On the Efficient Recognition of Clashes in Triple Stores

Project Thesis

supervised by
Prof. Dr. Ralf Möller
Associate Prof. Dr. Volker Haarslev

submitted by
Patrick Un

Abstract

Recognizing clashes between atomic concepts is a technique used in description logics tableau algorithm to check satisfiability of concepts w.r.t. to a TBox. In order to support significantly large number of concept terms for reasoning purpose in secondary storage, an RDF triple store can be used to store millions of these RDF triples. This project thesis explores the possibility of recognizing clashes efficiently on concepts modeled as RDF triples, stored in such RDF triple store, by implementing two EXPTIME tableau satisfiability checking algorithms proposed by Goré and Nguyen.

Acknowledgment

I would like to thank Prof. Dr. Ralf Möller
for providing me with an interesting project thesis topic
and for giving me the opportunity to carry on this work at
the Concordia University in Montréal, Canada.

I would also like to thank Prof. Dr. Volker Haarslev
for giving me the opportunity to perform this work under his friendly
guidance at the Faculty of Engineering and Computer Science at the
Concordia University, for his patience and for providing me with
excellent equipment in order to let me to work effectively.

Declaration

I declare that:
this work has been prepared by myself,
all literal or content based quotations are clearly pointed out,
and no other sources or aids than the declared ones have been used.

Hamburg, submitted on: 28th, April 2008
Patrick Wai Un (18792)

Contents

I. Introduction **11**

1. Context **11**

2. Related Works **12**
2.1. Motivation 12
2.2. Technical Prerequisites 12

3. Thesis Statement **13**

4. Document Structure **13**

II. Preliminaries **15**

5. Description Logics **15**
5.1. Fundamentals 15
5.2. Attributive Description Language $\mathcal{AL}$ 16
5.3. Subsumption 16
5.4. Satisfiability 16
5.5. Knowledge Representation 17
5.6. Description Logics Expressiveness 18

6. Ontology Modeling **19**
6.1. RDF and RDF Schema 19
6.2. OWL 19

7. Triple Store **20**
7.1. SPARQL Query Language for RDF 21
7.2. SPARQL Query Language Elements 22

III. Tableau Reasoning **27**

8. Tableau Calculus in Description Logics **27**
8.1. Inference 27
8.2. General Tableau algorithm 28
8.3. Tableau Calculus for an $\mathcal{ALC}$ TBox Γ 28
8.4. Tableau Calculus for $\mathcal{SHI}$ TBox Γ and RBox $\mathcal{R}$ 30

9. Tableau Decision Algorithms **31**
9.1. EXPTIME $\mathcal{ALC}$ Tableau Algorithm 32
9.2. EXPTIME $\mathcal{SHI}$ Tableau Algorithm 33
9.3. Satisfiability Algorithm in RDF Triple Store 35

IV. Analysis and Design **37**

10. Analysis **37**
10.1. Requirements Analysis 37
10.2. Functional Requirements 38
10.3. Use Cases 39

11. Design **41**
11.1. Modeling And-Or Graph Data Structure 41
11.2. Normalization and Mapping to RDF 47
11.3. Software Frameworks 53
11.4. System Architecture 53

V. Implementation **55**

12. Implementation **55**
12.1. RDF Framework Choice 55
12.2. Overview of the Implementation 56
12.3. Tableau Algorithms Implementation 57
12.4. Mapping Tableau Rules to SPARQL Queries 66

VI. Conclusion **69**

13. Summary **69**
13.1. Evaluation 69
13.2. Outlook 70

VII. SPARQL Query Interface and Implementation Classes **73**

Listings

1. Example RDF data triples . . . 22
2. Simple SPARQL query for the example RDF data . . . 23
3. Simple SPARQL query results . . . 23
4. SPARQL query using prefix names . . . 23
5. SPARQL query using FILTER . . . 24
6. SPARQL query result using FILTER . . . 24
7. SPARQL query using alternative graph patterns matching . . . 24
8. SPARQL query results using alternative graph patterns matching . . . 25
9. SPARQL query using result modifiers . . . 25
10. An example of SPARQL ASK query . . . 25
11. An example of SPARQL ASK query returning false . . . 25
12. Modeling content for an and-or graph node . . . 42
13. Node content indirection with references to concepts . . . 45
14. Modeling node status for and-or graph nodes . . . 46
15. Modeling node type for and-or graph nodes . . . 46
16. Modeling successor and predecessor links for and-or graph nodes . . . 47
17. A simple owl:Class definition in an OWL document . . . 48
18. First approach in class normalization . . . 48
19. Another approach in class normalization . . . 48
20. An approach in class normalization together with mapped metadata . . . 49
21. Implementation of the proposed $\mathcal{SHI}$ tableau algorithm . . . 57
22. Implementation of root node creation program logic in the *ReasonerTripleStorePersistentModelImpl* class . . . 60
23. Implementation of *propagate* method . . . 63
24. Implementation of the $\mathcal{ALC}$ bottom rule in the *ReasonerTripleStorePersistentModelImpl* class . . . 64
25. SPARQL query as *template* string to evaluate two atomic $\mathcal{ALC}$ concepts for clash 67
26. Class implementing the interface of SPARQL query processing *(abridged)* . . . 73
27. Class containing SPARQL query templates for both $\mathcal{ALC}$ and $\mathcal{SHI}$ concepts *(abridged)* . . . 85

List of Figures

1. Architecture of a knowledge representation system based on description logics [**?**] [3] 18
2. Tableau rules of the satisfiability algorithm for $\mathcal{ALC}$ [**?**, **?**] [1,2] 28
3. Example of application of $\mathcal{ALC}$ tableau rules for satisfiability checking [**?**] [5] . 30
4. Tableau rules for $\mathcal{SHI}$ tableau calculus w.r.t. $(\mathcal{R}, \Gamma)$ [**?**] [4] 31
5. Tableau decision algorithm for $\mathcal{ALC}$ [**?**] [5] 32
6. The *propagate* routine used for the $\mathcal{ALC}$ decision algorithm [**?**] [5] 33
7. Tableau decision algorithm for $\mathcal{SHI}$ [**?**] [4] 34
8. The *propagate* routine used for the $\mathcal{SHI}$ decision algorithm [**?**] [4] 35
9. Class diagram of the reasoner 53

Part I.
Introduction

1. Context

The semantic web is an evolving extension of the existing web in which the semantics of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content. At its core, the semantic web comprises a set of design principles and a variety of enabling technologies. Major elements of the semantic web are expressed in formal specifications. Some of these include *Resource Description Framework* (RDF) using syntactically a variety of data interchange formats (e.g. RDF/XML, N3, Turtle, N-Triples), and notations such as *RDF Schema* (RDFS) and the *Web Ontology Language* (OWL), all of which are intended to provide a formal description of concepts, terms, and relationships within a given knowledge domain.

The development of the semantic web proceeds in step building a layer on top of another. Figure [1] shows "layer cake" of the semantic web which describes the organization of the semantic web technologies. RDF is the basic data model to represent semantic information. RDF schema which is based on RDF, provides modeling primitives for organizing information into hierarchies. Key primitives of RDF schema are classes and properties. It is the supports the authoring of ontologies. OWL is a more powerful semantic modeling language that builds on top of RDF and RDF schema. However, beside the syntactic preliminaries of the different available technologies, it is necessary to reason on knowledge of the content of the web and the concepts of the domain of discourse from this lower layers upward; we must be able to derive new pieces of knowledge from existing set of explicit knowledge in the domain. This is where *description logics* comes in.

Description logics is the recently thoroughly researched family of knowledge representation formalism that represent the knowledge of an application domain by describing the relevant concepts of the domain as terminology and using these terminology concepts to specify properties of individuals in the domain. Description logics are useful for modeling and reasoning about structured knowledge, fitting into the structural provision of RDF, RDF schema and OWL technologies. With regard to the semantic "layer cake", description logics belong to the upper unifying logic and proof layers.

The *tableau method* is used in description logics for a variety of purposes: to determine if the concepts in a *terminology box* (TBox) is satisfiable; to check if the individuals in an *assertion box* (ABox) is consistent and whether the individuals are satisfiable w.r.t. to the TBox, etc. A *semantic reasoner* is a piece of software that implements the functionality for the mentioned purposes and is able to infer logical consequences from a set of asserted facts or logical axioms.

2. Related Works

2.1. Motivation

In fact, many of the currently semantic reasoners utilize the tableau method at it core for checking satisfiability and consistency. Traditional tableau calculus method build an and-or graph structure to proceed that leads to double-exponential time algorithm because each tableau branch may have exponential length, thus the entire tableau may have a double-exponential number of nodes [**?**] [5].

Goré and Nguyen proposed two EXPTIME decision procedures for $\mathcal{ALC}$ and $\mathcal{SHI}$ using global caching in a sound manner for checking satisfiability w.r.t. a TBox [**?**] [5] [**?**] [4]. They have shown that by using global caching and propagation of both satisfiability and unsatisfiability through the and-or graph, the decision procedure reduces the search space of the tableau and keeps it sound.

On the other hand, many of the semantic reasoners utilize tableau method that is memory based. That means ontology data is loaded into the main memory for efficient reasoning and decision purpose. However, the amount of main memory is not infinite. There is apparently a limit in the total amount of ontology data that a specific reasoner can handled because of the limit posed by the main memory (in fact even considering using virtual memory). In order to store ontology data persistently, relational databases can be used. The ontology data can be stored in persistent storage (secondary storage) and retrieved on demand by the reasoner. In the arena of semantic data reasoning, a *triple store* is used. A triple stores is a persistent storage building mainly on traditional proved relational database management systems (RDBMS) which contains ontology data stored as RDF triples. It provides an API for programmers to retrieve and manipulate the stored ontology data basis.

By utilizing the storage capacity of a triple store, significantly more ontology data can be handled by a reasoner since not all of the data must be held in main memory at once. Consequently, it seems obvious that if a triple store can become a back-end storage for the tableau method based reasoner, a larger amount of ontology data can be handled than by traditional main memory based tableau reasoners. In his thesis work, Bastian Spanneberg [**?**] [6] has provided a prototype implementation of such a system for an $\mathcal{ALC}$ decision procedure.

2.2. Technical Prerequisites

One of the main prerequisites for such system to work properly is that the ontology data which is generally modeled with the OWL language must be mapped to RDF-triples which can be stored in the triple store. The mapping can be referred to as normalization of the OWL modeled concepts. They must be first transformed into RDF-triple style for storing into the triple store. Comparing to many available main memory based reasoners which reason on OWL data structures directly in memory, contemporary triple stores are designed mainly to operate on RDF-triples and only capable of manipulating data structure on this level. Bridging this discrepancy is a main motivation for the design.

Furthermore, in order to experiment with more expressive description logics decision procedure such as the proposed EXPTIME $\mathcal{SHI}$ tableau which incorporates global caching and extends basic $\mathcal{ALC}$ with *transitive*, *inverse roles* and role *hierarchies*, there is an incentive to

implement the prototype further to enable the $\mathcal{SHI}$ tableau since there has been so far no such effort done yet. This means that the design must account for the normalization of concept terms beyond OWL class concepts toward property concepts. For the EXPTIME $\mathcal{SHI}$ tableau with global caching. It is an extended tableau decision procedure with additional tableau rules. It is necessary to translate these rules into triple store API operations and queries in order to enable the reasoner to build the tableau and check satisfiability with regard to the store RDF-triples. From this reasoning, it is obvious that the stored RDF-triples must be normalized and encoded in such a way so that meta-data about the ontology concepts (OWL concept terms) is simultaneously imported and stored inside the triple store.

