Poznan University of Technology

T H E S I S

to obtain the degree of

Doctor of Philosophy

at the Council of the Faculty of Computing

by

Jarosław Ksawery BĄK

Rule-based query answering method for a knowledge base of economic crimes

Supervisor:
prof. dr hab. inż. Czesław JĘDRZEJEK

A dissertation prepared at
Institute of Control and Information Engineering,
Poznan University of Technology

Poznan, 2013

Dedicated to
my parents, Urszula and Krzysztof,
my best friend and brother Sławek,
my beloved wife Ula,
and my sister Kasia who will be always in our minds...

Abstract

Nowadays, the most of data processed in modern applications come from relational databases. Data stored in such sources are described only by their schema (a structure of data). Without strictly defined semantics there is often a mismatch problem with table and column names in databases. Moreover, it is rather difficult to query data at a more abstract level than only in a language of database relations and attributes. A lack of conceptual knowledge can be overcome by introducing ontologies. For the evaluation purposes, an ontology can be transformed into a set of rules. The additional rule-based knowledge allows reasoning and query answering at an appropriate abstract layer.

The main objective of this thesis is to propose a rule-based query answering method for relational data with formally defined semantics, expressed with additional knowledge represented as a set of Horn clauses. The set describes the source data at conceptual (ontological) level which consists of terms allowing to pose a query using the defined semantics. Reasoning is employed to obtain an answer for the query. Moreover, the rule-based query answering (RQA) method will be used with a knowledge base of economic crimes to support prosecutors and investigators in their work. An integration of relational data, an ontology and rules allowing RQA, make the problem challenging. In addition, the formulation of a knowledge base of economic crimes requires thorough analysis of real crime cases.

We focus on developing methods which provide a simplified and more convenient way to create queries than using structural constructions from SQL. Applying an ontology and rules to relational data requires increasing the performance of a reasoning employed in a query answering. Therefore, we investigate optimization techniques in a query evaluation performed in Datalog and rule-based systems. Moreover, we identify a proper formalism which combines description logics with rules without the loss of decidability. In addition, we analyse economic crimes, namely fraudulent disbursement and money laundering to construct a knowledge base expressed in a description logic with rules.

This thesis makes the following contributions. We propose two methods of a rule-based query answering applicable to relational databases with formally defined semantics. The first method uses hybrid reasoning implemented in the Jess engine and combines forward and backward chaining in a query answering.

In the second approach we devise novel modifications of a magic transformation introducing the extended, goal- and dependency-directed rules. This approach employs only the forward chaining. In both methods the Rete reasoning algorithm is applied. In addition, we propose a straightforward mapping method between ontology predicates and relational data. Moreover, we formulate a knowledge base of economic crimes, namely fraudulent disbursement and money laundering, as an ontology defined in the Horn-$\mathcal{SHIQ}$ description logic with Horn clauses. At the end, our research work has been implemented in the Semantic Data Library tool, allowing us to validate our RQA methods.

We present experiments with the implemented methods and the knowledge base of economic crimes. As a result, we confirm that our methods increase the performance and scalability of the Rete-based Jess reasoning engine. Moreover, the extended rules method is more general than the hybrid one and can be applied in any Rete-based reasoning engine. We conclude indicating directions for future research.

Acknowledgements

I would like to express my gratitude to prof. Czesław Jędrzejek, my supervisor. His support, advice and encouragement throughout the years helped me in many aspects of my work and my life. He always allows me to express my ideas, providing relevant feedback while asking open questions and showing important research directions.

I would like to thank dr Grażyna Brzykcy, for teaching me skills to express and to formalize scientific ideas, and for her interest in my work. I am grateful to dr Jolanta Cybulka, for her help and support in "semantics" research applied in this thesis. I would like to thank dr Adam Meissner, dr Jacek Martinek and dr Krzysztof Zwierzyński, for many fruitful discussions which improved my work.

I would like to acknowledge the support from Jacek Więckowski, the prosecutor who provided fruitful discussions and crime analysis while preparing the domain ontology.

I would like to thank all my colleagues, for exchanging ideas, for daily conversations on various important topics and for their technical and scientific support. Among them, I would like to thank dr Paweł Misiorek, dr Andrzej Szwabe, Michał Blinkiewicz and Przemek Walkowiak with who I share the room.

Apologies for not listing everyone by name. I appreciate helpful exchange of ideas, valuable feedback and invaluable advice. It is a pleasure to work with you every day.

I gratefully acknowledge the help and support of my friends which gave me the motivation in preparing this thesis.

Most importantly of all, I would like to thank my family for love, support and patience. I would like to thank my parents, for their faith in me and devotion of time. Special thanks to my wife for her understanding, love and continuous support every day. I would like to thank my brother, for exchanging ideas, for keeping me motivated and for his support in difficult times.

Acknowledgement. While working on the PhD I was supported by the following grants: *Polish Plaform for Homeland Security, A tool supporting investigative procedures using automatic inference* number 0014/R/2/T00/06/02 and *Rule-based query answering method for a relational database with the use of an ontology-based knowledge* number UMO-2011/03/N/ST6/01602.

Contents

List of Figures

List of Tables

Nomenclature

a, b, c	constants, individuals
$?x, ?y, ?z$	variables
$\bar{X}$	set of variables and/or constants
$\rightarrow$	implication
,	conjunction in rules
$\top$	universal concept
$\bot$	bottom concept
$\sqcap$	intersection (in DL syntax)
$\sqcup$	union (in DL syntax)
$\sqsubseteq$	subsumption of concepts (in DL syntax)
$\models$	logically entails
$\not\models$	does not logically entail
$\circ$	composition
$\exists$	Existential quantifier (in OWL represented as “someValuesFrom”)
$\forall$	Universal quantifier (in OWL represented as “allValuesFrom”)

CHAPTER 1
Introduction

Many methods have been developed for over twenty years to manage large amounts of data in relational databases. Currently, these approaches are very efficient and scalable. However, the relational data model is not able to capture complex relationships between objects such as relation composition (e.g., "being someone's uncle means being a brother of his/her parent"), cardinality restrictions (e.g., "each person has exactly two arms"), logical negation (e.g., "every person not being a man must be a woman") [Kremen 2012]. For complex systems we need to manage not only data, but also knowledge, as well as processing gathered data with richer means than offered by RDBMS (Relational Database Management System) and SQL (Structured Query Language). Knowledge management techniques require data semantics to be explicitly given. The resources to express it in relational databases are limited to a database schema (syntactic structure of data), which is not sufficient in the knowledge management. There is also a mismatching problem with table and column names in a database without strictly defined semantics.

In the last decade, the use of ontologies in information systems has become more and more popular in various fields, such as web technologies, database integration, multi agent systems, natural language processing, etc. Ontology-based information system design enters enterprise systems. Applications based on SNOMED CT[1] maintained by the International Health Terminology Standards Development Organisation are becoming small industry. Ontology-based systems enter enterprise systems, mostly for Business Process Management, Software Configuration Management and Software Evolution Management. Also so called Big Data has important semantic dimension. "The Big Data market is projected to reach $18.1 billion in 2013, an annual growth of 61%. This puts it on pace to exceed $47 billion by 2017" [Kelly 2013]. Yet, new technologies are a supplement to RDBMS. It seems, that a promising approach would be integrating them with RDBMS enabling queries with strictly defined semantics.

It is difficult to pose a query, which is the most common form of data retrieval, at a higher level of abstraction than in a language of database relations and attributes. The convenient way to express the domain semantics is to use ontologies which describe data at the conceptual, and formally defined, level, thus a lack of a conceptual knowledge in databases can be overcome by introducing ontologies.

[1] http://www.ihtsdo.org/

For evaluation purposes an ontology can be transformed into a set of rules. This kind of additional rule-based knowledge allows reasoning and query answering at an appropriate abstract level and relieves a user of using structural constructions from SQL. This kind of a query evaluation is called a rule-based query answering method, which has been investigated in this thesis. There is, however, another approach, more desirable in commercial applications – to transform ontology directly into SQL [Poggi 2008].

Our research concerns also the problem of applying the rule-based query answering to a knowledge base of economic crimes in order to discover crime activities and to suggest legal sanctions for crime perpetrators.

The most of the research work in the legal area relies on using ontologies in the field of information management and exchange [Biasiotti 2008, Casellas 2008] not reasoning [Breuker 2009]. Other solutions, developed for instance in FF Poirot project [Kerremans 2005, Zhao 2005] concern descriptions of financial frauds, mainly the Nigerian letter fraud and fraudulent Internet investment pages. The ontologies developed in this project are not publicly available.

This chapter introduces the problem of rule-based query answering and its application to a knowledge base of economic crimes. Section 1.1 gives a formal definition of the problem and presents the motivation of the thesis. We discuss the involved challenges in section 1.2, illustrating the main difficulties. We conclude with a list of major contributions in section 1.3. The structure of the dissertation is presented in section 1.4.

1.1 Motivation and Problem Statement

The problem of a rule-based query answering is an ongoing subject of research. The problem is strictly connected with reasoning. We now define the problem of the rule-based query answering.

Definition 1.1 (Rule-based query answering)**.** *Let Q be a query, R a set of rules, and let A be a reasoning algorithm. An evaluation of query Q which is supported by reasoning performed w.r.t. algorithm A and set R is called the rule-based query answering (RQA).*

It is worth noticing that Definition 1.1 is general and does not establish the form of a query, applied rules or a reasoning algorithm according to which an evaluation is performed.

Generally speaking, there are two kinds of reasoning methods applied in the rule-based query answering task. The first one is a backward chaining methods, where reasoning is goal-driven. In this case our goal is the query posed to the system. This scheme of reasoning is implemented, for instance, in Prolog engine,

and takes the form of the SLD resolution. In the backward reasoning technique facts are obtained only when they are needed in derivations.

On the contrary, forward chaining approaches, which are data-driven, need reasoning about all facts, which are stored in the working memory. Some of the inferred facts are useless and many rules are fired unnecessarily, which has a negative impact on the efficiency of the answering process. Moreover, because all facts should exist in the working memory, the scalability of reasoning task is poor due to the limited RAM memory. This drawback occurs also in the backward chaining.

The results of the OpenRuleBench initiative [Liang 2009a] show that efficiency of tabling Prolog and deductive database technologies surpasses the ones obtained from the corresponding pure rule-based forward chaining engines.

Aforementioned faults have had an important influence on our motivation. In this thesis we are interested in methods which increase efficiency of a reasoning applied in a rule-based query answering task. Thus, we concentrate on RQA which eliminates mentioned drawbacks in a state-of-the-art reasoning engine combined with a relational database (which acts as a storage location) and an ontology (which constitutes formally defined semantics of data). Thus, the aim of the thesis can be formulated as:

Development of a rule-based query answering method
applicable to relational databases with formally defined semantics.

We also assume that the rule-based query answering method is used with the knowledge base of the selected economic crimes: fraudulent disbursement and money laundering. We assume that our system is able to determine legal sanctions for crime perpetrators and to discover crime activities and roles (of particular types of owners, managers, directors and chairmen) using concepts, appropriate relations and rules. Rules and queries are used to reflect data concerning documents and their attributes, formal hierarchy in a company, parameters of transactions, engaged people actions and their legal qualifications. The similar assumptions were proposed in the work [Bezzazi 2007] which concerns cybercrimes only. The knowledge base of economic crimes is represented as an ontology which, for the evaluation purposes, is transformed into a set of rules.

The proposed method of RQA should apply to any domain (although it would require appropriate ontologies for each domain). The part of the thesis's title "*for a knowledge base of economic crimes*" results from the fact that our work on a rule-based query answering have been started in the economic crimes area.

The possible benefits, of the proposed rule-based query answering for a knowledge base of economic crimes, are the following:

- Design of a new method for an integration of a rule engine, an ontology and a relational database.

- Evaluation of queries posed to a relational databases in the terms of the conceptual level, thus one getting an easier way to create a query than using structural constructions from SQL. As a result, queries can be created by non-engineers, e.g. investigators or policemen.
- Increase of the efficiency of a rule-based query answering in a state-of-the-art reasoning engine. We decided to use the Jess engine [Hill 2003] which implements the Rete algorithm [Forgy 1982].
- Design of a rule-based query answering method which can be applied in other Rete-based engines, for instance, Drools 5 [Community 2012].
- Support for investigators which work to solve economic crimes and analyse data from various documents and bank transfers. They get a way in which legal sanctions are suggested according to analysed (concluded) data.
- Application in other domains than economic crimes, although it would require appropriate ontologies for each domain. Our methods are general and can be used in every application, which requires additional knowledge for query evaluation or need to offer an easier way of query creation than with the traditional SQL.

1.2 Research Challenges

In order to complete the problem of a rule-based query answering for a knowledge base of economic crimes we need to cope with the following challenges:

- Identification of a proper formalism that combines description logic with rules. Usually, an arbitrary combination is undecidable which means that results of reasoning (and thus query answering) may be incorrect or the reasoning process may fall into an infinite loop, so an answer cannot be obtained. The overall combination should be expressed in a set of rules. The type of permissible rules should be investigated.
- Designation of a reasoning algorithm which should ensure the decidability of a rule-based query answering in the selected formalism. This challenge requires the choice of a rule-based engine which performs reasoning with respect to an inference algorithm.
- Development of an RQA method which outperforms the current state-of-the-art methods for query evaluation in a rule-based system. In this challenge we need to cope with decidability, complexity and efficiency issues in the chosen inference algorithm.

- Integration of ontology predicates and a relational data in the form of a mapping between these two elements. The mapping results in the combination of a rule-based system with a relational database.

- Construction of a knowledge base of economic crimes which can be applied in a real world crime case. It means that it should reflect the mechanisms of a crime scheme and should be able to suggest legal sanctions. We need to determine crime activities and roles of particular types of perpetrators.

1.3 Main Contributions

This thesis studies the problem of evaluating queries in a rule-based system where data is stored in a relational database and queries can be posed at a conceptual (ontological) level. The thesis makes the following contributions:

- We formulate a knowledge base of economic crimes: fraudulent disbursement and money laundering as a minimal ontology model. The ontology describes a real crime case using Horn-$\mathcal{SHIQ}$ language and is implemented in the OWL Web Ontology Language supported by rules written in the Semantic Web Rule Language. Description of the minimal ontology model was published in [Bak 2010b, Bak 2010c, Jedrzejek 2011b, Bak 2013] and is described in Chapter 3.

- We propose a novel approach to query a relational database at a conceptual level using a hybrid approach. The approach combines forward and backward chaining performed by the Jess engine in a rule-based query answering task. This is the Jess-dependent method. This work was published in [Bak 2009] and is detailed in Chapter 4, Section 4.1.

- We design a new modification of a magic transformation [Beeri 1987] which introduces *extended rules*. The approach exploits forward chaining that is based on the Rete algorithm [Forgy 1982] and combines the Jess engine, a relational database and a Horn-$\mathcal{SHIQ}$ [Hustadt 2005] ontology. This is the Rete-dependent method. Extended rules method was published in [Bak 2011a] and presented in Chapter 4, Section 4.2.

- We propose a straightforward mapping method between a knowledge base and a relational database. The method is based on so-called *essential* predicates and SELECT-PROJECT-JOIN SQL queries. Our technique was published in [Bak 2008], extended in [Bak 2009] and described in Chapter 4, Section 4.4.

- We develop a new SDL (Semantic Data Library) framework, offering the implementation of both our methods of a rule-based query answering: hybrid and extended rules reasoning. The framework was presented at two RuleML Challenge competitions in 2010 [Bak 2010a] and in 2011 [Bak 2011b]. Detailed description of the SDL's implementation is presented in Chapter 5.

- We evaluate our two approaches of a rule-based query answering for a knowledge base of economic crimes. Performed experiments demonstrate that optimizations of a rule-based query answering are possible in the state-of-the-art Jess reasoning engine which is an implementation of the Rete algorithm. The evaluation of both our methods were published in [Bak 2010a, Bak 2011b] and are discussed in Chapter 6.

1.4 Structure of this Thesis

Chapter 1 introduces the problem of a rule-based query answering for a knowledge base of economic crimes. The goals accompanied by the research challenges and our main contributions are described. The remaining chapters are organised as follows:

- Chapter 2 introduces necessary terminology, presents the theoretical background and provides an overview of selected economic crimes: fraudulent disbursement and money laundering. We present the first-order logic-based formalisms of knowledge representation that are relevant to the thesis: a family of description logics, Datalog- and rule-based systems. We describe methods that combine description logics with datalog-like rules. Moreover, we provide a brief overview of a rule-based query answering and its optimization methods. At the end of the chapter we familiarize a reader with two economic crimes: fraudulent disbursement and money laundering.

- Chapter 3 describes our knowledge base of economic crimes: fraudulent disbursement and money laundering. We present the example real world crime case - the Hydra case in which both crimes occurred. We present the main parts of the created minimal ontology model with rules: determining legal sanctions for crime perpetrators, discovering crime activities and roles of perpetrators. We summarize discussing the related work.

- Chapter 4 presents our contributions in rule-based query answering. We describe two ways of applying reasoning process in the rule-based query answering task. In the first one, the *hybrid* reasoning (forward and backward chaining) is used and in the second one only forward chaining and *extended rules* are executed. The RQA method uses the reasoning process to obtain an

answer for a given query. During this process facts from database are gathered and used to derive new facts according to a given set of rules. Next, the answer is constructed and presented. In both our methods the Horn clauses are used as the form of permissible rules. Our work is based on the Rete reasoning algorithm which is, for instance, implemented in the Jess engine. Moreover, the problem of the combination between ontology predicates and a relational database is also provided in this chapter. We summarize with possible applications of the developed methods and a discussion of the related work.

- Chapter 5 provides the description of our Semantic Data Library (SDL) which is an implementation of all our RQA methods. SDL supports both hybrid and extended rule reasoning as well as the mapping method between ontology predicates and a relational data. We present the transformation of an ontology into a set of Horn clauses which is also supported by SDL.

- Chapter 6 contains the experimental evaluation of our methods. We compare our RQA algorithms with methods available in the Rete-based engine. Once again, Jess is used as an example reasoning engine. The results justify that our approaches beat pure query answering in the Jess engine.

- Chapter 7 summaries our approaches and our main results, including a discussion of work's limitations. The future prospects of the research are also presented.

CHAPTER 2

Preliminaries

In this chapter we introduce necessary terminology, basic definitions and results that are required in later parts of this work. The syntax and semantics of the first-order logic is explained in Section 2.1.1. Section 2.1.2 describes a rule language known as *Datalog* with its syntax and semantics. In Section 2.1.3 the fundamentals of rule-based systems are presented with an example implementation. Section 2.1.4 describes how the rule-based query answering task can be solved. Section 2.1.5 shows Description Logics syntax and semantics. Currently existed methods for combining Datalog-like rules and Description Logics are provided in Section 2.1.6. The description of economic crimes: fraudulent disbursement and money laundering is shown in Section 2.2.

2.1 Theoretical Background

In the following sections we provide basic definitions of logic-based formalisms that are relevant to this thesis. These formalisms are: Datalog (rules) and Description Logics (ontologies). Both of them are families of knowledge representation formalisms based on first-order logic (FOL).

2.1.1 First-order Logic

In this section we present the basic description of the syntax and semantics of the first-order logic. More detailed introduction can be found in [Ligeza 2006], [Lloyd 1984] and [Polleres 2011].

2.1.1.1 Syntax

Each formal language consists of a set of symbols that are legal in this language. The set of allowed symbols is defined as an *alphabet*.

Definition 2.1 (Alphabet)**.** *An alphabet of the first-order logic consists of the following symbols:*

- *A set of constants:* $a, b, c, ...$
- *A set of variables:* $?x, ?y, ?z, ...$

- *A set of function symbols: $f, g, ...$, where each function symbol has assigned arity (a natural number).*
- *A set of predicate symbols: $p, q, ...$, where each predicate symbol has assigned arity (a natural number).*
- *A set of logical connectives: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication) and $\leftrightarrow$ (material equivalence).*
- *Two quantifiers: $\exists$ (existential) and $\forall$ (universal).*
- *A set of punctuation symbols: '(', ')' and ','.*

In definition 2.1 each function and predicate symbol has an *arity* assigned to it. The arity represents the number of arguments that the function/predicate has. For instance, the mathematical function *f(x, y, z) = x + 2y + 3z* has three arguments, and is therefore of arity 3. A symbol of arity 1 is called a *unary* symbol; a symbol of arity 2 is called a *binary* symbol; a symbol of arity *n* is called *n-ary* symbol. The arity of a symbol may be 0. Such symbols are constants. According to definition 2.1 we can build *terms* and *formulae*.

Definition 2.2 (Term)**.** *A term is defined as follows:*

1. *A variable is a term.*
2. *A constant is a term.*
3. *If f is an n-ary function symbol with arity n and $t_1, t_2, ..., t_n$ are terms, then $f(t_1, t_2, ..., t_n)$ is a term.*

Definition 2.3 (Ground term)**.** *A term which does not contain any variables is called a ground term.*

Definition 2.4 (Atom)**.** *Let p be a predicate symbol with arity n. Let $t_1, t_2, ..., t_n$ be terms, then $p(t_1, t_2, ..., t_n)$ is an atom (or atomic formula). An atom which does not contain any variables is called a ground atom.*

More complex formulae can be built with the use of logical connectives ($\neg, \wedge, \vee$ etc.).

Definition 2.5 (Well-formed Formula)**.** *A well formed formula (or just formula) is defined as follows:*

1. *An atom is a formula.*
2. *If B and H are formulae then:*

- $\neg B$ *is a formula*
- $B \wedge H$ *is a formula*
- $B \vee H$ *is a formula*
- $B \rightarrow H$ *is a formula*
- $B \equiv H$ *is a formula*

3. *If* B *is a formula and* x *is a variable, then* $(\exists x H)$ *and* $(\forall x H)$ *are formulae.*

Definition 2.6 (Scope of Variables)**.** *If* x *is a variable and* H *is a formula then the scope of* x *in* $\exists x H$ *and of* $\forall x$ *in* $\forall x H$ *is* H*. Combinations of* $\exists x$ *and* $\forall x$ *bind every occurrence of* x *in their scope. An occurrence of a variable which is not bound is called free.*

Definition 2.7 (Open and Closed Formula)**.** *A formula is open if it has free variables. If a formula has no free variables then it is closed.*

Definition 2.8 (Literal)**.** *Let* H *be an atom. Then* $\neg H$ *and* H *are called literals, whereas* H *is called positive literal, while* $\neg H$ *is called negative literal.*

Definition 2.9 (First-order Language)**.** *A First-order language is defined over an alphabet and consists of the set of all well-formed formulae that can be constructed from the symbols of the alphabet. A FOL language is called function-free if it does not contain any function symbol (a set of functions symbols is an empty set).*

2.1.1.2 Semantics

Informally, the semantics of the first-order logic language is defined by attributing meaning (or truth values) to well-formed formulae (sentences). The sentences are mapped to some statements about a given domain through a process known as *interpretation*. If an interpretation gives the *true* value to a sentence then it is said to satisfy the sentence. Such an interpretation is called a *model* for the sentence. If an interpretation does not satisfy a sentence then it is called a counter-model.

Definition 2.10 (Interpretation)**.** *An interpretation* $\mathcal{I}$ *consists of the following:*

1. *A non-empty set* $\triangle^{\mathcal{I}}$ *called the universe of* $\mathcal{I}$ *or the domain of the interpretation. The members of* $\triangle^{\mathcal{I}}$ *are called individuals of* $\mathcal{I}$*.*

2. *An interpretation function* $\cdot^{\mathcal{I}}$*, which assigns elements of the alphabet to* $\triangle^{\mathcal{I}}$ *satisfying the following conditions:*

   - *Each constant* c *is mapped to an element* $c^{\mathcal{I}} \in \triangle^{\mathcal{I}}$

- *Each function symbol f of arity n is mapped to a function:*

$$f^{\mathcal{I}} : (\triangle^{\mathcal{I}})^n \rightarrow \triangle^{\mathcal{I}}$$

- *Each predicate symbol p of arity n is mapped to a function:*

$$p^{\mathcal{I}} : (\triangle^{\mathcal{I}})^n \rightarrow \{true, false\}$$

Definition 2.11 (Assignment)**.**

1. *Variable Assignment. A variable assignment is a mapping function σ, which assigns an element $c \in \triangle^{\mathcal{I}}$ to every variable x from a set of variables $\bar{X}$:*

$$\sigma : \bar{X} \rightarrow \triangle^{\mathcal{I}}$$

2. *Term Assignment. The term assignment w.r.t. σ of the term $t \in \triangle^{\mathcal{I}}$ is defined as:*
   - *Each variable assignment is given according to σ,*
   - *Each constant assignment is given according to $\mathcal{I}$,*
   - *If $t'_1, t'_2, ..., t'_n$ are term assignments of $t_1, t_2, ..., t_n$ and f' is the assignment of the function symbol f with arity n according to $\mathcal{I}$, then $f'(t'_1, t'_2, ..., t'_n) \in \triangle^{\mathcal{I}}$ is the term assignment of $f(t_1, t_2, ..., t_n)$.*

Definition 2.12 (Truth Values)**.** *The valuation of formula F is defined as follows:*

- *If the formula is an atom $p(t_1, t_2, ..., t_n)$ with arity n then the value is obtained by calculating the value of $p'(t'_1, t'_2, ..., t'_n)$ where p' is the mapping assigned to p by $\mathcal{I}$ and $t'_1, t'_2, ..., t'_n$ are the term assignments of $t_1, t_2, ..., t_n$ w.r.t. to σ and $\mathcal{I}$.*
- *The truth values of formulae B and H are given in Table 2.1*

Table 2.1: Truth values for formulae.

B	H	$\neg B$	$B \wedge H$	$B \vee H$	$B \rightarrow H$	$B = H$
$true$	$true$	$false$	$true$	$true$	$true$	$true$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$false$	$false$	$true$	$false$	$false$	$true$	$true$

- *The truth value of the formula $\exists x F$ is true if and only if there exists $c \in \triangle$ such that the formula F has truth value "true" w.r.t. $\mathcal{I}$ and $\sigma(x/c)$; otherwise it has value "false".*

- *The truth value of the formula $\forall x F$ is true if and only if for all $c \in \triangle^{\mathcal{I}}$ F is "true" w.r.t. $\mathcal{I}$ and $\sigma(x/c)$; otherwise it has value "false".*

Definition 2.13 (Satisfiability)**.** *A formula F is satisfiable if and only if there exists an interpretation $\mathcal{I}$ and variable assignment over some domain $\triangle^{\mathcal{I}}$ which makes the formula true.*

Definition 2.14 (Unsatisfiability)**.** *A formula F is unsatisfiable if and only if there does not exist any interpretation $\mathcal{I}$, variable assignment and domain $\triangle^{\mathcal{I}}$ satisfying the formula.*

Definition 2.15 (Model)**.** *If an interpretation $\mathcal{I}$ satisfies the formula F and each variable assignment then it is said that $\mathcal{I}$ is a model of F, denoted $\mathcal{I} \models F$.*

Definition 2.16 (Tautology)**.** *A formula F is a tautology if every interpretation $\mathcal{I}$ is a model of F. This can be denoted as $\models F$.*

Definition 2.17 (Logical consequence, Logical Implication, Entailment)**.** *A formula F is a logical consequence of a set of formulae Σ (denoted as $\Sigma \models F$) if and only if every model of Σ is also a model of F. In this case we say that Σ entails F or F is logically implied by Σ. $\Sigma \not\models F$ means that F is not a logical consequence of Σ.*

Definition 2.18 (Logical equivalence)**.** *Two formulae F and H are said to be logically equivalent (denoted by $F \equiv H$) if both $F \models H$ and $H \models F$ (so F and H have exactly the same models).*

2.1.2 Datalog as a First-order Rule Language

Datalog [Gallaire 1978] is a rule language which represents a fragment of the first-order logic and basic logic programming dialect. A natural way of understanding the notion of a "rule" in classical logic is to consider implication. It means that the rule is a formula which has an implication operator ($\rightarrow$ or $\Rightarrow$) as a most important connective. Variables in a rule are universally quantified. The implication is applicable to all individuals that satisfy the premise. It is worth noticing that Datalog has originally been developed for querying databases because rules and queries have much in common. As a result, new kind of database systems has emerged called *deductive database systems* where query language and (usually) storage structure are designed around a logical model of data [Ramakrishnan 1993].

In this section we introduce Datalog as a particularly restricted rule language. In this section we present its syntax and semantics, and we provide description of reasoning techniques that are commonly used with Datalog programs.

We start with an example which gives the intuition of rules and Datalog. The following rule describes a *Woman* which is also a *Mother*:

$$Woman(?x) \wedge hasChild(?x, ?y) \rightarrow Mother(?x)$$

The rule can be interpreted as:

- a way of querying a given database for all mothers. Information about *Woman* and *hasChild* are stored in the database while *Mother* is derived from these data as a query result
- a "view" on data which hides the complexity of the data that can also be aggregated
- a way to add new set of data to a database when premises are satisfied.

Characteristic of Datalog as a rule language is a feature called *recursion* which allows the result of a rule can to be used as one of the rule's premises. Datalog uses the *closed world* semantics which also occurs in relational databases. It means that facts that cannot be proven are considered false. More information about Datalog syntax and its semantics can be found in [Abiteboul 1995] and [Hitzler 2009b].

2.1.2.1 Syntax

Datalog is a subset of logic programming and its syntax is similar to the family of languages in this field. In this section we provide a definition of Datalog syntax and we introduce some fundamental differences between Datalog and logic programming in the context of their syntaxes.

Definition 2.19 (Datalog rule)**.** *A Datalog rule is an expression of the following form:*

$$p_1(\bar{X}_1), p_2(\bar{X}_2), ..., p_n(\bar{X}_n) \rightarrow h(\bar{X}_h) \tag{2.1}$$

where $n \geq 1$*,* $p_1, ..., p_n, h$ *are predicates (relation names) of appropriate arities and* $\bar{X}_1, ..., \bar{X}_n, \bar{X}_h$ *are sets of terms.*

Definition 2.20 (Datalog safety)**.** *Each variable occurring in* $\bar{X}_h$ *must occur at least in one of* $\bar{X}_1, ..., \bar{X}_n$*.*

Definition 2.21 (Datalog program, the body and the head)**.** *A finite set of Datalog rules is called a Datalog program. The premise of a Datalog rule is called the body (denoted as B) while the conclusion is called the head of a rule (denoted as H).*

The (2.1) form of a rule is logically equivalent to a disjunction of literals where at most one is positive. Such (2.2) formulae are called *Horn clauses*. From that reason a Datalog program can be viewed as a set of Horn clauses.

$$\neg p_1(\bar{X}_1) \vee \neg p_2(\bar{X}_2) \vee ... \vee \neg p_n(\bar{X}_n) \vee h(\bar{X}_h) \tag{2.2}$$

It is worth noticing that, according to the given nomenclature, the body of the rule can be called as: premises, antecedents, conditions or if-part of the rule. The head of the rule can be called as: conclusion, consequent or then-part of the rule.

It is common to omit the universal quantifier ($\forall$) since all variables in Datalog are universally quantified. Rules with an empty body are called *facts* while rules with an empty head are called *constraints* and are used to express the fact that interpretations satisfying the conditions in the body of the rule are not admitted. Since Datalog has originally been developed for querying databases, a Datalog program *P* has direct connection with a database. An *extensional* relation is an n-ary predicate which is used to represent facts stored in a database according to the extensional (database) schema, denoted *edb(P)*. Any other predicate is called an intensional relation which is defined by a rule (by appearing in the head of a rule). Set of all intensional relations forms the intensional (database) schema, denoted *idb(P)*. The schema of *P*, denoted *sch(P)*, is the union of *edb(P)* and *idb(P)*.

The distinction between extensional and intensional relations is a pragmatic one. It is useful for processing Datalog programs written for querying: rules whose heads contain only intensional predicates can be rewritten or compiled without a knowledge of a database instance to be queried. It is easy to fuse extensional relations in intensional ones by writing extensional predicates from the database instance as intensional predicates. Conversely, in any such created Datalog program an intensional relation can be redeclared as a (new) extensional predicate [Bry 2007].

Definition 2.22 (Datalog conjunctive query)**.** *A Datalog conjunctive query is a Datalog rule of the following form:*

$$p_1(\bar{X}_1) \wedge ... \wedge p_n(\bar{X}_n) \rightarrow answer(\bar{X}_a)$$

where $n \geq 0$*, the* p_i *are extensional predicates,* $answer$ *is an intensional predicate,* $\bar{X}_a$*,* $\bar{X}_1, ..., \bar{X}_n$ *are lists of terms of appropriate arities and the rule is range restricted, i.e., each variable in* $\bar{X}_a$ *also occurs in at least one of* $\bar{X}_1, ..., \bar{X}_n$ *(Datalog safety).*

Recursion is a programming technique involving the use of an item (predicate, function, procedure, algorithm etc.) which calls itself in each step of a computation. The termination condition is needed to stop a process of repeating items and to provide the result of the computation. In recursive Datalog (which is an extension

of a nonrecursive Datalog [Abiteboul 1995]) the recursion allows that a result of a rule can also be used as one of the rule's premises. An example of a rule which exploits recursion is the following:

$$ancestorOf(?x, ?y) \wedge ancestorOf(?y, ?z) \rightarrow ancestorOf(?x, ?z) \qquad (2.3)$$

The predicate *ancestorOf* is used both in the head and the body of the rule. It means that the result of the rule can also be used in the next application (firing) of the rule.

The major difference between Datalog and logic programming is that logic programming permits function symbols whereas Datalog does not. It means that a logic program can construct and manipulate complex data structures encoded by terms involving function symbols which is not possible in a Datalog program. Another important issue concerns the typical use of Datalog and logic programming. Is is assumed that a database is relatively large while a set of rules (Datalog program) is relatively small. In logic programming it is assumed that data are incorporated directly into the program. It means that every change of data modifies the logic program.

2.1.2.2 Semantics

In this section we present the Datalog semantics and we point out the main differences between Datalog and logic programming.

The formal semantics of Datalog is determined by the fact that it is a sublanguage of first-order logic. As usual for first-order logic, the semantics of Datalog is *model-theoretic*. There also exist equivalent approaches to define Datalog semantics: *proof-theoretic* approach and *fixpoint* approach. We describe model-theoretic approach in details while two other approaches are presented informally. More detailed information can be found in [Abiteboul 1995] and [Hitzler 2009b].

In model-theoretic approach a model in Datalog is a special kind of interpretation that makes a given Datalog program true. The key idea of this approach is to view the program as a set of first-order sentences.

Definition 2.23 (Datalog interpretation)**.** *A Datalog interpretation $\mathcal{I}$ consists of the following:*

1. *A non-empty interpretation domain $\triangle^{\mathcal{I}}$ which is a set of individuals.*
2. *An interpretation function $\cdot^{\mathcal{I}}$, which establishes the mapping from symbols into $\triangle^{\mathcal{I}}$:*
   - *If $c \in \triangle$ is an individual name, then $c^{\mathcal{I}} \in \triangle^{\mathcal{I}}$, so c is interpreted as an element of the domain,*

- *If $p \in \mathcal{P}$ is a predicate symbol of arity n, and $\mathcal{P}$ is a set of permissible predicates, then p is interpreted as an n-ary relation over the domain.*

Definition 2.24 (Variable Assignment)**.** *A variable assignment σ for $\mathcal{I}$ is a mapping function $\sigma^{\mathcal{I}} : \bar{X} \rightarrow \triangle^{\mathcal{I}}$, where $\bar{X}$ is a set of variables. For a term $t \in \triangle \cup \bar{X}$ we write $t^{\mathcal{I},\bar{X}}$ to mean $t^{\mathcal{I}}$ if $t \in \triangle$, and $t^{\bar{X}}$ if $t \in \bar{X}$. For an interpretation $\mathcal{I}$ and a variable assignment σ for $\mathcal{I}$, the truth value of a Datalog formula is defined in the following way:*

- $\top^{\mathcal{I},\bar{X}} = true$ *and* $\bot^{\mathcal{I},\bar{X}} = false$.
- *For a Datalog atom $p(t_1, t_2, ..., t_n)$ we set $p(t_1, t_2, ..., t_n)^{\mathcal{I},\bar{X}} = true$ if we find that $< t_1^{\mathcal{I},\bar{X}}, t_2^{\mathcal{I},\bar{X}}, ..., t_n^{\mathcal{I},\bar{X}} > \in P^{\mathcal{I}}$, and $p(t_1, t_2, ..., t_n)^{\mathcal{I},\bar{X}} = false$ otherwise.*
- *For a conjunction $p_1 \wedge p_2 \wedge ... \wedge p_n$ of Datalog atoms $p_1, p_2, ..., p_n$ we set $(p_1 \wedge p_2 \wedge ... \wedge p_n)^{\mathcal{I},\bar{X}} = true$ if each $p_i^{\mathcal{I},\bar{X}} = true$ for $i = 1, ..., n$. We set $(p_1 \wedge p_2 \wedge ... \wedge p_n)^{\mathcal{I},\bar{X}} = false$ otherwise.*
- *For a Datalog rule $B \rightarrow H$, where B represents an arbitrary conjunction of Datalog predicates, we set $(B \rightarrow H)^{\mathcal{I}} = true$ if we find that either $B^{\mathcal{I},\bar{X}} = false$ or $H^{\mathcal{I},\bar{X}} = true$. We set $(B \rightarrow H)^{\mathcal{I}} = false$ otherwise.* [Krötzsch 2010]

Definition 2.25 (Datalog satisfiability)**.** *A Datalog rule $B \rightarrow H$ is satisfied by an interpretation $\mathcal{I}$ if $(B \rightarrow H)^{\mathcal{I}} = true$. $\mathcal{I}$ satisfies a Datalog program if it satisfies all rules of the program. The program (rule) which is satisfied by some interpretation is called satisfiable or consistent.*

Definition 2.26 (Datalog model)**.** *If an interpretation $\mathcal{I}$ satisfies a Datalog program (rule) then $\mathcal{I}$ is called a model for the program (rule).*

The semantics of Datalog relies on the *closed world assumption (CWA)* which allows to treat facts as if they record complete information about the world their describe. Facts that are not stored in a database are considered false. Since there can be many (potentially infinite) facts in a database, we need to define which of them are needed to satisfy a Datalog program. It can be achieved by the use of the *minimal model*.

Definition 2.27 (Datalog minimal model)**.** *A Datalog model is called minimal if it consists of the smallest set of facts that makes a Datalog program (rule) true.*

The *proof-theoretic* approach is based on obtaining proofs of facts. A fact is in the result of a Datalog program (rule) if there exists a proof for it using the rules

and the database facts. Facts can be derived in two ways: *bottom-up* and *top-down*. In the bottom-up method we start from the known facts and derive all possible facts. In the top-down method we start from the fact to be proven and we try to derive lemmas that are needed for the proof. The method provides the intuition of a technique called resolution which comes from the logic programming area [Lloyd 1984].

The *fixpoint* approach assumes that we can define the semantics of the program (rule) as a particular solution of a fixpoint equation. The solution is based on an operator called *immediate consequence operator* which produces new facts starting from known facts. The immediate consequence operator of a Datalog program *P*, denoted $\mathcal{T}_P$, is the mapping from instance $\mathcal{K}$ over *sch(P)* to $\mathcal{K}$. It means that for each $\mathcal{K}$, $\mathcal{T}_P(\mathcal{K})$ consists of all facts that are immediate consequences for $\mathcal{K}$ and *P*. $\mathcal{K}$ is a fixpoint of $\mathcal{T}_P$ if $\mathcal{T}_P(\mathcal{K}) = \mathcal{K}$ [van Emden 1976, Abiteboul 1995].

More detailed information about the proof-theoretic and fixpoint approaches are presented in [Abiteboul 1995].

Heaving established the semantics and syntax of Datalog we now provide the important differences between Datalog semantics and logic programming semantics. Since function symbols are not allowed in Datalog (in contrast to logic programming), each Datalog program has always a finite model whereas model in logic programming may be infinite [Abiteboul 1995].

According to the lack of function symbols, the complexity of reasoning in Datalog is PTIME (polynomial-time) while in logic programming the complexity is usually higher and even undecidable [Dantsin 2001].

