conjunctive query answering, incremental reasoning.

Racer is a reasoner implemented in functional language LISP with the goal to prove the tableaux calculus for SHIQ and makes use of multiple optimization techniques for enhancing the reasoning, like dependency-oriented backtracking, axioms transformation, models reunion, caching, etc.

HermiT makes use of SHIQ(D) reasoning and is freely available for non-commercial use. Takes at input the OWL file and makes different inference tasks such as consistency checking, subsumption identification between classes, computation of the partial order of classes from the OWL file, and others. Differs from other reasoners by the use of hyper-tableaux algorithms, which are much less deterministic than the normal tableaux ones.

F-OWL is an inference engine for OWL data that relies on frame systems for the reasoning over ontologies task. Reads the OWL ontology from a URI, extracts the RDF triples, and converts them to a format that is conforming with the frames style, which is literally introduced into the engine. Makes use of "flora" rules defined in the Flora-2 language to verify the ontology's consistency and deduce hidden knowledge.

Hoolet relies on First Order Predicate Logic (FOL). The ontology is translated into a collection of axioms and is sent to the FOL reasoner for the verification of consistency. It is then extended for processing rules by adding a parser for the RDF rules syntax. Relies on an OWL-DL reasoner with support for SWRL rules. Makes use of a primitive technique that is related to the ones for homogeneous translations. The reasoning support is given by the direct translation of the ontology to a collection of axioms that are communicated to the theorem prover Vampire for the checking of consistency. The used technique is not scalable and FOL is undecidable.

KAON2 is based on OWL DL and F-Logic and it is an infrastructure for the management of ontologies that are created in OWL DL, SWRL, or F-Logic. Supports answers at conjunctive queries, without non-distinctive variables. Is the successor of the KAON project, which was used as an extension at RDFS. The idea behind this implementation is to reduce the SHIQ knowledge based on declarative logical programs (Datalog) disjunctive with finite DL rules, reason on the hybrid logic by reusing optimization techniques from databases. It translates an ontology and a rule into an axiom by making use of a common logical language, which is why is considered to form part of the class of homogeneous transformation techniques. Currently cannot handle nominals, if an ontology contains a class *owl:oneOf* or a restriction *owl:hasValue*, which are actually other names for nominal concepts, then any task of reasoning will throw an exception. Also cannot deal with very large numbers found in cardinality formulae. Problems have been remarked also in an ontology that contains a maximum cardinality restriction out of two: the reasoner is not capable to answer queries on it.

1.3.3. Triple Stores

AllegroGraph is a closed-source triple store that was created for storing of RDF data, which is the standard format of linked data. Currently, it is being used in open-source, commercial projects and those of the Defense Department. It is also the component for data storing of the project TwitLogic, which has as its main objective to bring Semantic Web technologies over the data on Twitter. Has been developed to meet the W3C standards for RDF, so it is considered an RDF database. It's a reference implementation of the SPARQL protocol, a query language for linked data having the same purpose as SQL for relational databases. It was proposed by the Swiss company Franz Inc., which created also Allegro Common Lisp, an implementation of Common Lisp, which is a dialect of the functional language Lisp. Its functionality is being provided in the languages Java, Python, Lisp, and other APIs. The first version of AllegroGraph has been made available at the end of the year 2004.

Virtuoso Universal Server is a middleware hybrid engine and database that combines the functionality of a traditional RDBMS, Objectual-RDBMS, virtual database, file server, and web applications, RDF, XML, simple text into a single heterogeneous system. Instead of having dedicated servers for each of the above functionality domains, Virtuoso is a universal server that has a unique multi-thread server process that implements multiple protocols. The software has been created by the company OpenLink Software, the open-source edition being known under the name OpenLink Virtuoso.

Mulgara is a triple store and a branch of the project Kowari. It is open-source, scalable, transaction resilient and their instances can be interrogated using languages such as iTQL and SPARQL. Does not use a relational database due to a large number of joins on tables from the relational systems when working with metadata. In exchange, this is a completely innovative database optimized for the management of metadata. Its models hold the metadata in the form of some sentences, subject-predicate-object, very like the triples from RDF data. Metadata can be imported and exported from Mulgara in RDF format or Notation3.

1.3.4. Data Transformers

One of the important categories of tools created in order to support the evolution of the Semantic Web is the ones for transforming (converting) from the classic data models to the semantic ones, such as would be relational data to RDF, XML to RDF, etc. Conversion of relational databases towards RDF triples stores that can be queried using the SPARQL language is known under the name RDB-to-RDF.

In September 2012 W3C launched the recommendation R2RML as a standard language for describing the associations between the relational data and RDF, which represented a milestone in the evolution of the Semantic Web. R2RML encourages developers of RDB-to-RDF to conform with a standard mapping language. These tools are classified as R2RML or non-R2RML.

One of the first researches that had been done in this domain was that of Teswanich et al. [190], which created a tool for the transformation of RDF documents and schema into relational databases. They affirmed that this association is good for avoiding the learning curves associated with the new tools and to benefit from the advantages of classic relational tools without losing the ones of the new Semantic Web technologies and standards. Their proposed tool is called RDF2RDB and does data replication, RDF triples data are put into a relational schema and requires information about schema definitions, information that are stored into ontologies. Problems related to their solution are especially those related to the large memory space consumed by the duplication of the RDF store and the need for synchronization of the changes between the 2 deposits. These problems had been addressed by the next generation of tools that had been created for this purpose.

D2RQ is a platform for accessing data from relational databases in the form of virtual read-only RDF graphs without being needed to be replicated into an RDF store. Was proposed by Bizer et al. [46] in 2004 and consists of 3 major components:

- D2RQ Mapping Language: a declarative language for describing the relations among the ontological and relational data models
- D2RQ Engine: a plugin for the Semantic Web toolkits Jena and Sesame that uses mappings for rewriting the API calls into SQL queries for the relational model and transmits the results of these queries to the above layers
- D2R Server: an HTTP server that is used for providing a view of the type LinkedData, one of type HTML for debugging, and a SPARQL endpoint on top of the database

Using D2RQ, we can: interrogate a non-RDF database using the SPARQL language, access information from a non-RDF base using the APIs provided by Jena and Sesame, access the content of a database under the form of data linked across the Web, and does SPARQL interrogations onto the database. The supported databases include: Oracle, SQL Server, MySQL, PostgreSQL, HSQLDB, Interbase/Firebird.

A tool that is very related to the one previously presented is R2D (RDF-to-Database) which has been proposed by Ramanujam in [160]. This has as main goal to transform the RDF data at runtime into an equivalent normalized relational schema, acting as a bridge between the two models and making available the existing relational tools also for the RDF stores. In comparison to other tools with similar purposes, R2D held the capacity to process also RDF void nodes and containers. Void nodes are those that are neither URIs nor literals but are being used to

BUPT

associate a resource with a set of properties that represent some complex data. They are one of the main components of RDF graphs and the focus from the development of R2D has been put on their relationalization. Also enhanced were the translations between SQL to SPARQL interrogations that now have pattern matching and data aggregation techniques.

(CWM) Closed World Machine is a software system for data processing of general-purpose for the Semantic Web. Has been created by the father of it, Tim Berners-Lee [216], [220] between the years 2000 and 2010. It is similar in purposes to what Sed and Awk are for simple text files, or XSLT is for XML. It is a semantic reasoner that uses the forward chaining technique that can be used for tasks such as interrogation, transformations, verifications, and filtering of information. The language at its foundation is RDF extended with rules and could use the RDF/XML or Notation3 serializations. It is capable to make the following operations:

- pretty parsing and printing of RDF formats: XML, Notation3, N-triples
- storing of triples into a deposit that is similar to a table
- making inferences by using the forward chaining techniques
- realization of other functions, such as string comparisons, resources extraction, all of them using a suite of extensible plugins

GRDDL (Gleaning Resource Descriptions from Dialects of Languages) [219] is a tool for obtaining RDF data from XML documents, particularly XHTML pages. HTML pages creators can associate the documents with transformation algorithms, generally represented in XSLT, by using an <link> element in the header of the document. Alternatively, information required for obtaining the transformation can be stored in an associated metadata document or a names space. Clients that read the documents can follow links across the Web by making use of techniques described in the GRDDL specification to discover the corresponding transformations.

Among other examples of tools from the category of mapping of data models worth mentioning: RDF123 [96], Triplify [17], SquirrelRDF, P2P, SPASQL, RelationalOWL, METAmorphose.

1.3.4. Public Ontologies

DBpedia is an effort to publish structured data extracted from Wikipedia. Data are being published in RDF format and put on the Web for use under the GNU Free Documentation license, thus allowing the automated agents to realize inferences and interrogations over the data

FOAF (Friend-of-a-Friend) is a popular vocabulary on the Semantic Web that uses RDF to describe relations among people and environmental things. Allows intelligent agents to understand the many connections between people, them and their services, or with other things from their life.

SIOC (Semantically Interlinked Online Communities) provides a vocabulary of terms and relations that model data spaces on the Web. Examples include: blogs, discussion forums, feeds subscriptions, etc.

GoPubMed is a search engine based on knowledge of biomedical terms. Allow users to find information much more easily than other engines do. Gene Ontology and Medical Subject Headings serve as tables of content for structuring millions of articles from the database MedLine.

NextBio is a database that consolidates large volumes of experimental data from domains such as life sciences, marked and connected through biomedical ontologies. Can be accessed through an inference engine having a graphical interface. Researchers can contribute with their solutions by introducing data inside the base which supports data about genes and proteins expressions and sequence-oriented data, but it constantly extends to support others.

Eagle-i is an open-source Semantic Web platform for the introduction and publishing of information about the resources that are used in biomedical research. Contains a component called SWEET (Semantic Web Entry and Editing Tool), an RDF database, a search engine. All the components are organized using an ontology to create interoperability with other platforms. The software, documentation, and other information are accessible from the Harward University of Medicine.

Dublin Core

Tables 1 and 2 make a survey with respect to the capabilities brought by the new Semantic Web technologies, as they have been presented during the course of this chapter, and compares them with the ones of the traditional Web, as they have been stated in the literature read in conducting this research. The results are convincing enough, in my opinion.

Characteristic	**Semantic Web**	**Classic Web**
Universal representation of data	RDF	X
Reusable data models	RDF, OWL	X
Intrinsic distributed data models	RDF, OWL	X
Standard interrogation languages	SPARQL	X
Validation, classification, and processing of information	Inference engines, reasoners, classifiers	X
Offer descriptions of content structures in the form of semantics	metadata	X
Tasks automation based on machine-readable semantics	Deductive inference, reasoning	X
Industrial application domains	lots	few

Table 1.1**:** New characteristics of Semantic vs. Classic Web

Characteristic **Model**	**Databases**	**XML**	**RDF**	**OWL**
Expressivity	medium	small	medium	large
Accessibility	small	medium	large	large
Flexibility	small	medium	medium	large
Inference	small	small	medium	large

Table 1.2: Comparisons among Semantic and Classical Web models

1.4. Conclusions

This chapter represents the introduction of the current thesis, this is why it has more of a review scope, that is to introduce the reader to the domains approached by my research. I began with a section where I presented the World Wide Web, starting with its history, evolution through the 3 generations until the present day, architecture, and the three fundamental technologies.

Then I moved forward and talked about the principal domain of my research, which is the Semantic Web. This is a term synonymous with the third generation of the Web, i.e. Web 3.0, having as its main purpose to conduct the evolution of the classical Web towards a Web of structured data that can be automatically processed by machines by annotating the resources with semantic contents. The three fundamental technologies of this Web are: URI, RDF, and OWL, each of these being discussed briefly in the chapter. There are also levels in the architecture which have not yet been created standards, like Unifying Logic, Trust, and Proof. This fact is due to the big number of problems that exist under the umbrella of these domains, problems that had been placed by experts into the broader field of uncertainty.

In another section are presented the industrial domains in which the Semantic Web find most applications, among these I chose to discuss business process management, knowledge management systems, e-commerce, e-science grids, and social networks. The chapter concludes with another important section in which is discussed the current state of researc intoh the domain, in which are presented the most important developments that had been brought to sustain the growth of the Semantic Web project.

As contributions to this chapter I can enumerate the reviews of the 2 domains that are for interest of my research with the aim to give the reader a conceptual image of it, analyses and comparisons between the technologies of the classical Web and the ones of the new Semantic Web aiming to evidentiate the advantages of the latter, how are they expected to change things in this world and, nonetheless our lives. These analyzes are presented in the form of plain textual narrations and graphically with tables.

BUPT

page intended left blank

BUPT

2. LOGICS AS SUPPORT FOR THE SEMANTIC WEB

In this chapter, I will discuss the extremely important roles that logical knowledge representation formalisms have in supporting processes on the Semantic Web, especially those for the creation of ontology languages. Ontologies play a crucial role on the Semantic Web, being used especially to create descriptions of the meanings of resources, descriptions that are subsequently being used by automated agents in the realization of different tasks, such as inferences and reasoning over the knowledge. It has been noticed by scientists a similarity between ontologies and logical knowledge representation systems, such are the Description Logics (DLs), which led to the design of ontology languages on top of them.

.

I will present the main ontology languages that have been created as technologies of the Semantic Web, then I will move forward and discuss the logical formalisms domain in which I will provide the reader with definitions, will make comparisons between the existing formalisms. Further on I will present a domain that has a bigger importance for the Semantic Web, that of Description Logics. Here it will be shown the most important languages proposed by the domain literature, make analyzes and comparisons between their capabilities of representation, then I will talk about the inference problems from a logical knowledge base and present the main categories of algorithms that had been constructed in literature for their solvation with a focus on the most important ones, the tableaux algorithms. The chapter concludes with a section in which will be stated a series of complexity results of reasoning in some DL languages together with their proof, as it has been proposed in the literature by the scholars of the DL domain.

2.1. Ontologies

2.1.1. Introduction

"Ontology" is a term borrowed from philosophy, where it refers to the science that deals with describing the types of entities from the real world and the existing connections among them. In Computer Science, an ontology is an explicit specification of a conceptualization of a domain from the real world and has as purpose to offer a common shared vocabulary that contains the most important concepts of the domain together with their properties and constraints. Ontologies are being used for defining a common vocabulary of an application domain, vocabulary that can be shared, reused, exchanged in various heterogeneous systems, by humans, automated agents, etc. [109].

The Semantic Web has as its main purpose the creation of resources that can be understood by computers and whose information can be shared and processed both by automated tools and also humans. This distribution of information among

BUPT

agents require some semantic markups, which are annotations of Web pages with information about their content that can be understood and interpreted by the agents on the Web. To make ensure that different agents have a common understanding of terms ontologies are used in which are defined the terms and relations from a domain application, and that thus establishes a common terminology between agents. In the Semantic Web context, ontologies play a key role in sustaining the automated processes (intelligent agents) to access the information. Particularly, they are used to provide structured vocabularies that describe the terms and their relations, allowing agents to have a correct and non-ambiguous interpretation of their meanings. For example, a Pizza ontology can hold information such as Mozarella and Gorgonzola are some specialties of cheese, that cheese is not meat, and that a vegetarian pizza is one whose toppings do not contain meat or fish. This information allows the term ”Pizza garnished with Mozarella and Gorgonzola” to be unambiguously interpreted as a specialization of the term ”Vegetarian Pizza”. Terms whose meanings are defined in ontologies can be used in semantic markups that describe the content and functionality of resources accessible over the Web [154].

Reasoning is an important process in the quality insurance of ontologies. In the design phase, it is used for finding contradictions among concept definitions and deriving implicit relations. In integration and interoperability phases of ontologies is being used for the computation and testing of the concept hierarchy, while in deployment to determine the consistency of the facts set, infer new relations between individuals, etc. [111].

The use of ontologies requires a well-defined specification language and that is also compatible with the Web and existing tools. Its syntax should be both intuitive for humans and compatible with the existing Web standards and its semantics should be formally specified in order to be able to provide a shared understanding. Nevertheless must be taken into account the expressivity power, i.e. the language should be sufficiently expressive to offer sufficient details in the concept expressions and relations but not excessive so that it makes the reasoning an undecidable process (impossible, untractable). The creation of the RDF Schema language represented an early attempt at an ontology language. Due to the fact that the constructors offered for the creation of ontologies were some basic ones, more expressive languages have been proposed in the literature, like SHOE, DAML+OIL, OWL [112].

Ontologies are a technology for knowledge representation that is used especially in domains that require the storage of some large volumes of data and making inferences over them, such as deduction, consistency checking, query answering, etc. Some of the most important domains are: eScience, Medecin, Geography, Geology, Agriculture, Defence, and nevertheless the Semantic Web. Its applications are prevalent in life sciences, which had been used by the developers of large-scale ontologies, such as SNOMED [187], a clinical terms ontology that contains more than 400.000 concepts and 400 relations that is currently being used at Columbia Presbyterian Medical Center, or BioPAX (Biological Pathways Exchange) [145], GO (Gene Ontology) [192], MGED (Microarray and Gene Expressions Data Ontology) [144], as well as many others. For example, biologists use ontologies to annotate data of their genes sequencing experiments in order to make possible the answering to complex queries, such as: ”what DNA link products interact with insulin receptors?”. To be able to answer this question is needed a reasoner that not just identifies the individuals that are (possibly implicitly only)

BUPT

instances of the link products between DNA and insulin receivers but also identify what pairs of individuals are linked (possibly implicitly only) by the property *interactsWith* [112].

Other examples of ontologies applications are:

- United Nations Food and Agricultural Organization (FAO): uses OWL in order to develop a wide range of ontologies from the domains of agriculture, fishing, etc.
- Semantic Web for Earth and Environment Technologies (SWEET): are a series of ontologies that had been developed at the US National Aeronautics and Space Agency (NASA), the laboratory of Jet Propulsion. These implement ontologies that describe space, the biosphere, the Sun, etc. It is currently being extended with a number of efforts about the sciences of space and earth and it was augmented in the GEON project in order to cover solid ground, and in the project, Virtual Solar-Terrestrial Observatory to contain more information about the atmosphere [162]
- An ontology used by General Motors in a project to sustain activities of enhancing the quality of the processes from the assembly line in various production sites

For introductory material into this domain, readers are referred to [109], [90], [91], [92], [98] where are related to the basic notions about the Semantic Web and its fundamental technology, ontologies. The ones interested in assimilating more advanced notions I invite to read a series of valuable works of some scholars in the domain, such as Horrocks [110], [111], [112], [114], Baader [25], [30]. They focus especially on the ontology languages that have been created for the Semantic Web, such as DAML+OIL and OWL, and put emphasis on their relationship with the Description Logic languages (DL) and how they had been developed on top of these.

2.2.2. Ontology Languages

Standard ontology languages of the Semantic Web that had been created by W3C are OWL (2004) and OWL2 (2009). Since these have briefly been described in chapter 1 I will not be recalling them again here but discuss others that had been created before them and influenced their design and development. For OWL and OWL2 I will offer a larger discussion in the next section where I will talk about the DL application domains, ontology languages being one of these.

SHOE

One of the first attempts at creating an ontology language for deployment on the Web was SHOE. This is a language that relies on the Frames Logic and has an XML syntax that allows it to be embedded into XML documents. Relies on URI references for names, an innovation that has subsequently been taken by other languages. Laid emphasis on the fact that ontologies will be tightly connected and subject to exchange. Thus, SHOE included a number of directives that allow the importing of existing ontologies, local renaming of imported constants, and specification of information about versioning and compatibility among ontologies. This way of thought influenced the extra-logic vocabulary of OWL which was designed in order to partially deal with this sort of problems. SHOE had a lower

BUPT

influence on the syntax and semantics of OWL due to the fact that did not rely on RDF and did not have formal semantics.

DAML-ONT

In 1999 was started the DAML program (DARPA Agent Markup Language) with the main goal being that to lay the ground for a future generation of"semantic" Web. As a first step, it has been decided that the adoption of a common ontology language will facilitate the semantic interoperability along various projects that constitute the program. RDFS, which already was proposed as a standard by W3C, has been seen as a starting point but it was not sufficiently expressive to meet DARPA's needs. A new language has been created, called DAML-ONT that extended RDF with constructors from the Object-Oriented and Frames representation paradigms. It was tightly integrated with RDFS but, even though this was useful from the perspective of compatibility, it led to the rise of some serious problems in the design. Same as RDFS, DAML-ONT suffered from a semantically inadequate specification and it was soon found out that this could lead to disagreements related to the exact meaning of terms from an ontology. Moreover, the DAML-ONT properties restrictions, same as in RDFS, had a global scope more than a local one and, even if this was benefic for domains and properties restrictions of RDFS, the global cardinality restrictions, for example, are hard to understand or have doubtful usability.

OIL

Approximately in the same time with the development of the DAML-ONT language, a group of European researchers with objectives similar to the ones of DARPA proposed a new ontology language, OIL (Ontology Inference Layer). This was the first ontology language that tried to mix elements from Description Logics, frames languages, and Web standards (e.g. XML, RDF). A strong focus was put on the formal restrictions and was explicitly designed in order for its semantics to be specified by means of a translation to the DL SHIQ. The structure though was relied on the frames paradigm, using compound classes definitions in the style that has been presented in Section 1. OIL has both XML and RDF syntaxes, but, even though the RDF syntax has been designed in order to maintain compatibility with RDFS it was not concerned with precise details about the RDF semantics, that were not formally defined at that time.

DAML+OIL

It became obvious for the previous both groups that their objectives could have been best fulfilled by combining their efforts together, the result being the DAML+OIL language. The development of the language has been taken by a committee mostly comprised of members from the 2 teams and the institution Joint US-EU Ad Hoc Agent Markup Language Committee. The unified language has some formal semantics given by its own model theory in the DL style, instead of a translation to an equivalent DL. The DL-derived constructors of the OIL language had been inherited also by the new language, but the frames structures have been mostly banished in the favor of DL axioms that were easier to integrate with the RDF syntax. Influenced by DAML-ONT it is more strongly related to RDF but provided significance for those parts of RDF that were consistent with its syntax and with the DL-style model theoretics. This didn't seem like a big problem due to the fact that RDF didn't have at that time a proper formal specific significance and this was the cause for some serious problems when DAML+OIL was used as the foundation for the newer ontology language, OWL.

2.2. Description Logics

2.2.1. Introduction

Description Logics (DLs) are a family of logic-based knowledge representation formalisms that have been created in order to be able to represent and reason over the data from an application domain in a structured and easy-to-understand manner. Rely on a family of common languages, called descriptions, that have a set of constructors for creating concept descriptions (classes) and roles (relations). These descriptions are used as axioms and assertions from the knowledge bases and could be reasoned by means of DL reasoners. On the other side, DLs are well equipped with logic-based formal semantics and differ from their predecessors, like Frames or Semantic Networks [25].

As it has been already stated, good quality ontologies have vital importance for the Semantic Web, their construction, integration, and evolution depend greatly on the existence of some good semantics and powerful reasoning tools. As DLs have both of them, that makes them some ideal candidates for the ontology languages. This fact was known ever since 30 years ago but back then there existed a major discordance between the capabilities of one of the technologies and the requirements of the other. Due to research in DL over the last decades, this gap has become sufficiently narrow to be able to build stable bridges [110].

Description Logics have distinctive logical properties. They put the focus on the decidability of important reasoning problems, like the satisfiability of the concept descriptions or knowledge base, providing decidable reasoning services, like the tableaux algorithms that are used to deduce implicit knowledge from the ones explicitly stated. Highly optimized reasoners, such as FaCT++, Racer, HermiT demonstrated that tableaux algorithms for very expressive DLs can lead to a good performance even when are being applied over large knowledge bases [25].

DLs are characterized by ttouctors provided in order to create complex concepts and roles descriptions out of the atomic ones.an The basic language is $\mathcal{AL}$ (acronym for Attributive Language). This offers constructors just for conjunction ($\cap$), value and existential restrictions ($\forall$,$\exists$). The other languages of the DL family are extensions of this one, each newly added constructor is being associated with a letter in the language's name. For example $\mathcal{ALC}$ is $\mathcal{AL}$ extended with concept negations (C-complements), $\mathcal{ALC}_{R+}$ is $\mathcal{ALC}$ extended with transitive roles (R+). In modern DL languages, $\mathcal{S}$ is considered as a minimal language, being used as a shorthand for $\mathcal{ALC}_{R+}$ in the purpose of reducing name size. It was named so due to its relation with the propositional modal S(4) logic. The other languages of the family are obtained from this by the addition of new constructors in order to be able to represent new characteristics [154].

Most important of these constructors are:

- $\mathcal{H}$: role inclusion axioms (role hierarchies)
- $\mathcal{R}$: disjunction
- $\mathcal{O}$: nominals (singleton classes {x})
- $\mathcal{I}$: roles inverses
- $\mathcal{N}$: number restrictions ($\leq n\ R,\ \geq n\ R$)
- $\mathcal{Q}$: qualified number restrictions ($\leq n\ R.C,\ \geq n\ R.C$)
- $\mathcal{F}$: functional number restrictions

The most well known and largely used languages of the S family are: $\mathcal{SI}$, $\mathcal{SH}$, $\mathcal{SHI}$, $\mathcal{SHf}$, $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, $\mathcal{SHIN}$, $\mathcal{SHOIN}$. Table 1 shows the constructors of this family that are being used at the creation of concept descriptions.

Constructor	**Syntax**	**Semantics**
universal (top)	T	Δ^I
nothing (bottom)	$\perp$	ϕ
concept name	CN	$CN \subseteq \Delta^I$
general negation	$\neg C$	$\Delta^I \backslash C^I$
conjunction	$C \cap D$	$C^I \cap D^I$
disjunction	$C \cup D$	$C^I \cup D^I$
existential restriction	$\exists R.C$	$\{x \in \Delta^I \mid \exists y.(x,y) \in RI \wedge y \in C^I\}$
value restriction	$\forall R.C$	$\{x \in \Delta^I \mid \exists y.(x,y) \in R^I \rightarrow y \in C^I\}$

Table 2.1: Syntax and semantics of the constructors of DL $\mathcal{S}$

Leţs take a look at how to create a complex concept from a set of atomic concepts and roles by making use of the constructors of that language.
Let *Plant* $\in C$ be a concept ame, and *eat, partOf* two roles. Then $\forall$*eat.(Plant*$\cup$ $\exists$*partOf.Plant)* is a concept in accord with the syntax that is shown in Table 1.

2.2.2. History and Evolution

The history of Description Logics started with the attempt to formalize the Semantic Networks, which, in their turn represented some attempts of offering a sort of labeled graphs-based natural representation. A major problem with Semantic Networks was that there did not exist any formal meaning for the graphs, a fact that leads to conflicts regarding what complex graphs actually mean when were removed from a system that provided data modeling facilities. Another early influence on Description Logics had the Frame Systems that shared many of the Semantic Networks traits but they grouped related information into a frame.

KL-ONE was a knowledge representation system that shared many of the traits of Semantic Networks and Frame Systems. Not long after its creation had been attempts to provide full-fledged formal semantics to KL-ONE. Together with these formal semantics KL-ONE can be seen as the first proto DL. [50].

In the rest of this section I will make a brief presentation regarding the history of Description Logics.

Research in DL can be partitioned into 4 phases, as it was stated by [25]:

Phase 0 (1965-1980)

This is the pre-DL phase in which had been introduced the Semantic Networks and Frame Systems as specialized techniques for representation of knowledge in a structured manner, which later were criticized due to the lack of any kind of formal semantics. A method for the solvation of these problems represented the Structural Inheritance Networks that were proposed by Brachman, which have been implemented into KL-ONE, the first DL system.

Phase 1 (1980-1990)

This phase was especially concerned with systems implementation: KL-ONE, K-REP, KRYPTON, BACK, LOOM. These systems relied on the so-called *structural subsumption algorithms*, that firstly normalized the concept descriptions then recursively compare the syntactic structure of these descriptions. These algorithms are usually efficient in polynomial time but have the drawback that they are complete only for small expressive DLs, thus for the more expressive ones cannot detect all inclusion and instance relations. During this phase had been created the first logical considerents of the semantics of formalisms that made possible formal investigations in the complexity of reasoning for DLs. For example, in [50] was shown that the smallest additions to the expressiveness of the representation formalism can lead to the intractability of the inclusion problem. Schmidt-Schauβ [173] showed that inclusion in the language from the basis of KL-ONE is undecidable, and Nebel [146] that the use of a Tbox formalism that allows for introduction of abbreviations for complex descriptions makes subsumption undecidable if the DL allows for conjunction and value restriction constructors (these have been supported by all DL systems at that time). As a reaction to these negative complexity results the developers of the CLASSIC system, the first industrial-scale DL system, have carefully restricted the expressivity power of the system.

Phase 2 (1990-1995)

Began with the introduction of a new algorithmic paradigm for DL, that is the so-called *tableaux algorithms*. These algorithms work over the DLs that are propositionally closed, that is those which contain all boolean operators, and are complete also for the most expressive languages of the family. In order to decide the consistency of a knowledge base, a tableaux algorithm creates a model of it by structurally decomposing the concept descriptions, this way infers new constraints on the model's elements. The algorithm terminates either when all attempts to build a model failed with obvious contradictions or ends with a canonical model. As in propositionally closed DLs the subsumption and instance checking can be reduced to consistency, this means a consistency checking algorithm is able to resolve all the reasoning tasks stated above. Among the first systems that employed these algorithms were Kris and Crack which proved that optimized implementations of these algorithms led to a good behavior of the system even if the worst case complexities of the corresponding reasoning problems are no longer polynomials. This phase has known also a profound analysis of the complexity of reasoning in various DLs and the important observation that DLs are strongly related to Modal Logic.

Phase 3 (1995-2000)

This phase is characterized by the development of the inference procedures for very expressive DLs, either being relied on the tableaux method or on a translation into Modal Logic. Highly optimized systems, like FaCT, Racer, DLP proved that tableaux algorithms for very expressive DLs led to an acceptable system behavior in practice even when processing large knowledge bases. In this phase the relationship with Modal Logic and decidable fragments of FOL has been thoroughly studied and its applications in the databases domain have been investigated, such as schema reasoning, query optimization etc.

Currently, we are in phase number 4 where the results of previous phases are being used to develop industrial-scale DL systems that employ very expressive DLs, having applications into the Semantic Web and knowledge representation and integration in domains such as medicine and bioinformatics. In the academy, the interest in less expressive DLs grew with the purpose of creating tools that are able to process very large knowledge bases (both terminological and assertional).

2.2.3. DL Knowledge Bases

A DL knowledge base contains two types of information [154]:

- intensional (TBox, RBox): general knowledge about the application domain
- extensional (ABox): knowledge about a specific situation of the world

a) TBox

A TBox (terminological) is a finite set of statements of the form $C \subseteq D$ (concept inclusion), $C \equiv D$ (concept equivalence). Statements from the TBox are known under the name of *terminological axioms*.

A TBox is *general* if it has a finite set of concept axioms and it allows cycles and general concept inclusions (GCI).

Example:

$WildAnimal \equiv Anumal \cap \neg \exists owner.\top$

$Mammal \cap \exists body\text{-}part.Hunch \equiv Camel \cup Dromedary$

A TBox is *inextensible* if contains only primitive concept definitions, the concepts' names occur at most once on the left-hand side, and do not contain cyclic definitions or GCIs.

Example:

$Elephant \equiv Mammal \cap \exists body\text{-}part.Trunk$

$Mammal \equiv Elephant \cup Lion \cup Zebra$

Reasoning in the presence of a TBox is much more difficult than without it, especially if the definitions contain cycles. A terminological cycle is a recursive concept inclusion, or one or many mutual recursive inclusions.

Example:

$Person \subseteq \forall hasParent.Person$

$Person \subseteq \exists hasParent.Mother,\ Mother \subseteq \exists hasChild.Person$

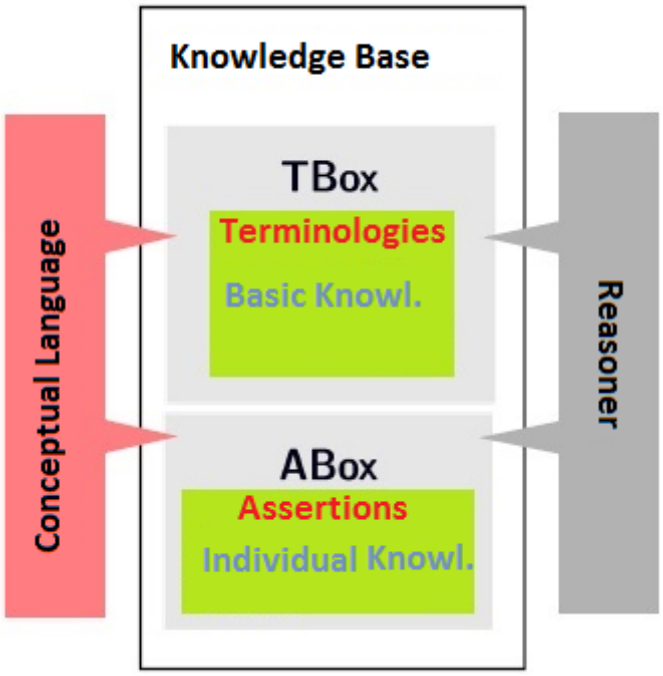

Fig.2.1: Architecture of a DL system

b) Rbox

Let $\mathcal{L}$ be a descriptive language, RN,SN∈ R roles names, $R_1,R_2 \in R_{dsc}(\mathcal{L})$ two $\mathcal{L}$-roles. A RBox (roles) is a finite set of propositions of the form:

- Func(RN), or RN∈F, where F⊆R is a functional role set
- Tranz(SN), or SN∈R_+, where $R_+ \subseteq R$ is a set of transitive roles
- $R_1 \subseteq R_2$ (roles inclusion), $R_1 \equiv R_2$ (roles equivalence)

The statements from RBox are called role axioms. The types of axioms that can occur into a Rbox depend on the expressivity of that language. The languages that offer roles axioms are the ones of the $\mathcal{S}$ family, starting from the most primitive one, $\mathcal{S}$, which, as has been stated in the previous section is $\mathcal{ALC}$ extended with transitive role axioms. Similarly to constructors, each role axiom is associated with a letter in the name of the DL (e.g. $\mathcal{F}$ for functional roles, $\mathcal{R}_+$ for transitive roles, $\mathcal{H}$ for roles inclusions). We can extend the $\mathcal{S}$ language with role inclusion axioms and obtain the language $\mathcal{SH}$, which in its turn can be extended to $\mathcal{SHf}$ by including functional roles axioms. An interesting fact is that certain restrictions exist between the three types of axioms, in that the set of functional roles (f) must be distinctive from the transitives one ($\mathcal{R}_+$), this is due to the fact that it is still an open problem whether or not DLs that employ both functional and transitive roles are decidable.

c) Abox

Let $\mathcal{L}$ be a descriptive language, a,b∈I individual names, C∈ $C_{dsc}(\mathcal{L})$ a $\mathcal{L}$-concept, R∈ $R_{dsc}(\mathcal{L})$ a $\mathcal{L}$-role. An ABox (assertional) is a finite set of propositions of the form:

- *a:C* – concept assertions
- *(a,b):R* - role assertions

An ABox describes a certain situation of the world with respect to some individuals, of an application domain in terms of concepts and roles. The statements from ABoxes are called *assertions* (individuals axioms). For example:

grass1:Plant, stone1: Earth, Ganesh, Bokhara:Elephant
(Ganesh,grass1):eat, (Bokhaexisteat

There exists two important assumptions about ABoxes. The first is called "Unique Name Assumption" (UNA), which says that if a,b∈ I are two distinct individuals then their interpretations with respect to I are also distinct (denoted $a^I \neq b^I$). Without UNA it should be stated a different assertion in order to assign that a and b are two distinct individuals.
The second one is called "Open World Assumption" (OWA), which states it cannot be assumed that all knowledge from the base are complete. This fact is intrinsic in that an ABox (or, more generally, a knowledge base) can have many models, only a few of the aspects lay under the constraints of assertions. For example, role assertion (*Ian, Jeff*):*hasPhdStudent* states that *Ian* has a Ph.D. student, *Jeff* in all its models, in some of these *Jeff* is the only Ph.D. student of *Ian* while in others *Ian* can have also other PhDs [154].

2.2.4. Syntaxes and Semantics

a) Syntaxes

All DL family languages have the same syntax, which is represented by concept and role names, concepts and role constructors, composite expressions, etc. In this section, I will try to make a brief description of them.
Unlike FOL, DL names (of concepts, roles, individuals, etc.) do not contain variables [25].
The atomic types signature is:

- concept names (classes): A, B (the equivalent of unary predicated from FOL)
- role names (relations, properties): r, s (the equivalent of binary predicates from FOL)
- individuals: Anna, John, Italy (the equivalent of constants from FOL)

Concepts operators:

- booleans: negation (¬), conjunction (∩), disjunction (∪)
- a restricted form of quantifiers: existential (∃), value (∀)
- numericals: $(\leq it, \geq, >, =)$

BUPT

Role operators:
- selection (.)
- composition (o)

Special concepts:
- T (top, thing, general), used as shorthand for AU ¬A
- ⊥ (bottom, nothing, inconsistent), used as shorthand for A∩ ¬A

In what follows I will show some examples of constructing complex concepts and roles expressions from atomic ones by making use of the language constructors. Also, I will specify the equivalent formulas from FOL syntax.
Examples of schema axioms:
- Rich⊆ ¬Poor (concept subsumption)
- Cat ∩ ∃sits-on.Mat ⊆ Happy (concept intersection)
- BlackCat ≡ Cat ∩ ∃has-color.Black (concept equivalence)
- sits-on ⊆ touches (role inclusion)
- Trans(part-of) (transitivity)

These are equivalent to the following FOL statements:

- ∀x.(Rich(x) → ¬Poor(x))
- ∀x.(Cat(x)Λ∃y.sits-on(x,y)ΛMat(y)→Happy(x))
- ∀x.(BlackCat(x) ↔ Cat(x)Λ∃y.(has-color(x,y)ΛBlack(y)))
- ∀x,y.(sits-on(x,y) → touches(x,y))
- ∀x,y,z.(sits-on(x,y)Λsits-on(y,z) → sits-on(x,z))

Examples of data axioms (facts) from FOL:

- concepts: BlackCat(Felix), Mat(Mat1)
- roles: sits-on(Felix,Mat1)

b) Semantics

Description Logics have model theoretic semantics defined in terms of interpretations. An interpretation I is formed from a domain $\Delta^{\mathbf{I}}$ and a function $.^{I}$, where the domain is a set of objects and the function relates each individual name a∈ I to an element $a^{I} \in \Delta^{\mathbf{I}}$, each concept name CN to a subset $C^{I} \in \Delta^{\mathbf{I}}$ and each role name RN∈R to a binary relation $RN^{I} \subseteq \Delta^{\mathbf{I}} \times \Delta^{\mathbf{I}}$ [154].

The interpretation function can be extended to concept expressions in the obvious way.

$(C \cup D)^{I} = C^{I} \cup D^{I}$

$(C \cap D)^{I} = C^{I} \cap D^{I}$

$(\neg C)^{I} = \Delta^{\mathbf{I}} \backslash C^{I}$

$\{x\}^{I} = \{x^{I}\}$

$(\exists R.C)^I = \{x| \exists y.(x,y) \in R^I \wedge y \in C^I\}$

$(\forall R.C)^I = \{x| \exists y.(x,y) \in R^I \rightarrow y \in C^I\}$

$(\leq n\ R)^I = \{x| \#\{y|(x,y) \in R^I\} \leq n\}$

$(\geq n\ R)^I = \{x| \#\{y|(x,y) \in R^I\} \geq n\}$

I will present below an example of how to create an interpretation $I=(\Delta^I, .^I)$ of a given concept: $\forall eat.(Plant \cup \exists partOf.Plant)$.

Domain is given by the set of individuals and roles:

Δ^I = {Ganesh, Bokhara, Balavan, grass1, stone1},

The interpretation function is defined as following:

$Plant^I$ = {grass1}

eat^1 = {(Ganesh,grass1), (Bokhara,stone1)}

$partOf^I = \Phi$

Thus we have the following interpretations:

$(\exists partOf.Plant)^I = \Phi$

$(Plant \cup \exists partOf.Plant)^I$ = {grass1}

$(\forall eat.(Plant \cup \exists partOf.Plant))^I$ = {Ganesh, Bokhara, grass1, stone1}

c) Semantics of Knowledge Bases

An interpretation I satisfies (models) a TBox axiom A (denoted I |= A) [154]:

$I \models C \subseteq D$ iff $C^I \subseteq D^I$, $I \models C \equiv D$ iff $C^I \equiv D^I$

$I \models R \subseteq S$ iff $R^I \subseteq S^I$, $I \models R \equiv D$ iff $R^I \equiv D^I$

$I \models R^+ \subseteq S$ iff $(R^I)^+ \subseteq S^I$

I satisfies a TBox T (denoted I |=T) if satisfies any axiom of it.

An interpretation I satisfies (models) an ABox axiom A (denoted I |= A):

$I \models x:C$ iff $x^I \in C^I$, $I \models (x,y):R$ iff $(x^I,y^I) \in R^I$.

I satisfies an ABox A (denoted I |= A) if it satisfies any axiom from it.

I satisfies a knowledge base K (denoted I |= K) if satisfies both its T and A.

2.2.4. Relationships with Other Logics

In this section I will talk about the relations that exist between DL and other logical formalisms, namely Predicatives and Modals. I will consider that the reader is familiar with these formalisms; but for those that are not I recommend some introductory materials in the domain, such are those of Stanford [237], [238]. In this section I will discuss and analyze the relationship between the primitive DL $\mathcal{ALC}$ together with certain extensions of it with these logics. For readers willing for a

BUPT

deeper analysis of these logics is recommended the work [47] as well as chapter 4 of Baader's vast book [28].

a) DLs and Predicate Logic

Most of DL languages are being seen as fragments of First Order Predicates Logic (First Order Logic - FOL), even though there exist ones that contain operators (like transitive roles closure or fixpoints) that require Second-Order Logic. The main reason for using DLs instead of general FOL at knowledge representation is that the majority of DLs are decidable fragments of FOL [25], which means there exist efficient procedures for deciding the main inference problems about which I will talk largely in section 3 of the current chapter where will be presented the reasoning services and techniques from DL knowledge bases.

If concept names are seen as unary relations and role names as binary ones, we can define two translation functions π_x şi π_y that inductively map $\mathcal{ALC}$ concepts to first-order formulas with one free variable, x or y.

$\pi_x(A) = A(x)$ $\qquad$ $\pi_y(A) = A(y)$

$\pi_x(C \cap D) = \pi_x(C) \wedge \pi_x(D)$ $\qquad$ $\pi_y(C \cap D) = \pi_y(C) \wedge \pi_y(D)$

$\pi_x(C \cup D) = \pi_x(C) \vee \pi_x(D)$ $\qquad$ $\pi_y(C \cup D) = \pi_y(C) \vee \pi_y(D)$

$\pi_x(\exists r.C) = \exists y.r(x,y) \wedge \pi_y(C)$ $\qquad$ $\pi_y(\exists r.C) = \exists x.r(y,x) \wedge \pi_x(C)$

$\pi_x(\forall r.C) = \forall y.r(x,y) \rightarrow \pi_y(C)$ $\qquad$ $\pi_y(\forall r.C) = \forall x.r(y,x) \rightarrow \pi_x(C)$

This being said, a TBox T and an ABox A can be translated the following way, where ψ[x/a] denotes the formula obtained from ψ by replacing each occurrence of x by a:

$\pi(T) = \bigwedge_{C \subseteq D \in T} \forall x.(\pi_x(C)) \rightarrow \pi_x(D))$

$\pi(A) = \bigwedge_{a:C \in A} \pi_x(C)[x/a] \wedge \bigwedge_{(a,b):r \in A} r(a,b)$

As it can be easily observed, this translation preserves the semantics: we can see the DL interpretations the same as the ones from FOL and vice-versa, and it can be easily shown that the translation preserves the models. The direct consequence of this fact is that reasoning in DL is equivalent to the inference from FOL.

Theorem 1. Let (T,A) be an $\mathcal{ALC}$ knowledge base, C and D $\mathcal{ALC}$ concepts (possibily complex) and *a* an individual name. Then:

- (T,A) consistent *if* π(T) ∧ π(A) consistent
- (T,A) |= C⊆D *if* (π(T) ∧ π(A)) → (π({C⊆D})) valid
- (T,A) |= a:C *if* (π(T) ∧ π(A)) → (π({a:C})) valid

This translation not only provides an alternate manner for defining the semantics of $\mathcal{ALC}$ but at the same time tells us that all the inference problems for $\mathcal{ALC}$ knowledge bases are decidable. Actually, the translation of a knowledge base uses only the variables x and y, and thus yields a formula in the two variables fragment of FOL, which is known to be decidable in exponential non-deterministic time (NExpTime) [28]. Alternatively, we can rely on the fact that this translation uses quantification only in a restricted form, so yields a formula into the guarded

fragment which is known to be decidable in exponential deterministic time (ExpTime). So, the exploration of the relation between DL and FOL gives us even "free" complexity upper bounds. For $\mathcal{ALC}$ though (and many other DLs) the upper bounds obtained this way are not necessarily the optimal ones, fact that justifies the development of the dedicated reasoning procedures for DLs.

The translation of more expressive DLs may be direct or more difficult, depending on the additional constructors. Role inverses may be easily captured in both guarded and two-variable fragments by simply interchanging variables. For example:

$$\pi_x(\exists R^-.C) = \exists y.R(y,x) \Lambda \pi_y(C).$$

Number restrictions may be captured using equalities or number quantifiers. It is a known fact that the two variables fragment with numbers quantifiers is decidable in exponential non-deterministic time. Transitive roles instead can't be expressed with the help of only two variables, while the three variables fragment is known to be undecidable. The guarded fragment, when constrained to the so-called 'action' guarded fragment can still be able to capture a variety of characteristics, like number restrictions, role inverses, fixpoints and still be decidable in exponential deterministic time [28].

b) DLs and Modal Logic

Description Logics have a strong relation to the Modals, even though they have been individually created. This similarity has been discovered relatively late, but since then it has been successfully exploited in order to transfer decidability and complexity results as well as reasoning techniques. Is not hard to notice that $\mathcal{ALC}$ concepts can be considered as syntactic variants of the multi-modal logic **K** formulas: the Kripke structures can be seen as DL interpretations (and vice-versa). We can see thus concept names as propositional variables and role names as modal parameters, and I'll do this correspondence by means of the rewriting operator ↔, that allows $\mathcal{ALC}$ concepts to be translated into modal formulas (and vice-versa).

ALC concept		**Modal Formula K**
A	↔	*a,* for a concept A and propozit. variable *a*
C∩D	↔	C ∧ D
C∪D	↔	C V D
¬C	↔	¬C
∀r.C	↔	[r]C
∃r.C	↔	(r)C

Let's denote by $\dot{C}$ the modal formula obtained after rewriting of the $\mathcal{ALC}$ concept C.

The translation of the DL knowledge bases is a more complex problem. A TBox T is satisfied only inside those structures where, for each C⊆D , $\neg\dot{C}$ V $\dot{D}$ globally preserves (i.e. in any world of our Kripke structure, or equivalently, in any element of our interpretation domain). We can express this by means of the universal modality, which is a special modal parameter U which is interpreted as

total relation in all Kripke structures. Before moving forward to ABoxes, I will define the properties of the correspondence by now[25].

Theorem 2. Let T be an $\mathcal{ALC}$ Tbox; E, F two (possibly complex) $\mathcal{ALC}$ concepts. Then:

i) F is satisfiable w.r.t. T if $\dot{F} \wedge \bigwedge_{C \sqsubseteq D \in T} [U](\neg \dot{C} \vee \dot{D})$ satisfiable

ii) T |= E⊑F *if* $\bigwedge_{C \sqsubseteq D \in T} [U](\neg \dot{C} \vee \dot{D}) \wedge \dot{E} \wedge \neg \dot{F}$ este unsatisfiable

Similar with TBoxes, neither ABoxes don't have a direct correspondence in Modal Logic, but they can be seen as a special case of a modal constructor, which is *nominals*. These are special propositional variables that exist in exactly one world, and are the foundational elements of Hybrid Logic and they come with a special modality, the @ operator, that allows to refer to the only world inside which the nominal takes place. For example, $@_a\psi$ takes place if in the world in which a holds ψ holds as well.Thus, an ABox assertion of the form *a:C* is equivalent to the modal formula $@_a \dot{C}$, while the assertion *(a,b):r* corresponds to $@_a(r)b$. In the latter formulae, we can see that nominals can be both parameters of the @ operator, like *a*, and propositional variables, like *b.* Worth mentioning that the use of individual names inside Aboxes corresponds to formulas in which nominals are being used in a more restrained fashion. Some DLs, like $\mathcal{SHOIN}$ or $\mathcal{SROIQ}$, allow for more general use of nominals, which is indicated by the letter O in DL's name [28].

As it is the case with FOL, some DL constructors have correspondents in Modal Logic while others don't. Number restrictions correspond to gradual modalities, which have known a limited attention until the moment the connection with DLs was found. In certain variants of Propositional Dynamic Logic (PDL), a modal logic for reasoning with programs, we find deterministic programs which correspond to unqualified number restrictions $\leq 1R.\mathrm{T}$. Similarly we find inverse programs which correspond to role inverses, and regular program expressions, that are equivalent to the roles constructed with transitive-reflexive closure, union, and composition [28].

2.2.5. Most Common DL Languages

As I have previously stated, Description Logics are a logic-based family of formalisms used for representation of the knowledge of an application domain and that have model theoretic semantics. They are used especially in applications that requires the storing of knowledge and performing various reasoning operations on them. Rely on a family of common languages, called *descriptors,* which have a set of constructors for creating descriptions of concepts (classes) and roles (proprieties). These descriptions are used in axioms and assertions of knowledge bases and can be reasoned by special DL systems.

Let's take an example of a concept called: "A man married to a professor and having at least 5 children all doctors". This can be expressed in DL by the following concept:

$Human \cap \neg Female \cap \exists married.Professor \cap (> 5\ hasChild) \cap (\forall hasChild.Student)$

This description contains the conjunction ($\cap$) and negaţie ($\neg$) boolean operators, as well as others such as:

- existential restriction ($\exists R.C$)
- value restriction ($\forall R.C$)
- number restriction (> n R)

The main role that descriptive languages have is that of describing the entities from the application domain (concepts, roles, constraints). As it can be thought, a concept wants to represent a class of objects that share some common characteristics while a role is a relationship that links an object to another object or to a data value. A descriptive language is constituted from an alphabet of names of concepts, roles, individuals, and a set of constructors for each of these[25].

Table 1 presents some of the most common DL languages that have been created in the domain literature by renowned scholars, like Baader, Horrocks, Donini, et al. In the works read by me during my Ph.D. studies.