3. Thesis Statement

The objectives of this thesis are:

- Find out clashes between concept terms in a triple store efficiently and effectively, design a method to normalize the OWL ontology concept terms, storing them with necessary, facilitating metadata as RDF-triples in a persistent RDF triple store for processing in secondary memory.
- Implement a prototype reasoner for the proposed tableau algorithms for $\mathcal{ALC}$ and $\mathcal{SHI}$ by Goré and Nguyen [**?**] [5] [**?**] [4] toward the persistent RDF triple store. Both $\mathcal{ALC}$ and $\mathcal{SHI}$ description logics should be handled accordingly by corresponding tableau rules. The prototype comprises an implementation of the necessary tableau rules for $\mathcal{ALC}$ and $\mathcal{SHI}$ as API methods to manipulate the stored RDF-triples in the secondary storage.

Optimization techniques including global caching and status propagation should be incorporated into the implemented reasoner prototype.

4. Document Structure

Part 2 of this document is a brief introduction to description logics for checking satisfiability w.r.t. TBox using tableau method. It explains the proposed EXPTIME tableau algorithms by Goré and Nguyen [**?**] [5] [**?**] [4], first about the $\mathcal{ALC}$ then the extended $\mathcal{SHI}$ algorithm.

Part 3 describes the issues concerning the triple store programming and querying techniques. Part 4 accounts for a conceptual design of the normalization of terminology box (TBox) input to be imported into triple store; subsequently the design to combine tableau algorithm with the triple store is described.

Part 5 describes the implementation of the design approach of the reasoner. The final part is a summary of the document. It accounts for an evaluation of results of the implemented prototype as well as provide an outlook for the work.

Part II.
Preliminaries

5. Description Logics

This section describes description logics as the important central reasoning formalism used in this thesis. It does not replace a detailed account on the subject.

5.1. Fundamentals

Description logics are a family of knowledge representation formalism that represent the knowledge of an application domain called the "world" by first defining the relevant concepts of the domain (its terminologies), and then using these concepts to specify properties of individuals occurring in the domain. Description logics language is equipped with a formal logic-based semantics; with emphasis on reasoning as a central service. It enables one to infer implicitly represented knowledge from explicit knowledge contained in the domain knowledge base. It incorporates the following characteristics:

- The basic syntactic building blocks are atomic concepts, atomic roles (binary predicates) and individuals (constants).
- The expressive power of the language is restricted in the regard that it uses a finite set of constructors for building complex concepts and roles.
- Implicit knowledge about concepts and individuals can be inferred with inference procedures.

The basis atomic concepts can be combined using constructors to form complex expression, for instance, $Female$, $Person$ and $Woman$ are atomic concepts. Individuals represented by the concept of "persons that are not female" can be written as

$$Person \sqcap Female$$

whereas individuals of human beings of both genders can be represented by

$$Female \sqcup Male$$

While atomic concepts are concerned with unary logical premises, relationship between atomic concepts are represented by atomic roles which are binary constructors such as hasChild, hasParent, etc. It is obvious that on both side of the role constructor, atomic concepts can be attached to form expressions. Furthermore, atomic roles can have value restrictions that are either existential or universal. For instance

$$\exists hasChild.Female$$

is an existential quantification and means "individuals having a female child" conceptually. While a universal quantification on roles such as

$$\forall hasChild.Female$$

means "individuals all of whose children are female". Apparently, there is still more space for creative logical modeling of the "world" or the domain once it gets more complex. For detailed elaboration on the topic please refer to the description logic handbook [**?**] [3].

5.2. Attributive Description Language $\mathcal{AL}$

In order to define formally the basic elements of description logic language semantically, a minimal set of useful language called *attributive language* - $\mathcal{AL}$ is explained here for the sake of clarifying basic concepts. Description logics syntax rule elements with regard to construction and the combination of atomic concepts are:

$$C, D \longrightarrow A|\top|\bot|\neg A|C \sqcap D|\forall R.C|\exists R.T$$

An *interpretation* $\mathcal{I}$ is a non-empty set $\Delta^{\mathcal{I}}$ (*domain of interpretation*) and an *interpretation function* that assign to the atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ as well as to an atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. It denotes semantically that every atomic concept A that is mapped by the interpretation function belong to the domain of interpretation and every binary role R belongs to a binary combination of domain of interpretation. Furthermore, the following inductive definitions further formalize the previous syntax rule elements:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$$
$$\bot^{\mathcal{I}} = \emptyset$$
$$(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$$
$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$
$$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \forall b, (a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$$
$$(\exists R.\top)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \exists b, (a, b) \in R^{\mathcal{I}}\}$$

The equivalence of two concepts C and D, denote by $C \equiv D$ means that $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ with the domain of discourse.

5.3. Subsumption

In description logics reasoning, the terminology *subsumption*, written as

$$C \sqsubseteq D$$

is the basic inference mechanism which checks whether the concept denoted by D (the subsumer) is a correct and more general set of concepts than the set denoted by C (the subsumee), i.e. the subsumption procedure checks if C denotes a subset of the the set denoted by D. The most general concept that subsumes any other concept is the *universal concept*, denoted with ($\top$). While the most specific one which subsumes no other concept is the bottom concept, denoted with ($\bot$). The semantics of subsumption with regard to the previous explanation on *interpretation* is defined as: $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

5.4. Satisfiability

A special case of subsumption is concept *satisfiability* checking. It means the checking of a concept expression does not necessarily denote the empty concept ($\emptyset$). To carry over to the subsumption notion, it simply means that the subsumer is the bottom concept ($\bot$), meaning that the concept is not satisfiable.

5.5. Knowledge Representation

Knowledge representation in description logics is carried out within a knowledge base. It comprises of two components: a *TBox* and an *ABox*.

A TBox contains intensional knowledge in the form of terminological axioms. Basic form of this knowledge is a concept *definition* written as

$$Woman \equiv Person \sqsubseteq Female$$

and such a set of definitions are the constituents of the terminology in the TBox. There are two important assumptions concerning the terminology:

- One and a unique definition for each concept name is solely allowed, no different definitions share the same name.
- Definitions cannot contain *cycles*, i.e. concepts are not defined in a way that refer to themselves or via other indirectly back to them.

The basic main purpose of construction of terminologies is *classification*. It follows directly by the nature of subsumption in that more generally defined concepts subsumes more specific concepts, or contrarily more specific concepts are subsumed by more general ones. It becomes apparent that classification of terminologies helps to build a taxonomic concept hierarchy starting from the most general ones on top to the most specific ones at the bottom.

An ABox contains assertional knowledge in the form of *individual* memberships about the domain, for instance

$$Female \sqcap Person(Diana)$$

means that the individual $Diana$ is a female person and it is obvious that she is a $Woman$. It is called a *concept assertion* because it asserts an individual that belongs to certain defined concepts. Furthermore, as mentioned binary predicates which are called roles can also be asserted; represented as

$$hasChild(Diana, Marc)$$

is a *role assertion* that states that the individual Diana has the individual Marc as her child. Consequently, the main purpose for an ABox is *instance checking* that verifies if a certain individual in an instance of a certain defined concept w.r.t. the TBox. Additionally *realization* operation finds the most specific concept to which a specific individual is an instance of; and *retrieval* finds an individual in the knowledge base that is instance of a given concepts.

Finally *consistency checking* examines if the individuals in the ABox are consistent w.r.t. the TBox. However reasoning upon an ABox from a computational viewpoint can be complexer than reasoning upon TBox concepts. Reasoning in description logics with TBox and ABox is shown in figure 1.

Inference in description logics underlies an important assumption, called the *open-world assumption*. It means that either the inference of the reasoning procedure finds a prove for the reasoning task at hand or that a concept is satisfiable if there is an interpretation $\mathcal{I}$ such that $C^{\mathcal{ALC}} \neq \emptyset$; otherwise it concludes that the available knowledge in the knowledge base is insufficient or that represented information is incomplete. However, it does not state the the represented information is false or the concept is unsatisfiable.

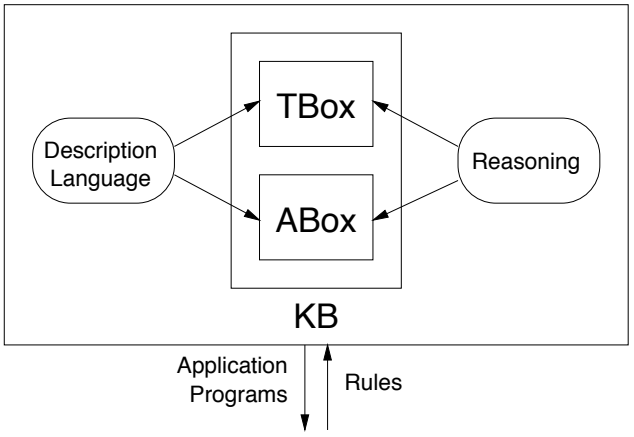

Figure 1: Architecture of a knowledge representation system based on description logics [?] [3]

5.6. Description Logics Expressiveness

Description logics has different degree of expressiveness which have different computational properties and complexity. In the following a brief description is given to explain the aspect. It paves the way to understand the two types of description logics flavors of language $\mathcal{ALC}$ and $\mathcal{SHI}$ used in the thesis later. Notice that the expressiveness in description logics is conventionally denoted with letters, while adding an extra letter, for instance, to the previously described minimal $\mathcal{AL}$ language such as $\mathcal{ALC}$ means adding certain constructors to make the language more expressive.

- $\mathcal{AL} \longrightarrow$ attributive language
  This serves as the basis for more expressive language flavors.
- $\mathcal{C} \longrightarrow$ complex concept negation
  It is allowed to negate arbitrary concepts.
- $\mathcal{H} \longrightarrow$ hierarchical roles
  It supports a hierarchy of roles made up of roles and sub-roles.
- $\mathcal{R} \longrightarrow$ limited complex role inclusion
  This class supports reflexive and disjunct roles.
- $\mathcal{O} \longrightarrow$ *nominals* in DL
  It supports enumeration of value restrictions.
- $\mathcal{I} \longrightarrow$ inverse properties
  It supports inversed role definitions.
- $\mathcal{N} \longrightarrow$ cardinality restrictions
  It supports restriction of properties by cardinality.
- $\mathcal{D} \longrightarrow$ data type properties
  It supports the usage of data types.

Conventionally, there is an abbreviation for the $\mathcal{ALC}$ expressiveness, called $\mathcal{S}$. According to the mentioned flavors, $\mathcal{ALC}$, aka. $\mathcal{S}$ means there is support for the basic attributive language plus

negation of arbitrary concepts. Consequently, $\mathcal{SHI}$, aka. $\mathcal{ALCHI}$ means that there is support for $\mathcal{ALC}$ plus the support for roles hierarchy and inversed roles.

6. Ontology Modeling

The following account does not intend to be an extensive discussion on the different topics but serves as an introduction to the different relevant technologies that are used in the course of this thesis.

6.1. RDF and RDF Schema

Resource description framework (RDF) is a W3C recommendation, a directed labeled graph-based formalism for representing metadata and semantics of information in a machine accessible way. RDF comprises a simple data model consisting of statements. These statements are made up of subject, predicate and object. Such combinations are called a triples. A triple describes a resource or a property. Resources in RDF are identified by a uniform resource identifier (URI). Properties can be URIs. Within a triple, the subject and the predicate are identified by a URI, while the object besides a URI, can also be a literal. The common serialization format of the RDF data model is using XML.