Extending Datalog with a negation or/and disjunction (disjunctive Datalog) implies that query answering and reasoning go beyond the polynomial-time [Patel-Schneider 2007].

Since work presented in this dissertation is based on extension and modification of query evaluation in Datalog-like rules we omit logic programming issues. Curious reader can find detailed information about logic programming in [Lloyd 1984].

2.1.2.3 Reasoning

Datalog was proposed as a way of querying relational databases with conjunctive queries and recursion. The evaluation of query may be done with *bottom-up* or *top-down* techniques. Both techniques are described and presented using an example which considers family relationships (FR). Let us consider a Datalog schema *sch(FR)* which consists of: an edb predicate *hasChild* used to express a relation between a parent and a child; and two idb predicates: *hasSiblings*, *hasCousin* which represent siblings relationship and cousin relationship, respectively. The *edb(FR)* is presented in Table 2.2 whereas *idb(FR)* contains rules (2.4) and (2.5). In both techniques we will evaluate query (2.6), looking for all cousins of a person p_{14}. We

point out that the same variable (e.g. $?x$) may occur in different rules (queries) and have different meaning. For instance, $?x$ in rule (2.4) is not to be confused with $?x$ in query 2.6, or in rule (2.5). It means that the set of rules and a query have no variables in common (it can always be ensured by renaming the variables of the rule/query).

$$hasChild(?x, ?y),\ hasChild(?x, ?z) \rightarrow hasSiblings(?y, ?z) \tag{2.4}$$

$$hasChild(?x, ?z),\ hasSiblings(?x, ?y),\ hasChild(?y, ?w) \rightarrow hasCousin(?z, ?w) \tag{2.5}$$

Table 2.2: An example extensional database - *edb(FR)*.

hasChild	Parent	Child
	p_{11}	p_{12}
	p_{11}	p_{13}
	p_{12}	p_{14}
	p_{13}	p_{15}
	p_{21}	p_{22}
	p_{21}	p_{23}
	p_{22}	p_{24}
	p_{23}	p_{25}

$$hasCousin(p_{14}, ?x) \rightarrow \tag{2.6}$$

A bottom-up evaluation starts with a set of facts and uses rules to infer new facts until no new facts can be inferred (until the fixpoint is reached). At the end of evaluation an answer is obtained (the answer may be available during the reasoning process, but in general it cannot be obtained until the fixpoint is reached). Table 2.3 illustrates steps of bottom-up evaluation of the Datalog program *FR* with schema *sch(FR)* = *edb(FR)* $\cup$ *idb(FR)*. It is easy to notice that for each tuple from Table 2.2 the appropriate fact is created with a predicate *hasChild*, e.g. $hasChild(p_{11}, p_{12})$ as a first tuple. In step 1 facts are added; next the reasoning algorithm uses known facts to produce new facts using rules: (2.4) in step 2, and (2.5) in step 3. When the fixpoint is reached we obtain an answer for query (2.6): a person p_{15} is a cousin of a person p_{14}. In the bottom-up approach the rules are treated as "factories" producing new facts from already proven ones. As a result, we obtain some other results, e.g. $hasCousin(p_{24}, p_{25})$, that are not relevant to our query.

A top-down evaluation is a more complicated approach. A major class of algorithms of this technique is represented by the *SLD resolution* which has been taken from the logic programming area [Kowalski 1974], excluding function symbols. The SLD resolution is focused on proving *goals*. A goal is considered

Table 2.3: Bottom-up evaluation of the Datalog program *FR*.

Step	Added/Inferred Facts	Used rule
0	$\emptyset$	-
1	$hasChild(p_{11}, p_{12})$, $hasChild(p_{11}, p_{13})$, $hasChild(p_{12}, p_{14})$, $hasChild(p_{13}, p_{15})$, $hasChild(p_{21}, p_{22})$, $hasChild(p_{21}, p_{23})$, $hasChild(p_{22}, p_{24})$, $hasChild(p_{23}, p_{25})$	-
2	$hasSiblings(p_{12}, p_{13})$, $hasSiblings(p_{22}, p_{23})$	(2.4)
3	$hasCousin(p_{14}, p_{15})$, $hasCousin(p_{24}, p_{25})$	(2.5)

as a query or as a fact that should be proved (obtained as a new goal from a previous goal in the reasoning process). Generally speaking, the SLD resolution tries to find a substitution θ that maps facts (terms) to variables such that query Q logically follows from a Datalog program P. For instance, in our example we need to find the substitution θ such that $\theta(hasCousin(p_{14}, ?x)) = \theta(hasCousin(?z, ?w))$. Such a substitution is called a unifier. The substitution $\theta(?z) = p_{14}$, $\theta(?w) = p_{15}$ and $\theta(?x) = p_{15}$ is a unifier for $hasCousin(p_{14}, ?x)$, because $\theta(hasCousin(p_{14}, ?x)) = \theta(hasCousin(?z, ?w)) = hasCousin(p_{14}, p_{15})$.

In a top-down evaluation the initial goal (query) is transformed, obtaining new goals from a previous ones, until the empty goal is reached. As a result, facts that are irrelevant to a given query are not taken into account. The acronym SLD stands for Selection rule-driven Linear resolution for Definite clauses. *Definite clauses* are clauses (rules) with a single predicate in the head of the rule whereas *rule-driven* refers to the rule used for selecting the atom which is evaluated as a first (in Prolog it is always the leftmost atom). The success or failure of SLD resolution does not depend on the rule for selecting atoms [Abiteboul 1995]. Let us introduce some important definitions which are relevant to the SLD resolution.

Definition 2.28 (Most general unifier)**.** *Let A, B be two atoms. A unifier for A and B is a substitution θ such that $\theta A = \theta B$. A substitution θ is more general than substitution ν, denoted $\theta \hookrightarrow \nu$, if for some substitution ν', $\nu = \theta \circ \nu'$. A most general unifier (mgu) for A and B is a unifier θ for A, B such that, for each unifier ν of A, B, we have $\theta \hookrightarrow \nu$.* [Abiteboul 1995]

"Clearly, the relation $\hookrightarrow$ between unifiers is reflexive and transitive but not antisymmetric. Let $\approx$ be the equivalence relation on substitutions defined by $\theta \approx \nu$ iff $\theta \hookrightarrow \nu$ and $\nu \hookrightarrow \theta$. If $\theta \approx \nu$, then for each atom A, $\theta(A)$ and $\nu(A)$ are the same modulo renaming of variables. " [Abiteboul 1995]

Definition 2.29 (SLD resolvent)**.** *Let G be a goal of the form $A_1, ..., A_m, ..., A_n \rightarrow$, R a rule $B_1, ..., B_k \rightarrow B$ and let θ be the mgu of A_m and B. Assuming that G and*

R have no variables in common then G' is an SLD resolvent of G and R using θ if G' is the goal $\theta(A_1, ..., A_{m-1}, B_1, ..., B_k, A_{m+1}, ..., A_n) \rightarrow$.

Definition 2.30 (SLD derivation)**.** *An SLD derivation from a goal $G = G_0$, G_1, ... with a program P_I (integrating the facts) consists of a sequence G_0, G_1, ... of goals and a sequence θ_0, θ_1, ... of mgu's such that for each i, G_i is a resolvent from G_{i-1} with some rule in P_I using θ_i. An SLD refutation is a finite SLD derivation which has the empty goal as its last goal.*

Definition 2.31 (SLD tree)**.** *An SLD tree T w.r.t. a program P and a goal G is a labelled tree where every node of T is a goal and the root of T is G and if G is a node in T then G has a child G' connected to G by an edge labelled (R, θ) iff G' is an SLD resolvent of G and R using θ.* [Bry 2007]

As we can see from the aforementioned definitions the computation of mgu's provides an answer for a query where each SLD resolvent is implied by some G and some R. An SLD derivation ends with the empty goal because in the SLD resolution we try to prove that the negation of a goal is false [Gallier 1987, Chapter 9]. If it is, then we obtain the empty goal. Otherwise, the proof of the goal cannot be obtained. An SLD tree represents all SLD derivations with a fixed selection rule which determines the choice of the selected atom (the leftmost atom in Prolog). Once, an atom has been selected, we can search for all possible unifications. An SLD may have an infinite subtree that corresponds to an infinite sequence of application of some rule (set of rules). In this case, the SLD resolution falls into infinite loop. The completeness of an SLD resolution depends on the search strategy which determines the order in which we visit nodes in an SLD tree (the choice of the clause to unify with the atom). "To be complete, an SLD resolution must visit every leaf of a finite branch of an SLD tree within a finite number of steps. A search strategy with this property is called *fair*. Obviously not every search strategy is fair. For example the depth first search strategy used by Prolog is not fair. An example of a fair search strategy is breath first search." [Bry 2007]

Let us now consider the top-down evaluation of query (2.6). Assuming that facts from Table 2.2 are integrated with the Datalog program *FR* we are obtaining an answer for query (2.6). The top-down evaluation with SLD resolution is presented in Table 2.4. For better understanding of our example we slightly modified rules: (2.4) and (2.5) with rules (2.7) and (2.8), respectively. These rules have no variables in common (which has been done by renaming the variables of the rules).

$$hasChild(?x_1, ?y_1),\ hasChild(?x_1, ?z_1) \rightarrow hasSiblings(?y_1, ?z_1) \tag{2.7}$$

$$hasChild(?x_2, ?z_2),\ hasSiblings(?x_2, ?y_2),\ hasChild(?y_2, ?w_2) \rightarrow hasCousin(?z_2, ?w_2) \tag{2.8}$$

In step 1 the query is unified with rule (2.8). Next, a substitution $?z_2/p_{14}$ is obtained. It means that a variable $?z_2$ can be substituted by value p_{14}. In step 3, the further substitution is computed. The computation of substitutions driven by rules is carrying on until the value of variable $?x$ is obtained in step 8. As a result, the substitution obtained in step 9 is an answer for our query, which is the same as in the bottom-up evaluation: $?x = p_{15}$.

Table 2.4: Top-down evaluation of the Datalog query $hasCousin(p_{14}, ?x)$.

Step	Goal	Used rule	Used fact	Substitution
0	$hasCousin(p_{14}, ?x)$	-	-	-
1	$hasCousin(p_{14}, ?x)$	(2.8)	-	$?z_2/p_{14}$
2	$hasChild(?x_2, p_{14})$	(2.8)	$hasChild(p_{12}, p_{14})$	$?x_2/p_{12}$
3	$hasSiblings(p_{12}, ?y_2)$	(2.8)	-	-
4	$hasSiblings(p_{12}, ?z_1)$	(2.7)	-	$?y_1/p_{12}$
5	$hasChild(?x_1, p_{12})$	(2.7)	$hasChild(p_{11}, p_{12})$	$?x_1/p_{11}$
6	$hasChild(p_{11}, ?z_1)$	(2.7)	$hasChild(p_{11}, p_{13})$	$?z_1/p_{13}$
7	$hasSiblings(p_{12}, ?y_2)$	(2.8)	-	$?y_2/p_{13}$
8	$hasChild(p_{13}, ?w_2)$	(2.8)	$hasChild(p_{13}, p_{15})$	$?w_2/p_{15}$
9	$hasCousin(p_{14}, ?x)$	(2.8)	-	$?x/p_{15}$

It is worth noticing that the bottom-up evaluation of a Datalog program can be done regardless a query. It means that we start with a set of facts and use rules to infer new facts until no new facts can be inferred (until the fixpoint is reached). The top-down evaluation, on the contrary, requires a query from which the reasoning process can start. In this approach we are focused on proving particular facts that are required to obtain an answer for the query. As a result, the top-down technique inhibits the inference of facts that are irrelevant to the query. Both evaluation techniques for a Datalog program are sound and complete. More detailed information about Datalog query evaluation can be found in [Abiteboul 1995].

2.1.3 Rule-based Systems

This section presents systems that use rules to express and manipulate knowledge. Such systems are called *rule-based systems* or *rule-based expert systems* [Nalepa 2009]. These systems found a wide range of applications in many fields including: decision support, medicine, business, data analysis, natural language processing and in other artificial intelligence applications. In this section we present a brief description of rule-based systems. We present the Rete reasoning algorithm with a state-of-the-art implementation - the Jess engine [Hill 2003].

A rule-based system is used as a way to derive new facts from the given ones according to the defined set of rules. Such a system consists of few elements:

- a list of rules (rule base), which forms a kind of a knowledge base
- a working memory, which contains facts. The working memory changes during the reasoning process
- an inference engine, which generates a new fact (or takes an action) based on an interaction between facts and the rule base
- a user's interface (e.g. a console).

Usually, a rule-based system processes data only in its working memory. According to a *forward chaining* mechanism (bottom-up evaluation), commonly used in reasoning tasks, a user gets information as a set of inferred facts. In this set it is hard to find a fact or facts which the user is interested in. Thus, the user has to pose a query to the rule-based system to obtain the necessary facts. This is a better way than looking through the working memory manually. The forward chaining approach needs reasoning about all facts in the working memory. Therefore, some of the inferred facts are useless and many rules are fired unnecessarily. It has a negative impact on efficiency of the answering process. One way of increasing efficiency and scalability of a deduction process is to use a *backward chaining* method (top-down evaluation). This scheme of reasoning is implemented, for instance, in the Prolog engine (see Section 2.1.2.3. By the backward reasoning technique facts are obtained only when they are needed in derivations.

2.1.3.1 Rules and Facts

In rule-based systems we apply the following form of a rule:

$$p_1(\bar{X}_1),\ p_2(\bar{X}_2),\ ...,\ p_n(\bar{X}_n),\ AP\ \rightarrow\ h(\bar{X}_h) \tag{2.9}$$

Each p_i (and h) is a predicate symbol, and $\bar{X}_i$ represents a vector of variables and constants, which appear in the atom $p_i(\bar{X}_i)$ as arguments. AP denotes a set of additional predicates, which are used for comparisons and tests, for example: x < 2, y $\leqslant$ x, etc. Each rule consists of the two parts: the left-hand-side, which is called the body, and the right-hand-side, which is called the head. In general, both parts are represented by sets of atoms interpreted conjunctively. In the body of the rule we use premises (patterns, conditions), which have to be satisfied by appropriate atoms (facts) to allow a rule to be fired and to produce conclusions from the rule's head. We assume that the body of a rule may be empty. In this case, the rule is called a fact. Rules of the form (2.9) belong to the class of Horn clauses [Lloyd 1984] (if there are several predicates in the head, the rule can be easily transformed into Horn clauses with the Lloyd-Topor transformation [Lloyd 1984]).

2.1.3.2 The Rete Algorithm

The Rete algorithm, used to match atoms (facts) and rules (patterns) in the rule-based systems, is fast and efficient. The algorithm was invented by Charles Forgy [Forgy 1982], and the word 'Rete', a Latin name for 'net', explains the idea of the algorithm. The Rete-based system builds the network of nodes, where each node (except the root) corresponds to a pattern appearing in the left-hand-side (the body part) of a rule. Every path from the root node to the leaf node defines all patterns of the rule. The name of the rule is associated with each leaf node. Each node remembers which facts match that pattern. When a fact or a combination of facts satisfies all patterns in the rule, the leaf node is reached and the corresponding rule is put in the agenda. Then, the conflict resolution strategy decides which rule should be fired first. In a case where two (or more) rules contain the same pattern, they share only one node in the Rete network. As a result the network contains the number of nodes which corresponds to the number of different patterns in the rules [1]. Such optimization increases extremely (by several orders of magnitude) the performance of the rule-based system in comparison to the naive approach, where the system checks each rule according to the facts in the working memory. The performance of the Rete algorithm is weakly dependent of the number of rules in the system. In the Jess engine, which is a Rete-based system, "the runtime will be proportional to something like $R'F'^{P'}$, where R' is a number less than R, the number of rules; F' is the number of facts that change on each iteration; and P' is a number greater than one but less than the average number of patterns per rule". [Hill 2003] Therefore, it is better to create more rules but with smaller number of different patterns per rule.

2.1.3.3 Forward and Backward Chaining in the Jess Engine

In this section we present a state-of-the-art implementation of the Rete algorithm in the form of the Jess engine. In this tool the forward chaining method starts with facts available in the working memory and checks if any rule can be fired. The rule is fired when all its patterns are satisfied by facts from the working memory. The head of the rule can add (modify or retract) a new fact (or facts) in the working memory. These new facts can fire other rules. The reasoning engine infers until there are no new facts to match any of the rules, as usual in the bottom-up evaluation.

The backward chaining method in Jess requires a special declaration for templates (similar to predicates in a FOL language). Rules to match the templates are defined and the rule compiler rewrites such rules and adds the *need-* prefix to in-

[1] It is a general idea. There exist some additional nodes which are used by alpha and beta networks [Forgy 1982] created by the Rete algorithm.

form the Jess engine when the rule has to be fired (when we need some fact). If a rule fires and there is a way to obtain needed facts, they appear in the Jess's working memory. The *need-* facts are the so called triggers (in the Jess language terminology). These facts correspond to the goals in the backward reasoning method. The generation of the need- goals is inefficient in the Jess engine because the backward chaining is simulated by the forward chaining. The other reason for this inefficiency is that the Jess engine creates trigger facts (with *need-* prefix) during execution of a query (our goal) and then calculates rules activations (but it does not fire any of the rules). This procedure does not appear in the forward chaining mode, so the reasoning process is much faster. As we can see these two methods have important drawbacks if we want to use them in a rule-based query answering task. The best solution would be to reason in a goal-directed fashion (like in the backward chaining scheme) but with the efficiency of the forward chaining.

2.1.4 Rule-based Query Answering

This section describes the rule-based query answering task [Bry 2007], which is of a special interest in the context of this thesis. It has many times been subject to research and a variety of techniques emerged covering a range of different approaches. These techniques are separated, generally, into two classes: top-down or bottom-up evaluation. Since these reasoning methods have been presented in Section 2.1.2.3, we now describe a broad family of heuristic techniques which are used as optimizations in rule-based query answering. Some of presented optimizations can be used in both top-down and bottom-up evaluation. Since our work is focused more on forward reasoning than backward one, we put more attention to describe optimization techniques for bottom-up evaluation of queries. Let us present, informally, these two reasoning methods in the context of the rule-based query answering and some efficiency issues.

The first of the aforementioned evaluation classes is a backward chaining method (top-down evaluation), where reasoning is goal-driven. In this case the goal is the query posed to the system. This scheme of reasoning is implemented, for instance, in Prolog engine, and takes the form of the Selection rule-driven Linear resolution for Definite clauses (SLD resolution). In the backward reasoning technique facts are obtained only when they are needed in derivations.

On the contrary a forward chaining approach (bottom-up evaluation), which is data-driven, needs reasoning about all facts. In the working memory some of the inferred facts are useless and many rules are fired unnecessarily. It has a negative impact on the efficiency of the answering process. Moreover, as all facts should exist in the working memory, the scalability of reasoning task is poor due to the limited RAM memory. This drawback occurs also in the backward chaining.

The naive (without any optimization) and semi-naive (which tries to avoid recomputing the same facts) methods that are based on deduction techniques using forward reasoning are often inefficient. They do not use the constants occurring in queries for restricting the search space. Such a restriction is used in backward reasoning [Bry 1990].

Many optimizations techniques have emerged during the development of deductive databases and the evaluation of recursive queries. Most of the work was done in eighties and early nineties, when knowledge based systems were created for the first time. Some of these optimizations are based on direct evaluation of queries whereas others are based on rule-rewriting techniques.

The most comprehensive approaches concerning optimizations of bottom-up query evaluation were given in [Bancilhon 1986b, Beeri 1987, Bry 1990, Bry 2007]. Top-down optimization techniques are described in [Bancilhon 1986b, Dietrich 1989, Abiteboul 1995].

The results of the OpenRuleBench initiative [Liang 2009a] show that efficiency of tabling Prolog and deductive database technologies surpasses the ones obtained from the corresponding pure rule-based forward chaining engines.

The following subsections describe optimization methods which are of a special interest in the context of our work.

2.1.4.1 Sideways Information Passing and Adorned Rules

Sideways information passing strategy (sip strategy) is an optimization technique which indicates how bindings in the head of a rule should be passed to the body of that rule, and in which order body atoms should be evaluated [Sagiv 1984]. For a set of rules P and a query Q, there usually exist many different sip strategies. Without evaluating all of them, it is not easy do decide whether a chosen sip strategy is better or worse than another one [Sippu 1990]. For instance, let us consider rule (2.10). Several sip strategies (2.11) - (2.16) are available for rule (2.10) and the query $h(?x, ?z)$ (we do not consider the bindings of variables in this query).

$$p_1(?x, ?y),\ p_2(?y, ?w),\ p_3(?x, ?z) \rightarrow h(?x, ?z) \tag{2.10}$$

$$p_1(?x, ?y),\ p_2(?y, ?w),\ p_3(?x, ?z) \rightarrow h(?x, ?z) \tag{2.11}$$
$$p_1(?x, ?y),\ p_3(?x, ?z),\ p_2(?y, ?w) \rightarrow h(?x, ?z) \tag{2.12}$$
$$p_3(?x, ?z),\ p_1(?x, ?y),\ p_2(?y, ?w) \rightarrow h(?x, ?z) \tag{2.13}$$
$$p_3(?x, ?z),\ p_2(?y, ?w),\ p_1(?x, ?y) \rightarrow h(?x, ?z) \tag{2.14}$$
$$p_2(?y, ?w),\ p_3(?x, ?z),\ p_1(?x, ?y) \rightarrow h(?x, ?z) \tag{2.15}$$
$$p_2(?y, ?w),\ p_1(?x, ?y),\ p_3(?x, ?z) \rightarrow h(?x, ?z) \tag{2.16}$$

The choice of the particular sip strategy should be driven by bindings of variables in the given query. Let us assume that variable $?x$ is bound in the query. Then, assuming that the evaluation of a rule is done with a left-to-right fashion, sip strategies (2.15) or (2.16) provide inefficient evaluation since none of the variables $?y$ or $?w$ are bound. Sip strategies use so-called *adornments* to express bindings of variables.

Definition 2.32 (Adornment)**.** *An adornment of a predicate is a sequence of b's (bound) and f's (free) indicating the status of the arguments of the predicate. The adorned rule is obtained by replacing each atom in the rule by its adorned version.*

For example, to indicate that in predicate $p(?x, ?y)$ only the variable $?x$ is bound, we write $p^{bf}(?x, ?y)$. For each adorned predicate p^a and for each rule, where p occurs in the head, we should choose the sip and use it to generate an adorned version of the rule. Since predicates may appear with several adornments, we may attach several distinct sips to several versions of the same rule, one to each version. Such process of creating adorned rules starts from the given query. For the query predicate q we replace it by the adorned version of q where adornment is determined by bindings of variables in the query. Next, the adorned rules are generated according to the chosen sip and adornment of q. For example, for the following rule:

$$p(?x,\ ?y)\ \rightarrow\ q(?x,\ ?y)\ \text{ and the query: }\ q(10,\ ?y)\ \rightarrow$$

the following adorned rule is generated:

$$p^{bf}(?x,\ ?y)\ \rightarrow\ q^{bf}(?x,\ ?y) \tag{2.17}$$

2.1.4.2 Magic Transformation and Other Rule-rewriting Techniques

Magic transformation (or magic sets) is an optimization technique which allows to rewrite the rules for each query and simulates the sideways information passing of bindings that occur in top-down evaluation but performing forward reasoning. The general method relies on the transformation of a program *P* (set of rules) and a query *Q* into a new program, $magic(P \cup Q)$, as shown in [Nilsson 1995]. Magic transformation modifies each original rule by additional predicates to ensure that the rule will fire only when the values for these predicates are available.

In rule-based systems with facts and rules deduction processes are often performed with the bottom-up scheme of evaluation. But effective query answering process should be combined with a goal-directed fashion (like in the top-down reasoning). A bottom-up evaluated magic program avoids a blind generation of conclusions by inserting special conditions into each rule of the program P. The new predicates - $call\_p$ for each original predicate p - are used in [Nilsson 1995] to define the conditions. In magic transformation for each rule:

$$p_1(\bar{X}_1), p_2(\bar{X}_2), ..., p_n(\bar{X}_n) \rightarrow h(\bar{X}_h)$$

a set of new rules is defined in the following way:

$$call\_h(\bar{X}_h), p_1(\bar{X}_1), p_2(\bar{X}_2), ..., p_n(\bar{X}_n) \rightarrow h(\bar{X}_h)$$

$$call\_h(\bar{X}_h), p_1(\bar{X}_1), p_2(\bar{X}_2), ..., p_{i-1}(\bar{X}_{i-1}) \rightarrow call\_p_i(\bar{X}_i)$$

where $i \in \{1, n\}$

Definition 2.33 (Magic template)**.** *Atoms call_p are called magic templates and can be interpreted as needed or called atoms. Magic templates (or called atoms) are differentiated from proper ones by annotation with C symbol.*

One can see the new rules as plans for the rule's head evaluation, but plans augmented with goal of the evaluation. Let us now consider an example of magic transformation of rule (2.18).

$$p_1(?x, ?y),\ p_2(?y, ?z),\ p_3(?z, ?w)\ \rightarrow\ h(?x, ?w) \tag{2.18}$$

As a result, the set of the following rules is generated:

$$h_1(?x,\ ?w)^C,\ p_1(?x,\ ?y),\ p_2(?y,\ ?z),\ p_3(?z,\ ?w)\ \rightarrow h_1(?x,\ ?w)$$

$$h_1(?x,\ ?w)^C\ \rightarrow p_1(?x,\ ?y)^C$$

$$h_1(?x,\ ?w)^C,\ p_1(?x,\ ?y)\ \rightarrow p_2(?y,\ ?z)^C$$

$$h_1(?x,\ ?w)^C,\ p_1(?x,\ ?y), p_2(?y,\ ?z)\ \rightarrow p_3(?z,\ ?w)^C$$

The magic approach has been shown to be sound and complete [Nilsson 1995]. The original magic sets optimization [Bancilhon 1986a] was extended in [Beeri 1987] with the following techniques:

- Generalized Magic Sets (GMS).
- Generalized Supplementary Magic Sets (GSMS).
- Generalized Counting (GC).
- Generalized Supplementary Counting (GSC).

GMS optimization combines optimal sip strategy (a sip that computes a minimal number of fact) with original magic transformation. As a result, values occurring in a given query create a *seed* which provides an initial value for the magic predicate corresponding to the query. Since magic sets and GMS suffers from the fact that it duplicates the work it does in computing the magic sets when computing

the corresponding predicates (that is, when firing the modified rules) GSMS strategy was proposed. “In GSMS and GSC, all results that are potentially useful later, are stored. Thus, they trade-off additional memory (and possibly, increased lookup times) for the time gained in avoiding some duplicate firings of rules. GC and GSC refine the notion of a relevant fact by essentially numbering the magic sets. This means that they avoid many unnecessary firings by starting at the query node and working outwards. They do this at the cost of maintaining a system of indices (and of course, are applicable only for a restricted set of data and rules).” [Beeri 1987]

Other rule-rewriting technique is called the *Alexander Method* [Rohmer 1986]. The method was developed independently and it is essentially the Generalized Supplementary Magic Sets strategy. In other words the Alexander Method re-expresses the GSMS technique in a different terminology.

There were also other improvements and modifications of magic approach [Bry 2007].

2.1.4.3 Other Optimizations

In this section we provide a brief description of other optimization techniques that are centered around a top-down evaluation. First of them is an approach known as "Query-Subquery" (QSQ) [Vieille 1986]. In this technique an original query, in which constants occur, is divided into a set of *subqueries*. The subqueries are systematically evaluated till the answer is obtained.

Another interesting approach was presented in [Bry 1990] where the Backward Fixpoint Procedure (BFP) aims to perform top-down reasoning by a bottom-up meta-interpreter which operates on Datalog rules in addition to data. The BFP provides the meta-interpretation framework which reconciles the bottom-up and top-down methods to compute recursive queries. This work shows that rewriting-based and resolution-based methods can be interpreted as a specialization of the Backward Fixpoint Procedure.

According to the work presented in [Brass 2010] we also believe that the bottom-up approach has still room for improvements in order to increase the performance of the rule-based query answering task.

2.1.5 Description Logics

In this section we present the introduction to description logics (DLs) which are a family of knowledge representation languages. We provide their syntax, semantics and reasoning tasks.

Description logics are used to represent the terminological knowledge of an application domain in a structured and formally defined way. The domain of interest is described by *concepts* (unary predicates) and *roles* (binary predicates). Con-

cepts and roles are defined using the concept and role constructors provided by a particular DL.

Historically, DLs are related to semantic networks, conceptual graphs and frame languages [van Harmelen 2007]. Description logics with their formal, logic-based, semantics allow for automated reasoning while their restricted syntax (formulae with at most two variables) provides decidability and human-readable form [Nalepa 2011].

Definitions presented in this section can be found in [Baader 2003] and [van Harmelen 2007].

2.1.5.1 Syntax

In description logics two kinds of atomic symbols are allowed: *atomic concepts* (denoted by *A*, *C* and *D*) and atomic roles (denoted by *R* and *S*). Atomic symbols provides elementary descriptions from which we can build complex descriptions using concept and role constructors. Description logics are distinguished by the constructors they provide. The language $\mathcal{AL}$ (attributive language), introduced in [Schmidt-Schauss 1991], is the minimal language of practical interest. The other languages of this family are extensions of $\mathcal{AL}$. Concept descriptions in $\mathcal{AL}$ are formed according to the following syntax rule:

$$C, D \rightarrow A \mid \top \mid \bot \mid C \sqcap D \mid \exists R.\top \mid \forall R.C \mid \neg C$$

The atomic negation ($\neg C$) in $\mathcal{AL}$ can only be applied to atomic concepts. In the scope of an existential quantification over a role only the top ($\top$) concept is allowed (this restriction is called a limited existential quantification). These limitations do not exist in more expressive languages which may be obtained by adding further constructors to $\mathcal{AL}$. Some of them are presented in Table 2.1.5.1.

Extending $\mathcal{AL}$ by any subset of the constructors yields a particular DL $\mathcal{AL}$ language which name is defined by a string composed from constructors' symbols. For example, the union of concepts is indicated by $\mathcal{U}$ whereas full existential quantification by $\mathcal{E}$. Since these constructors can be expressed using negation of arbitrary concepts ($C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ *and* $\exists R.C \equiv \neg\forall R.\neg C$) the letter $\mathcal{C}$ is used instead of the letters $\mathcal{UE}$. The language $\mathcal{ALC}$ extended with transitive roles (denoted by $\mathcal{R}^+$) is named $\mathcal{ALC}_{\mathcal{R}^+}$. In order to avoid very long names for expressive DLs, the abbreviation $\mathcal{S}$ was introduced for $\mathcal{ALC}_{\mathcal{R}^+}$. The language $\mathcal{S}$ may be extended with inverse roles (denoted by $\mathcal{I}$), functional roles (denoted by $\mathcal{F}$), role hierarchy (denoted by $\mathcal{H}$), role chains (denoted by $\mathcal{R}$), nominals (denoted by $\mathcal{O}$), number restrictions (denoted by $\mathcal{N}$) and qualified number restrictions (denoted by $\mathcal{Q}$). More constructors can be found in [Baader 2003, Appendix 1].

Description logics may be extended with *concrete domains* which allow for the use of types such as integers ("3" for example) and predicates such as integer

Table 2.5: The syntax of description logics.

Expression	Syntax	Symbol
Universal concept	$\top$	$\mathcal{ALC}$
Bottom concept	$\bot$	
Atomic concept	A	
Atomic role	R	
Conjunction	$C \sqcap D$	
Disjunction	$C \sqcup D$	
Exists restriction	$\exists R.C$	
Value restriction	$\forall R.C$	
Negation	$\neg C$	
Role hierarchy	$R \sqsubseteq S$	$\mathcal{H}$
Role chains	$R_1 \circ ... \circ R_n \sqsubseteq S$	$\mathcal{R}$
Inverse role	R^-	$\mathcal{I}$
Transitive role	$R \in R^+$	$\mathcal{S}$
Functional role	$\leq 1R$	$\mathcal{F}$
Nominals	$a_1, ..., a_n$	$\mathcal{O}$
Number restriction	$\geq nR$ $\leq nR$	$\mathcal{N}$
Qualified number restriction	$\geq nR.C$ $\leq nR.C$	$\mathcal{Q}$

comparisons [Baader 1991]. Concrete domains, known as Datatypes, are denoted by appending (D) to the name of the logic, for example $\mathcal{SHIQ}$(D).

Some of description logics, because of their practical importance, have been chosen as ontology languages: OWL [Horrocks 2003, Consortium 2006] and OWL 2 [Motik 2008, Consortium 2012]. For example, OWL Lite provides the expressiveness of $\mathcal{SHIF}$(D), OWL DL is based on $\mathcal{SHOIN}$(D) and OWL 2 is equivalent to $\mathcal{SROIQ}$(D).

To give an idea of what can be expressed using constructors provided by a particular description logic we present a set of examples in Table 2.1.5.1. Suppose *Person*, *Female* and *Male* are atomic concepts whereas *hasParent*, *hasFather*, *hasChild*, *hasGrandparent* and *hasAncestor* are atomic roles. Then *Person*$\sqcap$*Male* represent persons that are males while *Person*$\equiv$*Male*$\sqcup$*Female* express that *Person* is *Male* or *Female*. Description $\exists$*hasParent.Male* denotes, intuitively, parents which are males (fathers). Otherwise, $\forall$*hasParent.Person* denotes that all parents are persons. Negation $\neg$*Male* describes a concept which is not *Male*, for example, *Female*.

Table 2.6: The example use of expressions in description logics.

Expression	Syntax	Example
Universal concept	$\top$	
Bottom concept	$\bot$	
Atomic concept	A	$Male$
Atomic role	R	$hasParent$
Conjunction	$C \sqcap D$	$Person \sqcap Male$
Disjunction	$C \sqcup D$	$Person \equiv Male \sqcup Female$
Exists restriction	$\exists R.C$	$\exists hasParent.Male$
Value restriction	$\forall R.C$	$\forall hasParent.Person$
Negation	$\neg C$	$\neg Male$
Role hierarchy	$R \sqsubseteq S$	$hasFather \sqsubseteq hasParent$
Role chains	$R_1 \circ ... \circ R_n \sqsubseteq S$	$hasParent \circ hasGrandparent \sqsubseteq hasAncestor$
Inverse role	R^-	$hasParent \equiv hasChild^-$
Transitive role	$R \in R^+$	$hasAncestor^+$
Functional role	$\leq 1R$	$\leq 1hasFather$
Nominals	$a_1, ..., a_n$	$blue, red, yellow$
Number restriction	$\geq nR$ $\leq nR$	$\geq 3hasChild$ $\leq 2hasParent$
Qualified number restriction	$\geq nR.C$ $\leq nR.C$	$\geq 3hasChild.Male$ $\leq 2hasChild.Female$

A role hierarchy *hasFather*$\sqsubseteq$*hasParent* provides information that a role *hasFather* is subsumed by a role *hasParent*. In a role chains we can express a propagation of one property along another one. In our example, properties *hasParent* and *hasGrandparent* propagate along property *hasAncestor*. A role *hasChild* is an inverse role of *hasParent* whereas *hasAncestor* is a transitive role. Description $\leq$*1 hasFather* denotes that a person can have at most one father. Number restriction $\geq$*3 hasChild* describes the concept that there should be at least three children while $\leq$*2 hasParent* denotes that each person have at most two parents. Qualified number restriction $\geq$*3hasChild.Male* expresses that there are at least 3 sons whereas $\leq$*2hasChild.Female* describes that there are at most two daughters.

2.1.5.2 Semantics

The formal semantics of description logics is defined by *interpretations* $\mathcal{I}$ which come from the fact that DLs are a subset of the first-order logic.

Definition 2.34 (DL interpretation)**.** *An interpretation* $\mathcal{I} = \triangle^{\mathcal{I}}, \cdot^{\mathcal{I}}$ *consists of a non-empty set* $\triangle^{\mathcal{I}}$ *called the domain of* $\mathcal{I}$*, and a function* $\cdot^{\mathcal{I}}$ *that maps every*

DL-concept to a subset of $\triangle^{\mathcal{I}}$*, and every role name to a subset of* $\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$. [van Harmelen 2007]

Two concepts C, D are *equivalent* ($C \equiv D$) if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$. An inclusion (subsumption) $C \sqsubseteq D$ is satisfied in $\mathcal{I}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A *general concept inclusion* axiom (GCI) is an expression of the form $C \sqsubseteq D$, where C and D are arbitrary DL concepts.

The semantics of constructors available in $\mathcal{ALC}$ and its extensions is presented in Table 2.1.5.2.

Table 2.7: The semantics of description logics.

Expression	Syntax	Semantics	Sym.
Universal concept	$\top$	$\triangle^{\mathcal{I}}$	$\mathcal{ALC}$
Bottom concept	$\bot$	$\emptyset$	
Atomic concept	A	$A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$	
Atomic role	R	$R^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$	
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	
Exists restriction	$\exists R.C$	$a \in \triangle^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}$ *and* $b \in C^{\mathcal{I}}$	
Value restriction	$\forall R.C$	$a \in \triangle^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}}$ *implies* $b \in C^{\mathcal{I}}$	
Negation	$\neg C$	$\triangle^{\mathcal{I}} \setminus C^{\mathcal{I}}$	
Role hierarchy	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$	$\mathcal{H}$
Role chains	$R_1 \circ ... \circ R_n \sqsubseteq S$	$R_1^{\mathcal{I}} \circ ... \circ R_n^{\mathcal{I}} \subseteq S^{\mathcal{I}}$	$\mathcal{R}$
Inverse role	R^-	$(a,b) \mid (b,a) \in R^{\mathcal{I}}$	$\mathcal{I}$
Transitive role	$R \in R^+$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$	$\mathcal{S}$
Functional role	$\leq 1R$	$a \in \triangle^{\mathcal{I}} \mid \mid \{b \mid (a,b) \in R^{\mathcal{I}}\} \mid \leq 1$	$\mathcal{F}$
Nominals	$a_1, ..., a_n$	$a_1^{\mathcal{I}}, ..., a_n^{\mathcal{I}}$	$\mathcal{O}$
Number restriction	$\geq nR$ $\leq nR$	$a \in \triangle^{\mathcal{I}} \mid \mid \{b \mid (a,b) \in R^{\mathcal{I}}\} \mid \geq n$ $a \in \triangle^{\mathcal{I}} \mid \mid \{b \mid (a,b) \in R^{\mathcal{I}}\} \mid \leq n$	$\mathcal{N}$
Qualified number restriction	$\geq nR.C$ $\leq nR.C$	$a \in \triangle^{\mathcal{I}} \mid \mid \{b \mid (a,b) \in R^{\mathcal{I}}$ *and* $b \in C^{\mathcal{I}}\} \mid \geq n$ $a \in \triangle^{\mathcal{I}} \mid \mid \{b \mid (a,b) \in R^{\mathcal{I}}$ *and* $b \in C^{\mathcal{I}}\} \mid \leq n$	$\mathcal{Q}$

A description logic knowledge base (KB) consists of two parts: terminological part (called the *TBox*) and assertional part (called the *ABox*). A TBox is a set of axioms dealing with how concepts and roles are related to each other (in other worlds, it is a finite set of both type of axioms: general concept inclusion and role inclusion). An ABox is a set of axioms asserting that: an individual is an instance of a given concept; and a pair of individuals is an instance of a given role. A TBox can be perceived as a schema of a KB which expresses domain knowledge whereas

an ABox constitutes data which store information about individual objects in the domain.

Definition 2.35 (Assertional axiom, model)**.** *An assertional axiom is of the form a : C or (a, b) : r, where C is an DL-concept, r is a role name, and a and b are individual names. A finite set of assertional axioms is called an ABox. An interpretation $\mathcal{I}$ is a model of an assertional axiom a: C if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and $\mathcal{I}$ is a model of an assertional axiom (a, b) : r if $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$; $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it is a model of every axiom in $\mathcal{A}$.* [van Harmelen 2007]

More intuitive notation of ABox axioms, which can be found in the literature [van Harmelen 2007] is, e.g., *C(a)* and *r(a, b)*. For example, if we want to express that *Chris* is a father of *Mike*, we can make the following assertions: $Child(Mike)$, meaning that Mike is a child; $Father(Chris)$, meaning that Chris is a father; and $hasFather(Mike, Chris)$ which express the fatherhood relationship between these two individuals.