Name	**Explanation**
$\mathcal{AL}$	a DL that has concepts: atomic, universal, bottom, negation, intersection, value restriction, limited existential restriction
$\mathcal{ALC}$	$\mathcal{AL}$ extended with total concept negation ($\mathcal{C}$)
$\mathcal{S}$	$\mathcal{ALC}$ extended with transitive role axioms ($\mathcal{R}_+$)
$\mathcal{SI}$	$\mathcal{S}$ extended with inverse roles ($\mathcal{I}$)
$\mathcal{SH}$	$\mathcal{S}$ extended with role hierarchies ($\mathcal{H}$)
$\mathcal{SHI}$	$\mathcal{SH}$ extended with role inverses ($\mathcal{I}$)
$\mathcal{SHF}$	$\mathcal{SH}$ extended functional role axioms ($\mathcal{F}$)
$\mathcal{SHQ}$	$\mathcal{SH}$ extended with qualified number restrictions ($\mathcal{Q}$)
$\mathcal{SHIQ}$	$\mathcal{SHQ}$ extended with role inverses ($\mathcal{I}$)
$\mathcal{SHIF}$	$\mathcal{SHI}$ extended with functional role axioms ($\mathcal{F}$)
$\mathcal{SHOIN}$	$\mathcal{SHI}$ extended with nominals ($\mathcal{O}$) and number restrictions ($\mathcal{N}$)
$\mathcal{SROIQ}$	$\mathcal{S}$ extended with nominals ($\mathcal{O}$), qualified number restrictions ($\mathcal{Q}$), role inverses ($\mathcal{I}$), role inclusion axioms and disjunction ($\mathcal{R}$)

Table 2.2: Main DL languages existing in literature

Next, I will try to make a series of discussions and analyses around those languages with respect to their characteristics, what can be expressed (described) with their constructors, inherent growth in complexity due to the addition of the new constructors, and many other interesting facts. I'll begin the discussion with the most basic of them, the DL $\mathcal{ALC}$.

The DL $\mathcal{ALC}$ (acr. Attributive Language with concept Complements) has been proposed by the scholars Schmidt-Schaus and Smolka in year 1991 [174], there where has been created also a scheme for the names of the languages: starting from a primitive DL, the addition of a constructor is indicated by a letter in the name of the DL; for example, $\mathcal{ALC}$ is obtained from $\mathcal{AL}$ by adding the complement operator ($\neg$) and $\mathcal{ALE}$ by adding existential restrictions ($\exists R.C$).

Particularly, $\mathcal{ALC}$ has been extended with several characteristics that have a large impact in the creation of ontology languages, such are the qualified number restrictions, role inverses, transitive roles, subroles, concrete domains, nominals.

With number restrictions is possible to specify the number of relations of a certain type in which individuals can participate. For example, we may want to state that a person may be married with at most one individual, or to extend the definition of the previous concept *HappyMan* in order to state that it has between 2 and 4 children:

$$Person \subseteq \leq 1 \text{ married}$$

$$HappyMan \subseteq \text{Human} \cap \neg\text{Female} \cap \exists\text{married}.\text{Doctor} \cap \forall\text{hasChild}.(\text{Doctor} \cup \text{Professor}) \cap \geq 2\ \text{hasChild} \cap \leq 4\ hasChild$$

By means of qualified number restrictions can be additionally specified the types of individuals that are being counted by that restriction. For example, we can further extend the definition of HappyMan in order to say that it has at least 2 children that are doctors:

$$HappyMan \subseteq \text{Human} \cap \neg\text{Female} \cap \exists\text{married}.\text{Doctor} \cap \forall\text{hasChild}.(\text{Doctor} \cup \text{Professor}) \cap (\geq 2\ \text{hasChild}.\text{Doctor}) \cap \leq 4\ hasChild$$

Using inverses of roles, transitive roles and subroles we can, besides *hasChild*, also use its inverse, *hasParent*, to say that *has Ancestor* is transitive and that *hasParent* is a subrole of *hasAncestor*.

Concrete domains integrate DLs with concrete sets, such are these of real numbers, integers, strings, or with concrete predicates defined on sets, like number comparisons ($\leq$), constants (≤ 7), strings (*isPrefixOf)* etc. Unfortunately, in their unrestricted form, concrete domains can have dramatic effects on the decidability of reasoning in DL. Due to that, in practice, it is being used a restricted form of these, called *data types*.

The nominal constructor allows the use of individuals inside concept descriptions. If *a* is an individual name then it is called "*nominal*", which is interpreted as the singleton set.

The *one-of* constructor extends nominals to a finite set of individuals and can be specified with the disjunction:

$$\{a_1,a_2,\ldots a_n\}=\{a_1\}\cap\{a_2\}\ldots.\cap\{a_n\}$$

Nominals can have dramatic effects over the complexity of reasoning.

The letter "$\mathcal{S}$" is used for the abbreviation of $\mathcal{ALC}$ extended with transitive role axioms ($\mathcal{R}+$) in order not to perplex too much the language names when new features are being added. Its name comes from the modal logic S4, which is in a certain manner related to DLs [154].

2.2.6. Application Domains

DLs have applications in a vast array of domains, such as that of databases (schema design, data integration with schema, answering to queries), communications equipment configurations, software systems for information and documentation etc [28].

Maybe the most important application, and one that is in trend with the current research, is the one on the Semantic Web for creation and development of ontology languages. As I have already stated in previous sections, high-quality ontologies play a crucial role on the Semantic Web, and their construction, integration, evolution depends on the availability of some good semantics and powerful reasoning tools. Since DLs have both, this makes them ideal candidates for the realization of ontology languages. This fact was known for 30 from now, but back then existed a major dissonance between the capabilities of one and the requirements of the other[110]. Due to the research from the last decade in the DL domain this gap has become sufficiently narrow in order to allow to construct stable bridges. This association allows to exploit of more than 20 years of research in DLs, such as decidability and complexity of the important reasoning problems (satisfiability, inclusion) and to use existing reasoner systems, such as FaCT++, Pellet, Racer, HermiT to provide reasoning services for the applications. An ontology could be seen as corresponding to a DL knowledge base, consisting of sets of concepts, roles, and individuals. Also similar to DL, ontology classes can be simple names or expressions constructed from the atomic classes and properties using a set of constructors provided by that language.

BUPT

Constructor DAML	Sintaxa DL	Exemplu
intersectionOf	$C_1 \cap ... \cap C_n$	Teen∩Girl
unionOf	$C_1 \cup ... \cup C_n$	Professor∪Doctor
complementOf	¬C	¬Positive
oneOf	$\{x_1,...x_n\}$	{Tom, Anne}
allValuesFrom	∀P.C	∀hasColor.White
someValuesFrom	∃r.C	∃hasColor.Blue
hasValue	∃r.{x}	∃oneOf.{Christians}
minCardinality	≥ n r.C	≥2 hasMark.Good
maxCardinality	≤ n r.C	≤1 hasMark.Insufficient
inverseOf	r^-	$hasChild^-$

Table 2.3: DAML+OIL Constructors and equivalent DL syntax

Every ontology language relies on a description logic. For example $\mathcal{SHIQ}$ extended with concrete domains corresponds to the ontology language DAML+OIL. $\mathcal{SHIQ}$ is an expressive DL that, unlike others that are focused especially on concept constructors, it allows also to create complex roles [111]. Table 1 shows the DAML+OIL set of constructors together with the equivalent DL syntax.

OWL language has at its foundation also Description Logics. OWL Lite is based on the DL $\mathcal{SHIF}$ extended with nominals and concrete domains, $\mathcal{SHIF}(\mathcal{D})$. OWL DL, as even its name suggests, relies on a descriptive logic, namely $\mathcal{SHOIN}(\mathcal{D})$. The third version is OWL Full, which doesn't have correspondent any logic since it's undecidable due to its big power of expressivity and contains "paradoxical" features. If the sets of mathematical constructors of OWL DL and Full are practically the same, the latter doesn't impose restrictions on the actual use of these constructors, such as allowance of deterministic closure, unsatisfiability of global restrictions through axioms (proprieties hierarchies, extensions of simple object proprieties), the definition of classes that are in the same manner properties and have themselves as instances, etc. [112]. Next, I will try to make a broader talk about the OWL language and cover also its second, more recent version, OWL2.

a) OWL

OWL is the standard language for developing ontologies on the Semantic Web that has been created by the W3C's WOW Group and whose semantics can be defined by means of a translation to an expressive DL, this being one of the objectives of its design. This association allows OWL to take advantage of the rese in DL made during the last decades, such are those about decidability and complexity of the key inference problems, the existing DL reasoner systems (FaCT++, Pellet, Racer) in order toarch provide reasoning services to the applications.

An OWL ontology is seen as corresponding to a TBox from DL together with a roles hierarchy, that describes the application domain in terms of classes (coresp. concepts) and properties (coresp. roles). An ontology consists from a set of axioms that assert certain things, for example inclusion relations between classes and properties. Similarly as into a standard DL, OWL classes can be simple names or expressions built from atomic ones using a large array of constructors. The set of constructors supported by OWL together with the equivalent DL syntax is presented in Table 2, while in Table 3 is shown the set of axioms supported by OWL.

OWL Constructor	**DL Syntax**	**Example**
intersectionOf	$C_1 \cap \ldots \cap C_n$	Human∩Male
unionOf	$C_1 \cup \ldots \cup C_n$	Doctor∪Lawyer
complementOf	$\neg C$	¬Male
oneOf	$\{x_1\} \cup \ldots \cup \{x_n\}$	{John}∪{Mary}
allValuesFrom	$\forall P.C$	∀hasChild.Doctor
someValuesFrom	$\exists r.C$	∃hasChild.Lawyer
hasValue	$\exists r.\{x\}$	∃cityOf.{France}
inverseOf	r^-	$hasChild^-$
minCardinality	$(\geq n)$	(≥1 hasChild)
maxCardinality	$(\leq n)$	(≤1 hasParent)

Table 2.4: OWL language constructors

OWL Axiom	**DL Syntax**	**Example**
subClassOf	$C_1 \subseteq C_2$	Human⊆ Animal∩Biped
equivalentClass	$C_1 \equiv C_2$	Man≡Human∩Male
subPropertyOf	$P_1 \subseteq P_2$	hasDaughter ⊆ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost ≡price
disjointWith	$C_1 \subseteq \neg C_2$	Male ⊆ ¬ Female
sameAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush}≡{G_Bush}
differentFrom	$\{x_1\} \subseteq \neg\{x_2\}$	{john}⊆ ¬{peter}
TransitiveProperty	P transitive role	has Ancestor is transitive role
FunctionalProperty	$\top \subseteq (\leq 1\ P)$	$\top \subseteq (\leq 1\ hasMother)$
SymmetricProperty	$P \equiv P^-$	isSiblingOf ≡ $isSiblingOf^-$

Table 2.5: OWL language axioms

The complete XML serialization of OWL is not shown because it is very prolix. It is easy to see that, except for individuals and data types, the OWL constructors and axioms can be translated to $\mathcal{SHIQ}$. As I have stated in the previous section, OWL Lite is equivalent to $\mathcal{SHIN}(\mathcal{D})$ while OWL DL to $\mathcal{SHOIN}(\mathcal{D})$.

OWL ontology language has not been designed in a void; a big number of influences existed on it, some of them imposed by the WOW Group of W3C. Since it is also an effort from the development activity of the Semantic Web, it should have fit into its stack of languages, together with XML and RDF. Since there already existed a few general-purpose ontology languages on the traditional Web, OWL must have kept as much as possible compatibility with those ones, including SHOE, OIL and DAML+OIL. One of the biggest influences that laid marks on the development of OWL came from its predecessor, DAML+OIL, from the Description Logics, the Frames paradigm and, nevertheless from RDF. Particularly the formal specification of the language was influenced by DLs, the shallow structure by Frames paradigm, while the RDF/XML interchangeable syntax by the forward compatibility with the RDF language.

Despite its inherent success, the OWL language could not satisfy the requirements of all users. After intense talks among users, theorists and implementers it was decided to be addressed some of these problems by means of an incremental revision of the language, called OWL 1.1. The initial scope of this new version was to exploit recent research works from DL in order to address some of the expressivity limitations of this one. As the design of the new version made progresses, it has been decided to address also the performance requirements by exploiting research in smaller DLs that have desired computational properties [114].

b) OWL2

Because OWL relies intensively on Description Logics, OWL2 too shares many of the typical characteristics of DLs. In particular, it describes the domain in terms of individuals, classes (DL concepts), property (DL roles), data types and values (DL concrete domains). Individual names (ex. "John") refers to elements of the domain, concepts (e.g. 'university') describe individual sets sharing some common traits, roles (e.g. 'studiesAt') describe relations between pairs of individuals, data types (ex. 'integer') describe sets of data values. Class descriptions can be formed of all the previously stated elements by making use of a variety of constructors.

Similar with a DL knowledge base, an OWL2 ontology consists of a set of axioms (facts) that describe the application domain. For example, to assert that Absolvent is a *subclassOf* Student, that John *isA* student, or that John *hasAge* 18. Finally, also just like a descriptive logic, OWL2 can be seen as a fragment of FOL and it was given some formal semantics based on the first-order model theory, even though it could be well given by a translation to DL, or even FOL. [114].

Manchester Syntax	**DL Syntax**
Class: A subClassOf: C Class: A equivalentTo: C EquivalentClasses: $C_1,\ldots C_n$ DisjointClasses: $C_1,\ldots C_n$ Class: A disjointUnionOf: $C_1,..C_n$	$A\subseteq C$ $A\equiv C$ $C_i\equiv C_{i+1}$, $1\leq i<n$ $C_i\subseteq \neg C_j$, $1\leq i<j\leq n$ $A\equiv C_1 \cup \ldots \cup C_n$
ObjectProperty: P subPropertyOf: R ObjectProperty: P equivalentTo: R EquivalentProperties: $R_1,\ldots R_n$ DisjointProperties: $R_1,\ldots R_n$ ObjectProperty: P inverseOf: R ObjectProperty: P domain: C ObjectProperty: P range: C ObjectProperty: P characteristics: Functional ObjectProperty: P characteristics: InverseFunctional ObjectProperty: P characteristics: Reflexive ObjectProperty: P characteristics: Ireflexive ObjectProperty: P characteristics: Symmetric ObjectProperty: P characteristics: Assymetric ObjectProperty: P characteristics: Transitive	$P \subseteq R$ $A\equiv C$ $R_i\equiv R_{i+1}$, $1\leq i<n$ $R_i\subseteq \neg R_j$, $1\leq i<j\leq n$ $P\equiv R^-$ $\exists P.T \subseteq C$ $T \subseteq \forall P.C$ $T \subseteq \leq 1\ P$ $T \subseteq \leq 1\ P^-$ $T \subseteq \exists P.self$ $\exists P.self \subseteq \bot$ $P\equiv P^-$ $P \circ P \subseteq P$
SameIndividual: $i_1,\ldots i_n$ DifferentIndividuals: $i_1,\ldots i_n$ Individual: i Types: C Individual: i_1 Facts: P i_2 Individual: i_1 Facts: not P i_2 Individual: i_1 Facts: T v Individual: i Facts: not T v	$i_i = i_{i+1}$, $1\leq i<n$ $i_i \neq i_j$, $1\leq i<j\leq n$ i:C (i_1,i_2):P i_1: $(\neg\exists P.\{i_2\})$ (i,v):T i: $(\neg\exists P.\{v\})$

Table 2.6: Classes, properties and individual axioms of OWL2

Because OWL is an ontology language for the Semantic Web, it exhibits some particularities other than the majority of description logics and makes certain things differently than these. These distinctive elements start with the names used in OWL2, which are IRIs and that are the basis for the names on the Semantic Web. Because IRIs tend to be too large, the syntax of OWL2 offers facilities for shorthands for names, relatively similar to Qnames from SPARQL. OWL2 heavily relies on the facilities for datatypes found in XML Schema, like floating-point numbers in place of mathematical types common in most of DLs. The set of supported data types and facets (constraints) are defined in the OWL2 datatype mapping.

OWL2 has many syntaxes. The standard syntax of the Semantic Web, RDF/XML, is the only one that all OWL2 implementations must support [35]. Because the RDF/XML syntax though is a very prolix and hard to read other syntaxes has been created for OWL2, including an XML one for the integration with XML tools, a functional style one that is being used for precision and in formal documents (http://www.w3.org/TR/owl2-syntax/), and an easily-readable one created in the purpose of presentation at humans, called the Manchester syntax (http://www.w3.org/TR/owl2-manchester-syntax/). As I previously affirmed, OWL2 has at its basis the description logic $\mathcal{SROIQ}(\mathcal{D})$ and provides a large variety of operators for the construction of the more complex classes and properties expression. One part of these constructors are presented in Table 6, the Manchester syntax together with the equivalent DL one.

Similarly with Description Logics, OWL2 has first-order model theoretic semantics, called Directs Semantics (http://www.w3.org/TR/owl2-direct-semantics/). These semantics are basically equivalent to the translation of the ontology into a $\mathcal{SROIQ}(\mathcal{D})$ knowledge base and then apply the standard DL semantics. These model theoretic semantics represent the last free will of the significance of OWL2 constructors. Generally it suffices to understand the informal significance as it was described earlier and as in the OWL2 used guides, such as the Primer (http://www.w3.org/TR/owl2-primer/).

In order for OWL2 to remain decidable it is necessary to impose some global restrictions over the structures of ontologies. These restrictions correspond with the ones that are being used in the same purpose at the definition of SROIQ(D) knowledge bases. They are called global because depend on the ontology as an entire unity, and not just of one singleton expression or axiom. For example, some of the restrictions regard the properties hierarchy which depends on the set of property axioms that occur in the imports closure of the ontology. I will recall here a few of the most important global restrictions [114]:

- simple properties distinguishing: a property is simple if its existence doesn't depend on any other one
- the structure of properties chains is generally restricted in order to satisfy an acyclicity condition that is necessary in order to ensure the decidability of the language
- restrictions imposed on the datatypes axioms and values ranges; particularly must satisfy the uniqueness and aciclicity conditions
- the use of anonymous individuals inside axioms is constrained
- IRIs used for naming entities and ontologies in OWL2 must not be from the reserved vocabulary (used by language itself).

Ontology language		DL language
OIL		SHIQ –translated semantics Frames Paradigm
DAML+OIL		SHIQ(D)
OWL	OWL Lite	SHIF(D)
	OWL DL	SHOIN(D)
	OWL Full	- (undecidable)
OWL2	OWL2 EL	εL^{++}
	OWL2 QL	DL Lite
	OWL2 RL	DLP

Table 2.7: Ontology languages and corresponding DL formalisms

2.3. Inference and Reasoning Tasks in DL Knowledge Bases

2.3.1. Introduction

In this section I will continue my investigation into Description Logics, this time the focus will be put on the inference problems from the DL knowledge bases, the reasoning services created in order to solve them (in the form of decision procedures), and the practical reasoner systems that implement these techniques inside them. I will talk then the complexity results of these problems as they are for a few particular DLs, as they have been found by the scientists in the domain, like Baader, Horrocks, Calvanese, Sattler, etc. Also will be made a series of analyzes and comparisons regarding how the addition of new constructors to a language influence the complexity of reasoning (growth in expressivity) starting with the most basic DL, $\mathcal{ALC}$. Will be shown what combinations of constructors are most 'harmless' (don't change the complexity of reasoning), and which generate real 'explosions' in complexity. Will presented the results for the important DLs and enumerated the other ones, and references in literature will be provided where further information related to the domain could be found. A state-of-art with the most recent research in DL inference and reasoning will be made in which will be presented some of the most important articles read for the creation of my research.

The most primitive form of inference on the concepts expressions is *subsumption*, generally stated in the form $C \sqsubseteq D$. Subsumption determination is the problem of verifying if the concept denoted by D (subsumer) is considered more general than the one denoted with C (subsumed). In other words, subsumption tests if the first concept always represents a subset of the second concept. For example, we may be interested in knowing whether $Woman \sqsubseteq Mother$. In order to verify this type of relationship must be taken into account the relations defined inside the terminology. As will be explained during the following sections, under corresponding restrictions we can embed this sort of knowledge directly into the concept expressions, thus making subsumption over concept expressions the standard inference task in DL.

Another form of typical inference over the concept expressions is *satisfiability*, which is the problem of testing whether a concept expression does not denote the void concept. Actually, concept satisfiability is a particular case of subsumption in which the subsumer is the void concept, which means that a concept is not satisfiable.

Even though the meaning of concepts has been already specified using some logical semantics, the creation of DL decision procedures has been influenced for a long time by the Semantic Networks, where concepts were seen as nodes and roles as links in the graph. Subsumption between concept expressions was recognized as the key inference task, and the fundamental idea of the subsumption algorithms from the early stages was to transform two input concepts into labeled graphs and to verify if one can be contained by the other. This method is called the *structural comparison* and the relationship between the concepts being compared *structural subsumption*. A careful analysis of the structural subsumption algorithms yielded that they are safe but not always complete in terms of logical semantics: anytime they yield the answer "Yes" the answer is the correct one, but when yield "No" the answer can be incorrect. In conclusion, structural subsumption is generally weaker than the logical one [28].

The need for complete subsumption algorithms has been motivated by the fact that for the usage of knowledge representation systems it is often needed to have a guarantee that the system did not fail in the verification of subsumption. Due to this cause new algorithms have been developed that do not use a network-style of representation anymore, and these can be shown to be complete. These algorithms have been developed by specializing the settings for deductive reasoning to the DL subsets of First-Order Logic (FOL), as it was made for the Tableaux calculus by Schmidt-Schauβ and Smolka [174] in 1991, and also for further specialized methods.

In the work "Tractability of Subsumption in Frame-based Description Languages" by Brachman&Levesque [49] from 1984 it was affirmed that there exists a compromise between the power of expressivity of a language and the difficulty of reasoning over the representations created using it. In other words, as the language is more expressive the reasoning is much harder to achieve. They showed an example of this compromise by making an analysis of $\mathcal{FL}^-$ (Frame Language), which contains concept intersection, value restrictions and a simple form of existential quantification. They proved that for such a language the subsumption problem can be solved in polynomial time, and the addition of a constructor for role restrictions makes subsumption a coNP-hard problem (the extended language has been called $\mathcal{FL}$). This paper introduced at least 2 new ideas:

i) the efficiency of reasoning over the knowledge structures can be studied using the tools of the computational complexity theory

ii) different combinations of constructors could lead to languages with different computational properties

An immediate consequence of the previous observations is that we can formally and methodically study this compromise between the power of expressivity of the language and the complexity of reasoning, which is at its turn defined in terms of the allowed constructors. After the initial paper, a number of results of this compromise for concept languages have been yielded, and these results allow us to make a pretty clear picture of complexity of reasoning for a wide array of concept languages. Moreover, the problem of finding the optimal compromise, that is the

BUPT

most expressive extensions of $\mathcal{FL}^-$ with resp. to a set of constructors that still keeps subsumption in polynomial time has been extensively studied by Buchheit&Donini [57].

Some of the assumptions that lie at the basis of this research line are those to use the complexity in its worst case as a measure of the efficiency of reasoning in DL, or, more generally, into the representation formalisms. Such an assumption has been thus criticized as not characterizing correctly the performance of systems or take into consideration more general case behaviors. If this observation suggests that computational complexity alone can not be sufficient for addressing most of the performance problems, research into the computational complexity of DLs definitely led to a deeper understanding of the problems that arise in the implementation of the reasoner systems. Next will be presented some of the contributions brought to this area.

Foremost, the study of computational complexity of reasoning in Description Logics led to a clearer understanding of the properties of language constructors together with their interaction. This fact doesn't yield value just from a theoretical point of view but also gives insights to the creator of deduction procedures with indications regarding the constructors of that language and their combinations which are hard to work with, as well as the methods to solve them.

Secondly, the complexity results have been obtained by means of a general technique for the satisfiability checking into concept languages, that relies on a form of tableaux calculus [174]. This technique proved itself extremely useful for studying both the correctness and complexity of algorithms. More precisely, it provides a parametrical algorithmic framework with respect to the language constructors. The algorithms for the concept satisfiability and subsumption obtained this way also led directly to the practical implementation by application of a series of intelligent control strategies and optimization techniques. The most recent knowledge representation systems for DLs adopt the tableaux calculus, as it is stated by [114].

Thirdly, the analysis of pathological cases from this framework led to the discovery of incompletenesses in the algorithms that had been developed for the systems being implemented. This thing proved to be useful in defining the test sets for checking the implementations. For example, comparisons of the implemented systems largely benefited from the results of complexity analysis.

After the compromise between the expressivity and tractability of reasoning has been sufficiently analyzed and the applicability range of the inference techniques has been experimented with, it existed a change of focus in the theoretical research of reasoning in DL. Interest grew in relating DLs to the modeling languages that were being used in databases management. Besides these, the discovery of relations with expressive Modal Logics stimulated the study of so-called "very expressive" DLs. These languages, besides the fact that contain very general mechanisms for concepts definition (e.g. cyclic definitions), provide also a rich set of constructors to form complex concepts and roles expressions. For such languages, the power of expressivity is big enough that the new challenge became the enrich of the language while in the same time keeping the decidability of reasoning. Worth mentioning fact that this new direction in theoretical research was accompanied by a modification in the implementation of KR systems that relied on very expressive DLs.

BUPT

In sub-section two I will try to make a short presentation of the reasoning techniques for very expressive DLs [25].

2.3.2. Inference Tasks

Inference tasks from DL knowledge bases lay into two main categories:

a) standard
b) non-standard

In continuation of this sub-section I will try to discuss about each category in part and to provide references in the literature where more insights can be found.

A) Standard Tasks

Description Logics have distinguished logical properties. They put emphasis on the decidability of the key reasoning problems, like satisfiability and subsumption of concepts or the knowledge base, provide reasoning solutions that are decidable, such are the tableaux algorithms that deduce implicit knowledge from the explicitly stated ones. Highly optimized DL reasoners, such as FaCT++, Pellet, HermiT, Racer proved that the tableaux algorithms for highly expressive DLs lead to a good performance even over large knowledge bases. Reasoning over a knowledge base is the process of deducing implicit knowledge from the ones that are explicitly stated.

Inference problems can be divided into two categories:

a) general: represents the checking of the truth value of a sentence
b) complex: are built from the general ones

Let $\mathcal{L}$ be a Description Logic, $\mathcal{K}$ a knowledge base, C and D two concepts, and *a* an individual.

General inference problems, as it was affirmed by [154], are the following:

1) Satisfiability of the knowledge base:
   - $\mathcal{K}$ is satisfiable if it has a model (is non-contradictory)
2) Concept satisfiability:
   - a concept C is satisfiable w.r.t. $\mathcal{K}$ if there exists a model I of it s.t. $C^I \neq \phi$
3) Concept subsumption:
   - $C \sqsubseteq D$ w.r.t. $\mathcal{K}$ (written $\mathcal{K} \models C \sqsubseteq D$) if in any model I of $\mathcal{K}$ we have $C^I \sqsubseteq D^I$ (all instances of C are also instances of D)
4) Concept equivalence:
   - $C \equiv D$ w.r.t. $\mathcal{K}$ (written $\mathcal{K} \models C \equiv D$) if they include themselves one another w.r.t. $\mathcal{K}$ ($\mathcal{K} \models C \sqsubseteq D$ si $\mathcal{K} \models D \sqsubseteq C$)
5) Instance checking:
   - an individual *a* is an instance of a concept C w.r.t. $\mathcal{K}$ (written $\mathcal{K} \models a:C$) if in any model I of $\mathcal{K}$ we have $a^I \in C^I$ (is an element of the interpretation of C)

BUPT

- two individuals *(a,b)* are an instance of a role *r* w.r.t. $\mathcal{K}$ (written $\mathcal{K}$ |= (a,b):r) if in any model I of $\mathcal{K}$ holds (a^I,b^I)∈r^I (is an element of the interpretation of r)

In the case of a DL that provides all boolean operators (such as $\mathcal{ALC}$), all the above-stated problems are resoluble to the satisfiability of the knowledge base [170]. For example:

$(T, A) \mid = a{:}\, C \; iff \; (T, A \cup \{a{:}\, \neg C\})$ is inconsistent.

These problems can be transformed into reasoning w.r.t. a TBox T, that means reasoning w.r.t. knowledge base (T, ϕ). This is known under the name of *terminological reasoning*. An important property of DLs says that reasoning with TBox is not influenced by ABox, which means that satisfiability w.r.t. (T,A) coincides with satisfiability w.r.t. T with condition that A is consistent (has a model) [111]. Complex inference problems are: classification, realization and extraction [154].
Classification is the problem of putting a new concept in the corresponding place into a hierarchy of concepts. This is done by verification of inclusion between every single concept of the hierarchy and the new concept.
Extraction (i.e. query answering) is the problem of finding a set of individuals that are instances of a certain concept description (possibly complex).
Realization is the process when, given a knowledge base (T,A), we want to check the consistency of A w.r.t. T and then compute the most specific concepts that instantiate a certain individual *i*.

In order to provide a reasonable and predictive behavior of the DL system these inference problems should be at least decidable and preferably of low complexity. For that, the expressive power of the DL must be limited in a benefic manner, but also not too restricted so the important notions of the domain to not being possible to be captured. This compromise between the power of expressivity and the complexity of reasoning has been one of the major problems in DL research. The investigations include both theoretical research, which represents determining of the complexity in the most unfavorable case of the reasoning problems, but also practical, i.e. developing systems and techniques for the optimization and empirical evaluation of the behaviors when are being applied to real-world tests and applications. If an application requires an expressive power greater than one that could be provided by a decidable DL then that DL is being introduced into an application program or other KR formalism instead of using an undecidable DL [25].

B) Non-Standard Tasks

Non-standard inference tasks could serve a variety of purposes, among which worth mention support in the construction and maintenance of knowledge bases, or getting insights about the knowledge represented within them. Among the most well-known non-standard inference tasks in DL worth mentioning: computing of the least common subsumer and the most specific concept, unifying/matching, concept rewriting [28]. Below I will try to make a brief presentation of what each of them means.

Least common subsumer (LCS) of a set of concepts is the minimal concept that includes all of them.

The minimality condition implies that it exists no other to include all concepts from the set and to be less general (subsumed) than the *LCS*. The notion was studied for the first time by Cohen [65] and has subsequently been used for several tasks: inductive learning from examples for concept descriptions, vivification of knowledge bases (as a means to represent disjunction inside languages that don't allow it) as well as in the bottom-up construction of DL knowledge bases (starting from the concept instances). The notion of *LCS* is tightly connected to the one of *most specific concept* (MCS) of an individual, i.e. the simplest concept description for which the individual is an instance given the assertions from the knowledge base. The minimality condition is explained in the same way as for *LCS.* More generally, an *MSC* of a set of assertions of individuals can be defined as the *LCS* of the *MSC* associated to each individual. Based on the *MSC* calculus of a set of individuals assertions we can incrementally build a knowledge base [26]. Worth mentioning the fact that the techniques created for computing *LCS* and *MSC* rely on the compact representation of concept expressions, which are built either by using the structural subsumption method or by defining of a correct normal form.

Another tool to support the construction and maintenance of DL knowledge bases that go beyond the DL standard inference services is *concept unification*. This task, as it is stated in [29], is an operation that can be seen as the process of loosening the equivalence between two concept expressions. More precisely, two concept expressions are unified if it can be found a substitution of variables inside concept expressions such that the result is two equivalent concepts. Intuition is that, in order to be able to find possible overlappings among concept definitions we can treat certain concept names as variables and discover by means of unification that two concepts, possibly independently defined by the different knowledge designers, are actually equivalent. Consequently, the knowledge base can be simplified by introduction of a unique definition for the unified concepts.

As usually, *matching* is defined as a special case of unification where variables occur in only one of the two concept expressions. In addition, in the DL framework we can define unification and matching based on subsumption, instead of equivalence [28]. In a similar fashion as with the other non-standard inferences, the calculus of unification and matching uses some sort of special representations for expressions of concepts and it has been proved to be decidable for small DLs. Finally there was an extensive work laid over the problem of concept rewriting. Given a concept represented into a source language, concept rewriting means to find a concept, possibly represented into another target language, that is related to the initial concept by equivalence, subsumption, or any other relation. To be able to represent rewriting must be provided a set of constraints between the concepts from the source language as well as the target. Rewriting can be applied to the translation of concepts from a knowledge base into another or the reformulation of concepts during the process of construction and maintenance of the logical base. Besides, concept rewriting has been addressed in the context of queries rewriting by using views, in databases management, and recently in the framework of information integration. Within this environment can be applied techniques for concept rewriting in order to automatically generate the queries that allow a system to gather information from a set of sources. Given an initial specification of the interrogation according to a global common language and a set of constraints that express the relation between the global schema and the individual information sources, the problem that arises is to find out the interrogations that will be placed to the local sources, that offer answers (possibly approximated) at the original interrogation [60].

2.3.3. Reasoning Algorithms

In this section will be discussed the reasoning techniques for the problems that had been presented in the previous section.
The precondition for obtaining some efficient reasoning algorithms is to be decision procedures [197], i.e.:

- safe: every positive answer is correct
- complete: every negative answer is correct
- terminate: always is provided an answer

These techniques are implemented into practical reasoning systems, for example FaCT, FaCT++, Pellet, Racer, HermiT, KAON2, Hoolet, OWLim etc. There are two main techniques to create a reasoning algorithm: to reuse existing algorithms from FOL or to create new ones for DL [25].

The first method, since the main part of DLs are contained within the two variables fragment of FOL, we can reduce the DL problems to others known of inference (e.g. $\mathcal{L}^2$, $\mathcal{C}^2$). The cost is that the complexity of decision procedures obtained this way is greater than normally.

For the second method, in the beginning have been employed the so-called *structural subsumption algorithms*. These algorithms compare the syntactical structure of concepts in order to solve the inclusion problem. The disadvantage is that they are efficient only for primitive DLs, like $\mathcal{AL}$, $\mathcal{ALC}$, and are not complete for those that contain total negation and disjunction, such are the ones from $\mathcal{S}$ family. For these languages, that have a greater expressiveness, the most efficient have been proven to be the tableaux algorithms [25].

Reasoning in the $\mathcal{EL}$ family, which lies at the basis of the profile with the same name as the ontology language OWL2 (i.e. OWL2 EL), is realized by means of the completion algorithms, or consequence-oriented. OWL2 RL relies on the rule-based algorithms and OWL2 QL query rewriting techniques in order to create robust decision procedures. These 3 algorithms are efficient especially on less expressive logics, such are those of the corresponding DL profiles [114].

Low complexity decision procedures could be obtained by exploiting the connection between DL and Propositional Modal Logic. Schild [171] was the first one to remark that the language $\mathcal{ALC}$ is a syntactic variant of the multi-modal propositional logic **K** and that its extension with transitive closure of roles corresponds to the Propozitional Dynamic Logic (PDL) [28]. In particular, few of the algorithms used with Modal Logic for deciding the satisfiability are very similar to the tableaux algorithms created for DLs. This link between DLs and Modal Logic has been exploited to transfer decidability results from the latter to DLs. Instead of using tableaux algorithms decidability of certain Modal Logics (and thus the corresponding DL) can be proved by establishing the finite model property of the logic, that is to show that a concept/formula is satisfiable if it is so in a finite interpretation or by applying the tree automata.

In the previous section I stated that all the main inference problems can be reduced to the consistency of ABoxes provided that DL has conjunction and negation constructors. Descriptive languages of almost all the early systems as well as few of today's did not offer negation. For such DLs concept subsumption can be computed in general by means of the structural subsumption algorithms, which are algorithms that compare the syntactic structure of concept descriptions (possibly normalized).

Even though are in general very efficient, the disadvantage is that they are complete only for rather simple languages, with small power of expressiveness. In particular, DLs with full negation and disjunction cannot be dealt with by these algorithms. For such languages the so-called tableaux algorithms proved to be the most efficient. Into the field of Description Logics the first tableaux-based algorithm has been created by Schmidt-Schauβ & Smolka in 1991 [174] for deciding the satisfiability of $\mathcal{ALC}$ concepts. Starting then this technique has been used to obtain safe and complete satisfiability algorithms for a large variety of DLs derived from $\mathcal{ALC}$. For example, Hollunder, Baader, Donini & Sattler showed for languages with number restrictions, for transitive closure of roles, transitive roles, concrete domains (such as numbers). In addition has been extended to the consistency problem for ABoxes and TBoxes that allow for sets of inclusion axioms together with many other features [57], [22]. Into a separate section I will present the way a tableaux algorithm works for deciding the satisfiability of $\mathcal{ALCN}$ concepts, then I will show how this can be extended to an algorithm for consistency of ABoxes, and finally I will explain how we can take into account also general subsumption axioms.

Instead of creating new algorithms for reasoning in DL, we can try to reduce the problem to one of known inference from the logiccs. For example, decidability of the inference problem for $\mathcal{ALC}$, as well as in many other DLs can be obtained as a consequence of the known decidability results from the two-ariable fragment of FOL. The $\mathcal{L}^2$ language consists of all FOL formulae that can be built using symbols for predicates and constants (apart from function ones) using just variables x and y. Decidability in $\mathcal{L}^2$ has been proven in [141]. It is easy to see that, by correcty reusing the variables names, any $\mathcal{ALC}$ concept description can be translated into an $\mathcal{L}^2$ formula with one free variable [47]. A direct translation of the concept description $\forall R.(\exists R.A)$ results the formulae $\forall y.(R(x,y) \rightarrow (\exists z.(R(y,z)\text{^}A(z))))$. Because sub-formula $\exists z.(R(y,z)\text{^}A(z))$ does not contain the x variable, it can be reused: renaming of the link variable z with x determines the equivalent formula: $\forall y.(R(x,y) \rightarrow (\exists x.(R(y,x)\text{^}A(x))))$, which it is easy to see that uses only two variables. This relation between $\mathcal{ALC}$ and $\mathcal{L}^2$ shows that any extension of $\mathcal{ALC}$ with constructors that can be expressed by means of only two variables yields a decidable DL. Number restrictions and role compositions are examples of constructors that cannot be expressed in $\mathcal{L}^2$. Number restrictions thus can still be expressed in $\mathcal{C}^2$, which is the extension of $\mathcal{L}^2$ by counting quantifiers, which has been shown to be decidable [89], [152]. Must mention thus that the complexity of decision procedures obtained this way is in general greater than normally. For example, satisfiability in $\mathcal{L}^2$ is a NExpTime-complete problem, while satisfiability of $\mathcal{ALC}$ concepts is only PSpace [28].

A) Structural Subsumption Algorithms

As I have previously stated during this chapter, early DL systems relied on the so-called *structural subsumption algorithms*. These algorithms work in two phases: first the concept descriptions that are being tested for subsumption are normalized, then the syntactical structures of the normal forms are compared.

The advantage was that they were able to decide subsumption in polynomial time. Some of the first complexity results for DLs showed that these algorithms were not polynomials, and moreover, not even decision procedures. For example, almost all of the early systems relied on unveiling of the concept definitions, fact that can cause an exponenţial explosion in the size of the concepts. The coNP-hardness result of Nebel [146] for subsumption with respect to definitorial TBoxes showed that this explosion cannot be avoided in presence of constructors for conjunction and values restrictions. Besides this, the early structural subsumption algorithms were not complete, that is they were not capable to detect all valid subsumption relations. These negative results, cumulated with the invention of tableaux algorithms for expressive DLs, which showed a good behavior in practice, have been the main reason for which the structural techniques (and altogether the search for DLs with polynomial subsumption problems) have been banished at the end of '90. More recent research on the complexity of reasoning in DLs featuring existential restrictions (instead of ones for value) led to a partial rehabilitation of the technique [27], [54].

When it is tried to find a DL with a polynomial subsumption problem it is certain that not all Boolean operations are allowed since in this case will be inherited NP-hardness from Propositional Logic. Also, it should be noted the fact that we cannot dispense of the conjunction since we should be capable to affirm that more than one property must hold when we define a concept. Finally, if we want to call the logic a DL will also be needed a constructor for roles. The following 2 languages are minimal DL candidates [28]:

- $\mathcal{FL}_o$: has as concept constructors conjunction, value restrictions ($\forall r.C$) and universal concept (T)
- $\varepsilon\mathcal{L}$: has concept constructors conjunction, existential restrictions ($\exists r.C$) and universal concept (T)

In the rest of this sub-section I will try to realize a study about the problem of subsumption in these two minimal DLs. Even though subsumption without a TBox was proved to be polynomial in both cases, εL showed a safer behavior for the complexity of subsumption in presence of TBoxes.

i. Subsumption in $\mathcal{FL}_0$

In this sub-section I will show the ideas behind this technique for the primitive DL $\mathcal{FL}_0$, which features the conjunction and values restrictions operators, then I will show how the concept *bottom* ($\bot$), atomic negation ($\neg A$) and number restrictions ($\leq n\ R, \geq n\ R$) are being handled. Obviously, $\mathcal{FL}_0$ and its extension with the bottom concept and atomic negation are sub-languages of $\mathcal{AL}$, while the addition of number restrictions to the resulted language yields the DL $\mathcal{ALN}$.

Foremost will be considered the case of concept subsumption without a TBox. There are two methods for creating a structural subsumption algorithm in this case that relies on different normal forms. One uses the equivalence $\forall r.(C \cap D) \equiv \forall r.C \cap \forall r.D$ as a rewriting rule from left to right or vice-versa. Here I will consider left-to-right, but all the early structural subsumption algorithms relied on normal forms obtained by rewriting in the opposite direction.

By using the rewrite rule $\forall r.(C \cap D) \rightarrow \forall r.C \cap \forall r.D$ together with the associativity, comutativity and idempotence of ∩ constructor any $\mathcal{FL}_0$ concept can be transformed into an equivalent one which is a conjunction of concepts of the form $\forall r1.A, \ldots rm.A$, for m role names $r_1,\ldots r_m$ $(m \geq 0)$ and a concept name A. Abbreviate $\forall r_1. \ldots \forall r_m.A$ with $\forall r_1. \ldots r_m.A$, where $r_1,\ldots r_m$ is considered a word over the alphabet of role names. In addition, instead of $\forall w1.A \cap \ldots \cap \forall wl.A$ we will write $\forall L.A$, where $L \coloneqq \{w1, \ldots wl\}$ is a finite set of words over Σ. Term $\forall \phi.A$ is considered the equivalent of universal concept, which means that it can be added to a conjunction without affecting the meaning of concept. By means of these abbreviations, any pair of $\mathcal{FL}_0$ concepts, C and D, containing the concept names $A_1,\ldots A_k$ can be written as following:

$C \equiv \forall U_1.A_1 \cap \ldots \cap \forall U_k.A_k$, si $D \equiv \forall V_1.A_1 \cap \ldots \cap \forall V_k.A_k$

, where U_i,V_i are finite sets of words over the alphabet of role names. This normal form provides us the following characterization of subsumption of $\mathcal{FL}_0$ concepts, as it was affirmed in [28]:

$C \subseteq D$ iff $Ui \supseteq Vi,\ for\ any\ i, 1 \leq i \leq k.$

As the dimension of normal form is polynomial in the size of original concepts and subsumption tests $Ui \supseteq Vi$ can also be performed in polynomial time, this yields a decision procedure in polynomial time for subsumption in $\mathcal{FL}_0$.

This characterization of subsumption by means of that for finite sets of words can be extended to definitorial Tboxes as follows: a TBox T can be translated into a finite automaton of words $\mathcal{A}_T$ whose states are concept names from T and transitions are inducted by value restrictions from T. In figure 2 is presented an example of such automaton. A definition of this translation can be found in [22], where it is considered the more general case of cyclic TBox. In the case of definitorial TBoxes, which are by definition acyclic, the resulted automata are also acyclic.

$$\begin{aligned} A &\equiv C \sqcap \forall r.B \sqcap \forall s.\forall r.P \\ B &\equiv \forall s.C \\ C &\equiv \forall r.P \end{aligned}$$

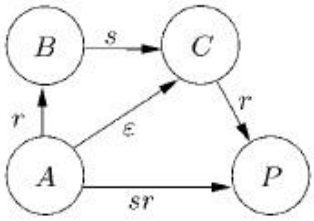

Fig.2.2: A $\mathcal{FL}_0$ definitorial TBox and the corresponding automaton

Let's call a concept name a *defined concept* in a definitorial Tbox if it occurs on the left-hand side of a concept definition, and a *primitive concept* otherwise. For a defined concept A and a primitive one T the language $L_{AT}(A,P)$ is the set of all paths of labeling the words of A_T from A to P. Languages $L_{AT}(A,P)$ represent all values restrictions that must be satisfied by instances of the concept A. With this intuition in mind it should not surprise us that subsumption w.r.t. $\mathcal{FL}_0$ definitorial TBoxes could be characterized in terms of that of languages accepted by acyclic automata [28].

Next is a characterization of subsumption in $\mathcal{FL}_0$ w.r.t. definitorial TBoxes:

$A \subseteq_{\mathrm{T}} B$ iff $L_{AT}(A,P) \supseteq \mathrm{L}_{\mathrm{T}}(B,P)$,

for all primitive concepts P. In example from fig. 2 we have:

$L_{AT}(A,P) = \{r, sr, rsr\} \supseteq \{\mathrm{sr}\} = L_{AT}(B,P)$, so $A \subseteq_{\mathrm{T}} B$ but $B \not\subset_{\mathrm{T}} A$.

How the subsumption problem for languages accepted by the finite acyclic automata is coNP-complete [86], this reduction shows that the problem of subsumption in $\mathcal{FL}_0$ w.r.t definitorial TBoxes is coNP. As it was stated by [146], this reduction works also in the opposite direction, which yields the inferior matching boundary. In the presence of general TBoxes subsumption problem in $\mathcal{FL}_0$ becomes same complicated as for $\mathcal{ALC}$, that is ExpTime-hard [27].

Din cele de mai sus putem construi următoarea teoremă.

Theorem 7: Subsumption in $\mathcal{FL}_0$ is polynomial without TBoxes, coNP-complete w.r.t. definitorial TBoxes and ExpTime-complete with general TBoxes.

If we extend $\mathcal{FL}_0$ by constructors that are able to express unsatisfiable concepts then must, on one side, to change the definition of the normal form, while on the other, the structural comparison of the normal forms must take into account the fact that an unsatisfiable concept is being subsumed by any other concept. The simplest DL in which this thing happens is $\mathcal{FL}_\bot$, that is the extension of $\mathcal{FL}_0$ by the concept bottom [28].

An $\mathcal{FL}_\bot$ concept description is in normal form if it has the following form:

$$A_1 \cap \ldots A_m \cap \forall R_1.C_1 \cap \ldots \forall R_n.C_n$$

, where $A_1,\ldots A_m$ are distinct concept names (others than $\bot$), $R_1,\ldots R_n$ are distinct role names and $C_1,\ldots C_n$ are $\mathcal{FL}_\bot$ concept descriptions in normal form. This kind of normal form can be easily computed, basically it is calculated only the $\mathcal{FL}_0$ normal form of the description (where $\bot$ is treated as an ordinary concept) :

$$B_1 \cap \ldots B_k \cap \forall R_1.D_1 \cap \ldots \forall R_n.D_n.$$

If any of the B_i terms is the bottom concept then the entire description is replaced with the concept $\bot$, otherwise same procedure is applied recursively for D_j terms. For example, the $\mathcal{FL}_0$ normal form of the concept description: $\forall R.\forall R.B \cap A \cap \forall R.(A \cap \forall R.\bot)$ is $A \cap \forall R.(A \cap \forall R.(B \cap \bot))$, from which results the $\mathcal{FL}_\bot$ normal form: $A \cap \forall R.(A \cap \forall R.\bot))$.

The structural subsumption algorithm for $\mathcal{FL}_\bot$ works in the same way as for $\mathcal{FL}_0$, with the only difference that $\bot$ is subsumed by any other concept description. For example:

$$\forall R.\forall R.B \cap A \cap \forall R.(A \cap \forall R.\bot) \subseteq \forall R.\forall R.A \cap A \cap \forall R.A$$

because the recursive comparisons of their normal forms $\mathcal{L}_\bot$:

$$A \cap \forall R.(A \cap \forall R.\bot) \text{ and } A \cap \forall R.(A \cap \forall R.\mathrm{A})$$

finally leads to the comparison of A with the bottom concept.

The extension of $\mathcal{FL}_\bot$ with atomic negation (i.e. negation that is applied only to concept names) can be treated in a similar fashion. During the process of computing the normal form, the negated concept names are being handled the same way as the normal ones. But if a name and its negation occur on the same level as the normal form then the $\bot$ concept will be added, which can be dealt with in the way it was previously shown.

BUPT

For example, $\forall R.\neg A \cap A \cap \forall R.(A \cap \forall R.B)$ is foremost transformed into $A \cap \forall R.(A \cap \neg A \cap \forall R.B)$, then $A \cap \forall R.(\bot \cap A \cap \neg A \cap \forall R.B)$, and finally $A \cap \forall R.\bot$. Structural comparison of the normal forms treats negated concepts in the same way as normal ones.

If we consider the language $\mathcal{ALN}$, the additional presence of number restrictions leads to a new type of clash. On one side, similarly as in the case of atomic negation, number restrictions may clash one to another (e.g. $\geq 2\ R,\ \leq 1\ R$). On the other side, *at-least* $\geq n\ R$ restrictions, for $n \geq 1$, are conflicting with value restrictions $\forall R.\bot$ that forbid role succesors. When the normal form is being calculated we can again treat number restrictions as concept names then deal with the new conflicts by introduction of $\bot$ and use it for normalization, as it has been previously shown. During the process of structural comparison of the normal forms must also be considered the inherent subsumption relations between the number restrictions (ex. $\geq n\ R\ \subseteq \geq m\ R\ daca\ n \geq m$). A broader presentation of a structural subsumption algorithm that works on a data structure of graph type for a language derived from $\mathcal{ALN}$ is made in [47].

ii. Subsumption in $\varepsilon\mathcal{L}$

In contrast with the negative complexity results of subsumption with respect to $\mathcal{FL}_0$ TBoxes, subsumption in $\varepsilon\mathcal{L}$ remains polynomial even in the presence of general TBoxes, as it was stated in [54]. The polynomial-time subsumption algorithm for $\varepsilon\mathcal{L}$ that will be discussed below classifies a TBox T, that is calculated simultaneously all subsumpiton relations among the concepts from T. The algorithm works in 4 steps:

a) TBox normalization
b) translation of the obtained TBox into a graph
c) completes the graph using the completion rules
d) read subsumption relations from the normalized graph

A general $\varepsilon\mathcal{L}$ Tbox is normalized if contains only CGIs of the form:

$$A_1 \cap A_2 \subseteq \mathrm{B},\ \mathrm{A} \subseteq \exists \mathrm{r}.\mathrm{B},\ \text{sau}\ \exists \mathrm{r}.\mathrm{A} \subseteq \mathrm{B},$$

, where A, A_1, A_2, B are concept names or universal (top) concept. A given Tbox can be transformed into a normalized one by application of the normalization rules. Instead of describing these rules into the general case I will illustrate them here by an example in which I will underline the CGIs that need further rewrites:

$$\exists r.A \cap \exists r.\exists s.A \subseteq \mathrm{A} \cap \mathrm{B} \ \rightarrow\ \exists r.A \subseteq \mathrm{B}_1, \mathrm{B}_1 \cap \exists \mathrm{r}.\exists \mathrm{s}.\mathrm{A} \subseteq \mathrm{A} \cap \mathrm{B},$$
$$B1 \cap \exists r.\exists s.A \subseteq \mathrm{A} \cap \mathrm{B} \ \rightarrow\ \exists r.\exists s.A \subseteq \mathrm{B}_2, \mathrm{B}_1 \cap \mathrm{B}_2 \subseteq \mathrm{A} \cap \mathrm{B},$$
$$\exists r.\exists s.A \subseteq \mathrm{B}_2 \ \rightarrow\ \exists \mathrm{s}.\mathrm{A} \subseteq \mathrm{B}_3, \exists \mathrm{r}.\mathrm{B}_3 \subseteq \mathrm{B}_2,$$
$$B_1 \cap B_2 \subseteq \mathrm{A} \cap \mathrm{B} \rightarrow \mathrm{B}_1 \cap \mathrm{B}_2 \subseteq \mathrm{A}, \mathrm{B}_1 \cap \mathrm{B}_2 \subseteq \mathrm{B}.$$

For example, in the first normalization phase I will introduce the abbreviation B_1 for the description $\exists \mathrm{r}.\mathrm{A}$. It can be said that B_1 needs to be made equivalent to $\exists \mathrm{r}.\mathrm{A}$, that is add the CGI $B_1 \subseteq \exists \mathrm{r}.\mathrm{A}$. It can be shown that adding just $\exists r.A \subseteq B_1$ it is enough to obtain a TBox that is subsuming equivalent, that is a TBox that induces the same relations between concept names that occur in the original. All normalization rules preserve the equivalence in this sense and if one uses an according strategy, which generally delays the application of the rule last applied in my example, then the normal form can be calculated in linear time.