While RDF gives a syntax for metadata annotation, it does not provide means to specify meaning (semantics) to the ontology vocabulary. This is where RDF schema comes in. With RDF schema concept terms defined as classes and relationship or properties between these terms can be defined. RDF schema building blocks are RDF schema language elements such as *Class*, *Property*, *type*, *subClassOf*, *range*, *domain*, etc. However, RDF schema is insufficient to describe resources in full detail. By and large, it lacks the capability to specify constraints and cardinality information; furthermore, on its own it lacks the expressive power to provide sufficient reasoning support. Serialization RDF/XML is the syntax recommended by W3C.

6.2. OWL

The web ontology language (OWL) is the attempt to provide the required expressive power to enhance reasoning support. It has three variants:

1. **OWL-Full**,
2. **OWL-DL**,
3. **OWL-Lite**.

Among them OWL-Lite and OWL-DL are the variants that are associated with description logics directly. They have well researched computational properties. OWL-Lite supports primarily classification and simple constraint features. OWL-DL provides expressiveness in modeling without losing computational completeness and decidability for reasoning. OWL-Full provides both maximal expressiveness and syntactical freedom but without computational guarantees. It is worth to notice that every legal OWL-Lite ontology is a legal OWL-DL ontology, while every legal OWL-DL ontology is also a legal OWL-Full. There is an obvious hierarchical relationships between the different OWL variants, for OWL-Lite is a subset of OWL-DL and OWL-DL is a subset of OWL-Full. However, because of the lack of computational guarantee and undecidability, OWL-Full is currently not suitable for using practically with description logics. OWL-DL

which is equivalent to $\mathcal{SHOIN}(\mathcal{D})$ [**?**] [3] description logics, benefits from well researched activities. It has well defined semantics and formal complexity and decidability properties and supports reasoning algorithms; consequently it has become a key ontology language used with description logics.

Building upon RDF and RDF schema, the fundamental concept of OWL is an OWL *Class*. It is denoted with `owl:Class`. It corresponds to a concept in description logics. A *subclass* introduces a subsumed by ($\sqsubseteq$) relationship. It is denoted with `rdfs:subClassOf`. While an OWL *Class* is a definition of a name that describes a set of individuals, OWL *Individuals* serves to be instances or members of OWL *Class* concepts. For the purpose of this thesis, *individuals* which corresponds to *nominals* in description logics is not relevant.

In order to related OWL Class concepts with each other, OWL properties are used. There are fundamentally two types of properties: An OWL *ObjectProperty* relates between instances of an OWL Class and RDF literals or XML schema types. Additionally, an OWL *DatatypeProperty* relates between two instances of OWL Classes. For both types of properties, an `rdfs:domain` axiom asserts that the subject of the OWL property statement must belong to the OWL Class extension (instance) of the referenced Class description. Further an `rdfs:range` axiom asserts that the values of the OWL property must belong to the OWL Class instance or the data type values range referenced. Table 1 shows a set of available OWL language elements and OWL constructors together with their corresponding description logics axioms syntax as described in the previous section about description logics. Again, it is not intended to be a complete list of OWL constructors, but serves as a good connection between the described description logics axioms to the OWL language because OWL will be used later for ontology modeling to support description logics reasoning. For a complete comprehensive reference on this topic, it is recommended to refer to the W3C recommendations.

Besides simple OWL classes, with additional set constructors shown in table 1, it is possible to form complex classes with disjunction (owl:unionOf), conjunction (owl:intersectionOf), enumeration (owl:oneOf) and negation (owl:complementOf) operations. In a nutshell, OWL-Lite derives from RDF schema necessary constructs. It can model general arbitrary class concepts including build-in support of `owl:Thing` ($\top$) and `owl:Nothing` ($\bot$). From RDF schema, it utilizes its construct to define subclasses and sub-properties. Individuals are instances of classes. It supports equivalence of classes, properties and individuals, whereby restriction on properties can be existential or universal. It also supports cardinality of maximal, minimal or exact ranges from 0 to 1; support of complex classes operations is only restricted to intersection, without support of union or negation. OWL-DL augments this deficiency by supporting also union, disjointness and negation in complex classes operations. Additionally, OWL-DL supports boolean combination of arbitrary class concepts via *and*, *or*, and *not* enumerated classes. There is not restriction on the upper bound of cardinality support of properties.

7. Triple Store

An RDF triple store is a database that is optimized for RDF triples storage. It support three main types of operations:

1. triples(graph pattern)
2. add(triple)

Table 1: OWL language elements with corresponding description logics axioms

OWL Constructors	DL Axioms	Examples
`owl:Thing`	$\top$	Things
`owl:Nothing`	$\bot$	Women $\sqcap\neg$ Female
`owl:Class`	A	Animal (arbitrary concept)
`owl:complementOf`	$\neg A$	$\neg$ Male
`owl:intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male $\sqcap$ Youth
`owl:unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	Male $\sqcup$ Female $\sqcup$ Boys
`rdfs:subClassOf`	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Mammals $\sqcap$ Biped
`owl:equivalentClass`	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
`owl:disjointWith`	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$ Female
`owl:oneOf`	$\{x_1\} \sqcap \ldots \sqcap \{x_n\}$	$\{Bob\} \sqcup \{Alice\}$
`owl:allValuesFrom`	$\forall P.C$	$\forall$ hasChild.Doctor
`owl:someValuesFrom`	$\exists P.C$	$\exists$ hasChild.Grandmother
`owl:ObjectProperty`	P_1	hasChild
`owl:DatatypeProperty`	P_2	timeStamp
`rdfs:subPropertyOf`	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
`owl:equivalentProperty`	$P_1 \equiv P_2$	isCosting $\equiv$ hasPrice
`owl:inverseOf`	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
`owl:TransitiveProperty`	$P \sqsubseteq P^+$	ancestor$^+$ $\sqsubseteq$ ancestor
`owl:FunctionalProperty`	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq 1\ P$
`owl:maxCardinality`	$\leq nP$	$\leq$ 1hasChild
`owl:minCardinality`	$\geq nP$	$\geq$ 5hasChild

3. remove(triple)

These are atomic operations. The *triples* operation uses a certain given RDF graph pattern to match stored triples and returns the set of matches. The *add* operation inserts a triple of the form subject, predicate and object into the triple store. The *remove* operation deletes a given triple from the triple store. Since RDF triple store is a database for RDF triples, it must store the triples that adheres to the RDF recommended format:

- the subject must be a resource identifier, i.e. a universal resource identifier (URI) reference,
- the predicate can be a resource identifier or a property,
- the object must be either a resource identifier or a literal value, i.e. the object represents a value without a URI.

There are currently a collection of triple stores implementation, some being open source while others are licensed commercially.

7.1. SPARQL Query Language for RDF

In order to search and retrieve RDF triples in an efficient manner, independent of specific triple store implementations or any middleware that interfaces with the triple stores, it is necessary to develop a platform-independent querying and results retrieving mechanism for this purpose.

W3C has published a query language recommendation, called *SPARQL query language for RDF*. The W3C RDF data access working group has published the SPARQL recommendation consisting of three parts: a specification of the SPARQL query language which lays down the syntactics and semantics of the query language for RDF data, a specification of a SPARQL protocol which uses web services description language (WSDL) to describe a means for conveying SPARQL queries to a SPARQL query processing service and returning the results via web services requests and replies; and a specification of an XML format that describes variable binding and boolean results formats in the SPARQL query results. This is intended to used in tandem with the SPARQL protocol via web services.

7.2. SPARQL Query Language Elements

The SPARQL query language is used extensively in this thesis to query and manipulate the RDF triples stored in the triple store, therefore it is worth to describe briefly some of the important relevant language elements. For detailed description, one should refer to the W3C recommendation.

The SPARQL query language uses *internationalized resource identifiers*(IRIs) [RFC3987] which is a subset of RDF URI references without spaces. The IRI references within a SPARQL query are delimited by (<>), additionally beside supporting IRIs, SPARQL also supports literal, i.e. string values delimited by a pair of (""); typed literals which are associated with specific XML schema types are supported also. Typed literals are represented as string values concatenated with their corresponding XML schema type URI by two (∧∧) together.

In a nutshell, querying RDF data with SPARQL is about *graph pattern* matching. It means that a SPARQL query contains a certain graph pattern which is compared against the RDF triples for matches. Results are the subset of graph patterns that match a subset of the stored RDF triples. There are four SPARQL query forms:

1. **SELECT** ⟶ returns all or a subset of query pattern match bound to one or more SPARQL query variables.
2. **ASK** ⟶ returns a boolean value indicating whether a query pattern matches or not.
3. **DESCRIBE** ⟶ returns an RDF graph which describes the resource being queried.
4. **CONSTRUCT** ⟶ returns an RDF graph constructed by substitution of variables in a set of query templates.

Among these query forms, the **SELECT** and **ASK** query forms are of direct relevance to the implementation of the current reasoner.

The following example code clarifies how these query forms are used. Supposed there is the following RDF data 1 in the triple store.

Listing 1: Example RDF data triples

```
@prefix reasoner:    <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>

reasoner:graph-node-01 reasoner:name "Node-01"^^xsd:string .
reasoner:graph-node-01 reasoner:nodeType <http://www.sts.tu-harburg.de/dl/gore/exp-owl-
    reasoner/nodes/graphNode> .

reasoner:graph-node-02 reasoner:name "Node-02"^^xsd:string .
```

```
reasoner:graph-node-02 reasoner:nodeType <http://www.sts.tu-harburg.de/dl/gore/exp-owl-
    reasoner/nodes/graphNode> .

reasoner:graph-node-03 reasoner:name "Node-03"^^xsd:string .
reasoner:graph-node-03 reasoner:nodeType <http://www.sts.tu-harburg.de/dl/gore/exp-owl-
    reasoner/nodes/graphNode> .

reasoner:graph-node-04 reasoner:name "Node-04"^^xsd:string .
reasoner:graph-node-04 reasoner:nodeType <http://www.sts.tu-harburg.de/dl/gore/exp-owl-
    reasoner/nodes/graphNode> .

reasoner:graph-node-05 reasoner:name "Node-05"^^xsd:string .
reasoner:graph-node-05 reasoner:nodeType <http://www.sts.tu-harburg.de/dl/gore/exp-owl-
    reasoner/nodes/graphNode> .

reasoner:graph-node-06 reasoner:name "Node-06"^^xsd:string .
reasoner:graph-node-06 reasoner:nodeType <http://www.sts.tu-harburg.de/dl/gore/exp-owl-
    reasoner/nodes/graphNode> .

reasoner:graph-node-06 reasoner:name "Node-06"^^xsd:string .
reasoner:graph-node-06 reasoner:nodeType <http://www.sts.tu-harburg.de/dl/gore/exp-owl-
    reasoner/nodes/graphNode> .

reasoner:graph-node-RT reasoner:name "root"^^xsd:string .
reasoner:graph-node-RT reasoner:nodeType <http://www.sts.tu-harburg.de/dl/gore/exp-owl-
    reasoner/nodes/rootNode> .
```

A simple SPARQL query that queries for pattern matches on nodes can be written as follows 2:

Listing 2: Simple SPARQL query for the example RDF data

```
SELECT ?node
WHERE {
    ?node <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodeType> <http://www.
        sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/graphNode>
}
```

Here the variable *node* which is proceeded by a question mark is called a *query variable*. The query variable binds the result set of graph pattern matches and returns them in the result set. The SELECT keyword can take one or more query variables separated by space. The WHERE keyword introduces a condition clause which resembles a WHERE clause in a traditional SQL query; the WHERE clause is enclosed within the braces. There can be more than one clause enclosed. In this case, there is more clauses, they are separated by a dot. The query result looks like the following 3.