The use of *concrete domains*, as mentioned in Section 2.1.5.1, which allows to use *concrete datatypes* is very important in practical applications. It provides real numbers, integers, strings and other types of data as well as concrete predicates defined on these sets, e.g., numerical comparisons, string comparisons or comparisons with constants.

Definition 2.36 (Concrete domain)**.** *A concrete domain is a pair $(\triangle_D, \Phi_D)$, where $\triangle_D$ is an interpretation domain and Φ_D is a set of concrete domain predicates with a predefined arity n and an interpretation $d^D \subseteq \triangle_D^n$. An admissible concrete domain D is equipped with a decision procedure for checking satisfiability of finite conjunctions over concrete predicates.* [Motik 2004]

Description logics are based on so-called *open world semantics (OWA)* which is in opposite to the closed world semantics that occur in relational databases and logic programming [Lifschitz 1996]. It means that information in a DL knowledge base is considered to be incomplete. Under OWA, every information has to be explicitly defined (even the negation of some fact). The absence of information only indicates lack of knowledge and it is considered as *unknown*.

2.1.5.3 Reasoning

The important feature of DL system is that it not only stores KB of a given domain but it offers to perform reasoning tasks for TBox and ABox. Typical reasoning tasks for TBox are to check the satisfiability of a given description (i.e., non-contradictory), and whether one description subsumes another one (whether one is more general than the other one). Important tasks for an ABox are to determine whether it is *consistent* (it has a model), and whether a particular individual is

an instance of a given concept description. Satisfiability and consistency tasks are performed to determine whether a knowledge base is meaningful at all. Subsumption tasks are useful to construct a hierarchy of concepts (and hierarchy of roles, if $\mathcal{H}$ is included). A concept description can be perceived as a query, describing a set of objects, which one is interested in. As a result, using instance tests, one can retrieve the set of individuals that satisfy the query.

A DL knowledge base, comprising TBox and ABox, provides semantics which makes KB equivalent to a set of axioms in first-order logic. According to this, it contains implicit knowledge which can be made explicit through reasoning.

Different reasoning algorithms for DL knowledge bases have been proposed. The use of concrete algorithm depends on constructors that are used in a particular DL [Baader 2003]. The subsumption of concepts in simple languages with little expressivity (without negation and disjunction) can usually be computed by so-called *structural subsumption algorithms*. These algorithms compare the syntactic structure of (possibly normalized) concept descriptions. More expressive languages can be handled by so-called *tableau-based algorithms*. The first tableau-based algorithm for description logics was proposed in [Schmidt-Schauss 1991] for satisfiability of $\mathcal{ALC}$-concepts. Since then, many approaches has been presented for DLs extending $\mathcal{ALC}$: for languages with number restrictions [Baader 1999], transitive roles [Horrocks 1999] and many more [Baader 2003], among them extensions to the consistency problem for ABoxes [Haarslev 2000].

General idea of a tableau-based algorithms relies on a set of *tableau expansion rules*. These rules correspond to the logical constructors of a given DL language. A typical algorithm starts from a generated individual a and imposes the constraint $a \in C$ on it. Next, using inference mechanism, which consist of applying a set of expansion rules, the algorithm tries to prove the satisfiability of the concept C by constructing a model, an interpretation $\mathcal{I}$ in which $C^{\mathcal{I}}$ is not empty. Concluding, "a *tableau* is a graph which represents such a model, with nodes corresponding to individuals (elements of $\triangle^{\mathcal{I}}$) and edges corresponding to relationships between individuals (elements of $\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$)" [Baader 2003].

Other approach, not tableau-based, tries to transform a DL knowledge base to disjunctive Datalog and performs reasoning techniques from deductive databases [Hustadt 2004, Hustadt 2007]. Disjunctive Datalog reduction and approaches that try to combine DLs with Datalog and logic programming are of a special interest for the thesis and are presented in Section 2.1.6.

2.1.6 Combining Description Logics with Datalog

This section presents approaches that try to combine the family of description logics with a Datalog-like rules. Such combination is interesting because of advantages it can provide, e.g. instance reasoning and conjunctive query answering w.r.t.

a terminology written in some DL and with some data, integrated by a set of rules. The integration of these two formalisms is very important for the Semantic Web, where DLs are focused on conceptual knowledge and reasoning whereas rules are focused on non monotonic reasoning. As a result, many practical applications require both: DLs (in the form of an ontology) and rules (usually in the form of Horn clauses).

The integration of DLs and rules is not trivial since DLs use open world assumption whereas rules exploit a closed world assumption. Moreover, they differ in expressive power and assumptions underlying these two formalisms.

We describe several methods of this combination, among them the Horn-$\mathcal{SHIQ}$ language is of a special interest in the context of this thesis. It has been chosen as an implementation language and is presented in details. Next subsections provide a short survey of approaches that have been proposed so far.

2.1.6.1 Semantic Web Rule Language

The *Semantic Web Rule Language* (SWRL) is a combination of OWL and rules which extends, syntactically, the syntax of OWL DL and OWL Lite with additional constructs providing Horn-like rule axioms. The approach was presented in [Horrocks 2004a] (called there as OWL Rule Language) and in [Horrocks 2005] where SWRL name appeared. According to the SWRL syntax, each rule is of the following form: $B \rightarrow H$ (as in Section 2.1.1.1), where B and H are (possibly empty) conjunctions of atoms. The atoms have the form of unary or binary predicates (classes or properties) where variables, individuals and constants are allowed. "Variables in SWRL are typed: those ranging over individuals are distinct from those ranging over data-values. Variables must also be safe, in the sense that every variable in the consequent of a rule must also appear in the antecedent. Even with this restriction, however, the satisfiability problem for SWRL knowledge bases is known to be undecidable." [Krisnadhi 2011]

2.1.6.2 Description Logic Programs

Description Logic Programs (DLP) approach, proposed in [Grosof 2003], tries to reconcile description logics and rules by restricting these formalisms to the fragment that can be expressed in both of them. As a result, DLP defines a decidable intersection of description logic and logic programming. DLP imposes certain constraints on $\mathcal{SHOIQ}$ language which provide a transformation of all axioms to Horn clauses preserving the semantics of the chosen DL. DLP language provides a decidable combination of rules and description logic, achieves significant gains in reasoning complexity [Volz 2004] and, moreover, its limitations will rarely matter in practice [Hitzler 2005].

2.1.6.3 DL-safe Rules

DL-safe rules approach proposed in [Motik 2004] for $\mathcal{SHOIN}$(D) and [Motik 2006a] for $\mathcal{SHIQ}$(D) considers decidable combination of OWL DL and rule axioms. The decidability is preserved by forcing each rule to be *DL-safe* which means that each variable is bound only to individuals explicitly occurred in the assertional part of the knowledge base. It is similar to dalalog safety described in Section 2.1.2.1. It is worth noticing that only atomic concepts are allowed to occur in rules. In case when complex class description matters, they can be easily eliminated from rules by introducing a new atomic concept which is a superclass of the complex one. As a result, DL-safe approach provides more expressive power than either of the two elements alone. Moreover, the algorithm for query answering with DL-safe rules was also proposed in [Motik 2004].

2.1.6.4 Description Logic Rules

Description Logic Rules (DL Rules) proposed in [Krötzsch 2008a] relies on assumption that the body of each DL rule must be *tree-shaped*. It means that the body B of a rule "contains exactly one root t, and there is exactly one path from t to any terms in B. In this case, t is the root of B" [Krötzsch 2010]. The general idea assumes that pairs of variables connected by roles in the rule form a directed graph which should be a tree where variables are vertices and roles are edges. If the graph is not a tree then the body is not tree-shaped. The important notice is that DL Rules do not increase the expressive power of description logic. Instead of it, DL Rules identifies fragments of SWRL that can be expressed (or rather emulated) by a DL knowledge base [Krötzsch 2010]. But the combination of DL Rules with DL-safe rules provide a new tractable (polynomial) rule language called ELP. ELP provides the integration of the expressivenesses of the OWL 2 profiles: OWL 2 EL and OWL 2 RL within a rule-based formalism [Krötzsch 2008b]. This combination goes beyond the scope of this dissertation. The curious reader is referred to [Krötzsch 2010].

2.1.6.5 Horn-$\mathcal{SHIQ}$

Horn-$\mathcal{SHIQ}$, introduced in [Hustadt 2005], is a fragment of $\mathcal{SHIQ}$ in which disjunctions are not allowed. A Horn-$\mathcal{SHIQ}$ knowledge base can be translated into a set of Horn clauses. This approach is similar to Description Logic Programs but differs in the fragment of the intersection between DL and logic programming (namely, Horn logic). DLP is focused on straightforward transformation of DL into rules while Horn-$\mathcal{SHIQ}$ allows to define more complex relationships between these formalisms, e.g. it provides arbitrary use of existential quantifiers

which are not supported by DLP. Moreover, Horn-$\mathcal{SHIQ}$ encompasses also a description logic DL-lite [Calvanese 2006] (see Section 2.1.6.6), because of the introduction of recursive axioms of form (2.20). The description of Horn-$\mathcal{SHIQ}$, presented herein, is based on [Hustadt 2007], [Krötzsch 2006, Hitzler 2009a] and [Eiter 2008a].

Definition 2.37 (Horn-$\mathcal{SHIQ}$)**.** *Horn-$\mathcal{SHIQ}$ is a fragment of $\mathcal{SHIQ}$ DL and it can be translated into first-order Horn clauses only if every general concept inclusion axiom can be normalized into one of the following forms:*

$$A \sqcap B \sqsubseteq C \tag{2.19}$$
$$\exists R.A \sqsubseteq B \tag{2.20}$$
$$A \sqsubseteq \forall R.B \tag{2.21}$$
$$A \sqsubseteq \exists R.B \tag{2.22}$$
$$A \sqsubseteq \geq mS.B \tag{2.23}$$
$$A \sqsubseteq \leq 1S.B \tag{2.24}$$

where A, B, C are concept names, R is a role, S is a simple role, and $m \geq 1$.

Definition 2.37 is called a normal form of Horn-$\mathcal{SHIQ}$, to which each Horn-$\mathcal{SHIQ}$ knowledge base can be rewritten [Eiter 2008a]. The normal form can be transformed into Datalog syntax (Horn clauses). As a result, Horn-$\mathcal{SHIQ}$ knowledge base KB is satisfiable iff a translation of it into Datalog syntax $\Omega(KB)$ is satisfiable as shown in [Motik 2006a]. Details of Horn-$\mathcal{SHIQ}$ transformation into Datalog syntax can be found in [Motik 2006a] and [Hustadt 2007].

According to a normal form of Horn-$\mathcal{SHIQ}$, it provides the following expressivity of Web Ontology Language in version 1.1 (OWL 1.1):

- inclusion of simple concepts (e.g. $Man \sqsubseteq Person$)
- concept disjointness (e.g. $Man \sqcap Woman \sqsubseteq \bot$)
- domain restrictions (e.g. $\exists wifeOf.\top \sqsubseteq Woman$)
- range restrictions (e.g. $\top \sqsubseteq \forall wifeOf.Man$)
- functionality restrictions (e.g. $\top \sqsubseteq \leq 1 wifeOf$)
- participation constraints (e.g. $Wife \sqsubseteq \exists wifeOf.Man$)
- role inclusions (hierarchy of roles) (e.g. $wifeOf \sqsubseteq spouseOf$)
- inverse roles (e.g. $wifeOf \equiv husbandOf^{-}$)
- symmetric roles (e.g. $spouseOf \equiv spouseOf^{-}$)

- equivalence of roles (e.g. $spouseOf \equiv consortOf$)

As mentioned before, disjunctions are not allowed, so the following expression is not expressible in Horn-$\mathcal{SHIQ}$: $Man \sqcup Woman \equiv Person$. The loss of disjunction and imposed constraints (2.19) - (2.24) guarantee that inference tasks in Horn-$\mathcal{SHIQ}$ can be solved in polynomial time (PTIME) w.r.t. the size of data (data complexity). It is important feature which makes this formalism highly practical in use. An algorithm for conjunctive query answering, presented in [Eiter 2008a], shows that the entailment problem for conjunctive queries is EXPTIME-complete (combined complexity). As a result, Horn-$\mathcal{SHIQ}$ provides the balance between expressivity and ABox reasoning scalability.

2.1.6.6 Other approaches

Aforementioned approaches do not exhaust the research on a combination of description logics with rules. In this section we provide a short overview of other important methods in this area.

A classical work on integrating rules with DLs under the semantics of first-order logic was presented in CARIN [Levy 1996]. The work showed that it is difficult to combine these two formalisms and even a very simple description logic combined with an arbitrary Horn logic is undecidable. In [Levy 1996] a description logic $\mathcal{ALCNR}$ combined with a function-free Horn logic provides decidability with the use of *role-safe* rules which require that in every role atom at least one variable that appears in the atom also appears in an atom of a base predicate (i.e., a predicate that does not appear in the consequent of a Horn rule, and is not a concept or role predicate).

In $\mathcal{AL}$-log approach [Donini 1998] a description logics $\mathcal{ALC}$ is combined with Datalog where the safety condition occurs: each variable in the head of the rule must also occur in the body of this rule. Moreover, for a Datalog program P and ontology O the following conditions must be met:

1. The set of Datalog predicate symbols appearing in P is disjoint from the set of concept and role symbols appearing in O.
2. The constants of P coincide with individual names from O.
3. P consists of constrained clauses where each one is accompanied by zero or more constraints of the form $C(t)$, where C is an $\mathcal{ALC}$ concept description and t is a variable.

The constraints restrict the values of variables to instances of concepts. The query answering procedure, based on SLD-resolution, was also proposed in [Donini 1998].

$\mathcal{DL}$+log formalism [Rosati 2006] integrates description logics with disjunctive Datalog. The approach assumes a new safeness condition, called *weak safeness*,

which states that every head variable must occur in a non-DL literal in the rule body. As a result, weak safeness allows for the presence of variables that only occur in DL-atoms in the body of the rule. It is worth noticing that DL predicates are interpreted under open world assumption while Datalog predicates are interpreted under closed world assumption. The satisfiability problem of a $\mathcal{DL}$+log knowledge base is decidable iff Boolean query containment problem for (union of) conjunctive queries is decidable in the DL used.

Another combination of description logics and disjunctive Datalog was proposed in [Motik 2007] as hybrid MKNF (*Minimal Knowledge and Negation as Failure*) which is based on logic proposed in [Lifschitz 1991]. Hybrid MKNF approach is a seamless integration of closed- and open-world reasoning within a single framework. It is easy to switch between closed- and open-world views on arbitrary predicates from description logic and logic programming. The integration is "*faithful* in the sense that it provides exactly the same consequences as DL and LP, respectively, if the other component is empty." [Motik 2007] Hybrid MKNF formalism generalizes other approaches mentioned earlier: SWRL, CARIN, DL-safe rules and $\mathcal{AL}$-log [Motik 2006b]. A variant of hybrid MKNF approach considering nondisjunctive knowledge bases was also proposed [Knorr 2007] under well-founded semantics.

The combination of disjunctive Datalog (with negation under answer set semantics) and description logics is adopted by *dl-programs* [Eiter 2008b]. In this approach rules can contain special atoms (in their bodies) which are interpreted as queries (dl-queries) to a description logic ontology. Moreover, description logic atoms (*dl-atoms*) can specify data which is provided as input to queries, and answers to the queries affect what my be inferred using the rules. As a result, the bi-directional flow of information between DL component and Datalog program is enabled [Drabent 2009].

Some other recent approaches that try to integrate OWL 2 ($\mathcal{SROIQ}$ or a subset of it) with rules are the following:

- DAAL [Nalepa 2010] language (Description And Attributive Logic) which integrates Attributive Logic with description logic aiming that this integration can be translated and represented using XTT2 rule language [Nalepa 2008].

- ELP [Krötzsch 2008b] is an OWL-based language which combines $\mathcal{EL}^{++}$ [Baader 2008] based profile of OWL 2 with Description Logic Programs and DL-safe rules. ELP provides combined polynomial complexity.

Since we consider only formalisms that are used in OWL 1.1, these approaches are beyond the scope of this dissertation.

2.2 Description of Economic Crimes

In this section we provide descriptions of two economic crimes that we are focused on, namely, fraudulent disbursement and money laundering.

An economic crime, according to the Polish Penal Code (PC) [Sejm 1997], is a forbidden act or neglect in order to achieve economic (usually financial) gain. Such crimes affect organizations worldwide (public and private), despite increasing regulatory actions [PricewaterhouseCoopers 2009]. As a result, typical victims of economic crimes include organizations, groups of people and individuals against which the crime was directed. In some cases a group of victims is extended by ordinary citizens which are indirectly affected by an economic crime (e.g. when an illegal act was directed against a public organization).

Association of Certified Fraud Examiners (ACFE)[2] in its Occupational Fraud and Abuse Classification System divides crimes into three categories: corruption, asset misappropriation and fraudulent statements. Next, asset misappropriation is divided into cash and non-cash (in our terminology money, the cash being form of money and non-monetary assets). The large category of money related crimes are fraudulent disbursements.

According to PricewaterhouseCoopers[3] (PwC) economic crimes can be divided into: asset misappropriation, accounting fraud, bribery and corruption, intellectual property infringement, money laundering, tax fraud, illegal insider trading, market fraud involving cartels colluding to fix prices, espionage, and others. As we can see, the classification of PwC is wider than ACFE.

The global economic crime survey [PricewaterhouseCoopers 2011] shows that top three types of economic crimes are: asset misappropriation (72%), accounting fraud (24%) and bribery and corruption (24%). In comparison to the earlier report from PwC [PricewaterhouseCoopers 2009] asset misappropriation increased 5% whereas accounting fraud and bribery and corruption decreased, 14% and 3% respectively. Moreover, most frauds are committing by internal fraudsters (56%) [PricewaterhouseCoopers 2011]. As a result, economic crimes are very costly (losses may be higher than five million US dollars on average). It appears, from aforementioned reports, that economic crimes have an important influence on business, organizations and even ordinary people.

Next subsections provide a brief overview of a fraudulent disbursement (as a subset of asset misappropriation) and a money laundering.

[2] http://www.acfe.com/
[3] http://www.pwc.com/

2.2.1 Fraudulent Disbursement

A fraudulent disbursement is the most common form of asset misappropriation [ACFE 2008]. In this economic crime an employee uses his position in a company to cause a payment for some inappropriate purpose. Fraudulent disbursements are divided into the following schemes [Albrecht 2008, ACFE 2008]:

- Billing schemes which involve employers making payments based on false invoices for personal purchases.
- Payroll schemes in which payment is based on false documentation (resemble billing schemes) and indicates that compensation is fraudulently due to an employee.
- Expense Reimbursement schemes in which claims of fictitious or inflated business expenses appear.
- Check Tampering schemes in which a person steals funds by forging or altering an organization's check.
- Register disbursement schemes in which an employee makes false entries or no-sale transactions to conceal the removal of cash.

The most common fraud scheme of fraudulent disbursement is the billing scheme [ACFE 2008]. As a result, this scheme is of a special interest in the context of this thesis and we now describe it in the details according to descriptions provided in [ACFE 2008] and [Albrecht 2011]. Curious reader can find detailed information about other economic crime schemes (including fraudulent disbursement schemes) in [Albrecht 2011].

In a billing scheme the perpetrator submits or alters an invoice that causes her or his employer to issue a payment for fictitious goods or services, inflated invoices, or invoices for personal purchases. For example, an employee may create a shell company and then bills his employer for non-existent services or goods. As a result, the employer buys non-existent goods or services, overpriced or not needed merchandises.

The purpose of most billing schemes is to obtain an illicit gain of cash for the fraudster. In a typical billing scheme an employee creates fraudulent documents, which can include orders, invoices, receiving reports and so on. These documents cause the victim organization to make a money transfer or to pay in cash. Usually, the cash hits the perpetrator's pocket. In some cases perpetrators create a dummy (shell) company which send invoices to the victim organization. As a result, the money goes to the account of the shell company. Another way to achieve an illicit gain is to generate fraudulent disbursement by using the invoice of a non-accomplice vendor or by paying some invoice twice and then asking for one of the

payment to be returned to the perpetrator's account. In case, when the documents were issued to purchase personal goods or services, the perpetrator falsifies the nature of the order, claiming that they were bought on the organization's behalf. In all cases the perpetrator achieves an illicit gain whereas the employer is the victim.

Summarizing, the categories of billing schemes include:

- Setting up shell companies to submit invoices to the victim organization
- Paying a non-accomplice vendor's statements
- Making personal purchases with organization's funds

Billing schemes are very expensive, the median cost of billing schemes in the ACFE Report to the Nation on Occupational Fraud and Abuse [ACFE 2012] is 100.000 US dollars. Moreover, "since the majority of most businesses' disbursements are made in the purchasing cycle, larger thefts can be hidden through false billing schemes than through other kinds of fraudulent disbursements. Employees who utilize billing schemes are just going where the money is." [Albrecht 2011]

2.2.2 Money Laundering

Money laundering is the most common tax fraud scheme [PricewaterhouseCoopers 2011] which consist of actions that try to legitimise the proceeds of a crime by concealing the source, identity or destination of funds. The phrase "money laundering" indicates money which is "dirty" because it was generated illegally. Money which is "laundered" is made to appear that it has legitimate origin [Albrecht 2011]. We now describe the money laundering process and schemes according to descriptions presented in [Salinger 2004], [Albrecht 2011], [FATF 2012] and [JMLSG 2012].

The process of legitimizing illegally gained money involves the number of various transactions and actions. Usually, it consists of the following three steps:

1. Placement. In this stage, the launderer inserts "dirty" money into a legitimate financial institution. Such action requires making cash deposit to a bank, This stage is the riskiest one because large cash deposit may rise so-called "red flags" and banks are required to report details of such transaction to the governmental institution (in Poland it is *Generalny Inspektor Informacji Finansowej* [4]).
2. Layering. This stage is the most complex step in the money laundering process. The launderer tries to make the dirty money difficult to trace. Various transactions are performed to distance funds from their source and to make the flow of funds difficult to follow. The funds may be transferred through

[4] http://www.mf.gov.pl/ministerstwo-finansow/dzialalnosc/giif/aktualnosci

accounts located in different countries attached to different names. Layering may also include making deposits and withdrawals, exchanging money to a new currency, purchasing high-value items (e.g. cars, jewellery, yachts) to change the form of the assets.

3. Integration. In this stage the funds re-enter the legitimate economy in a form that appears to come from a legal transaction. The transaction may involve investing in some business or the sale of assets bought in the previous stage. As a result, the money reintroduced as "clean" into the economy can be consumed by the launderer. If documents that can track earlier stages do not exist, it is extremely difficult to reveal a laundering scheme.

As we can see, money laundering refer to a wide area of illegal activities and it is not only associated with mafia or drug dealers but often with prominent and respectable individuals and organizations.

Generally speaking, there are as many different money laundering schemes as there are people involved in them [Albrecht 2011] but we can distinct four common techniques [Salinger 2004]:

1. Bank methods which include various exchange transactions in which cash is converted, for example wire transfers. In this method the launderer is required to change cash into any non-cash form.
2. Smurfing method which is based on making many cash deposits into several bank accounts. Each deposit is required to be under the minimum amount that banks report to the governmental institution. As a result, a significant amount of cash can laundered without being detected.
3. Currency exchange method in which cash in domestic currency is exchanged into foreign currency and later processed through banks. Converted cash can be introduced into legitimate banking channels and consumed by the launderer.
4. Double-invoicing method in which the launderer hides financial gains by sending invoices to a domestic or foreign organization for an overpriced merchandises and the depositing the profits. As a result, the source of the funds is disguised by manipulating business documents (invoices, account books etc.).

Money laundering often takes place across many countries. That is why the Financial Action Task Force[5] (FATF) was established in 1989 as an international governmental body that strives to combat money laundering. FATF publishes recommendations on fighting with money laundering and even the financing of terrorism. A detailed list of the FATF recommendations can be found on the FATF website.

[5] http://www.fatf-gafi.org/

Money laundering often accompanies other illegal activities. During an inquiry, investigators are able to discover the number of crimes and criminals by uncovering a money laundering scheme. An example case of fraudulent disbursement accompanied by money laundering is presented in Section 3.1.

Both economic crimes, fraudulent disbursement and money laundering, have a damaging effect on business, financial institutions as well as ordinary customers and citizens. This is the reason why we are interested in support investigators in the process of analysing data at the semantic level with the rule-based query answering methods.

CHAPTER 3

Knowledge base of economic crimes

In this chapter we present the ontological modelling of knowledge concerning the class of linked economic crimes, namely the fraudulent disbursement accompanied by money laundering. We present the example real world crime case - the Hydra case in which both crimes occurred. We describe the main parts of the created ontology and rules: determining legal sanctions for crime perpetrators, discovering crime activities and roles of people engaged in a crime.

Economic crimes are particularly difficult to model [Kingston 2005] and code into an expert system. For example, fraudsters use many types of schemes, techniques and transactions to achieve their goals, so it has seemed impossible to construct a simple conceptual model of any generality. Only recently has the integrated use of semantics expressed by means of ontologies and rules achieved the capability of analysing large practical problems, such as applying reasoning over legal sanctions on the basis of investigation facts and rules appearing in penal codes.

It is worth noticing that a definition of money laundering differs between jurisdictions [FATF 2012]. The Polish Penal code [Sejm 1997] defines it in a way similar to the UK law as taking any action with property of any form which is either wholly or in part the proceeds of a crime that will disguise the fact that the property is the proceeds of a crime. Here we restrict the notion to engaging in financial transactions to conceal the identity, source, or destination of illegally gained money.

Our approach is based on the suitable application of an ontology that forms a "minimal layer" - it contains only necessary concepts that follow the logical order of uncovering a crime. The approach is so-called the *Minimal Ontology Model* (MinOn). The ontology contains necessary concepts, roles and rules for taking decisions in companies of various sizes and activities related to transactions and whether persons in these companies formally documented their activities. The ontology makes it possible to differentiate roles of key people in the crime scheme, and map their crimes into a specific set of penal code articles in differentiated ways. Moreover, the MinOn ontology was established on a basis 10 large cases for which investigation, indictment and sentence data had been complete (or almost complete). Out of these cases the *Hydra case* was related as the most general, and delivering rich enough crime scheme.

We apply the MinOn model to describe over 85% of relevant information for the *Hydra* case, and, as a result, we are able to effectively infer from these facts

the legal qualifications for this case. As a result, the model is realistic and contains many variants of a given crime typology.

The conceptual minimal ontology model consists of eight layers of concepts, structured in order to use available data on facts to uncover relations. Using these concepts and appropriate relations and rules, we are able to map crime activity options (roles of particular type of managers). This makes it possible to phrase these roles in the language of penal code sanctions. Finally, the roles of persons in the crime are mapped into a set of sanctions.

In Section 3.1 we present a real crime case, so-called the *Hydra Case*. In Section 3.2 we describe the employed methodology to construct the minimal ontology model. Section 3.3 contains the detailed description of the MinOn model where we derive rules that define logical activities appearing in the Penal Code based on physical activities (e.g. signing a document that contains untrue content is a falsification). Additionally, the section is devoted to mapping logical activities corresponding to legal sanctions for crime perpetrators. Conclusions and future work are presented in Section 3.5.

3.1 The Hydra Case

In this section we describe the most clean real case of fraudulent disbursement accompanied by money laundering, which is so-called *Hydra Case* (HC) [Więckowski 2009].

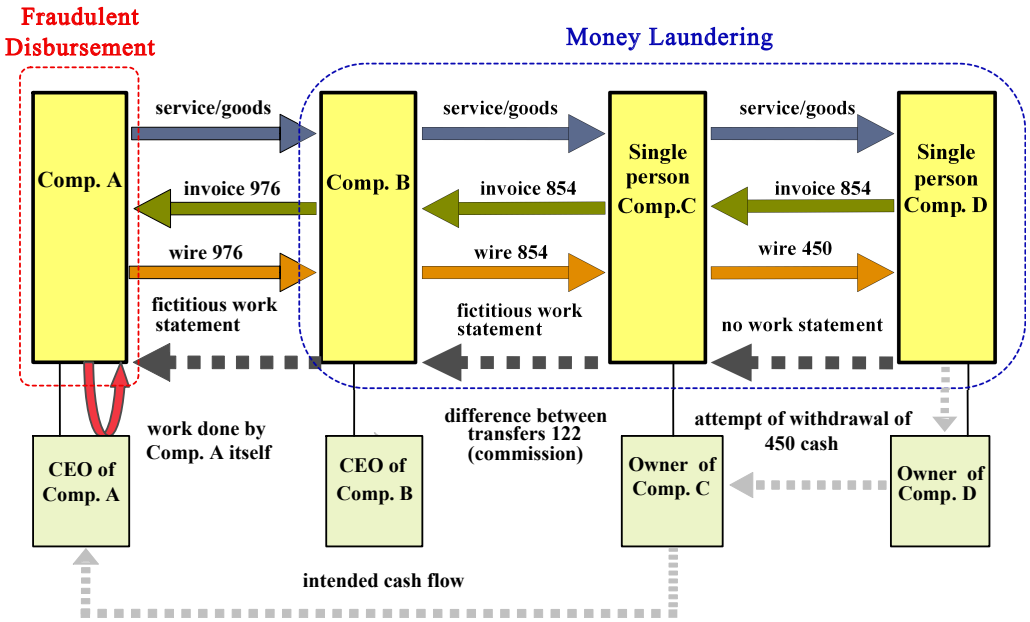

Figure 3.1: The Hydra case [Martinek 2008].

The (whitened) facts of the economic crime considered here were initially written on 7 pages in a textual form by a public prosecutor and then processed by a knowledge engineer [Martinek 2008], to obtain a description in a natural-like language (with sentence schemes like "Company *C1* hires company *C2* to do work *W* at location *L*."). On the basis of facts the crime scheme in Figure 3.1 was created, which illustrates acting agents (companies and persons) and chains of their activities. These activities are divided into 4 types of flows of: services/goods (here, the construction work), invoices accompanied by money amounts, wired money amounts and authorized legal documents concerning the services/goods. In the real case, fitting to this scheme, the Chief executive officer, (CEO) of company A (*Hydra*) subcontracts construction work. The work is then consecutively subcontracted through a chain of phony companies B, C, and D (Hermes, Dex, Mobex). Each company is getting a commission for money laundering and falsifies documents stating that the contracted work had been done. Actually, what was to be done as "subcontracted construction work" company A did itself. At the end of the chain, the owner of a single person company D attempted to withdraw cash, and there was a suspicion that this cash had reached the management of company A "under the table" ("intended cash flow" in Figure 3.1).

The skeletal story of the Hydra case, depicted in Figure 3.1 is described as follows (not all details are shown, i.e. family relationships between persons, levels of responsibility in a company, company's financial activities etc.; also, no temporal relations are given):

- Company A hires company B to do work W at location L.
- Company B hires single person company C to do work W at location L.
- Single person company C hires single person company D to do work W at location L.
- Company A does work W at location L.
- Company B does no work W at location L on behalf of A.
- Single person company C does no work W at location L on behalf of B.
- Single person company D does no work W at location L on behalf of single person company C.
- Persons P1 and P2 representing companies A and B respectively, sign work acceptance document D1 related to work W not done at location L.
- Persons P3 and P4 representing companies B and C respectively, sign work acceptance document D2 related to work W not done at location L.
- Single person company D issues invoice I1 to single person company C for work W not done at location L.
- Single person company C issues invoice I2 to company B for work W not done at location L.

- Company B issues invoice I3 to company A for work W not done at location L.
- Company A makes payment to company B for invoice I1 related to work W not done at location L, transferring money from account A1 in bank B1 to account A2 in bank B2.
- Company B makes payment to single person company C for invoice I2 related to work W not done at location L, transferring money from account A2 in bank B2 to account A3 in bank B3.
- Single person company C makes payment to single person company D for invoice I3 related to work W not done at location L, transferring money from account A3 in bank B3 to account A4 in bank B4.
- Person P5 is the owner of single person company D.
- Person P5 gives an order to bank B4 to perform operation O on account A4 of single person company D existing in this bank.
- O is a cash withdrawal from account A4 in bank B4.
- Bank B4 blocks operation O on account A4 of single person company D existing in this bank, ordered by person P5.

In the next stage the sentences were broken into RDF triples, which were the basis for concepts, relations and rules design. Such a representation largely facilitates asking relevant questions about connections between financial entities and people associated with them, which is conducive to evidence building and assigning a sanction for a crime.

The Hydra case is simple but powerful crime scenario in the form of asset misappropriation. We decided to use this case as a base of the MinOn model, since it is the most clean case of fraudulent disbursement accompanied by money laundering. The legal implications of the model has been verified by legal experts within the Polish Platform for Homeland Security framework.

Next section describes the employed methodology in ontology creation supported by rules.

3.2 Ontology Design Method

In contemporary information systems not only structures and formats of the processed data should be explicitly given – the system's data and processes are to be also semantically specified, forming the conceptual model of the system [Goczyła 2011]. Such a model may be defined in the form of an ontology and used during the creation of the system or its exploitation or both. Our aim is to have an ontological model that aims at conceptualizing economic crimes in order to support the exploitation phase of the police system under construction [Cybulka 2008].

Particularly, having described the semantics (with the relevant ontological expressiveness) of crime-sensitive data, we can automate: the acquisition of this data to the system, the extraction of data from the system (by means of semantic queries) and possibly the data exchange between similar systems (in the paradigm of the semantics-directed translation). The police system is to support the teamwork to collectively detect the crime schemes on the basis of the gathered data. Here, ontology also has its role by means of giving a model of a workflow among collaborating users ('agents') [Cybulka 2009]. The legal sanctions levied against people engaged in a crime may be deduced with the help of rules that are also part of the ontology. Now, we use our model in experiments that are the part of the prototyping phase, so currently this ontology is not linked to the Police system.

The ontology is created according to the chosen method, the application of which results in obtaining a layered ontological structure with the foundational level on top of it and the application level at its bottom. The foundational level semantic entities were identified with the use of content design pattern technique of the adopted method, namely the pattern of constructive descriptions and situations (c.DnS). A content design pattern serves as a conceptual mean (which determines a conceptual expressiveness of the ontology) that is suitable to represent a chosen view on reality. Every foundational ontology involves such means and may be treated as a content design pattern. The only operations that we do on content patterns are their specialization ("is subsumed by") and linking them via the basic formal relations (coming from the foundational ontology). The application level entities of our layered ontological structure, concerning real crime cases, were manually separated from the motivating crime scenarios. As usual, this level may be divided into two parts: a domain-based and a task-based one. The former concerns the conceptualization of "a domain" whose attributes and relations are of interest while the latter supports the realization of the functionality of a certain application ("a task"). The domain-based ontology is engineered in the OWL language for we want to "name" concepts, relations and their instances. The task-based part is implemented via rules in SWRL language.

The foundational level of economic crimes ontology is beyond the scope of this thesis since we are focused on the application level. In this thesis we present the MinOn model which consists of two parts: domain-based part and task-based part. More detailed information about the foundational level ontology can be found in [Cybulka 2009], [Cybulka 2010b] and [Bak 2013]

3.2.1 Ontology Overview

In information technology, an ontology is a set of concepts and roles in some particular domain of knowledge. The ontology defines domain knowledge (objects and properties) and also should provide operational knowledge on use (how do we

use the objects?, what answers can we get?, and how could we query?). In general, the model represents machine readable projection of a larger domain expressed in a formalized language. We follow less general but more practical bottom-up path. We are interested in legal case description we build hierarchy of objects possessing inheritance, along with their properties such as attributes, and restrictions that apply to the class. We apply rules to support reasoning about combined each other roles and concepts.

We concentrate on most precise description of a single case. However, we do not limit ourselves with only facts of the case. We consider also possible variants of the case together with their legal implication.

The MinOn model has been developed in language Horn-$\mathcal{SHIQ}$ (tractable part of OWL 1.1) which supports maximum expressive power (in rule-based representation of an ontology) without loss of decidability and computational completeness. We define Horn clauses in SWRL language, which extends the expressivity of OWL supporting the use of ontology axioms in rules. We also use SWRLB language (SWRL Built-ins) to extend SWRL with additional functions. Generally, we use SWRLB Comparisons to compare variables and to put some constraints on them.

Such a model with ontology and rules needs an appropriate reasoner. There are several of them, for example: Pellet [Parsia 2012] and KAON2 [KAON2 2012]. Our data are stored in a relational database, so we have to use a reasoner which supports querying 'on-the-fly' according to defined semantics and with the use of rules. We decided to use our own tool - the Semantic Data Library which is presented in details in Chapter 5.

We stress that some of the presented facts, for example *ApprovalOfWorkNotDone(?d)*, will be put into the system by a prosecutor or other person connected with an investigation. We also want to mention that all variables appearing in rules which have different names are treated as having different values. To express that we need to use SWRLB constructions (for example: swrlb:equal ('='), swrlb:notEqual ('!='), swrlb:greaterThan ('>') etc.).

3.2.2 Adopted Method

To build the ontology of economic crimes (both domain and task parts) the elements of the engineering methods proposed in [Gangemi 2007, Gómez-Pérez 2008] were used. The NeOn method suggests formulating the specification of an ontology by expressing:

1. Purpose of the ontology.
2. Scope of the ontology.
3. Level of formality.

4. Intended users of the ontology.
5. Intended uses ("functionality") of the ontology.
6. Requirements specification (in the form of competency questions and answers).
7. Pre-glossary of concepts and individuals.

During the specification phase of an ontology creation, the source knowledge is processed, enabling to single out relevant 'pieces of semantics' that are to be put into the ontology. The formulation of the ontology specification is as follows.

Purpose, scope and level of formality of the ontology. The fraudulent disbursement's and money laundering ontology aims at providing the knowledge model of this kind of economic crime. The scope is determined by the motivating scenario, which is a description of one particular case of a crime that is an instantiation of one scheme of such an offense. Such a representation largely facilitates asking relevant questions about connections between financial entities and people associated with them, which is conducive to evidence building and assigning a sanction for a crime. The ontology is formalized in OWL-DL and SWRL ontological languages.

Intended users of the ontology. Among the users we distinguish crime domain experts (who create crime scenarios) and also humans and software agents who communicate with the knowledge base. The software agents mentioned are processes of automate knowledge acquisition, extraction and exchange.

Intended uses of the ontology. Ontology may be used as a semantic reference for domain experts who organize and communicate knowledge concerning new cases of fraudulent disbursement crime. For example, with the existent *c.DnSForAFraudulentDisbursement* tuple in mind, the expert, if necessary, can specialize it with new concepts or can build a new c.DnS "capsule" and link it to the existing ones via the existent formal relations. Also, such experts can query the knowledge base (see Chapter 6) containing data related to crimes, with the use of the defined rules.

Requirements specification and pre-glossary of terms. While the domain-based part of the ontology is embedded in the foundational level, the task-based part is crafted strictly to a task, meaning it uses only necessary concepts that follow in the logical order of uncovering a crime. At the first stage, goods or services transfer data is analysed with relation to three basic flows: of money, of invoices, and of other documents (i.e., confirming that the service or goods have been delivered). Also, at this stage responsible or relevant people within companies are

identified and associated with particular illegal activities. The major features of this part of the ontology are the following:

- only facts contributing to evidence or possible sanctions are kept
- answers to difficult questions are left to a human: i.e. deciding whether the work has or has not been done; (this requires sending an expert to the field who will do a construction inspection, taking testimonies, finding that a company that presumably did the job was a straw company, i.e. with no experience in construction, having no equipment, etc.; in some cases finding out that the work was underpriced or overpriced is very difficult but a critical issue in a case)
- considered events or attributes are reduced to a minimum, for example:
  - at the first stage of the Hydra case analysing it was not necessary to deal with the place of construction, for the scheme would be a crime no matter where the construction was taking place (for a given jurisdiction); however, this information has to appear in the indictment
  - an invoice can be issued or received, we combine these two events into a single one; invoices may be lost or destroyed – in cases for which these facts will be of importance, and then possibly we would have to enhance the model
- knowledge about the case appears explicit as presented by facts, and implicit – such as regular business procedures; once the payment is approved, it is then executed and we are not interested who actually did it; such an approach of complementing a scenario with "external knowledge" is similar to that taken in Abraxas project [Aleven 2003]; this spares us representing a trade code.