In the next step the classification graph is being constructed: G_T=(V, VxV, S,R), in which:

- V is the set of concept names that occur in the normalized TBox T (incl. the universal concept)
- S labels nodes with sets of concept names
- R labels edges with sets of role names

It can be shown that the sets of labels satisfy the invariants:

- $B \in S(A)$ implies $A \subseteq_T B$, i.e. S(A) contain just the subsumers of A with resp. to T
- $r \in R(A,B)$ implies $A \subseteq_T \exists r.B$, i.e. R(A,B) contain only roles r such that $\exists r.B$ includes A with resp. to T

Initially set S(A):={A,T} for all nodes $A \in V$, and R(A,B):= ϕ for all edges $(A,B) \in VxV$. Evidently the above invariants are satisfied by these initial sets of labels.

(R1)	$A1 \cap A2 \subseteq B \in T$ and $A1, A2 \in S(A)$ *then* add B to S(A)
(R2)	$A1 \subseteq \exists r.B \in T$ and $A1 \in S(A)$ *then* add r to R(A,B)
(R3)	$\exists r.B1 \subseteq A1 \in T$ and $B1 \in S(B), r \in R(A,B)$ *then* add A1 to S(A)

Table 2.8: Completion rules for subsumption in εL w.r.t. general TBoxes

Labels of the nodes and edges are extended by application of the rules in Table 1, where we assume that a rule is applicable only if it extends a set of labels. It is easy to see that these rules preserve the above stated invariants. For example, let's consider the most complicated rule, (R3). Obviously, $\exists r.B_1 \subseteq A_1 \in T$ implies $\exists r.B_1 \subseteq_T A_1$ and the assumption that invariants are satisfied before the application of the rule yields $B \subseteq_T B_1$ and $A \subseteq_T \exists r.B$. The subsumption relation $B \subseteq_T B_1$, at its turn implies $\exists r.B \subseteq_T \exists r.B_1$. After applying the transitivity of the subsumption relation $\subseteq_T$ obtain: $A \subseteq_T A_1$ [28].

The fact that subsumption in $\varepsilon\mathcal{L}$ w.r.t. general TBoxes can be decided in polynomial time is an immediate consequence of the statements:

a) Rules applications ends after a number of polynomial steps
b) If no rule is applicable anymore then $A \subseteq_T B$ if $B \in S(A)$

In what concerns the first statement should be noticed that the number of nodes is linear while the one of edges quadratic in the size of T. In addition, the dimension of the labels sets is bounded by the number of concepts and roles names and each application of a rule extends at least one label. Regarding the equivalapplicationthe second statement, the direction "if" follows from the fact that the invariants preserve under the rules aplication. In order to prove the "only-if" track let's assume that $B \notin S(A)$, then the following interpretation is I, a model of T in which $A \in A^I$ but $A \notin A^I$:

- $\Delta^I := V$
- $r^I := \{(A', B') | r \in R(A', B')\}$, $for\ all\ roles$
- $B'^I := \{(A' | B' \in S(A'))|\}$, $for\ all\ concepts\ A'$

More insights about this could be found in literature in [54],[27].

Theorem 8: Subsumption in $\varepsilon\mathcal{L}$ is polynomial with resp. to general TBoxes.

In [27] this result is extended to $\varepsilon\mathcal{L}^{++}$, which is an extension of $\varepsilon\mathcal{L}$ with nominals, universal concept, a restricted form of concrete domains and a form of role-value mappings. In addition, also here it is shown that all other constructor additions to $\varepsilon\mathcal{L}$ make subsumption w.r.t. general TBoxes a problem ExpTime-complete.

Worth mentioning the fact that these results do not present interest only from a theoretical perspective. In fact, large bio-medical ontologies, such as Genes Ontology [192] or SNOMED [187] can be represenated in $\varepsilon\mathcal{L}$, and a first implementation of the subsumption algorithm for $\varepsilon\mathcal{L}$ presented earlier acts very good over the large knowledge bases.

For larger DLs structural subsumption algorithms generally are not complete. Particularly, they cannot handle disjunctions, complete negations and complete existential restrictions. For languages that include such features the Tableaux technique for the construction of subsumption algorithms proved to be the most efficient [25].

B) Tableaux Algorithms

Instead of directly testing the subsumption of concept descriptions these algorithms use negation in order to reduce subsumption to satisfiability of concepts [28], and that is:

$C \subseteq \mathrm{D}$ iff $\mathrm{C} \cap \neg \mathrm{D}$ unsatisfiable

Let A,B two concept names, R a role name. Let's assume that we want to know whether $(\exists R.A) \cap (\exists R.B)$ is subsumed by $\exists R.(A \cap B)$. This means that we must verify the description:

$$C = (\exists R.A) \cap (\exists R.B) \cap \neg(\exists R.(A \cap B))$$

for unsatisfiability. Foremost we push all negation signs as far as possible inside the description using the DeMorgan rules and the ones for quantifiers. As result is obtained the description:

$$C_0 = (\exists R.A) \cap (\exists R.B) \cap \forall R.(\neg A \cup \neg B))$$

which is in normal negated form, i.e. negation occurs only in front of concept names. Then I will try to construct a finite interpretation I i.e. $C_0^I \neq \phi$. This means that there must be an individual in Δ^{I} that is an element of C_0^I. The algorithm generates such an individual, say *b*, and imposes restriction $b \in C_0^I$. Since C_0 is a conjunction of three concept descriptions this means that *b* must satisfy three constraints: $b \in (\exists R.A)^I, b \in (\exists R.B)^I, and\ b \in \left(\forall R.(\neg A \cup \neg B)\right)^I$. From the first one we can deduce that there must exist an individual *c* such that $(b,c) \in R^I and\ c \in A^I$. Analogously, $b \in (\exists R.B)^I$ implies the existence of an individual *d* , $(b,d) \in R^I\ and\ d \in B^I$. In this situation we must not assume that c=d because this would impose too many constraints on the newly introduced individuals in order to satisfy the existential restrictions of *b*.

BUPT

$\cap -\boldsymbol{Rule}$: **if** A contains $(C_1 \cap C_2)(x)$ but not both $C_1(x)$ and $C_2(x)$ **then** $A' = A \cup \{C_1(x), C_2(x)\}$
$\cup -\boldsymbol{Rule}$: **if** A contains $(C_1 \cup C_2)(x)$ but neither $C_1(x)$ nor $C_2(x)$ **then** $A' = A \cup \{C_1(x)\}$, $A'' = A \cup \{C_2(x)\}$
$\exists -\boldsymbol{Rule}$: **if** A contains $(\exists R.C)(x)$ but there is no individual *z* s.t. C(z) and R(x,z) are in A **then** $A' = A \cup \{C(y), R(x,y)\}$, with *y* an individual name not in A
$\forall -\boldsymbol{Rule}$: **if** A contains $(\forall R.C)(x)$ and $R(x,y)$ but not $C(y)$ **then** $A' = A \cup \{C(y)\}$
$\geq -\boldsymbol{Rule}$: **if** A contains $(\geq n\, R)(x)$ and doesn't exist individuals $z_1,\ldots z_n$ s.t. $R(x,z_i)$ and $z_i \doteq z_j$ in A **then** $A' = A \cup \{R(x,y_i),\ 1 \leq i \leq n\} \cup \{y_i \doteq y_j, 1 \leq i < j \leq n\}$, where $y_1,\ldots y_n$ are individual names not in A
$\leq -\boldsymbol{Rule}$: **if** A contains individuals $y_1,\ldots y_{n+1}$ s.t. $(\leq n\, R)(x)\ and\ R(x,y_1),\ldots R(x,y_{n+1})$ are in A but $y_i \doteq y_j$ not in A , for $i \neq j$ **then** for any pair y_i, y_j , with $i > j, and\ y_i \neq y_j$ not in A , ABox $A_{ij} = [y_i/y_j]A$ is obtained from A by replacing each occurrence of y_i cu y_i

Table 2.9: Transformation rules of the satisfiability algorithm

Since:

- for any existenţial restriction the algorithm introduces a new individual as a role filler (r-filler) and this individual must satisfy the constraints expressed by that restriction. Since *b* must satisfy also the value restriction $\forall R.(\neg A \cup \neg B)$, and *c,d* had been introduced as r-fillers of *b*, we obtain the additional constraints: $c \in (\neg A \cup \neg B)^I\ and\ d \in (\neg A \cup \neg B)^I$.

and:

- the algorithm uses value restrictions in interaction with role relations in order to impose new constraints on the individuals. Now $c \in (\neg A \cup \neg B)^I$ means that $c \in (\neg A)^I\ or\ c \in (\neg B)^I$, and must choose one of these variants. If choose $c \in (\neg A)^I$, this will be in contradiction with the other one, $c \in (A)^I$, which means that this path of the search lead to a clash. Thus we must choose $c \in (\neg B)^I$. Analogously must choose $d \in (\neg A)^I$ in order to satisfy constraint $d \in (\neg A \cup \neg B)^I$ without creating a clash with $d \in (B)^I$.

thus:

- for disjunctive constraints the algorithm tests both alternatives in successive attempts; it backtracks if is being arrived to a contradiction, i.e. if the same individual satisfies restrictions that are in obvious conflict

In my example all constraints have been satisfied without arriving to an obvious clash. This means that concept C_0 is satisfiable, and thus $(\exists R.A) \cap (\exists R.B)$ is not subsumed by $\exists R.(A \cap B)$. The algorithm generated an interpretation I following this process:

BUPT

Δ^I = {b,c,d}; R^I={(b,c),(b,d)}; A^I={c} si B^I ={d}.

For this interpretation $b \in C_0^I$, which means that $b \in ((\exists R.A) \cap (\exists R.B))^I$ but $b \notin \left(\exists R.(A \cap B)\right)^I$

In the second example I will add a number restriction to the first concept from the above example, i.e. is wanted to know whether $(\exists R.A) \cap (\exists R.B) \cap \leq 1R$ subsumed by $\exists R.(A \cap B)$. Intuitively, the answer should be YES since $\leq 1R$ from the first concept insures that A's r-filler coincides with B's , thus exists an r-filler in $A \cap B$.

The Tableaux algorithm for satisfiability proceeds as stated above with the only difference that there is in plus the constraint $b \in (\leq 1\ R)^I$. In order to satisfy this constraint the two r-fillers of *b*, i.e. *c* and *d* must be identified one with another. So, if an *at-most* number restriction is violated then the algorithm should identify different role fillers.

In the example, the individual c=d must belong to both A^I and B^I , that, cumulated with $c = d \in (\neg A \cup \neg B)^I$ leads to a clash. Thus the search for a counter-example for the subsumption fails and the algorithm deduces that $(\exists R.A) \cap (\exists R.B) \cap \leq 1\ R \subseteq \exists \mathrm{R}.(\mathrm{A} \cap \mathrm{B})$.

Before starting to describe the algorithm more formally there must be introduced a corresponding data structure in which it should be able to represent the following type of constraint: "*a* belongs to interpretation of C", or "*b* is an R-filler of *a*". The original paper of Schmidt-Schauβ & Smolka [174], as well as many other works about tableaux algorithms for DLs introduce the notion of 'system of constraints' in this purpose. But if we look closer at the types of constraints that need to be expressed we observe that they can be represented as ABox assertions.

As we saw in the second example above, the presence of '*at-most'* number restrictions may lead to identification of different individual names. Due to that we will not impose the unique name assumption (UNA) over the ABoxes considered by the algorithm. In exchange will be allowed explicit inequality assertions of the form: $x \doteq y$ for individual names x,y, with the semantics that an interpretation I satisfies $x \doteq y$ if $x^I \neq y^I$. These assumptions are supposed to be symmetrical, i.e. saying that $x \doteq y$ belongs to an ABox $\mathcal{A}$ is identical to $y \doteq x$ belongs to $\mathcal{A}$.

Let C_o be an $\mathcal{ALCN}$ concept in normal negated form. In order to test its satisfiability the algorithm starts with the ABoxl $\mathcal{A}_0$= {$C_0(x_0)$} and applies the transformation rules over the ABox until no one could ever be applied. If the complete ABox obtained this way does not contain any obvious contradiction (clash) then $\mathcal{A}_0$ is consistent and the concept C_0 is satisfiable, in the other case it is unsatisfiable [28].

The transformation rules that deal with disjunction and '*at-most'* type of restrictions are non-deterministic, in sense that a given ABox is transformed into finitely many new Aboxes such that the original one is consistent if at least one of the new Aboxes are consistent. Due to this will be considered finite sets of ABoxes S={$\mathcal{A}_1$,...$\mathcal{A}_k$} instead of singleton ones. This type of set is consistent if it exists an i, $1 \leq i \leq k$ such that $\mathcal{A}_i$ is consistent.

A transformation rule, as it is shown in table 9, is applied to a finite set of ABoxes, $\mathcal{S}$ as following: gets an element $\mathcal{A}$ from $\mathcal{S}$ and replaces it with an ABox $\mathcal{A}'$, two ABoxes $\mathcal{A}'$ and $\mathcal{A}''$, or with a finite number of ABoxes, $\mathcal{A}_{ij}$ [28].

BUPT

The following proposition is an immediate consequence of the definition of transformation rules.

Proposition 1 (Safety): Let's assume that $\mathcal{S}'$ is obtained from the finite set of ABoxes $\mathcal{S}$ by application of a transformation rule. Then $\mathcal{S}$ is consistent if $\mathcal{S}'$ is.

The second important property of the transformation rules is that the process always terminates.

Proposition 2 (Termination): Let C_0 be an $\mathcal{ALCN}$ concept description in normal negated form. There cannot exist an infinite sequence of rules applications:

$\{\{C_0(x_0)\}\} \rightarrow S_1 \rightarrow S_2 \rightarrow \cdots$.

In what follows I will try to present a short demonstration of this theorem, as it has been made in [28]

Lema 1: Let $\mathcal{A}$ be an ABox contained in $\mathcal{S}_i$, for an $i \geq 1$.

- For any individual $x \neq x_0$ of $\mathcal{A}$ there exists a unique sequence $R_1,\ldots R_l$, with $l \geq 1$, of role names and a unique sequence $x_1,\ldots x_{l-1}$ of individual names s.t. $\{R_1(x_0,x_1), R_2(x_1,x_2),\ldots R_l(x_{l-1},x)\} \subseteq \mathcal{A}$. In this case we say that 'x occurs on level l in $\mathcal{A}$'.
- if $(x) \in A$, for an individual x from level l, then the maximal depth of roles of C (that is the maximal nesting of roles constructor) is bounded by the maximal roles depth of C_0 - l. As such, the level of any individual from $\mathcal{A}$ is bounded by the maximal roles depth of C_0.
- if $C(x) \in A$ then C is a sub-description of C_0. In consequence, the number of different concept assertions over x is bounded by the size of C_0.
- the number of different role succesors of x in $\mathcal{A}$ (i.e. individuals y such that $R(x,y) \in A$, for a role name R) is bounded by the sum of numbers that occur in the '*at-least*' restrictions from C_0 + the number of existential restrictions from C_0.

Starting from $\{\{C_0(x_0)\}\}$ is obtained, after a finite number of rules applications, a set of ABoxes $\hat{S}$ in which no more rules could be applied. An ABox $\mathcal{A}$ is complete if no transformation rules can be applied onto it. The consistency of a set of complete Aboxes could be decided by looking for clashes. An ABox $\mathcal{A}$ contains a clash if one of the following situations take place:

i) $\{\bot(x)\} \subseteq A$, for an individual x

ii) $\{A(x), \neg A(x)\} \subseteq A$, for an individual x and a concept A

iii) $\{(\leq n\, R)(x)\} \cup \{R(x,y_i),\ 1 \leq i \leq n+1\} \cup \{y_i \neq y_j | 1 \leq i < j \leq n+1\} \subseteq A$, for individuals $x,y_1,\ldots y_{n+1}$, a non-negative integer n and a role R

Obviously, an ABox that contain a clash is not consistent, which means that if all Aboxes from $\hat{S}$ contain a clash then $\hat{S}$ is inconsistent, thus, from the safety Lema (1) we have also that $\{C_0(x_0)\}$ is inconsistent, consequently C_0 is unsatisfiable.

BUPT

But if only one of the complete ABoxes from $\hat{S}$ is clash-free though, then $\hat{S}$ is consistent. By rule safety this implies the consistency of $\{C_0(x_0)\}$ and thus the satisfiability of C_0.

Proposition 4 (Completeness): Any complete and clash-free ABox $\mathcal{A}$ has a model.

This fact can be proved by defining the canonical interpretation I_A induced by $\mathcal{A}$:

i) domain Δ^{I_A} of I_A consists of all individuals from $\mathcal{A}$

ii) for all atomic concepts C we define $C^{I_A} = \{x | C(x) \in A\}$

iii) for all atomic roles R we define $R^{I_A} = \{(x, y) | R(x, y) \in A\}$

By definition, I_A satisfies all roles assertions from $\mathcal{A}$. By induction on the structure of concept descriptions it is easy to show that it satisfies also the concept assertions. The inequality assertions are satisfied since $x \doteq y \in A$ hold only if x,y are different individuals.

The statements made in Lema 1 imply that the canonical interpretation has the shape of a tree whose depth is linearly bounded by the size of the concept C_0 and the branching factor by the sum of numbers that occur in C_0's *at-least* restrictions + the number of existential restrictions from C_0. In conclusion, $\mathcal{ALCN}$ holds the *finite tree model property*, i.e. any satisfiable concept C_0 it is so in a finite interpretation I which has the shape of a tree rooted in C_0.

To sum up, we have seen during this last sub-section that transformation rules reduce $\mathcal{ALCN}$ concepts satisfiability that are in normal negated form to the consistency of a finite set of complete ABoxes. In addition, the consistency of the set can be decided by looking for obvious contradictions.

i. Complexity Considerations

The Tableaux algorithm for $\mathcal{ALCN}$ concept satisfiability presented earlier may be needing exponential execution time and space. Actually, the size of the canonical interpretation that is constructed by the algorithm may be exponential in that of the concept descriptions. For example, let's consider the below C_n concepts inductively defined as:

$C_1 = \exists R.A \cap \exists R.B$

. . . .

$C_1 = \exists R.A \cap \exists R.B \cap \forall R.C_n$

Evidently, the size of C_n grows linearly with n. Given the input description C_n, the earlier satisfiability algorithm generates a complete and clash-free ABox whose canonical model is a binary tree of depth n, so it contains 2^{n+1} -1 individuals.

Not lastly, this algorithm can be modified such that to be needing only polynomial space, the main reason is that different branches of the tree model generated by the algorithm may be investigated separately. How the NP-space class dovetails with P-space, as it was stated in [168], it is sufficient to describe a non-deterministic algorithm using only polynomial space, that is for each non-deterministic rule we may simply assume that the algorithm picks the right alternative.

BUPT

Basically, the modified algorithm works as follows [28]:

i) starts with $\{C_0(x_0)\}$

ii) applies →∪ and →∩ rules as much as possible and checks clashes of the form: $A(x_0), \neg A(x_0), \perp (x_0)$

iii) generates all direct successors of x_0 using the rules → ∃ and →≥

iv) generates the necessary identifications of these successors using the rule →≤ and verifies clashes caused by *at-most* restrictions

v) iteratively handles the successors in this manner

Since after identification the remained successors can be separately dealt with, the algorithm must store a single path of the tree model that will be generated together with the direct successors of individuals from this path and information regarding which of these successors are to be investigated next. We already know that the length of the path is linear with the size of the input concept C_0, so the only obstacle in the road to a Pspace algorithm is that the number of an individual's direct successors on the path also depends on the numbers found in *at-least* restrictions. If assume that these numbers will be written in the base 1 representation (where the size of the representation coincides with the number actually represented), this thing would not be a problem anymore. For bases larger than 1 though (e.g. numbers in decimal notation), the number being represented could be exponential in the size of the representation. For example, the representation of 10^n-1 requires 10 digits in decimal, so we are unable to introduce all successors required by *at-least* restrictions only in polynomial space with the size of concept descriptions if numbers from the description are written in decimal notation.

It proves though that the big majority of successors required by *at-least* restrictions are not necessary at all. If an individual *x* obtains at least one R-successor following the application of → ∃ rule, then the →≥ rule must not be applied to *x* for the role R'; otherwise we can simply introduce an R-successor representatively. In order to detect inconsistencies due to number restrictions conflicts we must add a new type of clash: $\{(\leq R)(x), (\geq m\, R)(x)\} \subseteq \mathrm{A}$ for non-negative integers n<m. The canonical interpretation obtained by the modified algorithm must not satisfy the *at-least* restrictions of C_0. It can be though easily modified to an interpretation that can do that by duplicating the R-successors, more precisely the entire sub-tree that starts from them [28].

Lemma 2: Satisfiability of $\mathcal{ALCN}$ concept descriptions is PSpace-complete. The above theorem states that the problem is in PSpace. The hardness result comes from the fact that satisfiability is PSpace-hard for the super-language $\mathcal{ALC}$, which can be proved by a reduction from the validity of Quantified Boolean Formulae[174]. Since satisfiability and subsumption of $\mathcal{ALCN}$ concepts may be reduced one to another in linear time this thing shows that $\mathcal{ALCN}$ concept subsumption is PSpace-complete.

ii. An extension to the consistency problem for ABoxes

The tableaux algorithm for satisfiability of $\mathcal{ALCN}$ concepts presented in section a) can be extended to one that decides $\mathcal{ALCN}$ Aboxes consistency. Let $\mathcal{A}$ be an $\mathcal{ALCN}$ ABox such that all its concepts are in normal negated form.

BUPT

To test the consistency of the ABox foremost we add inequality assertions $a \neq b$ for each pair of distinct individuals *a*,*b* that occur in $\mathcal{A}$. Let's call $\mathcal{A}_0$ the ABox obtained in this way. The consistency algorithm applies the transformation rules shown in table 9 over the singleton set $\{\mathcal{A}_0\}$.
Safety and completion of the rules set can be demonstrated as previously. Unfortunately, the algorithm is not required to terminate only in case when a specific strategy is imposed onto the order of rules application. For example, consider the ABox:

$$A_0 = \{R(a,a), (\exists R.A)(a), (\leq 1\ R)(a), (\forall R.\exists R.A)(a)\}$$

By applying the rule → ∃ to *a* we can introduce a new R-successor *x* of *a*:

$$A_1 = A_0 \cup \{R(a,x), A(x)\}$$

Rule → ∀ adds the assertion $(\exists R.A)(x)$, which at its turn triggers the application of the rule → ∃ to *x*, obtaining the ABox:

$$A_2 = A_1 \cup \{\exists R.A(x), R(x,y), A(y)\}$$

Since *a* has two R-successors in A_2 , the rule →≤ is applicable to *a*. Replacing each occurrence of *x* with *a* we obtain the ABox:

$$A_3 = A_0 \cup \{A(a), R(a,y), A(y)\}.$$

Except for the individual names and assertion A(a), the ABox $\mathcal{A}_3$ is identical to $\mathcal{A}_1$. Due to this reason, we can continue as above in order to obtain an infinite chain of rule applications. Termination can still be regained by putting the condition that the generative rules (i.e. →≥, → ∃) are applied if none of the others is. In the earlier example, this strategy would prevent application of the rule → ∃ to *x* in the ABox $A_1 \cup \{(\exists R.A)(x)\}$ because rule →≤ is also applicable. After application of rule →≤ , rule → ∃ is not applicable anymore due to the fact that *a* already has un R-successor that belongs to $\mathcal{A}$.

Using a similar idea it can be reduced the consistency problem for $\mathcal{ALCN}$ ABoxes to $\mathcal{ALCN}$ concept satisfiability [103]. Fundamentally, this reduction works in the following manner: in a preprocessing step the transformation rules are applied only on 'old' individuals (i.e. ones presented in the original ABox), then it can look for roles assertions, i.e. for every individual of the preprocessed ABox the algorithm is applied to the conjunction of its concept assertions [103].

iii. Extension to general inclusion axioms

In the previous three sub-sections I considered the satisfiability problem for concept descriptions and that of consistency for ABoxes without any base TBox. Actually, for acyclic Tboxes, definitions can be extended in a simple fashion. The extension is no longer possible if general subsumption axioms of the form $C \subseteq D$ are allowed, where C and D are concept descriptions (possibly complex). Instead of considering a finite number of such axioms $C_1 \subseteq D_1$, … $C_n \subseteq D_n$, it is sufficient to consider only one: $T \subseteq \hat{C}$, where

$$\hat{C} = (\neg C_1 \cup D_1) \cap \ldots (\neg C_n \cup D_n)$$

The axiom $T \subseteq \hat{C}$ says that any individual must belong to the concept $\hat{C}$. The tableaux algorithm presented earlier can be modified in order to take into account this axiom: all individuals, both the original ones and the ones generated by the →≥ and → ∃ rules are being asserted as pertaining to $\hat{C}$. This modification may lead to the non-termination of the algorithm though. For example, let's see what happens if

the algoritmul is applied for checking the consistency of the ABox:
$\mathcal{A}_0$=$\{A(x_0), (\exists R.A)(x_0)\}$ w.r.t. the axiom T ⊆ ∃R. A.

The algorithm generates an infinite sequence of ABoxes $\mathcal{A}_1$,$\mathcal{A}_2$,… and individuals x_1,x_2, … such that:

$A_{i+1} = A_i \cup \{R(x_i, x_{i+1}), A(x_{i+1}), (\exists R.A)(x_{i+1})\}$.

How all individuals x_i get the same concept assertions as x_0, we may make the statement that the algorithm came into a cycle. Termination can be regained by trying to detect these cyclic computations then attempt blocking the application of the generative rules: application of the rules →≥ and → ∃ over an individual *x* is blocked by an individual *y* from an ABox $\mathcal{A}$ if:

$\{D|D(x) \in A\} \subseteq \{D'|D'(y) \in A\}$.

The base idea behind the blocking technique is that the individual being blocked *x* may use the role successors of y instead of generating new ones[28]. For example, instead of generating a new R-successor for x_1 from the previous example we can simpler use the R-successor of x_0. This leads to an interpretation I in which:

$\Delta^I = \{x_0, x_1\}, A^I = \Delta^I, R^I = \{(x_0 x_1), (x_1, x_1)\}$.

Evidently, I is a model of $\mathcal{A}_0$ and of the axiom T ⊆ ∃R. A.

In order to avoid cyclic blocking (of *x* by *y* and vice-versa), consider an enumeration of individual names and state that an individual *x* may be blocked only by individuals *y* that are in front of it in the enumeration. This fact, together with other technical assumptions makes sure that an algorithm that uses this technique is safe and complete, as well as terminates [57], [22]. Thus, consistency of $\mathcal{ALCN}$ Aboxes with resp. to general subsumption axioms is decidable.

It should be noticed the fact that now the algorithm is not in PSpace anymore because it can generate role paths of exponential length before blocking arise. Actually, even for the simple DL $\mathcal{ALC}$, satisfiability with resp. to a single general subsumption axiom is known to be ExpTime-hard [172].

The tableaux algorithm presented above is one NExpTime-complete. Using the translation technique mentioned at the beginning of this section can be shown that $\mathcal{ALCN}$ ABoxes and GCIs can be translated into PDL for which satisfiability may be decided in exponential time. A tableaux algorithm for $\mathcal{ALC}$ with GCIs has been talked about broadly in [72].

iv. Extension to other language constructors

Tableaux technique for the construction of algorithms for concept satisfiability and Aboxes consistency may also be used for languages with other constructors (for concepts or roles). Fundamentally, each newly added constructor requires a new transformation rule, and this can be generally obtained by simply considering the semantics of that constructor. Safety property of this kind of rule is generally easy to prove; it is harder to do so with completion and termination since must be taken into account also the interactions between rules. As I have previously stated, termination, in some cases cannot be gained unless the rules application process is restrained by a certain strategy. This kind of strategy though may be imposed only if it is insured that it does not corrupt completion [28].

BUPT

C) Automata-based Algorithms

There exist different instances of the automata-based technique that differs with respect to the DL on which they are being used as well as the type of automaton employed. These techniques have in common the following ideas [25]:

- show that the DL in cause holds the *tree model property*
- make a translation from the pair C,T (with C a concept and T a TBox) into a tree automata A_C,T suct that it accepts tree models of C w.r.t. T
- applies the emptiness test for the automata model used at A_C,T in order to see if C has a tree model w.r.t. cu T

Complexity of the satisfiability algorithm obtained this way depends on that of translation and of the emptiness test, which, at its turn depends on the type of automata used.

A type of non-deterministic automata that works on infinite trees of fixed arity is the looping tree automata [201]. In this case, translation is exponential but the emptiness test is polynomial in the size of the exponential automaton obtained after translation. Thus, the entire algorithm runs in exponential deterministic time (ExpTime-complete). Can also be used an alternative tree automata, where there is possible a polynomial translation but the emptiness test is exponential [143].

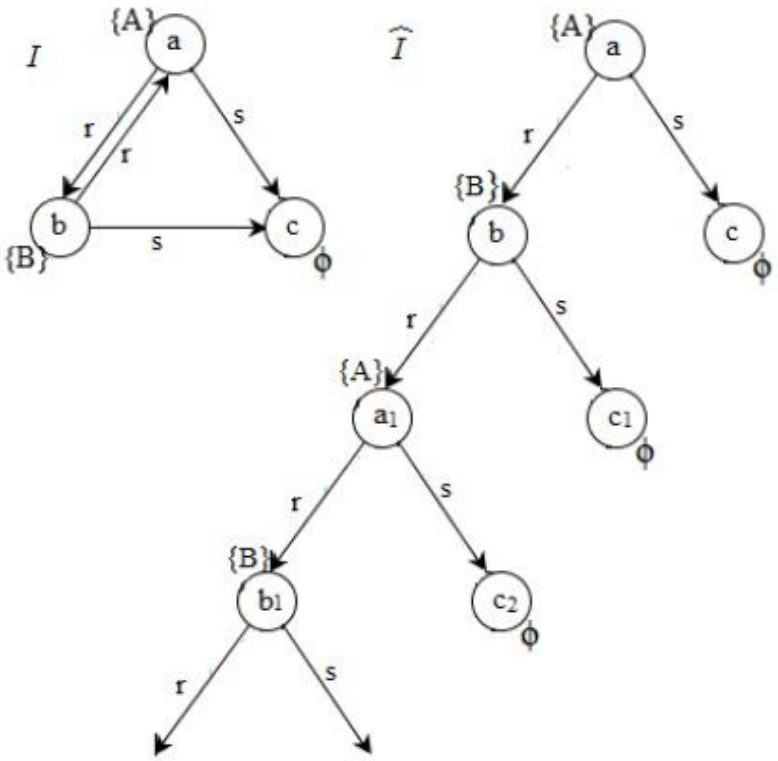

Fig.2.3: Unveiling of a model into a tree

BUPT

Another type are the amorph tree automata [45] that, instead of fixed arity trees can contain arbitrary branching. This fact simplifies translation but uses a more complicated automata model. For very expressive DLs (such are those that allow for transitive closures of roles), the looping tree automata introduced above is not sufficient because are required also some acceptance conditions (Buchi) that allow the occurrence of infinitely many states in any of the paths.

Tree model property of a language means that for every concept C that is satisfiable w.r.t. a TBox T, it has a model in shape of a tree [25]. The $\mathcal{ALC}$ holds this property. One way to show this is to apply the tableaux algorithm for $\mathcal{ALC}$ over the knowledge base (T,a:C) and to give up of blocking, then the execution of the algorithm generates the tree model. Another method is to use the technique called 'unveiling' in which an arbitrary model of C with resp. to T is developed into a bissimilar interpretation having the shape of a tree. This fact proves that satisfiability in $\mathcal{ALC}$ w.r.t. general Tboxes can be decided using the automata-based technique. Figure 3 shows an example of unveiling. The graph on the right side is an interpretation I: nodes are elements of Δ^I, nodes labels express to which concept name belongs while the labeled edges express the roles relations. For example: $a \in \Delta^I$ belongs to A^I, has an r-successor on *b* and s-successor on *c*. I is a model of the concept A w.r.t. the Tbox T={A⊆∃r.B, B⊆∃r.A, A⊆∃r.B, A∪B⊆∃s.T}.
The graph from the right side represents a portion of the unveiled model in which the start node is *a*. The unveiled corresponding interpretation I is a model of T and satisfies $a \in A^I$.

Given a looping tree automata A, the emptiness test decides if the accepted language is empty: $L(A)=\phi$. The problem can be solved in multiple ways, by making use of top-down and bottom-up techniques, that lead to some polynomial non-determinişti and deterministic algorithms, respectively. A broader discussion about automata-based techniques can be found in [28].

2.3.4. Some Results of the Complexity of Reasoning

In this section will be presented some complexity results for the problems of inference that had been discussed in section 2 for some DL languages, culled and selected from the domain literature in the realization of the current thesis. Among the scientists that investigated the complexity of reasoning in various DLs by attempting different combinations of constructors worth mentioning: Franz Baader, Ian Horrocks, Ulrike Sattler, Francesco Donini, Diego Calvanese, John Brachman, H.Levesque and many others, and in section 4 that will incheia this chapter I will discuss some of their masterpiece works.

I expect that the reader be familiar with the basic notions about the algorithms complexity, such as the time and space necessary for execution, classes of complexity (PSpace, ExpTime, NExpTime, coNP), what does it mean that a problem is undecidabile, the degree of membership and hardness of a class. For those who do not hold these basic knowledge are invited to see the book of Papadimitriou called "Computational Complexity" in [156] or other introductive materials into the computational complexity.

The algorithm for consistency checks if a knowledge base consisting of a set of assertions and one of terminological axioms is non-contradictory. The algorithm for instances checks the relationship among individuals: an individual *i* is instance of a concept description C if it is always interpreted as an element of its interpretation.

BUPT

In order to ensure a reasonable and predictible behaviour of a DL system the inference problems should be decidable and preferably of low complexity. Thus, the expressive power of a DL must be restricted in a balanced manner: if it is too small then the important notions of the domain could not be captured, while if is too big then makes reasoning an untractable process. The investigation of this tradeoff between expressivity and complexity of reasoning has been one of the most important problems of the research in DL. Thanks to the results of research in DL during the last decades this hole became sufficiently narrow in order to allow to construct stabile bridges [25].

A) Consistency of $\mathcal{ALC}$ ABoxes is PSpace-complete

In the previous sections, we saw a tableaux algorithm that decides the consistency of $\mathcal{ALC}$ Aboxes with resp. to Tboxes. I will consider here firstly just ABoxes and I will explain how the algorithm can be implemented in polynomial space, that is I will show that the consistency of $\mathcal{ALC}$ Aboxes is a PSpace problem. Then I will show that it cannot be done better, so the problem is PSpace-hard.

For these considerations, we must see how to measure the length of the input description. Given an ABox $\mathcal{A}$, its size is the length required to write $\mathcal{A}$, assuming that the length required for writing the names of concepts and roles is 1.

The formal definition of the size of Aboxes is:

$|\mathcal{A}| = \sum_{a:C\in A}(|C|+1) + \sum_{(a,b):r\in A} 3$

$|A| = 1, where\ \ A\ a\ concept\ name$

$|\neg D| = |D| + 1$

$|D1 \cap D2| = |D1 \cup D2| = |D1| + |D2| + 1$

$|\exists R.D| = |\forall R.D| = |D| + 2$

In what follows let's take a look at the tableaux algorithm. Firstly worth mentioning the fact that, in absence of a TBox, neither the $\equiv -rule$ nor $\subseteq -\text{rule}$ are not applicable anymore; then the fact that the tableaux algorithm constructs a completion forest in a monotone manner, i.e. all the expansion rules either add concepts to the nodes labels or new nodes to the forest, but they do not erase anything. The forest built this way consists of two parts: for each individual name from $\mathcal{A}$ the forest contains a root node which will be called *old node*. The edges between old nodes come from the role assertions from $\mathcal{A}$, and so they can occur without restrictions. Other nodes, such are the ones from the completion trees that are not roots are generated by the $\exists -rule$, and will be called *new nodes*. We'll call the other rules "augmenting" because they only increase the labels of the existing nodes. Unlike the edges between old nodes, the ones between new nodes have a certain shape: each new node lays into a completion tree that is rooted into an old node.

Let's consider the nodes labels. Inițially, for an old node x_a, $\mathcal{L}(x_a)$ contains the concepts C from the assertions $a:C \in \mathcal{A}$. Other concepts are added by means of the expansion rules, and we observe that these rules add only subconcepts of the concepts that occur in $\mathcal{A}$. Because there are at most $|\mathcal{A}|$ those subconcepts, each node label could be stored in polynomial space in $|\mathcal{A}|$. Moreover, for each concept D from the label of a new node x, its predecessor contains a larger concept. This way, the maximum size of concepts from the nodes' labels decreases strictly alongside of a path constituted of new nodes, and thus the depth of each completion tree in the completion graph is bounded by: max{|C| | $a:C \in \mathcal{A}$}.

BUPT

Finally it must be stated that the expansion rules can be applied in a random order: the proof for correctness of the algorithm does not rely on an order of application. Thus we may very well use the following order: foremost all the augmentation rules are being exhaustively applied to the old nodes. In the next step we will treat each old node in particular and build the tree rooted on it in a depth manner, that is, for an old node x_a we will successively deal each existential restriction $\exists r.C \in \mathcal{L}(x_a)$:

- apply the $\exists - rule$ in order to generate an r-successor x_0 with $\mathcal{L}(x_0) = \{C\}$,
- apply the $\forall - rule$ exhaustively at this r-successor of x_a that can further add other concepts to $\mathcal{L}(x_0)$, and
- apply recursively the same procedure to x_0, i.e. apply exhaustively the augmentation rules then test each of the existential restrictions at turn

As usual, the algorithm stops if encounters a clash, otherwise, if all existential restrictions of a new node have been handled then it can be erased, including its label and reuse the occupied space. By using this method we can investigate the complete tree rooted in the old node x_a and holding a unique branch in memory at a given time. This branch is of a length that is linear in $|\mathcal{A}|$, and can be thus stored with all its labels in size polynomial of $|\mathcal{A}|$.

Continuing with the investigation of all trees in this manner, the algorithm requires a space that is only polynomial in $|\mathcal{A}|$. This approach is called the "tracing method" due to fact that it traces only the tree-shaped part of a completion tree [28].

In order to show that it cannot be done better, I will prove that the $\mathcal{ALC}$ ABox consistency is PSpace-hard even in case of Aboxes that contain a single assertion {a:C}. Proof is made by a reduction of the QBF validity problem, which is known to be PSpace-hard. A QBF formula ϕ is of the form:

$$Q_1 p_1 . Q_2 p_2 . \ldots Q_n p_n . \varphi,$$

for $Q_i \in \{\forall, \exists\}$ and φ a boolean formula over $p_1,\ldots p_n$. The validity of QBF is inductively defined as follows:

- $\exists p.\phi$ valid if ϕ[p/t] or ϕ[p/f] are valid
- $\forall p.\phi$ valid if ϕ[p/t] or ϕ[p/f] are valid

For example, $\forall p.\exists q.(p \vee q)$ is valid and $\forall p.\forall q.\exists r((p \vee r) \rightarrow q)$ is not valid.

Since QBF validity is PSpace-hard, it is only left to prove that for a certain QBF ϕ we can construct in polynomial time an $\mathcal{ALC}$ concept C_ϕ such that ϕ is valid only if $\{a:C_\phi\}$ is consistent. As an immediate consequence, $\mathcal{ALC}$ Aboxes consistency and concepts satisfiability are PSpace-hard.

The idea underlying the above reduction is to build a concept C_ϕ such that any of its instances x_0 is the root of a tree of depth n such that for every $1 \leq i \leq n$ we have the following relations:

- if $Q_i = \exists$ then $r \ldots r - succesor$ (i-1 times) of x_0 has an $r - succesor$ that can be either in p_i or $\neg p_i$

BUPT

- if $Q_i = \forall$, each $r \dots r - successor$ (i-1 times) of x_0 has two $r - succesors$, one in p_i and other in $\neg p_i$

Up to this point, for a QBF $\phi = Q_1 p_1 . Q_2 p_2 \dots Q_n p_n . \varphi$ we define C_ϕ as the following, where $\hat{\varphi}$ is the DL equivalent of φ obtained by replacing the ∧ with ∩ and ∨ with ∪ in φ:

$$C_\phi \coloneqq L_1 \cap \forall r . (L_2 \cap \forall r . \left(L_3 \cap \dots \forall r . (L_n \cap \hat{\varphi})\right) \dots),$$

where:

$$L_i \coloneqq D_i \cap \begin{cases} \exists r . T \; if \;\; Q_i = \exists \\ \exists r . p_i \cap \exists r . \neg p_i \; if \;\; Q_i = \forall \end{cases}$$

$$D_i \coloneqq \bigcap_{j<i} (p_j \to \forall r . p_j) \cap (\neg p_j \to \forall r . \neg p_j)$$

By means of this definition we assure that if $x_0 \in C_\phi^I$ and exists a path $(x_0, x_1) \in r^I, \dots (x_{i-1}, x_i) \in r^I$, then $x_i \in L_i^I$ and L_i is responsible for the branching format discussed above. The D_i concepts assure that if any of the x_i is (or not) an instance of p_i , for j<i, then so it is (not) x_{i+1}. These observations, together with fact that x_n is an instance of $\hat{\varphi}$ ensures that ϕ is valid if {a:C_ϕ} consistent.

Theorem 4: Satisfiability and subsumption of $\mathcal{ALC}$ concepts and their Aboxes consistency are PSpace-complete problems.

B) Adding general TBoxes

As is has been affirmed in one of the anterior sections, $\mathcal{ALC}$ concept satisfiability with resp. to general Tboxes can be decided in exponential time, i.e. this problem is ExpTime-complete. Again, it can be shown that it cannot be done better, so it is ExpTime-hard. Demonstration is though beyond the scope of the current chapter because requires the introduction of certain knowledge about "machinery of complexity theory". A possible solution is by adaptation of the ExpTime-hardness proof of PDL from [82]. This proof uses a polynomial reduction of the word problem for polynomially bounded alternative Turing machines to the satisfiability of PDL formulas.

When is translated into its equivalent DL, the formula for reduction of this proof has the form: $C \cap \forall r^* . D$, where C,D are $\mathcal{ALC}$ concepts and r^* is the transitive-reflexive closure of r, i.e. this concept uses a constructor that does not exist in $\mathcal{ALC}$. It is not hard to observe though that $C \cap \forall r^* . D$ is satisfiable if C is so with resp. to the TBox $\{T \subseteq D\}$. This is the case because *r* is the only role that occurs in C and D. For readers who wish to learn more insights about the relation between Tboxes and PDL are invited to read the work in [87].

Worth mentioning also that, for definitorial $\mathcal{ALC}$ Tboxes this explosion in complexity from PSpace to ExpTime-hard doesn't take place. We will see next that there are DLs in which even the presence of definitorial Tboxes may lead to growth in complexity.

BUPT

C) The effect of other constructors

In section 2 was made a discussion around the extensions with different constructors of the primitive DL $\mathcal{ALC}$. In this section, I will discuss about the influence that these constructors have on the complexity of reasoning.

Generally, number restrictions are harmless; with a single exception, even the qualified ones may be added to a DL without increasing its complexity of reasoning. For example, $\mathcal{ALCQ}$ concept satisfiability is still PSpace, while the consistency of general $\mathcal{ALCQ}$ knowledge bases is ExpTime-hard [88].

Transitive roles are the most harmless; all DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ can be extended with transitive roles without being affected their computational complexity. A dangerous interaction is that with roles hierarchies: concepts satisfiability of $\mathcal{ALC}$ extended with transitive roles and roles hierarchies is ExpTime-hard, while concept satisfiability of $\mathcal{ALC}$ extended with only one of these features is Pspace-complete [28]. This inherent growth in complexity is due to the fact that these 2 characteristics are used for Tbox internalization: given a TBox T and an $\mathcal{ALC}$ concept, E, that contains the role names $r_1,\ldots r_n$, we have that E is satisfiable w.r.t T if the following concept:

$$\exists r.E \cap \forall r. \bigcap_{C \subseteq D \in T} (\neg C \cup D)$$

is satisfiable w.r.t. $\{r_1 \subseteq r, r_n \subseteq r\}$, where r is a new transitive role. The first conjunct assures that the extension of E is non-empty, while the second that any element from a model satisfies any GCI from T. So, in $\mathcal{ALC}$ extended with transitive roles and role hierarchies we can reduce the reasoning polynomially with resp. to a TBox to that pure with concepts, so the pure concept reasoning is ExpTime-hard. In the additional presence of number restrictions there is the need to have a special concern not to use super-roles of transitive roles inside number restrictions because this leads to undecidability [28]. As consequence, expressive DLs ,like $\mathcal{SHIQ}$ allow the use in number restrictions only of simple roles.

Nominals and role inverses are also in majority of cases innocuous: concept satisfiability in $\mathcal{ALCOQ}$ and $\mathcal{ALCI}$ extended with transitive roles are PSpace, but concept satisfiability in $\mathcal{ALCIO}$ is ExpTime-hard. This growth in complexity is again caused by the fact that by means of role inverses and nominals can be internalized Tboxes. Intuitively, we use a nominal as a "spy point", that is an individual which has all elements from a model as t-successors, and we use role inverses to assure the behavior of this spy point. More exactly, a concept E is satisfiable w.r.t. a TBox T if the following concept is satisfiable:

$$o \cap (\exists t.E) \cap (\forall t. \bigcap_{r \in R} \forall r. \exists t^{-}.o) \cap \forall t. \bigcap_{C \subseteq D \in T} (\neg C \cup D),$$

where *o* is a nominal, R is the set of roles *r* that occur in T or E and their inverses r^{-}, and *t* is a role that is not found in R. The third conjunct assures that *o* 'sees' all elements from a connected model, i.e. x_0 is an instance of the above concept in a connected model I and there exists an element $y \in \Delta^I$, then $(x_0, y) \in t^I$.

In the end worth mentioning the fact that nominals, role inverses and number restrictions altogether have a dramatic influence over the complexity: satisfiability of $\mathcal{ALCOIQ}$ concepts is NExpTime-hard [194], while satisfiability of $\mathcal{ALCIO}$, $\mathcal{ALCIQ}$,, $\mathcal{ALCOQ}$ concepts is ExpTime-hard (with resp. to Tboxes) [27].

For results of complexity in other DLs created by the scholars of the domain together with their proofs, I think the best material that should be consulted is Baader's book intuitively called "Description Logics Handbook" [28], especially chapter 3 which is dedicated exclusively to the field of DL complexity of reasoning (but also others that treat the domain). Readers who seek introductory materials are invited to browse Ian Horrocks' courses in the references section about reasoning in DL [104],[105],[106].

D) A List with Results of Complexity of Reasoning in DL

A plethora of Description Logics had been created until present day in literature by scientists, each having a specific name. Although suggestive, these names do not offer too much insight about what constructors that language holds. This makes the big majority of the results of complexity of reasoning hard to understand by the non-expert users. In order to clear things about what constructors exist in each language here I will use 2 lists: first one shows the concept constructors and second one of roles. For example, the pair of lists $(\cap, \exists R, \forall R.C)(\cap, °)$ says that the language has as concept constructors: conjunction ($\cap$), unqualified existential quantification ($\exists R$), universal role quantification ($\forall R.C$), and for role constructors: conjunction ($\cap$) and composition ($°$). To many combinations of concept constructors have been given names that now are generally being used. For example, the above list of concept constructors is being known in the literature as being the DL $\mathcal{FL}^-$. Here I will use a syntax proposed in [27] and write $\mathcal{FL}^-(\cap, °)$, which means $\mathcal{FL}^-$ augmented with role conjunctions and compositions, in order to make it recognized for the domain researchers.

Appendix A presents a catalog with a series of results to the tasks of satisfiability and subsumption for some of the main DLs that had been culled from the works in literature read for the realization of this thesis. Satisfiability and subsumption refers to problems with simple concept expressions. When these problems are considered with resp. to a set of axioms I will explicitly state this fact. Moreover, when constructors of a DL allow the reduction of concept subsumption ($C \sqsubseteq D$) to satisfiability ($C \cap \neg D$) will be stated just satisfiability. In this list I tried to use symbols of DL constructors as much as possible; for some I used abbreviations, such as: unqualified number restrictions ($\geq n\, R, \leq n\, R$) are denoted with <> R, qualified number restrictions ($\geq n\, R.C, \leq n\, R.C$) with <>R.C. If a constructor is allowed only for names (either concepts or roles) will be applied for the word $name$.

BUPT

2.4. State-of-Art of the Researches

Because the current chapter is intended as a review aiming to offer the reader insight from the application domain, the vast majority of materials read in conducting the current research is mostly theoretic. These materials (resources) include books, faculty courses, PowerPoint slides on the Internet etc. A presentation of the most important is done in the continuation of this section.

The most rightful work with which I will start this presentation is the book of Baader et al. from year 2003, called "Description Logics Handbook: Theory, Practice, Applications" [28]. This is an exhaustive work that spans over many chapters in order to create a comprehensive guide for the domain of Description Logics. It is divided into 3 main parts. The first one presents introductory aspects from the domain, the second shows implementations of some DLs into concrete systems, while the last deals with the application domains where DLs are most frequently being used. For my thesis I studied mainly the first 3 chapters where the main reasoning tasks of DL knowledge bases are discusses and the techniques used for their solvation (decision procedures), the complexity of these algorithms for some members of DL family ($\mathcal{AL}$, $\mathcal{EL}$, $\mathcal{FL}^{0}$ augmented with various 'flavors').

Another valuable paper is that of Baader, Buchheit and Hollunder [24], that represents the 23rd chapter of the book "Handbook of Automated Reasoning" [165]. Its goal is to offer a complete presentation of the results of reasoning in presence of Tboxes for very expressive DLs (i.e. those that contain all concept constructors, inverses of roles and number restrictions). Presents the reasoning problems for unrestricted models from PDL, techniques of reasoning and results for the finite DL models. It is also being taken a look over the boundary between decidability and undecidability in very expressive DLs.

Also one of the most notable works from this domain read for the creation of the current thesis is that of Jeff Pan, that was his doctorate thesis titled "Description Logics: Reasoning Support for the Semantic Web" [154]. This book starts from the basic notions, such as general aspects of the Semantic Web, ontologies and Description Logics languages, and with each chapter advances towards more and more complex notions, such as reasoning tasks and services, complexity of the reasoning algorithms, optimization techniques for assuring the decidabilităţii. Its contributions are two decidable extensions of the OWL ontology language in order to support customized data types and predicates, that is OWL-E and OWL-Eu. Also proposes a framework for reasoning that supports a wide array of decidabile DLs that provide customized datatypes and predicates.

Another pioneer in this domain, English scholar Ian Horrocks, delighted us with some valuable works here too. One of the largest that tries to present the DLs domain even from their birth and until present time is [25], which is actually the third chapter of the book "Handbook of Knowledge Representation". Starting with their definition and history, then continuing with the relation with other logical formalisms (Modal, Predicative) and a separate section dedicated to the basic DL $\mathcal{ALC}$ in which discuss its syntax, semantics and main inference problems: consistency, satisfiability, equivalence and subsumption. Also in a separate section is talked about the main technique for solving these problems: the tableaux algorithms, while in other are mentioned others: the structural algorithms and those automata-based. At the end are presented a series of complexity results for the above problems solved by using the tableaux technique.