Listing 3: Simple SPARQL query results

```
node
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-01>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-02>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-03>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-04>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-05>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-06>
```

It is shown that the query variable is bound to 6 results, all of them being IRIs. One of the tedious thing of such long IRIs is obvious if every query graph pattern must contain a long URI reference. This is where *prefix* expressions come in. Denoted with a PREFIX keyword in the SPARQL query, a namespace URI is allowed to be bound to a namespace prefix name to be used in the SPARQL query such as 4. The results remain the same for the same query.

Listing 4: SPARQL query using prefix names

```
PREFIX rdf:          <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
PREFIX rdfs:         <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl:          <http://www.w3.org/2002/07/owl>
PREFIX reasoner:     <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
PREFIX nodes:        <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
SELECT ?node
WHERE {
    ?node reasoner:nodeType nodes:graphNode
}
```

As it is obvious here the WHERE clause is briefer than the previous query, also notice that the angle brackets are not necessary if the prefix name is used.

Further, if it is desired that certain condition is ruled out from the rule set, a *filter* denoted with the FILTER keyword can be used. A filter is a predicate that evaluates to a boolean value, i.e. true or false. The result set will only contain the subset of the query results that corresponds to the condition in which the filter part evaluates to true, for instance, if it is not desired to retrieve any node of node type graphNode in the example data; a filter can be used as follows 5:

Listing 5: SPARQL query using FILTER

```
PREFIX rdf:          <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:         <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl:          <http://www.w3.org/2002/07/owl>
PREFIX reasoner:     <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
PREFIX nodes:        <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
SELECT ?node
WHERE {
    ?node reasoner:nodeType ?nodetype .
FILTER ( str(?nodetype) != "http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes
    /graphNode" )
}
```

In the example, the SPARQL query introduces another placeholder query variable *nodetype* which is not selected but serves to be bound in the filter predicate. The filter uses the built-in *str* function to return a lexical form, i.e. string form of the bound *nodetype* variable and checks that its value is not that on the right hand side. The result is obvious here 6 as shown. Only the root node is selected and bound to the *node* variable.

Listing 6: SPARQL query result using FILTER

```
node
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/reasoner:graph-node-RT>
```

A SPARQL query can handle two different alternative graph pattern by using the UNION keyword. In the next example 7 shown, the WHERE clause consists of two graph pattern groups enclosed inside the braces. The UNION keyword combines both graph pattern as alternatives and tries to find matches that satisfy both patterns.

Listing 7: SPARQL query using alternative graph patterns matching

```
PREFIX rdf:          <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:         <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl:          <http://www.w3.org/2002/07/owl>
PREFIX reasoner:     <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
PREFIX nodes:        <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
SELECT ?node
WHERE {
    { ?node reasoner:nodeType nodes:graphNode }
UNION
    { ?node reasoner:nodeType nodes:rootNode }
}
```

Listing 8: SPARQL query results using alternative graph patterns matching

```
node
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/reasoner:graph-node-RT>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-01>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-02>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-03>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-04>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-05>
<http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/graph-node-06>
```

The result consists of matches that satisfy both graph patterns as shown in 8. Furthermore, it is allowed to modify, for instance, the result ordering of the result set by using the ORDER BY and either ASC (ascending) and DESC (descending) keywords such as shown in 9.

Listing 9: SPARQL query using result modifiers

```
PREFIX rdf:          <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:         <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl:          <http://www.w3.org/2002/07/owl>
PREFIX reasoner:     <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
PREFIX nodes:        <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
SELECT ?node
WHERE {
    { ?node reasoner:nodeType nodes:graphNode }
UNION
    { ?node reasoner:nodeType nodes:rootNode }
}
ORDER BY ASC (?node)
```

Another query form ASK returns boolean result instead of result set bound to query variables. In the example shown in 10, it is clear that an ASK query does not take a query variable as in a SELECT query. It is nevertheless similar to a SELECT query, although the *node* variable is still present in the WHERE clause, it is not selected. The ASK query obviously will return true.

Listing 10: An example of SPARQL ASK query

```
PREFIX rdf:          <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:         <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl:          <http://www.w3.org/2002/07/owl>
PREFIX reasoner:     <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
PREFIX nodes:        <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
ASK
WHERE {
    ?node reasoner:nodeType nodes:rootNode
}
```

It is also possible to use a filter predicate to introduce condition for evaluation and to use alternative patterns matching using the UNION keyword, just as the SELECT query form allows.

Listing 11: An example of SPARQL ASK query returning false

```
PREFIX rdf:          <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:         <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl:          <http://www.w3.org/2002/07/owl>
PREFIX reasoner:     <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
PREFIX nodes:        <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
ASK
WHERE {
    ?node reasoner:nodeType ?nodetype .
FILTER ( str(?nodetype) != "http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes
    /graphNode"
         && str(?nodetype) != "http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/
             nodes/rootNode" )
}
```

Therefore the SPARQL ASK query as shown in 11 with regard to the example RDF data will return false.

These examples are intended not as a detailed account on all the features of the SPARQL query language but a selection of the relevant language elements that are used in this thesis. Further information can be found in the specifications published by W3C.

Part III.
Tableau Reasoning

8. Tableau Calculus in Description Logics

This part of the thesis introduce the tableau reasoning algorithm in general. Subsequently, two tableau algorithms for $\mathcal{ALC}$ and $\mathcal{SHI}$ as well as the proposed algorithms for these description logics languages by Goré and Nguygen [**?**] [5] [**?**] [4] are explained.

8.1. Inference

Checking satisfiability of concepts is one of the most important inference procedures, for other inferences can be *reduced* to the checking of (un)satisfiability of concepts. For instance, let Γ be a TBox as described in the previous section 5.5, then for two arbitrary concepts C, D w.r.t. TBox Γ checking:

- **Satisfiability** of C means to check whether there exists an *interpretation* $\mathcal{I}$, also called a *model*, of all the axioms in Γ so that $C^{\mathcal{I}} \notin \emptyset$. Then $\mathcal{I}$ is a model of C.
- **Subsumption** ($C \sqsubseteq D$) verifies that C is *subsumed* by D w.r.t. Γ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\Gamma \longrightarrow \Gamma \models C \sqsubseteq D$
- **Equivalence** ($C \equiv D$) verifies that $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $C^{\mathcal{I}}$ in $\Gamma \longrightarrow \Gamma \models C \equiv D$.
- **Disjointness** ($C \sqsubseteq \neg D$) verifies that $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model $C^{\mathcal{I}}$ of Γ

With this in mind the checking of unsatisfiability, equivalence and disjointness can be reduced to the checking of subsumption as the following:

- C is unsatisfiable means C *is subsumed by* $\perp$,
- C and D are equivalent means C is subsumed by D and D is subsumed by C,
- C and D are disjoint means $C \sqsubseteq D$ is subsumed by $\perp$.

Furthermore, reasoning system allows one to negate concepts that are combined by $\sqsubseteq$ operator. Checking subsumption, equivalence and disjointness of concepts can be reduced to checking satisfiability of concepts for the following reasons (with regard to the previous concepts):

- C *is subsumed by* D means $C \sqsubseteq \neg D$ is *unsatisfiable*,
- C and D are *equivalent* means both ($C \sqsubseteq \neg D$) and ($D \sqsubseteq \neg C$) are *unsatisfiable*,
- C and D are *disjoint* means $C \sqsubseteq D$ is *unsatisfiable*.

Because of this, it is sufficient to develop inference algorithms that decide the satisfiability of concepts if the language supports conjunction and negation of arbitrary concepts $\longrightarrow$ this is the starting point and core principle underlying the tableau calculus, an inference mechanism that operates on checking satisfiability of concepts.

8.2. General Tableau algorithm

The fundamental principle is to use negation to reduce subsumption to unsatisfiability of concepts. To put it in another way, for two concepts, C and D, $C \sqsubseteq D$ iff $C \sqcap \neg D$ is *unsatisfiable*. The *negation normal form* of concepts is used which means negation only occurs directly in front of concept names.

The $\rightarrow_{\sqcap}$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but not both $C_1(x)$ and $C_2(x)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x), C_2(x)\}$.

The $\rightarrow_{\sqcup}$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x)\}, \mathcal{A}'' := \mathcal{A} \cup \{C_2(x)\}$.

The $\rightarrow_{\exists}$-rule
Condition: $\mathcal{A}$ contains $(\exists r.C)(x)$, but there is no individual name z such that $C(z)$ and $r(x, z)$ are in $\mathcal{A}$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y), r(x, y)\}$ where y is an individual name not occurring in $\mathcal{A}$.

The $\rightarrow_{\forall}$-rule
Condition: $\mathcal{A}$ contains $(\forall r.C)(x)$ and $r(x, y)$, but it does not contain $C(y)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$.

Figure 2: Tableau rules of the satisfiability algorithm for $\mathcal{ALC}$ [**?**, **?**] [1,2]

For an ABox $\mathcal{A}$, let C_0 be an $\mathcal{ALC}$ concept in *negation normal form*, for testing the satisfiability of concept C_0, the general tableau algorithm starts with $\mathcal{A}_0 = \{C_0(x_0)\}$; the set of *tableau transformation rules* or simply *tableau rules* as shown in figure 2 can be applied in arbitrary order to the ABox $\mathcal{A}$ to transform it into a finite set of ABoxes, until no more rules can be applied. An ABox $\mathcal{A}$ is *complete* if none of the tableau rules applies to it. The ABox contains a *clash* if $\{P(x), \neg P(x)\} \subseteq A$, meaning that for some individual named x w.r.t. some concept P, i.e. $P(x) \cap \neg P(x)$ *is subsumed by* $\perp$. An ABox is called *closed* if it contains a clash as described previously, else it is *open* otherwise. A closed ABox $\mathcal{A}$ is *inconsistent*; otherwise any complete and open ABox $\mathcal{A}$ is *consistent*, or put it the other way: consistency checking can be approached by looking at obvious clashes.

8.3. Tableau Calculus for an $\mathcal{ALC}$ TBox Γ

The following notations are used in the description of tableau calculus for $\mathcal{ALC}$; here A stands for *atomic concepts* while C and D are conventionally names for arbitrary concepts. R is a notation for role name.

$$C, D ::= \top | \perp | A | \neg C | C \sqcap D | C \sqcup D | C \sqsubseteq D | C \doteq D | \forall R.C | \exists R.C$$

The general notion described in the previous sections are appropriate for checking satisfiability of concepts axioms in a TBox Γ. It is important to notice that concepts are transformed into

the negation normal form as described in the previous section 8.2, i.e.:

$$C \sqsubseteq D \Longrightarrow C \sqcap \neg D \equiv \neg C \sqcup D$$

meaning the negation normal form of the arbitrary concepts C and D is a conjunction of C with the negated concept D or a disjunction of negated C and D. Additionally, equivalence between C and D can be transformed into:

$$C \doteq D \Longrightarrow (C \sqsubseteq D) \sqcap (D \sqsubseteq C) \equiv (C \sqcap \neg D) \sqcap (D \sqcap \neg C)$$

With regard to the subsequently tableau algorithm proposed for $\mathcal{ALC}$ by Goré and Nguyen, the fundamentals of tableau calculus for $\mathcal{ALC}$ with the corresponding tableau rules and an example w.r.t. TBox Γ are described in [**?**] [5].