The competency questions that lay the ground for our model are:

- Between what entities (companies and people) are the transactions?
- What is a record of business activities and bank accounts of these entities?
- What is a record of tax statements of these entities?
- What are subjects of transactions?
- What was the ground for payments?
- What are documents of transactions?
- What is a hierarchy of management in involved companies?
- What is the decision structure and who (meaning positions, not people) authorizes particular decisions (signs relevant documents) within the structure?

- Which persons can be associated with relevant activities (for a given crime mechanism)?
- Who knew about these activities?
- Who could possibly benefit from a crime?
- What are possibly legal sanctions related to a given crime typology?
- Who are accomplices in wrong doing?
- What were the roles of crime perpetrators (organizers, helping parties, straw companies and straw persons)?

3.2.3 Applied Rules

Rules play a very important role in the layered architecture of the Semantic Web. They are used for freely mixing of property and class expressions which is not allowed in OWL. Generally, rules in the Semantic Web are needed for:

- inferencing about OWL properties and classes,
- mapping ontologies in data integration,
- transforming data from one to another format,
- querying with the use of complex queries based on OWL, SWRL etc. axioms,
- and many more.

Usually, rules are distinguished into deduction rules, production rules, normative rules, reactive rules, defeasible rules, etc. In our approach we apply two kinds of rules: deduction and production rules. We use deduction rules to infer about facts in the knowledge base. They add new implicit statements about connections between persons, documents, money transfers and legal sanctions. According to them we can discover crime scheme and suggest legal sanctions for people involved in crime. These rules are defined in SWRL language with the use of SWRL Built-ins. Deduction rules are also used for querying Jess engine's working memory. Query rule contains only the body part and after hybrid or extended rules reasoning (executed by SDL with Jess) activations of this rule are obtained as query results.

Production rules are used for mapping between ontology axioms (properties and classes) and data stored in a relational database. These rules are defined in Jess language. Their creation is supported by the SDL-GUI module which is the part of the SDL tool. Mapping utilizes simple rule that every "essential" axiom (property or class) has defined appropriate SQL query for mapping (see Section 4.4).

It is worth noticing that some of the presented rules contain several predicates in the head. In this case, rules can still be considered as Horn clauses, since they can be easily transformed into Horn clauses with the Lloyd-Topor transformation

[Lloyd 1984]). Several predicates in the heads of rules is employed for the sake of increasing their readability.

3.3 Minimal Ontology Model

The minimal ontology model consists of eight layers that are structured in order of uncovering the facts and are presented in Table 3.1. The competency questions presented in Section 3.2.2 were related to top five levels of ontology. We could ask more detailed questions. For example, what is additional information relevant to sanctions (criminal records, relapse into crime after having served a sentence, coercion on some persons by other perpetrators)? In this thesis such information belongs to levels 7 and 8 of the ontology structure and was not dealt with in the present MinOn model.

Table 3.1: Layers of concepts for analysis of economic crimes.

Type	Concern details
1.	General entities as: Companies, Institutions, Single person companies, levels of authorization, documents having legal meaning. Money transfer between companies.
2.	Invoice flow between companies. Tax statements.
3.	Work/Services flow.
4.	Roles of decisive people in companies who accepted work in the chain of command.
5.	Mapping potential roles coming from positions in companies to particular activities resulting in a financial crime.
6.	People not related to companies but being a part of crimes. Other relations of people.
7.	Information about people, e.g., whether they were sentenced in the last 10 years, their criminal connections; school or business etc., connections.
8.	Additional factors (e.g., learning about averted criminal plans).

A definition of the MinOn model in application to financial crimes, expressed in Horn-$\mathcal{SHIQ}$ language using the editor Protégé 4.1[1] has a modular structure and contains the following modules:

- Person.owl, describing persons as social entities and groups of persons,
- Document.owl, specifying the legal meaning of documents and their content,
- LegalProvision.owl, defining legal acts and sanctions,
- Action.owl specifying activities,
- Object.owl describing other entities, i.e. goods (work or service),
- MinimalModel.owl defining general concepts and roles, it also contains rules,
- Institution-Organization.owl describing legal entities (rather than dealing with intentions, it is more important to establish who knew about criminal activities, and whether a crime was perpetrated by a group).

As relates to sanctions they are specified by a certain number of rules that define what are conditions of a given crime, what constitutes evidence and how various activities have to combine to be subjected to a particular sanction.

For example, as is stated in many legal theory texts, fraud must be proved by showing that the defendant's actions involved between five to nine [Podgor 1999] separate elements presented in Table 3.2.

3.3.1 Domain-based Part of the Ontology

The domain-based part of the MinOn ontology contains necessary concepts and roles to describe the crime schemes of fraudulent disbursement accompanied by money laundering.

Recalling the requirements regarding the ontology (see Section 3.2.2), we formulated a sequence of competency questions to be answered on the basis of information contained in the formalized statements. As it was stated there the questions serve to reveal only "necessary concepts that follow in the logical order of uncovering a crime". For example, on the basis of the presented Hydra formalization the following questions may be asked. Answers to them enabled to distinguish three basic groups of ontological concepts.

1. Q1: What are the general entities?
   The following social agents, abstracts and actions:
   - company
   - single person company
   - institution (a bank)

[1] http://protege.stanford.edu/

Table 3.2: Fraud attribute representation.

No.	Fraud attribute	How it is determined
1	A false statement of a material fact (in some works these two are separated)	Explicitly (falsified document as proof).
2	Damage to the alleged victim as a result	Explicitly (payment for work not done).
3	Knowledge on the part of the defendant that the statement is untrue	Conditional (could know, could not know, should know – these are the variants of the scheme analysed).
4	Intent on the part of the defendant to deceive the alleged victim	Testimony.
5	Justifiable reliance by the alleged victim on the statement	Implicitly (had fraud been committed payment is automatic – this knowledge comes from normal operation procedures in a company).

- decision maker (representative) in a company
- (legally valid) documents
- money transfer (between companies)
- bank account.

2. Q2: What are the main document's flows?
   The following actions:
   - invoice flow between companies
   - work acceptance flow between companies
3. Q3: What are other important flows?
   The following action:
   - goods/services flow.

Let us look in more detail at concepts connected with social persons and documents. Figure 3.3 and Figure 3.2 present taxonomies of these type concepts. The

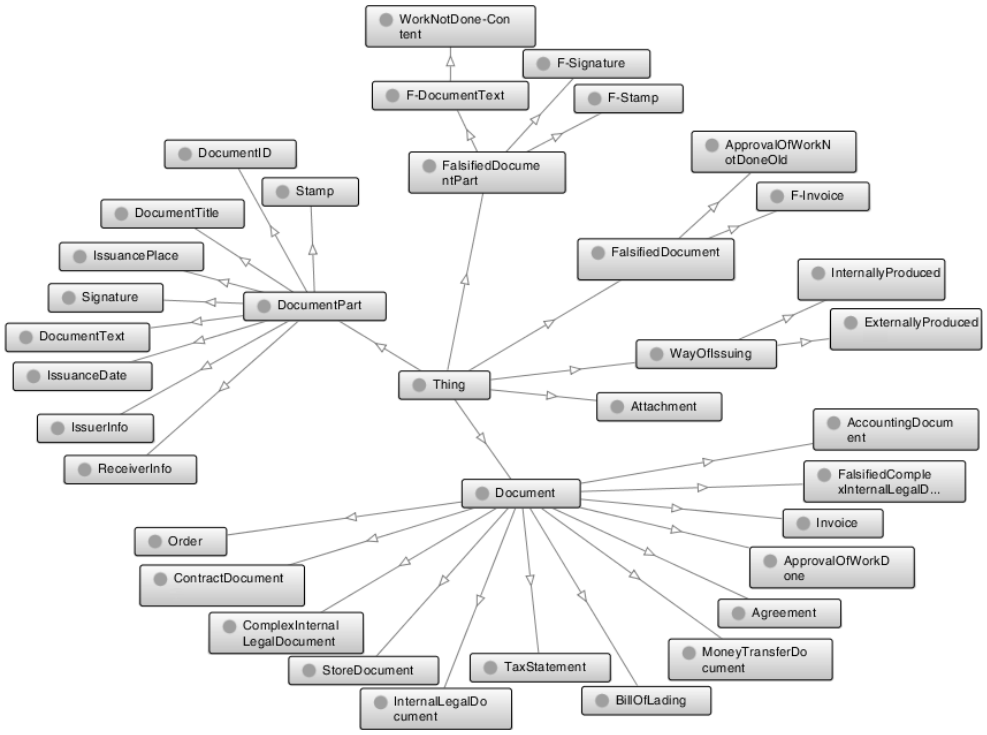

Figure 3.2: Taxonomy of concepts concerning documents.

former lists social persons while the latter – documents. Social persons and documents mentioned here constitute a fraudulent disbursement situation. We assume that a company has a multi-level structure of authorization. In the case of Hydra it is a three-level structure entailing the following chain of activities: acceptance of the construction work done by a company B at a given site is first signed by a manager in a company A responsible for a work supervision at this site (MiddleLevelManager); this is followed by a signature of the higher level manager – a Director of the company responsible for supervision of all sites. A Director may be authorized to accept invoices and to order a payment – technically this is and was done by a written authorization on the back of the invoice. The CompanysPrincipal might not have known that the work was not being done. However, he was the one who signed the contract for subcontracting and thus could be implicated. Had the CompanysPrincipal of A been a person who on the basis of the work acceptance document had ordered the payment of A to B, upon issuance of an invoice by B, he would be directly implicated. However, in reality the case was more complex. In order to represent elementary activities, we need to formalize the concept of a complex legal document and the concept of hierarchical chain of responsibility in a company. All of these are described by rules in Section 3.3.2.

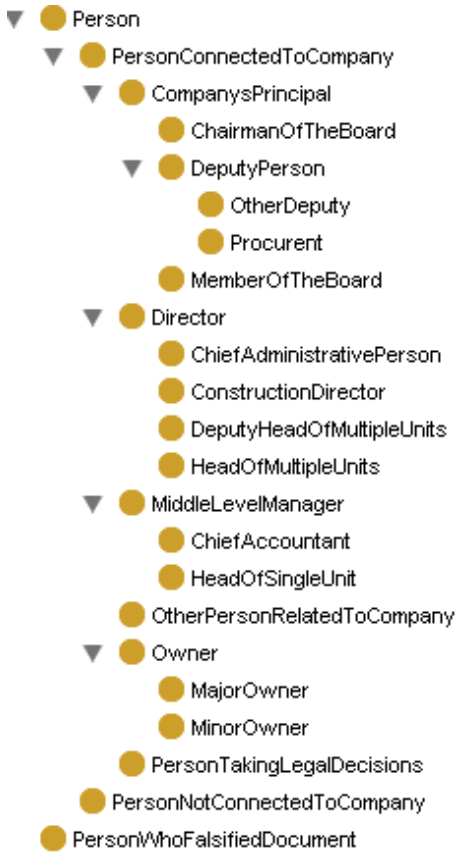

Figure 3.3: Taxonomy of concepts concerning social persons.

MoneyTransfer concept belongs to general semantic entities distinguished on the basis of the question Q1. We give it the following definition (3.1). This definition means that a money transfer has one distinctive value, it occurs at exactly one time instant between a pair of companies, and it is connected with paying for an invoice.

$$
\begin{aligned}
MoneyTransfer \sqsubseteq\ & \\
& \exists flowsFrom.Company \sqcap \\
& \forall flowsFrom.Company \sqcap \\
& \exists flowsTo.Company \sqcap \\
& \forall flowsTo.Company \sqcap \\
& (= 1\ occurs \sqcap \forall occurs.TimeInstant) \sqcap \\
& (= 1\ hasValue \sqcap \forall hasValue.float) \sqcap \\
& \exists isPaymentFor.Invoice \sqcap \\
& \forall isPaymentFor.Invoice
\end{aligned}
\tag{3.1}
$$

It is essential to recognize that documents may require the signing by a subset of principals within a company according to a statute. In the Hydra case the board consisted of 5 members, and the chairman of the board was authorized to sign documents without the consent of the others. Since no involvement of the remaining 4 members was found, here the principal is the CEO. We adopt the 3-level deep company management structure – the "legal view" on hierarchy, which determines corporate lines of accountability. We assume that these 3 levels of PersonTakingLegalDecisions, starting from the highest levels, are: CompanysPrincipal, Director and MiddleLevelManager. Members of the executive board are principals, the only people who can authorize contracts. Certain activities are performed by those in lower ranks: here directors and middle level managers are legally bound. To show versatility and flexibility of the model, we admit more options for the crime scheme ("who could do what?"), one of which happened in the real case.

We can also define concepts representing activities described in the Penal Code. For example, PersonWhoFalsifiedDocument (3.2) is a (social) person connected to a company who can make decisions and is authorized to sign legally binding documents (that within a criminal activity may have a falsified content).

$$\begin{aligned} PersonWhoFalsifiedDocument \sqsubseteq \\ & PersonTakingLegalDecisions \sqcap \\ & \exists worksFor.Company \sqcap \\ & \forall worksFor.Company \sqcap \\ & \exists isAuthorizedToSign.LegalDocument \sqcap \\ & \exists signs.(\exists posseses.FalsifiedContent) \end{aligned} \tag{3.2}$$

The domain-based part of the ontology contains 92 concepts and 60 roles and 3 OWL datatype properties.

3.3.2 Task-based Part of the Ontology

This part of the ontology implements the following categories of queries, concerning:

- (legal) documents and their properties
- hierarchical chain of responsibility in a company
- executed transactions
- actions made by persons
- legal sanctions.

We have to account for the varying size of incriminated companies – the number of levels of responsibility ranges from one to three. Thus, the size of companies is measured by the number of levels of responsibility. For the Hydra case (Figure 3.1), A is a large company (3 levels of responsibility), B and D are medium size companies (2 levels) and C is a small company. The managers appearing in these companies are the CompanysPrincipal (CEO or Owner) – at the top level, a Director for construction (at the middle level) and The MiddleLevelManager responsible for a given construction (at the bottom level).

Rules related to consecutive concepts are numbered. These rules are related to the fraudulent disbursement and money laundering crime.

Several concepts and rules are defined to achieve ability to describe legal documents:

- ContractDocument - a document that is drawn up between two parties. This ContractDocument is between two companies, and is signed by principals of these companies. The signature on behalf of the company can be individual or joint, depending on the structure of the company. The following general rules for the ContractDocument are defined (6 rules); we present only 2 examples here, including rule on a contract between a large and a medium company (3.3), and a rule on a contract between two medium companies (3.4):

$$Document(?d),\ CompanysPrincipal(?p1),\ CompanysPrincipal(?p2),\ isSignedBy(?d,?p1),\ isSignedBy(?d,?p2),\ differentFrom(?p1,?p2) \rightarrow ContractDocument(?d) \tag{3.3}$$

$$Document(?d),\ MajorOwner(?p1),\ MajorOwner(?p2),\ isSignedBy(?d,?p1),\ isSignedBy(?d,?p2),\ differentFrom(?p1,?p2) \rightarrow ContractDocument(?d) \tag{3.4}$$

- InternalLegalDocument – a document drawn up in the company that may be authorized in stages up to the highest level of authority. It is signed hierarchically by the persons with different levels of responsibility.

- ComplexInternalLegalDocument - a virtual hierarchical document which could consist of several physical documents, that together authorize a payment (here ComplexInternalLegalDocument consists of a construction work acceptance document, and a payment authorization signature on the back of

an invoice). The series of authorizations reflects the structure of the company from the lowest to the highest rank of management. ComplexInternalLegalDocument is defined with the following rules: a rule on a complex internal legal document (3.5) and a rule on a complex internal legal document signed on back on invoice (3.6):

$$\begin{aligned} &ApprovalOfWorkDone(?d),\ Work(?w),\ Invoice(?i),\\ &concerns(?i,?w),\ concerns(?d,?w),\\ &isSignedBy(?i,?p2),\ isSignedBy(?i,?p1),\\ &worksFor(?p1,?c),\ worksFor(?p2,?c),\\ &hasLevelOfResponsibility(?p1,?l1),\\ &hasLevelOfResponsibility(?p2,?l2),\\ &lessThan(?l1,?l2),\ differentFrom(?d,?i)\\ &\rightarrow\\ &ComplexInternalLegalDocument(?i) \end{aligned} \tag{3.5}$$

$$\begin{aligned} &ApprovalOfWorkDone(?d),\ Work(?w),\ Invoice(?i),\\ &concerns(?i,?w),\ concerns(?d,?w), isSignedBy(?i,?p2),\\ &isSignedOnBackOfInvoiceBy(?i,?p2),\\ &worksFor(?p1,?c),\ worksFor(?p2,?c),\\ &hasLevelOfResponsibility(?p1,?l1),\\ &hasLevelOfResponsibility(?p2,?l2),\\ &lessThan(?l1,?l2),\ differentFrom(?d,?i)\\ &\rightarrow\\ &ComplexInternalLegalDocument(?i) \end{aligned} \tag{3.6}$$

- FalsifiedComplexInternalLegalDocument - ComplexInternalLegalDocument with approval of work which was not done. FalisfiedComplexInternalLegalDocument is calculated with the following rule:

$$\begin{aligned} &ComplexInternalLegalDocument(?d1),\\ &ApprovalOfWorkNotDone(?d2),\\ &Work(?w),\ concerns(?d1,?w),\ concerns(?d2,?w),\\ &differentFrom(?d1,?d2)\\ &\rightarrow\\ &FalsifiedComplexInternalLegalDocument(?d1) \end{aligned} \tag{3.7}$$

$$
\begin{aligned}
&ComplexInternalLegalDocument(?d1),\\
&ApprovalOfWorkNotDone(?d2),\\
&Work(?w),\ concerns(?d1,?w),\ concerns(?d2,?w),\\
&isSignedBy(?d1,?p1),\ isSignedBy(?d2,?p2),\\
&differentFrom(?d1,?d2),\ differentFrom(?p1,?p2)\\
&\rightarrow\\
&FalsifiedComplexInternalLegalDocument(?d1),\\
&isSignedBy(?d1,?p2)
\end{aligned}
\tag{3.8}
$$

The rule (3.7) defining the falsified complex document consisting of work approving document and accepted invoice. This two documents authorize the payment. The rule (3.8) refers to the previous one but specifies who signed the two constituent documents.

- Transaction - consists of a contract between two companies, the work, an invoice issued for work and payment. It is defined with the following rule on a transaction between two companies:

$$
\begin{aligned}
&ComplexInternalLegalDocument(?i),\ ContractDocument(?d),\\
&Invoice(?i),\ MoneyTransfer(?mt),\\
&Work(?w),\ Company(?c1),\ Company(?c2),\\
&concerns(?d,?w),\ concerns(?i,?w),\\
&flowsFrom(?mt,?c2),\ flowsTo(?mt,?c1),\\
&isIssuedBy(?i,?c1),\ isReceivedBy(?i,?c2)\\
&\rightarrow\\
&Transaction(?d),\ hasInvoice(?d,?i),\ hasMoneyTransfer(?d,?mt),\\
&transactionFrom(?d,?c2),\ transactionTo(?d,?c1)
\end{aligned}
\tag{3.9}
$$

If a contract, work or invoice document turns out to be a FalsifiedDocument, then the Transaction will be classified as a FalsifiedTransaction.

- Formal hierarchy - the management structure in a company. Cardinality of managers at each level is 1. In future, we will allow some decisions to be taken as a group, if roles of managers in the group at the same level were the same.

Definitions of Logical Activities Appearing in the Polish Penal Code

Using rules we can also query the knowledge base about legal sanctions reached by the public prosecutor and the judge in accordance with the activities of key persons

involved in the Hydra case. At first we characterize these activities. We take into consideration activities of three types of social agents, namely CompanysPrincipal, Director and MiddleLevelManager (see Figure 3.3). The MiddleLevelManager and the Director (this is what happened in reality, a version in which the Director approved the money transfer) committed intellectual falsification. This is established beyond doubt, so they are fraudsters unconditionally. The CompanysPrincipal acts as unconditional fraudster only when he signed the falsified document. There are three other cases of CompanysPrincipal's activity (see the sequences listed below). In the first one he might have intent to commit the crime for he should have known that the work has not been done. In the second case, he might not have known that the work has not been done, so he was probably negligent. The last case deals with CompanysPrincipal's conditional involvement in the crime, because additional information is needed to prove that he is the part of it. This decision is left to an investigator or a prosecutor and may be established in several ways:

1. Through a guilty plea during testimonies or in a court.

2. With the help of other members of the scheme, testifying that he was part of it.

3. By observing money transfers to his account which cannot be accounted for.

In principle, we could try to design rules for these concepts. Here, these properties are determined by a human (an investigator or a prosecutor). Even if he was guilty, he could claim being under duress while giving the testimony (a victim of coercion by prosecutors), or could claim innocence due to mental incompetence at the time. Summing up the CompanysPrincipal's involvement in the crime, the following theoretical possibilities exist. The CompanysPrincipal could be a part of the scheme (or even the organizer of the scheme), or would have approved a payment without knowing that the work had not been done. Suppose the principal claims he is innocent. If he was not implicated by the MiddleLevelManager and/or the Director of company A, nor by the CompanysPrincipal of B, nor by money coming to his account, then the CompanysPrincipal was not a part of a scheme. The full model should decide whether the CompanysPrincipal was negligent (leading to nonfeasance), since the CompanysPrincipal was obliged to verify work acceptance documents (he may be charged on the basis of Art. 296 §4 of Penal Code).

All things considered, the activities of the three types of social persons (agents) form the four following sequences. These agents correspond to the "real world persons" who are employed in certain positions. The sequences are presented in Table 3.3.

Presented sequences express persons' actions that may occur in a crime. The sequences are alternative crime schemes. Roles of agents (their positions at work)

Table 3.3: Options in the fraudulent disbursement case of Hydra: detailed activities of key persons of company A having legal meaning.

Number		**Sequence of activities 1**	**Sequence of activities 2**	**Sequence of activities 3**	**Sequence of activities 4**
Activities		Principal orders payment	Director orders payment	Director orders payment	Director orders payment
Manager's sequence of activities	Principal	Principal accepts the document and orders payment	Should have known that the work has not been done, if he was not negligent	Might not have known that the work has not been done	Part of the crime scheme
	Director	Cosigns falsified construction work acceptance document	Director accepts the document and orders the payment	Director accepts the document and orders the payment	Director accepts the document and orders the payment
	Middle Level Manager	Falsified construction work acceptance document	Falsified construction work acceptance	Falsified construction work acceptance	Falsified construction work acceptance

are used in rules that define sanctions (since the level of responsibility affects the levied penalty).

The success of our model relies on clear separation of the nature of the facts we use:

1. Concepts and facts in the MinOn model.
2. Concepts and facts in the external sources. This, for example, includes knowledge of how a standard company operates. An invoice accepted by a Principal or Director goes through an accountant and an administrative officer who actually executes the money transfer. We do not attempt to describe these activities, unless there is a crime at these stages.
3. Certain facts are left for manual input by an investigator and a prosecutor.

In MinOn we explicitly use conditions 1 and 2 from Table 3.2 in the case of the Director and the MiddleLevelManager. Element 3 is used to judge the involvement of the Principal. Elements 3 and 4 are combined, as knowing about a fraud also indicates intent. Element 5 is not analysed, because it results from standard procedures in a company: once a payment was authorized, it is executed.

The decisive document falsifier (for example, an agent issuing an invoice for work not done) having the intent to do fraudulent disbursement to a company by illegally transferring money from it, and who gains some amount of this illicit money, may be defined by the following rule:

$$\begin{aligned}
&PersonWhoFalsifiedDocument(?x),\ Company(?c),\\
&initiates(?x,?mt),\ MoneyTransfer(?mt),\\
&achieves(?x,?g),\ IsPartOfTheScheme(?x,?k)\ \ //k\ -\ given\ case\\
&IllicitPersonalGain(?g),\ hasValue(?g,?v2),\ Value(?v2),\\
&hasValue(?mt,?v1),\ Value(?v2),\ ?v2 \leq ?v1\\
&hasGoal(?x,?go),\ hasIntentToCommitCrime(?x,?go)\\
&\rightarrow\\
&FraudsterInACompany(?x,?c)
\end{aligned} \tag{3.10}$$

The rules for crime activities from Table 3.3 were defined to obtain results from the current state of the knowledge base. Rules enable users to determine which sequence of activities is appropriate to a crime schema. The following rules were defined for sequence of activities 1:

$$\begin{aligned}
&FalsifiedComplexInternalLegalDocument(?d),\ isSignedBy(?d,?p1),\\
&isSignedOnBackOfInvoiceBy(?d,?p2),\ MiddleLevelManager(?p1),\\
&Director(?p2),\ Principal(?p3),\ orders(?p3,?m),\ Payment(?m),\\
&accepts(?p3,?d),\ knowsAbout(?p3,?d)\\
&\rightarrow\\
&FraudulentDisbursementCrime(?p1,1),\\
&FraudulentDisbursementCrime(?p2,1),\\
&FraudulentDisbursementCrime(?p3,1)
\end{aligned} \tag{3.11}$$

$$\begin{aligned}
&FraudulentDisbursementCrime(?p1,1),\\
&FraudulentDisbursementCrime(?p2,1),\\
&FraudulentDisbursementCrime(?p3,1),\\
&MiddleLevelManager(?p1),\ Director(?p2),\ Principal(?p3),\\
&worksFor(?p1,?f1),\ worksFor(?p2,?f1),\ worksFor(?p3,?f1)
\end{aligned}$$

$$\rightarrow$$
$$inComplicityWith(?p1, ?p2),$$
$$inComplicityWith(?p2, ?p3),$$
$$inComplicityWith(?p1, ?p3) \quad (3.12)$$

Next set of rules constitutes the knowledge of particular criminal activities. Contrary to many works in legal ontologies, we do not introduce plans and intentions because these are extremely difficult to describe. Some of the constructed rules are as follows:

- The rule defining complicity of persons working on behalf of the same company; one person - a construction manager - falsifies ApprovalOfWorkDone document, and the second one approves the payment of the Invoice by signing the back of this document.

$$Company(?c),\ NoWork(?w),\ ContractDocument(?d1),$$
$$ComplexInternalLegalDocument(?d2),\ Person(?p1),\ Person(?p2),$$
$$concerns(?d1, ?w),\ concerns(?d2, ?w),$$
$$worksFor(?p1, ?c),\ worksFor(?p2, ?c),$$
$$knowsAbout(?p1, ?w),\ knowsAbout(?p2, ?w),$$
$$isSignedBy(?d1, ?p1),\ isSignedOnBackOfInvoiceBy(?d2, ?p2),$$
$$differentFrom(?p1, ?p2),\ differentFrom(?d1, ?d2)$$
$$\rightarrow$$
$$inComplicityWith(?p1, ?p2) \quad (3.13)$$

- The rule defining complicity of persons working on behalf of different companies executing a fraudulent transaction.

$$Company(?c1),\ Company(?c2),\ NoWork(?w),\ Transaction(?t),$$
$$worksFor(?p1, ?c1),\ worksFor(?p2, ?c2),$$
$$knowsAbout(?p1, ?w),\ knowsAbout(?p2, ?w),$$
$$transactionFrom(?t, ?c1),\ transactionTo(?t, ?c2),$$
$$differentFrom(?p1, ?p2),\ differentFrom(?c1, ?c2)$$
$$\rightarrow$$
$$inComplicityWith(?p1, ?p2) \quad (3.14)$$

- The rule defining the MoneyLaundering act committed by the first company in the money laundering chain (if A paid B, this refers to company B; the company A is not indicted). Here A is Hydra and B is Hermes.

$$NoWork(?w),\ Invoice(?i),\ Transaction(?t),$$

$$Company(?c1),\ Company(?c2),\ MoneyTransfer(?mt),$$
$$hasInvoice(?t, ?i),\ concerns(?i, ?w),\ hasMoneyTransfer(?t, ?mt),$$
$$flowsFrom(?mt, ?c1),\ flowsTo(?mt, ?c2),\ differentFrom(?c1, ?c2)$$
$$\rightarrow$$
$$MoneyLaundering(?c2), relatedTo(?c2, ?t) \tag{3.15}$$

- The rule defining the money laundering act committed by next companies in the money laundering chain (e.g. companies C, and D, that is Dex and Mobex).

$$NoWork(?w),\ Invoice(?i),\ Transaction(?t),$$
$$Company(?c1),\ Company(?c2),\ MoneyTransfer(?mt),$$
$$MoneyLaundering(?c1),\ hasInvoice(?t, ?i),\ concerns(?i, ?w),$$
$$hasMoneyTransfer(?t, ?mt),\ flowsFrom(?mt, ?c1),$$
$$flowsTo(?mt1, ?c2),\ differentFrom(?c1, ?c2)$$
$$\rightarrow$$
$$MoneyLaundering(?c2),\ relatedTo(?c2, ?t) \tag{3.16}$$

- The rule defining the sanction PC art. 299 §1 related to money transfer for work not done (pertains to managers of company B and C).

$$Art\_299\_1(?a),\ NoWork(?w),\ MoneyLaundering(?m),$$
$$Company(?m),\ ApprovalOfWorkNotDone(?d),\ Transaction(?t),$$
$$Person(?p),\ relatedTo(?m, ?t),\ worksFor(?p, ?m),$$
$$knowsAbout(?p, ?w),\ concerns(?t, ?w),$$
$$concerns(?d, ?w),\ isSignedBy(?d, ?p)$$
$$\rightarrow$$
$$fallsUnder(?p, ?a) \tag{3.17}$$

- The rule defining the sanction PC art. 299 §1 when the ApprovalOfWork-Done document does not exist. It happens down the chain that companies do not bother even create documents. In this case there were no documents for fictitious work approval between C and D (Dex and Mobex).

$$Art\_299\_1(?a),\ NoWork(?w),\ MoneyLaundering(?m),$$
$$Company(?m),\ FalsifiedComplexInternalLegalDocument(?d),$$
$$Transaction(?t),\ Person(?p),\ relatedTo(?m, ?t),$$
$$worksFor(?p, ?m),\ concerns(?d, ?w),\ concerns(?t, ?w),$$

$$knowsAbout(?p, ?w),\ isSignedBy(?d, ?p)$$
$$\rightarrow$$
$$fallsUnder(?p, ?a) \tag{3.18}$$

- The rule defining the sanction PC art. 299 § 1 when a company accepts laundered money (here Mobex).

$$Art\_299\_1(?a),\ NoWork(?w),\ MoneyLaundering(?m),$$
$$Company(?m),\ Transaction(?t),\ Person(?p),\ relatedTo(?m, ?t),$$
$$transactionTo(?t, ?m),\ worksFor(?p, ?m),\ concerns(?t, ?w),$$
$$knowsAbout(?p, ?w)$$
$$\rightarrow$$
$$fallsUnder(?p, ?a) \tag{3.19}$$

- The rule defining the sanction PC art. 299 § 5 (since he/she was aware what was the purpose of the scheme and collaborated with others involved - we know this from testimonies and signing relevant documents). This rule is related to persons in the same company.

$$Art\_299\_5(?a1),\ Art\_299\_1(?a2),\ NoWork(?w),$$
$$FalsifiedComplexInternalLegalDocument(?d),$$
$$Person(?p1),\ Person(?p2),$$
$$fallsUnder(?p1, ?a2),\ fallsUnder(?p2, ?a2),$$
$$knowsAbout(?p1, ?d),\ knowsAbout(?p2, ?d),$$
$$knowsAbout(?p1, ?w),\ knowsAbout(?p2, ?w),$$
$$inComplicityWith(?p1, ?p2),\ differentFrom(?p1, ?p2)$$
$$\rightarrow$$
$$fallsUnder(?p1, ?a1),\ fallsUnder(?p2, ?a1) \tag{3.20}$$

- As Rule 8 but related to persons in different companies.

$$Art\_299\_5(?a1),\ Art\_299\_1(?a2),\ NoWork(?w),$$
$$Company(?c1),\ Company(?c2),\ ContractDocument(?d),$$
$$Person(?p1),\ Person(?p2),$$
$$fallsUnder(?p1, ?a2),\ fallsUnder(?p2, ?a2),$$
$$knowsAbout(?p1, ?d),\ knowsAbout(?p2, ?d),$$
$$knowsAbout(?p1, ?w),\ knowsAbout(?p2, ?w),$$
$$inComplicityWith(?p1, ?p2),$$

$$worksFor(?p1, ?c1),\ worksFor(?p2, ?c2),$$
$$differentFrom(?p1, ?p2),\ differentFrom(?c1, ?c2)$$
$$\rightarrow$$
$$fallsUnder(?p1, ?a1), fallsUnder(?p2, ?a1) \tag{3.21}$$

- The rule defining the sanction PC art. 299 §5 based on ApprovalOfWorkNotDone for workers in 2 different companies (who did not signed a contract document, as in rules 8 and 9)

$$Art\_299\_5(?a1),\ Art\_299\_1(?a2),\ NoWork(?w),$$
$$Company(?c1),\ Company(?c2),\ Person(?p1),\ Person(?p2),$$
$$fallsUnder(?p1, ?a2),\ fallsUnder(?p2, ?a2),$$
$$knowsAbout(?p1, ?w),\ knowsAbout(?p2, ?w),$$
$$inComplicityWith(?p1, ?p2),$$
$$worksFor(?p1, ?c1), worksFor(?p2, ?c2),$$
$$differentFrom(?p1, ?p2), differentFrom(?c1, ?c2)$$
$$\rightarrow$$
$$fallsUnder(?p1, ?a1),\ fallsUnder(?p2, ?a1) \tag{3.22}$$

Mapping of Logical Activities to Legal Sanctions for Crime Perpetrators

Legal sanctions were reached by an expert prosecutor and the judge assessing previously described activities. At first we present some explanatory notes concerning fraudulent disbursement and money laundering sanctions in the Polish Penal Code [Sejm 1997]. From analysis of real cases of the considered crime it follows that most defendants were accused as follows:

1. Art. 296 §1-3 PC – strictly: negligence leading to damage to a company (for personal benefit). The asset misappropriations including fraudulent disbursement (FD) are prosecuted in Poland based on this article. The phrasing of the crime in the Polish Penal Code does not exactly agree with its meaning.
2. Art. 296 §4 PC – unknowing negligence leading to damage to a company.
3. Art. 284 §2 PC – personal benefit resulting from activities sanctioned under art. 296.
4. Art. 294 §1 PC – the offense specified in 284 §2 PC with regard to property of considerable value.
5. Art. 286 §1 PC – fraud (intentionally deceiving a person (here, a legal person - a company), which results in a damage to the company.

6. Art. 271 §3 PC – lying or issuance (signing) a false document with regard to a circumstance having legal significance.
7. Art. 273 §1 PC using a document mentioned in art. 271 §3 PC.
8. Art. 299 §1 and 5 PC – money laundering (conscious and together with other persons, constituting a crime group).
9. Art. 18 §1 PC – directing illegal activity performed by another person.

Using the defined and derived properties, the logical characterization of a suspect's activities are illustrated in Table 3.4.

Legal sanctions reached by an expert prosecutor and the judge can be defined as rules. Here, we present rules for every variant from Table 3.4:

1. Variant 1:

$$
\begin{aligned}
&FalsifiedComplexInternalLegalDocument(?d),\\
&MiddleLevelManager(?p1),\ Director(?p2),\\
&inComplicityWith(?p1,?p2),\ Principal(?p3),\\
&NotInComplicity(?p3),\\
&Art\_286\_1(?a1),\ Art\_294\_1(?a2),\ Art\_284\_2(?a3),\\
&Art\_273(?a4),\ Art\_271(?a5)\\
&\rightarrow\\
&fallsUnder(?p1,?a2),\ fallsUnder(?p1,?a5),\ fallsUnder(?p2,?a1),\\
&fallsUnder(?p2,?a2),\ fallsUnder(?p2,?a3),\ fallsUnder(?p2,?a4),\\
&Innocent(?p3)
\end{aligned}
$$

2. Variant 2:

$$
\begin{aligned}
&FalsifiedComplexInternalLegalDocument(?d),\\
&inComplicityWith(?p1,?p2),\ MiddleLevelManager(?p1),\\
&Director(?p2),\ Principal(?p3),\ notInComplicity(?p3),\\
&Negligent(?p3),\\
&Art\_286\_1(?a1),\ Art\_294\_1(?a2),\ Art\_284\_2(?a3),\\
&Art\_273(?a4),\ Art\_271(?a5),\ Art\_296\_4(?a6)\\
&\rightarrow\\
&fallsUnder(?p1,?a2),\ fallsUnder(?p1,?a5),\\
&fallsUnder(?p2,?a1),\ fallsUnder(?p2,?a2),\\
&fallsUnder(?p2,?a3),\ fallsUnder(?p2,?a4),\ fallsUnder(?p3,?a6)
\end{aligned}
$$

Table 3.4: Logical characterization of activities of key persons of company A in variants of a fraudulent misappropriation scheme. Legal sanctions are those reached by an expert prosecutor and the judge.

Number		**Variant 1**	**Variant 2**	**Variant 3**
Activities		Middle Level Manager and Director part of complicity. Principal not part of the scheme.	Middle Level Manager and Director part of complicity. Principal not part of the scheme but negligent.	Middle Level Manager and Director part of complicity. Principal part of the scheme, actually the organizer.
Legal sanctions	Principal	Innocent	art. 296 §4 PC	art. 296 §1-3 PC (depending on the cost of the damage) and art. 284 §2 PC and art. 294 §1 PC
	Director	art. 286 §1 PC i art. 294 §1 PC, art. 284 §2 PC, art. 273 §1 PC;	art. 286 §1 PC i art. 294 §1 PC, art. 284 §2 PC, art. 273 §1 PC;	art. 296 §1-3 PC (depending on the cost of the damage) and art. 284 §2 PC and art. 294 §1 PC art. 273 §1 PC
	Middle Level Manager	art. 294 §1 PC; art. 271 §3 PC	art. 294 §; 1 PC art. 271 §3 PC	art. 296 §1,2, 3 PC and art. 284 §2 PC; and art. 294 §1 PC; art. 271 §3 PC

3. Variant 3:

$$FalsifiedComplexInternalLegalDocument(?d),$$
$$MiddleLevelManager(?p1),\ Director(?p2),\ Principal(?p3),$$
$$inComplicityWith(?p1, ?p2),\ inComplicityWith(?p2, ?p3),$$
$$inComplicityWith(?p1, ?p3),\ Organizer(?p3),$$
$$Art\_296(?a1),\ Art\_294\_1(?a2)\ Art\_284\_2(?a3),$$

$$Art\_273(?a4),\ Art\_271(?a5)$$
$$\rightarrow$$
$$fallsUnder(?p3, ?a1),\ fallsUnder(?p3, ?a2),$$
$$fallsUnder(?p3, ?a3),\ fallsUnder(?p2, ?a1),\ fallsUnder(?p2, ?a2),$$
$$fallsUnder(?p2, ?a3),\ fallsUnder(?p2, ?a4),\ fallsUnder(?p1, ?a1),$$
$$fallsUnder(?p1, ?a2),\ fallsUnder(?p1, ?a3),\ fallsUnder(?p1, ?a5)$$

3.4 Discussion of the Related Work

It is fair to say that Hydra case was quite narrow. For comprehensive domains Breuker stated "Not one of these ontologies is used for reasoning The reasoner is only used for consistency checking." [Breuker 2009]

The presented analysis shows that for the narrow categories of economic crimes our method is highly effective and reasons out the correct legal sanctions. It is because the considered cases are based on hard facts, so the relevant knowledge can be expressed in logic. In general, legal matter is less precise, much more open for interpretation that sometimes boarders on arbitrariness. For example, European VAT regulations are such that the border between criminal and legal behaviour is very thin, and thus a careful VAT avoidance offender is practically immune from punishment.