BUPT

The same author in [110], [111], [112] presents the role of DLs as languages for the development of ontologies that can be used on the Semantic Web. Firstly is made an introduction into the domain then is explained the reason for which they are fit as ontologiy languages, especially due to their well-defined semantics and the existing reasoner tools. Separate sections are dedicated to particular DL languages, such as $\mathcal{ALC}$, $\mathcal{SHIQ}$, are discussed syntaxes, semantics, reasoning tasks etc.

Also a comprehensive work coming from the same authors that provides advanced notions is found in [114], but this time is being discussed OWL. OWL is the standard ontology language of the Semantic Web that roots its characteristics from several representative languages, such as Descriptin Logics and Frames Paradigm, and also RDF, the foundation proposed by W3C for the Semantic Web. Presents its predecessors, languages that influenced its creation such as SHOE, DAML-ONT, OIL, DAML+OIL, the problems that were faced by the group who dealt with its development related to the syntax, semantics, computations, expressivity.

(Dalwadi et al., 2012) make in [68] a study about Semantic Web technologies and discuss some of the inference engines that support reasoning over the OWL ontologies. Present the main techniques that are usually employed in the reasoning process, which are based on DL, on theorem provers from FOL, or on a combination between FOL and general logics, then provide examples of concrete reasoner systems from each category. Finally he makes a comparison of the major reasoners currently existing on the market, showing what algorithms each one relies on, what is the underlying logical formalism, as well as many other traits that were presented in tabular form.

Turhan makes in [196] a guided tour of the DL domain, stating he did that in order not to make "yet another" review paper of the domain but to present resources, courses, documentation where the reader can find information about this domain. A particular section is dedicated to reasoning in DL, where he also presents materials from the literature.

Readers willing to gain more advanced knowledge about reasoning and complexity results in a large array of Description Logics, undecidability, results borrowed from other related logics are invited to read chapters 1,2 and 3 of Baader's handbook in [28]. Complexity results are presented under the form of proven theorems in order also to show the deduced results. At the end of chapter 3 it is being produced a list with some complexity results for satisfiability and subsumption in a wide range of DLs, asides which is stated the name of the person who discovered that language and proved the result. Also the resources for faculty courses of Horrocks [107], [108], Baader [21] and few presentational slides of Horrocks in which explain how a tableaux algorithm works and introduces fundamental notions about inference and reasoning in DLs [105], [106]. Readers are advised also to look upon the Wikipedia sources about these domains, that is a trustful source of information that could be validated by the author(s) of this thesis himself.

In [197] Turhan makes an introduction into the DL systems. Are presented the general notions of the domain, such as knowledge representation formalisms, the phases on which went through the researches in DL, general components of a DL knowledge base (intensional, extensional). Also are being presented the connections that DLs hold with other logical formalisms, like First Order Logic or

Modal Logic, the reasoning services provided by DL systems (both terminological and assertional), what are the main characteristics of tableaux algorithms for reasoning, such as termination, complexity, and are given concrete examples on the basic language $\mathcal{ALC}$.

In (Braun, 2006) [56] it is proposed a new system for reasoning in DL called KAON2. Its main characteristics are that relies on non-tableaux techniques for the realization of inferences, techniques which are borrowed from the area of databases. As it is well known, DL systems can be related to relational databases in the sense that a TBox is equivalent to a schema and an ABox to the data from the recordings. The advantages of this new algorithm are that it relies on state-of-art techniques from the domain of database management, such as the optimization of query order, magic sets transformations, avoiding this way the drawbacks of tableaux solutions.

2.5. Conclusions

In this chapter I set out the approach of the role that logic, or, better said logical knowledge representation formalisms, to the development, standardization and evolution of the Semantic Web.

History of logical paradigms for knowledge representation knew three important development phases: Semantic Networks, Frame Systems and Description Logics. The latter presents the advantage that has logic-based formal semantics, unlike its other two predecessors. The decision to build the ontology languages of the Semantic Web on top of the Description Logics paradigm was due to the fact that these provide some model-theoretic semantics that are well-defined in terms of interpretations and that it exists a variety of reasoning systems created in this respect.

The chapter began with an introduction in ontologies, which are the main domains where DLs find applications, and what are the most important languages for their specification that have been created. I presented then the domain of Description Logics, showed a brief historical parcourse with the evolution phases through which they passed from their inception, syntaxes and semantics of the DL knowledge bases, and has been explained the reason for what they form some good candidates to the ontologies languages. Has been subsequently shown what are their relations with other logical representation formalisms, such as First Order Predicate Logic (FOL) or the Modal Logic. In case of the former, DLs are decidable fragments and are preferred in place of these to the knowledge representation task because they make reasoning process tractable, while in case of the latters DL concepts can be seen as propositional variables, roles as modal parameters and interpretations as Kripke structures.

Had been subsequently presented DL languages that have been proposed by now in the domain literature, especially those of the major scholars of the domain (Baader, Horrocks, Donini etc). Had been presented in the form of a table the most important languages of the $\mathcal{S}$ family then had been made a series of studies, discussions, analyses about what type of knowledge can be described by their means (concepts and roles constructors) and what new characteristics must be added to a language in order to create another one, starting discussion with the

BUPT

most of all, $\mathcal{AL}$ and towards the most complicated ones (e.g. $\mathcal{SROIQ}$). A separate section has been dedicated to the DL's domains of applications, the most important, as it is well known, the ontology languages of the Semantic Web. Were discussed the two major ontology languages created by now, OWL and OWL2, were specified inside tables the DL languages that represent each one's basis in what concerns the syntax and semantics for constructors of concepts and roles, axioms etc. in order to enhance the reader's understanding about the relations between the two formalisms.

Inference tasks from the DL knowledge bases have been described in the next section, there were also explained the reasoning services created for their solvation and the concrete reasoning systems that implement such techniques. Had been presented and analyzed some complexity results of those problems in the case of some particular DLs, as they had been demonstrated in the domain literature by the scholars. Were made analyzes and comparisons about the way how the addition of new features (concepts and roles constructors) influence the complexity of reasoning in that language starting from the most basic one, $\mathcal{AL}$. Had been stated what combinations of constructors are the most harmless (do not change the reasoning complexity) and which are the ones that generate real "explosions" in complexity. Next followed the presentation of the reasoning algorithms (decision procedures) that are used for solving the inference tasks from DL bases, with a special emphasis on tableaux ones, that currently are the most widely used for determining the satisfiability and subsumption between DL concepts. The chapter concludes with a list of results for complexity in some of the most important DLs that were gathered by me from literature, and in the last section I made a state-of-art with the most important works read for the creation of this chapter, and altogether are being provided to the reader references in literature where they can expand their knowledge about that certain domain that is discussed there.

BUPT

page intended left blank

BUPT

3. REASONER SYSTEM FOR LOGICAL KNOWLEDGE BASES SATURATION

In this chapter I propose a reasoning system for a special type of knowledge bases that, besides the usual ones, i.e. facts and rules, also contains another one that represents some restrictions in respect to what kind of data can exist into the database. The proposed system applies repeatedly the set of rules on the set of facts in order to deduce the implicit ones while in the same time checks them with the constraints from the set before introducing into the database. This procedure is aimed to deduce all implicit knowledge from the initial set of facts. The chapter begins with an introductory section into the basic terminology of logical knowledge bases, especially the k-derivation and saturation procedures, in order to familiarize the reader with terms being used during the parcourse of the chapter in which will be described the functionality of the logical reasoning system. I created models of the proposed system in the form of UML class and activity diagrams that present the structure and functionality of it. The chapter will conclude with a series of comparations of the proposed derivation algorithm with others more primitive ones, in order to show its superiority especially in resp. to complexity and computation time.

3.1. Logical Knowledge Bases: General Notions

In this section I will try to introduce the reader into the theoretical notions about the logical knowledge representation formalisms. I will lead my focus on what logical knowledge bases are, what are their main components and roles, what operations we can do upon them etc.

I will consider here the existential positive syntactic fragment of First-Order Logic, that is FOL(∃,∧), which is consisted of formulas created using only the quantifiers (∃, ∀) and the connectors implication (→) and conjunction (∧) (thus no disjunction and negation). A special type of constant is used to denote falsity in formulas, called "bottom" concept, and has the symbol (⊥).

A *Vocabulary* is an entity consisting of three disjoint sets:

$$Voc = (C, V, P) \quad (1)$$

where C is a finite set of constants, V an infinite set of variables and P a finite set of predicate. A function $ar:P\rightarrow N$ associates a natural number to any predicate $p\in P$ which states its arity. A *term* over *Voc* is a constant $t\in C$, or a variable $t\in V$. An *atomic formula* (or simply *atom*) on *Voc* is of the form:

$$p(t_1,\ldots t_n) \quad (2)$$

where $p\in P$, $ar(p)=n$, si $t_1,\ldots t_n$ are terms from *Voc*. Here I will use the convention of using capital letters for constants names and small letters for variables.

BUPT

A basic atom is one that does not have any variables (only constants). A conjunction of atoms is called *conjunct*, while one of basic atoms is called *basic conjunct.* A variable from a formula is called *free* if it does not lie in the scope of any quantifier. A formula is *closed* if it does not have any free variables (also called *proposition*).

A method to represent knowledge about the real world is to grasp facts. This is seen as the most basic form of knowledge. A fact can be represented by a basic atom because this one is used too to denote primitive forms of information. For example, the chunk of information "John teaches Tom" is primitive (atomic). To be able to represent incomplete knowledge existentially quantified variables are being used within facts, and not just constants, as it was affirmed by [1]. To give a more formal definition: "a *fact over Voc is the transitive closure of a conjunction of atoms*". For example: "*Professor John teaches a student whose name he does not knows*" can be represented by the FOL formula:

$$\exists x(Professor(John) \wedge Student(x) \wedge Teaches(John,x)) \quad (3)$$

Another form of knowledge representation are *rules*. Rules had been extensively applied in the knowledge representation systems and in those expert. These are general formulae that have variables and unknown individuals, being known also under the name *existential rules*. These are logical formulae that allow the inferring of new facts from the ones existent, considered an ontological layer that imposes the expressivity of the knowledge base by encoding the so-called *intensional knowledge*. To give a more formal definition: an existential rule is a closed formula of the form:

$$R = \forall \vec{y}((\forall \vec{x}B) \rightarrow (\exists \vec{z}H)) \quad (4)$$

where B and H are called the *body* and *head*, respectively, of the rule and are conjunctions of atoms, with vars(B) = $\vec{x} \cup \vec{y}$ and vars(H)= $\vec{z} \cup \vec{y}$. I used here the notation with *terms(F)* and *vars(F)* in order to denote the set of terms and variables, respectively, from a a fact F; I noted by $\vec{x}$ a shorthand of a sequence of variables (x_1,x_2,…x_n), duplicates excluded, which makes a fact to be regarded as a set of atoms. Variables $\vec{z}$ are called the existential variables of the rule [1]. Existential rules had been proposed in order to solve the limitations of Horn clauses (found especially in logic programming) by allowing more than one atom in rules' head, as it can be observed also in the above definition, in order to represent existential variables and the possibility to have unrestricted arity of predicates. It can be easily thought that this power of expressivity comes in the same time with the reduction of decidability, some problems from the existential rules framework are undecidable (such is determination). A vast array of existential rules classes that ensure decidability while in the same time keeping expressivity had been proposed in literature (to be seen [31])**.** To be able to represent knowledge about the real world we must take into account also the negative ones, which are those that dictate how things should not be. Existential rules do not contain negation, which makes it hard to represent this type of knowledge. The notion of *negative constraints* is the logical counterpart of integrity check constraints from the relational databases in order to forbid certain values for inputs and preserve data semantics. Their definition is like this: a negative constraint is a rule of the form:

$$R = \forall \vec{x}(B \rightarrow \perp) \quad (5)$$

which signifies a rule with no head, only body (head is the ”bottom” concept). The negative constraints have an important role for the logical knowledge bases, serving as logical devices for detecting inconsistencies in the factual part. Triggering of a negative constraint is interpreted as presence of some inconsistency in the knowledge base[1]. An example of negative constraint:

$$N=retiredFrom(x,y) \wedge worksAt(x,y) \rightarrow \bot \qquad (6)$$

which says that it is impossible for a person to be retired from an institution while in the same time to work there.

After the 3 main formalisms for knowledge representation have been presented, now I can provide the reader the definition of what a knowledge base is, as it was given in [1]. A *knowledge base* (abr. KB) over a vocabulary *Voc* is a triple $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ constituted from a finite set of facts, rules and negative constraints.

Generally, what we must do when we have a set of facts supplemented with one of rules and negative constraints is to use rules in order to deduce implicit knowledge from facts while in the same time to be careful not to violate any of the stated negative constraints. The result of this procedure is a set of facts that extends the initial one. In classical logic this procedure is known as the *Modus-Ponens inference principle*, while in Datalog is referred as the *Elementary Production Principle* [61]. In order for this to work rule’s body must match with some facts from the set F, in other words, there is a substitution of variables that makes the body of the rule R resemble with some facts from F. Before moving on to the last definitions of the section, those for rules applications, derivation sequence and saturation, it is necessary to give some others: substitution and homomorphism.

Given a set of variables V and one of terms T, a *substitution* σ of V with T (written $\sigma:V\rightarrow T$) is an association from V to T. Given a fact F, σ(F) denotes the fact obtained from F by replacing each occurrence of $x\in V\cap vars(F)$ with σ(x). A *homomorphism* from a fact F to another one F’ is a substitution σ of vars(F) with terms(F’) such that $\sigma(F)\subseteq F'$ [1].

A rule $R=B \rightarrow H$ is *applicable* to a fact F if exists a homomorphism σ from B to F. The application of the rule with resp. to σ produces a new fact $\alpha(F,R, \sigma) = F \cup \sigma^{safe}(H)$, where σ^{safe} denotes the safe substitution that replaces the existential variables with others safe ones. α(F,R, σ) is called the *immediate derivation* of F. The new (unintroduced) variables are used in order to avoid attribution of variables that are already in use to the new facts, which would cause a problem when rules are reapplied on the new facts.

Rules can be applied in a certain order, for example it is possible that a rule R_2 not be applicable to any fact from the initial set but to become so only following application of other rules. This gives birth to some processes called *derivation sequence* and *R-derivation*. In what follows I will try to give the formal definition of those terms.

BUPT

Let F be a fact and $\mathcal{R}$ a set of rules. A fact F' is called the *R-derivation* of F if exists a finite sequence (F_0,F_1,…F_n) called *derivation sequence* in which F_0=F, F_n=F' and for any 0≤i<n exists a rule $R_i \in \mathcal{R}$ that is applicable to F_i and F_{i+1} is his immediate derivation. These, together with other notions can be found in [31].

When having an initial set of facts $\mathcal{F}$ and one of rules $\mathcal{R}$ we are interested in unveiling all implicit knowledge from the facts by using rules. This process is called *saturation procedure* that uses a breadth-first search scheme and forward chaining. We start with a derivation sequence in which F_0 is $\mathcal{F}$ and at each step is produced a new fact F_i from the current one F_{i-1} by computing all homomorphisms from the bodies of all rules to F_{i-1} then is made the corresponding rule application. The fact F_k obtained after the k^{th} step is called the *k-saturation of* $\mathcal{F}$. Next I will give the formal definition of the process, as it was related in [31].

Let F be a fact and $\mathcal{R}$ a set of rules. Π($\mathcal{R}$, F) is the set of homomorphisms from the bodies of all rules to F.

$$\Pi(\mathcal{R}, F) = \{(R,\sigma) \mid R \in \mathcal{R} \text{ and } \sigma \text{ homomorphism from body}(R) \text{ to } F\} \quad (7)$$

The direct saturation of an arbitrary fact F with $\mathcal{R}$ is defined as follows:

$$Cl_R(F) = \bigcup_{(R,\sigma)\in\Pi(R,F)} \sigma^{safe}(\text{head}(R)) \quad (8)$$

K-saturation of F with $\mathcal{R}$ is denoted with $Cl_R^k(F)$ and is inductively computed as:

$$Cl_R^0(F)=F \text{ and for } i>0,\ Cl_R^i(F)=Cl_R(Cl_R^{i-1}(F)) \quad (9)$$

Denote with $Cl_R^\infty(F) = \bigcup_{k\in N} Cl_R^k(F)$, where $Cl_R^\infty(F)$ is possibly infinite (i.e. derivation could possibly never terminate).

The saturation procedure is being known in the database community under the name of *"naive chase"* [59],[4] and it has been used especially in the database repairs domain that do not respect certain functional dependencies. Other types of chases that had been mentioned in literature are core and skolem, these differing only by the way in which they treat existential variables and redundancy [69],[137].

Since this is the main objective of this chapter, to create a system that realizes the saturation of a logical knowledge base, next I will provide an example of how this process works in order for the reader to make a better idea regarding the scope and mode of operation of the proposed system.
Consider the following knowledge base $\mathcal{K}$ = ($\mathcal{F}$, $\mathcal{R}$, $\mathcal{N}$):

- $\mathcal{F}$: {p(A), q(B), s(B)}
- $\mathcal{R}$: {p(x) →r(x,y), p(x)Λs(y) →p(y), q(x) →r(x,y), r(x,y) →t(x)}
- $\mathcal{N}$: ϕ.

BUPT

The derivation procedure, as it has been stated in the definition, represents a breadth-first search with forward chaining that works by matching each rule over the set of facts and after each iteration the set of newly produced facts is added to the ititial one, and the process restarts with the next iteration. In our case we have:

step1 - $Cl_R^0(\mathcal{F}) = \mathcal{F}$

step2 - $Cl_R^1(\mathcal{F}) = \mathcal{F} \cup \{r(A,y_1), p(B), r(B,y_2)\}$, by applying R_1 with $\sigma_1 = \{(x,A)\}$, R_2 with $\sigma_2 = \{(x,A), (y,B)\}$, R_3 with $\sigma_3 = \{(x,B)\}$

step3 - $Cl_R^2(\mathcal{F}) = Cl_{\mathbf{R}}^1(\mathcal{F}) \cup \{t(A)\}$, by applying R_4 with $\sigma_4 = \{(x,A),(y,y_1)\}$, R_4 with $\sigma_5 = \{(x,B), (y,y_2)\}$.

step4 - no more facts are being produced from rules applications, so the process ends after 3 steps.

3.2. State of the Research

The system that is proposed in this chapter is a private one, being specially created for the needs of an institution from France in which I worked together with other staff members from there; this is Universite de La Rochelle. Everything that is being presented in this chapter represents original work, the ideas, solutions and creations of the thesis' author. The most important resource read in creation of the system was the PhD thesis of one of my colleagues from La Rochelle, [1], which is from the domain of knowledge representation and reasoning, argumentation frameworks, dialectical explanations. Another resource worthy of mention is the book of Dung [74] which discusses the abstract argumentation frameworks and was used in [1] to instantiate such a framework with arguments stored in a knowledge base and using a logical language for representation, Datalog+. Poggi [159] discusses about ontology-based data access inside the logical knowledge bases. Croitoru et al. [66] proposed a theorem of representations between the preferred /stable extensions and the data repairs of Lembo [131], which is crucial for exploiting the explanatory power of argumentations in order to explain the accept of interrogations under the semantics of Consistent Query Argumentations (CQA).

The most related domain in literature to which I can compare my work done here are the business rule engines (Business Rule Management - BRM) from object-oriented languages (I will consider here Java). Business rules are being used inside Java applications with the aim to separate the business logic from the source code, a fact that facilitates application development and maintenance. The most important frameworks for business rule management from the Java programming language that have been created until present are: Drools [218], OpenL Tablets, EasyRules, RuleBook [218], JRules [221], etc. Readers who wish to find out more about these are forwarded towards the references section where are specified the links to their documentation. The main architectural components of such systems are: the memory (facts and rules base), Unifier (algorithm), Agenda. The typical usage of such a system inside an OO application is to trigger rules over the set of facts when certain conditions are fulfilled. For example, set discount to the prices of winter clothes in the first day of the Spring, set a discount to objects whose type is that of precious metal or stone, etc. Thus, as it can be seen, these engines are being used for making light logical operations into an application.

BUPT

It can be guessed that using none of these frameworks it is not possible to achieve the purpose of my system, that is to obtain the saturation of a logical knowledge base by repeatedly making derivations. That is why I conceived the proposed system for the realization of intensive logical tasks, and was designed from scratch in the OO environment for this purpose.
The contributions of the proposed system compared to the frameworks of rules described above are:

- knowledge base, in its raw logical form (i.e. sets of facts, rules and restrictions) are being represented in FOL syntax FOL (First Order Logic), thus it can be understood by anyone that is familiar with this syntax
- grows the expressiveness of the classes of knowledge; for example could be expressed quantifiers for variables from within the facts and rules, a fact which can be useful in certain situations
- addition of a new class of knowledge, the negative constraints, which impose some additional restrictions onto the knowledge base in order to keep its consistency. Thus, after new facts has been produced following rules applications, before adding them to the memory are being checked with the set of constraints in order to see if they clash with anyone, and if affirmatively then the base rises an anomaly
- uses optimization techniques, such as forward chaining and the Backtracking algorithm for finding all the possible solutions into an iterative, ordered and progressive manner, a thing that simplifies the computations especially when we are dealt with very large knowledge bases (as will be shown in the experiments section).

3.3. Design

For being capable to process it and apply the derivation algorithm over it must find a way to represent the logical knowledge base as an object-oriented structure. In the rest of this section I will describe my found design methodology for solving the initial problem.

3.3.1. Objectual Structure

Each type of logical knowledge (atom, fact, rule, constraint) is being represented in the OO environment by a class that defines a set of specific attributes and corresponding methods.

An atom (predicate) which, as I stated in the introduction section, has the form: $P(t_1,t_2,\ldots t_n)$, has been represented in OO environment by a class named *Atom* that has 2 attributes: a string that contains its name and a collection of string values that store its values. It have been provided 3 constructors (2 parameterized and one for copy), getter and setter methods for each attribute and two Print methods in order to represent the atom at the console in its familiar form for the reader: name followed by a list of values between parenthesis separated by comma.

A fact, as it was affirmed in its definition from section 1, is the existential closure of a conjunction of atoms. This has been represented in the OO environment by a class called *Fact* that has as instance variables a collection of Atom objects and,

BUPT

similarly with the Atom class, defines 2 constructors and 2 methods for pretty-printing the fact in its usual form.

A rule was represented by a class *Rule* which has as instances 2 objects of type Fact: one represents the body (premise) and the other the head (consequence).

A negative constraint is a rule that has no head, so this was specified by a class that has as instance variable only one object of type Fact, which represents its body.

This structure of the system as has been described above is presented in fig.2, that is the UML class diagram of the system.

The logical knowledge base is stored in external files, one for each type in particular (fact, rule, constraint) and is being represented in raw FOL syntax. The main operations performed by my reasoning system are as follows:

- read the external files with data from the base, creates instance objects for each record from the base in which puts its data and creates collections in which puts all created objects
- applies the derivation algorithm over the base in objectual form (collections of objects) in order to deduce all implicit knowledge from the explicit ones; the process repeats in a recursive fashion until saturation is achieved (of the knowledge base)
- after the set of new (saturated) facts has been computed it is translated to FOL formulas and appended to file with the iniţial set of facts

The process is shown in fig.2, which is the UML activity diagram of the system.

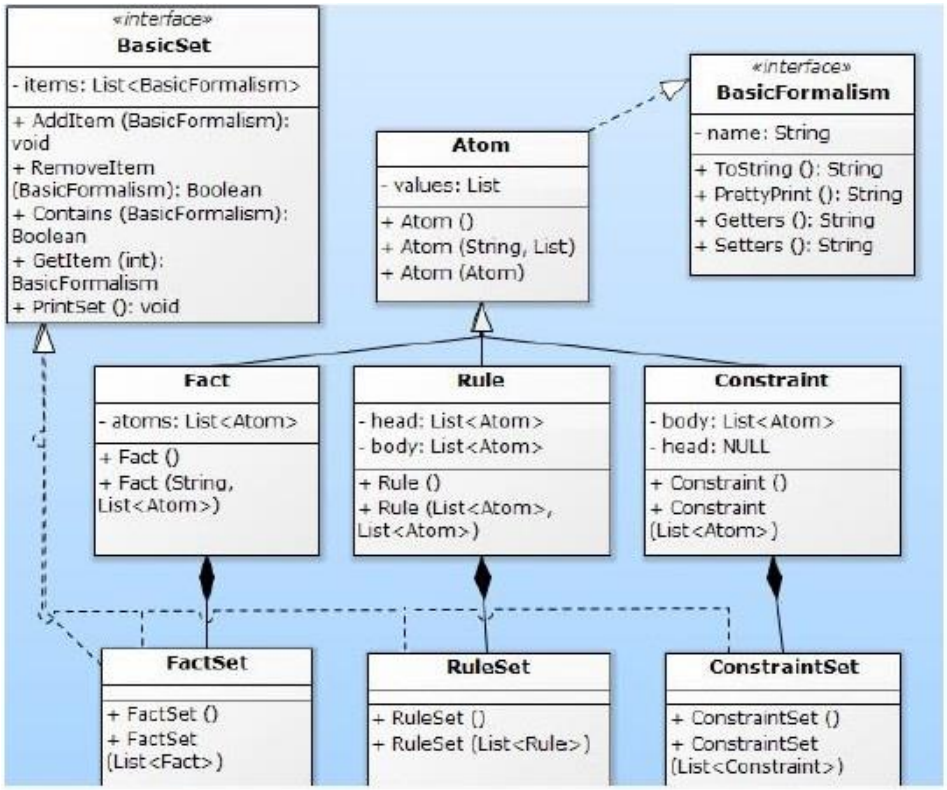

Fig.3.1: UML class diagram of the reasoner

3.3.2. Derivation Algorithm

The algorithm that relies at the basis of the proposed inference engine for producing new facts is specified by the next pseudocode.

Algorithm R-derivation:

Inputs: facts set, rules set, constraint set

Outputs: deduced set of facts

Variables: Dictionary[] (stores the variables of a rule together with their values found during the process of matching the facts)

Begin

while (new facts are produced by rules applications)

begin

1. *Take each rule from the set and tries to find a set of facts that matches her*
2. *Take each conjunct from the rule's body and try to find a fact from the set that matches it – a process in 3 steps*
   - *2.1. Verify if the conjunct has the same name with the fact*
   - *2.2. Verify if the conjunct has the same arity with the fact*
   - *2.3.Verify in Dictionary[] if the atom has variables that were assigned (this is in case in matching of previous conjuncts); if yes then check if their values are identical with the corresponding ones from the fact*
3. *If all previous 3 conditions have been fulfilled then the fact matched the conjunct, so substitute the conjunct's variables in Dictionary[] with the values corresponding from the fact*
4. *If there was a conjunct in the current rule that was not unified by any fact from the set then stop searching for other conjuncts since the current rule cannot be matched*
5. *If all conjuncts of the rule have been matched by some facts from the set then rule has been matched, thus the fact in the rule's head is computed based on the values of variables stored into Dictionary (i.e. rule's substitution)*
6. *Validate the new fact with the set of constraints in order to see if clashes with someone*
7. *If the fact is valid then add it to the set of produced facts, otherwise is being discarded*

END.

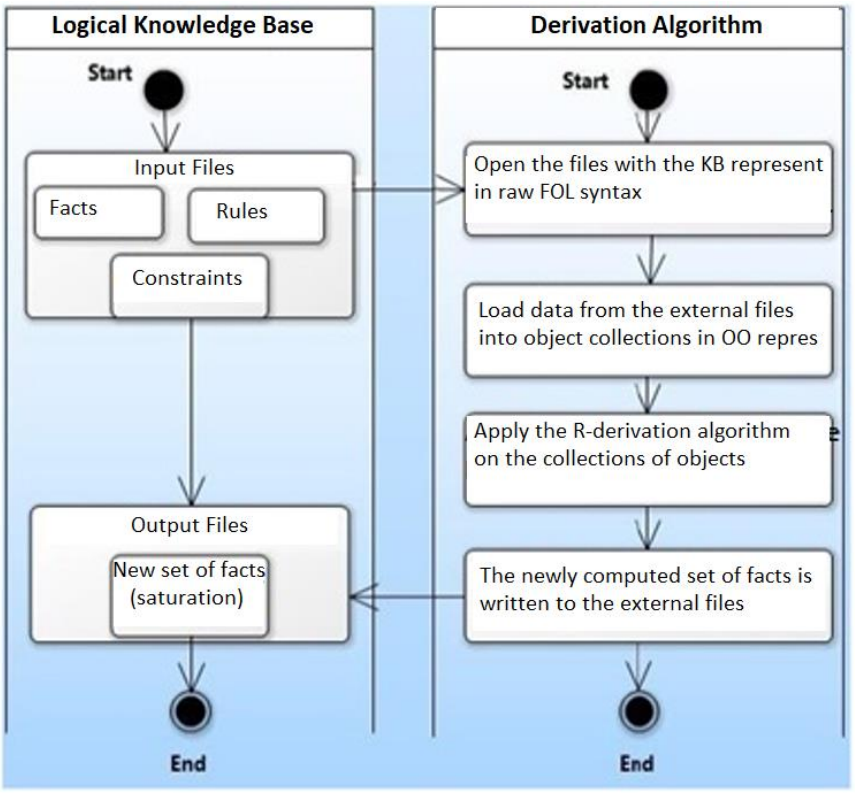

Fig.3.2: UML activity diagram of the reasoner

3.4. Implementation

In this section I will give details regarding the implementation of the proposed reasoner system.
For the implementation I chose an object-oriented programming language, Java, due to its numerous pros it offers, such as scalability, reliability, portability on different machines.

After the representation of the knowledge base under the form of an objectual structure it is possible to make the desired computations. In this case, the aim of the system is to deduce new information from the explicitly represented ones by applying the rules over the set of facts until is achieved saturation. Here I will try to detail and explain the logic behind the proposed derivation algorithm.

In the most simple case, that is when a fact is constituted from a single atom and the rule's head and body are also basic atoms, the rule matches the fact if its body has same name and arity with the fact.
Example:

F=Professor(John), R=Professor(x) ->Persan(x)

In this case it is obvious that the rule matches with the fact since both conditions are fulfilled: the body has same name and arity with the fact. After the application, a new fact is produced: *Person(John)*.

In practice though, facts and rules are not that simple, in fact they are conjunctions of atoms with arity bigger than 1. Let's take another example:

F_1=Professor(John), F_2=TeachesAt(John,Stanford),
R=Professor(x)ΛTeachesAt(x,y) -> University(y)

First conjunct of the rule matches the fact F_1 and the second conjunct the fact F_2 on both conditions (name and arity). Also should be remarked a correspondence between the variables names: x is the variable of the first conjunct and is also the first variable of the second conjunct, which means that the values of the facts that match the conjuncts must be correlated too in this manner. This is the third condition that should be fulfilled in order to ensure a correct match of rules. For example, if the second fact would have been: *F_2=TeachesAt(Mary,Stanford)* then the rule would not have been matched anymore by the facts due to this condition.

In order to implement this third condition into my algorithm I used a data structure of type Dictionary that keeps data in the form of (key-value) pairs. This contains the set of variables (distinct) of the rule together with the values found during the process of conjuncts matching over the set of facts. When passing on to the next conjunct of the rule for finding a matching fact for it , first thing that the algorithm does is to lookup into the Dictionary all its component variables in order to check if there are ones that have assigned values (that means they are variables also found in previous conjuncts) and if affirmative then will be replaced with their corresponding values. After a conjunct has been matched by a fact following all three conditions then to the variables of the conjunct are assigned the corresponding values from the fact and are written in the Dictionary. As it can be noticed, the Dictionary plays the role of rule's substitution (see sect.1 for definition).

In the implementation of the proposed system had been considered two cases that lead to two different implementation strategies but that have different computation complexities:

a) facts from the knowledge base are basic atoms
b) facts from the knowledge base are in the general form (conjuncts of atoms).

For the first case has been created the following strategy: in order to find a match for a rule (i.e. a subset of facts) their conjuncts are sequentially considered and for each of them is looked up into the set of facts for one that matched the conjunct after all the 3 conditions stated above (name, arity, variables). If all conjuncts of the rule have been analyzed and matched by some facts then the rule is considered to have been matched so the fact from the head is calculated by replacing its variables with their corresponding values found during the matching process. This way a new fact is being produced.

As it can be thought, this problem of matching a rule on a set of facts doesn't have only one solution but many, i.e. there are multiple subsets of facts that match the body of a rule and in order to produce new facts. Due to this reason, in order to compute all the possible solutions of the problem my derivation algorithm relies on an algorithm which is very popular in this type of problems, which is *Backtracking*. One of the most renowned problems that requires the application of Backtracking algorithm is the "Chess Queens Problem". This problem requires finding of all possible layouts of N queens on a chessboard of NxN squares such as not to attack each other.

The current problem, as can be seen, is very related to the queens problem because implies searching through N sets (better said, N times through same set) of elements that needs to fulfill 3 conditions for being possible to be introduced in the final solution and the result is a vector with the N elements found. After a solution has been found the searching does not stop but moves a position backwards and looks into the rest of elements for another fact that may match the conjunct. It stops when the counter variable *k* of the algorithm's main *while* loop reaches value -1, which means that all possibilities had been iterared and all solutions found.

For the second case, the strategy is somehow reversed. Now we cannot iterate through the conjuncts of a rule in order to find a fact for each one that matches it because facts are also atoms conjuncts. In this case we iterate through the set of facts and lookup into the conjuncts array of the rule to see if there exists a subset that matches the fact over the above 3 conditions. The process halts after all rule's conjuncts had been matched, so a solution has been found and the fact from the head is subsequently deduced. This strategy also relies on the Backtracking algorithm, so as the former, but now the space of possible solutions and the result vector are interchanged.

3.5. Experiments and Results

The proposed derivation system (both versions) has been tested on three types of knowledge bases: small, medium and large. The small one is units order (<10 elements in each set), the medium is tens order and the latter has hundreds order.

My optimized solution has been compared with the most basic one in terms of operations made and computation times. This basic solution works by making combinations of subsets of facts of length k (where k is the number of rule's atomic conjuncts) from the entire set of facts until when finds all subsets of facts that match the rule on all 3 conditions. The results of the experiments are presented in 2 tables, one for each version of the algorithm.

The tests have been realized on a Toshiba Satellite computer running at inside an Intel Core 2 i3 of 2.15GHz, 3GB DDR3 and 300GB hard-disk space on which was installed a 32 byte Windows 7 OS. Worth saying that the obtained computation results may vary with multiple factors, such as the OS type, CPU architecture, I/O access time, and many other elements.

Solution	**Knowledge Base Type**					
	Small		**Medium**		**Large**	
	NrOps	Time(ms)	NrOps	Time(ms)	NrOps	Time(ms)
Our	15840	10	2556000	690	444072000	3940
Basic	41050	30	4558099	950	1880432151	7950

Table 3.1: Experimental results (vers. 1)

Solution	Knowledge Base Type					
	Small		Medium		Large	
	Nr.Op	Time(ms)	Nr.Op	Time(ms)	NrOps.	Time(ms)
Our	17200	11	3507890	790	990548203	6200
Basic	51200	39	5007990	990	3077900330	10890

Table 3.2: Experimental results (vers. 2)

3.6. Conclusions

In this chapter was proposed an object-oriented reasoning system that is used in the scope of obtaining the saturation of any type of logical knowledge base.

A logical knowledge base is a triple (F,R,N) composed from a set of facts, rules and negative constraints. The main goal was to implement a derivation algorithm (reasoning system) that works by successively applying the set of rules over the factual part of the base in order to deduce new knowledge from the ones explicitly stated while in the same time taking care not to violate any negative constraint stated in the set N. This thing is known in the classical logic as the Modus-Ponens inference. The process continues upon achieving saturation of the knowledge base, i.e. new facts cannot be produced by application of rules over the factual part.

The concrete implementation of the algorithm was realized into the object-oriented language Java. The knowledge base is constituted from collections of objects instance of classes. The algorithm takes as input the knowledge base sets (F,R,N) and produces as output a new (saturated) set of facts. The algorithm relies on Backtracking technique in order to find all possible solutions to the problem (all sets of facts that match a rule's conjuncts) and it is a breadth-first search with forward chaining. Experiments with the system have been performed over three types of knowledge bases that vary in the size (order) of the sets: small (unit), medium (tens), big(hundreds). Had been provided the computation times of the algorithm in each of the cases. The proposed optimized solution has been compared with another primitive one that works by searching all combinations of facts that match a rule. The results comparisons are shown in tabular form for each type of logical KB.

The chapter presented also a state-of-art from the domain of logical reasoning systems, being discussed the frameworks from the Java language that were created in this purpose, such are JRules, Drools, RuleBook, but I explained that none of these systems is not capable to perform the task of my proposed system, which is the production of saturation of a knowledge base. Due to that I designed my system from scratch in this purpose, starting with OO data structure, the algorithm for matching rules on facts and the representation of the knowledge in the syntax of First-Order Logic (FOL). Another reason for which I cannot compare my work here with others is that the proposed system has been specifically created for the needs of an institution from France where I worked during the course of an Erasmus stage near the universityţs staff there, so all this work represent our own ideas, solutions, contributions.

The project of the implemented system can be downloaded from my Drive account at address: https://drive.google.com/open?id=1QzbIogncFL-b-zymGP2JhtXtQN9ZQwDT .

101 *page intended left blank*

BUPT

4. INTRUSION DETECTION AND PREVENTION SYSTEMS

This chapter constitutes the start of the second part of this thesis in which will be discussed the use of Semantic Web technologies in the construction of cyberdefence systems. In this chapter I will conduct a study in the fields of cybersecurity and systems for detection and prevention of computer attacks in order to present the reader some of the most important aspects, like elementary notions, domains where they find most applications, phases of evolution, detection methodologies, typical architectures and security capabilities. The discussion will subsequently follow with a class of IDPSs that will later be used during this thesis, that are those for networks, and everything that was presented until then about IDPSs generally speaking will be particularized for this case, such as architectures and components, security capabilities, management capabilities, and many others. Chapter's contributions are the models in the form of diagrams and figures that had been created in Adobe Photoshop 3.0, and that present graphically the notions that are detailed in text, such as IDPS architectures with their main components, IDPS components deployment inside a security infrastructure for the networks of an organization, etc.

4.1. Introduction to Computers Security

Computers Security (also known under the names of Cybersecurity, IT Security) is the science that deals with protection of computer systems for not being theft or created damages at various of their architectural levels: hardware, software and/or data within, as well as perturbation or unauthorized use of their services. Include control of physical access to hardware, protection against attacks that derive from the network access, bad use by the operators deliberately or accidentally. This area of science knew evergrowing importance due to the fact that it relies on the Internet services and computer networks of the society (e.g. WiFi, Bluetooth) and also due to the growth in popularity of smart devices, like mobile phones, television, devices from Internet of Things etc. [138].

Computer Security is a science of critical importance in almost every industry that relies on computing equipment. Today most electronic devices (PCs, laptops, mobile phones) have in their implementation software called firewalls, but these do not make them 100% secure against attacks. There are many methods by which a computing system can be attacked: network-based, files download from unauthorized sites, connecting to unauthorized WiFi networks, resource consumption, electromagnetic radiation etc. These may be protected by using some very reliable hardware and software components. Having some strong internal interactions of proprieties, software complexity may stop the security errors and software fails.

BUPT

The most important industrial domains that need protection against cyber-attacks, as it was stated by sources from Wikipedia are [221]:

1) Financial/Banking systems: Web sites and applications that store information about credit cards, bank accounts or brokering represent the most prominent targets for attackers caused by the huge and fast gaining potential (money transfer, shopping, information selling etc).
2) Aviation: aeronautics industry is constituted by a multitude of complex systems that present all high vulnerabilities to attacks. A blackout at the airport may cause damages, radio transmissions on which the aircraft relies on may be jammed, and the radio signals control over seas and oceans is hard to support because surveillance goes only 200km offshore.
3) Automotive: if it is being gained unauthorized access to the network of the car's control zone many things can happen. Computerized timing of the engines, road control, brakes anti-locking, doors locks, airbags, driver assistance systems makes those damages possible, and in case of automated pilot cars these damages could climb even higher. Automotive information security doesn't concern only the production but also discovery, measuring and packing the vulnerabilities.
4) Industrial equipment: many of the industrial equipment functionalities and utilities are being controlled remotely by the computers, such as the communication coordination, power grid, opening and closing of the valves from the water and gas pipes. These machines can be hacked by means of the local radio communications or the Internet.
5) Internet of Things and the physical vulnerabilities: Internet of Things (IoT) represents the network of real world (physical) objects embedded with electronic equipment, sensors, software and network connectivity to the network that allows them to be controlled from distance and exchange data between them [182]. Even though it offers possibilities to the integration of real world into the computing systems, it also provides opportunities to the attackers. Cyberattacks are very likely to become an increasing threat. In this idea, if a door's lock is connected to the Internet and can be controlled by means of the mobile phone, then an infractor can burglar the house only if he steals that phone. People are prone to lose more than their credit cards in a world controlled by IoT devices [79].

According to the security guide provided by NIST [169], intrusion detection is the process of monitoring the events that arise into a single system or network of computers and their analysis in order to observe possible signs of incidents, such as violations or threats to the security policies, to the acceptable uses or to the standard security practices. Incidents can be of multiple types, like malware (worms, spies), gaining unauthorized access to systems, authorized users that erroneously use their privileges or that try to obtain other unauthorized ones.

Even though the big part of the incidents have a malicious nature, there exists other benign ones, such would be the case when a person misspells from the keyboard the address of a system and accidentally tries to connect to that host without holding authorization.

An Intrusion Detection System (IDS) is a software that automates the intrusion detection processs. An Intrusion Prevention System (IPS) is a software that holds all capabilities of an IDS and besides are added the attempts to try stop the incidents that are being detected [169].

In this chapter I will make an introduction, as broad as short as possible, onto the IDS and IPS systems. For abbreviation reasons, from now on in this thesis I will use the term ”IDPS” in order to refer to both technologies.

Due to the increasing dependency on the information systems and the prevalence and potential impact of intrusions on these, IDPSs have become a necessary addon for the security infrastructures in every organization. IDPSs are especially focused on incidents identification, like is, for example, when an attacker succeeded in compromising a system by exploiting one of its vulnerabilities. The IDPS then reports the incident to the security administrator that will rapidly initiate the response actions for the incident in order to minimize the damages that were produced. Also the IDPS can log information that can support the persons who deal with handling of the incident. Many IDPSs can be configured to recognize violations of security policies. For example, some systems may be configured with settings similar to those of the firewalls rules, allowing them to identify traffics from networks that violate the policies of security and acceptable use of organizations. Other IDPSs may monitorize file transfer and identify those that seem suspect, such as it would be the copying of a large database onto the machine of a user. Many IPDSs can also identify reconnaissance, which precedes the imminence of an attack. For example, some tools of attacks and forms of malware (especially worms) make reconnaissance activities (such as ports and hosts scan) in order to identify the targets of subsequent attacks. An IDPS must be capable to block reconnaissance and notify security administrators for making concrete actions, such as changing other security controls in order to prevent similar incidents. Since reconnaissance is something common on the Internet, detection is often performed within internal protected networks [169].

Other than the ones presented above, other uses of IDPSs found by the organizations are:

- identification of problems from security policies: an IDPS can provide a certain degree of control of the quality for the implementation of security policies, such is the duplication of firewalls’ rules sets and alerting the administrator about observing the network traffic that should have been blocked by the firewall but wasn’t due to a configuration error
- documentation of existing threats of the organizations: IDPSs log information about the detected situations. Understanding of frequency and characteristics of the attacks against the resources of an organization is an important thing for the identification of corresponding security measures for the protection of resources. Information can also be used to educate the management regarding the threats with which the organization deals

BUPT

- blocking the individuals from violating the security policies: if individuals are aware that their actions are being monitored by the IDPS technologies for possible violations of security policies then they would not commit such crimes due to the risk of being caught
- some IDPSs are capable to modify their de security profiles when a new threat is being detected. For example, an IDPS may be capable to gather multiple details about a particular session after some malicious activity has been detected within that session. It may also modify settings about the moment when some alerts are fired or what priorities must be assigned to subsequent alerts after a certain threat has been detected.

IPS technologies differ from ones of IDS by one big characteristic: they are capable to respond to a threat by trying to stop it from the attempt to succeed. Use several response techniques, such as [169]:

- stopping the attack: this can be done in one of the following ways: termination of network connexion or of user session used by the attacker, blocking the access from attacker's account to the target, such as the IP addresses or other attributes of the attacker
- changing the security environment: the IPS is capable to change the configuration of other security controls in the attempt to block the attack. Among the most well-known examples for this task are: reconfiguration of a network device (e.g. switch, router, firewall) to block the attacker's access, changing the host firewall of a target in order to block all subsequent attacks. Some IPSs may even apply patches on a host if the IPSs detects that the host has vulnerabilities
- modifying the content of the attack: some IPS technologies may delete or replace malicious parts of an attack in order to make him benign. A classical example is when an IPS deletes an infected file that is being attached to an email then allows the email to reach its destination. A more complex example is an IPS that acts as a proxy and normalizes the incoming requests, which means that the proxy repatches the content of the requests and removes the information from headers. This may lead to certain attacks being removed as part of the normalization process.

Another characteristic that is common to IDPS technologies is that these cannot provide a 100% accurate detection. The situation when an IDPS erroneously classifies a normal activity as being malicious is called a *false positive*. When the IDPS does not identify a malicious activity is called a *false negative*. It is not possible to eliminate all false positives and negatives, in the majority of cases the decrease of ones has the effect of increasing the others. Many organizations choose to shrink the rate of false negatives at the expense of the growth of positive ones, a fact which translates in that many malicious activities are identified but are required multiple analysis tools in order to differentiate false positives from the real malicious events. The process of changing the configuration of an IDPS with the goal to improve detection accuracy is called *tuning* [169].

The great part of IDPS technologies contains additional characteristics that compensate the use of evasion techniques. *Evasion* means the changing of the format or times of malicious activities such that their appearance changes but the effect stays the same. Hackers use evasion techniques for preventing the IDPS technologies to detect their attacks. For example, a hacker encodes text characters in a certain way, knowing the fact that the target understands the encoding and hopes that any monitoring IDPS doesn't. Many IDPS technologies can prevent the common evasion techniques by replicating the special processing that is being made by the targets. If the IDPS can see the activity in the same way as the target does then the evasion would not succeed in hiding the attacks [169].

Domain literature states that IDSs went through multiple phases of evolution until the present day [214], those being:

- attacks signatures
- attacks taxonomies
- attacks ontologies

The first types of IDSs relied on attacks signatures, which are some syntactic representations of them. This technique is not a very efficient one for multiple reasons, such would be the signatures have a generic nature, relies on some languages that are specific to some particular domains and depend on some specific environments and systems. Due to these causes, they lack extensibility and are not well fitted for communication in heterogeneous environments. Attack signatures contain vague semantic information and lack a solid base for any formal logic, the smallest variation in business logic invalidates them.

The second phase of IDS evolution is represented by the use of taxonomies. As central components in IDS's functionality, taxonomies have the goal to characterize and classify information of attacks, and a language for describing the instances of concepts [202].

The current phase in the evolution represents the use of Semantic Web technologies, such as ontologies, distributed agents, data mining, rules etc. Cybersecurity systems that are created by means of ontological technologies represent a new line of defense that are able to detect sophisticated attacks and even 0-Day (previously unseen) due to their abilities to capture the context of information and filter them by certain criteria. Various generic security controls, such are signature-based firewalls, intrusion detection and prevention systems, cryptography devices have been proposed, but their efficacy against Web threats is restricted due to the great rigidity they have. For obtaining efficient mitigation and blocking of the attacks the system should understand the context of information that it processes and filter the contents based on their effects on the target application. For these reasons security frameworks that rely on ontologies are used in this kind of situations [163].

An ontology is an explicit specification of the conceptualization of an application domain that captures its context (the interpretation of words from a specific domain). Ontological models are flexible in defining the concepts to the level of detail that is desired, easily extendible and provide inference capabilities in order to allow making reasoning over the instances of data. Artificial Intelligence and Semantics fields rely on formal ontologies for the sharing and reusing of knowledge among different software entities [199].

BUPT

The most important classes of detection methodologies that are being used in IDPS technologies, as it has been stated by the NIST guide in [169], are:

i. signature-based detection
ii. anomaly-based detection
iii. stateful protocol analysis

Most IDPSs make use in their implementation of multiple methodologies, either individual or combined, in order to provide a wider range and more accurate detection.
In the rest of this section each of these methodologies will be taken and briefly presented their characteristics.

4.1.1. Signature-based Detection

A signature is a pattern that corresponds to a known attack. Detection based on signatures is the process of comparing the signatures against observed events in order to identify possible signs of incidents. Examples of signatures are: a Telnet attempt with username 'root' (which is a violation of a security policy), an email with subject "Free pictures" and a file in attachment with the name 'freepics.exe' which are the characteristics of a known form of malware, a record from the log of an operating system having the status code 645 which says that the audit of that host has been deactivated.

Signature-based detection is very efficient at detecting attacks that are already known, but it is practically useless in case of those before unseen, the 'disguised' ones by means of evasion techniques, and many variants of the known ones. For example, if an attacker modified the malware from the previous example to use a file named 'freepics2.exe' then the signature would not recognize it anymore [163].

This is the most simple detection method since it does not do anything than compare the current activity unit, like a package or a record from a log, with a list of signatures relying mainly on string comparison operations. Signature-based detection techniques have small understanding about the application and network protocols and can not keep track or understand the state of complex communications. For example, they cannot pair a request with the corresponding answer, such as to know whether a request to a Web server for a certain page generated a response with status 403, which means that the server refused to fulfill it. They also are not capable to remember the previous requests when working on the current ones. These limitations prevent the signature methods to detect attacks that span multiple events if none of them does not contain a clear indication regarding an attack.

4.1.2. Anomaly-based Detection

Anomaly-based detection is the process of comparing the definitions of activities that are considered normal against observed events in order to observe significant deviations. An IDPS that uses this type of detection holds profiles that represent normal behaviors of some things like users, hosts, network connexions or applications. Profiles are being built by monitoring the characteristics of activities that are considered normal over a certain period of time. For example, the profile for a network may indicate that Web activities contain in average 13% bandwidth at the boundary with Internet during the main working hours. The IDPS uses then statistical methods to compare the activity currently being observed with the thresholds found for the profile, such would be to detect when the Web activity

consumes much more bandwidth than normal and to alert the security administrator about the anomaly. These profiles may be created for multiple attributes of the behaviors, such as the number of emails sent by a user, the number of failed login attempts to a host, the degree of processor usage on a host in a certain period of time.

The main benefit of this technique is that it can be very efficient for detecting attacks unknown until then. For example, let's assume that a computer infects with a new type of virus. This virus consumes the resources of the computer, sends many emails, initiates numerous network connexions and does other things that are very much deviated from the profiles established on that computer.