Let Γ be a TBox containing $\mathcal{ALC}$ concepts in the mentioned *negation normal form*. A general $\mathcal{ALC}$ tableau rule (ρ) is written as:

$$(\rho)\Gamma : \frac{X}{Y_1 | \ldots | Y_k}$$

consists of a TBox Γ, a numerator X, and a finite list of denominators $Y_1, Y_2, \ldots, Y_k$ representing finite sets of concepts. Such a tableau rule is read downwards as "if the numerator X is satisfiable w.r.t. the TBox Γ containing $\mathcal{ALC}$ concepts, then some denominators $Y_1, \ldots, Y_i$ is also satisfiable w.r.t. Γ". The concepts in the numerator are called *principal concepts*. The union of to sets of concepts $X \cup Y$ is written as $X; Y$ while the union of a set of concepts X with a concept set element $\{C\}$, $X \cup \{C\}$ is denoted with $X; C$. Consequently, with resemblance to the tableau rules for an ABox as described in the previous section 8.2, the tableau rules for an $\mathcal{ALC}$ TBox Γ are as follows:

$$(\bot) \quad \frac{X; A; \neg A}{\bot} \qquad (\sqcap) \quad \frac{X; C \sqcap D}{X; C; D} \qquad (\sqcup) \quad \frac{X; C \sqcup D}{X; C | X; D}$$

$$(\exists R) \quad \Gamma : \frac{X; \exists R.C}{\{D : \forall R.D \in X\}; C; \Gamma}$$

The ***bottom rule***($\bot$), the ***and rule***($\sqcap$), and the ***or rule***($\sqcup$) are static and the ***restriction rule***($\exists R$) is transitional.

A tableau w.r.t. TBox Γ containing $\mathcal{ALC}$ concepts for a finite set of concepts X is a ***tree***, i.e. a *graph* structure with root $(X; \Gamma)$ and from the root containing finite sets of concepts obtained from the parent nodes by applying the previously described $\mathcal{ALC}$ tableau rules in the given order to the parent node. The expansion of nodes stops when no more rules are applicable, then expansion has come to an *end node*. Since a tableau tree contains finite set of branches, a branch in a tableau is *closed* if its *end node* contains only $\bot$, i.e. there is a *clash* of concepts. An entire tableau is *closed* if *all* of its branches are *closed*; otherwise the tableau is *open*. A finite set of concepts X is *consistent* and *satisfiable* w.r.t. the TBox Γ if every tableau w.r.t. Γ for X is open. Goré and Nguyen have further proved the correctness and completeness in [**?**] [5].

With an example [**?**] [5] clarifying the application of the tableau rules, let the TBox Γ be:

$$\Gamma = \{A \sqsubseteq B \sqcap C\} \Longrightarrow \{\neg A \sqcup (B \sqcap C)\}$$

and the finite set of concepts X that must be checked for satisfiability w.r.t. Γ:

$$X := \{(\exists R.A \sqcap \exists R.(A \sqcap \neg B)) \sqcup (\exists R.A \sqcap \exists R.(A \sqcap \neg C))\}$$

Figure 3 shows the application of the $\mathcal{ALC}$ tableau rules (the subscript Λ marks the application of a rule in each step).

$$\begin{array}{c}
\neg A \sqcup (B \sqcap C);\ (\exists R.A \sqcap \exists R.(A \sqcap \neg B)) \sqcup^* (\exists R.A \sqcap \exists R.(A \sqcap \neg C)) \\
\hline
\begin{array}{c}
\neg A \sqcup (B \sqcap C); \exists R.A \sqcap^* \exists R.(A \sqcap \neg B) \\
\hline
\neg A \sqcup (B \sqcap C); \exists R.A; \exists R^*.(A \sqcap \neg B) \\
\hline
A \sqcap^* \neg B; \neg A \sqcup (B \sqcap C) \\
\hline
A; \neg B; \neg A \sqcup^* (B \sqcap C) \\
\hline
\begin{array}{c|c}
\dfrac{A; \neg B; \neg A}{\bot} & \begin{array}{c} A; \neg B; B \sqcap^* C \\ \hline A; \neg B; B; C \\ \hline \bot \end{array}
\end{array}
\end{array}
\quad\Bigg|\quad
\begin{array}{c}
\neg A \sqcup (B \sqcap C); \exists R.A \sqcap^* \exists R.(A \sqcap \neg C) \\
\hline
\neg A \sqcup (B \sqcap C); \exists R.A; \exists R^*.(A \sqcap \neg C) \\
\hline
A \sqcap^* \neg C; \neg A \sqcup (B \sqcap C) \\
\hline
A; \neg C; \neg A \sqcup^* (B \sqcap C) \\
\hline
\begin{array}{c|c}
\dfrac{A; \neg C; \neg A}{\bot} & \begin{array}{c} A; \neg C; B \sqcap^* C \\ \hline A; \neg C; B; C \\ \hline \bot \end{array}
\end{array}
\end{array}
\end{array}$$

Figure 3: Example of application of $\mathcal{ALC}$ tableau rules for satisfiability checking [**?**] [5]

All the *end nodes* are closed because of the application of the *bottom rule* ($\bot$) since there are clashes in every branch of the tableau. It means that the entire tableau is closed and hence the given set of concept X is *unsatisfiable* w.r.t. TBox Γ.

8.4. Tableau Calculus for $\mathcal{SHI}$ TBox Γ and RBox $\mathcal{R}$

The description logics $\mathcal{SHI}$ extends the $\mathcal{ALC}$ description logics with transitive roles, inverse roles and role hierarchies. In order to use the tableau calculus for $\mathcal{SHI}$, it is necessary to extend the described $\mathcal{ALC}$ tableau calculus described in the previous section 8.3. For a finite set of role names, denoted with RNames, let RNames$^-$ $=\{r^-|r \in RNames\}$ be the set of inversed roles. Let R and S be notations for arbitrary $\mathcal{SHI}$ roles. Consequently, for a $\mathcal{SHI}$ role R, the inverse role is denoted with $R^- = r^-$ if $R = r$; and $R^- = r$ if $R = r^-$. A similar set of notations describing the TBox concepts as those in the previous section can be used:

$$C, D ::= \top|\bot|A|\neg C|C \sqcap D|C \sqcup D|C \sqsubseteq D|C \doteq D|\forall R.C|\exists R.C$$

These are concepts contained in the TBox Γ. Additionally, a $\mathcal{SHI}$ RBox $\mathcal{R}$ contains a finite set of role inclusion axioms $R \sqsubseteq S$ and roles transitivity axioms $R \circ R \sqsubseteq R$ for the arbitrary $\mathcal{SHI}$ roles R and S given as follows:
$R \sqsubseteq R \in \mathcal{R}$ for every $\mathcal{SHI}$ role R
$R \sqsubseteq S \in \mathcal{R}$ implies $R^- \sqsubseteq S^- \in \mathcal{R}$
$R \circ R \sqsubseteq R \in \mathcal{R}$ implies $R^- \circ R^- \sqsubseteq R^- \in \mathcal{R}$
$R \sqsubseteq R' \in \mathcal{R}$ and $R' \sqsubseteq R'' \in \mathcal{R}$ imply $R \sqsubseteq R'' \in \mathcal{R}$

On the basis of the $\mathcal{ALC}$ tableau rules, a general tableau rule for $\mathcal{SHI}$ is written as follows:

$$(\rho)\frac{X}{Y_1|\ldots|Y_k}$$

here X is the numerator of a $\mathcal{SHI}$ tableau rule w.r.t. $(\mathcal{R}, \Gamma)$, denoting a finite set of concepts, where $\mathcal{R}$ is a $\mathcal{SHI}$ RBox and Γ a TBox. The denominator consists of $Y_1, Y_2, \ldots, Y_k$ which denotes finite sets of concepts. Figure 4 shows the tableau calculus for $\mathcal{SHI}$ description logics.

$$(\bot)\ \frac{X\ ;\ C\ ;\ \overline{C}}{\bot} \qquad (\sqcap)\ \frac{X\ ;\ C \sqcap D}{X\ ;\ C \sqcap D\ ;\ C\ ;\ D} \qquad (\sqcup)\ \frac{X\ ;\ C \sqcup D}{X\ ;\ C \sqcup D\ ;\ C \mid X\ ;\ C \sqcup D\ ;\ D}$$

$$(H)\ \frac{X\ ;\ \forall R.C}{X\ ;\ \forall R.C\ ;\ \forall S.C}\ \text{if } S \sqsubseteq R \in \mathcal{R} \qquad (H^{\dagger})\ \frac{X\ ;\ \forall^{\dagger} R.C}{X\ ;\ \forall^{\dagger} R.C\ ;\ \forall^{\dagger} S.C}\ \text{if } S \sqsubseteq R \in \mathcal{R}$$

$$(cut_{\forall})\ \frac{X}{X\ ;\ \forall R.C \mid X\ ;\ \exists R.\overline{C}}\ \text{if } \forall R.C \in \widetilde{Sf}_{\mathcal{R}}(X \cup \mathcal{T})$$

$$(cut_{B})\ \frac{X}{X\ ;\ C \mid X\ ;\ \overline{C}\ ;\ \forall^{\dagger} R^{-}.\exists R.\overline{C}}\ \text{if } \forall R.C \in \widetilde{Sf}_{\mathcal{R}}(X \cup \mathcal{T})$$

$$(cut_{5})\ \frac{X}{X\ ;\ \forall R.C \mid X\ ;\ \exists R.\overline{C}\ ;\ \forall^{\dagger} R^{-}.\exists R.\overline{C}}\ \begin{array}{l}\text{if } \forall R.C \in \widetilde{Sf}_{\mathcal{R}}(X \cup \mathcal{T}) \\ \text{and } R \circ R \sqsubseteq R \in \mathcal{R}\end{array}$$

$$(\exists R)\ \frac{X\ ;\ \exists R.C}{trans_{\mathcal{R}}(X, R)\ ;\ C\ ;\ \mathcal{T}}\ \text{where}$$

$$\begin{aligned} trans_{\mathcal{R}}(X, R) = {} & \{D \mid ((\forall S.D \in X) \vee (\forall^{\dagger} S.D \in X)) \wedge (R \sqsubseteq S \in \mathcal{R})\} \\ & \cup \{\forall S.D \in X \mid (R \sqsubseteq S \in \mathcal{R}) \wedge (S \circ S \sqsubseteq S \in \mathcal{R})\} \\ & \cup \{\forall^{\dagger} S.D \in X \mid (R \sqsubseteq S \in \mathcal{R}) \wedge (S \circ S \sqsubseteq S \in \mathcal{R})\} \end{aligned}$$

Figure 4: Tableau rules for $\mathcal{SHI}$ tableau calculus w.r.t. $(\mathcal{R}, \Gamma)$ [**?**] [4]

The rules $(\bot)$, $(\sqcap)$, $(\sqcup)$, (H), $(H^{\dagger})$, $(cut_{\forall})$, (cut_B) and (cut_5) are static rules while $(\exists R)$ is a transitional rule.