Our model contains several restrictions:

- Some facts are assumed given: knowsAbout, assignment for Organizer. Some concepts such as intentions or that someone should have known that he was participating in a transaction linked to a criminal act (e.g. VAT evasion) are much more difficult to define. This would require much more expressive power than used in presented models.

- At a given level of detail when presenting facts exceptions appear. In the presented crime types these are rare, e.g. a person who signed a false document could have been very sick or blackmailed. In general one should use defeasible logic [Governatori 2012]. It has not been our goal to completely displace a human. The system's result is to be verified by a lawyer.

- Some researchers question a straightforward use of a logic model for legal reasoning. Instead, the basis for reaching mutually acceptable conclusions, such as a verdict in a court of justice is argumentation that includes debate and negotiation. Recently, however, theories have been proposed that are an extensions of Transaction Logic and provide unifying framework for defeasible reasoning called Logic Programs with Defaults and Argumentation Theories [Fodor 2011].

For at least 15 years, ontology-based artificial intelligence and knowledge engineering techniques have been applied to the legal domain, [Breuker 2009, Casellas 2008]. The intense effort went in several directions, among others:

1. Legal information extraction and legal document management [Biasiotti 2008].

2. Formalization of the theory of law and the design of legal core, topical and domain ontologies, the widely known of which are:

   - CLO, Core Legal Ontology[2], which is anchored in the foundational ontology of DOLCE+ (DOLCE extended by the ontology of Descriptions and Situations[3]). It contains the core concepts of the legal domain that were established coming down from the foundational level.
   - LKIF-core, Legal Knowledge Interchange Format [Hoekstra 2007] that also contains core legal concepts which were separated in the middle-out manner, meaning that law "users" were asked to define of about 20 most meaningful legal concepts, forming, after some selection and extension, a base for the ontology. Then, this basic layer may be both specialized and generalized to obtain new terms.
   - FOLaw, the functional ontology of law [Valente 1995] is in fact a certain formalization of law, in which the whole legal knowledge is classified by the author into six classes: normative-, world-, responsibility-, reaction-, creative- and meta- knowledge.
   - Frame-based ontology of law of Visser, van Kralingen and Bench-Capon [Visser 1997] that also gives a formalization of the legal knowledge. It has two layers: a generic one defining the terms of a norm, an action and a legal concept, and a specific layer concerning a certain statute (regarding a concrete domain of interest).
   - Knowledge-based ontology of Mommers [Mommers 2002] who continues the work of [Visser 1997] and extends it to encompass different legal and philosophical attitudes regarding the nature of law, the possibility of representing legal epistemology and differentiating between a knowledge and beliefs.
   - Topical ontology of financial fraud [Zhao 2005] and VAT [Kerremans 2005], which are examples of a thematic ontology. They are anchored in a foundational level of SUMO, Suggested

[2] http://www.loa-cnr.it/ontologies/CLO/CoreLegal.owl
[3] http://www.loa.istc.cnr.it/ontologies/DLP_397.owl

Upper Merged Ontology[4] and its two specializations, namely financial ontology and the ontology of services.

- OPJK, Ontology of Professional Judicial Knowledge [Casellas 2008] is a legal ontology that is to support the mapping of questions posed by junior judges to a set of stored frequently asked questions. The work on the IURISERVICE [Casellas 2008, Casanovas 2009] application that uses OPJK started in 2001. A version 2 had been in a pilot mode since 2005. Despite its effectiveness the system was deployed in Spain only in 2010, mainly due conservative attitude and partly resistance of Spanish judges circles.

3. Using formal models of legal knowledge to support legal reasoning. In the models with narrowly defined entities, one can argue whether child's bicycle is a vehicle or not, if a camper is a house or a vehicle, and what is the legal implication of these facts. For broader range of aspects notable was an introduction of intermediate concepts that allow differentiation between cases [Aleven 2003, Wyner 2008]. In Wyner's work logical relations were designed to achieve a decision that a trade secret has been misappropriated.

As regards to financial fraud ontology, the one developed within the FF Poirot project [Zhao 2005] is very broad, and consequently difficult to handle. Some concepts used there may help to gain evidence, but in an indirect way (perpetrator's personality or a trust is a moral rather than a legal concept). There is already a wealth of lessons coming from the above-mentioned advances. Most researchers think that upper level (meaning core or foundational) ontologies alone cannot serve as a semantic base of practical information systems. The reasoner can only be used for checking the consistency of these ontologies but the ontologies themselves cannot be used for reasoning about gathered data. On the other hand topical or domain ontologies alone are too restricted and can be used with success for closed technical realms [Darlington 2008]. Doing systematic specializations of upper level ontologies with domain concepts, individuals, relations, attributes and rules is difficult. In these limited scope cases when this has been done, such as testing a Dutch traffic code on consistency and completeness [Breuker 2009], the results are superior to these that can be achieved by a human.

It appears that one of the most critical issues is the size of an ontology required to model a given domain. In [Corcho 2003] the attempt was made to build an ontology of French law. From an initial list of 118000 legal terms, a list of 16681 fundamental terms was finally considered (unfortunately it is not known how many of them were related to the legal financial area). Recalling Breuker [Breuker 2009], it is crucial to determine how such terms affect understanding that is able to make

[4] http://www.ontologyportal.org/

up a coherent "macro-structure" – a model . Suppose, we think of a possible legal connotations related to a term "invoice". Certainly, we can issue, receive, accept, sign, co-sign, forge, falsify, or initiate a payment related to an invoice. These contexts would appear in every financial domain. In some cases invoice can be destroyed to prevent the investigation of accounting practices in a given company. Consideration of losing an unsigned paper invoice is not necessary – one can always issue a paper duplicate or print it. Completely irrelevant for the legal domain is considering such events as spilling coffee on an invoice or making a paper plane out of an invoice in its physical form. Eating an invoice (as a means of destroying it) may have medical rather than legal aspect, since it is not important from a legal point of view how it was destroyed. It is now easier to understand while, in spite of having relatively large ontologies, FF Poirot project was only able to describe the Nigerian letter fraud and fraudulent Internet investment pages. The used ontologies had too many concepts one could not reason with and at the same time they were lacking important concepts (we cannot show it exactly since FF Poirot ontologies are not publicly available).
There exist first attempts to quantify the size of ontologies, e.g. [Zhang 2010]. These, done together with ontology merging, aligning and reuse are the important steps in the ontology-based system engineering. In the presented context our work should be seen as an effort to create an ontology that serves as a conceptual model of the fraudulent disbursement (and crimes linked to it), which, apart from supporting the earlier mentioned tasks, can be used to deduce the sanctions that might be levied against people engaged in this crime. It is done via reasoning with rules constituting the task-based part of the ontology. To our knowledge, the work on mapping of crime activities into criminal law articles has been done only for cyber crimes [Bezzazi 2007], which have a much narrower scope. Our method enables to build models for different economic crimes. The method relies on having a general enough ontology contents design pattern (the ontology of criminal processes and investigating procedures [Cybulka 2009]) that can be specialized by concepts of a given type of an economic crime. Our general method is also a specialization of a pattern, which is a foundational ontology of *constructive descriptions and situations* [Gangemi 2008]. It is worth noticing that this design pattern is expressive enough to incorporate other existent ontologies (for example any of the listed above core legal ontologies) with a rather minor effort.

3.5 Conclusion

In this chapter we presented the ontological model of the fraudulent disbursement and money laundering crime expressed using OWL classes and properties, and a reasonable number of rules (task-based application level of the ontology).

Our analysis accounts not only for crimes of people associated with Hydra – company A. We also give sanctions for people in companies B, C, and D – sanctions for money laundering. The model is not yet able to determine the duration of an appropriate penalty.

Because fraudsters may use many types of schemes, techniques and transactions to achieve their goals, we need a conceptual model of economic crime with significant generality. In the future, we intend to demonstrate that we can describe not just one case but a broad class of economic crimes, such as:

- CausingAssetMisappropriation
- CausingDamageToACompany.

We do not consider this task impossible, although we will always face additional factors necessary to extend our model. We could, for example ask: was only the CEO of the Management Board in Hydra implicated? At the time of the investigation, there was no proof otherwise. But apparently the rest of the Board knew about the scheme, because several years later they were indicted on a similar count.

We do not dwell on who exactly had the power to sign (in some cases there are disputes on the validity of supervisory board decisions); this fact must be established by a prosecutor. An extent to which the model can be generalized to treat such more complex schemes will be the subject of a further study. However, after performing quite a number of reasoning experiments on the 5 most common economic crime mechanisms in Poland (some results are presented in [Bak 2009], [Bak 2011a]), we are convinced that a general model can be constructed that handles a few most common economic crimes with 85% use of all pertinent facts (the Hydra case is somewhat easier than average). In this regard, application of intermediate levels of factors [Wyner 2008] could be helpful.

To our knowledge, the work on mapping of crime activities into criminal law articles has been done only for cyber crimes [Bezzazi 2007], which have a much narrower scope, although using result of work [Wyner 2008] it could be straightforward for the case. In work [Wyner 2008] only OWL ontology was used for TBox reasoning (although rules were discussed in a different aspect), whereas our approach uses ontology and rules.

The MinOn ontology is relatively small, but will be enhanced once we add other fraud type typologies (for example, the fuel crime [Cybulka 2010a] or the VAT carousel fraud [Jedrzejek 2011a]). However, the system is able to handle all possible variants of the Hydra crime depending on who in hierarchy of management took decisions leading to unlawful disbursements and knew about a scheme. In short for this particular crime the system has a sufficient expressive power to reason not only on proof of physical elements of offences but also on some circumstances of crime elements (intention, knowledge, or negligence). In fact, the system did a better job than a prosecutor in case of Hydra: the prosecutor chose to use a sanction

under 271 §3 PC instead of Art. 273 §1 PC for the Director of Hydra. In similar cases, prosecutors often had not assigned Art. 286 §1 PC – fraud, when an offender intentionally deceived not a physical person up in management but a legal person – a company, which resulted in a damage to the company.

At present, we do not use defeasible logic for rare exceptions. For example, lying or issuance (signing) a false document could occur when a person felt so badly that could not have understood his/her actions or had been blackmailed or threatened with a loss of life. Nevertheless, an inspection of around 10 real cases of the fraudulent disbursement type indicates high level of applicability of our description.

There are areas of detailed sanction determination where the current expressive power is not sufficient. If we have to consider details of intentions such as these appearing in 18 definitions of money laundering in various legal systems collected in [Unger 2007] the problem will became hard (as referred in [Breuker 2009]). Suppose we would like to distinguish between activities whose purpose is "hide the proceeds" or "make it appear legal". We cannot at present define these notions, in general. They would depend on context, which would make us go into deep detail and would require rules of such complexity that they would be impossible to handle (in an efficient way). Another important and difficult area is whether a sanction is to be determined for an act under Polish penal code (PC) and Polish tax penal code (TPC), much more lenient than PC. On one hand, an offender should be sentenced on all counts. On the other hand, there exists an interpretation that TPC is more detailed (*lex specialis derogat legi generali*) and should have preference over PC. Currently, Polish courts give sentences using either of these interpretations for seemingly very similar cases. This indicates that certain areas of the legal doctrine are very difficult to interpret not only by a machine but also for expert humans.

It is true that we selected the most favourable crime to be analysed with our model. Even in Hydra case we would face difficulty, whether in this case a criminal group is an organized criminal group. There is no definition of “an organized criminal group” in the Polish PC. Therefore, inferring the legal qualifications for this case (that is whether Article 258 §1 applies) is subject to interpretation that has to be provided by legal communities to design appropriate rules. In any case the results of our model have to be verified by leading legal experts.

We are convinced that the increased cooperation between legal community and knowledge engineers would possibly be of great use for society.

CHAPTER 4

Methods for a rule-based query answering

This chapter presents our approaches concerning a rule-based query answering (RQA) and the combination of an ontology with a relational database. We propose two methods of reasoning applied in RQA: *hybrid reasoning* and *extended rules reasoning*.

First, we introduce the number of assumptions that we have already made. In a rule-based query answering method we assume that there exists a knowledge base which contains two parts: intensional and extensional. The intensional knowledge is represented as a set of rules and describes the source data at a conceptual (ontological) level. The extensional knowledge consists of facts that are stored in the relational database as well as facts that were derived in the reasoning process. Queries can be posed in the terms of the conceptual level. Thus, one gets an easier way to create a query than using structural constructions from SQL (Structured Query Language). The rule-based query answering method uses the reasoning process to obtain an answer for a given query. During this process facts from a database are gathered and used to derive new facts according to a given set of rules. Next, the answer is constructed and presented. The other assumptions are the following:

- We express a conceptual knowledge with a Horn-$\mathcal{SHIQ}$ ontology combined with SWRL (Horn-like) [Horrocks 2004b] rules. Horn-$\mathcal{SHIQ}$ is a tractable fragment of OWL 1.1 [Grau 2006] which can be expressed as Horn clauses. Only unary or binary predicates are permissible, according to the terms that appear in OWL (since we use this standard as a way to express a conceptual knowledge).
- We transform a knowledge base, defined as a Horn-$\mathcal{SHIQ}$ ontology combined with SWRL rules, into a set of Horn clauses. As a result, the ontology-based knowledge is represented as a set of rules. We apply the Horn subset of the SWRL language which is a decidable fragment of SWRL. We have developed two methods: simple transformation and Horn-$\mathcal{SHIQ}$ transformation. These methods are described in Chapter 5.
- We assume conjunctive queries (CQ) only, which are built of predicates from an ontology (concepts and roles).

- We use the Datalog safety restriction and DL-safe rules to ensure the decidability of the reasoning process and thus the rule-based query answering.
- We apply rules that are Horn clauses [Lloyd 1984]. If there is conjunction of several predicates in the head, the rule can be easily transformed into Horn clauses with the Lloyd-Topor transformation [Lloyd 1984].
- We follow the closed-world semantics since this assumption occurs in both relational databases and Datalog. It means that facts that cannot be proven are considered false. It results from the fact that we are focused only on hard evidences in our knowledge base of economic crimes.
- We focus on forward chaining, thus we decided to use the state-of-the-art reasoning algorithm - Rete [Forgy 1982] which is implemented in the most popular and commercial forward reasoning engines, like Clips[1] Jess[2], Drools[3], OPSJ [4], IBM iLog[5] and others.
- We employ the *Jess* (Java Expert System Shell) engine [Hill 2003], since it implements Rete and it is one of the fastest commercial engines (with the free academic use). Jess can be easily integrated with the Java language (which is the implementation language of the SDL library which implements our approaches). The Jess engine supports both forward and backward chaining.
- We represent facts as RDF triples since OWL ontologies can be represented in RDF/XML syntax [Beckett 2004, Horridge 2006]. Each triple is of the following form:

$$(triple\ \ (p\ \ somePredicate)\ (s\ \ someSubject)\ (o\ \ someObject))$$

where p is a predicate name, s is a subject and o is an object, e.g., $(triple\ \ (p\ \ \ hasFather)\ \ (s\ \ \ Mike)\ \ (o\ \ \ Chris))$ which means that $Chris$ is a father of $Mike$. In our examples we use the following form of a triple: $p\ (s,\ o)$, e.g., $hasFather(Mike,\ Chris)$.

Next two sections describe our RQA methods. Section 4.4 presents the mapping between ontology predicates and a relational data. Section 4.5 presents the related work. In Section 4.6 we provide possible applications of proposed approaches and conclusions.

[1] http://clipsrules.sourceforge.net/
[2] http://www.jessrules.com/
[3] http://www.jboss.org/drools/
[4] http://www.pst.com/opsj.htm
[5] http://www-01.ibm.com/software/websphere/ilog/

4.1 Hybrid Reasoning Method

This section presents our hybrid reasoning method applied in a rule-based query answering. The method is Jess-dependent since we employ backward and forward reasoning implemented in the Jess engine. Jess-dependent means that we exploit the backward chaining implemented in Jess which is simulated by the forward chaining and requires Jess programs to be defined in a particular form.

The backward chaining method in Jess requires a special declaration for templates *(do-backward-chaining)*. The *do-backward-chaining* definitions are added to all *deftemplates* declarations which are used in backward chaining (for example: *(do-backward-chaining triple)*). One can define rules to match backward reactive templates. The rule compiler rewrites such rules and adds the *need-* prefix to inform the Jess engine when this rule has to be fired (when we need some fact). The *need-* prefix can be added manually during the rules creation. To fire a rule Jess needs a fact with a *need-* prefix in its working memory. Such fact can be added automatically (during reasoning) or manually (by the user), for example *(need-triple (predicate "hasSeller") (subject 3) (object ?y))*. If the rule fires and there is a way to obtain needed facts, they appear in the Jess working memory. The *need-* facts are the so called triggers (in the Jess language terminology). These facts correspond to the goals in the backward reasoning method. We apply manual addition of the *need-* prefix in rules as well as the generation of *need-* facts which allows the rules to be fired.

The hybrid reasoning method of RQA consists of the following elements:

- Set of facts.
- Set of rules for forward chaining.
- Set of rules for backward chaining.
- Set of mapping rules.
- Query algorithm.

The set of facts is stored in a relational database and are gathered during the reasoning process. Data (as triples) is obtained using a set of mapping rules. These rules define mappings between ontology predicates and the relational data. Ontology-based knowledge is represented as a set of rules for forward chaining. These rules are generated automatically by SDL and reflect SWRL rules and calculated hierarchies of concepts and properties. This set is transformed into a set of rules for backward chaining with *need-* prefix. All rules are in the form of Horn clauses and are written in the Jess language.

4.1.1 Generation of Rules for Backward Chaining

This section presents the transformation method of an ontology with rules for the backward chaining performed by the Jess engine. The transformation results in a script written in the Jess language. Overall transformation process is the following:

$$OWL + SWRL \;\; => \;\; Rules\ for\ FC \;\; => \;\; Rules\ for\ BC$$

An OWL ontology with SWRL rules is transformed using our *simple* transformation into set of rules for forward chaining (FC). This set is an input for the transformation into rules for backward chaining (BC). Both transformations are done automatically by SDL (see Chapter 5). The transformation methods differ in the technical details. We describe now the backward mode processing, because it is more complicated. For the clarity in this work, we do not present full URI addresses (only short names) and we use the following shortcuts: p - predicate, s - subject, o - object, and for *http://www.w3.org/1999/02/22-rdf-syntax-ns#type* - rdf:type.

The generation of the Jess script in backward mode is done in the following way:

- The template *triple* is created: *(deftemplate triple (slot p) (slot s) (slot o))* and information that *triple* is backward-reactive is added: *(do-backward-chaining triple)*,

- SWRL rules are directly transformed to Jess; for example the rule (*?x* and *?y* are the companies names and *?InV* is a number of issued invoice):

$$issuedVATIn(?x,\ ?InV),\ receivedVATIn(?y,\ ?InV) \\ \rightarrow\ TransactionBetween(?x,\ ?y)$$

  is transformed into the following rule:

```
(defrule Def-TransactionBetween
   (need-triple (p "TransactionBetween")(s ?x)(o ?y))
   (triple (p "issuedVATIn") (s ?x) (o ?InV))
   (triple (p "receivedVATIn") (s ?y) (o ?InV))
  =>
(assert(triple (p "TransactionBetween")(s ?x)(o ?y))))
```

- For the taxonomy of concepts/roles the appropriate rules are created; for example, for the hierarchy *VATInvoice is-a Document* the following rule is created:

```
(defrule HierarchyDocument
   (need-triple (p "rdf:type")(s ?x)(o "Document"))
   (triple (p "rdf:type")(s ?x)(o "VATInvoice"))
  =>
(assert (triple (p "rdf:type")(s ?x)(o "Document"))))
```

Such a generated set of rules can be loaded into a Jess engine to perform backward chaining. With the combination of mapping rules presented in Section 4.4 and a forward reasoning engine the rule-based query answering can be executed.

4.1.2 Query Algorithm for Hybrid Reasoning

Hybrid reasoning process supports rule-based query answering and uses two Jess engines: one for the forward chaining and one for the backward chaining. The queries are constructed in the Jess language in terms of ontology predicates. Queries can be presented as directed graphs (see Chapter 6). A mapping between the ontology predicates and relational data is used to express the semantics of the data. Data itself is stored in a relational database. The ontology and the mapping rules transformed into the Jess language format provide the additional semantic layer to the relational database. Such an approach allows for querying a relational database and reasoning using Jess, rules and an ontology.

The reasoning process is fully executed by the Jess engine and managed by the SDL library (see Chapter 5). We need to use two Jess engines, because backward chaining mode is very inefficient during queries execution. The reason for this inefficiency is that the Jess engine creates trigger facts (with *need-* prefix) during execution of a query and then calculates rules activations (but it does not fire any of the rules). This procedure does not occur in the forward chaining mode, so the answering process is much faster.

The backward chaining engine is responsible only for gathering data from the relational database. Data is added (asserted in Jess terminology) as triples into the engine's working memory. The forward chaining engine can answer a query with all constraints put on variables in a given query (=, !=, $<$, $>$ etc.). During the execution of a query the forward chaining engine does not reason (none of the rules is fired). Every Jess engine has its own working memory.

The beginning of the querying process involves loading a backward script generated (or written) in the Jess language into the backward engine. In the forward engine the template *triple* is created. Then the user can query about the properties and classes defined in the transformed ontology. A query is constructed in the Jess language and can be represented as a directed graph. The query algorithm is defined in Figure 4.1.

INPUT: Query Q, set of rules for backward chaining, set of mapping rules. Both sets are loaded into Jess engine for backward chaining.

OUTPUT: An answer for query Q.

METHOD:

Step 1. Create a rule from a given query Q and name it QUERYRULE. The query is the body of the rule, and the head is empty. Add QUERYRULE to the forward chaining engine.

Step 2. In the backward chaining engine, for all concepts/roles occurring in the query, do:

a) Add need-X fact/facts to the engine (where X is the current concept/role) with bounded variables (if it exists).

b) Run the engine - the reasoning process begins and during it the instances of the X predicates are obtained from a relational database.

c) If the group of queries is not empty, the aggregation is performed and one SQL query is executed. Results are added as triples to Jess working memory.

d) Copy results to the forward chaining engine, remembering variables bindings. If there is no result, the engine stops.

e) Clean the working memory of the backward chaining engine.

Step 3. In the forward chaining engine, get activations of the QUERYRULE. These activations contain facts that are results for a given query.

Figure 4.1: The hybrid reasoning and query algorithm.

Steps 1, 3 and 4 are executed in the forward reasoning engine, and step 2 in the backward reasoning engine. The Jess engine allows querying its working memory using a special function called *runQueryStar*. We decided to get facts from a rule activation because it is the most efficient way to obtain Jess query results (according to the Jess implementation [Hill 2003]).

For better understanding of the presented method, in Figure 4.2 we present an example with the following query: 'Find companies that received invoices issued by company *Comp1* on product ID=10'. The query, written in the Jess language, is the following:

```
(defquery ExampleQuery
   (triple (p "issuedVATIn") (s "Comp1") (o ?VIn))
```

```
(triple (p "receivedVATIn") (s ?Comp2) (o ?VIn))
(triple (p "refersToGood") (s ?VIn) (o 10)))
```

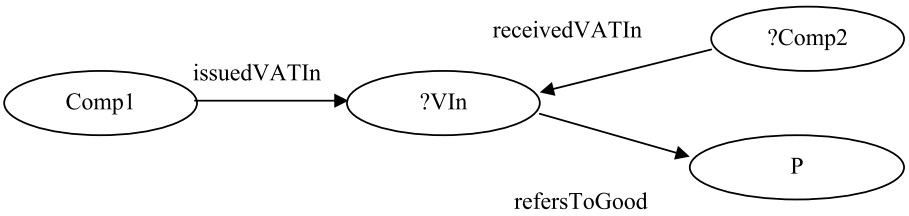

Figure 4.2: An example of a query involving three ontology predicates.

Our method is used to execute the query. In the first step, the QUERYRULE is created and added to the forward engine, so the querying process goes to the second step.

The second step is executed three times because of three relations in the query occurred. In this step the *(need-triple (p "issuedVATIn") (s "Comp1") (o ?VIn))* is asserted, and then the backward chaining engine is run. All results are copied to the forward chaining engine and values of variable *?VIn* are remembered. Then the working memory of the backward chaining engine is cleared. The second step is executed again, but now the *(need-triple (p "receivedVATIn") (p ?Comp2) (o ?VIn))* facts are asserted with bindings of variable *?VIn* (for example *(need-triple (p "receivedVATIn") (p ?Comp2) (o "8/2008"))*). When results are copied to the forward chaining engine the second step is executed again and the *(need-triple(p "refersToGood") (s ?VIn) (o 10))* are asserted with values of variable *?VIn*.

After reasoning in the backward engine, the query answering process goes to the third step. The example query is executed in the forward chaining engine and the query results are obtained.

4.2 Extended Rules Method

In this section we present a modified magic transformation algorithm which together with the Rete Pattern Matching algorithm increases speed and scalability of rule-based systems with the forward chaining. Our method is based on dependencies between variables appearing in predicates inside each rule. Our approach generates rules to be processed by the Rete-based engine which reflects the fact the extended rules method is Rete-dependent.

The original magic transformation [Bancilhon 1986a] is strongly connected with order of premises in the body of the rule. Thus, any permutation of atoms in the body gives a semantically equivalent rule. As a result, we can built different

sets of magic rules for diverse sequences of the atoms. This flexibility is very important in building efficient plans of a goal evaluation. Properly chosen subsets of magic rules form the basis of the *extended rules* in our our approach.

Arguments in atoms, particularly pertaining to the same variables, play a significant role, as they form information channels between atoms. In order to efficiently verify satisfaction of conditions from the rule body and infer a conclusion specified in the head, we are interested in finding dependencies between the atoms (predicates). We now define a subset of dependent predicates and extended rules.

Definition 4.1 (Dependent predicates)**.** *Let B and P be non-empty sets of atoms. A subset $Dep(B, P)$ of atoms from a set B which share a variable or a constant with some atom in the set P is called the set of dependent predicates.*

Assume that we have two sets of atoms: $B = \{p(x, y),\ q(z)\}$ and $P = \{r(y),\ s(w)\}$ then the set of dependent predicates is $Dep(B, P) = \{p(x, y)\}$. The generation method for extended rules with examples is presented in Section 4.2.1.

Definition 4.2 (Extended rules)**.** *Let R be a Horn clause, where unary or binary atoms are allowed. Such a rule is called the basic rule. A set of extended rules exR is generated automatically in the goal- and dependency-directed transformation from R for the evaluation purposes. The set exR is semantically equivalent to R.*

As mentioned earlier, ontology-based knowledge is represented as a set of rules (basic rules) and describes a source data at concept (ontological) level. Forward chaining in the integrated system is performed with extended rules, which are obtained by a goal- and dependency-directed transformation of the basic rules. The novel feature of our method is generality - every rule is generated so that includes all possible bindings of the head predicates, and variable dependencies, while in many implementations of the magic method the succession of bindings depends on a query. The presented method consists of the following elements:

- Two sets of facts: one including called facts (goals in a goal-directed reasoning, annotated with C) and the proper ones.
- Set of the basic rules.
- Set of the extended rules.
- Set of the mapping rules.
- Query algorithm.

The division of the facts is very important in our approach. *Proper facts* are directly derived from a relational database, or are inferred by rules from other proper facts. *Called facts* reflect *goals*. They are used to prevent firing more rules than is

required in the query evaluation process and are constructed during a generation of extended rules.

With the usage of the combination of proper and called facts, we can infer with the forward chaining scheme like with the backward one, where the reasoning is a goal-driven process.

The set of basic rules consists of rules which constitute the knowledge base. The set forms input data in our algorithm for the automatic generation of the extended rules. The set of extended rules is semantically equivalent to the set of basic rules. Extended rules contain called and proper predicates which reflect called and proper facts respectively. Together with the mapping rules (see Section 4.4), the extended ones are used in the query answering algorithm.

4.2.1 Generation of the Extended Rules

The extended rules are generated on the basis of the basic rules. In principle, we transform rules according to the magic transformation, the enhancement is proposed by the use of the dependent predicates.

During the generation process the special symbol *C* (for *called* atoms) can be added to predicates in rules. If a predicate does not contain any symbol it means that it matches only proper facts. If a predicate is annotated with *C* symbol, it matches only called facts. For example, the annotated predicate $p_1(?x,\ ?y)^C$ with variables *?x* and *?y*, can match following called facts: $p_1(x,\ y)^C$, $p_1(x,\ nil)^C$, $p_1(nil,\ y)^C$ or $p_1(nil,\ nil)^C$ where x and y are constants, and nil is a special value denoting an unbound variable.

Now we describe the generation process (an algorithm presented in Figure 4.3) of the extended rules. It is worth noticing, that each variable from the rule's head should occur in the rule's body (the Datalog safety restriction is used to guarantee algorithm decidability).

Every basic rule consists of the body B, additional predicates AP (see Section 2.1.3.1 and the head predicate *H*. In order to denote that B matches only proper or called facts, we mark it as B and B^C respectively. In the same way we indicate the head of the rule: H (adds proper facts) or H^C (adds called facts). Therefore, each basic rule is represented as follows:

$$B, AP \rightarrow H \tag{4.1}$$

In accordance with magic transformation, the body of the rule (4.1) is first augmented with the called predicate H^C to indicate an expected goal of the rule. To describe the needed (called) fact, one has to identify its arguments. Moreover, to define other extended rules for the basic rule (due to magic transformation), one can compose the appropriate subsets of different proper and called facts, and subsets of dependent predicates.

INPUT: Set of basic rules.

OUTPUT: Set of extended rules semantically equivalent to basic rules.

METHOD:

Step 1. For each basic rule (4.1) a rule of the following form is created:

$$B,\ AP,\ H^C\ \rightarrow\ H$$

where H^C contains patterns of attributes (the attributes may be bound or stay unbound) described with the *nil* alternative.

Step 2. For each basic rule (4.1) a new set of rules is generated, where none of the variables in the head predicate annotated with C symbol is bound. In this case, all variables are replaced by the *nil* value and rules are generated of the following form:

$$H^C \rightarrow P_i^C$$

where P_i is a predicate from the body of a basic rule ($B = P_1, ..., P_n$).

Step 3. For each basic rule (4.1) a new set of rules is generated according to the bindings of variables in the head. Get a set $D = Dep(B, H)$ of predicates from the set B, which depend on the bound variables in the head *H*, and create one rule for each dependent predicate D_i from the set D:

$$H^C \rightarrow D_i^C$$

Step 4. For each basic rule (4.1) a new set of rules is generated according to the bindings of variables in the head and dependent predicates. The bindings are connected from the body to the head by a chain of variables. This set contains rules in which called predicates are mixed with the proper ones with respect to the dependencies between variables.

Figure 4.3: The gsip strategy for the generation of extended rules.

An adornment of a rule, in our approach, is expressed by the use of nil value which represents a free variable. A variable that is bound is represented only by its name and a condition that checks if the variable's value is different from nil. Variables that are indicated only by "?" can be bound or free. For example, rule (2.17) is transformed into the following rule:

$$p(?x,\ nil),\ ?x \neq nil \rightarrow q(?x,\ nil)$$

We use ? sign when there is no matter if a value is bound or not. The following rule: $p(?x,\ ?y) \rightarrow q(?x)$ we can replace by another one: $p(?x,\ ?) \rightarrow q(?x)$. In this case, these two rules are equivalent to each other.

The algorithm defining a specialized magic transformation is based on the sip goal- and dependency-directed strategy. As our strategy is query-independent (the extended rules are generated only once), we call it the general sideways information passing (gsip) strategy. This algorithm is one of the main results of our work.

We stress out that all AP predicates are added to the body of each created rule if all variables appearing in predicates from AP also appear in the body.

Applying gsip algorithm to the following rule:

$$p_1\,(?x,\ ?y)\,,\ \ p_2\,(?y,\ ?w)\,,\ \ p_3\,(?w)\,,\ ?w \neq ?x \rightarrow h_1(?x,\ ?w)$$

we obtain the following sets of rules which correspond to the steps of the algorithm:

1. One rule which is generated by adding the goal connected with the head predicate:

$$p_1\,(?x,\ ?y)\,,\ p_2\,(?y,\ ?w)\,,\ p_3\,(?w)\,,\ ?w \neq ?x,\ h_1(?x,\ ?w)^C \rightarrow h_1(?x,\ ?w) \tag{4.2}$$

2. The set of rules with dependent predicates from the set

$$Dep(\{p_1\,(?x,\ ?y)\,,\ p_2\,(?y,\ ?w)\,,\ p_3\,(?w)\},\ \{h_1\,(?x,\ ?w)\})$$

$$= \{p_1\,(?x,\ ?y)\,,\ p_2\,(?y,\ ?w)\,,\ p_3\,(?w)\}$$

   where all the variables from the head are unbound. In such case, the variables are replaced by the *nil* value:

$$\begin{aligned} h_1(nil,\ nil)^C &\rightarrow\ p_1(nil,\ nil)^C \\ h_1(nil,\ nil)^C &\rightarrow\ p_2(nil,\ nil)^C \\ h_1(nil,\ nil)^C &\rightarrow\ p_3(nil)^C \end{aligned}$$

3. The set of rules with dependent predicates and different binding patterns of the head predicate:

$$\begin{aligned} h_1(?x,\ ?w)^C,\ ?x \neq nil &\rightarrow p_1(?x,\ nil)^C \\ h_1(?x,\ ?w)^C,\ ?w \neq nil &\rightarrow p_2(nil,\ ?w)^C \\ h_1(?x,\ ?w)^C,\ ?w \neq nil &\rightarrow p_3(?w)^C \end{aligned}$$

4. The set of rules with dependencies between proper and called predicates:

$$
\begin{aligned}
h_1(?x,\ ?)^C,\quad & p_1(?x,\ ?y) \rightarrow\ p_2(?y,\ nil)^C \\
h_1(?,\ ?w)^C,\quad & p_2(?y,\ ?w) \rightarrow\ p_1(nil,\ ?y)^C \\
h_1(?,\ ?w)^C,\quad & p_2(?y,\ ?w) \rightarrow\ p_3(?w)^C \\
h_1(?,\ ?w)^C,\quad & p_3(?w) \rightarrow\ p_2(nil,\ ?w)^C \\
h_1(?x,\ ?)^C,\quad & p_1\left(?x,\ ?y\right),\ p_2(?y,\ ?w) \rightarrow\ p_3(?w)^C \\
h_1(?,\ ?w)^C,\quad & p_1(?x,\ ?y),\ p_2(?y,\ ?w) \rightarrow\ p_3(?w)^C
\end{aligned}
\tag{4.3}
$$

Each extended rule in the fourth step of the algorithm is generated to pass only one binding of a variable from a proper fact to a called one. In such case, called predicates are mixed with proper ones in the rule's body and bindings are passed through a chain of variables from the body to the head. As a result we will obtain all possible bindings of variables that are strictly connected with a goal (in this case $h_1(\dots)$).

Each rule from the generated sets in steps 2-4 contains also a negated head predicate in the body. This technical modification is introduced because the Jess engine does not allow for duplicates in the working memory. The negation in this case should be understood as a condition: *if such fact does not exist in the working memory*. For example, the rule (4.3) is writen as follows:

$$h_1(?,\ ?w)^C,\quad p_1(?x,\ ?y),\ p_2(?y,\ w),\ not\ p_3(?w)^C \rightarrow\ p_3(?w)^C$$

In rules generated in Step 1 of the gsip algorithm we define *C* predicates with a *nil* alternative: $p(?x|nil,\ ?y|nil)^C$. It is a technical detail compatible with operation of the Rete algorithm. For example, if we have facts $p_1\left(1,\ 2\right),\quad p_2(2,\ 3),\quad p_3\left(3\right)$ and our goal is $h_1(nil,\ 3)^C$ the rule (4.2) will never fire because the variable *?x* is bound to *1* in the proper fact and to *nil* in the called fact. If we change our pattern to $h_1(?x|nil,\ ?w|nil)^C$ this rule will fire and we get a result. The pattern for h_1 can match the following facts: $h_1(x,\ y)^C$, $h_1(x,\ nil)^C$, $h_1(nil,\ y)^C$ and $h_1(nil,\ nil)^C$ where x and y are constants.

4.2.2 Query Algorithm for Extended Rules Reasoning

In our RQA method with extended rules a user poses a query to a rule-based system. The query is constructed from the predicates available in the knowledge base and from the additional predicates used for comparisons (<, ?, etc.). An answer is obtained as a result of the reasoning process using the forward chaining method. We assume that the engine contains a knowledge base constructed from extended

INPUT: Query Q, set of extended rules, set of mapping rules. Both sets are loaded into Jess engine.

OUTPUT: An answer for query Q.

METHOD:

Step 1. Create a special rule from a given query Q and name it *QUERYRULE*. The query constitutes the body of the rule. The head contains invocation of the Java method, which remembers bindings of the variables in the query when the rule is fired. The number of firings of the rule is the number of different results. Add *QUERYRULE* to the Jess engine.

Step 2. For every predicate p_i appearing in the query $Q = p_1, p_2, ..., p_n$ do the following:

1. Add predicate p_i with the C symbol and bound variables (if exist) to the engine's working memory. Replace all variables that are not bound with the *nil* value.
2. Run the engine - it reasons about facts in the working memory and generates partial SQL queries, which are grouped by the defined mapping
3. When reasoning stops, for every group of SQL queries, one aggregated SQL query is created and executed. The results are added as the instances of the according predicates (facts) to the working memory.
4. If there are activations of the rules in the engine, go to the point 2), or else go to the point 5).
5. Remember the bindings of the variables appearing in the predicate p_i.

Step 3. Return the results and remove *QUERYRULE* from the engine.

Figure 4.4: The reasoning and query algorithm performed with extended rules.

and mapping rules. Facts are stored in a relational database. The algorithm of the rule-based query answering with extended rules is presented in Figure 4.4.

The RQA method based on extended rules is similar to the hybrid reasoning method in the sense that both methods try to increase the efficiency of a rule-based query answering by the use of a goal-driven reasoning. In our hybrid reasoning goals are represented by facts preceded by *need-* prefix whereas in the extended rules method needed facts are marked with *C* symbol. The hybrid reasoning can

be applied only in the Jess engine while extended rules method can be applied in every Rete-based reasoning engine. This is the reason why the first method is called Jess-dependent and the second one Rete-dependent. Both methods increase the speed and the scalability of the Jess reasoning engine. The experiments that confirm our improvements are presented in Chapter 6. An example application of extended rules method is presented in Appendix A.

4.3 Complexity of Query Answering

Data complexity of conjunctive query answering in Horn logic is P-complete [Dantsin 2001]. Thus, data complexity of conjunctive query answering in Horn-$\mathcal{SHIQ}$ is in PTime with respect to data complexity [Calvanese 2013].

However, it is hard to define the computational complexity of our methods, because the Rete algorithm which is employed in reasoning and query answering is too complex to be described in general. That is because performance depends on declared rules and the data that is processed by them. However, the performance is the same as in the Jess reasoning engine (see Section 2.1.3.2. Additionally, the computational complexity of the executed SQL queries should be added. Due to the simple form of queries and the Rete algorithm, the computational complexity of SQL queries can be skipped.

We stress out that in both RQA methods sets of generated rules are fixed and contain the fixed taxonomy. Thus, we can consider data complexity as polynomial [Eiter 2008a], while standard reasoning in Horn-$\mathcal{SHIQ}$ is EXPTIME-complete in general [Krötzsch 2007].

Increased performance of the pure RQA in Jess, presented in Chapter 6, appears from the fact that modifications of patterns in the rules' bodies discriminate facts in the working memory. These discriminations reduce (especially in the extended rules method) the overall tests performed inside the Rete network. As a result, although the average number of patterns per rule growth, the reasoning performance increased. We assume that proposed optimization may apply in other reasoning engines (not Rete-based), but currently, we have not performed any tests. Such tests require implementation effort combined with analysis of other reasoning algorithms. This the reason for calling our methods as Rete-dependent, since we are convinced that the extended rules method can be applied in any Rete-based engine.