An initial profile is created over a period of time (days, weeks) which is named 'training period'. Anomaly-based detection profiles may be static or dynamic in nature. Once it has been created, a static profile remains unchanged only if the IPDS is given a command to generate another one. A dynamic profile is being constantly adjusted as events are being observed. Because systems and networks change regularly over time, corresponding measures of normal behavior also changes. A static profile will possibly become inaccurate and is needed to be periodically regenerated. Dynamic profiles do not exhibit this problem but are prone to evasion techniques. For example, an attacker can occasionally realize small malicious activities, then increase their frequency. If the rate of change is sufficiently small the IDPS may think that the malicious activity is just a normal behavior and include it in the profiles. Malicious activities can also be observed by the IDPS during the creation of the initial profiles [169]. Inadvertently including of malicious activities as part of a profile is one of the common problems of IDPSs that rely on this technique. In some cases administrators can modify the profiles in order to exclude activities which are known to be malicious. Another problem with regarding the construction of profiles is that, in some situations it can be very difficult to be made accurately due to the complex nature of required computations. For example, if a particular maintenance activity that perform large files transfers occurs only once per month may not be observed during the training period. When maintenance occurs it will be considered a deviation from the profile and will be triggered an alert. Anomaly-based IDPS products often exhibit many false positives due to the benign activities that deviate from the profiles, especially in diversified or dynamic environments. Another major problem of the anomaly techniques is that is being often difficult for analysis to find out why a certain alert has been triggered and to validate that an alert is benign and not a false positive due to the complexity of computations and number of events that may have been caused that alert.

4.1.3. State Protocol Analysis

State Protocol Analysis (SPA) is the process of comparing the predetermined profiles of the general accepted definitions of benign protocol activities for each protocol state against observed events. In contrast to anomaly-based detection, which uses profiles that are specific for hosts or networks, SPA uses universal profiles created by providers which specify how (and how not) the protocols must be used. The term "state" from its name composition means that the IDPS is capable to understand and trace the state of the protocols for network, transport and application that contain a notion about state. For example, when a user starts a FTP session, this initially is in the unauthentified state. Unauthentified users must be capable to do little staff in that state, such would be the visualization of help information or creating user names and passwords. An important part related to the

BUPT

understanding of state is pairing the requests with responses such that when a FTP authentication attempt occurs, the IDPS can determine if it is successful by looking after the state code in the response. After the user is successfully authentified, the session is in authentication state and users can do multiple commands. Realization of these commands while being in the 'unauthentified' state would have been considered suspicious, but in 'authentified' state is considered benign [169].

Stateful protocol analysis may identify unexpected command sequences, such as repeatedly performing the same command, or initiation of a command without making another one on which it relies. Another state tracing characteristic of the SPA is that, for protocols that perform authentication, IDPS can keep trace of the authentifier used for each session and records the one used for suspicious activities. This is useful when an incident is being investigated. Some IDPSs can also use the authentifier's information in order to define activities that are variously accepted by mutiple classes of users or for some specific ones.

The protocol analysis performed by the SPA methods usually includes checks of the reasonableness of individual commands, such is the minimum and maximum length of arguments. If a command usually has as an argument a user name and these have a maximum length of 20 characters then an argument with length 1000 characters is suspicious. If the large argument contains binary data then it is being even more suspect. SPA methods rely on models that are generally based on well-defined protocol standards of software providers or organizations of standards (e.g. Internet Engineering Task Force [IETF], Request for Comments [RFC]). The protocol models generally take into consideration variations from each implementation of the protocol. Many standards are not exhaustively complete in explaining the details of protocols, fact that causes variations among implementations. Also, a big number of providers either violate the standards or add some particular features, some of them can change the standards' traits. For particular protocols, complete details about protocols often are not available, thing that makes difficult for IDPS technologies to realize a comprehensive and accurate analysis. Because protocols are being reviewed and suppliers change their implementations, the IDPS protocols models should be updated in order to reflect these changes [169].

The main drawback of SPA models is that they are resource intensive due to the complexity of analysis and overuse required in state tracing for multiple simultaneous sessions. Another big problem is that these methods can not detect attacks that do not violate the characteristics of the behaviors generally accepted by protocols, such is the realization of a big number of benign activities in a short period of time that could cause a denial of service (DoS). Another still one is that the model of protocol that is being used by an IDPS may contrast with the way protocol is implemented in particular versions of applications or operating systems, or how different implementations of clients and servers of the protocol interact.

4.2. Components and Architectures for IDPSs

In this section I will make an overview of IDPS technologies that are found in almost any type of IDPS products. Additional information specific for each type of product will be provided in the next 4 sections. In this section is made a high-level discussion of the general architectures of IDPS systems together with their components, are being presented security capabilities, such as methodologies they use at identification of malicious activities. Also, in the end of the section will be recalled some notions about management capabilities of the technologies.

The general architecture of an IDPS solution is constituted from the following basic components:

1) Sensors (Agents): components for monitoring and analysis of activities. The term *sensor* is used in case of network IDPSs that utilize technologies of networks, wireless or network behavior analysis (as they will be presented in next sections). Term *agent* is used in the case of host IDPSs.
2) Management server: a centralized device that receives information from sensors and manages them. Some management servers do analysis on the event information and can identify activities that individual sensors are not capable to. Matching event data from multiple sensors (such would be finding the events fired from the same IP address) is called *correlation.* Management servers are available under two forms: appliances and also as software. Some deployments of small IDPS do not use management servers at all. In large IDPS deployments may exist more than one management server, while in some cases may be organized on multiple levels.
3) Database server: a database server is a warehouse for storing event information that has been recorded by sensors. Many IDPSs provide support for these components.
4) Console: is the interface component of IDPS that allows the use also by normal users and administrators. Software for console is usually installed on desktop or laptop computers. Some consoles are being used only in administrative purposes, such is the configuration of sensors or applying software updates, while others strictly for monitoring and analysis. Others offer both types of capabilities.

BUPT

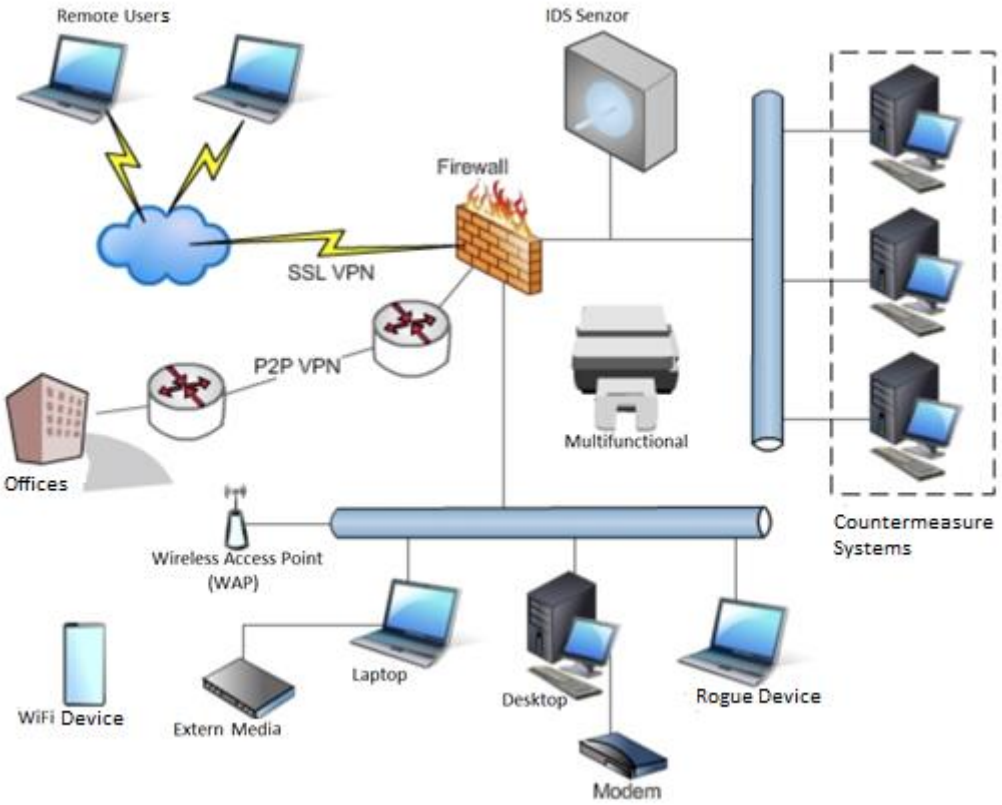

Fig.4.1: Architecture of an IDPS and component placements

IDPS components may be connected one to another by means of the networks of an organization or other network especially designed for security software management, called *management network.* If is being used this latter one, every sensor host has an additional network interface called *management interface* that connects to the management network. Also, each sensor host can not transmit any traffic between its management interface and any of its other network interfaces. As it can be seen in fig. 1 and 2, the most common placements for sensors are:

- between the organization's network and Extranet
- inside the DMZ Firewall to identify possible attacks to the DMZ servers
- between firewall and network to identify threats in case of firewall penetration
- in the Remote access environment
- between servers and users' community in order to identify attacks from the inside
- on the Intranet, FTP, or database environment

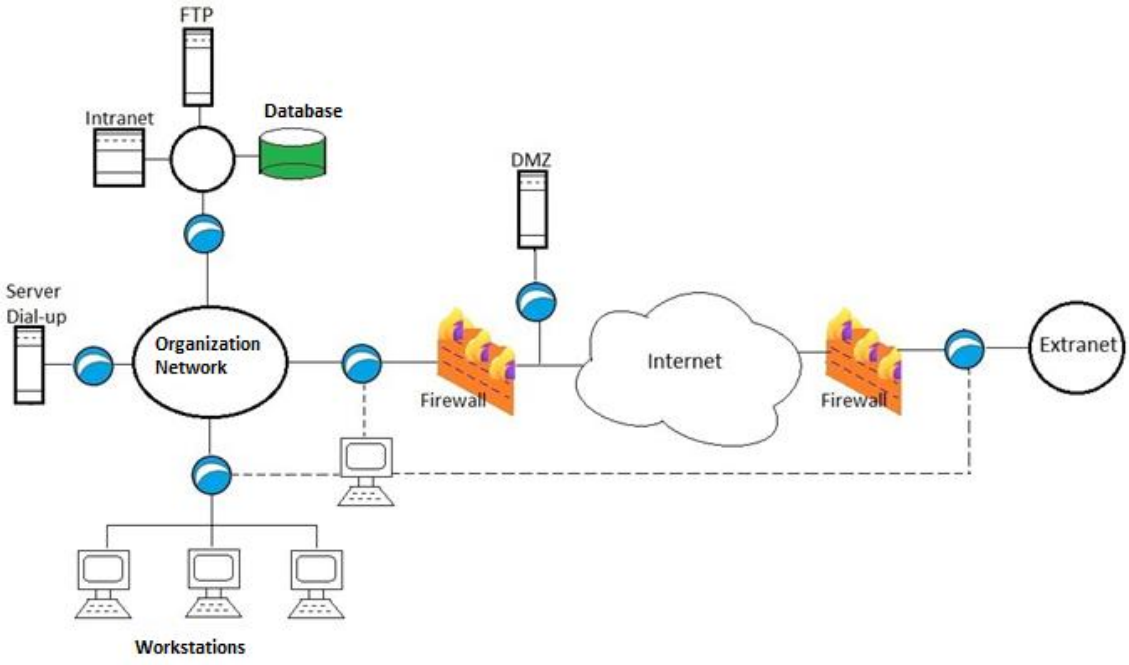

Fig.4.2: Sensors placement inside a network secured by IDPS

The idea is to establish the organization's network perimeter and identify all entry points. Once found, the IDPS sensors can be installed and must be configured in order to produce reports at the management console. Administrators will log to the console and will arrange the sensors, provide new signatures and review the logs.

Management servers, database servers and consoles are attached only to the management network. This architectural style efficiently isolates the management network frm the ones for production. The pros of this fact are that they hide the existence and identity of IDPS from hackers, protecting it from the attacks, and also ensuring that the IDPS has sufficient bandwidth in order to properly work under critical conditions, such as in the case of attacks with worms or distributed DoS over the monitored networks. The disadvantages include additional costs required for purchasing the network equipment and the hardware, and inconveniences for the users and administrators of the IDPS to use separate machines for management and monitoring.

If an IDPS is being deployed without a separate management network, another way to improve its security is to create a virtual one by means of a VLAN inside the standard networks. Use of a VLAN offers protection for IDPS communication but not as great as that provided by a separate management network. For example, a wrong configuration of a VLAN may lead to the exposure of IDPS's private data. Another concern is that, under adverse working conditions, such as DDoS attacks or major malware incidents, network devices separated by the main networks and organization's VLANs may become completely saturated, fact that will negatively impact over the performance and availability of the IDPS.

4.3. Security Capabilities

The most important security capabilities of IDPS technologies are, as it was stated in the NIST guide [169]:

- information gathering
- logging
- attacks detection
- prevention

4.3.1. Information Gathering

Some IDPS technologies offer information gathering capabilities, such as collecting information from a host or network regarding the monitored activity (for ex., identification of hosts, operating systems and applications that are being used, identifying of general network traits).

4.3.2. Logging

IDPSs usually perform extensive logging activities of the data regarding the observed events. These data are being used to confirm the validity of alerts, investigate incidents or correlate events among IDPS and other logging sources. The fields of data commonly used by the IDPS include the date and time of the event, type of the event, importance rate (such as priority, severity, confidentiality, impact) and the prevention action that was (eventually) being taken. Some specific types of IDPSs log also other data fields, such are those for networks that perform packet captures, and the ones for hosts that record users' IDs. IDPS technologies normally allow administrators to store logs locally and to send copies to centralized servers, such are software for information security and events management. In general, logs are being stored both local and central in order to support data integration and availability, such as, for example, an IDPS fail may allow attackers to corrupt its journals. Similarly, IDPSs should have synchronized clocks using the protocol Network Time Protocol (NTP) or by manual adjustments in order for the data in their logs to have correct time stamps.

4.3.3. Detection

IDPS technologies should offer a wide array of detection capabilities. Most of the products use a combination of the techniques, fact which helps to achieve a more accurate detection and a bigger flexibility for tuning and customization. The types of detected events and the detection accuracy greatly vary with the used IDPS technologies. The majority of IDPSs require small tuning and customization in order to improve their detection accuracy, usability and efficiency, such is setting the prevention actions to be realized only for certain particular alerts. Technologies vary greatly in regard to the tuning and customization capabilities. As stronger these capabilities for a product are as more the detection accuracy can be improved compared to the initial configuration. Organizations should pay much attention to the capabilităţile of tuning and customization of the IDPS technologies when evaluating products.
Examples of these capabilities, as it was presented by the NIST guide [169], are:

BUPT

i) *thresholds:* a threshold is a value that establishes a boundary between what represents a normal and an abnormal behaviour. Generally, it specifies a maximum acceptable level, like *x* failed connection attempts in 60 sec, or x characters for the length of a file name. Thresholds are most often used in anomalies detection and SPA

ii) *white lists* and *black lists:* a black (or 'hot') list is a list of discrete entities, like hosts, TCP and UDP ports, ICMP types and codes, users, applications, URLs, file names and extensions that were previously determined to be associated with malicious activities. These lists are generally used to allow IDPSs to recognize and block activities that are very likely to be malicious or to assign a greater priority to alerts that match with records stored in the black list. Some IDPSs generate dynamics black lists that are being used to temporarily block recently detected threats, such as activities from the attacker's IP address. A white list contains discrete entities that are considered benign. These are generally used on a granular basis to reduce or ignore false positives that imply known benign activities from trusted sources. White lists and black lists are most often used in signatures detection and SPA

iii) *alert settings:* the great part of IDPS technologies allows administrators to customize each alert. Examples of actions that are done for an alert are: its starting and stopping, setting of an implicit priority or severity level, specifying of what information should be recorded and what notification methods used (email, sms, pager), specifying of what prevention capabilities should be used. Some products halt the alerts if an attacker generates many of them in a short period of time, or also can temporarily ignore the entire subsequent traffic from the attacker; this is to prevent the IDPS for being overloaded with alerts

iv) *code visualization and editing:* some IDPS technologies allow admins to see a portion or the complete detection code. This is in general limited to signatures, but some technologies allow to be seen also other parts, like the programs used for performing stateful protocol analysis. Visualization of code may help analysts to determine why certain alerts had been generated, to validate the alerts and identify false positives. The capability to edit the detection code and to write another is necessary for complete customization of certain types of detection capabilities. For example, a certain alert may be generated by a complex series of events that imply multiple modules of code. IDPS customization in order to understand the organization's specific characteristics may not be possible without the apriori editing of code. Code editing is an operation that requires both programming and intrusion detection knowledge, and some IDPSs use proprietary languages which requires the programmer to learn it. Bugs that are introduced in code during the customization process may lead to the incorrect functioning of the IDPS or even to its failure, so administrators

BUPT

v) should treat code customization as any modification of the production systems code.

Administrators should review the tuning and customization operations periodically to ensure that they are still accordingly. For example, while lists and black lists must be regularly verified and all records validated in order to ensure that they are still necessary and correct. Thresholds and settings for alerts may require a periodic adjustment in order to compensate for changes from the environment and from threats. Edits made over the detection code may require replication when the product is being updated (e.g. put patches, upgraded). Administrators should also ensure that any products that gather references in the anomaly-based detection have the reference levels periodically redone depending on the needs in order to ensure an accurate detection.

4.3.4. Prevention Capabilities

Most IDPSs offer multiple prevention capabilities that vary with the technology. IDPSs usualy allow administrators to specify the configurations for the prevention capabilities of each type of alert. This includes prevention activation or deactivation, or the specification of which capability must be used. Some IDPS sensors have a learning or simulation mode that halts all prevention actions and, in exchange show when these would have been performed. This allows administrators to monitor and make fine-tune of the prevention capabilities before activating the prevention actions, which reduces the risk of blocking by mistake of benign activities.

4.4. Network IDPS Technologies

4.4.1. Components and Architectures

In this section I will make a discussion regarding IDPS technologies for networks. Firstly will be mentioned the main components of IDPS systems for networks and explain the architectures used for their deployment. Next are examined the security capabilities, such as methodologies for identifying unusual activities.

A network IDPS monitors the traffic from networks for certain segments or devices, analyze the network, transport and application protocols in order to identify suspicious activities. For readers that are not familiarized with the notions of networking, such as would be the TCP/IP protocols stack, the ISO/OSI standards, are invited to read the articles of Ahlawat in [6], [7], [8] where are described briefly these elementary notions.

A network IDPS consists of sensors, one or multiple management servers, multiple consoles and databases (optionally). A sensor for network IDPS monitors and analyzes network activities in one or multiple segments. Network interface cards (NIC) that will perform the monitoring are set in the promiscuous mode, i.e. they will accept all packets will see. Big majority of IDPS deployments use multiple sensors, those from the industrial domain have hundreds of sensors. Sensors are available on the market in two forms:

BUPT

- *devices*: this class of sensors consists of a specialized hardware and its corresponding software. Hardware is optimized for use as a sensor, such are specialized NICs and their drivers for efficient capture of packets, or specialized processors and other hardware components that have a role in the process of analysis. One part, or we can say all the software of IDPS can be stored inside the firmware for an increased efficiency. Devices often rely on a customized operating system, hardened, on which administrators can not directly access
- *software*: there exist sensors only in the form of software (no devices). Administrators install the software on hosts that meet certain specifications. Software sensors can include a personalized OS or can be installed on a generic one, same way as a normal application. Organizations should think about how to use management networks for the deployments of their network IDPSs anytime this fact is possible. If an IDPS is being deployed without a separate management network then organizations should consider if is necessary a VLAN or not for protecting the communications of the IDPS [169].

Besides the choice of the network components, administrators should choose also the location of sensors of the IDPS. Sensors can be deployed in two modes: inline or passive.

An *inline* sensor is deployed such that the network traffic that it monitors will pass through it, in a way similar with the flow of traffic associated to a firewall. Some inline sensors are actualy firewall/IDPS hybrid devices, while others are simple IDPS. The main reason for IDPS sensors deployment in the inline mode is to allow them to block attacks by ceasing the network traffic. Inline sensors are usually placed in the places in which firewalls and other security devices of the network are, and that is generally at the borders between networks.

BUPT

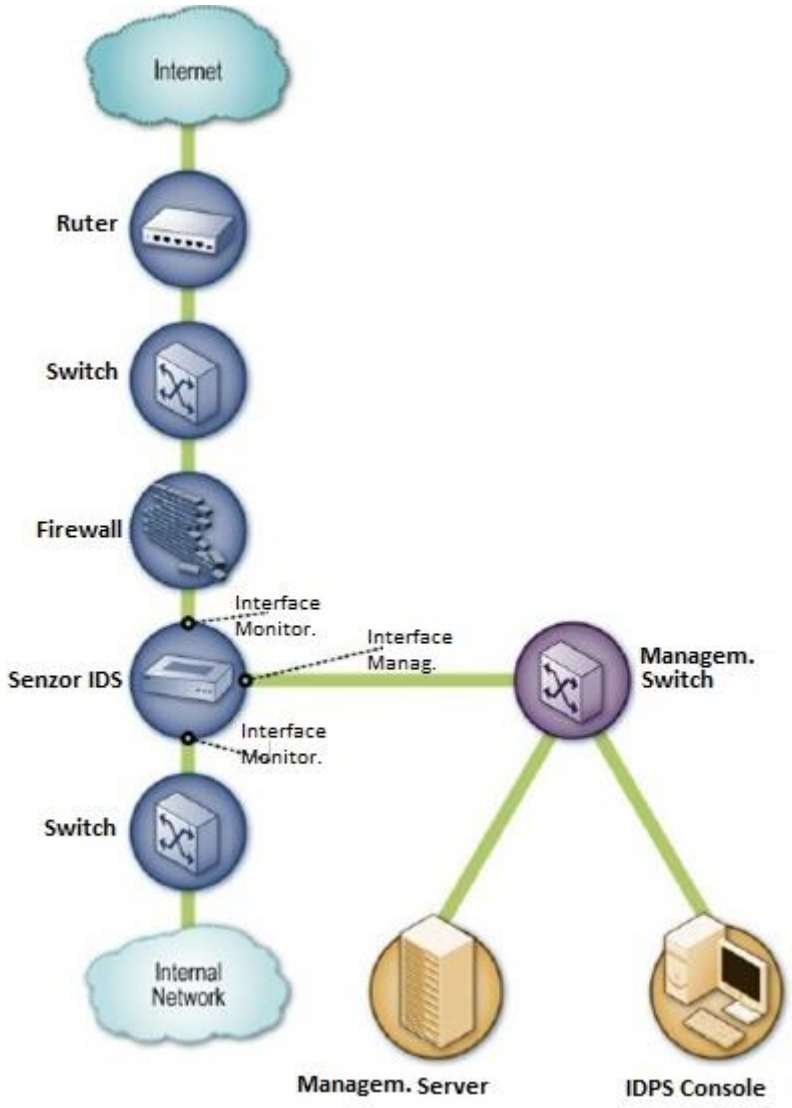

Fig.4.3: Typical example of NIDPS architecture with inline sensors

Inline sensors that aren't hybrid firewall/IDPS devices are often deployed on the more secured side of a network's division in order to have less traffic to deal with. In fig.3 is shown such a deployment of sensors. Sensors are placed on the less secured parts of networks divisions in order to offer more protection and reduce the workload on the division device, such as a firewall.

A *passive sensor* is deployed in order to monitor a copy of network traffic, nothing actually goes through him. Passive sensors are generally deployed to monitor key network locations (such as borders between networks) and the key segments, like the demilitarized zones (DMZs).

Passive sensors can monitor the traffic through various methods:

- *spanning port:* most of the switches rely on a spanning port, that is a port capable to see all the traffic of the network which passes through the switch. By connecting a sensor to a spanning port can allow this one the monitoring of traffic that enters and outs from many hosts. Even though this monitoring method is straightforward and cheap, trouble could still arise, such as when a switch is incorrectly configured the spanning port could not be capable to see all traffic, or that their usage is resource-intensive. By the way, many switches have only one spanning port and often it is needed to have multiple technologies, such as tools for network monitoring, forensic analysis, or other IDPS sensors to monitor the same traffic.
- *network taps:* a network *tap* is a direct connection between a sensor and the network's physical environment, such as wires, optical fiber, etc. The tap gives the sensor a copy of the entire traffic that has been worn onto that physical environment. The installation of a network tap generally implies a temporary blackout of the network and possible problems with the tap could prolong the downtime. In contrast to the spanning ports, network taps must be acquired as supplements for networks
- *balancer for IDPS load:* a load balancer for the IDPS is a device that aggregates and directs the network traffic towards the monitoring systems, such are the IDPS sensors. Receives copies of the network traffic from the spanning ports and network taps and combines the traffic from multiple networks. Distributes then copies of the traffic to one or multiple listener devices, including IDPS sensors, relying for that on a set of rules created by the administrator. These rules tell the balancer what traffic to send at each type of device, such as would be to send all the traffic to many IDS sensors, divide the traffic dynamically among many IDPS sensors based on the volume, divide the traffic among multiple IDPS sensors based on the IP addresses, protocols or other traits. Dividing the traffic among multiple IDPS sensors could cause a reduction in detection accuracy if similar events or parts from a single event are seen by different sensors. For example, let's assume that two sensors see different phases of the same attack; if each step is considered to be benign but the sum of steps overall is malign the attack could not be recognized.

BUPT

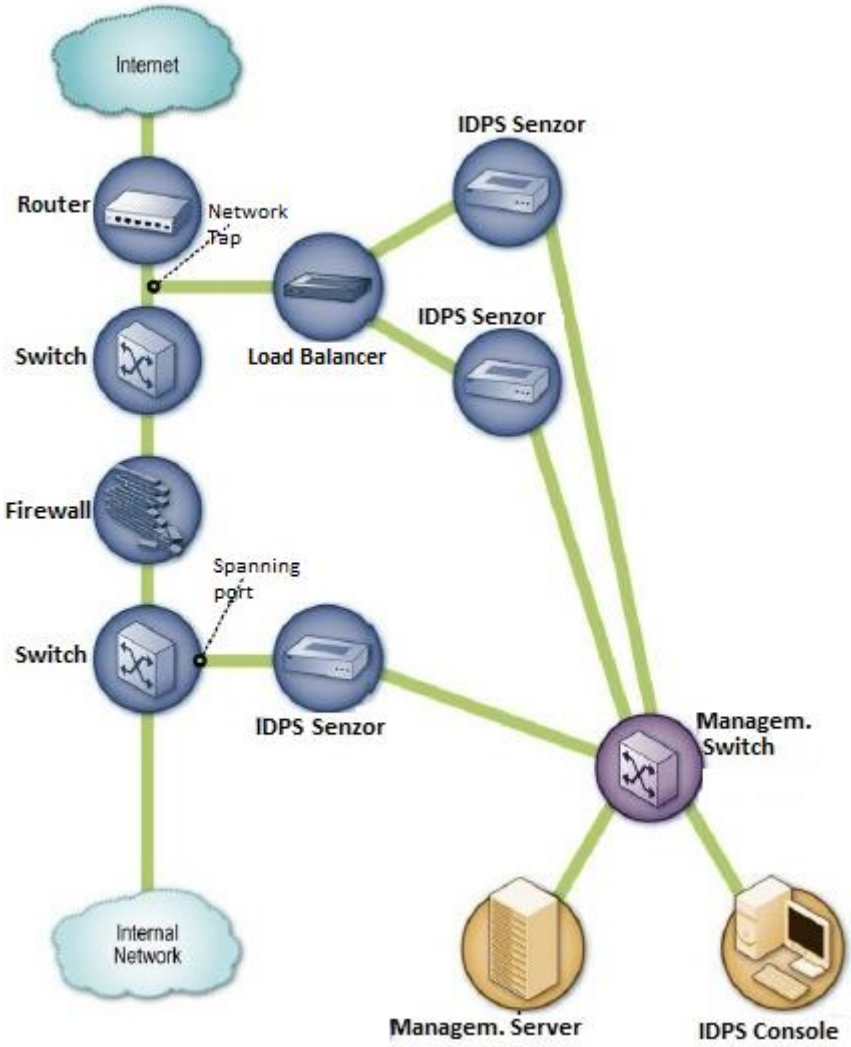

Fig.4.4: An example of NIDPS with passive sensors

Figure 4 shows an example of passive sensors that are connected to the network using balancers, network taps and spanning ports.
As it was previously stated, techniques by which a sensor to prevent requires that the sensor to be deployed in the inline mode, and not passive. Because the passive techniques monitor copies of the real traffic, they do not offer any secured means for the sensor to stop the actual traffic from fulfilling its target. In some cases, a passive sensor may depose packets on the network in order to destroy a connection, but this type of methods are less efficient than those inline. Generally, organizations should deploy sensors inline if is willing that the system perform prevention, and passive if only detection is needed.

4.4.2. Security Capabilities

As it was stated in an earlier section, IDPS technologies have a series of security capabilities, most important are: information gathering, logging, detection and prevention.
In this section I will talk about these capabilities for the particular case of network IDPSs because, as I previously said, I focused my research especially on this class of systems, as it will show up in the next chapters.

a) Information Gathering

Some network IDPSs offer some information gathering capabilities, a thing which proves that they can cull data about hosts and activities in networks that involve these hosts. Some examples of such capabilities are stated below:

- hosts identification: an IDPS sensor can build up a list of hosts on the organization's network sorted by the IP addresses or MAC. This list can be used as a profile for identifying new hosts on the network
- identifying the operation systems: an IDPS sensor may be capable to find out what operating systems the organization hosts use. For example, the sensor can track the ports in use on each host, a fact that can yield a certain SO (or family of them) (e.g. Windows, Unix, MacOS). Another method is to analyze the packets' headers for certain unusual traits that are being displayed by a certain SO, which is called 'passive fingerprint'. The sensors can also identify versions of applications, a thing that can also yield the SO being used.
- identifying of applications: for certain applications, IDPS sensors can identify their versions by keeping track of the ports that are in use and monitoring certain characteristics of the applications' communication. For example, when a client establishes a connexion with a server, this can reveal to him what version of the server software currently exists. Information about applications' versions can be used to identify potentially vulnerable applications, or unauthorized usage of those.
- identifying of network traits: some sensors of IDPSs gather general information about network traffic related to configuration of network devices and hosts. This type of information can be used for detecting changes in the configuration of networks.

b) Logging

Network IDPSs make intensive logging activities of the data about the observed events. These data can be used in order to confirm the validity of alerts, investigate incidents, correlate events among IDPSs and other logging sources.

BUPT

The fields of data usually logged by network IDPSs are:
- time stamps (gen. data and time)
- sessions and connexions IDs
- the type of alerts and events
- classification (priority, severity, impact, confidentiality)
- protocols of the levels of network, transport, application
- source and destination ID addresses
- TCP/UDP ports of the source and destination
- ICMP types and codes
- the number of bites transmitted on the connexion
- the content data decoded (e.g. applications' requests and responses)
- information regarding state (e.g. authentified users)
- prevention actions realized (optionally)

Most network IDPSs can perform packet captures. This is made after the occurrence of an alert, either to record subsequent activities in the connexion or record the entire connexion [169].

c) Detection

Network IDPSs offer a large and various range of detection capabilities. The big majority of products use a combination of techniques that rely on signatures, anomalies and stateful protocol analysis in order to make a more in-depth analysis of common protocols. Detection methods are generally closely interrelated; for example, a stateful protocol analysis engine may process activities from requests and responses which are examined for anomalies and compared with the signatures of known malicious activities. Some products use same techniques, such as software for analysis of network behavior (Network Behaviour Analysis - NBA).

Types of events that are most often detected by the sensors of network IDPSs are [169]:
- Application-level reckoning (e.g. banner grabbing, buffers overflow, formatting strings, passwords stealing, malware send, etc): most of network IDPSs analyze tens of application-level protocols, the most common analyzed are: DHCP, DNS, Finger, FTP, HTTP, IMAP, IRC, NFS, POP, rlogin/rsh, RPC, SMTP, SNMP, Telnet, TFTP; but same can be said for the those of databases, instant messaging, peer-to-peer file sharing software.
- transport level reckoning (e.g. port scanning, abnormal fragmenting of packets, SYN floods): transport level protocols mostly frequently analyzed are TCP şi UDP

BUPT

- network-level reckoning (e.g. fabricated IP addresses, illegal values of IP headers): network-level protocols that are most frequently analyzed are IPv4, ICMP, IGMP; many products added support for IPv6. The level of analysis of IPv6 that network IDPSs can achieve varies among products. Some do not offer at all IPv6 support or alert administrators of the existence of IPv6 activity, others can make basic processing of IPv6 and tunneled IPv6 traffic, like recording of the source and destination IP addresses, extraction of contents for an in-depth analysis. Some products can perform even complete analysis of IPv6 protocol, such as confirmation of IPv6 options validity, identification of abnormal use of the protocol. Organizations that have or will have in the near future need to monitor IPv6 activity should carefully assess the IPv6 analysis capabilities of their IDPS products.
- unexpected application services (e.g. tunneled protocols, backdoor, hosts on which unauthorized applications execute): these are generally being detected using SPA methods that are capable to determine if the activity from a connexion is consistent wih the expected protocol, or by means of anomaly-based methods that can identify changes in the networks' flows and open ports on the hosts.
- violations of policies (e.g. accessing of inadequate websites, use of interdicted application protocols): certain types of security policies violations may be detected by IDPSs that allow administrators to state characteristics of activities must not be allowed, such are TCP/UDP port numbers, IP addresses, websites names, and other data that may be identified following the analysis of traffic from networks

Some IDPSs may also monitor the initial negotiations effectuated when encrypted communications are established in order to identify client or server software that has known vulnerabilities or is being badly configured. This may include application-level protocols, like SSH or SSL, or virtual private networks, like IP Security (IPsec).
Network IDPSs sensors can find out whether an attack if probable to succeed. For example, as it was already set, sensors may know what software versions the web servers are using. If an attacker launches an attack against a web server that is not vulnerable to it then the sensor will produce a low priority alert, and if the server is vulnerable then the sensor produces a high priority alert. IDPS sensors are usually configured to stop the attacks even if it is or not probable that those will succeed. IDPS may log activities with different levels of priority function of what would have been their result if would not have been blocked.

I will make now a few clarifications regarding the accuracy of detecting the situations, same as it is stated in NIST guide from [169].

Since a long time ago, network IDPSs had been associated with high rates of false positives and negatives. The majority of early technologies used mainly signature-based detection, which is known to be accurate only for simple and already known threats.

BUPT

Recent technologies use a combination of methods in order to enhance detection accuracy and shrink the rates of false positives and negatives. Another problem regarding the accuracy of network IDPSs is that requires intensive tuning and customizations in order to take into consideration the characteristics of the observed environment. False positives and negatives can be reduced only due to the complexity of the monitored activities. A single sensor often monitors traffic that contains hundreds or even thousands of hosts. The number and variety of operating systems and applications that are being used in the monitored network, and also their degree of changing can be very high, which makes almost impossible the task of a sensor to understand those things. Besides these, sensors must monitor the activities for many combinations of clients and servers. For example, an organization may use 10 types and versions of web servers that are being accessed by users by means of 50 different types and versions of web browsers. Each browser-server combination may have some uniques features of communication, such as, for example, commands sequences, response codes that may affect the accuracy of the analysis. Security controls between servers and clients which modify the activity of the network, such are firewalls and proxy servers may cause additional difficulties.

In the most wanted scenario, the network IDPSs can interpret all the activity from the network in the same way terminals do. For example, different types of Web servers may interpret same request in multiple ways. Stateful protocol analysis tries to do this thing by copying the processing done by common types of clients and servers. This allows sensors to enhance their accuracy of detection. Many attackers use processing features that are specific to cliens and servers in their methods of attacks and evasions techniques, such are the processing of character encodings. Organizations should use network IDPSs that can compensate the use of common evasion techniques.

As it has been stated in the above section, network IDPSs need intensive operations of tuning and customization for enhancing the accuracy in detection. Examples of such capabilities are: thresholds for port scans and attempts of authentication inside applications, white lists and black lists with IP addresses of hosts, tuning for alerts. Certain products also provide code editing capabilities, which is generally limited to signatures but there are some cases when it allows access to programs, such are those for realization of SPA.

Some network IDPSs may use information related to the organizations' hosts for the enhancement of detection accuracy. For example, an IDPS may allow administrators to specify the IP addresses used by Web servers, mail servers, and other common types of hosts and state also the types of services provided by each host. This makes that the IDPS to be able to better prioritize alerts. For example, an alert for an Apache type of attack directed towards an Apache web server will have a priority higher than the same attack would be directed towards another type of server. Some IDPSs can import the results of vulnerabilities scans and use these to determine what attacks would succeed if not be blocked. This allows the IDPS to take better decisions regarding prevention actions and to more accurately prioritize the alerts.

In the end of this section I will discuss about the limitations of detection technologies.

Even though network IDPSs provide extensive detection capabilities, they hold also some significant problems most important being: encrypted network traffic

analysis, processing of large volumes of traffic data, endurance to attacks directed towards themselves.

Network IDPSs cannot detect attacks from within the encrypted traffic, such are Virtual Private Networks (VPN) connections, HTTP over SSL (HTTPS/SSL), or SSH sessions. As it was said previously, some network IDPSs can perform encrypted connections settings analysis, which can detect if client's or server's software has certain vulnerabilities or is badly configured. To assure that is being made sufficient analysis over the contents of encrypted traffic, organizations must use IDPSs that are capable to analyze the contents before being encrypted. An example of this fact is placing the IDPS sensors to monitor the unencrypted traffic (e.g. traffic that entered into the organization through a VPN gateway and was decrypted) and with the help of a software for host IDPSs to monitor activities inside the source or destination hosts.

Network IDPSs may lack capability to make complete analyzes for intense traffics. Passive sensors for IDPS may skip packets, fact which can lead incidents to pass unnoticed, especially if SPA methods are employed. In the case of inline IDPS sensors, skipping packets from big loads leads to faults in the network availability. Delays in the processing of packets may lead to a latency extremely high. In order to avoid such problems organizations that use inline IDPS sensors should choose the ones that are able to recognize the situations of massive loads and only specific types of traffic must pass through the sensors without performing the complete analysis or to give up the low priority traffic to reduce the load. Many providers try to optimize their sensors in order to achieve better performances under heavy loads by taking measures as: availability of a specialized hardware (network cards with big bandwidth) and re-compiling of software components in order to embed settings and other customizations made by administrators. Even though sellers classify their sensors after the capabilities of maximum bandwidth, the actual capacity of any product depends on many factors, the most important are:

- network, transport- and application-level protocols being in use and the depth of analysis that is being made for each protocol. Sellers often classify their products based on their ability to make a reasonable analysis of a 'mix' of protocols. The level of analysis that an organization wishes to achieve and the mix of its protocols may significantly differ from the test environment
- lifetime of connexions: a sensor can have fewer overload for a long term connection than for many short term consecutive connections
- number of simultaneous connections: sensors are limited to as many connections for which can be traceable

IDPS sensors are susceptible to various forms of attacks. Attackers may generate some unusually large volumes of traffic, likewise the distributed denial of service (DDoS), and anomalous activities, for example, unusual fragmented packets in the attempt to exhaust a sensor's resources or cause its failure. Another attack technique, called *blinding*, generates traffic in networks with the goal to cause many IDPS alerts. Usually, the traffic is specially created in order to take advantage of the typical configurations of the IDPS sensor. In many cases, blinding traffic is not directed towards any target, but the hacker executes the real attack separately, simultaneously with the blinding one. The attacker's goal is that the blinding traffic to cause any failure of the IDPS or generate such many alerts so that the ones for the real attack not be taken into account anymore. Many IDPS sensors are able to recognize common tools of DDoS and blinding, altogether with their techniques.

BUPT

They alert administrators about the attack then ignore the rest of activities in order to reduce their load. Organizations must choose products that offer functionalities that make them resilient to failures of attacks.

d) Prevention

Next, I will discuss the prevention capabilities of IDPS sensors grouped by their type, as it has been narrated in the security guidelines provided by NIST.

Passive sensors:

- *ending of the current TCP session:* a passive sensor tries to terminate a TCP session by sending reset packets at both endpoints (a process known under the name *cropping*). The sensor does this thing in order to make each side appear that the other one tries to close the connection. The purpose is that one of the sides closes the connection before the attack succeeds. One drawback of this technique is that the reset packets are not acknowledged on time because the attack traffic must be monitored and analyzed, the attack is detected and the packets sent over the network to the endpoints. Also, since this technique is applicable only at TCP protocol, it cannot be used for attacks carried in other packets, like UDP or ICMP. The session cropping technique is not very spread because other newer capabilities were created and proved to be more efficient

Inline sensors:

- *inline firewalling*: the majority of inline sensors provide firewalling capabilities that can be used in order to reject or discard activities that are considered suspicious in the network
- *shrink of the bandwidth use*: if a protocol is used un-accordingly, such as for performing DoS attacks, malware distribution, peer-to-peer file sharing, some inline IDPS sensors can limit the capacity of bandwidth the protocol will use. This prevents the malware activity to negatively impact the consumption of bandwidth for other resources.
- *modifying the malicious contents*: as it was shown a few sections ago, some inline IDPS sensors may clean portions from packets, which means that the malicious content is being replaced with a benign one and the newly obtained packet is sent to the destination. A sensor that acts as a proxy can perform automated normalization of the entire traffic, such would be re-packaging loads of the applications. This has as effect on curing some attacks from the packages headers even if the IDPS did not detected them. Some sensors may also empty the infected attachments of emails and other portions of content from the network traffic.

Both passive and inline:

- *reconfiguration of other network security devices:* many IDPS sensors may teach the network security devices, like firewalls, routers or switches to auto-configure in order to block certain traffics or re-route them. This can be useful in situations like keeping the attacker outside of the network or quarantining of a host that has been infected, e.g. moving it into a quarantine VLAN. This prevention technique is useful only for network traffic that can be distinguished by means of the characteristics of packets headers that are recognized by the network security devices, like IP addresses and port numbers
- *execution of a third-party program*: some IDPS sensors may execute an administrator program when a certain malicious activity is being detected. This may trigger any prevention activity specified by the administrator, such as reconfiguration of other security devices in order to block the malicious activity. Third-party programs are most frequently used when the IDPS supports the prevention actions specified by the administrator

The most part of IDPS sensors allow administrators to specify the configuration of prevention capabilities for each type of alert. This usually include activation or deactivation of prevention and specification of what capabilities to use. Some IDPS sensors have a learning or simulation mode embedded within that halts all prevention actions and, instead shows when such an action would have been made. This allows administrators to monitor and tune the configuration of prevention capabilities before activating them, which reduces the risk of erroneously blocking benign activities.

4.4.3. Management Capabilities

The main management activities of IDPSs are:

- implementation
- operation
- maintenance

a) Implementation

Once a network IDPS product has been selected, administrators must design an architecture, perform testing the components, secure the components and, in last phase deploy it. First step from the implementation of an IDPS represents the design of its architecture, whose considerations include:

a) the place where sensors must be placed
b) how safe the solution must be and what measures must be used in order to fulfill this goal, such would be to have many sensors to monitor the same activity in case one would fail or use multiple management servers such that to be able to use a backup server in case the main will fail

c) where the other components of the IDPSs will be located, like management servers, database servers, consoles and how many units of each will be enough for achieving the goals of usability, redundancy and load balancing
d) with what ohter systems the IDPS is to communicate:
    - ones it provides data, such as software for information security and event management, centralized logging servers, mail servers, paging systems
    - ones it sends prevention answers, such as firewalls, routers, switches
    - ones that manage the IDPS's components, like management software of networks or patches
e) whether it will be used or not a separated management network, and if yes then which will be its architecture, and if not then how the communications of IDPS components will be protected on the network
f) what other security controls and technologies must be changed in order to accommodate IDPS's deployment, like changing the firewalls' sets of rules

Organizations must think foremost to implement the components within a testing environment, and not a production one in order to avoid the probability that the occurred problems to create harm to the production networks. When components are deployed inside production networks, organizations should firstly activate only few sensors with prevention capabilities disabled. A new deployment normally generates a big number of false alerts until optimal tuning is reached, activation of multiple agents at once may overload the management servers and consoles, hardening administrators' work of tuning and customization. Many false positives are probable to be the same over sensors, thus it is useful to identify those positives either during the testing process or deployment of first sensors in order for them to be addressed before the general deployment of the IDPS. A phased deployment of the sensors is also useful in identifying the scalability problems [169].

The endeavor of implementation of an IDPS may require short periods of blackouts of systems' and networks' functionality for the installation of components. As it was previously affirmed, deployment inside a test environment can be particularly useful for discovering of the implementation problems in order for these to be solved correspondingly at deployment in production environment phase. Device-type components of IDPSs are generally easy to deploy. Administrators must provide sources of power, connect cables in the networks, start the devices and make general configurations (e.g. assign names to sensors, activate products by licenses etc.). Another task of administrators is to make updates to software and signatures in order to assure the actuality of IDPS's software.

Software-type components of the IDPS usually require more time for deployment operations than the hardware ones. Organizations foremost should acquire the necessary hardware, which could mean buying network cards of large bandwidth and ensuring that hardware is sufficiently sound for the IDPS; then administrators should install the operating systems that are compatible with the IDPS's software and to secure as much as possible the hosts. The operation of hosts consolidation includes updating the OSs, services and applications of the IDPS. Administrators also must realize IDPS software's configurations, as it is done for the hardware components.

BUPT

After component deployment must be configured the detection and prevention capabilities depending on the type of IDPS deployed. Without the realization of these configurations the IDPS can not detect only a low range of attacks.

Another basic activity in the implementation of IDPSs represents its own component security. This is a task of significant importance due to fact that IDPSs are at their turn the targets of the attackers. If an attacker manages to compromise an IDPS, this one becomes useless in the detection of future attacks on the hosts from the network which it was designed to protect. Moreover, IDPSs hold sensitive information, such as hosts configurations and known vulnerabilities that can be employed for the development of the attacks. Administrators must also perform certain activities in order to ensure that IDPS's components are protected accordingly:

- to create separate accounts for each IDPS user and assign the required privileges
- to configure firewalls, routers and other packet filtering devices in order to limit the direct access to the IDPS's components
- to ensure that all management communications of the IDPS are secured accordingly, either through physical or logical separations (management networks or VLANs, respectively), either by encrypting the communications. If the latter technique is employed then it must be performed by using FIPS-approved algorithms. Many products encrypt their communications by using the TLS protocol. For products that don't provide enough protection through encryption, organizations must use a virtual private network (VPN) or an encrypted tunneling method for the securitization of traffic

Some organizations also use strong authentication for the remote access to the IDPS's components, such as the two-factors one, that offer an additional level of protection.

b) Operation and Maintenance

Almost all IDPS products are created for being operated and maintained through a graphical user interface (GUI), called *console*. The Console allows admins to configure and update the sensors and management servers and to monitor their states (e.g. blackouts, packets failures, etc). Administrators also can manage users' accounts, personalize reports, and many other operations using the console. The functions that simple users of the IDPS can perform from the console include monitoring and analysis of the IDPS data, generation of the reports. The big majority of IDPS allow administrators to create accounts for each type of admin and simple user and to provide only the required privileges for each one's role. Console does these things by means of different menus and options based on the authenticated account's role. Some products also provide finer controls of the access such as specification for which sensor a user can monitor the data or administrators can change configurations. This thing allows a large IDPS deployment to be divided into multiple logical units.

BUPT

Some IDPS products have also command-line interfaces (CLI). Unlike GUIs, that are generally used for the remote control of the sensors and management servers, CLIs are normally used in a local management. Sometimes a CLI may be used remotely by means of an encrypted connection that is done through a secured shell or other methods. Consoles are much more usable than the command lines, which provide only a portion of functionalities of those ones.

Most majority of IDPS consoles have multiple capabilities for aiding users in their daily tasks. For example, they have „digging" capabilities, i.e. when a user analyzes an alert, details and information are organized on layers. This allows users to see information about multiple alerts simultaneously and show additional information about events of interest as needed. Some of the products allow users to see information from large ranges, such as packet captures (both on raw form and processed by means of a protocol analyzer), related alerts, as well as documentation about alerts. Usually, as much data the IDPS records, as easy it is for analysts to find the cause of incidents. Some consoles also offer event response capabilities, such would be transforming an alert into an incident case and offering workflow mechanisms that allow users to document additional information about the alert and to route to other users for a thorough review.

Other stuff offered are also various functions for reporting. For example, users and admins can use the console for running certain reports at some established periods of time and send by email the reports to the corresponding users or hosts. Many consoles also allow users to generate reports when it is necessary and to customize them. If an IDPS stores its logs into a database or file with an easily parsable format, queries or database scripts may be used in order to generate customized reports, especially if the console does not offer such functionalities.

Administrators must realize IDPS maintenance in a continuous manner, process that should include the following activities:

- observing the IDPS's components for operational and security problems
- checking at some well established time intervals whether the IDPS is functioning corresponding
- realization of regular checks of vulnerabilities
- receive notifications from vendors regarding the security problems of IDPS's components and responding to those notifications
- receiving notifications from IDPS's vendors about updates and perform their testing and deployment.

More about this subject will be discussed in the section below.

There are two types of IDPS updates: of software and of signatures. Software updates repair the software problems of the IDPS or add new functionalities, while those for signatures add new detection capabilities or enhance those existing. For many IDPSs, signatures updates require that the source code be modified or even replaced, thus they can be seen as a particular form of software updates. For others though, signatures are not implemented inside the code, thus an update is a change of the IDPS's configuration data.

Software updates can affect only one part or even all of the IDPS components, including sensors, management servers and consoles. Software updates for sensors and management servers (especially devices) are in general applicable by replacing an IDPS's CD with a new one and restarting of the device. Many IDPSs execute their software directly on the CD, so that is not necessary any additional software installation. Other components, like agents, need the administrator to install the software or apply the patches, either manually on each host or automatically by means of the IDPS's management software. Some vendors make available software and signatures updates to be downloaded from their Web sites. Often administrator interfaces of the IDPSs have functions for downloading and installing such updates.

Administrators should verify the integrity of the updates before applying them since it is possible to have been erroneously or unintentionally modified or replaced. The recommended verification method depends on the format of the update, so:

- files downloaded from a Web or FTP site: administrators must compare the checksums of files provided by vendors with the ones they calculate themselves
- updates downloaded automatically by means of IDPS's graphical UI: if an update is downloaded under the form of a single file or set of files then either the control sums provided by vendors must be compared with ones generated by the administrator, or the IDPS's user interface should perform a kind of integrity checking. In some cases, updates can be downloaded and installed as one single activity before verification of checksums. The IDPS's interface must check the integrity of each update as part of this operation
- removable media (CD, DVD, USB): vendors may not provide a specific way for the client to verify the authenticity of removable media that, apparently have been sent by them. If media verification is a problem then administrators must contact the vendors in order to find out how this can be done, such would be digital signatures checking from the content of media. Administrators should also consider scanning of the media for possible signs of malware, being careful that false positives can be triggered by the malware signatures from the media

IDPSs are typically realized so that software or signatures updates application not to have any consequence over the existing settings of tuning and customization. The main exception is code customization, which often must repeat when code updates from the vendors are installed. For any IDPS administrators must save periodically the configuration settings and before the application of software or signatures updates to ensure that existing settings are not lost by mistake [169].

Administrators must test software and signatures updates before application except for the emergency situations, such as would be when a signature identifies a new attack that produces damages in the organization and can't be detected or blocked by any other means. It is desirable to have at least one sensor that could be used strictly for updates testing. New detection capabilities can often cause the firing of numerous alerts, so signatures updates testing on a single agent may help to identify possible problematic signatures which need to be deactivated.

BUPT

In non-emergency situations, updates must be tested and deployed using the same habits that would be used for updating other major security controls, like firewalls or antiviruses. When updates are deployed inside production environments, administrators must deactivate certain signatures and do other reconfigurations as needed.

4.5. Conclusions

This chapter represents the beginning of the second major part of this thesis, namely the use of Semantic Web technologies inside the Cyber-Security domain. It has the purpose to show how these technologies are used at the construction of intrusion detection systems, how they support and enhance detection efficiency inside both singular hosts and computer networks. It is intended as an introduction in which are presented general notions from the domain of computer security, intrusion detection systems, like a review of the literature read by me.