The $\mathcal{SHI}$ tableau for X w.r.t. TBox and RBox $(\mathcal{R}, \Gamma)$ is a tree with root node $(X; \Gamma)$. Resembling the application of the $\mathcal{ALC}$ tableau rules for child node expansion of the tree, the *end node* of a specific branch is reached if no more $\mathcal{SHI}$ rule is applicable on that branch. A branch in a $\mathcal{SHI}$ tableau is *closed* if its end node carries only $\bot$. An entire $\mathcal{SHI}$ tableau is closed if all branches are closed. The $\mathcal{SHI}$ tableau is *open* if it is not closed. A finite set of concepts denoted with X is *consistent* w.r.t. $(\mathcal{R}, \Gamma)$ if every tableau for X w.r.t. $(\mathcal{R}, \Gamma)$ is open; otherwise X is inconsistent. concept X is *satisfiable* w.r.t. $(\mathcal{R}, \Gamma)$ if it is consistent w.r.t. $(\mathcal{R}, \Gamma)$. The proof of soundness, correctness and completeness is given by [**?**] [4]. There is also an example of the application of the $\mathcal{SHI}$ tableau calculus rules given in the same paper.

9. Tableau Decision Algorithms

Since recognizing clashes between a pair of atomic concepts A and $\neg A$ efficiently is important in order to determine whether a branch of a tableau is *open* or *closed* in an efficient manner; consequently for determining if the tableau is open or closed in order to check if some concepts X is *consistent* and *satisfiable* with a given TBox Γ. There have been much research effort for finding efficient decision algorithms, for the expansion of the entire tableau usually requires double exponential time. Goré and Nguyen have proposed two EXPTIME tableau satisfiability checking algorithms [**?**, **?**] [4][5], one using the $\mathcal{ALC}$ and the other the $\mathcal{SHI}$ tableau rules directly to create an *and-or graph* $G = \langle V, E \rangle$, i.e. a directed graph consisting of only *and nodes* (by applying the $(\sqcap)$ rule on the previous node content) and *or nodes* (by applying the $(\sqcup)$ rule). They have used optimization such as *global caching* and *propagation* of both satisfiability

and unsatisfiability in order to efficiently find out clashes.

9.1. EXPTIME $\mathcal{ALC}$ Tableau Algorithm

The $\mathcal{ALC}$ algorithm constructs an and-or graph with nodes keeping track of three attributes:
content: the set of concepts associated with the node
status: { *unexpanded*, *expanded*, *sat*, *unsat* }
kind : { *and-node*, *or-node* }

In order to check whether a given finite set of concepts X is satisfiable w.r.t. the given TBox Γ, the content of an initial node, called *root node* τ with status *unexpanded* is set to the union of the TBox content and the set of concepts, i.e. $\Gamma \cup X$. The $\mathcal{ALC}$ decision algorithm is shown in 5. Here the input for the algorithm is the union of the set of concepts X and the TBox Γ. The

Input: two finite sets of concepts Γ and X
Output: an and-or graph $G = \langle V, E \rangle$ with $\tau \in V$ as the initial node such that $\tau.status = \texttt{sat}$ iff X is satisfiable w.r.t. Γ

1. create a new node τ with $\tau.content := \Gamma \cup X$ and $\tau.status := \texttt{unexpanded}$;
   let $V := \{\tau\}$ and $E := \emptyset$;
2. while $\tau.status \notin \{\texttt{sat}, \texttt{unsat}\}$ and we can choose an unexpanded node $v \in V$ do:
   (a) $\mathcal{D} := \emptyset$;
   (b) if no $C\mathcal{ALC}$-tableau rule is applicable to $v.content$ then $v.status := \texttt{sat}$
   (c) else if ($\bot$) is applicable to $v.content$ then $v.status := \texttt{unsat}$
   (d) else if ($\sqcap$) is applicable to $v.content$ giving denominator Y then
   $v.kind := \texttt{and-node}$, $\mathcal{D} := \{Y\}$
   (e) else if ($\sqcup$) is applicable to $v.content$ giving denominators Y_1 and Y_2 then
   $v.kind := \texttt{or-node}$, $\mathcal{D} := \{Y_1, Y_2\}$
   (f) else
   i. $v.kind := \texttt{and-node}$,
   ii. for every $\exists R.C \in v.content$, apply ($\exists R$) to $v.content$ giving denominator $\{D \mid \forall R.D \in v.content\} \cup \{C\} \cup \Gamma$ and add this denominator to $\mathcal{D}$;
   (g) for every denominator $Y \in \mathcal{D}$ do
   i. if some proxy $w \in V$ has $w.content = Y$ then add edge (v, w) to E
   ii. else let w be a new node, set $w.content := Y$, $w.status := \texttt{unexpanded}$, add w to V, and add edge (v, w) to E;
   (h) if ($v.kind = \texttt{or-node}$ and one of the successors of v has status $\texttt{sat}$)
   or ($v.kind = \texttt{and-node}$ and all the successors of v have status $\texttt{sat}$) then
   $v.status := \texttt{sat}$, $propagate(G, v)$
   (i) else if ($v.kind = \texttt{and-node}$ and one of the successors of v has status $\texttt{unsat}$)
   or ($v.kind = \texttt{or-node}$ and all the successors of v have status $\texttt{unsat}$) then
   $v.status := \texttt{unsat}$, $propagate(G, v)$
   (j) else $v.status := \texttt{expanded}$;
3. if $\tau.status \notin \{\texttt{sat}, \texttt{unsat}\}$ then
   for every node $v \in V$ with $v.status \neq \texttt{unsat}$, set $v.status := \texttt{sat}$;

Figure 5: Tableau decision algorithm for $\mathcal{ALC}$ [?] [5]

output is an expanded *and-or graph* $G = \langle V, E \rangle$ with the root node τ so that $\tau.status$ = *sat* if

X is satisfiable w.r.t. Γ. The main while loop carries on processing the nodes until the status of τ is determined to be either in { *sat*,*sat* }, or until every node is expanded.

The main loop shown in figure 5 applies the $\mathcal{ALC}$ tableau rules in the order ($\bot$), ($\sqcap$), ($\sqcup$), ($\exists R$) to the current node v. The set $\mathcal{D}$ contains the contents of the resulting denominators of v. After the application of either one of the rules in every iteration, the algorithm creates the required successor in the and-or graph G only if it does not exist in the graph; otherwise an edge of the node is added to the set of edges E for the previously created node for tracking reachability purpose without creating another redundant node for the same content. Thus, global caching is a built-in property of this algorithm. If there is a clash between some atomic concepts for a node containing both A and $\neg A$, the bottom rule ($\bot$) applies and A becomes an end node with status set to *unsat*. On the other hand, when no more rules can be applied to the node, it becomes an end node with status set to *sat*. If the status of a node v cannot be determined to be either *sat* or *unsat* in the loop iteration, its status is simply set to *expanded*. In the step *h* and *i*, if the iteration can determine the status of v, this information of either either *sat* (satisfiability) or *unsat* (unsatisfiability) is propagated to the predecessor of v in G via the *propagate* routine as shown here 6.

Procedure $propagate(G, v)$
Parameters: an and-or graph $G = \langle V, E\rangle$ and $v \in V$ with $v.status \in \{\texttt{sat}, \texttt{unsat}\}$
Returns: a modified and-or graph $G = \langle V, E\rangle$

1. $queue := \{v\}$;
2. while $queue$ is not empty do
3. (a) extract x from $queue$;
   (b) for every $u \in V$ with $(u, x) \in E$ and $u.status = \texttt{expanded}$ do
      i. if ($u.kind = \texttt{or-node}$ and one of the successors of u has status $\texttt{sat}$)
         or ($u.kind = \texttt{and-node}$ and all the successors of u have status $\texttt{sat}$) then
            $u.status := \texttt{sat}$, $queue := queue \cup \{u\}$
      ii. else if ($u.kind = \texttt{and-node}$ and one of the successors of u has status $\texttt{unsat}$)
         or ($u.kind = \texttt{or-node}$ and all the successors of u have status $\texttt{unsat}$) then
            $u.status := \texttt{unsat}$, $queue := queue \cup \{u\}$;

Figure 6: The *propagate* routine used for the $\mathcal{ALC}$ decision algorithm [?] [5]

The main loop ends when the status of τ, the root node, becomes *sat* or *unsat* or all nodes of the graph have been expanded. In the latter case, all nodes with status not set to *unsat* will be set to *sat* in step 3 in the algorithm (mimicking assigning status *open* to all tableau branches of the tableau for which the algorithm loops). The *propagate* routine as shown in 6 gets the parameter: the graph G and a node of status either *sat* or *unsat* as input. It propagates the irrevocable status of v through the and-or graph and updates G with new information.

9.2. EXPTIME $\mathcal{SHI}$ Tableau Algorithm

On the basis of the $\mathcal{ALC}$ tableau decision algorithm described in the previous section, Goré and Nguyen have proposed another decision algorithm for $\mathcal{SHI}$. Similar to the $\mathcal{ALC}$ decision algorithm, the $\mathcal{SHI}$ algorithm shown in figure 7 also constructs and and-or graph with nodes keeping track of the same set of attributes:
content: the set of concepts associated with the node

status: { *unexpanded*, *expanded*, *sat*, *unsat* }
kind : { *and-node*, *or-node* }

An extension has been introduced to the algorithm to account for transitive, inverse roles and roles hierarchies contained in a $\mathcal{SHI}$ RBox $\mathcal{R}$. For checking whether a given finite set of concepts X is satisfiable w.r.t. $(\mathcal{R}, \Gamma)$, where $\mathcal{R}$ is the RBox and Γ the TBox, the content of the root node τ with status set to *unexpanded* is assigned with $X \cup \Gamma$.

Input: a $\mathcal{SHI}$ RBox $\mathcal{R}$, a TBox $\mathcal{T}$, and a finite set X of concepts
Output: an and-or graph $G = \langle V, E \rangle$ with $\tau \in V$ as the initial node such that $\tau.status = \texttt{sat}$ iff X is satisfiable w.r.t. $(\mathcal{R}, \mathcal{T})$

1. create a new node τ with $\tau.content := X \cup \mathcal{T}$ and $\tau.status := \texttt{unexpanded}$; let $V := \{\tau\}$ and $E := \emptyset$;
2. while $\tau.status \notin \{\texttt{sat}, \texttt{unsat}\}$ and we can choose an unexpanded node $v \in V$ do:
   (a) $\mathcal{D} := \emptyset$;
   (b) if no $\mathcal{CSHI}$-tableau rule is applicable to $v.content$ then $v.status := \texttt{sat}$
   (c) else if $(\bot)$ is applicable to $v.content$ then $v.status := \texttt{unsat}$
   (d) else if $(\sqcap)$ or (H) or $(H^\dagger)$ is applicable to $v.content$ giving denominator Y then
   $v.kind := \texttt{and-node}$, $\mathcal{D} := \{Y\}$
   (e) else if $(\sqcup)$ or $(cut_\forall)$ or (cut_B) or (cut_5) is applicable to $v.content$ giving denominators Y_1 and Y_2 (both different from $v.content$) then
   $v.kind := \texttt{or-node}$, $\mathcal{D} := \{Y_1, Y_2\}$
   (f) else
   i. $v.kind := \texttt{and-node}$,
   ii. for every $\exists R.C \in v.content$, apply $(\exists R)$ to $v.content$ giving denominator $trans_{\mathcal{R}}(v.content, R) \cup \{C\} \cup \mathcal{T}$ and add this denominator to $\mathcal{D}$;
   (g) for every denominator $Y \in \mathcal{D}$ do
   i. if some (proxy) $w \in V$ has $w.content = Y$ then add edge (v, w) to E
   ii. else let w be a new node, set $w.content := Y$, $w.status := \texttt{unexpanded}$, add w to V, and add edge (v, w) to E;
   (h) if ($v.kind = \texttt{or-node}$ and one of the successors of v has status $\texttt{sat}$) or ($v.kind = \texttt{and-node}$ and all the successors of v have status $\texttt{sat}$) then
   $v.status := \texttt{sat}$, $propagate(G, v)$
   (i) else if ($v.kind = \texttt{and-node}$ and one of the successors of v has status $\texttt{unsat}$) or ($v.kind = \texttt{or-node}$ and all the successors of v have status $\texttt{unsat}$) then
   $v.status := \texttt{unsat}$, $propagate(G, v)$
   (j) else $v.status := \texttt{expanded}$;
3. if $\tau.status \notin \{\texttt{sat}, \texttt{unsat}\}$ then
   for every node $v \in V$ with $v.status \neq \texttt{unsat}$, set $v.status := \texttt{sat}$;