4.4 Mapping Between Ontology Terms and Relational Data

In order to enable semantic access to relational data, it is necessary to express relational concepts in terms of ontology concepts, that is to define mapping between the relational schema and ontology classes (concepts) and relations (roles). Given such a mapping, one can transform relational data to RDF triples and process that copy in semantic applications. This method has obvious drawback, such as maintaining synchronization. Another method is to create a data adapter based on query rewriting. Such adapters can rewrite SPARQL [Consortium 2008] query to SQL [Falkowski 2009] query and execute it in RDBMS. This method could be fast in data retrieval, but without a reasoner, the full potential of ontology cannot be exploited. The third method is to generate semantic data from relational data "on-the-fly", on demand for the requesting application, and then process that data with a reasoner. We use this method to fill a gap between the relational data representation and the semantically described data.

An important step in our RQA methods consists of linking data stored in a relational database to the knowledge base. We accomplish this by creating a special rules which contain simple SQL queries in their bodies. These rules serve as mappings that are used to relate knowledge predicates and the corresponding database. This section presents a method for mapping creation between terms of rule-based system and relational data. We describe this straightforward method and present some examples. We also propose a grouping algorithm to improve the database query answering process.

The mapping method between terms of rule-based system and relational data is based on "essential" predicates. We assume that every "essential" predicate has a corresponding SQL query. "Essential" means that the instance of the predicate cannot be derived from a set of rules during the reasoning process. Instead, it can be obtained only in the direct way as a result of the SQL query evaluation in the database. For example, in the OWL hierarchy of classes *ChairmanOfTheBoard is-a CompanysPrincipal is-a PersonConectedToCompany is-a Person*, where the class *ChairmanOfTheBoard* is a subclass of the class *CompanysPrincipal* etc. The class *ChairmanOfTheBoard* is then represented as an "essential" predicate. Usually, "essential" axioms are lowest level taxonomy axioms. Predicates that are not "essential" can be mapped also, but there is no need for such mapping since instances of these predicates can be obtained during the reasoning process.

In the extended rules method "essential" predicates with the *C* symbol express that appropriate facts are needed in the reasoning process. These predicates are equivalent to *need-* facts in the hybrid reasoning method. Presented examples are

based on the extended rules method, since it is Rete-dependent and thus more general.

In the mapping method we assume also that the ontology which is used and transformed into rules is properly constructed (the ontology is classified without inconsistencies).

A predicate-database mapping is defined as a set of rules, where each rule is of the following form:

$$SQL\_query \rightarrow essential\_predicate \tag{4.4}$$

A body of each mapping rule contains SQL query which is defined manually by a user. We assume that every SQL query has the following permissible form:

$$SELECT\ [R]\ FROM\ [T]\ < WHERE > < C,\ AND,\ OR > \tag{4.5}$$

where:

- R are the attributes (columns) - one or two according to the unary or binary terms (OWL Class, OWL DataProperty or OWL ObjectProperty),
- T is a table which is queried,
- $WHERE$ is an optional clause to specify the constraints,
- C are the constraints in the following form: <attribute, comparator, value>, for example: *Age* $>$ *21*,
- AND - is the optional SQL command.

Only this form (4.5) of queries is permissible in our system. As an example, assume that we have a table *Employee* with the following attributes: *ID, Name, CompanyID* and *Position*. The example of SQL query for the concept *ChairmanOfTheBoard* can be defined as follows:

$$SELECT\ ID\ FROM\ Employee\ WHERE\ Position = Chairman;$$

When we want to apply more constraints, we may use $OR\ /\ AND$ clauses. The mapping process requires defining SQL queries for all "essential" classes and properties. Take the example of the mapping of the property *isSignedBy*:

$$SELECT\ IDDoc,\ IDEmp\ FROM\ Signature; \tag{4.6}$$

The execution of the query (4.6) results in obtaining all instances of the relation between *IDDoc* of the document and *IDEmps* of the employees that signed this document. If the query is executed, the results are added to the working memory as proper facts.

During the reasoning process many SQL queries are generated. We developed an algorithm which groups queries that correspond to the same essential term. These queries are aggregated and only one SQL query is executed. The algorithm is presented in Figure 4.5. This algorithm is enabled due to a very simple form (4.5) of the permissible SQL queries. For instance, if we have the following unknown (needed) facts: *isSignedBy (5, nil)*^{C} and *isSignedBy (10, nil)*^{C}, and the mapping query (4.6), our grouping algorithm will create a query:

$$SELECT\ IDDoc,\ IDEmp\ FROM\ Signature$$
$$WHERE\ (IDDoc = 5\ OR\ IDDoc = 10);$$

Then, the query is executed in the relational database and results are added as facts to the reasoning engine's working memory. It is worth noticing that mapping

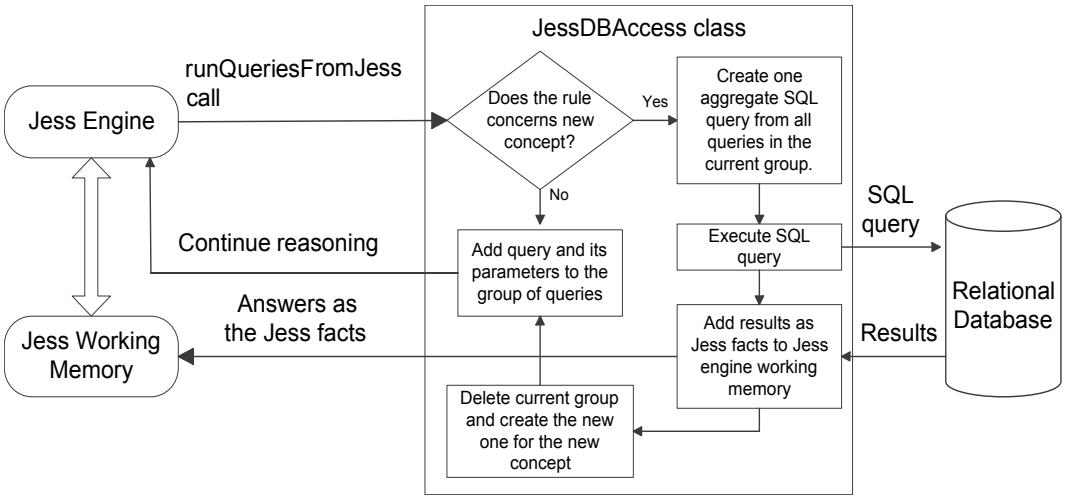

Figure 4.5: The grouping algorithm.

rules are automatically generated by SDL according to defined mappings between ontology predicates and SQL queries (see Chapter 5).

Function *runQueriesFromJess* allows accessing a relational database. It has the following parameters:

- name of the rule,
- SQL query defined for mapping,
- names of columns used to obtain results,
- variables values (if determined),
- name of the template used to add Jess fact (e.g. triple),
- connection to the relational database,

- instance of a Jess engine where facts should be added.

The presented algorithm of grouping SQL queries is implemented in the SDL library described in details in Chapter 5.

4.5 Discussion of the Related Work

In this chapter attention is focused on efficient rule-based query answering and an integration of system built of a rule-based component and a relational database. Our first solution was presented in [Bak 2009], where the hybrid system is described which consists of two reasoning engines, and relatively complex and costly data flow. Extended rules approach was published in [Bak 2011a]. Both our RQA methods are implemented in the SDL library.

More generally, we can find different strategies for processing conjunctive queries in relational databases. Most of the work, including that of Bancilhon [Bancilhon 1986a, Bancilhon 1986b], Ramakrishnan [Beeri 1987] and others [Lukácsy 2009] was done in eighties, when knowledge based systems were created for the first time. Various strategies of bottom-up, top-down and combined evaluation of queries are analysed by the authors together with different optimization techniques, such as magic set transformation of rules, or efficient counting and filtering.

A very close work is presented in [Brass 2010] where data-driven backward chaining is described. With automatically generated goals, a capability to represent unbound variables in goals and support for unification a system presented in the paper satisfies a lot of requirements of rule-based query answering systems. The main difference of our approach is that we modify a set of rules, whereas in [Brass 2010] they modified also a reasoning engine (architecture of a rule system).

Along the same lines contemporary efforts are also undertaken in view of the important task of data and systems integration. Two computation paradigms, namely relational databases and rule systems, need to be tightly and smoothly linked to better satisfy requirements of database and web programmers. This amounts for deriving queries from ontologies and thus deal with the semantic aspects of data. With standard languages of description logics the semantic web initiative contributes to a grow up of various rule systems. A thorough analysis of different technologies, performance and scalability results of the systems can be found in OpenRuleBench report [Liang 2009a].

There are several tools addressing the semantics-based relational querying problem or OWL to Jess mapping issue. OWLJessKB[6] project (discontinued since

[6] http://edge.cs.drexel.edu/assemblies/software/owljesskb/

2005) enabled mapping between OWL and Jess. In this approach, created templates (data structures in Jess) were stored as triples with URI (Uniform Resource Identifier)[7] addresses. OWLJessKB do not support SWRL and access to a relational database. Extended support of the OWL semantics and SWRL rules is provided by OWL2Jess [Mei 2005] and SWRL2Jess[8] tools. These tools do not provide relational database access or backward chaining.

Tools with database access include DataMaster [9], D2RQ[10] and KAON2[11]. DataMaster [O'Connor 2007b, O'Connor 2007a] is a Protégé-OWL[12] plug-in that allows to import a relational database structure or content into an OWL ontology. At present it just allows to populate the ontology with data from a relational database and to save this ontology to file and then query it with other Protégé plug-ins. Another tool is D2RQ [Bizer 2004, Bizer 2006], which allows treating relational databases as virtual RDF graphs. It works as a plug-in to RDF repositories, such as Jena[13] and Sesame[14], and it transforms relational data to RDF instances data. This transformation requires the description of a relation between an ontology model and a relational model, stated in terms of D2RQ Mapping Language. Data requested by the repository is transformed 'on-the-fly' to the RDF model. D2RQ acts as RDF adapter to relational data. In the ontology-based data access presented in [Poggi 2008] the mapping method was proposed. The main differences in comparison to our approach include: use of other formalism *DL-Lite*$_A$, top-down reasoning method, and a query rewriting technique. This approach goes beyond Datalog (because of applied function symbols) and does not use rules for directly accessing data. Rules are used to prepare an SQL query which will obtain required data. As a result, this approach differs in many aspects in comparison to the work presented in this thesis.

For more complex reasoning problems the leading tool is the KAON2 reasoning engine. This tool allows reasoning with OWL ontology and SWRL rules and enables to connect to the relational database through an additional mapping ontology (it has to be included into OWL ontology as an import). Mapping ontology has to be written manually. Such ontology enables to query the relational database 'on-the-fly' during reasoning.

As we can see, rule systems and reasoning engines form an interesting multiparadigm research area, where different methods and ideas are successfully applicable. Such significant examples are, for instance, ontology-based data access

[7] http://tools.ietf.org/html/rfc3986#section-3
[8] http://www.ag-nbi.de/research/owltrans/
[9] http://protegewiki.stanford.edu/wiki/DataMaster
[10] http://d2rq.org/
[11] http://kaon2.semanticweb.org/
[12] http://protege.stanford.edu/
[13] http://jena.apache.org/
[14] http://www.openrdf.org/

system, QuOnto [Poggi 2008] and Dlog [Lukácsy 2009] systems, specialized for Prolog.

4.6 Conclusion

In this chapter we presented two approaches that enable a rule-based query answering performed with the state-of-the-art Rete-based reasoning engine. Data is stored in a relational database and queries are posed in terms of an ontology concepts and roles.

In the hybrid reasoning method forward and backward chaining is used. In the extended rules method only forward chaining is executed. Both RQA methods use the reasoning process to obtain an answer for a given query. During this process facts from database are gathered and used to derive new facts according to a given set of rules. Next, the answer is constructed and presented. In both our methods the Horn clauses are used as the form of permissible rules. Our work is based on the Rete reasoning algorithm which is, for instance, implemented in the Jess engine. Moreover, the problem of the combination between ontology predicates and a relational database was also provided in this chapter.

These approaches allow the integration of an ontology, rules and database expressed in one format acceptable by the Jess engine. We believe that our approach, which enables complex queries to be created in a simple way, has advantages over SQL querying (creation of appropriate queries in SQL is more difficult). These methods can also be used in expert systems, which require many rules and lots of data from relational databases.

In our methodology an answer is always up-to-date, because a query is executed on the current state of the relational database. This is a very useful feature, because we do not have to prepare data to begin an execution of the query (in contrast to the forward chaining method in a pure reasoning engine).

The presented approaches have some significant limitations. Currently, the head of every rule contains only the *assert* command (it adds facts). We would like to be able to handle the *modify* and *retract* commands in the heads of the rules. We have to read data from a database and then we can test it (in the Jess terminology this means comparing values of variables). So we have to load some excess information. It would be better if we could test data during the load process and exclude data not fulfilling constraints.

Extended rules method is more general than the hybrid one. It depends only on the Rete algorithm. Extended, goal- and dependency-directed rules are constructed independently of a query, for all the binding patterns. Rule generation is performed only once, but with possibility to define diverse, specialised strategies.

Such approach increases also a scalability of a pure reasoning engine. Extended rules approach can be applied in every Rete-based reasoning engine.

The user of our system gets an easier way to pose queries (due to ontology origin of rules) than using structural constructions from SQL. The creation of queries, presented in the performance evaluation, is extremely difficult when we want to use pure SQL constructions. As a result, our method is useful in every system which requires data semantics to be explicitly given. Moreover, the graph-based representation of queries provides a very easy way to query and analyse such semantically described data.

Our methods largely remove a deficiency of the pure Jess engine mentioned on page 11 of [Liang 2009b]. It is worth noticing that engines were tested with all data in RAM memory, whereas our system is a complete platform that fetches only needed data into the working memory. This fact would be advantageous if combined rules execution and loading times were tested. The experiments that confirm the increased performance of the Jess engine are presented in Chapter 6.

CHAPTER 5

Implementation of SDL

In this chapter we present the Semantic Data Library (SDL) which is used to query a relational database at conceptual (ontological) level. SDL integrates a rule engine, a relational database and a set of rules obtained from the transformation of an ontology. We assume that the ontology can contain both OWL axioms and SWRL rules. This combination allows querying and inferring with data stored in a relational database using concepts (classes), roles (properties) and rules. Moreover, it simplifies the way of posing queries than using structural constructions from SQL. We describe an implementation of both rule-based query answering (RQA) methods and two transformation methods of OWL ontologies into sets of rules.

5.1 SDL Overview

SDL integrates ontologies, relational data and rules which represent domain knowledge. We need such tool when we have to pose complicated queries to the standard relational database. Due to the formally defined semantics we can pose a semantic query and get a corresponding semantic answer. SDL generates rules automatically which is very important for knowledge bases that often change.

The presented tool provides an easy way to query a relational database and both a query and an answer are based on the semantics defined in an OWL [Consortium 2006] ontology. The ontology describes data at the conceptual (ontological) level and introduces a formal definition of concepts and roles which do not exist directly in the database. For example, let us assume that we have a table $Persons(id,\ father\_id,\ mother\_id,\ gender)$. In the corresponding OWL ontology we can define the following concepts: $Grandfather$, $Grandmother$, $Cousin$ and roles: $hasBrother$, $hasSister$, $hasCousin$. These concepts and roles are not defined directly in the database. But with the use of the OWL ontology and SWRL [Horrocks 2004b] rules we can obtain instances of the aforementioned terms. Moreover, we can use these terms in queries which are in the form of directed graphs.

In the SDL implementation we apply rules that are obtained from a given Horn-$\mathcal{SHIQ}$ ontology. We adopt an approach that first calculates hierarchies of concepts and relations and then transforms these hierarchies into a set of rules. The TBox reasoning is performed by the Pellet engine [Sirin 2007]. Next, a classified form of

the ontology is transformed into rule definitions in the Jess language. SWRL DL-safe rules (if they occur) are also transformed into the Jess language with SWRL Built-ins[1] used as comparison predicates. SWRL extends the expressivity of OWL, supporting the use of ontology axioms in rules. Only unary and binary predicates are allowed.

Rules are also used to establish mappings between ontology axioms (properties and classes) and data stored in a relational database. These are defined in the Jess language and their creation is supported by SDL-GUI (see Section 5.5). They map "essential" axioms to appropriate SQL queries. "Essential" means that the instance of this axiom cannot be obtained from the taxonomy or rules, only directly from a database.

The SDL library is implemented in the Java language. We implemented our RQA approaches with the Jess reasoning engine. Extended rules method is directly applicable (not counting an interface modifications) to every engine, which exploits the Rete algorithm. Hybrid reasoning method is Jess-dependent since it uses Jess-specific backward chaining (see Chapter 4). All generated rules are in the form of Horn clauses.

5.2 SDL Architecture and Integration Process

The architecture of SDL, which covers both our approaches, is presented in Figure 5.1. The central part, which gathers input from other system elements and processes rules, are one [Bak 2011a] or two [Bak 2009] Jess engines used for forward and backward chaining.

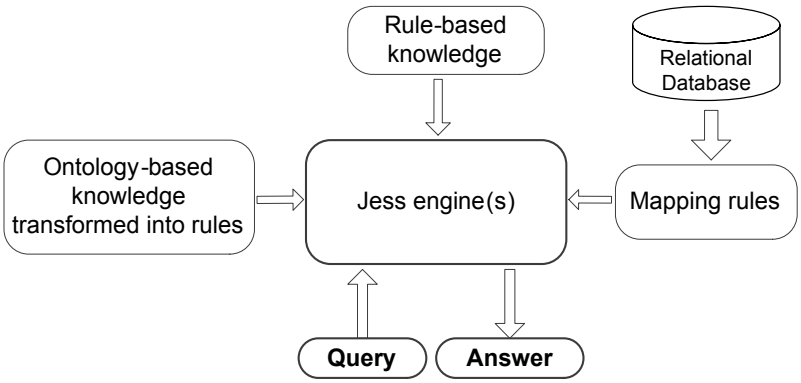

Figure 5.1: The architecture of the Semantic Data Library.

The hybrid approach [Bak 2009] exploits both forward and backward reasoning. The backward method is responsible for gathering data from a relational

[1] http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

database and the forward chaining is used to answer a given query. One instance of the Jess engine is created for each reasoning method. It means that we use two instances of the Jess engine in the hybrid approach.

Figure 5.2 presents the integration scheme of an OWL ontology with SWRL rules, the Jess engine and a relational database in RQA method with the hybrid reasoning. In this case, the transformation from OWL to Jess results in 3 Jess scripts: for forward and backward reasoning and a set of mapping rules. The set of rules which is obtained during the transformation from an OWL+SWRL ontology is saved as a Jess script file *FScript.clp*. The script is then transformed into a set of rules assigned to backward chaining (BC) - *BScript.clp*. Next, the mapping rules for BC are generated according to the defined mappings. A user can load: *BScript.clp* and a mapping rules Jess script; then establish a database connection and pose queries with SDL and Jess using our hybrid RQA method. *FScript.clp* can be deleted, it is needed only as a middle layer between an ontology and a set of rules assigned to backward chaining. It is worth noticing that rules assigned to BC are generated for all possible bindings of variables. It means they are aimed to determine if a given variable is free or bound. As a result, for each rule from a set stored in *FScript.clp* file the following number of rules is generated:

- 4 rules if a rule defines a property,
- 2 rules if a rule defines a class.

This generation method results from the implementation of the Jess engine [Hill 2003]. According to it, the backward chaining is simulated in the Jess engine and can be performed faster if we differentiate patterns in the body of a rule that match a backward chaining reactive templates. Determining bindings of variables in patterns improves the detection and matching of needed facts.

In the extended rules [Bak 2011a] approach we use one instance of the Jess engine, because only the forward reasoning method is used. *Extended* means that these rules are generated automatically from the basic ones for the evaluation purposes, and the modification is strongly connected with the magic transformation method. The set of basic rules consists of rules which constitute the knowledge base. The rule-based knowledge base comes from an OWL to Jess transformation. The set of extended rules is semantically equivalent to the set of basic rules. The extended rules are generated in the goal- and dependency-directed transformation. In this method we are interested in dependencies between variables appearing in predicates inside each rule. Together with mapping rules, the extended ones are used in the rule-based query answering algorithm. We implemented this approach according to the described algorithm. All sets of rules are generated automatically. Basic set of rules is generated from a given ontology. Extended set of rules is generated from the basic one. Mapping rules are generated from the defined mappings.

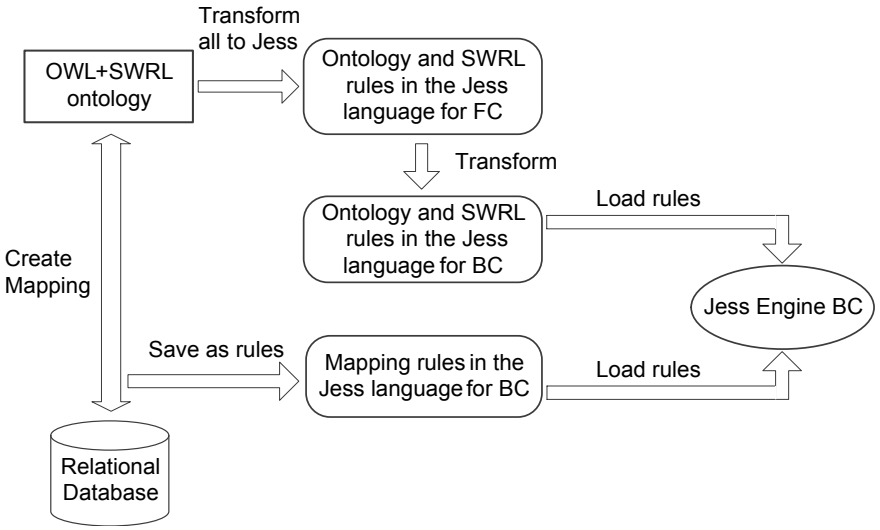

Figure 5.2: The integration scheme executed in the SDL library with the hybrid reasoning approach.

Figure 5.3 presents the integration scheme of an OWL ontology with SWRL rules, the Jess engine and a relational database in RQA method with the extended rules method. Such OWL+SWRL ontology is transformed into a set of rules in the Jess language. The set of rules is stored as a Jess script file *BRScript.clp*. The script is then transformed into a set of extended rules (*ExRScript.clp*). The mapping rules are automatically generated from a given mappings between ontology predicates and relational data. A user can load: *ExRScript.clp* and a mapping rules Jess script; then establish a database connection and pose queries with SDL and Jess. *BRScript.clp* can be deleted, it is needed only as a middle layer between a Horn-$\mathcal{SHIQ}$ ontology and a set of extended rules. It is worth noticing that a transformation in both our RQA methods need to be done only once (besides changes of an OWL ontology, SWRL rules or a database schema).

Both RQA methods employ a *triple* template to represent facts in generated Jess scripts. The triple consists of tree slots: *subject*, *predicate*, *object*. This template is augmented with a slot called *kind* used to represent proper or called facts in the extended set of rules. This slot does not occur in the hybrid approach. As mentioned in Chapter 4 we use shortcuts *p, s, o* to represent predicates, subjects and objects respectively. For better understanding of the representation we show an example of one of the generated extended rules (from the minimal ontology model) for obtaining instances of the class *ContractDocument* (we omit URIs in the example):

```
(defrule MAIN::Rule14
 (triple (kind P)(p "rdf:type")(s ?d)(o "Document"))
 (triple (kind P)(p "isSignedBy")(s ?d)(o ?p1))
```

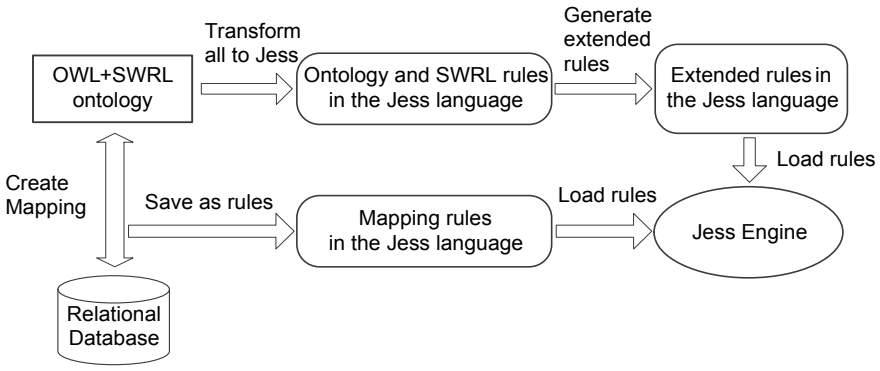

Figure 5.3: The integration scheme executed in the SDL library with the extended rules approach.

```
 (triple (kind P)(p "rdf:type")(s ?p1)(o "CompanysPrincipal"))
 (triple (kind P)(p "isSignedBy")(s ?d)(o ?p2))
 (test (neq ?p1 ?p2))
 (triple (kind P)(p "rdf:type")(s ?p2)(o "CompanysPrincipal"))
 (triple (kind C)(p "rdf:type")(s ?d)(o "ContractDocument"))
=>
(assert
(triple (kind P)(p "rdf:type")(s ?d)(o "ContractDocument"))))
```

5.3 OWL to Jess Transformation Methods

The SDL library supports two main methods of transforming OWL ontologies into rules expressed in the Jess language: *simple* and *Horn-$\mathcal{SHIQ}$*. The simple method transforms taxonomies of concepts and roles into a set of rules. These taxonomies are calculated by the Pellet engine first. SWRL rules and SWRLB predicates are also transformed into rules and Jess expressions. Symmetric properties of objects are also transformed into rules. The simple transformation can be done in the following modes:

Mode 1. Jess script assigned to forward chaining.

Mode 2. Jess script assigned to backward chaining.

Mode 3. Jess script assigned to forward chaining with extended rules.

The Horn-$\mathcal{SHIQ}$ transformation is an extension of the simple one. In this case, additional rules are generated according to OWL 1.1 Horn-$\mathcal{SHIQ}$ axioms. Rather than transforming the semantics of the OWL language into rules we create rules according to this semantics and a given ontology (in contrast to work presented in [Mei 2005] and [Meditskos 2008]). For example, when we have an ObjectProperty

inComplicityWith which is a SymmetricObjectProperty we create a rule which reflects that when an instance of this property occurs, a symmetric instance should also occur:

```
(defrule MAIN::HST-SymmetricProperty-inComplicityWith
   (triple (p "inComplicityWith") (s ?x) (o ?y))
 =>
(assert (triple (p "inComplicityWith") (s ?y) (o ?x))))
```

Currently, the implementation is prototypical and does not support all Horn-$\mathcal{SHIQ}$ axioms from the W3C specification [Grau 2006]. The SDL allows using simple atomic concepts (*A*, *C*), and roles (*R*). We assume that a concept *C* is simple if it is in one of the following form: A, $\exists R.A$, $\forall R.A$, or $\leq 1R.A$. Complex constructions are not supported. The universal and the existential quantifiers are used only as restrictions in the same way as presented in [Grosof 2003].

Currently supported OWL axioms are taken from the official Horn-$\mathcal{SHIQ}$ specification [Grau 2006] and cover the following list:

1. Class axioms:
   - equivalentClasses: URI | ObjectIntersectionOf | ObjectSomeValuesFrom
   - subClass: URI | ObjectUnionOf | ObjectIntersectionOf | ObjectSomeValuesFrom
   - superClass: URI | ObjectIntersectionOf
2. Property axioms: URI | equivalentObjectProperties | subObjectPropertyOf | objectPropertyDomain | objectPropertyRange | functionalObjectProperty | inverseFunctionalObjectProperty | symmetricObjectProperty

The Horn-$\mathcal{SHIQ}$ transformation can be executed in Mode 1 and Mode 3. In Mode 2 only simple transformation is implemented. SDL also provides the Horn-$\mathcal{SHIQ}$ transformation without hierarchy rules. This feature can be helpful to use scripts in different reasoning tasks.

5.4 Mapping Rules

The mapping method between ontology predicates and a relational data was described in Section 4.4. We now describe the implementation of this method.

Each mapping is transformed into one Jess rule. It means that we need exactly as many rules as defined mappings. The transformation is done according to the specified template:

Rule name: "Def-" + name of the mapping concept/relation

Body: ?r <- (need-triple (p "predicate's name") (s ?x) (o ?y)

Head: (call of the *runQueriesFromJess* function with its parameters) (retract ?r)

The *need-* facts which are triggers to fire rule are deleted from Jess working memory *(retract ?r)*. For this reason duplicates of firing the same rule do not occur. The example rule for property *MoneyTransferTo* between ID of the money transfer and the receiver's company name is shown below:

```
(defrule Def-MoneyTransferTo
  ?r <-(need-triple (p "MoneyTransferTo") (s ?x) (o ?y))
 =>
 (bind ?query (str-cat "SELECT Id, Receiver FROM transfers;"))
 (?*access* runQueriesFromJess
    "Def-MoneyTransferTo"  ?query
    "s;id;o;receiver;p;MoneyTransferTo;"
    (str-cat ?x ";" ?y ";")
    "triple"  ?*conn*  (engine))
 (retract ?r))
```

Function *runQueriesFromJess* allows accessing a relational database. The function takes the following arguments:

- name of the rule, e.g. `"Def-MoneyTransferTo"`,
- SQL query defined for mapping, e.g. `"?query"`,
- names of columns (assigned to slots in a template) used to obtain results, e.g. `"s;id;o;receiver;p;MoneyTransferTo;"`,
- variables values (if determined), e.g. `(str-cat ?x ";" ?y ";")`,
- name of the template used to add Jess fact, e.g. `"triple"`,
- connection to the relational database, e.g. `?*conn*`,
- instance of a Jess engine where facts should be added, e.g. `(engine)`.

An example of a mapping rule for a property *"MoneyTransferTo"* presents our mapping method in the hybrid reasoning approach. In the extended rules method the head of the rule is the same, but the body differs in a *need-* prefix and an additional *kind* slot. This difference is applied to all mapping rules in the extended rules method. For instance, the counterpart of the previous example in the extended rules approach is the following:

```
(defrule Def-MoneyTransferTo
 ?r <-(triple (kind C) (p "MoneyTransferTo") (s ?x) (o ?y))
 =>
 (bind ?query (str-cat "SELECT Id, Receiver FROM transfers;"))
```

```
(?*access* runQueriesFromJess
   "Def-MoneyTransferTo"  ?query
   "s;id;o;receiver;p;MoneyTransferTo;"
   (str-cat ?x ";" ?y ";")
   "Triple"  ?*conn*  (engine))
(retract ?r))
```

In Section 4.4 we present a mapping scheme where an SQL query is in the body of the rule and a predicate in the head. But, as we can see from the aforementioned examples the mapping scheme presented here is in reverse order. This difference results from the fact, that in Section 4.4 we presented a logical view on mapping. In this section the implementation in a goal-directed fashion requires reverse order, because SQL queries are posed only when an instance of a predicate is needed.

5.5 SDL Features

The SDL tool is implemented in the Java language. It is split into two modules:

- SDL-API (Application Programming Interface), which provides all functions,
- SDL-GUI (Graphical User Interface), which exploits SDL-API functions for defining the mapping between ontology terms and relational data; and provides automatic transformation of ontology into rules and the generation of Jess scripts.

The SDL-API module provides the following functionalities:

- reading a relational database schema,
- executing SQL query or procedure (results are added into Jess engine as facts),
- reading OWL ontology (with SWRL rules if available) and Jess scripts,
- Jess scripts generation (forward and backward chaining, extended rules, Horn-$\mathcal{SHIQ}$ transformation) from OWL ontology,
- mapping between ontology concepts/roles and relational data,
- executing a Jess query which consists of the concepts and roles from OWL ontology or templates defined in Jess language,
- rule-based query answering methods: hybrid and extended rules,
- Jess engine reasoning management (in forward and backward chaining).

SDL-GUI module of the library enables executing the following functions:

- reading ontology and viewing of concepts/roles hierarchies; the view contains classes hierarchy, object properties hierarchy and datatype properties hierarchy. These hierarchies are calculated by the Pellet engine [8],

- viewing a relational database schema which contains tables, views, columns and data types,
- mapping between ontology concepts/roles and relational data,
- populating an ontology with data from a relational database according to the specified mapping,
- creating Jess facts from a relational database according to the specified mapping,
- transforming OWL ontologies to Jess scripts,
- transforming Jess scripts into Jess scripts with extended rules (only *triple* template of facts is currently supported).

SDL supports interaction with the Pellet engine (for TBox reasoning with ontology and its classification), exploits OWL API [Horridge 2009, Horridge 2011] (for handling OWL files) and uses JDBC [2] library for MS SQL 2008 Server[3] access. The taxonomies of ontology classes and properties are classified by SDL-GUI with Pellet 2.3.0 and prepared for a user, who can define SQL mapping queries on these calculated taxonomies.

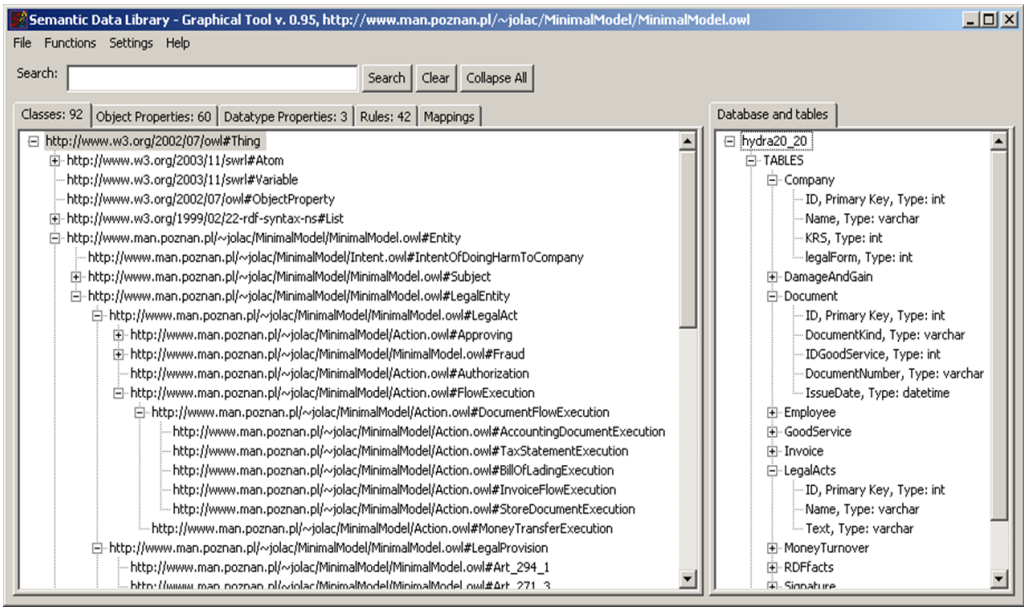

Figure 5.4: The SDL tool with minimal ontology model and database connection.

Figure 5.4 presents our minimal ontology model loaded into SDL-GUI and established connection to the corresponding relational database. A user gets a presentation of tables and views which exist in a database.

[2] http://msdn.microsoft.com/en-us/sqlserver/aa937724.aspx
[3] http://www.microsoft.com/en-us/sqlserver/default.aspx

5.6 Conclusion

In this chapter we described the SDL library. We presented two methods of transformation of an OWL ontology into Horn-SHIQ rules in the Jess language.

The SDL library is useful for queries creation because a user of our system gets an easier way to pose queries (due to ontology origin of rules) than using structural constructions from SQL. The creation of queries is extremely difficult when we want to use pure SQL constructions. The strictly defined semantics (in the form of an ontology) is another advantage of our tool.

In future, we are going to extend our approach to handle predicates with an arbitrary number of arguments. We will improve the rule-based query answering algorithm by using optimizations that concern extended rules and magic transformation.

Our SDL tool is available as a binary distribution and is free of charge for non-commercial academic usage (for universities only) and can be downloaded from the following Web site: http://draco.kari.put.poznan.pl/.

CHAPTER 6

Experimental Evaluation

In this chapter we present an experimental evaluation of two proposed rule-based query answering (RQA) methods applied to the knowledge base of economic crimes. The Semantic Data Library (SDL) is used to query a relational database at a conceptual (ontological) level. The minimal ontology model acts as a knowledge base, which is, for the evaluation purposes, transformed into set of Horn clauses. SDL uses and integrates the Jess engine, a relational database and the set of rules. This combination allows querying and inferring a crime scheme and identifies possible charges; in particular, it discovers crime activities and roles (of particular types of owners, managers, directors and chairs) using concepts, appropriate roles and rules. The performance evaluation is based on the financial crime minimal ontology model and artificially generated data sets.

The generated sets reflect data gathered during investigation and relevant to the potential charges. The ontology provides concepts and roles to which relational data is mapped and hierarchies of concepts and roles. Rules express dependencies and domain knowledge that allows querying about crime members' activities from a legal point of view. Therefore, the experimental evaluation is based on a single fraudulent disbursement typology (the Hydra case) combined with money laundering and core data related to this case. The rest of the data simulate variants of this crime by stochastically changing the values of the most important attributes. For some of these parameters no crime occurs, but generally various possible variants of persons' criminal activities are selected. The system goal is to detect a crime occurrence and bring proper charges based on people's activities. Despite handling a restricted class of crime, options of the crime case exhibit richness that is sufficient from a legal point of view. Instantiating the ontology allows querying of various aspects of the crime.

This experimental evaluation aims at proving that the system is practical and can be of big help if extended to a larger set of crimes. Since the analytic capability of institutions, such as the Police or Prosecutor's Office in Poland, is limited, the system has to hide the complexity of data structures and reasoning engines and expose friendly interfaces. One of the most costly and hard to trace crimes are economic crimes. The level of complicity and amount of proxy and scam companies involved make it difficult to identify details of crime schemes. Another issue is to properly formulate an indictment based on evidence, that, in particular, identifies roles and activities of members of a crime group.

SDL users will receive a convenient way to query a relational database and both a query and an answer are based on the semantics defined in an ontology. The ontology describes data at the conceptual (ontological) level and introduces a formal definition of concepts and roles which do not exist directly in the database. As a result, SDL handles the problem of gathering, managing, querying and interpreting data relevant to building the evidence necessary for an indictment. Moreover, SDL allows for classification of illegal activities and assignment of sanctions based on the roles of people in companies. As a result, the SDL tool will help investigators and prosecutors to conduct investigations of financial crimes and manage the data gathered.

During the development and research process, we have proposed and implemented two methods of querying a relational database: hybrid reasoning and forward reasoning with extended rules. In the following sections we:

- evaluate our minimal ontology model with both RQA methods,
- show that our approaches increase the scalability of the Jess engine and outperforms its rule-based query answering method.

Section 6.1 presents the Hydra-case-like simulated input data generation. In Section 6.2 we describe example queries selected according to the minimal ontology model. These queries are used to test different aspects of our query answering mechanisms. The performance evaluation and comparison with the pure Jess reasoning is given in Section 6.3. Section 6.4 contains concluding remarks and future work plans.

6.1 Generation of the Hydra-case-like simulated input data

For a practical demonstration of our rule-based query answering methods for a knowledge base of economic crimes (the minimal ontology model), we need data stored in a relational database. We implemented a generation tool in Java which enables us to generate a relational database with the size of a crime case as parameters: the number of companies and number of documents (invoices, work approval documents and money turnovers). The relational database reflects data gathered during investigation and relevant to the potential charges. The Hydra case generator provide data concerning:

- Information about employees and their position in a company - Employee table.
- Invoices with all obligatory elements (payer, seller, product, etc.) - Invoice table.

- Work approval documents (or the lack of them) - Document table.
- Signatures on documents - Signature table.
- Goods and services - GoodService table.
- Companies and their legal form - Company table.
- Money turnovers: money transfers, payments and withdrawals - MoneyTurnover table.
- Legal articles (name, ID and content) - LegalActs table.
- Information about illicit personal gains and damages to companies (with values) - DamageAndGain table.
- Other facts, like who knows about what (*Person knowsAbout document*) - these data result from testimonies. These facts are in the form of RDF triples (the table contains three columns: subject, predicate, object). These facts are stored in RDFfacts table.