Section 1 presents introductive notions about computer security, what are the most important industrial domains in which finds applications, what are intrusion detection and prevention systems and what are their main uses, and what are the most important techniques that are being used in detection (e.g. signatures, taxonomies, ontologies). These techniques, since they have a crucial importance for the functioning of any IDS, I dedicated a separate section for each one.

In section 2 has been discussed the general architecture of an IDS system, then had been talked its most important security capabilities: information gathering, logging, detection and prevention.

Beginning with section 4 the focus has been moved on the network IDPSs since these, unlike those for host, confront with some detection situations that are highly diversified, have some more complex architectures, types, technologies and knowledge required for the realization of the attacks in networks (especially on the Internet) are so much more advanced than in the case of single hosts, which means that also the ones required for their tracking are directly proportional. Two sections have been dedicated to presenting these types of systems; in the first had been discussed the main architectures together with their components while in the other the main security capabilities, which are the same as in general case but were explained for the specific case of networks environment. In the end of that section had been described the most important management tasks that administrators must perform for the deployment of the IDPS in a network and insurance its good functioning.

As any review work from a certain domain, the main objective here was also the realization of a presentation of the domain of cybersecurity and intrusion detection systems from the computer networks with information gathered, analyzed, selected, integrated from the resources read from the specialty literature, and also, for each particular notion, fact that was presented had been provided references in literature where the reader can embellish its knowledge.

BUPT

page intended left blankă

BUPT

5. ARTIFICIAL INTELLIGENCE AND SEMANTIC WEB TECHNOLOGIES USED IN THE CONSTRUCTION OF CYBERDEFENSE SYSTEMS

In this chapter I proposed a discussion about the intelligent defense systems, which are those that use in their functionality new and innovative techniques taken from other domains in order to increase their performances. Among these techniques are those of the newly Semantic Web and Artificial Intelligence. For each technique will be shown its applications in the detection process and the way it is incorporated within the IDS. The contribution will count also some models which present in a graphical manner the tales made textually with the aim to facilitate the understanding by reader of the detection process and methodologies. Will be presented in two tables a series of the most known IDS systems gathered by me from the domain literature that employ innovative technologies from Artificial Intelligence and Semantic Web in their functionality.

5.1. Artificial Intelligence in Intrusion Detection

Artificial Intelligence is a discipline of Computer Science that has as main objective to emulate the intelligent processes of humans by employing machines. His parents are considered to be Alan Turing and John McCarthy. The first published in year 1950 a reference paper for the domain in which proposed the idea of creating machines that were capable to think in similar manner with humans. He said that "human thought is a process that is extremely hard to define" and proposed the famous Turing Test [198]. The latter one introduced the term at a conference in Dartmouth, England in 1956. As the term itself suggests, 'intelligence' is artificial and is programmed by humans inside the machines in order to make them perform tasks in the same way as humans do and try to approximate as best as possible [202]. Two of the most important branches of this domain are Machine Learning (ML) and Deep Learning (DL), as it is presented in fig.1.

BUPT

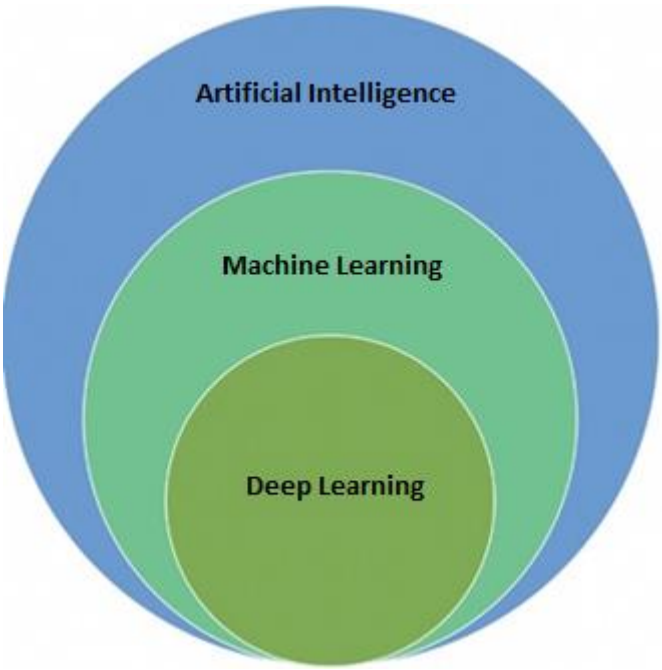

Fig.5.1: Main sub-branches of the AI domain

Machine Learning was defined by its pioneer, Arthur Samuel in year 1959 like: ”a field of study that focuses on offering computers the capacity to learn without being explicitly programmed”. Learning is a process in 5 steps, as it is stated in [166], and relies on establishing an implicit or explicit model by means of which the analyzed patterns are being classified or categorized. Its algorithms lay in three main classes: supervised learning, unsupervised learning and learning by consolidation (i.e. *reinforcement*).

Deep Learning represents the next generation of ML technologies. DL models rely on Artificial Neuronal Networks (ANNs) to create predictions totally independent of humans, unlike those of ML that still require humans’ intervention for achieving their objectives. The design of ANNs is inspired by the biological structure of the neuron of animals’ brains, featuring neurons and synapses between them, analyzing the data using a logical structure similar to the way of the process of conclusions drawing by humans [201].

In what follows will be described the main ML algorithms that have been used in intrusion detection field and examples will be provided of industrial-scale systems that employ these techniques within their detection methodologies. I will not enter details about the definitions and characteristics of each technique since this is outside the scope of this thesis, instead I will provide the reader with references in literature where he could find more information about the domain.

BUPT

5.1.1. Evolutionary Algorithms

Evolutionary Algorithms (or Calculus) (abbr. EA) represent a class of Machine Learning algorithms created to bring solutions to the global optimization problems, whose design was inspired by the biological evolution of living organisms [148]. Their aim is to find optimal solutions to problems, thus finding applications in multiple fields from Mathematics and Computer Science. The most renowned technologies of this class are: Genetic Algorithms (GA), Genetic Programming (GP) and Grammatical Evolution (GE) [139].

The fields from the intrusion detection domain where GA techniques were foremost applied are: automated model design, classification, optimization, features selection etc. The main benefits that they brought to this field, as it was affirmed by [134] are:

- providing an intrinsic parallelism, making them suit for the analysis of large volumes of data necessary in situations of detection
- are suited for behavior-based detection because they work with populations of solutions
- they grow the system's adaptability due to the fact that they are re-trainable
- support the dynamic generation of rules due to the propriety of evolution in time

According to Abdullah [3], the role of GAs in intrusion detection systems is that to derive a set of classification rules from the network audit data and the support-confidentiality framework is being used as fitness function for the evaluation of the quality of each rule. Generated rules are then used in the tasks of detection/ classification of events from the real network environments. The model of this process is shown in figure 2. Another similar work, [83] used genetic algorithms in order to select important traits from the set of test data KDDCup99 to perform a better analysis on the events from networks.

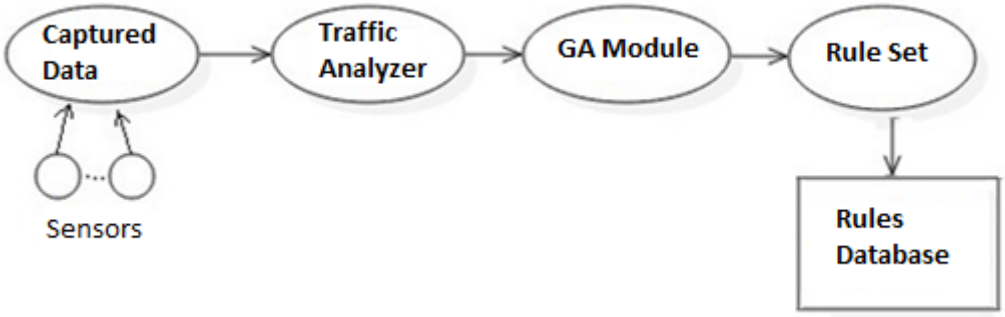

Fig.5.2: The placement of GA module inside IDPS

The second main technology, Genetic Programming (GP) also proved to be very useful in the domain of intrusion detection, same as GA. LaRoche&Heywood have discussed [128] about the use of GP for attacks detection in wireless networks 802.11 and stated what facts lead to the reduction of detection rates of GP methods for the attacks specific to the 802.11 protocol. One of the largest research system

BUPT

that have been developed until present and that relies on evolutionary calculus is ECJ27, a system for evolutionary calculus that was written in Java language, being described by some as "the most widely used library of evolutionary calculus of general purpose", which has been applied of course also in intrusion detection field.

5.1.2. Fuzzy Logic

Fuzzy Logic is a superset the classical (Boolean) logic extended with multiple truth values to be able to represent imprecision and uncertainty associated with behaviors found in the real world (also called multi-value logic) [75]. In the intrusion detection domain, fuzzy logic is an important technique used especially in analysis. Fuzzy systems are characterized by their capacity to reason over incomplete or uncertain data, a fact that makes them some useful tools for risks analysis and evaluation. (Ansari et al.) [14] proposed the use of fuzzy logic inside the Data Mining discovery rules generation in order to introduce cognitive aspects to support FRCP rectifications. (Alali et al.) [11] proposed a fuzzy rule-based inference system in order to better evaluate the risk of attacks occurrence. Fuzzy methods are used especially in the field of anomaly-based detection because the features that should be considered can be seen as fuzzy variables from the set. They proved to be especially efficient against threats such as port scanning and probes. The main drawback of this technique constitutes the intensive resource demands needed to realize the computations [191].

5.1.3. Clusters Analysis

Clusters Analysis and *Outliers* is a ML unsupervised technique that works by grouping (classifying) of a set of data entries into a specific group (groups) according to a similarity metric (distance). For a brief introduction and a list of the most important algorithms that have been created until present day, I invite the reader to see the work in [180]. The clustering and outliers techniques are most often being used in anomaly-based detection, where the latter are considered the anomalies. The advantage that they bring into the field is that the effort required for tuning the IDS is smaller because it detects the intrusions only from raw audit data. The attacks detector model that relies on clustering and outliers is presented in fig.3.

(Brahmi et al.) [52] present the effort of developing an IDS that uses the Artificial Intelligence technologies of multi-agent and ontologies and proposes a clustering algorithm having as main objective the increase of scalability and the detection capacity of the IDS. One of the broadest works read by me is that of (Agarwal&Hussain) [5], which is both a comprehensive review of the domain, and also propose a conceptual framework of an ideal IDS for Web that employs numerous technologies, some of these borrowed from tha AI domain.

BUPT

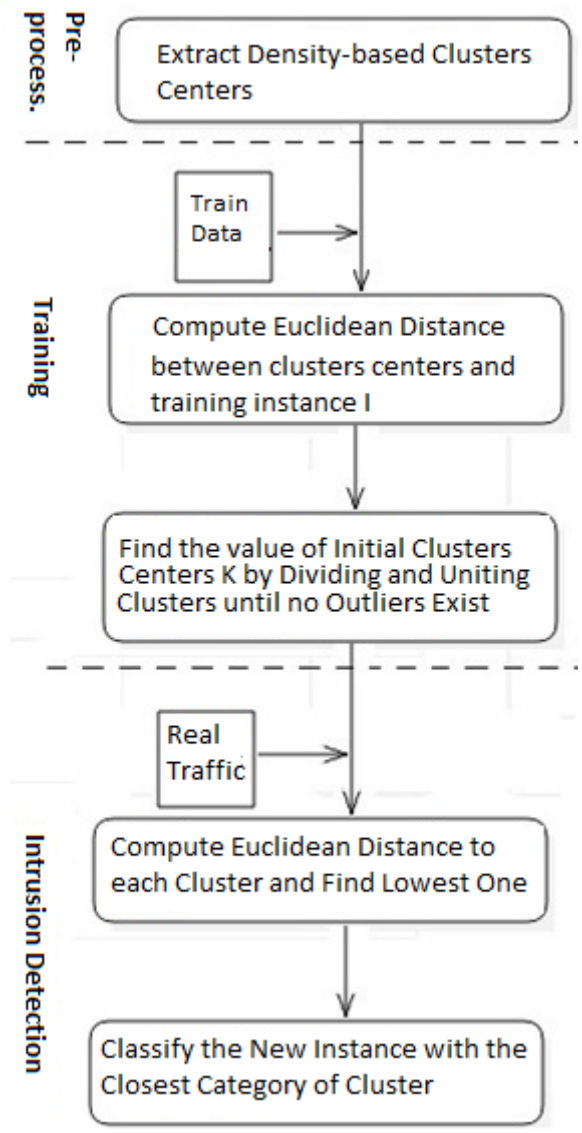

Fig.5.3: Clustering and outliers in attacks detection

5.1.4. Artificial Neural Networks

Artificial Neural Networks (ANN) are a mathematical model that process information in a manner that is inspired by the functionality of nervous systems found at living mammals [147]. Lee [130] presents the most important 10 ANN architectures that have been proposed until present day in the domain literature.

This AI technology can find applications in multiple domains due to its capacity of making correct decisions and recognition of forms and patterns and constitutes the nucleus of DL techniques. ANNs are used especially in anomaly-based detection for creating and learning the profiles of benign activities from the raw traffic data and detect by classification of the new events based on the established profiles [167]. The conceptual model of the detection process is presented in fig.4. In [13] is proposed a detection system that relies on ANNs for the detection and classification of attacks in computer networks, stating that this technique is superior to the traditional one based on signatures due to the fact that is capable to learn the behaviors of dynamic changes of users and systems and also showed an increased adaptability to changes. One of its disadvantages is that the training phase is a big time consumer. (Igor et al.) [116] made a study regarding the pros of using ANNs in anomaly-based detection systems.

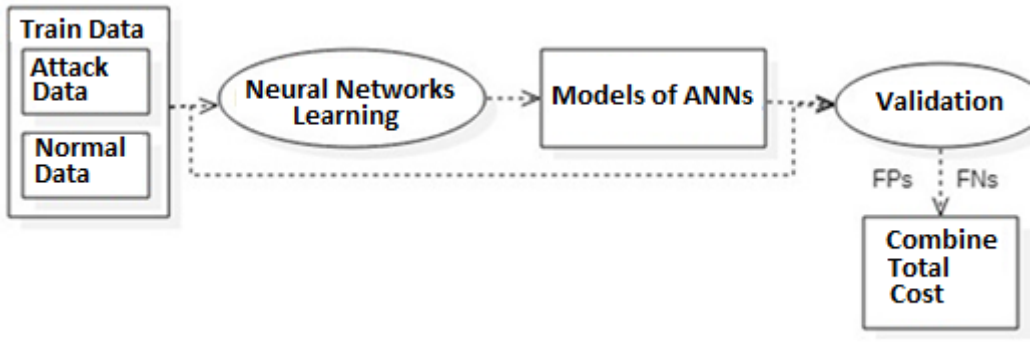

Fig.5.4: The use of ANNs in attack detection systems

5.1.5. Data Mining

Data Mining (DM) is a technique employed to the processing of large sets of data with the ultimate goal to find hidden information and patterns and establish relations in order to solve problems by analyzing data, thus being able to make predictions on the future directions of the behaviors. It is considered to be a specialized form of the technique of knowledge discovery from data (KDD) [78]. For the most important DM methods reader is referred to the site Educba [217].

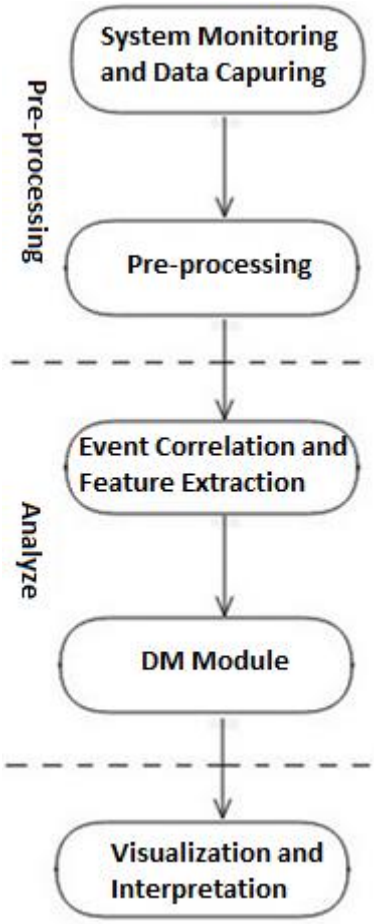

Fig.5.5: Data Mining in the process of attack detection

Data Mining techniques can be very useful in the field of attack detection because this activity implies the processing of some huge volumes of data. The role of Data Mining in the detection process is especially related to that of data analysis, more precisely it is focused on dimensionality reduction, clustering and classification, as it is shown in fig.5. (Brahmi et al.) [51] sustained the idea that the application of Data Mining techniques in the field of intrusion detection may lead to the increase of accuracy and speed the detection and in the same time also harden system's own security. They proposed a distributed IDS that used technologies of multi-agent, Data Mining and clustering in order to overcome the above-mentioned drawbacks. Other researchers stated that the advantage compared to the signatures technique is represented by the high level of accuracy in detection of known attacks and their variations [129].

The 5 techniques that had been presented in this section are not the only AI technologies that are employed in the detection of attacks and intrusions, but they

BUPT

are definitely among the most importante. For other technologies I invite the reader to see the works from [85], [195], [32], [33] where are presented also other ML techniques together with each one's role inside cybersecurity.

5.1.6. Industrial and Research Systems based on AI Technologies

Table 1 shows a survey with some of the most important research IDSs that employ AI technologies in detection, as it was studied in domain literature read in conducting the current research. Other related lists of comercial systems may be found in [85], [118], [129].

ANIDS product	Proprietary Organization	AI Techniques
EMERALD (Event Monitoring Enabling Response to Anomalous Live Disturbance)	SRI International	Rule-based Expert Systems, Bayesian Inference, Forward Chaining
NetSTAT (Network-based State Transition Analysis Tool)	University California Santa Barbara	State-Transition Analysis Rules
Bro	Lawrence Livermore National Laboratory	Application-Level Semantics, Pattern Matching
ComputerWatch	Secure Systems Dept, AT&T Communicat.	Expert Systems, Pattern Matching, Rule-based
AAFID (Architecture for Intrusion Detection using Autonomous Agents)	Purdue University	Autonomous Agents
Hummer	University of Idaho	Data Mining, Autonomous Agents
JAM (Java Agents for Meta-learning)	Columbia University	Artificial Neuronal Networks, Bayesian clasifier, Data Mining, Nearest Neighbor, Decision Trees, Rule-based Inference
Snort IDS	Cisco Systems	Statistical Analysis
MINDS (Minnesota Intrusion Detection System)	University of Minnesota	Data Mining, Pattern Matching, Clustering and Outliers
DMNIDS (Data Mining for Network Intrusion Detection Systems)	MITRE corporation	Data Mining, Clustering and Outliers
MADAM	Columbia University	Data Mining, Association Rules, Frequent episodes

Table 5.1: A list with some commercial NIDS and ML techniques used

BUPT

For a state-of-art regarding commercial NIDS that use AI technologies in the detection process the reader is invited to see the articles of the renown american company for business analysis and research Aite Group [122], [123], [124], [125].

Name	**Entity**	**AI technologies used**
Anagram	Intrusion Detection Systems Lab, Columbia University	Content modeling using N-grams
Autonomous Agents for Intrusion Detection (AAID)	CERIAS, Purdue University	Open-source platform / additional anomaly-based modules avaliable; distributed architecture
Bro	Lawrence Berkeley National Laboratory	Development platform, compatible with Snort rules, application-level semantics, events analysis, pattern matching, protocols analysis
Data Mining for Network Intrusion Detection	MITRE Corporation	Clustering techniques
Dependable anomaly Detection with Diagnosis	Various partners	Detection by diversification
EMERALD	SRI	Open-source Distributed Platform, rule-based, inference, Bayesian inference
Genetic Art for Intrusion Detection (GenArt IDS)	Northwestern University	Genetic Algorithms
GIDRE	University of Granada	Distributed architecture, stochastic modeling, pattern matching
Intelligent Intrusion Detection	Mississippi State University	Data Mining added with Fuzzy Logic
Minnesota Intrusion Detect. System (MNIDS)	University of Minnesota	Statistical Analysis, Pattern matching, Data Mining, Outlier detection
Network at Guard (N@G)	C-DAC	Development platform, Protocol anomalies detection, Statistical analysis
NetStat	University of California	
Neuro-Fuzzy Intrusion Detect.System (NFIDS)	University of Teheran	Fuzzy Logic, Artificial Neural Networks
OrchIDS	Ecole Normale Superieure de Cachan	Real-time events analysis, temporal correlation
Prelude	Yoann Vandoorselaere	Open-source Distributed Platform
Shadow	CIDER Project	Development Platform (old CIDER)
Snort	Marty Roesch	Open-source platform/ Multiple anomalies based modules

Table 5.2: Anomaly-based IDS research systems

BUPT

5.2. The Semantic Web in Intrusion Detection

The Semantic Web, as it was stated by its creator himself, Tim Berners-Lee, represents the vision of a Web completely automated of machines that communicate and perform the tasks in humans' place, the programs that deal with things based on their meanings (semantics), of data that have a universal format and structure in order to make them comprehensible by machines [42].

Semantic Web technologies, like "content" or "ontology", may be used in many fields of Computer Science. Each security method that relies on the notion of "content" may use Semantic Web technologies, and the intrusion detection systems are a good example [2].

Even if IDSs are the building blocks of security infrastructures, they suffer from a number of limitations, such as: safety, relevance, incompleteness and disparity in presence and manipulation of knowledge and attack detection. Big majority of IDSs use a centralized architecture that contains multiple nodes that communicate with a central processing one. This method suffers from the "single failure point" problem, i.e. in the situation when the central node is being attacked, the whole IDS is at danger. Also the transfer of all information to a single processing node puts great demands on the the network's resources and may lead to overhead. One solution to the above problems would be the integration of a multi-agent technology inside the IDS. The use of a multi-agent approach offers a series of advantages, as it was explained in literature: scalability, minimal network overhead, continuous and independent execution of agents, making thus system's resilience much stronger and ensuring its soundness [52].

Subsequently, the concept of "ontology" arose as a technology for knowledge representation and sharing of an application domain. In the intrusion detection field they are used to provide IDS with the capacity of sharing a common understanding about attacks and intrusions and to create signature rules. Use of ontologies in attacks detection domain brings with it the following boons, as it was stated in [2]:

- brings together semantic knowledge of a domain
- better express the IDS by constructing some superior signature rules by making use of Semantic Web languages (e.g. RIF, SWRL)
- make reasoning an intelligent process

Some scholars from Computer Science opened a new branch in information security, that of ontologies use together with their benefits. They made the following sentence: "ontologies are a new extremely promising paradigm in the informational security field by means of which we have a tool of unlimited event classification" [163].

Use of Semantic Web technologies in the construction of IDS systems is a late concept. Among the first researches that had been done in this field count those of Undercoffer [199], [200]. The former proposes an ontology that represents a model for computer attacks and stated that any taxonomical characteristic used at defining a computer attack must be limited in scope to those features that are observable and measurable at the target. The second work presents an ontology that defines relations among features that are observable by the IDS sensors.

BUPT

5.3. State-of-the-Art in Researches

In this section I will present some of the most important articles read in the creation of this chapter regarding AI techniques used in the construction of IDS.

A broad work that spans multiple areas from the computer security, beginning with the basic notions and until the most advanced is the paper of (Agrawal&Hussain) [5]. It is created as a review article focused on the intrusion detection from Web applications. Presents a brief overview of the main intrusion detection techniques, challenges faced by researchers at the construction of systems, security capabilities embedded inside most of the systems. They identified 9 dimensions of IDS funcţionalities based on which they compare 5 systems from literature. As contributions they propose a conceptual framework of an ideal IDPS that included a number of new functionalities as against the 5 systems they reviewed in their study.

The content from this chapter was inspired mainly by two articles: (Garcia-Teodoro et al.) [85] and (Tsai et al.) [195]. The former made a literary review about the domain of anomaly-based IDPS. They stated that currently exist three main techniques that are used in anomaly detection, that rely on: statistics, knowledge and machine learning. The latter category has been tackled more in-depth by the study, but it was one more of a theoretical fashion in which was talked about the characteristics of those techniques and less of their role in the construction of IDSs, that is what I wanted to do in the current chapter. The second main work is also a review from the same domain, the use of ML technologies in the construction of IDSs. Technologies are classified into 3 main categories: simple learners, combined learners and hybrid. The paper also realizes a comprehensive state-of-art, in which were surveyed 55 papers from the period 2000-2007 and examined the used methods, conducted experiments from the perspective of machine learning.

From the second domain that was discussed in this chapter, the Semantic Web, I will mention the works below, all sustaining the idea of using ontologies inside the cyberdefense domain.

(Razzaq et al.) [163] state that ontologies are an intelligent approach for information security, and affirmed that frameworks built by means of this new technology are ”a promising new line of defense that can be very efficient in detecting sophisticated attacks because they are capable to capture the context of information”.

Also they [164] proposed an ontology-based intrusion detection system for application-level that employed a Bayesian filter. The ontology was used as a knowledge base in order to provide a common understanding of the concepts of a domain and the ability to analyze information automatically, as well as inference and reasoning capabilities.

(Brahmi et al.) [52] created a hybrid IDS using a combination of many techniques from AI domain, like multi-agent, ontologies, clustering. Showed the advantages of distributed approach for IDS construction, affirmed that ontologies are a powerful tool used for representation and sharing of knowledge from a certain domain. These components provide IDSs the capacity to perform automated and continuous analysis and reason about the instances of attacks data.

BUPT

5.4. Conclusions

In this chapter have been shown and explained different methods and techniques that are employed in the field of attacks detection in order to build more robust, performant and intelligent systems with enhanced functionalities. Currently existing platforms may be split into two categories: commercial and research. Commercial systems tend to use well-established technologies, most of them relying on signature modules. The research ones contain the most recent and innovative technologies, such are those from AI or SW that have been presented here. I chose the domain of Artificial Intelligence and the newly-occurred Semantic Web because these are the most widely spread technologies for building intelligent and capable systems from any industrial domain, and cybersecurity is a very good example. For each AI technology presented has been explained the role it has in the intrusion detection field, as it has been stated in the literature read by me in the creation of the current chapter. My work also proposed models for each technique discussed that graphically present the detection methodology based on that technique. Also it has been made a review of industrial-scale systems that employ those techniques in their detection methodologies.

BUPT

page intended left blank

BUPT

6. ONTOLOGY FOR CYBERSECURITY IN NETWORKS OF COMPUTERS

In this chapter is proposed a model for the domain of cybersecurity under the form of an ontology that can be used by the detection systems for awareness in possibly dangerous situations. The ontology proposes to describe the domain by capturing the most important concepts and their relations. At its construction and evaluation were used state-of-art methodologies, such as Semantic Web's standard languages built especially for this purpose, like OWL (vers.2) and SWRL. Its efficiency was tested by incorporating into an application-level firewall, providing this one the knowledge base with the semantic rules. The resulted prototype IDS was tested and compared with other systems from the literature that do not employ semantic technologies in their detection processes, my proposed ontology-based system proved its superiority.

6.1. Introduction

Cybersecurity is the science that deals with the protection of communication and information systems by the attackers that try to exploit their resources or to produce them harms[138]. Finds utility in almost each aspect of today's society: daily life, organizations, corporations, government units, each and every one uses the Internet to communicate, interact and collaborate. Thus, the Internet infrastructure becomes a means of spreading, besides useful information, also of the malicious one which takes the form of cyber-attacks and computer intrusions. Almost any news source that we consult daily (such as radio, TV, magazines, sites) says that the number of cyberattacks lays on an ascending slope and founds at an alarming level.

Cyberspace represents a unique combination of human things and computers whose complex interactions happen in a global communication network. Due to this cause it presents a vulnerability at suspect users' situations. Situations awareness depends on the perception by users of the world's environment and understanding of its semantic structure [151]. Explanation of these vulnerabilities of systems from networks had been put by some researchers on the following reasons:

- an inadequate technological infrastructure: Internet is at its roots nothing else than a communications network built more than half a century ago in the scope of doing military communications
- situation awareness by the human factor of the cyberspace: how security operators and users perceive (recognize) the surrounding environment

The solution to all these problems, as it was proposed by some scholars, was that a full-fledged cybersecurity science must be created whose fundamental goal is

BUPT

to perceive the cyber-space as a hybrid framework of human-machine interactions in which security policies and personal data protection have a significant role [151].

Computer attacks and intrusions date back even from the beginnings of science (and especially of the Internet), the earliest ones being known to have occurred at the beginning of '80s. A system that is used in a pernicious situation to alert, mitigate, block another system or group of systems of the occurrence of some types of attacks is called Intrusion Detection System (IDS). Depending on the place where it operates and the detected attack, an IDS can be for:

- host: operate only locally in order to protect a single machine
- network: operate in a distributed system in order to protect many computing systems that are linked inside a network

Domain literature says that IDSs have gone through multiple phases of evolution until present day, the most important, according to [163], are:

1- attacks signatures
2- attacks taxonomies
3- attacks ontologies

First type of IDSs relied on attacks signatures, which are a syntactical representation of them. This technique isn't though a very efficient one because signatures contain a small volume of semantic knowledge and lack any trace of formal logic.

The second phase in the evolution represented the use of taxonomies. The central components of the functionality of an IDS are the taxonomy and a description language for its instances. Taxonomy has the role of characterizing and classifying the information about attacks.

The current phase in the evolution of IDSs is the use of Semantic Web technologies, and the most important are ontologies. Cyberdefense systems that are built using an ontological approach represent a promising new line of defense that can be capable to detect sophisticated attacks and even Zero-Day (previously unknown) due to their ability to capture the context of information and filter it on certain criteria [163].

An ontology represent a formal specification of the concepts and relations among the entities of an application domain. Unlike taxonomies, they have powerful constructors, such as machine-interpretable definitions of domain concepts and relationships among them, providing software systems with the ability to share a common understanding of the information and to be able to make reasoning over it. Ontologies are built in the scope of information sharing and reuse among the entities of a domain, in the present situation, cyberdefense systems.

The most important languages for ontology development that have been created by now are: RDF, RDFS, DAML+OIL, and, the most important one, OWL together with its last version, OWL2. All had been created as standards for the Semantic Web.

My ontological model that stores information about attacks has been constructed following a 7-steps methodology, as it will be explained more in detail in section 3. For the realization of the current research I endorsed on 2 works I read from the literature, those of Razzaq [163] and Zhu [202], but I tried to build a more comprehensive model, the ones from the above-mentioned papers were created only for specific levels of the OSI stack (more exactly application and HTTP

protocolul), respectively a common language for cybersecurity information sharing. The proposed ontology was developed into the Protégé 5 environment that was proposed at Stanford University and represented in the last version of the Semantic Web standard ontology language, OWL2.0.

For the construction had been studied a variety of resources from literature, such as catalogs of malware, dictionaries with attacks information, ontologies and schemas, and many others. These will be mentioned at the end of the next section.

6.2. State-of-Art in Researches

In this section I will present a list of some of the most important articles about semantic security that I read for the creation of the current chapter. This is an emerging research area that brings together the information security and semantic systems domains in order to create more efficient defense frameworks that fight more promising against cyberattacks which are continuously proliferating. A big majority of semantic technologies used for the creation of security systems are taken from the domains of Semantic Web and Artificial Intelligence, among the most important mention ontologies, multi-agents, learning algorithms, clustering etc.

(Razzaq et al.) [163] proposed two ontological models for the computer attacks created by means of the technologies and languages of the Semantic Web. One of them conceptualizes the domain of cyberattacks and the other one is for the communication protocol HTTP, both being developed with the goal in mind to enhance the capacity of an IDS to detect attacks at the application level. Their article is a vast work that tackles many fields from the domain of ontologies engineering, such as development methodologies, implementation and evaluation, storing etc. Their ontologies had been tested by using an application-level firewall and the obtained results were compared with those of other systems, proving to be superior for different evaluation parameters.

(Undercoffer et al.) [199] proposed an ontology that specifies a model of computer attacks using the Semantic Web technology languages DAML+OIL and DAMLJessKB. For the construction phase they studied the CERT/CC Advisories and ICAT sources regarding the classes of attacks, and the second source was the OS Linux kernel from which had been gathered over 190 attributes at the system, process and network levels. The ontology was tested by means of a distributed IDS and used in order to detect several types of known attacks, like Mitnick, Buffer Overflow or Denial of Service.

Zhu [202] affirmed that a necessary condition for achieving a good cyber-defense is to share patterns (formats) of attacks. He proposed an ontology for attack patterns as a common language for cybersecurity information sharing that has as main purpose to provide the defenders a good understanding and a systematic analysis regarding the perspectives of the hackers. This paper is also a vast effort that covers many theoretical notions from the ontology engineering domain. Most part presents the process of ontology development based on a 7-step methodology.

(Oltramari et al.)[151] proposed an ontology for secured operations in cyberspace, and explain that there is need for a good awareness of the environment in order to be performed a good defense. The objectives of the ontology are to increase the situations awareness by the cyber defenders, thus helping them in the processes of decision making depending on the state of the environment. The ontology has been created by reusing other existing ones, such as CRATELO, an

BUPT

ontological framework from the Alliance of Research in Cybersecurity (CSRA) in order to support new use scenarios.

(Baader et al.) [25] describe the role of Description Logics in the development of the Semantic Web. They present the benefits that DLs bring into the ontologies domain, for their construction, maintenance and integration. Most part of the article is dedicated to the discussion of SHIQ language, a description logic from literature.

(Obrst et al.) [150] developed an ontology of malware that allows integration of data from many sources and whose fundamental structure is represented by the Diamond Model of the malicious activity. Their effort was focused on savings by reusing some existing ontologies, extending them in order to deal with new security scenarios. Are presented the available resources for anyone who wants to build an ontology for cybersecurity (taxonomies, dictionaries, ontologies), languages for modeling security incidents (OpenIOC, IODefVeris), models for the existing attack patterns (WASC, SCAP etc).

In [70] Ding et al. made a literary review in which is discussed the domain of ontologies as a Semantic Web technology. Are presented the evolution of the ontology languages, like DAML+OIL, OWL, RDFS, comparisons are made over the constructors of each one, are represented the ontology management tools (stores, editors, processors), comparisons between the characteristics of the most known triple stores used at the persistence of ontologies instances, etc. Also are presented the real-world applications of ontologies in different domains, such as Web services composition and description, WSN networks state descriptions, publishing of data from the personal profiles etc.

For the construction of the ontology proposed in this chapter I studied a number of resources from the cybersecurity literature, such as taxonomies, catalogs, schemas, lexicas and ontologies. Had been laid many efforts for malware classification, such as patterns, variants and characteristics. These are good sources that can be consulted for concepts, abstractions, entities, attributes, relations etc. Below I will present those that I consider to be the most important for this study from the ones that I read.

MAEC is a language for addressing all sorts of malware and specify their characteristics and manifests based on attributes patterns such as behaviors, artifacts and attacks. It organizes information onto a 3 levels hierarchy, from the bottom to the upper-most, and abstracts the actions from their implementations (syntaxes and semantics). For more documentation to be seen source [233].

CAPEC is a taxonomy with attacks patterns of MITRE corporation [234]. Contains 68 categories and 400 patterns. Categories correspond to the MAEC mechanisms and patterns to the first two levels of MAEC. Another catalog with a similar purpose is WASC Threat Classification [235].

NetOps is an ontology developed by MITRE in 2009 with information about operations in computer networks. This represents blocks of missions of interest for the management of US Federal Government networks.

SCAP is a suite of specifications that standardize the format and nomenclature for the communication between security software products of the configuration information [236]. The most important for the work in the current chapter are:

- OVAL
- CPE
- CCE
- CVE

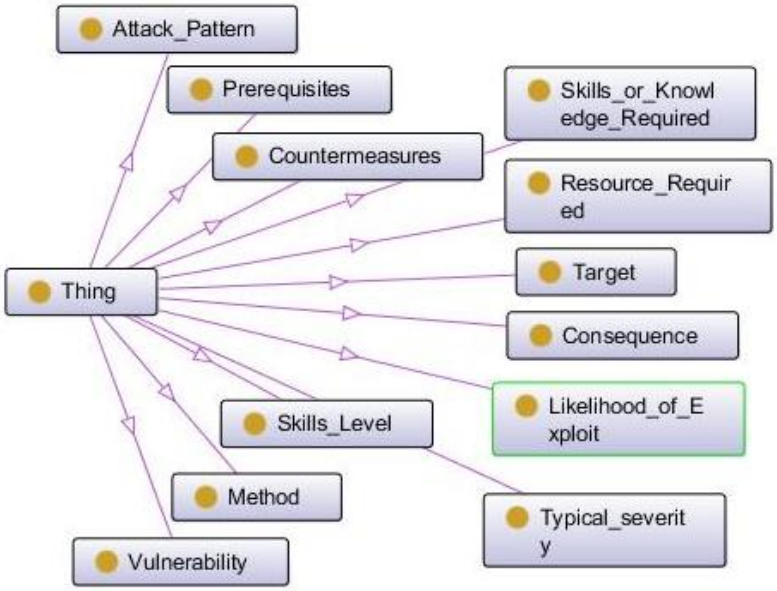

Fig.6.1: Top tier of the ontology and its classes (viz. Protégé)

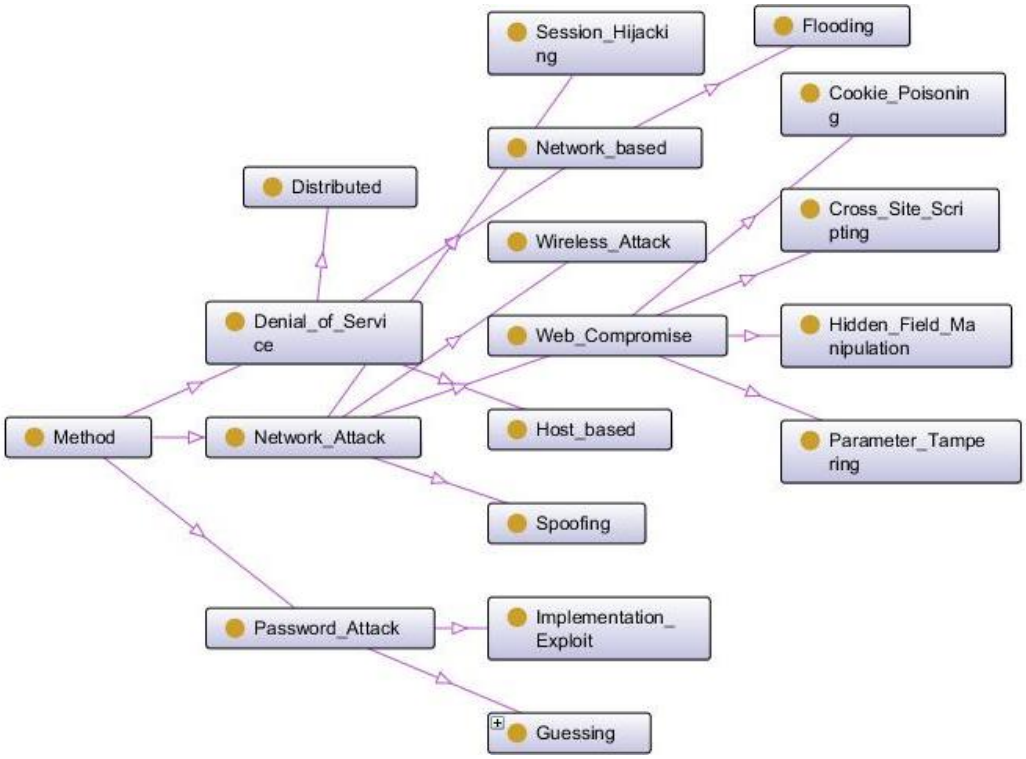

Fig.6.2: The *owl:Target* class extended with all its subclasses

6.3. Construction

The knowledge model used by the IDS in the process of attacks detection is proposed in the form of an ontology of cyber-operations. It was designed to contain knowledge about types of attacks and threats, consequences of attacks, countermeasures for mitigating and stopping the effects of attacks, vulnerabilities being exploited by attackers at each level, and much other information.

An ontology is an explicit specification under the form of a data structure that captures the important notions from a real-world domain and their relations. The process of ontologies design is an iteration to determine the scope, define concepts (classes), proprieties (relations), axioms, constraints and instances.

For the construction of my ontological model I chose the methodology OntologyDevelopment 101, which is a 7-step process for the development of domain ontologies, as it is affirmed in [149].
Next I will try to point out each of these steps in the case of constructing my ontology.

A) Ontology scope and objective

My ontology serves as a knowledge base that stores information from of the cybersecurity domain, about attacks, victims, operational methods, targets, countermeasures, knowledge and resources required etc. It is being used by the detection system for the verification of information from a newly occurred event in order to find out if it's an attack (known or unknown) and is being updated by this with new information from these situations.

B) Key concepts of the domain

In this step are identified the most relevant concepts for the domain that is modeled. These are the most general concepts of the ontology (upper-tier) from which all others are derived. In the current case, these concepts are: attack, vulnerability, method, source, target, consequence, countermeasure, exploit type, severity, required knowledge.

C) Classes and properties

Classes that describe the structure of the domain are being organized into a 3-levels hierarchy: upper, middle and bottom. Each concept from an inferior tier is derived from those on the next upper tier and captures some more specific concepts of the domain.

Properties (relations) in an ontology are of 3 types:

- of objects
- of data, and
- of annotations.

Object properties are relations that happen between classes and objects. They have the following names scheme: *hasProperty* and *isInversePropertyOf* in order to describe both sides of a relation (e.g. class A *hasSubClass* B, and class B *isSubClassOf* A). Data properties are the attributes of classes that describe their structure. For example: the class Attack has properties: *ID*, *Name*, *Level*.

Annotation properties have the main role to create a brief textual description of concepts (classes).

D) Facets of properties

Facets of properties are constraints and restrictions that apply to properties, such would be:

- cardinality: the number of values
- quantifiers: relation in which an individual participates
- data types: the types of values the data properties can take
- *hasValue* restrictions

E) Formalization

In this step the language for formalization (specification) of the ontology is chosen together with the environment where it will be developed. In this case I chose Protégé 5, which is today's most used ontology editor created at Stanford University, in California. For the specification language, I also headed towards a state-of-art of the domain, the Web Ontology Language, version 2 (OWL2.0).

F) Instances

I instantiated the classes of the ontology in order to create individuals that represent a form of concrete information about attacks and their relations, as it has been specified inside the ontology. Were created instances of attacks, consequences, vulnerabilities exploited, target components of the systems, solutions for the attacks effects mitigation, knowledge required by hackers, and many others. Instances of the ontology classes have been stored into a knowledge base which is queried and updated by the web application IDS during detection process.

The link towards the complete ontology will be stated in the Conclusion section of this chapter so that it can be downloaded. In that can be observed all classes, properties, constraints, instances, as they have been discussed during this section.

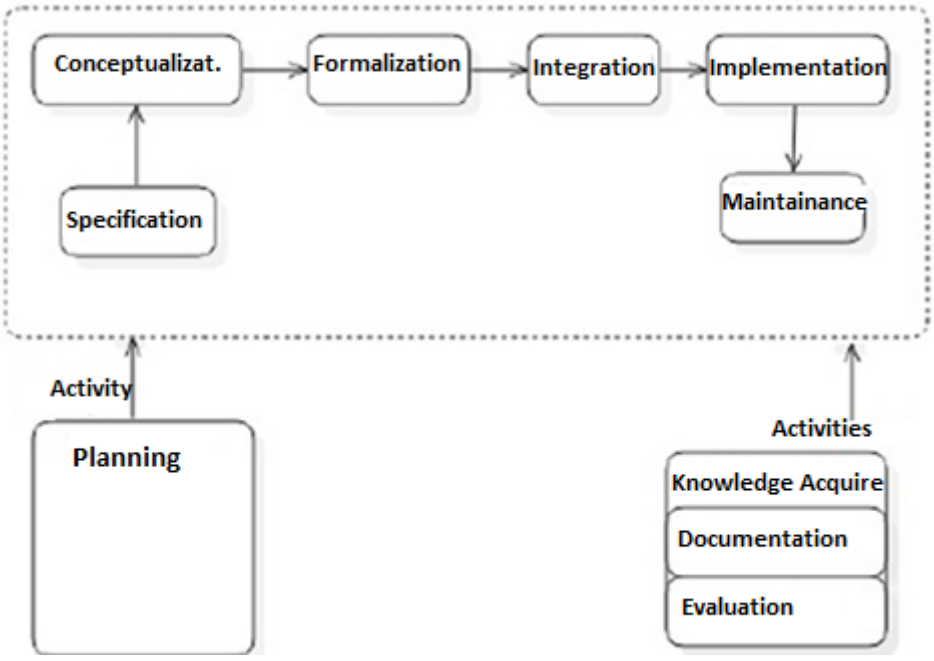

Fig.6.3: The procedure of ontology construction in OntologyDevelopment101

BUPT

6.4. Evaluation

Various instances of attacks and vulnerabilities have been tested using an application-level firewall. The proposed system is a new approach to the application of semantic technologies within the computer security field. The ontological model is kept inside a knowledge base from where it is accessed by the firewall using inference in order to detect new events. The system produced detection rates that are comparative (especially for the first 10 attacks mentioned by OWASP) with some of the best security solutions that currently exist today, like Snort or ModSecurity. The use of semantic rules allows the model to be more efficient in execution time by providing substantial reductions in search space and the result of a smaller false positives rate.

In this case, for evaluation I used OntoClean. This is a methodology for analyzing of ontological models of information systems based on some formal meta-properties, independent of the classes domain [93]. OntoClean offers formal reasoning about the common mistakes in the process of ontology design and helps to ontology validation by exposing the improper modeling choices and inconsistencies, removing of incorrect relations from the model by assigning meta-properties to each concept and application of rules in order to diagnose the misuse of relations. As effect, it removes the incorrect relations and exposes the implicit assumptions.

Among the 15 criteria as many that are mentioned in the literature, I chose 8 which I considered being the fittest for my ontology, which I will present next.

1) The first criteria states that all information from the model (definitions and descriptions of classes, properties, axioms) must be **accurate** and **valid** from the point of view of formality, in accord with some well-established standards from the domain in cause. The formal Correctness and validity of the ontology is assured by the OntClean methodology. OntoClean rules have been applied over classes and properties to ensure the model's correctness and meta-properties like *Rigid, Identity, Unity* to validate the taxonomical relations.

2) A model is **consistent** if all its relations are conforming with its characteristics. This means that all equivalence relations are reflexive, symmetrical and transitive; all inclusion relations are irreflexive, asymmetrical and transitive; all disjunction relations are reflexive, symmetrical and transitive. The model's consistency has been assessed by using a reasoner, the goal was to verify the uniformity and correctness of the knowledge base, data instances, assertions etc. In the current case I used FaCT++ , HermiT and Pellet, the most efficient proved to be FaCT++ which had done the above tasks in almost half of the time (approx. 312ms) compared to the others that had yielded 656ms and 858ms, respectively.

3) **Completeness** metric determines if the ontology covers accordingly the domain under consideration. The ontology proposed here, as it had already been affirmed in this chapter, was designed in the purpose to be a large model in order to capture as much information as possible from the domain so that the IDS that uses it to perform detection is efficient as possible. Moreover, it does not contain redundant semantics for terms and irrelevant axioms, only the essential ones are kept.

4) The proposed model is **expandable**, being possible to add with minimum effort new knowledge from the domain using the procedure of semantic alignment of ontologies. The Model can be deployed and reused for attacks detection by an IDS.

5) The Ontology is **clean**, each entity (concept, property, axiom, rule, description) is well specified and documented in natural language in order to be better understood, analyzed, reused, manipulated by users.

6) **Computaţional complexity, integrity and efficacy** measure the model's efficiency in terms of resources consumed, time restrictions, consistency checking etc. In order to apply this metric, I processed my model using an inference engine, FaCT++, which is a reasoner whose performance in terms of inference times is superior to the others. The ontology has been designed s.t. to avoid the generation and application of new rules for each newly occurred situation, a fact that would negatively affect the performance of the system. Any change inside the model may create new rules or regenerate the existing rules instances. Moreover, rules and semantic constraints of the ontology are applied to the concepts and properties, unlike traditional security methods based on rule signatures that work by capturing of only some characters or words and are prone to false positives.

7) **Performance** of a system is measured in terms of *Precision* and *Recall*. Precision is used for measuring the exactness, and recall for the completion. In the current situation, precision has been used for determining the number of actual attacks detected out of the generated alerts (incl. false positives), while recall to measure the number of detected attacks out of the total of generated alerts (incl. false negatives). There is also a third metric, *F-Measure*, in order to interpret the accuracy of a system that is computed as the ponderate sum of the first two. Adapted to my detection scenario, these are computed as follows:

$$Precizie = \frac{RataDetectie}{RataDetectie+NrPozFalsi} \quad (1)$$

$$Recall = \frac{RataDetectie}{RataDetectie+NrPozFalsi} \quad (2)$$

$$F - Measure = \frac{2*Precizie*Recall}{Precizie+Recall} \quad (3)$$

8) Task orientation: this criterion ensures that the model fulfills the functional requirements for which it had been developed. This fact will be proved in the next section, where will be shown details about the actual use of the ontology as part of the detection system.

BUPT

6.5. Deployment

The proposed system is a new approach for the application of semantic technologies inside the domain of information security. Various instances of attacks and vulnerabilities are tested by means of an application-level prototype of a firewall. The ontology is stored in the firewall's knowledge base from where it is accessed by use of inference and rules in order to detect the nature of newly occurred situations. Use of semantic rules makes the model be more efficient concerning time because it provides a substantial reduction in search space and yields small rates of false positives. The proposed system yielded detection rates that are comparative with some of the currently best existing systems, like Snort and ModSecurity. In fig.4 can be seen the architecture of the detection system in which are stated the main modules (components).

Next I will try to explain the detection process by the firewall. I will take as example an attack of type cross-site script (XSS) that a user injects into an application in encoded form.

$$\%3Cscript\%3E\%20alert(\%22This\%20is\%20cross\%20site\%20)script\%22)\%20\%$$

$$3C\%2Fscript\%3E\%20site\%20)script\%22)\%20\%3C\%2Fscript\%3E$$

The input field is being verified by the Parser module of codification, and in case it is then will be decoded. After decoding, the above string will look like the following:

$$< scrip > Alert("This\ is\ cross\ site\ scripting") </script >$$

After Normalization module, where it is being transformed and arranged by certain scales of the system, the request is being sent forwards to Protocol Validator and Analyzer modules.