Figure 7: Tableau decision algorithm for $\mathcal{SHI}$ [?] [4]

Similarly, the main loop of the algorithm continues processing until the status of τ is determined to be either *sat* or *unsat*, or until every node has been expanded. Inside the main loop. from the step 2b to 2f, the series of $\mathcal{SHI}$ tableau rules as described in 4 must be applied in the order $(\bot)$, $(\sqcap)$, (H), $(H^\dagger)$, $(\sqcup)$, $(cut_\forall)$, (cut_B), (cut_5), $(\exists R)$ to the current node v. The set $\mathcal{D}$ contains the contents of the resulting denominators of v and the set E stores the set of edges connection expanded nodes with newly created ones. If the $(\sqcap)$, (H), $(H^\dagger)$ rules are applicable, v has one denominator in $\mathcal{D}$; if the $(\sqcup)$, $(cut_\forall)$, (cut_B), (cut_5) rules are applicable, v has two de-

nominator in $\mathcal{D}$. The $(\exists R)$ rule gives one denominator to $\mathcal{D}$ for each concept $\exists R.C \in v.content$.

Here global caching is also used for creating successor in the graph G only if it has not been created already. The difference to the $\mathcal{ALC}$ algorithm is that in step (d), application of the (H), $(H^{\dagger})$ rules also contributes to making the node become an and-node beside just the $(\sqcap)$ rule; in step (e) the application of the $(cut_{\forall})$, (cut_B), (cut_5) rules contributes to making the node become an or-node beside the $(\sqcup)$ rule. Steps (2h) and (2i), analogous to the corresponding step in the $\mathcal{ALC}$ tableau decision algorithm, tries to compute the status of a node v to determine whether it is *sat* or *unsat*. If status cannot be determined to be either of these, the status of v is set to *expanded*; otherwise, again, if the status can be determined to be either *sat* or *unsat*, the propagate routine as shown in 8 updates the information of the predecessor of v with new information.

Procedure $propagate(G, v)$
Parameters: an and-or graph $G = \langle V, E\rangle$ and $v \in V$ with $v.status \in \{\texttt{sat}, \texttt{unsat}\}$
Returns: a modified and-or graph $G = \langle V, E\rangle$

1. $queue := \{v\}$;
2. while $queue$ is not empty do
3. (a) extract x from $queue$;
   (b) for every $u \in V$ with $(u, x) \in E$ and $u.status = \texttt{expanded}$ do
      i. if ($u.kind = \texttt{or-node}$ and one of the successors of u has status $\texttt{sat}$)
         or ($u.kind = \texttt{and-node}$ and all the successors of u have status $\texttt{sat}$) then
            $u.status := \texttt{sat}$, $queue := queue \cup \{u\}$
      ii. else if ($u.kind = \texttt{and-node}$ and one of the successors of u has status $\texttt{unsat}$)
         or ($u.kind = \texttt{or-node}$ and all the successors of u have status $\texttt{unsat}$) then
            $u.status := \texttt{unsat}$, $queue := queue \cup \{u\}$;

Figure 8: The *propagate* routine used for the $\mathcal{SHI}$ decision algorithm [**?**] [4]

The main loop ends when the status of the root node τ becomes either *sat* or *unsat* or all nodes have been expanded. In the latter case, all nodes which do not have status *unsat* will be set to status *sat*.

9.3. Satisfiability Algorithm in RDF Triple Store

With regard to the RDF triple store, the both proposed tableau algorithms are intriguing in the sense that an efficient satisfiability checking of concepts store in the triple store means find possible clashes of concepts efficiently. Since the RDF triple store contains RDF triples and that the arbitrary concepts are also stored as triples, it is obvious that a query language must be necessary in order to interact with these triples in the triple store. As introduced in section 7.1 and **??**, by realizing the propose algorithms described in section 9.1 and 9.2, it is obvious that the SPARQL query language for RDF must be used to realize the necessary *tableau rules* first, in order to apply these tableau rules to the persistent concepts data in the RDF triple store to check satisfiability of these concepts w.r.t. given TBox Γ.

Part IV.
Analysis and Design

10. Analysis

10.1. Requirements Analysis

In order to check satisfiability of concepts w.r.t. TBox and RBox as described previously in the proposed algorithms, it is necessary to implement a reasoner prototype, a type of satisfiability checker for both $\mathcal{ALC}$ and $\mathcal{SHI}$ types of TBox and concepts; with the $\mathcal{ALC}$ type of TBox containing a subset of concepts found in a $\mathcal{SHI}$ type of TBox. Thereby finding out clashes between concepts in the triple store will be an important operation of such reasoner using previously described tableau decision algorithms for $\mathcal{ALC}$ and $\mathcal{SHI}$.

The fact that TBox concepts are modeled using description logics constructors and must be available in negation normal form is an important premise to the requirement analysis of the reasoner. This means that the reasoner must be able to import a set of concepts and the TBox modeled in the requirement normal form. It must transform *equivalence definitions* [1] to *subsumption*, for instance, $C \equiv D \implies (C \sqsubseteq D) \sqcap (D \sqsubseteq C)$; and *subsumption* of concepts to *conjunction* ($\sqcap$), *disjunction* ($\sqcup$) and *negated* ($\neg$) concepts, for instance, $C \sqsubseteq D \implies C \sqcap \neg D$. Additionally, it must support complex expressions made up of arbitrary combination of concepts using constructors such as conjunction ($\sqcap$), disjunction ($\sqcup$) and negation ($\neg$) of arbitrary concepts since these are the fundamentally available premises in the negation normal form of the concept axioms modeled in the TBox. Consequently the reasoner must find a way to load in the concepts to check for satisfiability w.r.t. TBox. It turns out that such concepts should also be in the correctly transformed logic form for efficient processing purpose.

It turns out that the **OWL-DL** language by W3C is suitable for modeling the concepts since it has the necessary constructors that correspond to the description logics constructor and it allows modeling of both $\mathcal{ALC}$ and $\mathcal{SHI}$ concepts directly with its language constructors. It enhances OWL-Lite in some of the important aspects such as OWL-DL allows union and negation of class concepts which OWL-Lite does not for OWL-Lite only supports, for instance, the intersection of class concepts. Some of the $\mathcal{ALC}$ and $\mathcal{SHI}$ description logics expressions can be model in OWL-DL very intuitively, as shown in table 1. Therefore, it is required that the reason should handle TBox and concepts definition formulated as OWL-DL expressions or their logical combinations instead of handling the description logics symbolic expressions such as C, D directly. For a simple reason, since the OWL and RDF serialization format is generally XML-based [2] and there is proved API and libraries for dealing with OWL and RDF XML serialization formats, it is convenient for the reasoner to load these concepts definitions in RDF/XML format easily and correctly.

On the other hand, since OWL builds on the provision of RDF and RDF schema, it means that OWL modeled concepts with OWL constructors are different than RDF data. The OWL

[1] referring to concept definitions within a TBox
[2] W3C recommends RDF/XML as OWL serialization format

modeling language is more expressive as a modeling language than RDF and RDF schema can provide together. Therefore, OWL constructors and concept expressions are not necessarily directly mappable to RDF syntax considering that the concepts must be stored as RDF data inside a triple store because a triple store only stores RDF triples, i.e. tuples consisting of subject, predicate and object. In fact it is one of the major requirements that normalization of the concepts must be performed on the imported concepts. It means that it is necessary from the perspective of the triple store and RDF triples storage to transform the OWL constructors somehow into a set of RDF triples which when combined together must reflect the *semantics* of the concepts of the OWL constructors as accurate as possible; consequently it is required to "add" some extra *metadata* to the set of RDF triples corresponding to the respective OWL concepts so that the semantics of the concepts and processing information for the implementation tableau decision algorithm in the triple store can be stored together for further processing. It is necessary to do so because in terms of syntax, a RDF triple us rather simple: the RDF model is graph based rather than concept based. A triple simply relates certain subject with certain object via a binary predicate. Therefore it allows a rather large degree of freedom on what to assign to the subject, predicate and object of an RDF triple. But since the reasoner must handle concepts that are concerned with a semantics model rather than simple subject-object relationships, there is a discrepancy between the two types of models. Consequently complex OWL concepts, for instance, *subClassOf*, *subPropertyOf*, *EquivalentClass*, *EquivalentProperty*, etc. must be "mapped" in a way so that the "excess" information contained in the OWL constructors will be translated into *completely* without loss of information. This requirement will be discussed in the subsequent section on design.

Furthermore, the reasoner should support parsing the OWL constructors which are corresponding allowed concepts, roles and constructors in $\mathcal{ALC}$ and $\mathcal{SHI}$. Common but important OWL-DL constructors such as *intersectionOf*, *unionOf* and *complementOf* are required to be parsed correctly. The concept expressions can get more complex by nested combination of such constructors. In terms of RDF/XML serialization of the OWL concepts, the parser is required to handle the corresponding XML elements correctly, adding to the triple store the accurate and correction metadata about the OWL concepts.

Since the reasoner uses tableau algorithm, the described *tableau rules* must be implemented toward persistent storage of a triple store instead of main memory data structures. However, because the RDF triple store must be search for in order to find out the set of RDF data using RDF graph pattern matching, it is necessary for the reasoner to query the RDF triple store using a query language. As described in section 7.1, SPARQL is the dedicated language for this particular purpose. Therefore the reasoner is required to query the triple store using SPARQL queries. In fact, the tableau rules themselves are applied on and-or graph nodes with concept content. This means that the tableau rules themselves must be SPARQL queries also to select for appropriate set of RDF triples from the triple store. In the course of the discussion of the two tableau decision algorithms in section 9.1 and 9.2, it turns out that the algorithm builds an and-or graph of the set of concepts to check for satisfiability; consequently, the reason must store these and-or-graph nodes and node content which are also modeled as RDF triples accordingly in the triple store.

10.2. Functional Requirements

The functional requirements of the satisfiability checking reasoner are:

1. support $\mathcal{ALC}$ satisfiability checking a set of concepts w.r.t. a TBox using propose $\mathcal{ALC}$ tableau decision algorithm 9.1,
2. support $\mathcal{SHI}$ satisfiability checking of a set concepts w.r.t. a TBox, an RBox using proposed $\mathcal{SHI}$ tableau decision algorithm 9.2,
3. support efficient discovery of concept clashes using the algorithms to build and-or graph structure for concepts to check in a triple store,
4. support RDF/XML serialized OWL-DL concepts definitions of TBox, RBox and complex concepts as input,
5. support the necessary OWL-DL concept constructors allowed by $\mathcal{ALC}$ and $\mathcal{SHI}$ description logics [3],
6. support normalization and metadata driven mapping of the OWL concepts into RDF triples for storing in a RDF triple store,
7. support satisfiability checking on these persistent RDF triples in secondary storage instead of main memory,
8. support tableau rules application via SPARQL queries toward the RDF triple store,
9. support display indicating either satisfiability or unsatisfiability, together with timing information of the decision algorithms as output.