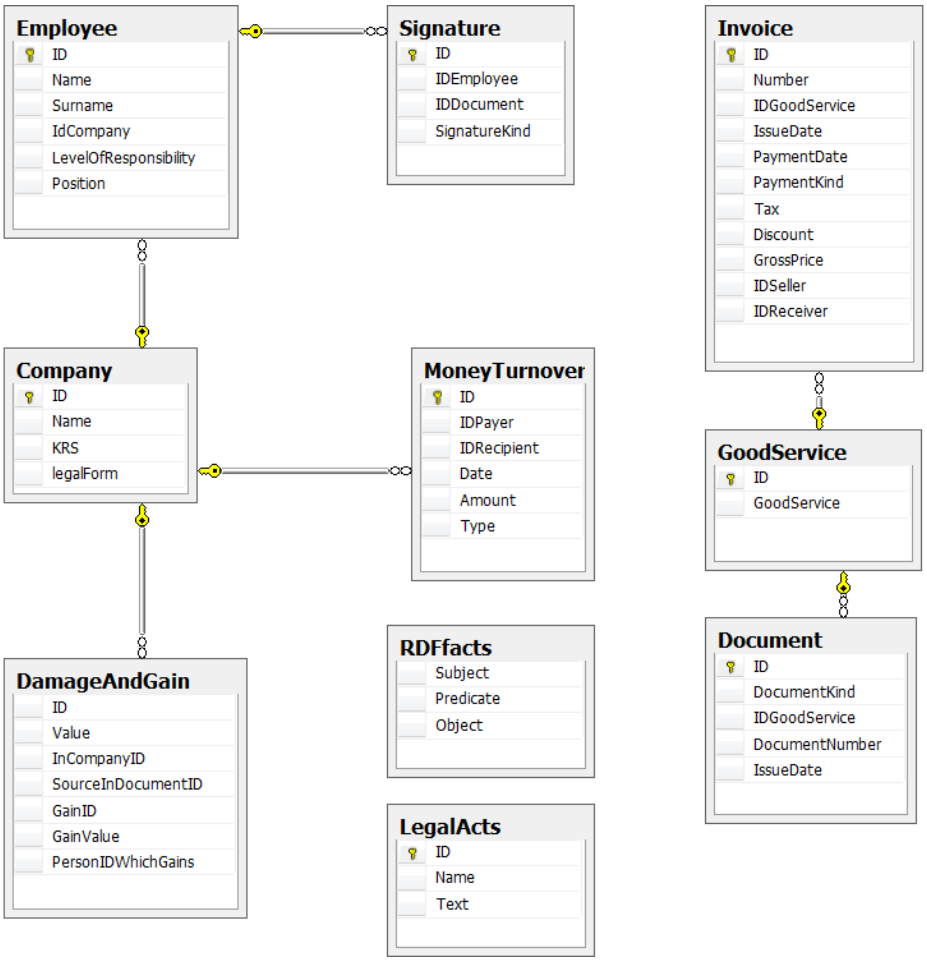

Figure 6.1: Database schema for the Hydra case.

The seed of the generator is fixed, so if the number of the documents is growing, the query results (i.e., the number of cases found criminal) from the bigger database contains all the results from the smaller database (and some extra results, possibly). The tool generates the Hydra case in four variants (as presented in Chapter 3) and also generates some obscuring data (other documents, invoices, etc.) to make reasoning more realistic. Every generated element has its own identification number. In this manner data are combined and internally coherent. The database schema is presented in Figure 6.1.

Data stored in relational databases is mapped to ontology predicates. A mapping between these two elements consist of 57 rules. We now provide two examples of mapping rules for extended rules reasoning (URI's are omitted). First example concerns *Director* concept and the following SQL query:

$$SELECT\ ID\ FROM\ Employee$$
$$WHERE\ Position\ =\ 'Director';$$

The generated mapping rule is the following:

```
(defrule MAIN::DB_Def-Director
  ?r <- (triple (kind C) (p "rdf:type") (s ?x) (o "Director"))
  =>
  (bind ?query
  (str-cat
     "SELECT ID FROM Employee WHERE Position = 'Director';"))
  (?*access* runQueriesFromJess
     "MAIN::DB_Def-Director"
     ?query
     "subject;ID;object;Director;predicate;hrdf:type;kind;P;"
     (str-cat ?x ";")
     "triple"   ?*conn*   (engine)))
```

Second example establish mapping between *knowsAbout* role and the following SQL query:

$$SELECT\ Subject,\ Object\ FROM\ RDFfacts$$
$$WHERE\ Predicate\ =\ 'knowsAbout';$$

The generated mapping rule is the following:

```
(defrule MAIN::DB_Def-knowsAbout
  ?r <- (triple (p "knowsAbout") (s ?x) (o ?y) (kind C))
  =>
  (bind ?query
  (str-cat "SELECT Subject, Object FROM RDFfacts
```

```
            WHERE Predicate = 'knowsAbout';"))
  (?*access* runQueriesFromJess
     "MAIN::DB_Def-knowsAbout"
     ?query
     "subject;SUBJECT;object;OBJECT;predicate;knowsAbout;kind;P;"
     (str-cat ?x ";" ?y ";")
     "triple"   ?*conn*   (engine)))
```

6.2 Example Queries

To realize what could be possible questions to the system, we present the relevant part of one count of charges in the Polish Penal Code:
"Article 296.

§1. Whoever, while under an obligation resulting from provisions of law, a decision of a competent authority or a contract to manage the property or business of a natural or legal person, or an organizational unit which is not a legal person, by exceeding powers granted to him or by failing to perform his duties, causes it to suffer considerable material damage, shall be subject to the penalty...

§2. If the perpetrator of the offence specified in §1 acts in order to gain a material benefit...

§3. If the perpetrator of the offence specified in §1 or 2 causes significant material damage of great extent...

§4. If the perpetrator of the offence specified in §1 or 3 acts unintentionally..."

The specificity and the simple nature of the Hydra case is that all persons, possibly including the CEO, knew that they were committing a crime; therefore, the model does not have concepts and data explaining their intentions. It is extremely difficult to model and answer predicates like: "exceed powers granted to him" (which could depend, for example, on taking an excessive risk) or "fail to perform his duties". A judge has to answer these questions that pertain to a given crime's attributes. At this stage our model does not contain such soft crime attributes. It contains facts and hard concepts, such as "cause a company to suffer considerable material damage". The system is not prepared to answer directly all questions a judge might ask. However, we could ask questions: all counts of criminal activities of a person *X*, or all persons subject to counts of a charge *C*.

We have prepared five queries to test different aspects of the query answering mechanism. Descriptions of queries are the following:

Q1. Query searches for a person (?p) which caused damage (?d) in a company (?c) and gained money in an illicit way. The damage has a high value (?h) which is more than 100.000 Polish zloties. The illicit personal gain is accompanied by the damage to the company.

Q2. Query searches for a company's ID (?c) in which person with ID=11 is a Principal and a person with ID=12 is a Director.

Q3. Query searches for a person (?p) which signed a falsified complex internal legal document (?d). It is a document that can consist of several documents that authorize the payment.

Q4. Query searches for a ID of the Penal Code article 299§1 and checks if the person with ID=11 is a Director and falls under this article.

Q5. Query searches for IDs of articles 299§1 and 299§5 and checks if the person with ID=84 is a Company's Principal and falls under these two articles.

The first query contains only variables (without any values) and exploits only hierarchy rules. The second query contains variables and values; it exploits ontological rules (for inComplicityWith symmetric property). The third query contains only variables and exploits hierarchy rules. The fourth and fifth queries contain variables and values, and exploit various characteristics of the knowledge base as coded by rules. The last two queries are computationally demanding - the property *fallsUnder* needs almost all rules to be fired, because it requires evidence why a person falls under a given article. Moreover, a rule can be fired more than once in case when appropriate facts exist in an engine's working memory. Graphical representation of the queries is presented in Figure 6.2.

Queries were executed with the use of the hybrid reasoning process, the extended rules reasoning and with the pure forward and backward reasoning in the Jess engine. Results of the executed queries are presented in Section 6.3.

All queries exploit mapping rules (because they are responsible for gathering data from the relational databases). For anonymity, the numbers in responses to queries are IDs of objects (persons, companies); if needed they can be transformed into real names. Queries without any values need more time to execute because SDL has to check all possible variable bindings from the database.

6.3 Performance Evaluation

For a practical demonstration of the SDL library we used the minimal ontology model with artificially generated data sets. These data sets contain information about: companies, employees, documents, invoices, money turnovers, legal sanctions for this class of crimes, etc. (see Section 6.1). We prepared three databases

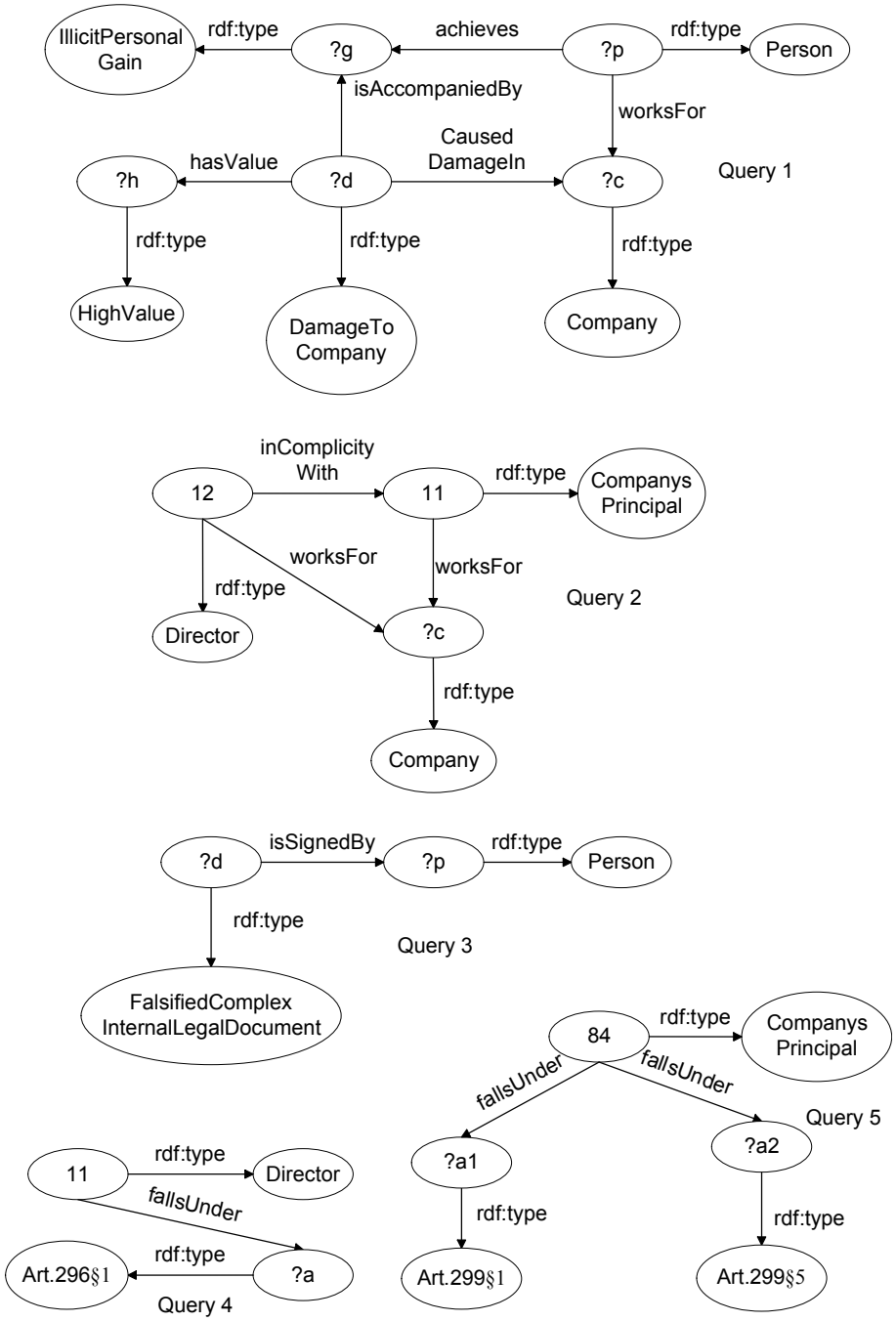

Figure 6.2: The test queries.

which differ in the size of the generated data. The number of companies and employees are the same in every database (20 companies and 240 people). Generated databases are the following: 20, 100, 200. These three numbers reflect parameters

of generation and are included in database's names. All tests were performed three times and average numbers were written in the tables.

The following tables: *Company*, *Employee*, *GoodService* and *LegalActs* are fixed in the size of data. The numbers of tuples generated in databases are presented in Table 6.1.

Table 6.1: Numbers of generated tuples in relational databases.

Table	Database 20	Database 100	Database 200
Company	20	20	20
DamageAndGain	12	102	218
Document	5	54	133
Employee	240	240	240
GoodService	20	20	20
Invoice	19	96	193
LegalActs	14	14	14
MoneyTurnover	15	84	177
RDFfacts	37	294	597
Signature	16	174	430
Total	**398**	**1098**	**2042**

A mapping between a relational database and ontology consists of 57 mapping rules. The minimal ontology model contains 42 SWRL DL-safe rules (75 Horn clauses – some SWRL rules contain conjunctions of predicates in their heads). The ontology was transformed into rules with the simple transformation, since it is sufficient for our five test queries. We present also an efficiency comparison between Horn-$\mathcal{SHIQ}$ transformation and the simple one. Sets of domain knowledge rules contain:

- 116 rules for forward chaining generated in the simple transformation.
- 369 rules for backward chaining generated in the simple transformation.
- 3444 rules for the extended rules reasoning generated in the simple transformation.
- 3585 rules for the extended rules reasoning generated in the Horn-$\mathcal{SHIQ}$ transformation.

We executed our five test queries on a PC machine with the following parameters:

- Processor: Intel Core2 Duo 2GHz, 4MB of cache memory.
- Random Access Memory: 2GB, 667MHz.
- Microsoft SQL Server 2008.

- Java Heap Space was set at 1024MB.

The results of a rule-based query answering with the hybrid reasoning method (HR) and the extended rules method (ExR) are presented in Table 6.2.

Table 6.2: Results of queries execution in our RQA methods.

Query and info		Database 20		Database 100		Database 200	
		HR	ExR	HR	ExR	HR	ExR
Query 1	[ms]	781	219	1 328	891	1 922	969
Results	[number]	54	54	474	474	1 036	1 036
Rules Fired	[number]	74	251	441	1 630	796	3 001
Query 2	[ms]	2 734	437	37 141	4 125	163 968	19 391
Results	[number]	1	1	1	1	1	1
Rules Fired	[number]	1 076	1 506	36 260	13 179	225 381	29 593
Query 3	[ms]	2 875	359	36 344	14 938	183 047	116 593
Results	[number]	18	18	322	322	1 004	1 004
Rules Fired	[number]	1 367	2 005	38 457	41 755	232 583	359 681
Query 4	[ms]	5 437	1 859	128 719	35 656	Time	347 110
Results	[number]	1	1	1	1	exceeded	1
Rules Fired	[number]	2 040	5 467	57 091	58 520	10 min.	597 711
Query 5	[ms]	9 312	1 234	Time	34 500	Time	343 469
Results	[number]	1	1	exceeded	1	exceeded	1
Rules Fired	[number]	2 540	5 828	10 min.	61 199	10 min.	608 925

As we can see, simple queries (1, 2, 3) are executed in an efficient way (if we look at how many rules were fired). For more complex reasoning (queries 4 and 5), further optimization is needed. Results presented in Table 6.2 show that our extended rules approach beats the hybrid one. Since we did not apply all possible optimizations (including supplementary magic sets, more efficient implementation, counting, rule-dependent sips), we are convinced that the efficiency of our method can be improved. Moreover, we notice that the algorithm presented in Figure 4.3 generates a number of excess rules. If we are able to remove such rules we will increase the performance of the algorithm described in Figure 4.4.

We compared our results with the Jess engine using forward and backward chaining separately. Appropriate scripts were loaded into two Jess engines. Facts (the same as in databases) were loaded from files. A comparison of our results with pure forward and backward reasoning in Jess system is presented in Table 6.3. The times of executing queries are measured for the same database (loaded from the files) and juxtaposed in the table under, respectively, **F** letter (for forward) and **B** letter (for backward). Numbers below **F** and **B** indicate times of data loading into working memory in milliseconds. While loading data from the third database, the

Table 6.3: Results of queries execution and comparison to the pure forward and backward Jess engines.

Query and info		Database 20			Database 100		
		F	**B**	**ExR**	**F**	**B**	**ExR**
Facts loading time [ms]		375	534		14 665	183 704	
Query 1	[ms]	281	328	219	15 938	19 906	891
Results	[number]	54	54	54	474	474	474
Rules Fired	[number]	1 873	1 873	251	15 567	15 567	1 630
Query 2	[ms]	234	266	437	14 875	19 344	4 125
Results	[number]	1	1	1	1	1	1
Rules Fired	[number]	1 820	1 820	1 506	15 094	15 094	13 179
Query 3	[ms]	250	281	359	14 516	19 688	14 938
Results	[number]	18	18	18	322	322	322
Rules Fired	[number]	1 837	1 837	2 005	15 415	15 415	41 755
Query 4	[ms]	250	250	1 859	14 319	19 718	35 656
Results	[number]	1	1	1	1	1	1
Rules Fired	[number]	1 820	1 820	5 467	15 094	15 094	58 520
Query 5	[ms]	250	266	1 234	14 559	20 065	34 500
Results	[number]	1	1	1	1	1	1
Rules Fired	[number]	1 820	1 820	5 828	15 094	15 094	61 199

size of the Java heap space was reached (in both engines), so the queries could not be executed. It seems obvious that for small databases, it is better to store data (facts) in the engines' working memory. But for the bigger databases, the problem with scalability occurs. In such cases our approach seems promising.

We also executed test queries with extended rules method and Horn-$\mathcal{SHIQ}$ transformation rules and compared them to the results achieved with the simple transformation rules. The results of the comparison are shown in Table 6.4, where **HS** and **Simple** express the execution of queries using Horn-$\mathcal{SHIQ}$ and simple transformation, respectively. An addition of Horn-$\mathcal{SHIQ}$ rules makes query answering process more complicated and computationally demanding. It results from fact that Horn-$\mathcal{SHIQ}$ transformation contains more OWL axioms than the simple transformation.

Presented results confirm that our approach significantly improves a scalability of a rule-based system. It is a very important issue, because in the forward chaining rule-based systems, facts have to be stored in the working memory which is, in general, limited by the RAM memory. If we store facts outside of the memory and load them only when they are needed, we achieve better scalability. Unfortunately, in a case when we pose a query without any bound variable, we have to load all

Table 6.4: Results of queries execution with Horn-SHIQ transformation and extended rules compared to the simple transformation.

Query and info		Database 20		Database 100		Database 200	
		HS	Simple	HS	Simple	HS	Simple
Query 1	[ms]	234	219	718	891	1 547	969
Results	[number]	54	54	474	474	1 036	1 036
Rules Fired	[number]	450	251	2 896	1 630	5 121	3 001
Query 2	[ms]	1 125	437	21 469	4 125	181 828	19 391
Results	[number]	1	1	1	1	1	1
Rules Fired	[number]	5 287	1 506	44 413	13 179	262 264	29 593
Query 3	[ms]	1 312	359	33 172	14 938	Java	116 593
Results	[number]	18	18	322	322	Heap	1 004
Rules Fired	[number]	6 228	2 005	53 842	41 755	Space	359 681
Query 4	[ms]	2 313	1 859	39 063	35 656	Java	347 110
Results	[number]	1	1	1	1	Heap	1
Rules Fired	[number]	6 748	5 467	63 213	58 520	Space	597 711
Query 5	[ms]	1 610	1 234	40 406	34 500	Java	343 469
Results	[number]	1	1	1	1	Heap	1
Rules Fired	[number]	7 111	5 828	66 061	61 199	Space	608 925

data from a database. In such a case our method will achieve worse results than the traditional one, because we lose some time for data loading.

The SDL's demonstration with above test queries and presented query answering methods are available on the demo site[1]. The minimal ontology model is added to the demo material. On the demo site a user has an option to pose her/his own query constructed from concepts and roles from the minimal ontology model. Two databases are available: Database 20 and 100. The SDL demo took part in RuleML Challenge competitions in 2010 [Bak 2010a] and 2011 [Bak 2011b].

6.4 Conclusion

In this chapter we demonstrated that the SDL library can be used to solve practical problems with formally defined semantics (in our case the minimal ontology model). We prepared and executed five queries that are used to test different aspects of the system.

The SDL library is useful for queries creation because a user of our system gets an easier way to pose queries (due to ontology origin of rules) than using structural constructions from SQL. The creation of queries, presented in the performance

[1] http://draco.kari.put.poznan.pl/

evaluation, is extremely difficult when we want to use pure SQL constructions. The strictly defined semantics (in the form of an ontology) is another advantage of our tool.

Finally, we justified that our approaches increase the scalability of the Jess engine and outperforms its rule-based query answering method.

In further research, we consider the development and implementation is the reasoning path, which can be presented to the user and can explain what evidence proves that a person falls under a concrete legal article. Analysis of justifications may give important information on ontology incompleteness, which often happens when a model is extended.

The future work could also consist in testing other ontologies with our tool. We could improve the rule-based query answering algorithm by using optimizations that concern extended rules and magic transformation.

CHAPTER 7

Conclusions and perspectives

The problem of RQA, undertaken in this thesis, can be perceived as a modification of deductive databases approach (DD) [Gallaire 1984]. The main difference is based on application of a description logic as an ontology, and thus the formal semantics. Since we employ the Horn-$\mathcal{SHIQ}$ language, the rule representation is similar to DD approaches. However, using ontologies we need to handle only unary and binary predicates, while in deductive databases n-ary predicates (relations) are permissible.

Deductive databases have been studied since 1980's, and main application of the rule methods in the database area have been inclusion of recursive queries into SQL engines. Altogether, they have been out of fashion for many years mainly due to lack of scalability. However, it is a fact that deductive databases are growing in importance in last few years [Winslett 2006, Hellerstein 2010, Huang 2011] and we are convinced that our research has a potential in commercial applications.

In recent years Datalog and related languages have been proposed for use in a wide range of practical settings including security and privacy protocols, program analysis, natural language processing, probabilistic inference, multiplayer games, telecom diagnosis, declarative networking and distributed systems [Hellerstein 2010]. Significant improvement of Datalog engine efficiency have been reported [Tekle 2011, Alviano 2012, Behrend 2011, Liu 2009].

This thesis concerned the following issues:

1. Presentation and evaluation of two novel methods of a rule-based query answering (RQA) applicable to relational databases with formally defined semantics. The methods have been incorporated into the developed SDL environment. The second method is related to the magic transformation, however no comparison with known versions of magic transformations on known results or accepted benchmarks has been done.

2. Development of the knowledge base of two economic crimes, namely fraudulent disbursement and money laundering. The knowledge base was implemented as a Horn-$\mathcal{SHIQ}$ ontology extended by DL-safe rules.

3. Demonstration that the proposed methods gain the performance of a Rete-based engine for these logic crime models.

We conclude our work pointing out the main results in Section 7.1 and indicating directions for future research in Section 7.2.

7.1 Main Results

We proposed two methods of a query evaluation in a rule-based system where data is stored in a relational database and queries can be posed at a conceptual (ontological) level. We designed a knowledge base of economic crimes which describes fraudulent disbursement and money laundering in order to discover crime activities and to suggest legal sanctions for crime perpetrators. The key contributions include the following results (listed in the order of significance):

- We devised a novel modification of a magic transformation which introduces extended rules. The approach exploits forward chaining that is based on the Rete algorithm and combines the Jess engine, a relational database and a Horn-$\mathcal{SHIQ}$ ontology. The combination allows to query a relational database at a conceptual level. The extended rules method is Rete-dependent. The method is defined over source data from a relational database, and with the Rete-based forward chaining reasoning over extended, goal- and dependency-directed rules. The novel approach is its generality. First of all, extended rules of the system are constructed independently of a query, for all the binding patterns. Moreover, rules generation is performed only once, but with possibility to define diverse, specialised strategies. Such approach increases also a scalability of a pure reasoning engine. In addition, the extended rules approach is more efficient than a query answering with standard forward or backward evaluation, outperforming our hybrid reasoning method. Finally, this approach is more flexible and more scalable in comparison to the pure Jess reasoning.

- We formulated a knowledge base of economic crimes: fraudulent disbursement and money laundering as a minimal ontology model. The ontology describes a real crime case, the Hydra case, using Horn-$\mathcal{SHIQ}$ language and is implemented in the OWL language supported by SWRL rules. Reasoning and query answering with the proposed knowledge base determine the possible types of legal sanctions for crime perpetrators and enable to discover their crime activities.

- We proposed a straightforward mapping method between a knowledge base and a relational database. The method is based on so-called *essential* predicates and SELECT-PROJECT-JOIN SQL queries. As a result, users of our system will be provided with simplified and more convenient way to

pose queries (due to ontology origin of rules) than SQL structural constructions. The creation of queries, presented in the performance evaluation, is extremely difficult when we want to use pure SQL constructions.

- We proposed a novel approach to query a relational database at a conceptual level using a hybrid approach. The approach combines forward and backward chaining performed by the Jess engine in a rule-based query answering task. This is a Jess-dependent method.

- We developed a new SDL (Semantic Data Library) framework, offering the implementation of both our methods of a rule-based query answering: hybrid and extended rules reasoning. The SDL tool can be used in systems that require many rules and lots of data which currently are processing in an inefficient way. Moreover, an answer for a query, evaluated with SDL, is always up-to-date because the query is executed on the current state of a relational database. Since SDL provides an application programming interface, it can be used as a library or as a standalone application.

- We evaluated our two approaches of a rule-based query answering for a knowledge base of economic crimes. We compared reasoning techniques available in the Jess engine with our proposed methods. Performed experiments justified the assumption that optimization of a rule-based query answering is possible in the state-of-the-art Jess reasoning engine, which is an implementation of the Rete algorithm.

7.2 Future Work

We consider the proposed methods of RQA as a starting point for further research direction towards modifications of the extended rules approach. This method is more general than the hybrid reasoning technique, which can be applied only in the Jess engine (it exploits Jess-specific reasoning).

In further research, we will consider new sideways information passing strategies that could be rule-dependent in a generation of the extended rules (rule-dependent sip strategy). Currently, we proposed a general sip which is applicable to each basic rule generated in the transformation of a Horn-$\mathcal{SHIQ}$ ontology into rules.

Another direction of the future work is the research towards OWL 2, which contains profiles assigned to reasoning with rules and a query answering. Since we considered only OWL 1.1 in this thesis, this direction seems a very natural consequence of our work.

In both our RQA methods the head of each rule contains only the *assert* command. We would like to be able to handle other possible commands like *retract* (it deletes data) or *modify*. The addition of these commands would allow to manage data according to its semantics and thus extend the possible field of application.

The future work should focus on concentration in developing the explanation service, which can present individual reasoning steps to a user and thus explain the results of a given query.

We have already obtained positive results of tests performed on the prototype system. The comparison of our results and those obtained in a pure Jess system seems to be an adequate and objective assessment of usefulness of our work. In future, we will be aimed at making detailed comparisons with systems using variants of magic transformation. Ultimately we would like also to apply the system to the OpenRuleBench suite of benchmarks. Such comparison requires an extension of our methods to handle predicates with an arbitrary number of arguments. Moreover, it requires a development effort to implement the combination of OpenRuleBench tests with our RQA methods. The comparison with ontology benchmarks like LUBM, is also planned. However, it would require a method for generating data stored in a relational database. It is worth noticing that in most benchmarks reasoning engines are tested with all data in RAM memory, whereas our system is a complete platform that fetches only needed data to the working memory. This fact would be advantageous if combined rules execution and loading times were tested.

In further research, we may also consider the comparison of our mapping method with other similar approaches, including Quonto, D2RQ, R2RML recommendation (RDB to RDF Mapping Language) and commercial systems for storing RDF data. A method for selecting essential predicates from an ontology would be valuable, since we need to choose these predicates manually.

We will continue our research aiming to elaborate a new method of rules transformation, which would allow for more efficient application of rules in a query answering task. The implementation in other Rete-based engines, e.g. Drools, is also recommended.

Finally, an obvious future research plan include the implementation and practical verification of the proposed methods in a tool that could be used by investigators, prosecutors and policemen each day of their work. We observed that a way of creating rules in a convenient way is needed. We have already started the research in this direction [Nowak 2012].

Rule-based systems will soon dominate in areas of combating banking and insurance fraud, as they enter into offers of major companies: IBM, FICO, Experian.

In addition, there are active start-ups with comprehensive offers: Logicblox[1] and Highfleet[2].

The final goal would be application of financial knowledge bases in jurisdiction on a wide scale. In addition to limitations discussed in Section 3.5 that are of a conceptual level there are other serious organizational and technical obstacles.

We put facts into structures manually and designed the minimal ontology based on 10 large cases. This number is much too small to use statistical methods. Obtaining data for live cases is legally restricted and in Poland almost impossible. Anonymization and cleaning data is very expensive (in one of the cases, H. Musialski case, the name of his company appeared in documents in 72 forms). Even if one becomes an expert for police, prosecutors or judges (which happened in the group within which we were working), the details of cases cannot be used.

To automatically process laws that are expressed in natural language, they must be made machine-readable. There are several approaches to making legal texts machine-readable, depending on the goals and purposes to be served by the processed text. Among the approaches, legal texts may be processed to link documents, to annotate for information extraction, and to translate them to a formal representation [Wyner 2012].

We need machine ontology instantiation, rather than manual processing. Our group is now preparing a prototype of the system that would be capable of doing this.

It is widely believed that electronic discovery (or e-discovery or eDiscovery) in investigations and court proceeding that deals with the exchange of information in electronic format, is a necessary condition to prosecute financial crimes. In the US "not one banker, not one executive on Wall Street" [Frontline 2013] faced criminal prosecution for deliberate packaging of toxic loans and selling them to investors with regard in financial crisis of 2008 because people involved at the top of a major financial organizations (for whom it is very difficult to prove *mens rea* – some kind of evil intent and the system treats them as people who made mistakes negligently), or because there are too many complexities involved.

In the US the e-discovery market, presenting legal data in electronic format, is expected to grow from $976 million in 2012 to over $2.3 billion in 2016, representing an average annual growth rate of over 24% over the next four years. It is rather difficult to expect this to happen in Poland in the next 10 years.

[1] http://www.logicblox.com/presentations/sigmod11-tutorial-all.pdf

[2] http://www.highfleet.com

APPENDIX A

Example Use of the Extended Rules Method

In this appendix we present an example application of the extended rules method to set of two rules. The application is based on the example presented in Section 2.1.2.3. For the convenience of analysing current example we present data and rules from the previous one.

The extensional part of the knowledge base is presented in Table A.1 whereas intensional part contains rules (A.1) and (A.2).

Table A.1: An example extensional database.

hasChild	Parent	Child
	p_{11}	p_{12}
	p_{11}	p_{13}
	p_{12}	p_{14}
	p_{13}	p_{15}
	p_{21}	p_{22}
	p_{21}	p_{23}
	p_{22}	p_{24}
	p_{23}	p_{25}

$$hasChild(?x, ?y),\ hasChild(?x, ?z) \\ \rightarrow hasSiblings(?y, ?z) \tag{A.1}$$

$$hasChild(?x, ?z),\ hasSiblings(?x, ?y),\ hasChild(?y, ?w) \\ \rightarrow hasCousin(?z, ?w) \tag{A.2}$$

Applying the gsip algorithm to the set of basic rules (A.1) and (A.2) we obtain the following sets of rules which correspond to the steps of the gsip algorithm presented in Figure 4.3:

1. One rule which is generated by adding the goal connected with the head predicate:

$$hasChild(?x, ?y),\ hasChild(?x, ?z),\ hasSiblings(?y, ?z)^C$$

$$\rightarrow hasSiblings(?y, ?z) \tag{A.3}$$

$$hasChild(?x, ?z),\ hasSiblings(?x, ?y),\ hasChild(?y, ?w),\ hasCousin(?z, ?w)^C \rightarrow hasCousin(?z, ?w) \tag{A.4}$$

2. The set of rules with dependent predicates where all the variables from the head are unbound. In such case, the variables are replaced by the *nil* value:

$$hasSiblings(nil, nil)^C \rightarrow hasChild(nil, nil)^C \tag{A.5}$$
$$hasSiblings(nil, nil)^C \rightarrow hasChild(nil, nil)^C \tag{A.6}$$
$$hasCousin(nil, nil)^C \rightarrow hasChild(nil, nil)^C \tag{A.7}$$
$$hasCousin(nil, nil)^C \rightarrow hasChild(nil, nil)^C \tag{A.8}$$

3. The set of rules with dependent predicates and different binding patterns of the head predicate:

$$hasSiblings(?y, ?z)^C,\ ?z \neq nil \rightarrow hasChild(nil, ?z)^C \tag{A.9}$$
$$hasSiblings(?y, ?z)^C,\ ?y \neq nil \rightarrow hasChild(nil, ?y)^C \tag{A.10}$$
$$hasCousin(?z, ?w)^C,\ ?w \neq nil \rightarrow hasChild(nil, ?w)^C \tag{A.11}$$
$$hasCousin(?z, ?w)^C,\ ?z \neq nil \rightarrow hasChild(nil, ?z)^C \tag{A.12}$$

4. The set of rules with dependencies between proper and called predicates:

$$hasChild(?x, ?y),\ hasSiblings(?y, ?)^C \rightarrow hasChild(?x, nil)^C \tag{A.13}$$

$$hasChild(?x, ?z),\ hasSiblings(?, ?z)^C \rightarrow hasChild(?x, nil)^C \tag{A.14}$$

$$hasChild(?x, ?z),\ hasCousin(?z, ?)^C \rightarrow hasSiblings(?x, nil)^C \tag{A.15}$$

$$hasChild(?y, ?w),\ hasCousin(?, ?w)^C \rightarrow hasSiblings(nil, ?y)^C) \tag{A.16}$$

$$hasChild(?x, ?z),\ hasSiblings(?x, ?y),\ hasCousin(?z, ?w)^C$$

$$\rightarrow hasChild(?y, nil)^C \tag{A.17}$$

$$hasChild(?y, ?w),\ hasSiblings(?x, ?y),\ hasCousin(?z, ?w)^C \rightarrow hasChild(?x, nil)^C \tag{A.18}$$

As we can see the algorithm generates a number of excess rules (for example (A.6) and (A.8)). This can be easily corrected by comparing rules each other, and by removing repetitions. Currently approach does not remove such excess rules.

We now evaluate query (A.19), looking for all cousins of a person p_{14} (it is the same query as in Section 2.1.2.3 query (2.6). The query is evaluated according to the algorithm presented in Figure 4.4.

$$hasCousin(p_{14}, ?x) \rightarrow \tag{A.19}$$

Table A.2: Example evaluation of query $hasCousin(p_{14}, ?x)$ with extended rules RQA method Step 2.

No.	Goal	Used rule	Added fact	Inferred fact
0	$hasCousin(p_{14}, nil)^C$	-	-	-
1	$hasCousin(p_{14}, ?x)^C$	(A.12)	-	$hasChild(nil, p_{14})^C$
2	$hasChild(nil, p_{14})^C$	DB	$hasChild(p_{12}, p_{14})$	-
3	$hasCousin(p_{14}, nil)^C$	(A.15)	-	$hasSiblings(p_{12}, nil)^C$
4	$hasSiblings(p_{12}, nil)^C$	(A.10)	-	$hasChild(nil, p_{12})^C$
5	$hasChild(nil, p_{12})^C$	DB	$hasChild(p_{11}, p_{12})$	-
6	$hasSiblings(p_{12}, nil)^C$	(A.13)	-	$hasChild(p_{11}, nil)^C$
7	$hasChild(p_{11}, nil)^C$	DB	$hasChild(p_{11}, p_{13})$	-
8	$hasSiblings(p_{12}, nil)^C$	(A.3)	-	$hasSiblings(p_{12}, p_{13})$
9	$hasCousin(p_{14}, ?x)^C$	(A.17)	-	$hasChild(p_{13}, nil)^C$
10	$hasChild(p_{13}, nil)^C$	DB	$hasChild(p_{13}, p_{15})$	-
11	$hasCousin(p_{14}, ?x)^C$	(A.4)	-	$hasCousin(p_{14}, p_{15})$
12	-	(A.20)	-	$saveResults(p_{15})$

In Step 1 the *QUERYRULE* is created of the following form:

$$hasCousin(p_{14}, ?x) \rightarrow saveResults(?x) \tag{A.20}$$

where $saveResults$ predicate is used to save value of variables each time when *QUERYRULE* fires.

In Step 2 the following reasoning is performed. We assume that “Added fact” means that the fact was added to the working memory and comes from the relational database, and “Inferred fact” means that the fact is a result of the reasoning process. In case when fact is added, it is denoted in column "Used rule" as "DB". Point 1 of Step 2 is represented as No. 0 in Table A.2 which results in addition of called fact $hasCousin(p_{14}, nil)^C$ in the reasoning engine’s working memory.

In Step 3 the result is returned and *QUERYRULE* is removed.

It is noticeable that the number of steps in the presented method is larger that for backward or forward chaining, nevertheless the execution time is shorter compared to these two methods. The reason for this issue is that extended rules are generated to pass only one binding of a variable from the body to the head of a rule.

Bibliography

[Abiteboul 1995] Serge Abiteboul, Richard Hull and Victor Vianu. Foundations of databases. Addison-Wesley, 1995. (Cited on pages 14, 16, 18, 20, 22 and 26.)

[ACFE 2008] Association of Certified Fraud Examiners ACFE. *Report to the Nation on Occupational Fraud and Abuse*. http://www.acfe.com/uploadedFiles/ACFE_Website/Content/documents/2008-rttn.pdf, 2008. Accessed: 04/04/2013. (Cited on page 42.)

[ACFE 2012] Association of Certified Fraud Examiners ACFE. *Report to the Nation on Occupational Fraud and Abuse*. http://www.acfe.com/uploadedFiles/ACFE_Website/Content/rttn/2012-report-to-nations.pdf, 2012. Accessed: 04/04/2013. (Cited on page 43.)

[Albrecht 2008] Chad Albrecht, Mary-Jo Kranacher and Steve Albrecht. *Asset Misappropriation Research White Paper for the Institute for Fraud Prevention*. http://www.theifp.org/research-grants/IFP-Whitepaper-5.pdf, 2008. Accessed: 04/04/2013. (Cited on page 42.)

[Albrecht 2011] W.S. Albrecht, C.C. Albrecht, C.O. Albrecht and M.F. Zimbelman. Fraud examination. Cengage Learning, 2011. (Cited on pages 42, 43 and 44.)

[Aleven 2003] Vincent Aleven. *Using background knowledge in case-based legal reasoning: a computational model and an intelligent learning environment*. Artificial Intelligence, vol. 150, pages 183–237, 2003. (Cited on pages 54 and 76.)

[Alviano 2012] Mario Alviano, Nicola Leone, Marco Manna, Giorgio Terracina and Pierfrancesco Veltri. *Magic-Sets for Datalog with Existential Quantifiers*. In Pablo Barceló and Reinhard Pichler, editors, Datalog in Academia and Industry, volume 7494 of *Lecture Notes in Computer Science*, pages 31–43. Springer Berlin Heidelberg, 2012. (Cited on page 125.)

[Baader 1991] Franz Baader and Philipp Hanschke. *A scheme for integrating concrete domains into concept languages*. pages 452–457, 1991. (Cited on page 31.)

[Baader 1999] Franz Baader and Ulrike Sattler. *Expressive number restrictions in Description Logics*. Journal of Logic and Computation, vol. 9, pages 319–350, 1999. (Cited on page 35.)

[Baader 2003] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi and Peter F. Patel-Schneider, editors. The description logic handbook: theory, implementation, and applications. Cambridge University Press, New York, NY, USA, 2003. (Cited on pages 30 and 35.)

[Baader 2008] Franz Baader, Sebastian Brandt and Carsten Lutz. *Pushing the EL Envelope Further*. In Kendall Clark and Peter F. Patel-Schneider, editors, In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions, 2008. (Cited on page 40.)