In these 2 two modules, the request is being unified with the semantic rules that are generated by the ontological model from the knowledge base in order to identify potentially malicious content in the input message. The Protocol Validator is responsible for the violation of protocol specifications and the Analyzer for other types of attacks. If the content of the input matches any of the generated rules then the input is blocked and it is made a description of the detected attack. Below is shown an example of rule that is generated by the IDS out of the ontological model.

$$[rule10: (?x\ rdf:type\ ex:HTTPRequest)\ (?y\ rdf:typeex:ResponseHeaders)\ (?z\ rdf:type$$

$$ex:ResponseSplitting)(?x\ ex:hasRequesHead\ ?y) \rightarrow (?x\ ex:hasAt\ ?z)]$$

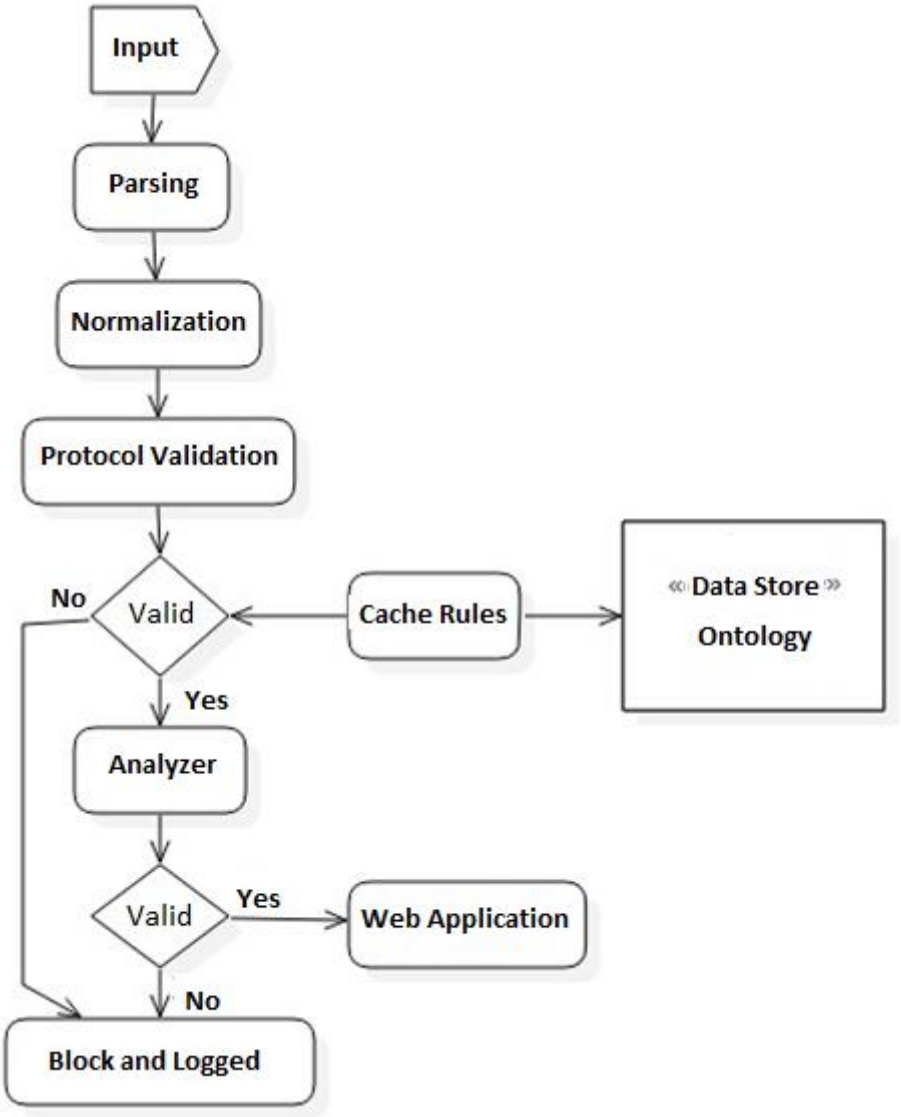

Fig.6.4: Ontology usage by the firewall

6.6. Tests and Results

In order to test the semantic model in the detection of different types of attacks in computer networks I used the Kyoto2006+ dataset [186], which contains information about attacks captured during a period of 3 years in the networks of University of Kyoto, Japan.

Tests had shown that the minimal detection rate of my system was close to that of Snort, that is 90% (little under Snort) and a false alarm rate of 0.6%, which is also comparable to that of Snort, 0.35%. Table 1 presents the evaluation results of the proposed system on the 3 performance metrics, together with those of Snort and ModSecurity in the attacks detection.

Metric / Solution	Precision	Recall	F-metric
My system	0.9050	0.8994	0.90713
Snort	0.9225	0.9031	0.9127
ModSecurity	0.8990	0.8750	0.8868

Table 6.1: Comparison of the proposed system with the existing security solutions

6.7. Conclusions

Many detection techniques had been created until present that struggles to keep pace with the increasing inherent complexity of applications, protocols and networks, which reverberates also in the increase of number of the attacks that can exploit them. Security frameworks that are built following an ontological approach represent the next generation of defense systems that present a series of advantages over the conventional techniques due to the fact that they are capable to capture the context of information and filter based on some well-established criteria.

In this chapter was proposed a method of creation of an ontology that can be used at enhancing the detection capacity of the attacks at all levels. The ontology is capable of increasing the capacity of the IDS of intrusion detection, providing a means to share a common understanding of attacks and the creation of signature rules. The use of ontologies and the OWL language in the detection of attacks yield a series of benefits, like:

i) gather semantic knowledge of the domain;
ii) better express of the IDS by constructing signature rules using the SWRL language;
iii) make reasoning an intelligent process

The ontology has been built and evaluated using state-of-art methodologies found in literature especially in this scope, like OntologyDevelopment 101 and OntoClean, and for its representation, I chose the standard ontology language of the Semantic Web, OWL2. The ontological model has been tested in the activity of attack detection using an application-level firewall that consults the ontology each time a suspect, previously unseen situation occurs. In the evaluation has been used the

testing dataset Kyoto2006+ of Kyoto University that was especially built for this scope.
The scores that resulted in the attacks detection by the proposed IDS were compared with another existing solutions: Snort and ModSecurity, these being sometimes above while others times below them for different types of attacks, the differences are negligible, though.

BUPT

page intended left blank

BUPT

7. INTRUSION DETECTION SYSTEM USING SEMANTIC WEB AND ARTIFICIAL INTELLIGENCE TECHNOLOGIES

In this chapter will be proposed an IDS based on recent and innovative techniques taken from the domains of Artificial Intelligence and the Semantic Web, as it was discussed in Chapter 4, with the aim to increase the efficiency in real attacks detection and to shrink that of errors. The proposed distributed IDS has as main aim the solvation of problems that centralized IDSs deal with, as it will be talked about during the course of this chapter, by employing a multi-agent architecture in which specific tasks are divided to each agent in part. Knowledge base is implemented as an ontology that captures the terms of a domain together with their relationships and is used by agents to find out the nature of detected events (normal data or attacks). One of the system's main functionalities is the possibility of detection attacks previously unseen (Zero-day), which is implemented in the agent that performs anomaly-based detection with the clustering algorithm and the ontology. The efficiency of the system was evaluated in terms of two functional requirements: scalability and detection capacity and had been compared with two other existing systems in the literature, a centralized and a distributed one.

7.1. Introduction

As the price of information processing and Internet accessibility shrinks more and more organizations become vulnerable to an increasingly high variety of attacks and cyberintrusions. Thus security in networks became a problem of critical importance. Consequently, software tools that are able to automatically detect a wide range of intrusions are of acute need. An Intrusion Detection System (IDS) has been used in order to detect and protect against attacks in a proactive manner and on a short period of time.

Although all IDSs are a fundamental component of the security infrastructures, they still have to face a big number of abridgments. They lack safety, relevance, disparity and completion in presentation and processing of the data, as well as the complexity of attacks. These things hinder the detection ability of the system since they cause the excessive generation of false alarms and shrinks the detection of the real attacks. Besides those, the big majority of IDS systems use a centralized

BUPT

arhitecture, which exhibits a series of problems, such are those that the central processing node may lead to a 'single point of failure, that is, anytime this is being attacked the whole IDS is at risk. Besides that, transferring all the information from the sensors randomly placed on the network to a central processing point puts great demands on the networkţs resources and can lead to its overload. Consequently, centralized IDSs suffer from the problem of scalability.

Moreover, communication and cooperation among the components of such a system are very low (or lack completely). In order to lean these problems a multi-agent architecture is used inside my IDS. The use of multi-agent systems for the intrusion detection offers a new alternative to IDS bringing together a series of benefits, as it has been stated in literature [52]:

- continuous and independent execution
- minimal overhead
- scalability

Thus, multi-agent technology increases systemţs resilience, ensuring its safety.

Simultaneously a new concept occurred, that of "ontology" as a powerful technique for the representation and sharing of a domain knowledge. It is capable to enhance the characteristics of intrusion detection by providing an ability to share a common understanding of the attacks and to design signature rules [2], [163].

In this respect, it is possible to design a multi-agent architecture on top of a knowledge base under the form of an ontology. This sort of architecture proved to be favorable for the development of IDSs.

In this research, I will propose a distributed IDS that uses a combination of techniques as multi-agent, ontology and clustering. In this purpose, my system uses a set of agents that are deployed to a number of tasks such as: data collecting, detection of the categories of attacks (known or unknown) and in the end alerting of the administrator. By means of some intensive experiments conducted on a network traffic from the real environment and a set of simulated attacks, it has been proved the efficiency of the proposed system in terms of scalability and detection capacity.

7.2. State of the Researches

Recently few approaches in the field of intrusion detection are dedicated to the integration of the multi-agent and ontologies technologies. The approaches from the distributed IDSs trend that rely on ontological structures proved to increase the IDS's accuracy, perform intelligent reasoning, and many others.

Must be noted that the first research about the application of ontologies in the intrusion detection field was the work of Jeffrey Undercoffer et al. [199] in the year 2003. Authors developed a target-centric ontology and represented it in the format of the DL language DAML+OIL. This ontology allows modeling of the domain of computer attacks and facilitates the process of reasoning in order to detect and prevent malicious situations.

(Mandujano et al.) [135] proposed a detection tool having a multi-agent architecture and an attacker-centered ontology, which they called FROID (First Resource for Outbound Intrusion Detection). This system tries to protect a set of nodes in a network by using the ontology OID, and is characterized by its intention

BUPT

to detect known attacks based on their signatures. Thus, the main drawback is that in case of the occurrence of a new attack the system will ignore it because it has not yet been registered in the database.

(Abdoli&Kahani) proposed a system called ODIDS [2]. This contains two types of agents: of IDS and a Master one. Based on the Semantic Web technologies, they built an ontology for the extraction of semantic relations among intrusions.
The main thing that can be imposed to this system is that the Master agent represents a single point of failure, and thus if a hacker can prevent it from functioning (e.g. halting or mitigation of host on which it executes) the entire system will be compromised. Another critic brought to this system is the high time consuming, a lot of time being required to make the connection between the Master and the IDS agents on the network and to send messages among them.

Azevedoln [20] constructed an autonomous model called AutoCore, which includes a set of intelligent agents as well as a domain ontology in order to make intrusion detection independently. The system uses the ontology as a knowledge base having high-level concepts as information. The agents are then responsible for the allowance of network traffic analysis and detection of malicious activities. The technique does not consider though the securing state, which is important in order to judge the false positives alerts and attack probability.

In [71] was constructed an IDS called MONI that relies on an ontological model. This system employs a multi-agent technique in order to achieve a distribution of detection activities. Besides this, MONI is enhanced with a case-based reasoning mechanism (CBR) in order to learn new attacks. Even if CBR is being considered a powerful reasoning paradigm and easy to be set up, it suffers from the problems of reverse engineering. This lack of flexibility in the knowledge representation is without doubt a limitation.

Having the same concern in mind, (Isaza et al.) [117] developed a multi-agent architecture for the detection and prevention of intrusions, called OntoIDSMA. The representation of known attacks was designed by means of an ontology-based semantic model that specifies rules for signatures and reaction. The authors integrated an ANN technique with the K-Means clustering algorithm for the detection of new attacks. The main drawback of ANNs is that the possibility to identify an intrusion is completely dependent on the system's training, data and methods being employed. Moreover, the configuration of a ANN is a delicate process and may affect the results in a significant manner. Besides these, the performances of K-Means and its efficiency as a detection method of new attacks depend on the random selection of the initial number of clusters. Thus, a bad choice for this number will have as effects the shrinking of detecting of the real number of intrusions and the growth of generating false alarms.

Because of its usability and importance, distributed intrusion detection is still a thriving problem. In this sense, the main objective of the current research is to create a hybrid distributed IDS that integrates:

a) ontology
b) multi-agent technique
c) unsupervised clustering algorithm

The basic idea of my approach is to address the limitations of centralized IDSs by taking advantage of the benefits of multi-agent and ontological paradigms.

BUPT

7.3. Proposed Distributed IDS

Multi-agent systems are one of the paradigms that are best suited for the attack detection in computer networks. Multi-agent technology distributes the resources and tasks, and thus each agent has its proper functionality, independent, fact which makes the system work faster.

IDS's architecture is consisted of a set of cooperative, communicating, and collaborative for the collecting and analysis of large traffic volumes and are called respectively: Sniffer, Misuse, Anomaly and Reporter. Figure 1 presents the conceptual scheme of the proposed IDS.

To be noted the fact that the combination of detection of known attacks and the unknown ones may lead to the increase of IDS's performance and its detection capacity. My IDS binds in an efficient manner the detection of both types of attacks. Contains an agent Misuse specialized on the detection of known attacks and an Anomaly one competent on the unknown ones. The steps of the IDS's working can be summarized as below:

a) Sniffer collects packets from the network; a distributed detection system must analyze a large volume of events gathered from different sources on the network. Thus, the Sniffer agent does the captured packets filtering which then converts to XML using the XStream library. Finally, the pre-processed packets are sent to the other agents for analysis
b) Misuse receives the converted XML packets from the Sniffer agent which transforms then in OWL format in order to be compatible with SWRL rules of the ontology, then it is ready to analyze the OWL packets in order to deduce those that correspond to the known attacks. This agent looks for attack signatures in the packets by consulting with the cyber-ontology. Consequently, if some similarity has been found between the OWL packets and SWRL rules that define the attack signatures then it creates an alert and sends it to the Reporter.
c) filtered packets from the network are sent to the Anomaly agent that relies on a clustering algorithm in order to detect the previously unknown ones; it sends an alert to Reporter if an unusual event was detected
d) Reporter agent generate reports and logs

The IDS detects the known attacks by means of Misuse agent that uses the attacks ontology to enrich the intrusion data and attack signatures with semantic relations.

BUPT

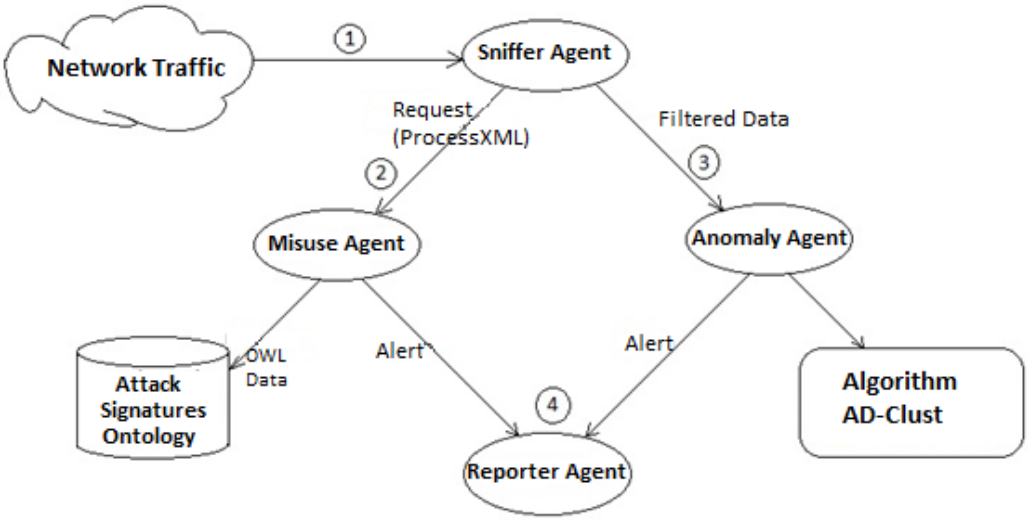

Fig.7.1: Conceptual schema of the proposed IDS

7.4. Attack Signatures Ontology

Since the last decades, (Raskin et al.) [161] opened a new field in research, one that deals with the use of ontologies in the domain of information security together with its inherent benefits. Ontologies are an extremely promising paradigm and powerful in the same time for this domain, they can be used as basic components for performing automated and continuous analysis based on some high-level policies that were defined for intrusions detection. Moreover, they provide the IDS an increased capacity for reasoning and analysis over data instances that represent intrusions. In addition, the property of ontology interoperability is essential for adaptation to the problems of systems distribution because cooperation between heterogeneous information systems are supported [20].

Inside the proposed IDS exists an ontology having the role to optimize the knowledge representation and embed more intelligence in content analysis. Moreover, the IDS integrate in its structure the interoperability among agents because they share the same ontological model. The proposed ontology is characterized by the network components, intrusion elements, classifications that define traffic signatures and the classes of rules. Fig. 2 presents a fragment of the ontology that contains the knowledge about intrusions. It allows the representation of the known attacks signatures database and is used by the Misuse agent. The power and usability of the ontology applied at the signature base problem offer a simple representation of the attacks expressed by means of semantical relations between intrusions data. May as well be inferred new knowledge about intrusions due to the ontology's capability to reason over the data and determine new behaviors. This thing enhances the decision-making process by the IDS.

BUPT

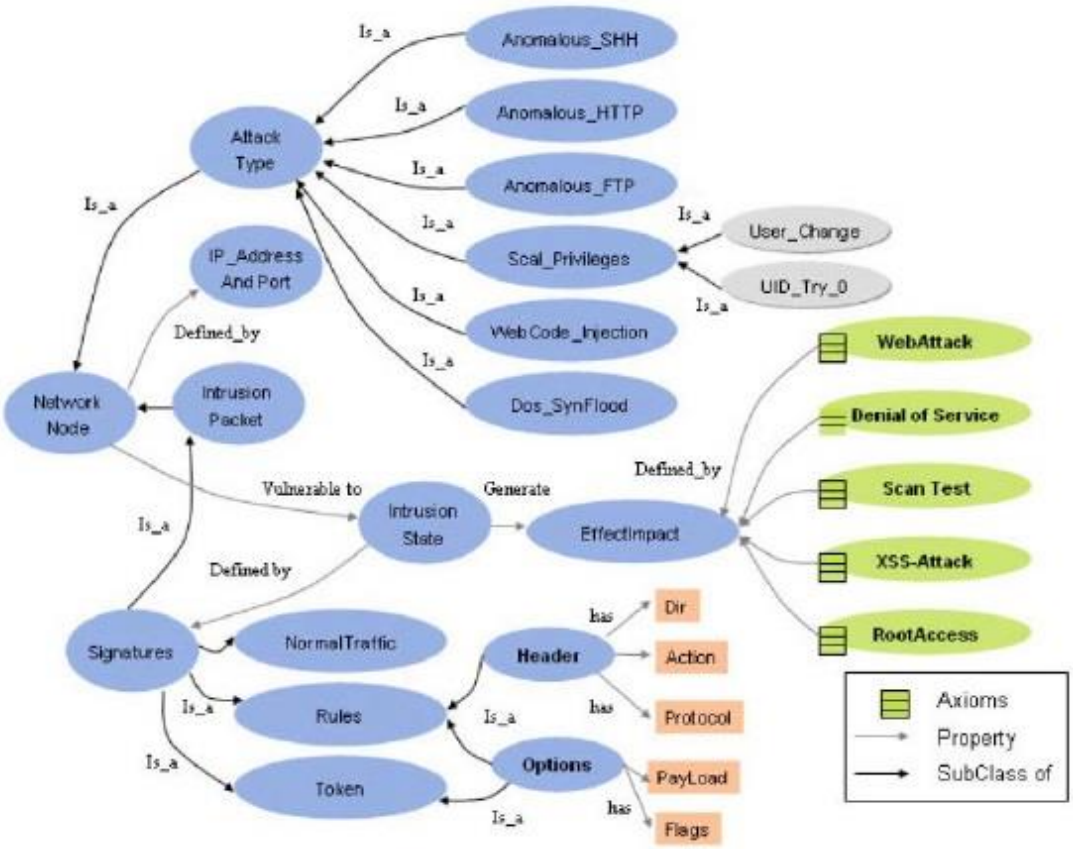

Fig.7.2: The attack signatures ontology

The signatures base contains the rules offered by the ontology, that allow a semantic representation for inference and reasoning. Rules are being extracted with the Semantic Web's language SWRL. This extends the ontology and enriches its semantics by deductive reasoning capabilities. It can process instances with variables of the form (?x, ?y, ?z). SWRL rules are constructed according to the schema: $Premise \rightarrow Consequence$, where both sides are conjuncts of atoms of the form: $a_1 \wedge a_2 \ldots \wedge a_n$. Variables are indicated by prefixing them with the question mark. The following example shows a rule represented in SWRL language:

$$NetworkHost(?z) \wedge IntrusionState(?p) \wedge GeneratedBy(?p,?z) \wedge SQLInjection(?p) \wedge DirectedTo(?p,?z) \rightarrow SystemSqlInjectionState(?p,?z)$$

Using this syntax, a rule which states that a combination of the properties: host on the network (z) and an intrusion state (p) has as result the attack property *SQL Injection*.

When I built the ontology, I designed and implemented multiple rules to define various attack signatures. These rules make possible the process of property inference and reasoning. Attack properties, e.g. WebAttack, SQLInjection, DoS, DDoS are defined as attributes in ontology that identify the type of intrusions.

Even if known attacks can be easily detected, the problem of detecting new ones, never seen before, remains. For this purpose, in addition to the Ontology agent using the ontology, the IDS also uses an Anomaly agent based on clustering analysis, which is discussed in detail in the next section.

7.5. Clustering Detection Algorithm

It is important to remember that data mining techniques applied in the field of intrusion detection can improve the accuracy of detection, speed and strengthen the security of the system [19]. Thus, Agent Anomaly represents the crossover of crystallized multi-agent and clustering techniques in the form of an algorithm. The idea behind this technique is that the amount of normal connection data is generally much higher than that of intrusions. When this occurs, attacks and anomalies can be detected based on the size of the clusters, ie large clusters correspond to normal data and the rest of the points to attacks, and are called single points (outliers).

The proposed algorithm is an unsupervised clustering algorithm built over a similar one, K-Means, in order to improve its quality when applied in the field of intrusion detection. The latter suffers from a long time complexity, which is an important factor in this area due to the large size of the packages. Moreover, the number of cluster dependencies and degeneration represent

weaknesses that make it difficult to use the K-Means algorithm to detect anomalies. The algorithm we propose combines two main categories of clustering: based on distance and density. Make use of the advantages of one to meet the limitations of the other, and vice versa

The steps of the algorithm workaround are given below:

a) extraction of density clusters considered centers of initial candidate clusters. This method is used as a pre-processing step

b) calculates the Euclidean distance between the candidate cluster center and the instance that will be assigned to the nearest cluster. For an xi instance and a day cluster center, the Euclidean distance is defined as:

c)

$$distance(\mathrm{x_i}, \mathrm{z_i}) = \sqrt{\sum_{i=1}^{n} (x_i - z_i)^2} \qquad (1)$$

The size of an instance neighborhood is specified by an input parameter. We will call this k 'to distinguish it from the k used by the K-Means algorithm. Thus, k 'specifies the minimum number of instances in a neighborhood and controls the granularity of the final clusters of the density-based method. If set to too high then few large clusters are found. To reduce the number of candidate clusters k 'to the

BUPT

expected one k we can iteratively unify two of the most similar clusters. If k 'is set too low then many small clusters will be generated. The clusters will be divided, new ones will be created to replace the empty ones and the courts will be reassigned to the new centers. This process will continue until there are no more empty clusters. Finally, the isolated points of the clusters will be removed to form new clusters in which the courts are more like each other. In this way, the value of the centers of the initial clusters k will be determined automatically by dividing or reuniting the clusters.

In the detection phase, our algorithm detects intrusions: for each instance it proceeds as follows:

- calculates the Euclidean distance and finds the cluster that has the smallest distance from i
- classify it by the category of the nearest cluster; if the distance between i and the cluster of normal data instances is the smallest then i will represent normal data, otherwise it is intrusion

7.6. Experiments and Results

To evaluate the performance of the proposed IDS in a realistic scenario, Sun's JDK 1.4 environment, JADE platform, JPCAP 0.7 and Eclipse development environment were used for implementation. The ontology for the attack signatures was made in Protégé 5.

In the experiments I tried to evaluate the performance of the proposed system in terms of:

- scalability: network bandwidth, detection delays, system response in time
- detection capabilities

During the evaluation process I compared the results of my system with those of the centralized IDS Snort and the multi-agent based on ontology, MONI. The experiments were performed on machines equipped with Pentium4 processors towed at 3GHz and 8GB main memory. We used machines that were connected via a switch, thus forming an interconnected network. We simulated the attacks using the Metasploit 3.5.1 tool. The types of simulated attacks are:

- Smurf (Denial of Service)
- Back Office (Backdoor)
- Spyware-Put Hijacker
- Nmap TCP Scan
- Finger User
- RPC Linux Statd Overflow
- DNS Zone Transfer
- HTTP IIS Unicode

7.6.1. Scalability Evaluation

To test the scalability of the proposed system, we studied the relationship between bandwidth consumption and a number of attacks. Furthermore, the variation of the detection delay according to the number of packets is studied. In addition, we will see how the response time varies with 8 types of attacks.

The maximum bandwidth consumed by the proposed system is 0.06Mb / s, and together with that of the MONI IDS it is smaller than that of Snort. This reduction in bandwidth consumption is due to the use of multi-agent technology. This is a desired feature in any distributed system. Figure 3a) presents the comparison of the 3 systems on bandwidth consumption; the proposed IDS is represented in blue color, Snort with yellow and MONI in red colors in the chart.

Part b) of Fig. 3 shows the detection delays depending on the number of packets. According to these results we can answer the question: "why IDS based on multi-agent techniques are better". The results clearly show that the detection delay of the 2 systems increases linearly with the number of packets. Moreover, the gap between the delay curves of our system and that of MONI is small because they both use the same technique. In addition, it can be deduced that our system is faster than Snort, which can be explained by the fact that agents operate directly on the host whenever an action needs to be taken, and their responses are faster than centralized systems where actions are performed by the central controller, as in the case of Snort. The color codes are preserved, as that of part A).

Part C) illustrates the response time required by the proposed system with respect to the types of attacks. On the one hand, it is noted that the detection of all attacks, on average, results in poorer response times compared to those of Snort due to its centralized detection engine. This shows how fast our system responds. On the other hand, in the MONI system the model

The ontological basis is developed in JADE, while ours was created in Protégé and queried using SWRL. The response time of our system is better than that of MONI due to the fact that in my system the inferred model is calculated only once before the matches start and is used in all queries. Figure 3 shows that my system outperforms MONI and allows the semantics of ontology to be exploited.

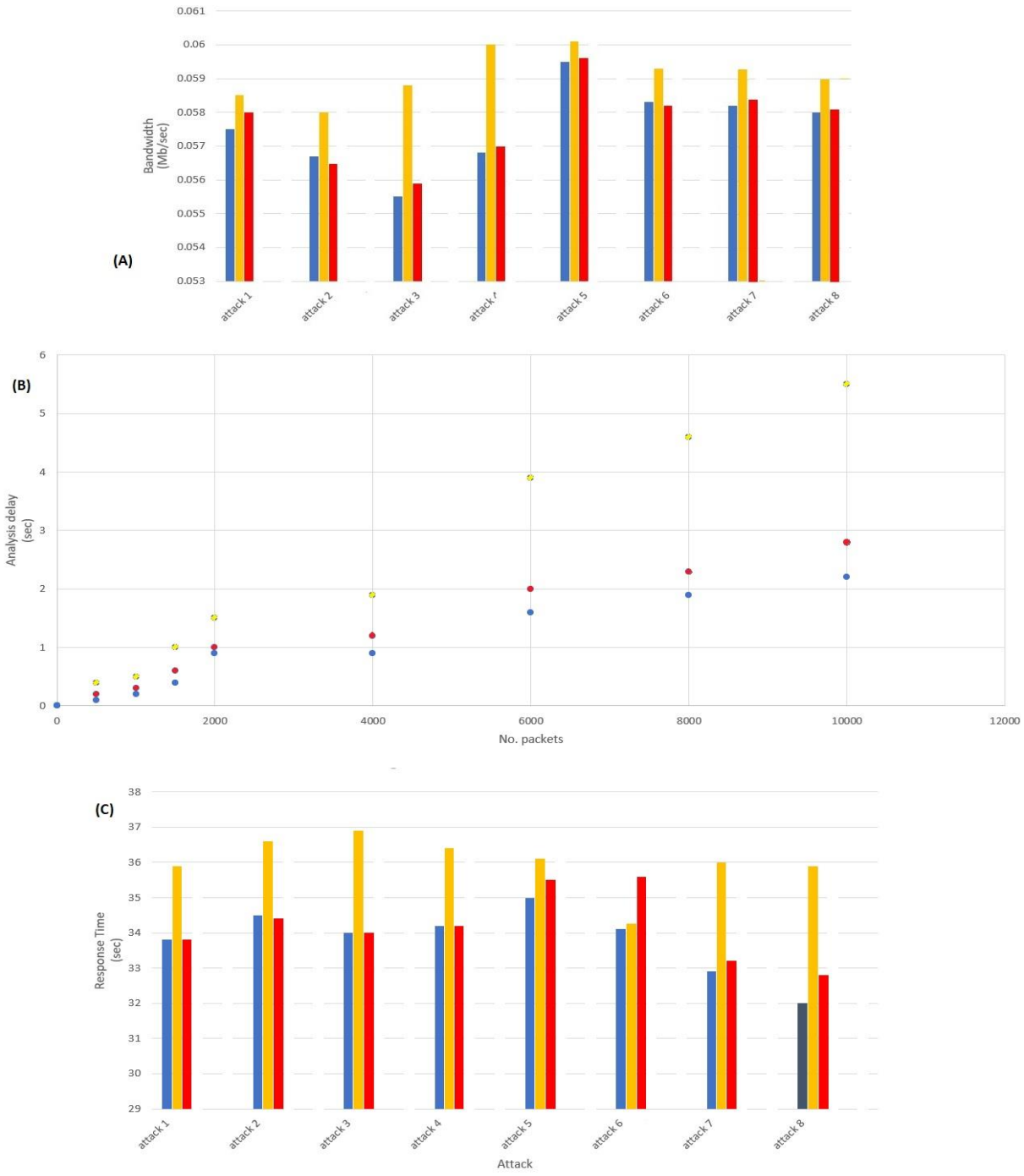

Fig.7.3: Bandwidth consumption, analysis delay and response time of ech IDS

In conclusion, it can be deduced from the results that the performance of the proposed system will not deteriorate too much with the increase in the number of attacks, which is justified by the low bandwidth consumption, short detection delays and fast response times. Also, when multiple machines are connected to the network, the IDS will support the load and deliver the answers quickly.

7.6.2. Evaluation of Detection Capacity

In order to evaluate the detection ability of an IDS usually two metrics are used [163]:

- detection rate (DR)
- false positives rate (FP)

The detection rate is the number of intrusions that were correctly detected. In contrast, the false positive rate is the total number of normal courts that have been incorrectly classified as attacks. Therefore, the value of DR must be as high as possible and that of FP as low as possible.

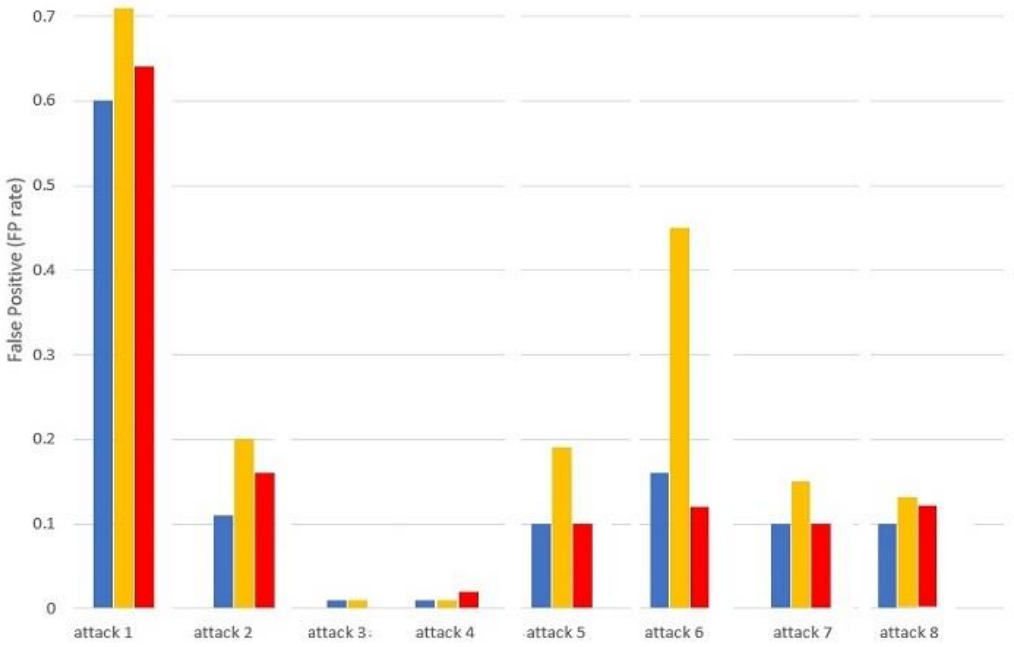

Fig.7.4a): False Positive rates of each IDS

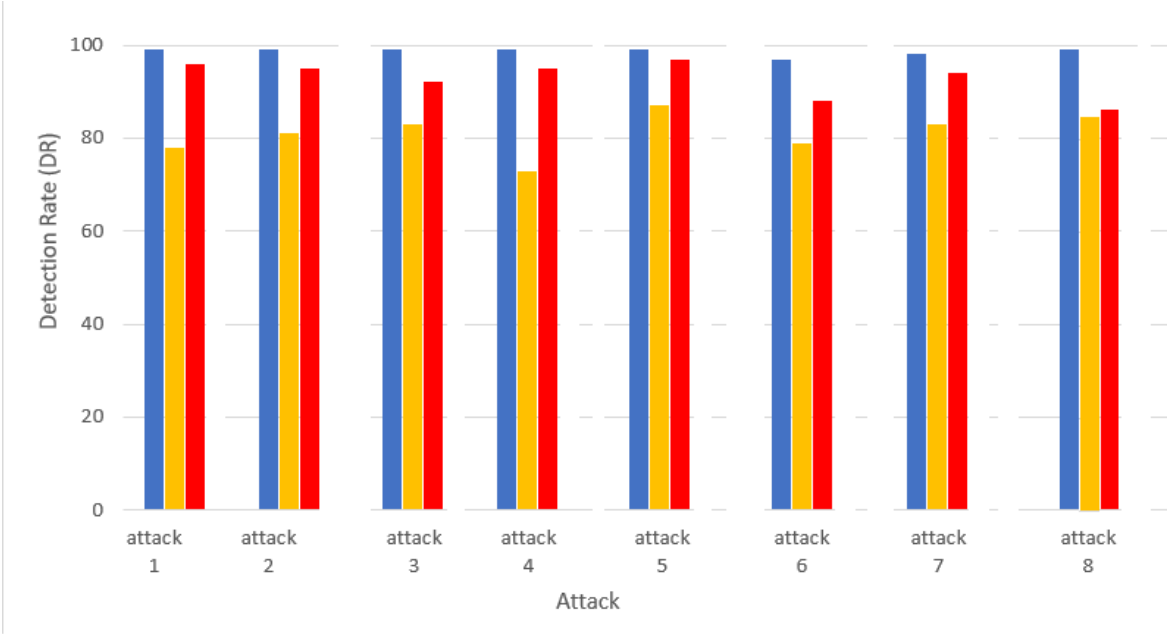

Fig.7.4b): Detection rates of each IDS

Figure 4a) shows that the FP rate of the system and that of MONI is significantly lower than that of Snort. This is due to the adaptive mechanisms used by the agents, allowing the systems to adapt better to the environment. As a result, the number of false alarms is reduced accordingly. For example, for attack type 3, Snort's FP rate can be as high as 0.019%, compared to 0.007% and 0.005% for MONI and my system.

In part b) we can see that the DR rate of my system is higher than that of MONI. Moreover, of the three IDS evaluated, Snort has the lowest value of this rate. For example, for attack 3, my system and MONI have 97.9% and 94.9%, respectively, and Snort's is 74.1%, due to its centralized architecture. Knowing that one of the main problems of today's IDS is the decrease in the number of false alarms, the main contribution is the decrease in the number of false alarms and at the same time maintaining a good detection rate.

7.7. Conclusions

Due to its usability and importance, distributed intrusion detection is a booming issue. Centralized IDS systems have been shown to suffer from a number of problems, such as high rates of false positives and negatives, low detection efficiency, low scalability, and especially when dealing with distributed attacks in network environments. . This is because the central processing node can lead to the problem of the "single drop point" and the transfer of all data from the nodes to a central processing point puts great demands on network transport and can lead to overload if there is no means of management.

In this chapter I have investigated some ways to address the above issues. For this purpose I proposed a hybrid multi-agent IDS based on a combination between an ontological model and a clustering technique. Multi-agent technology has the advantage that resources and tasks are distributed to multiple agents, each with its own independent functionality, which increases the speed of system execution. Our system contains 4 agents for performing the main detection tasks: Sniffer, Anomaly, Misuse, Reporter. The proposed grouping (clustering) detection algorithm considers large clusters as normal connection data and outliers are attacks. It is an unsupervised type, uses two types of clustering: distance and density, and is built on top of the well-known K-Means algorithm, modified for application in intrusion detection.

Experiments were performed to evaluate the performance of the proposed system in terms of scalability and detection. For the first time, we studied the relationship between bandwidth consumption and the number of attacks, the variation in the detection delay depending on the number of packets, and also the response time in respect of 8 attacks. For the second we used two metrics: the detection rate and the false positive rate. I compared the performances of the created system with those of 2 other similar systems: Snort and MONI, the results obtained clearly showing the superiority of my system.

BUPT

page intended left blank

BUPT

8. FINAL CONCLUSIONS AND FUTURE DIRECTIONS OF THE RESEARCH

This chapter draws the curtain over the work that has been done in this thesis and discusses a few possible directions for the development, other than those presented.

8.1. Conclusions

Even since the beginning of this thesis, the following two research problems had been stated:

(1) how is possible to contribute at the development and standardization of the third generation of World Wide Web recently occurred, namely Web3.0, known also as the *Semantic Web*

(2) how is possible to enhance the performance of cyber-defense systems for the detection of attacks in order to increase their rate of detecting for the real attacks (real positives) and slow down the number of false alerts (false negatives)

These two goals, even though at first glance they seem to exist in two different spheres of domains and don't have anything in common, this is not true, as it will be seen below. Based on these two objectives I divided this thesis into two big parts, each brings contributions to the respective domain.

First part's contributions are the ones below.

Chapter 1:

- initiates the reader into the main domain of the thesis, the Semantic Web; here have been presented the general notions about this, as its role in driving the evolution of the Classic (traditional) Web by creation of technologies that support the desired functionalities
- I made analyzes, comparisons between technologies of the classical Web and those newly created of the Semantic Web taking into consideration multiple criteria

BUPT

- most important contribution of the chapter represents a literary study (state-of-art) in which I presented the most important things that had been achieved in literature in order to support the development of the Semantic Web, among the greatest one are development environments of SW applications, reasoning systems, tools for conversions from classical data formats to the semantic ones etc. With this occasion I provided the reader references to works from literature where he can find out more about what has been briefly discussed there, offering him the possibility to enrich its knowledge

Chapter 2:

- presentation of the ”bottom rock” of the Semantic Web, that is the logical formalisms for the representation of knowledge from an application domain. These formalisms are fragments from the First Order Logic that tighten the expressivity with the scope to maintain tractability of reasoning process, and they are called Description Logics
- I had made comparisons, analyzes among the most important DL languages created in literature, especially ones of the S family concerning their capacities of knowledge representation (constructors of concepts and roles), concrete examples were shown where I explained what are the effects of adding a new constructor (e.g. inverses of roles, nominals, etc) onto the complexity of reasoning
- I have proposed a state-of-art in which I discussed the most important achievements in the domain from the last 2 decades, in terms of languages created by researchers or the reasoning algorithms, giving the reader references where he can find out more information

Chapter 3:

- I proposed a reasoner that performs deduction operations over logical knowledge bases which contain an additional (ontological) layer, which is one which specifies restrictions that must be applied to facts in order for these to exist in the KB

The second main part of the thesis comprises the following chapters.
Chapter 4:

- represents the beginning of the second main part of this thesis and it is dedicated to Computers Security. This chapter is meant to be an introduction in which are narrated the fundamental notions of the cybersecurity domain
- as contributions worth mentioning a literary study in which were presented the most important works read by me in this domain and I provided the reader references where he can find out more

Chapter 5:

- I made a study about the newest and innovative technologies that are being used in the development of cyberdefense systems, the most important coming from the domains of AI and SW
- had been proposed models in the form of UML diagrams that graphically show the workaround of processes inside those detection systems

Chapter 6:

- was proposed a model of cybersecurity information under the form of a Semantic Web ontology that is being used by an IDS when it must find out the nature of new situations (normal or attacks)

Chapter 7:

- I proposed a distributed IDS for computers networks with the aim to solve the problems faced by centralized IDSs
- the proposed IDS in its detection process relies on AI and SW technologies, such as multi-agent, clustering algorithms and ontologies

8.2. Future Directions of the Research

Some future directions for the efforts done in this thesis to which I've been thought about are the ones below.

Chapter 2 stated that DL-based ontologies play a crucial role on the Semantic Web, being used for the specification of the semantics of Web resources and that are capable of exploiting reasoner engines that had been created for the DL KBs in order to facilitate understanding by the machines of those resource descriptions. Their universal acceptance has been hindered by two major factors:

- semantic incompatibilities between the ontologies standard representation language on the Semantic Web (OWL) and RDFS, the universal representation data language on the Semantic Web
- lack of support for customized datatypes and predicates in OWL

A big part in the domain literature says that the main ways by which someone can support the development of the Semantic Web is by constructing of vocabularies and ontologies that conceptualize a certain domain, and by creation of inference engines in a chosen programming language that can process the resource descriptions and facilitate machine understanding. Chapter 3 proposed a reasoned system that produces the saturation of KBs that are represented in the syntax of FOL and was implemented in the OO language Java. I would like to extend this work by implementing the reasoner also in other languages. Even though Java was chosen mainly due to its portability (so can be executed in heterogeneous environments), there exist agents on the Semantic Web that are written in a certain language and import the system's modules only if it's written in same language. Also it can be extended the system's business logic in order to parse also the syntax of other logics, such as Modal or DLs.

BUPT

Part 2 discussed about the innovative techniques borrowed from domains of AI and SW and can be used at enhancing the detection performance. For this I analyzed 7 of those techniques, and 3 were used in the construction of my IDS in chapter 7. Because AI is a science that contains a big number of crated technologies, I am willing to study also others (besides those five from chapter 5) and learn how to use them to developing of IDSs.
Among the most important, as it was stated by literature, are Artificial Neural Networks, Genetic Algorithms and Bayesian filters.

BUPT

page intended left blank

BUPT

page intended left blank

BUPT

APPENDIX

Appendix A (Chapter 2)

DL	Constructor	Subsumption	Satisfiability	Creator
$\mathcal{FL}^-$	$(\cap, \exists R, \forall R.C)()$	PTime-complete		Brachman & Levesque, 1987
$\mathcal{AL}$	$(\cap, \exists R, \forall R.C, \neg name)()$	PTime-complete		Schmidt-Schauβ & Smolka, 1991
$\mathcal{ALN}$	$(\cap, \exists R, \forall R.C, <> R)()$	PTime-complete		Donini, 1997
$\mathcal{AL}^-$, $\mathcal{AL}^\circ$		PTime-complete		Donini, 1999
$\mathcal{FL}^-(\cap)$		PTime-complete		Donini, 1991
$\mathcal{ELIRO}$	$(\cap, \exists R.C, \{o\})(\cap, -)$	PTime-complete		Baader, 1998
$\mathcal{AL}\varepsilon$	$(\cap, \exists R.C, \forall R.C, \neg name)()$	NP-complete	NP-complete	Donini, 1992
$\mathcal{ALR}$	$(\cap, \exists R, \forall R.C, \neg name)(\cap)$	NP-complete	NP-complete	Donini, 1991
$\mathcal{AL}\varepsilon\mathcal{R}$	$(\cap, \exists R.C, \forall R.C, \neg name)(\cap)$	NP-complete	NP-complete	Donini, 1991
$\mathcal{FL}^-\varepsilon$	$(\cap, \exists R, \forall R.C)(\cap)$	NP-complete	NP-complete	Donini, 1991
$\mathcal{ALU}$	$(\cap,\cup, \exists R, \forall R.C, \neg name)(\cap)$	coNP-complete	coNP-complete	Donini, 1997
$\mathcal{ALN}(-)$		coNP-complete	PTime-complete	Donini, 1999
$\mathcal{FL}^-(\cap, -)$			NP-complete	Donini, 1999
$\mathcal{FL}^-(\cap, ^\circ)$			coNP-complete	Donini, 1999
$\mathcal{FL}^-(^\circ, -)$			coNP-complete	Donini, 1999
$\mathcal{AL}()$			in resp. cu un set de axiome acicl.: coNP-hard	Calvanese, 1996
$\mathcal{ALC}$	$(\cap,\cup, \exists R.C, \forall R.C, \neg name)()$	PSpace-complete	PSpace-complete	Schmidt-Schauβ & Smolka, 1991
$\mathcal{AL}\varepsilon\mathcal{N}$	$(\cap, \exists R.C, \forall R.C, \neg name, <> R)()$	PSpace-complete	PSpace-complete	Hemaspaandra, 1999
$\mathcal{ALCNR}$	$(\cap, \exists R.C, \forall R.C, \neg name, <> R)(\cap)$	PSpace-complete	PSpace-complete	Donini, 1997
$\mathcal{ALN}(\cap)$		PSpace-complete	PSpace-complete	Donini, 1997
$\mathcal{ALU}(\cap)$		PSpace-complete	PSpace-complete	Donini, 1997
$\mathcal{ALC}(\cap,\cup,^\circ)$			PSpace-complete	Massaci, 2001

$\mathcal{AL}\varepsilon()$			w.r.t. a set of cyclic axioms: PSpace-complete	Calvanese, 1996
$\mathcal{ALN}()$			w.r.t. a set of cyclic axioms of the form (A ≡ C): PSpace-complete	Kusters, 1998
$\mathcal{AL}$			w.r.t. a set of axioms: ExpTime-complete	Schmidt-Schauβ & Smolka, 1991
$\mathcal{ALC}_{trans}$	$(\cap,\cup,\exists R.C,\forall R.C,\neg name)(\cup,\circ,*,id,-)$		ExpTime-complete	Vardi & Wolper, 1986
$\mathcal{ALCIQ}_{reg}$	$(\cap,\cup,\neg,\exists R.C,\forall R.C,\mu x.C[x],\{o\})(-)$		ExpTime-complete	Sattler & Vardi, 2001
$\mathcal{ALC}$ with concrete domains			w.r.t. a set of acyclic axioms: NexpTime-complete	Lutz, 2001
$\mathcal{ALC}(\cap,\cup,\neg)$			NExpTime-complete	Lutz & Sattler, 2001
	$(\cap,\cup,\exists R.C,\forall R.C,\neg,\{o\},<>R.C)()$		NExpTime-complete	Tobies, 2001
$\mathcal{ALCNR}$	$(\cap,\cup,\exists R.C,\forall R.C,\neg,<>R.C)\ (\cap)$		w.r.t. a set of acyclic axioms: NExpTime-complete	Buccheit, 1993
$\mathcal{FL}^-(\circ,=)$		indecidabila	indecidable	Schmidt-Schauβ, 1989
$\mathcal{FL}^-(\circ,\subseteq)$		indecidabila	indecidable	Schmidt-Schauβ, 1989
$\mathcal{FL}^-(\circ,\subseteq,-,$ func, R|c)		indecidabila	indecidable	Patel-Schneider, 1989
$\mathcal{U}$	$()(\cap,\neg,\circ)$	indecidabila	indecidable	Schild, 1989
$\mathcal{ALCN}(\circ,\cup,-)$, $\mathcal{ALCN}(\circ,\cap)$,			w.r.t. a set of axioms: indecidable	Baader & Sattler, 1999

Table 1: A list with some DL languages and complexity results

BUPT

Appendix B (Chapter 6)

High Level (General Concepts)	Middle Level (Middle Concepts - Concrete)			
Attack	Denial of Service	Network		Flooding
		Host		
		Distributed		
	Buffer Overflow			
	Mitnick			
	Malware Installation	Viruses		
		Trojans		
		Worms		
		Spies		
	Identity Theft	Guessing		
		Implementation Exploit		
	Input Validation	HTTP Request Partitioning		
		HTTP Response Theft		
		Paths Traversing		
		Buffer Overflow		
		Defects Injection		
Source	Geo Location			
	IP Address			
Target	Software	Operation System	Windows	
			Linux	
			Mac OS	
		Aplicatie	Server	Web
				Databases
				Email
			Client	
	User			
	Network	Application		
		Presentation		
		Session		
		Transport		
		Physical		
	Hardware	Computer Equipment		
		Network Equipment		
		Peripheral Devices		
Technologies	SQL			
	PHP			
	ASP			
	JavaScript			

Vulnerabilities	Personal Data		
	Security Level		
	Structuring	Human resources	
		Organization Reputation	
		Information Resources	
		Organization Goods	
	Weak Configuration	Implicit Settings	
		Unprotected Files	
		Redundant Parts	
	Insufficient Authentication	Lack of Control at Function Level	
		Falses at Navigation between Sites	
		Unvalidated Redirects and Forwards	
		Erroneous Authentication Mechanisms	
	Insufficient Input Validation	SQL, LDAP and Xpath injections	
		Cross-Site Scripting	
		OS Commands Injection	
	Faults in OS Kernel	Unrestricted Resource Consumption	
		Lack of Resource Optimization Mechanisms	
Consecinte	Extenuarea Resurselor		
	Privilege Gain		
	Data Theft		
	Modifications in System		
Contra-masuri	Design		
	Implementation		
	Configuration		
Severitate	Low		
	Medium		
	Large		
Knowledge Required for the Attacker	Services Investigation		
	Certain Knowledge for Software		
	Certain Knowledge for Hardware		
	Knowledge about Methods of Attack		
	No Required Knowledge		
Inquisition	Target does only certain operations		
	Access to target	Physical	
		Remote	
	Target Operation		
	No Required Inquisition		

Policies	Rules	Constraints	
		Inference	
		Atributes Relations	
	Validation	Sintactical	Length Restriction
			Field Restriction
			Value Selection
			Transfer Mode
		Semantical	Data Formats
			Types Conversion
			Attributes Relation
			Fragments Combination
Required Resources	Material		
	Financial		
	Humans		
	Time		
	No Resources Required		

Table 2: Class hierarchy of the ontology

Propriety	**Domain**	**Range**	**Restriction**
hasSimilarAttack	Attack	Attack	Unicity
hasTarget	Attack	Target	Unicity
isTargetOf	Target	Attack	Some
hasVulnerability	Target	Vulnerability	Unicity
isVulnerabilityOf	Vulnerability	Target	Unicity
hasInquisition	Attack	Inquisition	Some
isInquisitionOf	Inquisition	Attack	Some
hasRequiredResource	Attack	Resource	Some
isRequiredResourceOf	Resource	Attack	Some
hasRequiredKnowledge	Attack	Required_Knowledge	Unicity - Some
isRequiredKnowledgeOf	Required_Knowledge	Attack	Some
exploits	Attack	Vulnerability	Unicity - Some
isExploitedBy	Vulnerability	Attack	Some
implies	Attack	Method	Unicity - Some
isImpliedBy	Method	Attack	Some
hasConsequence	Attack	Consequence	Unicity - Some
isConsequenceOf	Consequence	Attack	Some
worksAgainst	Countermeasure	Attack Vulnerability	Some
isWorkedAgainst	Attack Vulnerability	Countermeasure	Unicity - Some
hasSeverity	Attack	Severity	Unicity - Some
isSeverityOf	Severity	Attack	Some
hasExploitProbabilities	Attack	Exploit_Probability	Unicity - Some

isExploitProbabilityOf	Exploit_Probability	Attack	Some

Table 3: Object properties

Property	**Domain**	**Interval**	**Characteristic**
ID	Attack	Number	Functional
name	Attack	String	Functional
nbOccurrences	Attack	Number	Functional
similarAttacks	Attack	String	Functional