[Bak 2008] Jaroslaw Bak and Czeslaw Jedrzejek. *Querying relational databases using ontology, rules and Jess reasoning engine*. Studia z Automatyki i Informatyki, vol. T. 33, pages 24–44, 2008. (Cited on page 5.)

[Bak 2009] Jaroslaw Bak, Czeslaw Jedrzejek and Maciej Falkowski. *Usage of the Jess Engine, Rules and Ontology to Query a Relational Database*. In Proceedings of the 2009 International Symposium on Rule Interchange and Applications, RuleML '09, pages 216–230, Berlin, Heidelberg, 2009. Springer-Verlag. (Cited on pages 5, 78, 98 and 104.)

[Bak 2010a] Jaroslaw Bak, Maciej Falkowski and Czeslaw Jedrzejek. *Application of the SDL Library to Reveal Legal Sanctions for Crime Perpetrators in Selected Economic Crimes: Fraudulent Disbursement and Money Laundering*. In Monica Palmirani, M. Omair Shafiq, Enrico Francesconi and Fabio Vitali, editors, Proceedings of the RuleML-2010 Challenge, at the 4th International Web Rule Symposium, Washington, DC, USA, October, 21-23, 2010, volume 649 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010. (Cited on pages 6 and 123.)

[Bak 2010b] Jaroslaw Bak and Czeslaw Jedrzejek. *Application of an ontology-based model to a selected fraudulent disbursement economic crime*. In Proceedings of the 2009 international conference on AI approaches to the complexity of legal systems: complex systems, the semantic web, ontologies, argumentation, and dialogue, AICOL-I/IVR-XXIV'09, pages 113–132, Berlin, Heidelberg, 2010. Springer-Verlag. (Cited on page 5.)

[Bak 2010c] Jaroslaw Bak, Czeslaw Jedrzejek and Maciej Falkowski. *Application of an ontology-based and rule-based model to selected economic crimes: fraudulent disbursement and money laundering*. In Proceedings of the 2010

international conference on Semantic web rules, RuleML'10, pages 210–224, Berlin, Heidelberg, 2010. Springer-Verlag. (Cited on page 5.)

[Bak 2011a] Jaroslaw Bak, Grażyna Brzykcy and Czeslaw Jedrzejek. *Extended rules in knowledge-based data access*. In Proceedings of the 5th international conference on Rule-based modeling and computing on the semantic web, RuleML'11, pages 112–127, Berlin, Heidelberg, 2011. Springer-Verlag. (Cited on pages 5, 78, 98, 104 and 105.)

[Bak 2011b] Jaroslaw Bak, Maciej Falkowski and Czeslaw Jedrzejek. *The SDL Library: Querying a Relational Database with an Ontology, Rules and the Jess Engine*. In Stefano Bragaglia, Carlos Viegas Damásio, Marco Montali, Alun Preece, Charles Petrie, Mark Proctor and Umberto Straccia, editors, Proceedings of the 5th International RuleML2011@BRF Challenge, co-located with the 5th International Rule Symposium, Fort Lauderdale, Florida, USA, November 3-5, 2011, volume 799 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011. (Cited on pages 6 and 123.)

[Bak 2013] Jaroslaw Bak, Jolanta Cybulka and Czeslaw Jedrzejek. *Ontological Modeling of a Class of Linked Economic Crimes*. T. Computational Collective Intelligence, vol. 9, pages 98–123, 2013. (Cited on pages 5 and 51.)

[Bancilhon 1986a] Francois Bancilhon, David Maier, Yehoshua Sagiv and Jeffrey D Ullman. *Magic sets and other strange ways to implement logic programs (extended abstract)*. In Proceedings of the fifth ACM SIGACT-SIGMOD symposium on Principles of database systems, PODS '86, pages 1–15, New York, NY, USA, 1986. ACM. (Cited on pages 28, 87 and 98.)

[Bancilhon 1986b] Francois Bancilhon and Raghu Ramakrishnan. *An amateur's introduction to recursive query processing strategies*. SIGMOD Rec., vol. 15, no. 2, pages 16–52, June 1986. (Cited on pages 26 and 98.)

[Beckett 2004] Dave Beckett. *RDF/XML Syntax Specification (Revised)*. W3C recommendation, W3C, 2004. (Cited on page 82.)

[Beeri 1987] Catriel Beeri and Raghu Ramakrishnan. *On the Power of Magic*. In Journal of Logic Programming, pages 269–283, 1987. (Cited on pages 5, 26, 28, 29 and 98.)

[Behrend 2011] Andreas Behrend. *A Uniform Fixpoint Approach to the Implementation of Inference Methods for Deductive Databases*. CoRR, vol. abs/1108.5451, 2011. (Cited on page 125.)

[Bezzazi 2007] El-Hassan Bezzazi. *Building an Ontology That Helps Identify Criminal Law Articles That Apply to a Cybercrime Case.* In Joaquim Filipe, Boris Shishkov and Markus Helfert, editors, ICSOFT (PL/DPS/KE/MUSE), pages 179–185. INSTICC Press, 2007. (Cited on pages 3, 77 and 78.)

[Biasiotti 2008] Mariangela Biasiotti, Enrico Francesconi, Monica Palmirani, Giovanni Sartor and Fabio Vitali. *Legal Informatics and Management of Legislative Documents*, 2008. (Cited on pages 2 and 75.)

[Bizer 2004] Chris Bizer and Andy Seaborne. *D2RQ treating non-RDF databases as virtual RDF graphs*. In ISWC, 2004. (Cited on page 99.)

[Bizer 2006] Christian Bizer and Richard Cyganiak. *D2R Server – Publishing Relational Databases on the Semantic Web*. Poster at the 5th International Semantic Web Conference (ISWC2006), 2006. (Cited on page 99.)

[Brass 2010] Stefan Brass. *Implementation Alternatives for Bottom-Up Evaluation*. In Manuel Hermenegildo and Torsten Schaub, editors, Technical Communications of the 26th International Conference on Logic Programming, volume 7 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44–53, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. (Cited on pages 29 and 98.)

[Breuker 2009] Joost Breuker. *Dreams, awakenings and paradoxes of ontologies, invited talk presentation, 3rd Workshop on Legal Ontologies and Artificial Intelligence Techniques*. `http://ontobra.comp.ime.eb.br/apresentacoes/keynote-ontobra-2009.ppt`, 2009. Accessed: 04/04/2013. (Cited on pages 2, 74, 75, 76 and 79.)

[Bry 1990] François Bry. *Query evaluation in recursive databases: bottom-up and top-down reconciled*. Data Knowl. Eng., vol. 5, no. 4, pages 289–312, October 1990. (Cited on pages 26 and 29.)

[Bry 2007] François Bry, Norbert Eisinger, Thomas Eiter, Tim Furche, Georg Gottlob, Clemens Ley, Benedikt Linse, Reinhard Pichler and Fang Wei. *Foundations of Rule-Based Query Answering*. In Grigoris Antoniou, Uwe Aßmann, Cristina Baroglio, Stefan Decker, Nicola Henze, Paula-Lavinia Patranjan and Robert Tolksdorf, editors, Reasoning Web, volume 4636 of *Lecture Notes in Computer Science*, pages 1–153. Springer, 2007. (Cited on pages 15, 21, 25, 26 and 29.)

[Calvanese 2006] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini and Riccardo Rosati. *Data Complexity of Query Answering in Description Logics*. In Doherty et al. [Doherty 2006], pages 260–270. (Cited on page 38.)

[Calvanese 2013] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini and Riccardo Rosati. *Data complexity of query answering in description logics*. Artif. Intell., vol. 195, pages 335–360, 2013. (Cited on page 94.)

[Casanovas 2009] Pompeu Casanovas, Núria Casellas and Joan-Josep Vallbé. *An Ontology-Based Decision Support System for Judges*. In Proceedings of the 2009 conference on Law, Ontologies and the Semantic Web: Channelling the Legal Information Flood, pages 165–175, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press. (Cited on page 76.)

[Casellas 2008] N. Casellas. *Modelling Legal Knowledge through Ontologies. OPJK: the Ontology of Professional Judicial Knowledge*. PhD thesis, Institute of Law and Technology, Autonomous University of Barcelona, 2008. (Cited on pages 2, 75 and 76.)

[Community 2012] JBoss Community. *Drools - The Business Logic integration Platform*. http://www.jboss.org/drools/, 2012. Accessed: 04/04/2013. (Cited on page 4.)

[Consortium 2006] WWW Consortium. *OWL 1.1 Web Ontology Language*. http://www.w3.org/Submission/owl11-overview/, 2006. Accessed: 04/04/2013. (Cited on pages 31 and 103.)

[Consortium 2008] WWW Consortium. *SPARQL Query Language for RDF*. http://www.w3.org/TR/rdf-sparql-query/, 2008. Accessed: 04/04/2013. (Cited on page 95.)

[Consortium 2012] WWW Consortium. *OWL 2 Web Ontology Language Profiles (Second Edition)*. http://www.w3.org/TR/owl2-profiles/, 2012. Accessed: 04/04/2013. (Cited on page 31.)

[Corcho 2003] Oscar Corcho, Mariano Fernández-López and Asunción Gómez-Pérez. *Methodologies, tools and languages for building ontologies: where is their meeting point?* Data Knowl. Eng., vol. 46, no. 1, pages 41–64, July 2003. (Cited on page 76.)

[Cybulka 2008] Jolanta Cybulka, Czeslaw Jedrzejek and Jacek Martinek. *Police Investigation Management System Based on the Workflow Technology*. In

Enrico Francesconi, Giovanni Sartor and Daniela Tiscornia, editors, JURIX, volume 189 of *Frontiers in Artificial Intelligence and Applications*, pages 150–159. IOS Press, 2008. (Cited on page 50.)

[Cybulka 2009] Jolanta Cybulka. *Applying the c.DnS Design Pattern to Obtain an Ontology for Investigation Management System*. In Proceedings of the 1st International Conference on Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems, ICCCI '09, pages 516–527, Berlin, Heidelberg, 2009. Springer-Verlag. (Cited on pages 51 and 77.)

[Cybulka 2010a] Jolanta Cybulka. *Fuel Crime Conceptualization through Specialization of Ontology for Investigation Management System*. In Ngoc-Thanh Nguyen and Ryszard Kowalczyk, editors, Transactions on Computational Collective Intelligence II, volume 6450 of *Lecture Notes in Computer Science*, pages 123–146. Springer Berlin Heidelberg, 2010. (Cited on page 78.)

[Cybulka 2010b] Jolanta Cybulka. *Metoda tworzenia ontologii na potrzeby systemu wspomagającego prace dochodzeniowo-śledcze w sprawach przestępstw gospodarczych*. In Jacek Gołaczyński, editor, Informatyzacja postępowania sądowego i administracji publicznej, pages 359–370. Wydawnictwo C. H. Beck, 2010. (Cited on page 51.)

[Dantsin 2001] Evgeny Dantsin, Thomas Eiter, Georg Gottlob and Andrei Voronkov. *Complexity and expressive power of logic programming*. ACM Comput. Surv., vol. 33, no. 3, pages 374–425, September 2001. (Cited on pages 18 and 94.)

[Darlington 2008] Mansur J. Darlington and Steve J. Culley. *Investigating ontology development for engineering design support*. Advanced Engineering Informatics, vol. 22, no. 1, pages 112–134, 2008. (Cited on page 76.)

[Dietrich 1989] S.W. Dietrich. *A performance comparison of top-down recursive query evaluation strategies on Datalog benchmarks*. In System Sciences, 1989. Vol.II: Software Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on, volume 2, pages 621 –629 vol.2, jan 1989. (Cited on page 26.)

[Doherty 2006] Patrick Doherty, John Mylopoulos and Christopher A. Welty, editors. Proceedings, tenth international conference on principles of knowledge representation and reasoning, lake district of the united kingdom, june 2-5, 2006. AAAI Press, 2006. (Cited on pages 139 and 150.)

[Donini 1998] FrancescoM. Donini, Maurizio Lenzerini, Daniele Nardi and Andrea Schaerf. *AL-log: Integrating Datalog and Description Logics*. Journal of Intelligent Information Systems, vol. 10, pages 227–252, 1998. (Cited on page 39.)

[Drabent 2009] Włodzimierz Drabent, Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner, Thomas Lukasiewicz and Jan Małuszyński. *Hybrid Reasoning with Rules and Ontologies*. In François Bry and Jan Małuszyński, editors, Semantic Techniques for the Web, volume 5500 of *Lecture Notes in Computer Science*, pages 1–49. Springer Berlin Heidelberg, 2009. (Cited on page 40.)

[Eiter 2008a] Thomas Eiter, Georg Gottlob, Magdalena Ortiz and Mantas Šimkus. *Query Answering in the Description Logic Horn-$\mathcal{SHIQ}$*. In Proceedings of the 11th European conference on Logics in Artificial Intelligence, JELIA '08, pages 166–179, Berlin, Heidelberg, 2008. Springer-Verlag. (Cited on pages 38, 39 and 94.)

[Eiter 2008b] Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer and Hans Tompits. *Combining answer set programming with description logics for the Semantic Web*. Artificial Intelligence, vol. 172, no. 12–13, pages 1495 – 1539, 2008. (Cited on page 40.)

[Falkowski 2009] Maciej Falkowski and Czeslaw Jedrzejek. *An efficient SQL-based querying method to RDF schemata*. Control and Cybernetics, vol. 38, no. 1, pages 193–213, 2009. (Cited on page 95.)

[FATF 2012] Financial Action Task Force FATF. *Money Laundering*. http://www.fatf-gafi.org/pages/faq/moneylaundering/, 2012. Accessed: 04/04/2013. (Cited on pages 43 and 47.)

[Fodor 2011] Paul Fodor and Michael Kifer. *Transaction Logic with Defaults and Argumentation Theories*. In John P. Gallagher and Michael Gelfond, editors, ICLP (Technical Communications), volume 11 of *LIPIcs*, pages 162–174. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. (Cited on page 74.)

[Forgy 1982] C. L. Forgy. *Rete: A fast algorithm for the many pattern/many object pattern match problem*. Artificial Intelligence, vol. 19, pages 17–37, 1982. (Cited on pages 4, 5, 24 and 82.)

[Frontline 2013] PBS Frontline. *The Untouchables*. http://www.abc.net.au/4corners/stories/2013/03/18/3715426.htm, 2013. Accessed: 04/04/2013. (Cited on page 129.)

[Gallaire 1978] Hervé Gallaire and Jack Minker, editors. Logic and data bases, symposium on logic and data bases, centre d'études et de recherches de toulouse, 1977, Advances in Data Base Theory. Plemum Press, 1978. (Cited on page 13.)

[Gallaire 1984] Herve Gallaire, Jack Minker and Jean-Marie Nicolas. *Logic and Databases: A Deductive Approach*. ACM Comput. Surv., vol. 16, no. 2, pages 153–185, June 1984. (Cited on page 125.)

[Gallier 1987] Jean H. Gallier. Logic for computer science: Foundations of automatic theorem proving. Wiley, 1987. (Cited on page 21.)

[Gangemi 2007] Aldo Gangemi, Jos Lehmann, Valentina Presutti, Malvina Nissim and Carola Catenacci. *C-ODO: an OWL Meta-model for Collaborative Ontology Design*. In Natalya Fridman Noy, Harith Alani, Gerd Stumme, Peter Mika, York Sure and Denny Vrandecic, editors, CKC, volume 273 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007. (Cited on page 52.)

[Gangemi 2008] Aldo Gangemi. *Norms and plans as unification criteria for social collectives*. Autonomous Agents and Multi-Agent Systems, vol. 17, no. 1, pages 70–112, August 2008. (Cited on page 77.)

[Goczyła 2011] Krzysztof Goczyła. Ontologie w systemach informatycznych. Akademicka Oficyna Wydawnicza EXIT, 2011, Warszawa, Poland, 2011. (Cited on page 50.)

[Gómez-Pérez 2008] A. Gómez-Pérez, C. Suárez de Figueroa Baonza M. and B. Villazón. *NeOn Methodology for Building Ontology Networks: Ontology Specification, excerpt from NeOn Deliverable D5.4.1*. http://www.neon-project.org, 2008. (Cited on page 52.)

[Governatori 2012] Guido Governatori, Antonino Rotolo and Erica Calardo. *Possible World Semantics for Defeasible Deontic Logic*. In Thomas Ågotnes, Jan Broersen and Dag Elgesem, editors, DEON, volume 7393 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2012. (Cited on page 74.)

[Grau 2006] Bernardo Cuenca Grau. *OWL 1.1 Web Ontology Language Tractable Fragments*. http://www.w3.org/Submission/owl11-tractable/, 2006. Accessed: 04/04/2013. (Cited on pages 81 and 108.)

[Grosof 2003] Benjamin N. Grosof, Ian Horrocks, Raphael Volz and Stefan Decker. *Description logic programs: combining logic programs with description logic*. In Proceedings of the 12th international conference on

World Wide Web, WWW '03, pages 48–57, New York, NY, USA, 2003. ACM. (Cited on pages 36 and 108.)

[Haarslev 2000] Volker Haarslev and Ralf Moeller. *Expressive ABox Reasoning with Number Restrictions, Role Hierarchies, and Transitively Closed Roles*. Rapport technique, Hamburg, Germany, Germany, 2000. (Cited on page 35.)

[Hellerstein 2010] Joseph M. Hellerstein. *The declarative imperative: experiences and conjectures in distributed logic*. SIGMOD Rec., vol. 39, no. 1, pages 5–19, September 2010. (Cited on page 125.)

[Hill 2003] Ernest Friedman Hill. Jess in action: Java rule-based systems. Manning Publications Co., Greenwich, CT, USA, 2003. (Cited on pages 4, 22, 24, 82, 86 and 105.)

[Hitzler 2005] Pascal Hitzler, Rudi Studer and York Sure. *Description Logic Programs: A Practical Choice For the Modelling of Ontologies*. In 1st Workshop on Formal Ontologies Meet Industry, FOMI'05, Verona, Italy, June 2005, März 2005. (Cited on page 36.)

[Hitzler 2009a] Pascal Hitzler and Bijan Parsia. *Ontologies and Rules*. In Steffen Staab and Rudi Studer, editors, Handbook on Ontologies, International Handbooks on Information Systems, pages 111–132. Springer Berlin Heidelberg, 2009. (Cited on page 38.)

[Hitzler 2009b] Pascal Hitzler, Rudolf Sebastian and Markus Krötzsch. Foundations of semantic web technologies. Chapman & Hall/CRC, 2009. (Cited on pages 14 and 16.)

[Hoekstra 2007] Rinke Hoekstra, Joost Breuker, Marcello Di Bello and Er Boer. *The LKIF Core ontology of basic legal concepts*. In In Pompeu Casanovas, Maria Angela Biasiotti, Enrico Francesconi, and Maria Teresa Sagri, editors, Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT 2007, 2007. (Cited on page 75.)

[Horridge 2006] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector and Hai H Wang. *The Manchester OWL Syntax*. In In Proc. of the 2006 OWL Experiences and Directions Workshop (OWL-ED2006, 2006. (Cited on page 82.)

[Horridge 2009] Matthew Horridge and Sean Bechhofer. *The OWL API: A Java API for Working with OWL 2 Ontologies*. In OWLED, 2009. (Cited on page 111.)

[Horridge 2011] Matthew Horridge and Sean Bechhofer. *The OWL API: A Java API for OWL ontologies*. Semant. web, vol. 2, no. 1, pages 11–21, January 2011. (Cited on page 111.)

[Horrocks 1999] Ian Horrocks and Ulrike Sattler. *A Description Logic with Transitive and Inverse Roles and Role Hierarchies*. JOURNAL OF LOGIC AND COMPUTATION, vol. 9, no. 3, pages 385–410, 1999. (Cited on page 35.)

[Horrocks 2003] Ian Horrocks, Peter Patel-Schneider and Frank van Harmelen. *From $\mathcal{SHIQ}$ and RDF to OWL: The Making of a Web Ontology Language*. J. of Web Semantics, vol. 1, no. 1, pages 7–26, 2003. (Cited on page 31.)

[Horrocks 2004a] Ian Horrocks and Peter F. Patel-Schneider. *A proposal for an owl rules language*. In Proceedings of the 13th international conference on World Wide Web, WWW '04, pages 723–731, New York, NY, USA, 2004. ACM. (Cited on page 36.)

[Horrocks 2004b] Ian Horrocks, Peter F. Patel-schneider, Harold Boley, Said Tabet, Benjamin Grosof and Mike Dean. *SWRL: A semantic web rule language combining OWL and RuleML*. 2004. Accessed: 04/04/2013. (Cited on pages 81 and 103.)

[Horrocks 2005] Ian Horrocks, Peter F. Patel-schneider, Sean Bechhofer and Dmitry Tsarkov. *OWL Rules: A Proposal and Prototype Implementation*. Journal of Web Semantics, vol. 3, pages 23–40, 2005. (Cited on page 36.)

[Huang 2011] Shan Shan Huang, Todd Jeffrey Green and Boon Thau Loo. *Datalog and emerging applications: an interactive tutorial*. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, SIGMOD '11, pages 1213–1216, New York, NY, USA, 2011. ACM. (Cited on page 125.)

[Hustadt 2004] Ullrich Hustadt. *Reducing $\mathcal{SHIQ}$ - Description Logic to Disjunctive Datalog Programs*. pages 152–162, 2004. (Cited on page 35.)

[Hustadt 2005] Ullrich Hustadt, Boris Motik and Ulrike Sattler. *Data Complexity of Reasoning in Very Expressive Description Logics*. In IN PROC. IJCAI 2005, pages 466–471. Professional Book Center, 2005. (Cited on pages 5 and 37.)

[Hustadt 2007] Ullrich Hustadt, Boris Motik and Ulrike Sattler. *Reasoning in Description Logics by a Reduction to Disjunctive Datalog*. J. Autom. Reason., vol. 39, no. 3, pages 351–384, October 2007. (Cited on pages 35 and 38.)

[Jedrzejek 2011a] Czeslaw Jedrzejek, Jaroslaw Bak, Maciej Falkowski, Jolanta Cybulka and Maciej Nowak. *On the Detection and Analysis of VAT Carousel Crime*. In Katie Atkinson, editor, JURIX, volume 235 of *Frontiers in Artificial Intelligence and Applications*, pages 130–134. IOS Press, 2011. (Cited on page 78.)

[Jedrzejek 2011b] Czeslaw Jedrzejek, Jolanta Cybulka and Jaroslaw Bak. *Towards ontology of fraudulent disbursement*. In Proceedings of the 5th KES international conference on Agent and multi-agent systems: technologies and applications, KES-AMSTA'11, pages 301–310, Berlin, Heidelberg, 2011. Springer-Verlag. (Cited on page 5.)

[JMLSG 2012] Joint Money Laundering Steering Group JMLSG. *Money laundering/terrorist financing activities*. http://www.jmlsg.org.uk/other-helpful-material/article/money-laundering-terrorist-financing-activities, 2012. Accessed: 04/04/2013. (Cited on page 43.)

[KAON2 2012] KAON2. *KAON2 reasoning engine*. http://kaon2.semanticweb.org/, 2012. Accessed: 04/04/2013. (Cited on page 52.)

[Kelly 2013] Jeff Kelly. *Big Data Vendor Revenue and Market Forecast 2012-2017*. http://wikibon.org/wiki/v/Big_Data_Vendor_Revenue_and_Market_Forecast_2012-2017, 2013. Accessed: 04/04/2013. (Cited on page 1.)

[Kerremans 2005] Koen Kerremans and Gang Zhao. *Topical ontology of VAT: FF POIROT Deliverable 2.3*. http://starlab.vub.ac.be/research/projects/poirot/Publications/ffpoirot.d2.3.TopicalOntologyVAT-v1.1.pdf, 2005. Accessed: 04/04/2013. (Cited on pages 2 and 75.)

[Kingston 2005] John Kingston, Burkhard Schafer and Wim Vandenberghe. *No Model Behaviour: Ontologies for Fraud Detection*. In V.Richard Benjamins, Pompeu Casanovas, Joost Breuker and Aldo Gangemi, editors, Law and the Semantic Web, volume 3369 of *Lecture Notes in Computer Science*, pages 233–247. Springer Berlin Heidelberg, 2005. (Cited on page 47.)

[Knorr 2007] Matthias Knorr, José Júlio Alferes and Pascal Hitzler. *A well-founded semantics for hybrid MKNF knowledge bases*. In In Proceedings ASP-2007, pages 115–131, 2007. (Cited on page 40.)

[Kowalski 1974] Robert A. Kowalski. *Predicate Logic as Programming Language*. In IFIP Congress 74, pages 569–574, Stockholm, 1974. North-Holland. (Cited on page 19.)

[Kremen 2012] Petr Kremen and Zdenek Kouba. *Ontology-Driven Information System Design*. IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 42, no. 3, pages 334–344, 2012. (Cited on page 1.)

[Krisnadhi 2011] Adila Krisnadhi, Frederick Maier and Pascal Hitzler. *OWL and Rules*. In Reasoning Web, pages 382–415, 2011. (Cited on page 36.)

[Krötzsch 2006] Markus Krötzsch, Pascal Hitzler, Denny Vrandecic and Michael Sintek. *How to reason with OWL in a logic programming system*. In Thomas Eiter, Enrico Franconi, Ralph Hodgson and Susie Stephens, editors, RuleML, pages 17–28. IEEE Computer Society, 2006. (Cited on page 38.)

[Krötzsch 2007] Markus Krötzsch, Sebastian Rudolph and Pascal Hitzler. *Complexity Boundaries for Horn Description Logics*. In Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07), pages 452–457. AAAI Press, 2007. (Cited on page 94.)

[Krötzsch 2008a] Markus Krötzsch, Sebastian Rudolph and Pascal Hitzler. *Description Logic Rules*. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis and Nikos Avouris, editors, Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08), pages 80–84. IOS Press, 2008. (Cited on page 37.)

[Krötzsch 2008b] Markus Krötzsch, Sebastian Rudolph and Pascal Hitzler. *ELP: Tractable Rules for OWL 2*. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin and Krishnaprasad Thirunarayan, editors, Proceedings of the 7th International Semantic Web Conference (ISWC'08), volume 5318 of *LNCS*, pages 649–664. Springer, 2008. (Cited on pages 37 and 40.)

[Krötzsch 2010] Markus Krötzsch. Description logic rules, volume 008 of *Studies on the Semantic Web*. IOS Press/AKA, 2010. (Cited on pages 17 and 37.)

[Levy 1996] Alon Y. Levy and Marie-Christine Rousset. *CARIN: A Representation Language Combining Horn Rules and Description Logics*. In Wolfgang Wahlster, editor, ECAI, pages 323–327. John Wiley and Sons, Chichester, 1996. (Cited on page 39.)

[Liang 2009a] Senlin Liang, Paul Fodor, Hui Wan and Michael Kifer. *OpenRuleBench: an analysis of the performance of rule engines*. In Proceedings of the 18th international conference on World wide web, WWW '09, pages 601–610, New York, NY, USA, 2009. ACM. (Cited on pages 3, 26 and 98.)

[Liang 2009b] Senlin Liang, Paul Fodor, Hui Wan and Michael Kifer. *OpenRuleBench: Detailed Report*. http://semwebcentral.org/docman/view.php/158/69/report.pdf, 2009. Accessed: 04/04/2013. (Cited on page 101.)

[Lifschitz 1991] Vladimir Lifschitz. *Nonmonotonic databases and epistemic queries*. In Proceedings of the 12th international joint conference on Artificial intelligence - Volume 1, IJCAI'91, pages 381–386, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. (Cited on page 40.)

[Lifschitz 1996] Vladimir Lifschitz. *Foundations of Logic Programming*. pages 23–37, 1996. (Cited on page 34.)

[Ligeza 2006] Antoni Ligeza. Logical foundations for rule-based systems, 2nd ed., volume 11 of *Studies in Computational Intelligence*. Springer, 2006. (Cited on page 9.)

[Liu 2009] Yanhong A. Liu and Scott D. Stoller. *From datalog rules to efficient programs with time and space guarantees*. ACM Trans. Program. Lang. Syst., vol. 31, no. 6, pages 21:1–21:38, August 2009. (Cited on page 125.)

[Lloyd 1984] J. W. Lloyd. Foundations of logic programming. Springer-Verlag New York, Inc., New York, NY, USA, 1984. (Cited on pages 9, 18, 23, 56 and 82.)

[Lukácsy 2009] Gergely Lukácsy and Péter Szeredi. *Efficient Description Logic Reasoning in Prolog: The DLog system*. CoRR, vol. abs/0904.0578, 2009. (Cited on pages 98 and 100.)

[Martinek 2008] Jacek Martinek. *Hydra case formal structural description*. Rapport technique, Poznan University of Technology, Poznan, Poland, 2008. (Cited on pages 48 and 49.)

[Meditskos 2008] Georgios Meditskos and Nick Bassiliades. *A Rule-Based Object-Oriented OWL Reasoner*. IEEE Trans. on Knowl. and Data Eng., vol. 20, no. 3, pages 397–410, March 2008. (Cited on page 107.)

[Mei 2005] Jing Mei, Elena Paslaru Bontas and Zuoquan Lin. *OWL2Jess: A Transformational Implementation of the OWL Semantics*. In In Proceedings of International Workshops on ISPA, LNCS 3759, pages 599–608, 2005. (Cited on pages 99 and 107.)

[Mommers 2002] Laurens Mommers. *Applied legal epistemology. Building a knowledge-based ontology of the legal domain*. PhD thesis, Leiden University, Leiden, 2002. (Cited on page 75.)

[Motik 2004] Boris Motik, Ulrike Sattler and Rudi Studer. *Query Answering for OWL-DL with Rules*. In Journal of Web Semantics, pages 549–563. Springer, 2004. (Cited on pages 34 and 37.)

[Motik 2006a] Boris Motik. *Reasoning in description logics using resolution and deductive databases.* PhD thesis, Karlsruhe Institute of Technology, 2006. http://d-nb.info/1001691709. (Cited on pages 37 and 38.)

[Motik 2006b] Boris Motik and Riccardo Rosati. *Closing semantic web ontologies*. Rapport technique, 2006. (Cited on page 40.)

[Motik 2007] Boris Motik and Riccardo Rosati. *A faithful integration of description logics with logic programming*. In Proceedings of the 20th international joint conference on Artifical intelligence, IJCAI'07, pages 477–482, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. (Cited on page 40.)

[Motik 2008] Boris Motik and Ian Horrocks. *OWL Datatypes: Design and Implementation*. pages 307–322, 2008. (Cited on page 31.)

[Nalepa 2008] Grzegorz J. Nalepa and Antoni Ligeza. *XTT+ Rule Design Using the ALSV(FD)*. In Adrian Giurca, Anastasia Analyti and Gerd Wagner, editors, RuleApps, volume 428 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. (Cited on page 40.)

[Nalepa 2009] Grzegorz J. Nalepa. Languages and tools for rule modeling. Handbook of Research on Emerging Rule-based Languages and Technologies: Open Solutions and Approaches. Igi Global, 2009. (Cited on page 22.)

[Nalepa 2010] Grzegorz J. Nalepa and WeronikaT. Furmańska. *Integration Proposal for Description Logic and Attributive Logic – Towards Semantic Web Rules*. In NgocThanh Nguyen and Ryszard Kowalczyk, editors, Transactions on Computational Collective Intelligence II, volume 6450 of *Lecture Notes in Computer Science*, pages 1–23. Springer Berlin Heidelberg, 2010. (Cited on page 40.)

[Nalepa 2011] Grzegorz J. Nalepa. Semantic knowledge engineering. A rule-based approach. Wydawnictwa AGH, Kraków, 2011. (Cited on page 30.)

[Nilsson 1995] Ulf Nilsson and Jan Maluszynski. Logic, programming, and prolog. John Wiley & Sons, Inc., New York, NY, USA, 2nd édition, 1995. (Cited on pages 27 and 28.)

[Nowak 2012] Maciej Nowak, Jaroslaw Bak and Czeslaw Jedrzejek. *Graph-based Rule Editor*. In Hassan Aït-Kaci, Yuh-Jong Hu, Grzegorz J. Nalepa, Monica Palmirani and Dumitru Roman, editors, RuleML2012@ECAI Challenge and Doctoral Consortium at the 6th International Symposium on Rules, Montpellier, France, August 27th-29th, 2012, volume 874 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012. (Cited on page 128.)

[O'Connor 2007a] Martin J. O'Connor, Ravi D. Shankar, Samson W. Tu, Csongor Nyulas, Amar K. Das and Mark A. Musen. *Efficiently Querying Relational Databases Using OWL and SWRL*. In RR, pages 361–363, 2007. (Cited on page 99.)

[O'Connor 2007b] Martin J. O'Connor, Samson W. Tu, Csongor Nyulas, Amar K. Das and Mark A. Musen. *Querying the Semantic Web with SWRL*. In RuleML, pages 155–159, 2007. (Cited on page 99.)

[Parsia 2012] Clark & Parsia. *Pellet: OWL 2 Reasoner for Java*. `http://clarkparsia.com/pellet/`, 2012. Accessed: 04/04/2013. (Cited on page 52.)

[Patel-Schneider 2007] Peter F. Patel-Schneider and Ian Horrocks. *A comparison of two modelling paradigms in the Semantic Web*. Web Semant., vol. 5, no. 4, pages 240–250, December 2007. (Cited on page 18.)

[Podgor 1999] Ellen S. Podgor. *Criminal Fraud*, 1999. (Cited on page 57.)

[Poggi 2008] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe Giacomo, Maurizio Lenzerini and Riccardo Rosati. *Linking Data to Ontologies*. In Stefano Spaccapietra, editor, Journal on Data Semantics X, volume 4900 of *Lecture Notes in Computer Science*, pages 133–173. Springer Berlin Heidelberg, 2008. (Cited on pages 2, 99 and 100.)

[Polleres 2011] Axel Polleres, Claudia d'Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski and Peter F. Patel-Schneider, editors. Reasoning web. semantic technologies for the web of data - 7th international summer school 2011, galway, ireland, august 23-27, 2011, tutorial lectures, volume 6848 of *Lecture Notes in Computer Science*. Springer, 2011. (Cited on page 9.)

[PricewaterhouseCoopers 2009] PricewaterhouseCoopers. *The Global Economic Crime Survey*. http://www.pwc.com/en_GX/gx/economic-crime-survey/pdf/global-economic-crime-survey-2009.pdf, 2009. Accessed: 04/04/2013. (Cited on page 41.)

[PricewaterhouseCoopers 2011] PricewaterhouseCoopers. *The Global Economic Crime Survey*. http://www.pwc.com/gx/en/economic-crime-survey/download-economic-crime-people-culture-controls.jhtml, 2011. Accessed: 04/04/2013. (Cited on pages 41 and 43.)

[Ramakrishnan 1993] Raghu Ramakrishnan and Jeffrey D. Ullman. *A Survey of Research on Deductive Database Systems*. Journal of Logic Programming, vol. 23, pages 125–149, 1993. (Cited on page 13.)

[Rohmer 1986] J. Rohmer, R. Lescoeur and Jean-Marc Kerisit. *The Alexander Method - A Technique for The Processing of Recursive Axioms in Deductive Databases*. New Generation Comput., vol. 4, no. 3, pages 273–285, 1986. (Cited on page 29.)

[Rosati 2006] Riccardo Rosati. *DL+log: Tight Integration of Description Logics and Disjunctive Datalog*. In Doherty et al. [Doherty 2006], pages 68–78. (Cited on page 39.)

[Sagiv 1984] Yehoshua Sagiv and Jeffrey D. Ullman. *Complexity of a top-down capture rule*. Rapport technique, Stanford, CA, USA, 1984. (Cited on page 26.)

[Salinger 2004] L.M. Salinger. Encyclopedia of white-collar & corporate crime. Numeéro t. 1 de Encyclopedia of White-collar & Corporate Crime. SAGE Publications, 2004. (Cited on pages 43 and 44.)

[Schmidt-Schauss 1991] M. Schmidt-Schauss and G. Smolka. *Attributive concept descriptions with complements*. Artificial Intelligence, vol. 48, pages 1–26, 1991. (Cited on pages 30 and 35.)

[Sejm 1997] Polish Sejm. *Polish Penal Code*. http://isap.sejm.gov.pl/Download?id=WDU19970880553&type=3, 1997. Accessed: 04/04/2013. (Cited on pages 41, 47 and 71.)

[Sippu 1990] Seppo Sippu and Eljas Soisalon-Soininen. *Multiple SIP strategies and bottom-up adorning in logic query optimization*. In Proceedings of the third international conference on database theory on Database theory, ICDT '90, pages 485–498, New York, NY, USA, 1990. Springer-Verlag New York, Inc. (Cited on page 26.)

[Sirin 2007] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur and Yarden Katz. *Pellet: A practical OWL-DL reasoner*. Web Semant., vol. 5, no. 2, pages 51–53, June 2007. (Cited on page 103.)

[Tekle 2011] K. Tuncay Tekle and Yanhong A. Liu. *More efficient datalog queries: subsumptive tabling beats magic sets*. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, SIGMOD '11, pages 661–672, New York, NY, USA, 2011. ACM. (Cited on page 125.)

[Unger 2007] B. Unger and Elena Madalina. Busuioc. The scale and impacts of money laundering / brigitte unger with a contribution of elena madalina busuioc. Edward Elgar, Cheltenham, UK ; Northampton, MA :, 2007. (Cited on page 79.)

[Valente 1995] André Valente. *Legal knowledge engineering: A modelling approach*. PhD thesis, University of Amsterdam, 1995. (Cited on page 75.)

[van Emden 1976] M. H. van Emden and Robert A. Kowalski. *The Semantics of Predicate Logic as a Programming Language*. Journal of the Association for Computing Machinery, vol. 23, no. 4, pages 733–742, 1976. (Cited on page 18.)

[van Harmelen 2007] Frank van Harmelen, Vladimir Lifschitz and Bruce Porter, editors. Handbook of knowledge representation (foundations of artificial intelligence). Elsevier Science, 2007. (Cited on pages 30, 33 and 34.)

[Vieille 1986] Laurent Vieille. *Recursive Axioms in Deductive Databases: The Query/Subquery Approach*. In Expert Database Conf.'86, pages 253–267, 1986. (Cited on page 29.)

[Visser 1997] Pepijn Visser, Robert W. Van Kralingen and Trevor J. M. Bench-Capon. *A method for the development of legal knowledge systems*. In Proceedings of the 6th International Conference on Artificial Intelligence and Law, ICAIL 1997, Melbourne, Australia, pages 151–160, 1997. (Cited on page 75.)

[Volz 2004] Raphael Volz. *Web ontology reasoning with logic databases*. PhD thesis, 2004. (Cited on page 36.)

[Więckowski 2009] Jacek Więckowski. *Hydra case indictment analysis*, 2009. (Cited on page 48.)

[Winslett 2006] Marianne Winslett. *Raghu Ramakrishnan speaks out on deductive databases, what lies beyond scalability, how he burned through $20M*

briskly, why we should reach out to policymakers, and more. SIGMOD Record, vol. 35, no. 2, pages 77–85, 2006. (Cited on page 125.)

[Wyner 2008] Adam Wyner. *An ontology in OWL for legal case-based reasoning*. Artificial Intelligence and Law, vol. 3, pages 361–387, 2008. (Cited on pages 76 and 78.)

[Wyner 2012] Adam Wyner, Johan Bos, Valerio Basile and Paulo Quaresma. *An Empirical Approach to the Semantic Representation of Laws*. In JURIX, pages 177–180, 2012. (Cited on page 129.)

[Zhang 2010] Hongyu Zhang, Yuan-Fang Li and Hee Beng Kuan Tan. *Measuring design complexity of semantic web ontologies*. J. Syst. Softw., vol. 83, no. 5, pages 803–814, May 2010. (Cited on page 77.)

[Zhao 2005] Gang Zhao. *AKEM: an ontology engineering methodology in FF POIROT*. http://starlab.vub.ac.be/research/projects/poirot/Publications/ffpoirot.D6.8.AKEMinPOIROT.pdf, 2005. Accessed: 04/04/2013. (Cited on pages 2, 75 and 76.)