Table 4: Data properties

Property	**Domain**	**Interval**
Description	Attack	String
Version	Attack	Number
State	Attack	String
Execution course	Attack	Literal

Table 5: Annotation properties

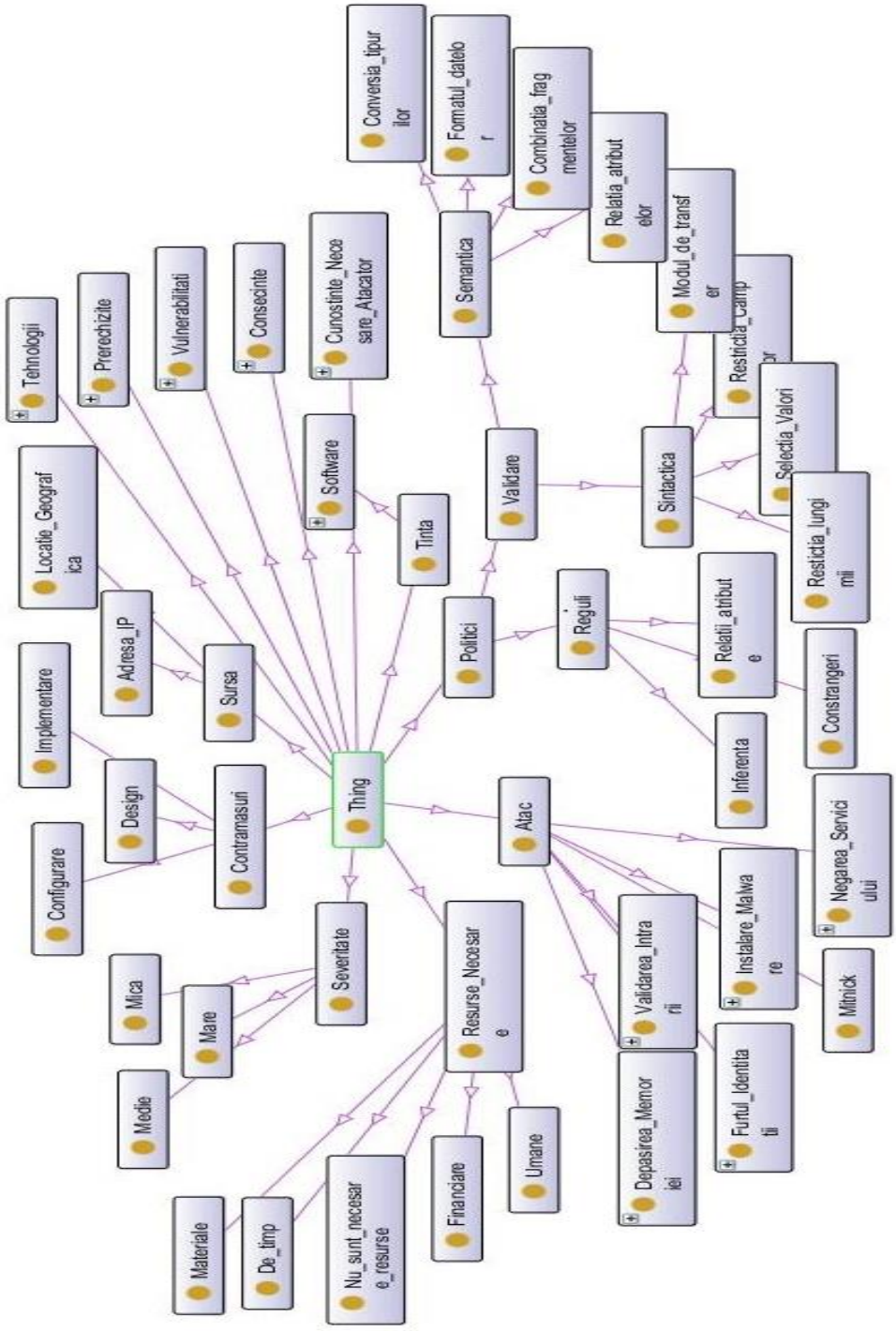

Fig.1: Visualization of ontology classes in Protégé environment

BUPT

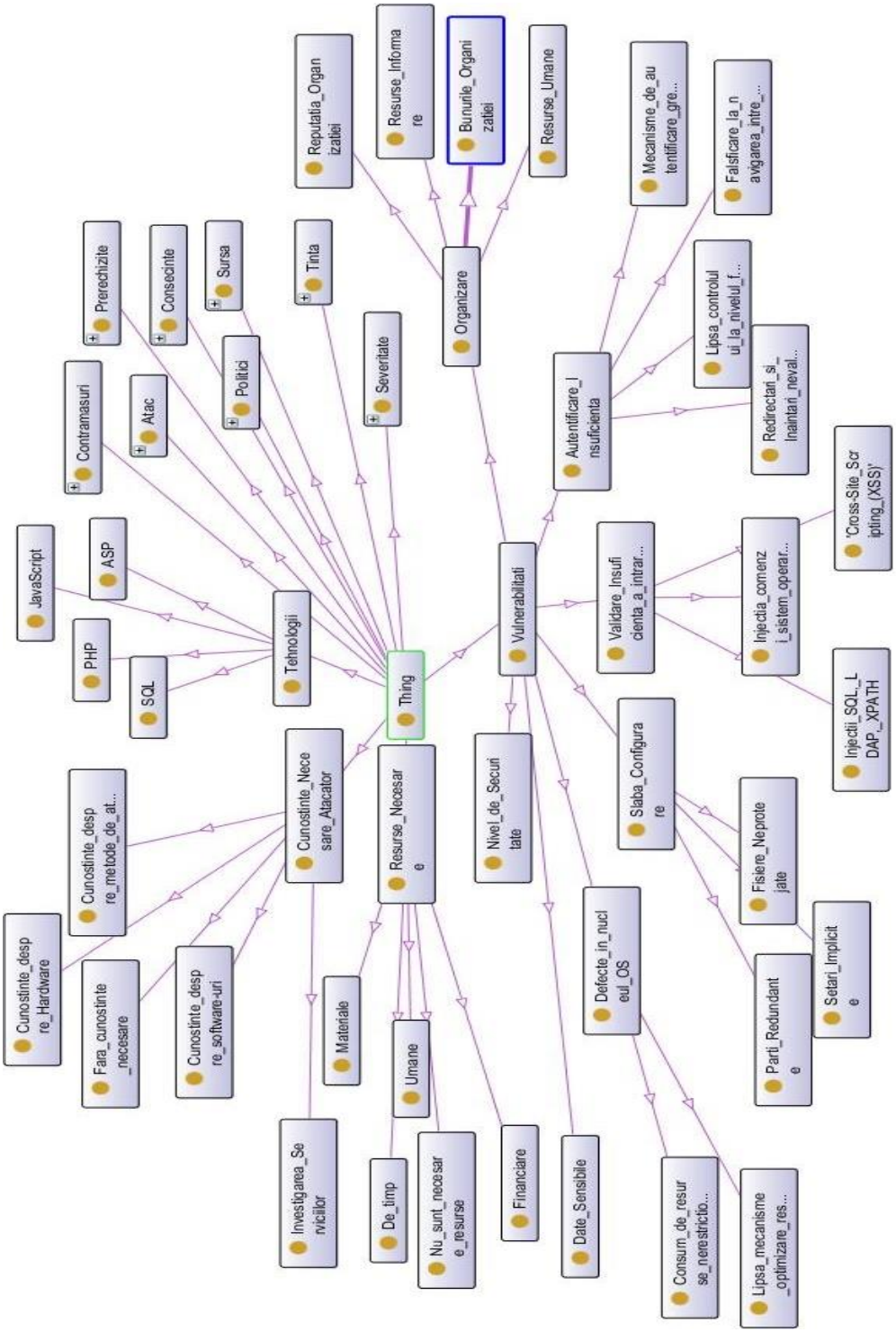

Fig.2: Visualization of ontology classes in Protégé

page intended left blank

BUPT

page intended left blank

BUPT

REFERENCES

[1] A.Abdallah, *Formalizing and Studying Dialectical Explanations in Inconsistent Knowledge Bases*, Phd Thesis, Universite de La Rochelle (2016)

[2] F.Abdoli, M.Kahani; *Ontology-based Distributed Intrusion Detection System*, Proceedings of 14th International Computer Conference (CSICC), Teheran, Iran (2009)

[3] B.Abdullah, I.Abd-Alghafar, G.Salama; *Performance Evaluation of a Genetic Algorithm-based Approach to Network Intrusion Detection System*, Proceedings of 13th International Conference on Aerospace Sciences and Aviation Technology (ASAT), Cairo, Egypt (2009)

[4] S.Abiteboul, R.Hull, V.Vianu; *Foundations of Databases*, Addison-Wesley Reading (1995)

[5] N.Agarwal, Z.Hussain; *A Closer Look at Intrusion Detection Systems for Web Applications*, Hindawi Journal on Security and Communication Networks, vol.2018

[6] A.Ahlawat, *The TCP/IP Reference Model*, https://www.studytonight.com/computer-networks/tcp-ip-reference-model (2011)

[7] A.Ahlawat, *Comparison Between OSI and TCP/IP Reference Models*, https://www.studytonight.com/computer-networks/comparison-osi-tcp-model (2011)

[8] A.Ahlawat, *Key Terms*, https://www.studytonight.com/computer-networks/key-terms-computer-networks (2011)

[9] L.Alhazzaa; *Intrusion Detection Systems using Genetic Algorithms*, Technical Report, King Saudi University (2002)

[10] L.Alton; *The Seven Most Important Data Mining Techniques*, https://www.datasciencecentral.com/profiles/blogs/the-7-most-important-data-mining-techniques, (2017)

[11] M.Alali, A.Almogren, M.Hasan, I.Rassan, A.Bhuyian; *Improving Risk Assessment Model of Cybersecurity using Fuzzy Logic Inference System*, Computers & Security, vol.74, Elsevier (2018)

[12] S.Al-Amro, D.Elizondo, A.Solanas; *Evolutionary Computation in Computer Security and Forensics: An Overview,* Computational Intelligence for Privacy and Security, vol.1, pp.25-34 (2012)

[13] S.Al-Janabi, H.Amjed-Saced; *A Neural Network-based Anomaly Intrusion Detection System*, Proceedings of 4th International Conference on Developments in eSystems Engineering, Dubai, UAE (2011)

[14] A.Ansari, T.Patki, A.Patki, V.Kumar; *Integrating Fuzzy Logic and Data Mining: Impact on Cybersecurity*, Proceedings of 4th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Haikou, China (2007)

[15] G.Antoniou, P.Groth, F.v.Harmelen, R.Hoekstra; *A Semantic Web Primer*, MIT Press (2012)

[16] M.d'Aquin, E.Motta, M.Sabou, S.Angeletou; *Toward a New Generation of Semantic Web Applications*, IEEE Intelligent Systems, vol.23,no.3, pp.20-28 (2008)

[17] S.Auer, J.Lehmann, S.Tramp, S.Hellmann; *Triplify: Lightweight Linked Data Publication from Relational Databases*, The 18th International Conference on World Wide Web (WWW), Madrid, Spain (2009)

[18] A.Aviad, K.Wecel, W.Abramowicz; *The Semantic Approach to Cybersecurity – Towards Ontology-based Body of Knowledge*, Proceedings of the 14th European Conference on Cyber Warfare and Security (ECCWS, 2015)

[19] C.Azad, V.Jha; *Data Mining in Intrusion Detection: A Comparative Study of Methods, Types and Data Sets*, International Journal of Information Technology and Computer Science (IJITCS), vol.5, no.8, pp.75-90 (2013)

[20] R.Azevedoln, E.Dantas, R.Santos, C.Rodriguez, M.Almeida, F.Freitas, W.Veras; *An Autonomic Ontology-based Multi-agent System for Intrusion Detection in Computer Environments*, International Journal of Informatics, vol.3, no.1, pp.1-7 (2010)

[21] F.Baader; *Introduction to Tableaux Algorithms for Description Logics*, https://lat.inf.tu-dresden.de/baader/Talks/Tableaux2000.pdf

[22] F.Baader; *Using Automata Theory for Characterizing the Semantics of Terminological Cycles*, Annals of Mathematics and Artificial Intelligence, vol.18, pp.175-219 (1996)

[23] F.Baader, S.Brandt, C.Lutz; *Pushing the EL envelope*, Proceedings of 19th International Joint Conference on Artificial Intelligence (IJCAI), pp.364-369 (2005)

[24] F.Baader, M.Buchheit, B.Hollunder; *Cardinality Restrictions on Concepts*, Artificial Intelligence, vol.88 (1996)

[25] F.Baader, I.Horrocks, U.Sattler; *Description Logics*, Handbook of Knowledge Representation, chapter 3, Elsevier (2008)

[26] F.Baader, R.Küsters; *Computing the Least Common Subsumer and the Most Specific Concept in the Presence of Cyclic ALN Concept Descriptions*, Proceedings of 22nd German Annual Conference on Artificial Intelligence (KI'98), Lecture Notes in Computer Science, vol.1504, pp.129-140, Springer, Berlin (1998)

[27] F.Baader, C.Lutz, M.Milicic, U.Sattler, F.Wolter; *Integrating Description Logics and Action Formalisms: First Results*, Proceedings of 20th National Conference on Artificial Intelligence (AAAI), MIT Press, pp.572-577 (2005)

[28] F.Baader, D.McGuiness, D.Nardi, P.Schneider; *The Description Logic Handbook: Theory, Applications and Implementations*, Cambridge University Press, New York, USA (2003)

[29] F.Baader, P.Narendran; *Unification of Concept Terms in Description Logics,* Proceedings of 13th European Conference on Artificial Intelligence (ECAI'98), pp.331-335. Ed. John Wiley&Sons, Brighton, UK (1998)

[30] F.Baader, A.Voronkov; *Logic for Programming, Artificial Intelligence and Reasoning*, Proceedings of 11th International Conference LPAR, Montevideo, Uruguay (2005)

[31] J.F.Baget, M.Leclere, M.Mugnier, E.Salvat; *On Rules with Existential Variables: Walking the Decidability Line*, Artificial Intelligence, vol.175, pp.1620-1654 (2011)

[32] Z.Bankovic, D.Stepanovic, S.Bojanic, O.Taladriz; *Improving Network Security using Genetic Algorithms Approach*, Computers&Electrical Engineering, vol.33 no.5-6, Elsevier Journal (2007)

[33] L.Banoth, M.Teja, M.Saicharan, N.Chandra; *A Survey of Data Mining and Machine Learning Methods for Cyber-Security Intrusion Detection,* International Journal of Research, vol.4 no.5 (2017)

[34] V.Barnett, T.Lewis; *Outliers in Statistical Data,* ISBN: 978-0-471-93094-5, Wiley (1994)

[35] D.Becket; *RDF/XML Syntax Specification Revised*, W3C Recommendation, http://www.w3.org/TR/rdf-syntax-grammar/ (2004)

[36] T.Berners-Lee; *WWW: Past, Present and Future*, IEEE Computer Magazine, vol.29 no.10 (1996)

[37] T.Berners-Lee; *World Wide Computer*, Communications of the ACM, vol.40, no.2 (1997)

[38] T.Berners-Lee; *Realizing the Potential of the Web*, Web-Weaving (chapter30), Butterworth-Heinemann (1998)
[39] T.Berners-Lee, R.Cailliau, J.Groff; *The World Wide Web*, Communications of the ACM (1994)
[40] T.Berners-Lee, R.Fielding, L.Masinter; *Uniform Resource Identifiers : Generic Syntax*, Network Working Group, Request for Comments 2396 (1998)
[41] T.Berners-Lee, M.Fischetti; *Weaving the Web*, Ed. Harper-Collins, San Francisco (1999)
[42] T.Berners-Lee, J.Hendler, O.Lasilla; *The Semantic Web*, Feature Article, Scientific American (2001)
[43] T.Berners-Lee, R.Swick; *Semantic Web Development,* Technical Report AFRL-IF-RS-TR-2006-294, New York (2006)
[44] T.Berners-Lee, J.Hendler, *From the Semantic Web to Social Machines: A research challenge for AI on the World Wide Web*, Journal of Artificial Intelligence, vol.174 no.2 (2010)
[45] O.Bernholtz, O.Grumberg; *Branching Time Temporal Logic and Amorphous Tree Automata,* Proceedings of International Conference on Concurrency Theory (CONCUR), Lecture Notes in Computer Science, vol.715, pp.262-277, Springer (1993)
[46] C.Bizer, A.Seaborne; *D2RQ: Treating non-RDF databases as virtual RDF graphs*, Proceedings of 3rd International Semantic Web Conference (ISWC), Hiroshima, Japan (2004)
[47] A.Borgida, P.Patel-Schneider; *A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic,* Journal of Artificial Intelligence Research, vol.1, pp.277-308 (1994)
[48] R.Brachman, H.Levesque; *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, California (1985)
[49] R.Brachman, H.Levesque; *Tractability of Subsumption in Frame-based Description Languages,* Proceedings of 4th National Conference on Artificial Intelligence (AAAI), Austin TX, USA (1984)
[50] R.Brachman, J.Schmolze; *An Overview of the KL-ONE Knowledge Representation System,* Journal of Cognitive Science, vol.9 no.2, pp.171-216 (1985)
[51] I.Brahmi, S.Yahia, H.Aouadi, P.Poncelet; *Towards a Multi-agent based Intrusion Detection System using Data Mining Approaches*, Proceedings of 7th International Workshop on Agents and Data Mining Interaction (ADMI), pp.173-194, Taipei,Taiwan (2011)
[52] I.Brahmi, H.Brahmi, S.Yahia; *A Multi-Agents Intrusion Detection System using Ontology and Clustering Techniques*, HAL Inria (2018)
[53] K.Brahmkstri, D.Thomas, S.Sawant, A.Jadhav, D.Kshiragar; *Ontology-based Multi-Agent Intrusion Detection System for Web Service Attacks using Self Learning*, Journal of Networks and Communications (NetCom), pp.265-274, Springer (2014)
[54] S.Brandt; *Polynomial Time Reasoning in a Description Logic with Existential Restrictions, GCI Axioms and What Else*, Proceedings of 16th European Conference on Artificial Intelligence (ECAI), pp. 298-302 (2004)
[55] M.Breunig, H.Kriegel, R.Ng, J.Sander; *LOF: Identifying density-based Local Outliers*, Proceedings of the National Information Systems Security Conference, 2000
[56] C.Braun; *KAON2 Algorithm: Non-tableau Reasoning with Description Logics*, Technical Report, University of Dresden (2006)

[57] M.Buchheit, F.Donini, A.Schaerf; *Decidable Reasoning in Terminological Knowledge Representation Systems*. Journal of Artificial Intelligence Research, vol.1, pp.109-138 (1993)

[58] A.Buji; *Genetic Algorithms for Tightening Security*, Master Thesis, University of Oslo, Oslo, Norway (2017)

[59] A.Cali, G.Gottlob, M.Kifer; *Taming the Infinite Chase: Query Answering under Expressive Relational Constraints*, Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning, pp.70-80 (2008)

[60] D.Calvanese, G.DeGiacomo, and M.Lenzerini; *Conjunctive Query Containment in Description Logics with n-Ary Relations*, Proceedings of the 1997 Workshop on Description Logic (DL'97), pp. 5-9 (1997)

[61] S.Ceri, G.Gottlob, L.Tanca; *What You Always Wanted to Know about Datalog*, IEEE Transactions on Knowledge and Data Engineering (1989)

[62] S.Ceri, G.Gottlob, L.Tanca; *Logic Programming and Databases*, Science and Business Media, Springer (2012)

[63] P.Chapke, R.Deshmukh; *Intrusion Detection System using Fuzzy Logic and Data Mining Techniques*, Proceedings of International Conference on Advanced Research in Computer Science Engineering and Technology (ARCSET), Unnao, India (2015)

[64] N.Choudhury; *World Wide Web and its Journey from Web1.0 to Web4.0*, International Journal of Computer Science and Information Technology (IJCSIT), vol.5, no.6 (2014)

[65] W.Cohen, A.Borgida, H.Hirsh; *Computing Least Common Subsumers in Description Logics,* Proceedings of 10th National Conference on Artificial Intelligence (AAAI'92), pp.754-760. AAAI Press/The MIT Press (1992)

[66] M.Croitoru, S.Vesic; *What can Argumentation do for Inconsistent Ontology Query Answering?*, Proceedings of the 7th International Conference on Scalable Uncertainty Management (SUM, 2013)

[67] M.Crosbie, E.Spafford; *Applying Genetic Programming to Intrusion Detection*, Proceedings of Association of Advanced Artificial Intelligence (AAAI), Fall Symposium on Genetic Programming, Cambridge, UK, pp.1-8, (1995)

[68] N.Dalwadi, B.Nagar, A.Makwana; *Semantic Web and Comparative Analysis of Inference Engines*, International Journal of Computer Science and Information Technologies, vol.3, pp.3843-3847 (2012)

[69] A.Deutsch, A.Nash, J.Rammel; *The Chase Revisited*, Proceedings of the 27th ACM-SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.149-158 (2008)

[70] L.Ding, P.Kolari, Z.Ding, S.Avancha; *Using Ontologies in the Semantic Web: A Survey*, Ontologies. Integrated Series in Information Systems, Springer, vol.14, pp.79-113, (2007)

[71] T.Djotio, C.Tangha, F.Tchangoue, B.Batchakui; *MONI: Mobile Agents Ontology-based for Network Intrusion Management*, International Journal of Advanced Media and Communication, vol.2, no.3, pp.288-307 (2008)

[72] F.Donini, F.Massacci; *Exp-Time Tableaux for ALC,* Artificial Intelligence, vol.124, no.1, pp.87-138 (2000)

[73] M.Drolet; *The Darwin Defense: Can Genetic Algorithms Outsmart Malware?*, https://www.csoonline.com/article/3237671/the-darwin-defense-can-genetic-algorithms-outsmart-malware.html (2017)

[74] P.M.Dung; *On the Acceptability of Arguments and its Fundamental Role in Non-monotonic Reasoning, Logic Programming and n-Person Games*, Artificial Intelligence, vol.77, pp.321-357, (1995)
[75] C.Elkan; *Fuzzy Logic Tutorial: What is, applications and examples,* https://www.guru99.com/what-is-fuzzy-logic.html (2006)
[76] J.Euzenat, J.Pin, R.Ronchaud; *Research Challenges and Perspectives on the Semantic Web*, Report of EU-NSF Strategic Workshop, Sophia-Antipolis, France (2001)
[77] M.Fahad, M.Qadir, S.Shah; *Evaluation of Ontologies and DL Reasoners*, Proceedings of 4th International Conference on Intelligent Information Processing, Beijing, China (2008)
[78] U.Fayadd, G.Piatetsky-Shapiro, P.Smith; *From Data Mining to Knowledge Discovery in Databases*, AAAI Journal, vol.17 no.3 (1996)
[79] Federal Trade Commission; *Internet of Things: Privacy and Security in a Connected World,* https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf (2015)
[80] D.Fensel, C.Bussler, Y.Ding, V.Kartvesa, M.Klein; *Semantic Web Application Areas*, Proceedings of 7th International Workshop on Application of Natural Language to Information Systems, Stockholm, Sweden (2002)
[81] D.Fensel, J.Hendler, H.Lieberman, W.Wahlster; *Spinning the Semantic Web: Bringing the Web to its Full Potential*, MIT Press (2003)
[82] M.Fischer, R.Ladner; *Propositional Dynamic Logic of Regular Programs,* Journal of Computer and System Sciences, vol.18, pp.194-211 (1979)
[83] N.Fouad, S.Hameed; *Genetic Algorithm based Clustering for Intrusion Detection*, Iraqi Journal of Science, vol.58 no.2, pp.929-938 (2017)
[84] S.Ganapathy, K.Kulothungan, S.Muthurajkumar, M.Vijayalakshmi, P.Yogesh; *Intelligent Feature Selection and Classification for Intrusion Detection in Networks: A Survey*, EURASIP Journal on Wireless Communications and Networking (2013)
[85] P.Garcia-Teodoro, J.Diaz-Verdejo, G.Macia-Fernandez, E.Vazquez; *Anomaly-based Network Intrusion Detection: Techniques, Systems, Challenges*, Computers and Security, vol.28, pp.18-28, Elsevier (2009)
[86] M.Garey, D.Johnson; *Computers and Intractability – A Guide to NP-completeness*, Ed. W.H. Freeman and company, San Francisco, California (1979)
[87] G.de Giacomo, M.Lenzerini; *Boosting the Correspondence between Description Logics and Propositional Dynamic Logics*, Proceedings of 12th National Conference on Artificial Intelligence (AAAI), pp.205-212 (1994)
[88] G.de Giacomo; *Decidability of Class-based Knowledge Representation Formalisms*, Phd thesis, Universita di Roma "La Sapienza" (1995)
[89] E.Grädel, P.Kolaitis, M.Vardi; *On the Decision Problem for Two-Variable First-Order Logic,* Bulletin of Symbolic Logic, vol.3,no.1, pp.53-69 (1997)
[90] T.Gruber; *Toward Principles for the Design of Ontologies used for Knowledge Sharing*, Workshop on Formal Ontology, book Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers (1993)
[91] N.Guarino; *Formal Ontology, Conceptual Analysis and Knowledge Representation*, International Journal of Human-Computer Studies, Special Issue: The role of formal ontologies in information technology, vol.43, pp.625–640, Academic Press (1995)
[92] N.Guarino; *Formal Ontology in Information Systems*, Proceedings of 1st Conference on Formal Ontology and Information Systems, pp.3–15, Trento, Italy (1998)

[93] N.Guarino, C.Welty; *Evaluating Ontological Decisions with OntoClean*, Journal of Communications of the ACM, vol. 45,no.2, pp.61-65, New York (2002)
[94] T.Ha, J.Sohn, Y.Cho; *OWLer: An Inference Engine for Ontologies on the Semantic Web*, Proceedings of 7th International Conference on Advanced Communication Technology, Phoenix, South Korea (2005)
[95] H.Halpin; *The Semantic Web: Origins of Artificial Intelligence Redux*, Proceedings of 3rd International Workshop on History and Philosophy of Logics, Mathematics and Computation (HPLMC-04), San Sebastian, Spain (2005)
[96] L.Han, T.Finin, C.Parr, J.Sachs, A.Joshi; *RDF123: From Spreadsheets to RDF*, Proceedings of 7th International Semantic Web Conference (ISWC), Karlsruhe, Germany (2008)
[97] F.v. Harmelen, V.Lifschitz, B.Porter; *Handbook of Knowledge Representation*, Foundations of Artificial Intelligence, Elsevier (2008)
[98] O.Hassanzadeh; *Introduction to Semantic Web Technologies and Linked Data*, Technical Report, Univ. of Toronto (2011)
[99] D.Heckerman; *A Tutorial on Learning with Bayesian Networks*, Microsoft Research Technical Report MSRTR-95-06 (2008)
[100] J.Hernandez-Castro, P.Isasi; *Evolutionary Computation in Computer Security and Cryptography*, New Generation Computing, vol.23, Springer (2005)
[101] M.Hewett, S.Lacoul; *A Hands-On Overview of the Semantic Web*, https://www.slideshare.net/shamod/a-hands-on-overview-ofthe-semantic-web, (2009)
[102] A.Hoffmann; *Artificial and Natural Computation*, International Encyclopedia of the Social and Behavioral Sciences, pp.27-31, Elsevier (2015)
[103] B.Hollunder; *Consistency Checking Reduced to Satisfiability of Concepts in Terminological Systems,* Annals of Mathematics and Artificial Intelligence, vol.18(24), pp.133-157 (1996)
[104] I.Horrocks; *Description Logics Reasoning*, Automated Reasoning with Analytic Tableaux and Related Methods (2005)
[105] I.Horrocks; *Description Logics Reasoning*, www.cs.man.ac.uk/horrocks/Slides/lpar05.ppt
[106] I.Horrocks; *Description Logics Reasoning*, www.cs.man.ac.uk/ horrocks/Teaching/cs646/Slides/pt3dlreasoning.pdf
[107] I.Horrocks; *Description Logics: A Formal Foundation for Ontology Languages and Tools- Part I*, https://www.cs.ox.ac.uk/ian.horrocks/Seminars/download/Horrocks_Ian_pt1.pdf
[108] I.Horrocks; *Description Logics: A Formal Foundation for Ontology Languages and Tools- Part II*, www.cs.ox.ac.uk/ian.horrocks/Seminars/download/Horrocks_Ian_pt2.pdf
[109] I.Horrocks; *Ontologies and the Semantic Web*, Communications of the ACM - Surviving the Data Deluge, vol.51, New York, USA (2008)
[110] I.Horrocks; *Description Logics: Formal Foundation for Ontology Languages and Tools*, Methods Cell Biolt, Oxford University Computing Laboratory (2007)
[111] I.Horrocks, F.Baader, U.Sattler; *Description Logics as Ontology Languages for the Semantic Web*, Mechanizing Mathematical Reasoning. Lecture Notes in Computer Science, vol.2605, Springer, Berlin, Germany (2005)
[112] I.Horrocks, F.van Harmelen, P.Schneider; *From SHIQ and RDF to OWL: The making of a Web Ontology Language*, Web Semantics: Science, Services and Agents on the World Wide Web, vol.1 (2003)

[113] I.Horrocks, M.Krotzsch, M.Simancik; *A Description Logics Primer*, Perspectives in Ontology Learning, chp.1, IOS Press (2014)
[114] I.Horrocks, P.Schneider; *Knowledge Representation and Reasoning on the Semantic Web: OWL*, Handbook of Semantic Web Technologies, pp.365-398, Springer (2011)
[115] I.Horrocks, D.Tsarkov; *FaCT++ Description Logics Reasoner: System Description*, International Joint Conference in Automated Reasoning, Seattle WA, USA (2006)
[116] H.Igor, J.Bohuslava, J.Martin, N.Martin; *Application of Neural Networks in Computer Security*, 24th International Symposium on Intelligent Manufacturing and Automation, Zadar, Croatia (2013)
[117] G.Isaza, A.Castillo, M.Lopez, L.Castillo; *Towards Ontology-based Intelligent Model for Intrusion Detection and Prevention,* Journal of Information Assurance and Security, vol.5, pp.376-383 (2010)
[118] K.Jackson; *Intrusion Detection Systems Product Survey*, Los Alamos National Laboratory Research Report, LA-UR-99-3883, New Mexico, USA (1999)
[119] J.Kaliapan; *Intrusion Detection using Artificial Neural Networks with Best Set of Features*, International Arab Journal of Information Technology. vol.12, no.6 (2015)
[120] H.Karande, P.Kulkarni, S.Gupta, D.Gupta; *Security against Web Application Attacks using Ontology-based Intrusion Detection System*, International Research Journal of Engineering and Technology (IRJET), vol.3, (2016)
[121] A.Khairkar; *Intrusion Detection System based on Ontology for Web Applications*, College of Engineering, Pune, India (2013)
[122] A.Knight; *The titans of AI and ML Arms Race in Cybersecurity*, https://www.aitegroup.com/report/titans-ai-and-ml-arms-race-cybersecurity, Aite Group (2019)
[123] A.Knight, *Top 10 trends in Cybersecurity, 2019: User Experience and Machine Learning*, https://www.aitegroup.com/report/top-10-trends-cybersecurity-2019-user-experience-and-machine-learning, Aite Group (2019)
[124] A.Knight; *A Cylance Case Study: Machine Learning in Insider Threat Incident Response,* https://www.aitegroup.com/report/cylance-case-study-machine-learning-insider-threat-incident-response, Aite Group (2018)
[125] A.Knight; *A Darktrace Case Study: ML Rising,* https://www.aitegroup.com/report/darktrace-case-study-ml-rising, Aite Group (2018)
[126] G.Kumar, K.Kumar, M.Sachdeva; *The Use of Artificial Intelligence-based Techniques for Intrusion Detection: A Review*, Artificial Intelligence Review, vol.34, no.4, Springer (2010)
[127] N.Kumar, N.Mehra; *Semantic Web Applications*, DESIDOC Journal of Library and Information Technology, vol.31, pp.217-225 (2011)
[128] P.LaRoche, A.Heywood; *802.11 Network Intrusion Detection using Genetic Programming*, Proceedings of 7th Workshp on Genetic and Evolutionary Computation (GECCO), pp.170-171, Washington, USA (2005)
[129] A.Lazarev, V.Kumar, J.Srivastava; *Intrusion Detection Systems: A Survey*, Managing Cyber Threats. Massive Computing, vol.5, Springer, Boston, USA (2005)
[130] J.Lee; *A Gentle Introduction to Neural Networks for Machine Learning*, https://www.codementor.io/james_aka_yale/a-gentle-introduction-to-neural-networks-for-machine-learning-hkijvz7lp (2018)
[131] D.Lembo, M.Lenzerini, R.Rosatti, D.Savo; *Inconsistency-Tolerant Semantics for Description Logics*, Proceedings of the 4th International Conference on Web Reasoning and Rule Systems, Bressanone, Italy (2010)

[132] P.Lloyd, P.Boyle; *Web-Weaving: Intranets, Extranets and Strategic Alliances*, Butterworth-Heinemann (1998)
[133] L.Ma, J.Mei, Y.Pan, K.Kulkarni, A.Fokone, A.Ranganathan; *Semantic Web Technologies and Data Management*, Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent Systems (IFAAMAS, 2013)
[134] P.Majeed, S.Kumar; *Genetic Algorithms in Intrusion Detection: A Survey*, International Journal of Innovation and Applied Sciences (IJIAS), Vol.5, no.3 (2014)
[135] S.Mandujano, A.Galvan, J.Nolazco; *An Ontology-based Multi-Agent Approach to Outbound Intrusion Detection*, Proceedings of 8th International Conference on Computer Systems and Applications (AICCSA), Cairo, Egypt (2005)
[136] D.Marinescu; *Nature-Inspired Algorithms and Systems*, Complex Systems and Clouds, pp.33-63, Elsevier (2017)
[137] B.Marnette; *Generalized Schema Mappings: from Termination to Tractability*, Proceedings of the 28th ACMSIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.13-22 (2009)
[138] D.McMorrow; *Science of Cybersecurity*, JASON Project, The Mitre Corporation, McLean, Virginia (2010)
[139] Z.Michalewicz, M.Michalewicz; *Evolutionary Computation Techniques and their Applications*, IEEE International Conference on Intelligent Processing Systems (ICIPS), Beijing, China (1997)
[140] N.Mkuzangwe, F.Nelwamondo; *A Fuzzy Logic-based Network Intrusion Detection System for Predicting the TCP SYN Flooding Attack*, Proceedings of 9th International Asian Conference on Intelligent Information and Database Systems (ACIIDS), pp.14-22, Kanazawa, Japan (2017)
[141] M.Mortimer; *On Languages with two Variables*, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, vol.21,pp.135-140 (1975)
[142] B.Motik, R.Shearer, I.Horrocks; *Hypertableaux Reasoning for Description Logics*, Journal of Artificial Intelligence Research, vol.36, pp. 165-228 (2009)
[143] D.Muller, P.Schupp; *Alternating Automata on Infinite Trees,* Theoretical Computer Science Journal, vol.54, pp.267-276 (1987)
[144] National Center for Biomedical Ontology, *Micro-Array and Gene Expression Data Ontology,* https://bioportal.bioontology.org/ontologies/MO, (2018)
[145] National Center for Biomedical Ontology, *Ontology of Biological Pathways,* https://bioportal.bioontology.org/ontologies/BP, (2010)
[146] B.Nebel; *Terminological Reasoning is Inherently Intractable*, Journal of Artificial Intelligence vol.43,no.2, pp.235-249 (1990)
[147] C.Nicholson; *A Beginner's Guide to Neural Networks and Deep Learning*, AI Wiki, https://skymind.ai/wiki/neural-network (2017)
[148] C.Nicholson; *A Beginner's Guide to Evolutionary and Genetic Algorithms*, AI Wiki, https://skymind.ai/wiki/evolutionary-genetic-algorithm (2017)
[149] N.Noy, D.McGuiness; *Ontology Development 101: A Guide for Creating your First Ontology*, Stanford Knowledge Systems Laboratory, Technical Report, KSL-01-05, Stanford, California (2001)
[150] L.Obrst, P.Chase, R.Markeloff; *Developing an Ontology of the Cyber-Security Domain*, Proceedings of 7th International Conference on Semantic Technologies for Intelligence, Defense and Security (STIDS, 2012)

[151] A.Oltramari, L.Cranor, R.Walls, P.McDaniel; *Building an Ontology of Cyber-Security*, Proceedings of the 9th Conference on Semantic Technology for Intelligence, Defense, and Security, Fairfax VA, USA (2014)
[152] L.Pacholski, W.Szwast, L.Tendera; *Complexity of Two-Variable Logic with Counting,* Proceedings of the 12th IEEE Symposium on Logic in Computer Science (LICS'97), pp.318-327, IEEE Computer Society Press (1997)
[153] S.Palmer; *The Semantic Web: An Introduction*, http://infomesh.net/2001/swintro/ (2001)
[154] J.Pan; *Desciption Logics: Reasoning Support for the Semantic Web*, PhD thesis, Univ. of Manchester (2004)
[155] J.Pan, C.Anumba, Z.Ren; *Potential Applications of the Semantic Web in Construction*, Proceedings of 20th Annual ARCOM Conference, Edinburgh, Scotland (2004)
[156] C.Papadimitriou; *Computational Complexity*, Addison-Wesley Publishing (1994)
[157] J.Parveen; *Neural Networks in Cybersecurity*, International Research Journal on Computer Science (IRJCS), vol.4,no.9 (2017)
[158] P.Patel, P.Trikha; *Interpreting Inference Engines for the Semantic Web*, International Journal on Advanced Research in Computer Engineering and Technology, vol.2 (2013)
[159] A.Poggi, D.Lembo, D.Calvanese, G. De Giacomo, M.Lenzerini; *Linking Data to Ontologies*, Journal on Data Semantics X. Lecture Notes in Computer Science, vol.4900,pp.133-173, Springer, Berlin, Heidelbegr (2008)
[160] S.Ramanujam, L.Khan, A.Gupta, S.Seida, *R2D: A Bridge between the Semantic Web and Relational Visualization Tools*, Proceedings of 3rd IEEE Conference on Semantic Computing (ICSC), Berkeley, California, USA (2009)
[161] V.Raskin, C.Hempelmann, K.Triezenberg, S.Nirenburg; *Ontology in Information Security: A Useful Theoretical Foundation and Methodological Tool*, Proceedings of the Workshop on New Security Paradigms (NSPW), pp.53-59, Cloudcroft, New Mexico, USA (2001)
[162] R.Raskin, M.Pan; *Semantic Web for Earth and Environmental Terminology(SWEET)*, Proceedings on Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data, Florida, USA (2003)
[163] A.Razzaq, Z.Anwar, F.Ahmad, K.Latif, F.Munir; *Ontology for Attack Detection: An Intelligent Approach to Web Application Security*, Computers&Security, vol.45, pp.124-146, Elsevier (2014)
[164] A.Razzaq, A.H.Farooq, N.Haider; *Ontology-based Application Level Intrusion Detection System using Bayesian Filter*, Proceedings of 2nd International Conference on Computer, Control and Communication (IC4), Karachi, Sindh, Pakistan (2009)
[165] J.Robinson, A.Voronkov; *Handbook of Automated Reasoning*, MIT Press, vol.1 (2001)
[166] M.Rouse; *What is Machine Learning?*, https://searchenterpriseai.techtarget.com/definition/machine-learning-ML (2018)
[167] Y.Sani, A.Mohamedou, K.Ali; *An Overview of Neural Networks Use in Anomaly Intrusion Detection Systems*, Proceedings of 7th IEEE Student Conference on Research and Development (SCORED), Serdang, Malaysia (2009)
[168] W.Savitch; *Relationship between Non-Deterministic and Deterministic Tape Complexities*. Journal of Computer and System Sciences, vol.4, pp.177-192 (1970)
[169] K.Scarfone, P.Mell; *Guide to Intrusion Detection and Prevention Systems (IDPS)*, Recommendations of the National Institute of Standards and Technology (NIST), Special Publication (2007)

[170] A.Schaerf; *Reasoning with Individuals in Concept Languages,* Journal of Data and Knowledge Engineering, vol.13, no.2, pp.141-176 (1994)
[171] K.Schild; *A Correspondence Theory for Terminological Logics*, Proceedings of 12th International Joint Conference on Artificial Intelligence (IJCAI), pp.466-471 (1991)
[172] K.Schild; *Terminological Cycles and the Propositional μ-Calculus*, Proceedings of 4th International Conference on the Principles of Knowledge Representation and Reasoning (KR'94), pp.509-520, Bonn, Germany (1994)
[173] M.Schmidt-Schauβ; *Subsumption in KL-ONE is undecidable*, Proceedings of the 1st International Conference on the Principles of Knowledge Representation and Reasoning (KR'89), pp.421-431, Los Altos, California (1989)
[174] M.Schmidt-Schauβ, G.Smolka; *Attributive Concept Descriptions with Complements*, Journal of Artificial Intelligence, vol.48,no.1, pp.1-26 (1991)
[175] T.Schneider, U.Sattler; *Description Logics: A Nice Family of Logics*, https://esslli2016.unibz.it/?page_id=160, 28th European Summer School in Logic, Language and Information (ESSLLI, 2016)
[176] B.Schueler; *Inference Techniques with respect to applications in Semantic Web*, Technical Report, Univesity of Dresden (2004)
[177] A.Schwartz, *Semantic Web in Breadth*, http://logicerror.com/semanticWeb-long (2002)
[178] A.Schwartz, J.Hendler; *The Semantic Web: A Network of Content for the Digital City*, http://blogspace.com/rdf/SwartzHendler (2002)
[179] M.Schmidt-Schaus, G.Smolka; *Attributive Concept Descriptions with Complements*, Journal of Artificial Intelligence, pp.1-27 (1991)
[180] G.Seif; *The Five Clustering Algorithms Data Scientists Need to Know*, https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68 (2018)
[181] K.Sequeira, M.Zaki; *ADMIT: Anomaly-based Data Mining for Intrusions*, Proceedings of 8th ACM SIGKDD International Conference, (2002)
[182] P.Sethi, S.Sarangi; *Internet of Things: Architectures, Protocols and Applications*, Journal of Electrical and Computer Engineering, pp.1-27, Hindawi (2017)
[183] N.Shadbolt, T.Berners-Lee, W.Hall; *The Semantic Web Revisited*, IEEE Intelligent Systems, vol.21, pp.96-101 (2006)
[184] R.Shanmugavadivu; *Network Intrusion Detection System using Fuzzy Logic*, Indian Journal of Computer Science and Engineering (IJCSE), vol.2, no.1 (2014)
[185] C.Sinclair, L.Pierce, S.Matzner; *An Application of Machine Learning to Network Intrusion Detection*, Proceedings of 15th Annual Computer Security Applications Conference (ACSA), Phoenix, Arizona (1999)
[186] J.Song, H.Takakura, Y.Okabe; *Statistical Analysis of Honeypot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation*, Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, pag.29-36, Salzburg, Austria (2011)
[187] K.Spackman, K.Campbell, R.Cote; *SNOMED RT: A Reference Terminology for Healthcare*, Journal of American Medical Informatics Association, pp.640-644 (1997)
[188] T.Takahashi, Y.Kadobayashi; *Reference Ontology for Cybersecurity Operational Information*, The Computer Journal, sect. Security in Computer Systems and Networks, vol.58,no.10 (2014)
[189] M.Taye; *Understanding Semantic Web and Ontologies: Theory and Applications*, Journal of Computing, vol.2 (2010)

[190] W.Teswanich, Chittayasothorn; *A Transformation from RDF Documents and Schemas to Relational Databases*, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, Canada (2007)
[191] S.Thakare, M.Ali; *Introducing Fuzzy Logic in Network Intrusion Detection*, Journal IJARCS, vol.3,no. 3 (2012)
[192] The Gene Ontology Consortium; *Gene Ontology: Tool for the Unification of Biology*, Journal of Nature Genetics, vol.25, pp.25-29 (2000)
[193] H.Tianfield; *Data Mining-based Cyber Attacks Detection*, System Simulation Technology, vol.13,no.2 (2017)
[194] S.Tobies; *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*, Phd thesis, RWTH Aachen, Germany (2001)
[195] C.Tsai, Y.Hsu, C.Lin, W.Lin; *Intrusion detection by Machine Learning: A Review*, Expert Systems with Applications, vol.36, pp.11994-120000, Elsevier 2009
[196] A.Turhan; *Introduction to Description Logics*, Proceedings of the 9th International Conference on Reasoning Web, Mannheim, Germany (2013)
[197] A.Turhan; *Introduction to Description Logics*, Technical University Dresden (ppt slides)
[198] A.M.Turing; *Computing Machinery and Intelligence*, Mind Journal, Oxford University Press (1950)
[199] J.Undercoffer, A.Joshi, J.Pinkston; *Modeling Computer Attacks: An Ontology for Intrusion Detection*, Proceedings of 6th International Symposium Recent Advances in Intrusion Detection, Pittsburgh, PA, (2003)
[200] J.Undercoffer, A.Joshi, J.Pinkston; *A Target-centric Ontology for Intrusion Detection,* Proceedings of 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico (2003)
[201] M.Vardi, P.Wolper; *Reasoning about Infinite Computations*, Information and Computation Journal, vol.115, no.1, pp.1-37 (1994)
[202] J.Wu; *Artificial Intelligence, Machine Learning and Deep Learning Explained Simply*, https://towardsdatascience.com/ai-machine-learning-deep-learning-explained-simply-7b553da5b960 (2019)
[203] A.Zamfira, R.Fat, C.Cenan; „Applying Semantic Web Technologies to Discover an Ontology of Computer Attacks", Scalable Computing: Practice and Experience Journal, West Univ. of Timisoara, vol.20, no.4 (2019)
[204] A.Zamfira, H.Ciocarlie; „Inference System to Achieve Saturation of (F,R,N) Knowledge Bases", Acta Technica Napocensis: Series Applied Mathematics, Mechanics and Engineering, Technical University of Cluj-Napoca, vol.62,no3 (2019)
[205] A.Zamfira, H.Ciocarlie; „A Network Intrusion Detection System Using Artificial Intelligence and Semantic Web Techniques", Acta Technica Napocensis: Series Applied Mathematics, Mechanics and Engineering, Technical University of Cluj-Napoca, vol.63,no4 (2020)
[206] A.Zamfira, H.Ciocarlie; „Developing an Ontology for Cybersecurity in Networks of Computers", Proceedings of 14th International Conference on Intelligent Computer Communication and Processing (ICCP'18), Cluj-Napoca, Romania (2018)
[207] A.Zamfira; „An Object-Oriented Reasoner for Saturation of Logical Knowledge Bases", Proceedings of 15th International Conference on Elearning and Software for Education (eLSE'19), Bucharest, Romania (2019)
[208] A.Zamfira; „The Feasibility of Artificial Intelligence in Emulating Human Behavior: An Analysis", Proceedings of 16th International Conference on Elearning and Software for Education (eLSE'20), Bucharest, Romania (2020)

[209] A.Zamfira, H.Ciocarlie; „Description Logics: Applications on the Semantic Web", Journal of Automation, Control and Applied Mathematics (ACAM), Technical University of Cluj-Napoca, vol.27,no.1 (2018)
[210] A.Zamfira, H.Ciocarlie; „Description Logics in Inference and Reasoning Services and Systems", Journal of Automation, Control and Applied Mathematics (ACAM), Technical University of Cluj-Napoca, vol.27,no.1 (2018)
[211] A.Zamfira, R.Fat, C.Cenan; „Using Artificial Intelligence and Semantic Web Technologies in Cyberdefense Systems", Bulletin of Politechnic Institute of Jassy, vol.65, no.2 (2019)
[212] A.Zamfira, R.Fat, C.Cenan; „Towards the Next-Gen Technologies for World Wide Web: The Semantic Web", Bulletin of Politechnic Institute of Jassy, vol.65, no.3 (2019)
[213] A.Zamfira, R.Fat, C.Cenan; „Towards the Next-Gen Technologies for Internet: The Internet of Things", Journal of Automation, Control and Applied Mathematics (ACAM), Technical University of Cluj-Napoca, vol.28,no.2 (2020)
[214] Y.Zhu; *Attack Pattern Ontology: A Common Language for Cyber-Security Information Sharing*, TU Delft Publication, Master Thesis (2015)
[215] D.Zyndros; *A Gentle Introduction to Algorithm Complexity Analysis*, https://discrete.gr/complexity
[216] *Closed World Machine*, https://www.w3.org/2000/10/swap/doc/cwm.html
[217] *Data Mining Methods*, https://www.educba.com/data-mining-methods/ (2017)
[218] *DROOLS*: https://docs.jboss.org/drools/release/7.17.0.Final/drools-docs/html_single/index.html
[219] *Gleaning Resources Descriptions from Dialects of Language*, https://www.w3.org/TR/grddl/ , W3C Recommendation (2007)
[220] *Information about CWM-TimBL's Closed World Machine*, http://infomesh.net/2001/cwm/ (2001)
[221] *JRules*: https://www.harukizaemon.com/blog/2004/03/09/jrules-a-brief-overview/
[222] *Ontology;* https://en.wikipedia.org/wiki/Ontology_(information_science)
[223] *OpenLink Virtuoso Universal Server,* http://docs.openlinksw.com/virtuoso/, OpenLink Document. (2018)
[224] *OWL2 Web Ontology Language Manchester Syntax,* W3C working draft, http://www.w3.org/TR/owl2-manchester-syntax/ , 2009
[225] *OWL2 Web Ontology Language Structural Specification and Functional-style Syntax*, W3C candidate recommendation, http://www.w3.org/TR/owl2-syntax/ (2009)
[226] *OWL2 Web Ontology Language Direct Semantics*, W3C candidate recommendation, http://www.w3.org/TR/owl2-direct-semantics/ (2009)
[227] *OWL2 Web Ontology Language Primer*, W3C working draft, http://www.w3.org/TR/owl2-primer/ , (2009)
[228] *RDF and SQL*, https://www.w3.org/wiki/RdfAndSql
[229] *RuleBook*: https://dzone.com/articles/rulebook-a-simple-rules-engine-that-leverages
[230] *Semantic Web*; https://en.wikipedia.org/wiki/Semantic_Web
[231] *SPARQL query language for RDF*, W3C recommendation, http://www.w3.org/TR/rdf-sparql-query/ , 2008
[232] *Computer Security,* https://en.wikipedia.org/wiki/Computer_security
[233] *MAEC*, http://maecproject.github.io/documentation
[234] *CAPEC*, http://capec.mitre.org
[235] *WASC*, http://projects.webappsec.org/w/page/13246978/threat
[236] *SCAP*, http://www.open-scap.org/security-policies/scap-security-guide
[237] *Propositional Logic*, http://intrologic.stanford.edu/notes/chapter_02.html
[238] *Modal Logic,* https://plato.stanford.edu/entries/logic-modal