There are some optional non-functional requirements, the reasoner:

1. should support many different RDF triple stores independent of vendors,
2. the decision process should end in worst case EXPTIME according to the proposed algorithms [4],
3. should advocate recommended standard and standardized API rather than relying on proprietary solutions,
4. should support logging for tracing processing steps,
5. should be able to let user to suppress logging if not desired,
6. should retain the RDF triples data of previous algorithm runs to facilitate reconstruction of algorithm processing.

10.3. Use Cases

Based on the previous requirements analysis, there are six use cases obtained as follows:

1. Import and parse OWL TBox and concepts for $\mathcal{ALC}$ DL
2. Import and parse OWL TBox, RBox and concepts for $\mathcal{ALC}$ DL
3. Normalize and persist OWL concepts to RDF triples

[3] refer to table 1 for information (excluding the last 3 types of constructors in the table)

[4] performance will depend on the efficiency of triple store, complexity of computation of the middleware or API layers in practice

4. Start run $\mathcal{ALC}$ tableau decision algorithm
5. Start run $\mathcal{SHI}$ tableau decision algorithm
6. Output decision algorithm result

The first use case is described in table 2.

Use Case:	Parse OWL TBox and Concepts for $\mathcal{ALC}$ DL
Summary:	The reasoner imports and parses the $\mathcal{ALC}$ TBox and a finite set of $\mathcal{ALC}$ concept definitions w.r.t. TBox as OWL documents input.
Actor:	User
Description:	If the actor initiates the reasoner to check satisfiability of a set of concepts w.r.t. $\mathcal{ALC}$ TBox, the reasoner imports and corresponding OWL documents, parses them for validity and forwards them to a content handler for handling the normalization and mapping to RDF triples.
Preconditions:	OWL documents containing $\mathcal{ALC}$ TBox axioms definitions and the set of finite $\mathcal{ALC}$ concepts must be valid and well formed as specified by the OWL language specification of W3C, additionally, the OWL documents must be available in RDF/XML serialization format.
Postconditions:	OWL documents are parsed and are declared valid, ready to be forwarded to OWL content handler for further processing
Invariance:	RDF Triple store remains in idle state, the OWL documents are not modified
Exceptions:	Parser can cause exceptions indicating the following situations: • OWL documents are not found • OWL documents are malformed • OWL documents are not valid because of syntax error

Table 2: Use Case 1: Parse OWL TBox and Concepts for $\mathcal{ALC}$ DL

The second use case is described in table 3.

The third use case is described in table 4.

The fourth use case is described in table 5.

The fifth use case is described in table 6.

The last use case is described in table 7.

Use Case:	Parse OWL TBox, RBox and Concepts for $\mathcal{SHI}$ DL
Summary:	The reasoner imports and parses the $\mathcal{SHI}$ TBox and a finite set of $\mathcal{SHI}$ concept definitions w.r.t. TBox as OWL documents input.
Actor:	User
Description:	The actor initiates the reasoner to check satisfiability of a set of concepts w.r.t. $\mathcal{SHI}$ TBox and $\mathcal{SHI}$ RBox. The reasoner imports and corresponding OWL documents, parses them for validity and forwards them to a content handler for handling the normalization and mapping to RDF triples.
Preconditions:	OWL documents containing $\mathcal{SHI}$ TBox axioms definitions and the set of finite $\mathcal{SHI}$ concepts must be valid and well formed as specified by the OWL language specification of W3C, additionally, the OWL documents must be available in RDF/XML serialization format.
Postconditions:	OWL documents are parsed and are declared valid, ready to be forwarded to OWL content handler for further processing
Invariance:	RDF Triple store remains in idle state, the OWL documents are not modified
Exceptions:	Parser can cause exceptions indicating the following situations: • OWL documents are not found • OWL documents are malformed • OWL documents are not valid because of syntax error

Table 3: Use Case 2: Parse OWL TBox, RBox and Concepts for $\mathcal{SHI}$ DL

11. Design

11.1. Modeling And-Or Graph Data Structure

The tableau decision algorithms build an and-or graph structure. It is intuitively understandable that a node of the and-or graph structure must be a structure modeled with RDF triples, just as the other mapped information, since the tableau decision algorithms work toward an RDF triple store. As mentioned in section 9.1, a node contains this attributes:
content: the set of concepts associated with the node
status: { *unexpanded*, *expanded*, *sat*, *unsat* }
kind : { *and-node*, *or-node* }
The first attribute *content* is used to bind the and-or graph node with its content. In terms of the tableau algorithm, the content represents for the *root node* the set of union the the concepts to check and the TBox axioms; whilst for any intermediate *child node* of the graph, the content represents the subset of the concepts which are momentarily being processed in the current iteration; which the current node has inherited from its predecessor by the application of one of the tableau rules. On the other hand, since an and-or graph structure as such is never a flat

Use Case:	Normalize and Persist OWL Concepts to RDF Triples
Summary:	The reasoner transforms the parsed OWL constructors and maps them to a set of RDF triples
Actor:	Reasoner
Description:	Upon finishing parsing the OWL documents for well-formed syntax and validity, the reasoner normalizes and maps the OWL concept constructors to a set of RDF triples according to some predefined mapping rules by the programmer, thereby adding also metadata for construction of the and-or graph nodes containing the concepts accordingly. The reasoner inserts the set of RDF triples using an API into the triple store.
Preconditions:	OWL documents are parsed, well-formed and valid
Postconditions:	One OWL concept constructor is normalized and mapped to a set of corresponding RDF triples, these are stored in the triple store persistently.
Invariance:	The OWL documents are not modified
Exceptions:	None

Table 4: Use Case 3: Normalize and Persist OWL Concepts to RDF Triples

structure, by naively linking one and-or graph node with every concept the node contains will not be practical because this will lead to exponential growth of the number of links that must be maintained. Furthermore because of the form of an RDF triple, it can only store a piece of information containing subject, predicate and object at a time, practically relating a subject to the object with binary link, such large number of links that must be stored as RDF triples will not be manageable. This best case of such naive approach will have a node linked to only one **atomic** concept that fits exactly in the subject of the RDF triple. However, for complexer expressions, by maintaining a link to every atomic concepts constituting the expression will be prohibitively ineffective. Therefore the and-or graph structure must be *flattened* in such a way that one and-or graph node only maintain one link to its content. This means that, for instance, the RDF triple that represents an and-or graph node could be modeled as follows [7]:

Listing 12: Modeling content for an and-or graph node

```
@prefix reasoner:    <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
@prefix nodes:       <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
@prefix reasoner-anonnode: <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner#>
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>


reasoner:and-or-graph-node-1 reasoner:hasContent  reasoner-anonnode:btnode-7e257c04
    -1063-44b9-8c3d-ba559011b69a
```

The point here in the example 12, one and-or graph node can be modeled to have a predicate **reasoner:hasContent** which relates the node to a *generated anonymous graph node* in RDF. From the perspective of an and-or graph node, this is a layer of *indirection* which allows the and-or graph node to be decoupled from its content simply by removing the link connecting the and-or graph node with the anonymous node, i.e. by simply deleting this triple. This approach has two advantages: firstly, the update of the content of an and-or graph node does not need to be necessarily updating the node itself. For instance, adding more concepts to a node can be

[7] the listing represents conceptually RDF triples data in the triple store

Use Case:	Start run $\mathcal{ALC}$ Tableau Decision Algorithm
Summary:	The reasoner runs the tableau decision algorithm [5] for $\mathcal{ALC}$ concepts w.r.t. an $\mathcal{ALC}$ TBox Γ when the $\mathcal{ALC}$ mode is chosen.
Actor:	User, Reasoner
Description:	When the user has chosen to start the reasoner in $\mathcal{ALC}$ mode, the $\mathcal{ALC}$ tableau decision algorithm is started. The decision algorithm constructs a *root and-or graph node*, or simply *root node* v created for the $\mathcal{ALC}$ tableau which has the content $\Longrightarrow v.content ::= X \cup \Gamma$ where X is the set of $\mathcal{ALC}$ concepts, Γ the $\mathcal{ALC}$ TBox. The root node itself must be modeled as RDF triple data. It iterates over the concepts modeled as RDF triples in the triple store and tries to apply the tableau rules described in section 8.3 in the specific order as mentioned to the content of the root node. In this process, new and-or graph nodes can be created with their contents set to the content of the denominators as specified by the respective tableau rules. Or graph edge connecting the already existing node to applicable content is created for the purpose of global caching, i.e. to reduce redundant node creation. The algorithm continues running until either the entire tableau is *closed* (all its branches are *closed*) or when all the nodes has status ***expanded***. The root node has at termination of the algorithm either ***sat*** or ***unsat*** set as status. The former indicates the the concepts to check are satisfiable w.r.t. the TBox Γ; unsatisfiable otherwise.
Preconditions:	All $\mathcal{ALC}$ OWL concepts and $\mathcal{ALC}$ TBox concepts are correctly mapped and stored in the RDF triple store, there are additionally metadata assisting construction of the and-or graph. There is a *root and-or graph node*, or simply *root node* v created for the $\mathcal{ALC}$ tableau which has the content $\Longrightarrow v.content ::= X \cup \Gamma$, i.e. the content of the root node is set to union of the TBox and all available concepts to check w.r.t. TBox Γ.
Postconditions:	Application of the tableau rules leads to insertion of new RDF triples and updates of existing RDF triples in the triple store. The and-or graph for the $\mathcal{ALC}$ tableau is built. The root node of the $\mathcal{ALC}$ tableau has either ***sat*** or ***unsat*** as status.
Invariance:	None
Exceptions:	None

Table 5: Use Case 4: Start run $\mathcal{ALC}$ Tableau Decision Algorithm

Use Case:	Start run $\mathcal{SHI}$ Tableau Decision Algorithm
Summary:	The reasoner runs the tableau decision algorithm [6] for $\mathcal{SHI}$ concepts w.r.t. a $\mathcal{SHI}$ TBox Γ and a $\mathcal{SHI}$ RBox $\mathcal{R}$ containing inverse, transitive and sub roles when the $\mathcal{SHI}$ mode is chosen.
Actor:	User, Reasoner
Description:	When the user has chosen to start the reasoner in $\mathcal{SHI}$ mode, the $\mathcal{SHI}$ tableau decision algorithm is started. The decision algorithm constructs a *root and-or graph node*, or simply *root node* v created for the $\mathcal{SHI}$ tableau which has the content $\Longrightarrow v.content ::= X \cup \Gamma \cup R$ where X is the set of $\mathcal{SHI}$ concepts, Γ the $\mathcal{SHI}$ TBox and $\mathcal{R}$ the $\mathcal{SHI}$ RBox. The root node itself must be modeled as RDF triple data. It iterates over the concepts modeled as RDF triples in the triple store and tries to apply the tableau rules described in section 8.4 in the specific order as mentioned to the content of the root node. In this process, new and-or graph nodes can be created with their contents set to the content of the denominators as specified by the respective tableau rules. Or graph edge connecting the already existing node to applicable content is created for the purpose of global caching, i.e. to reduce redundant node creation. The algorithm continues running until either the entire tableau is *closed* (all its branches are *closed*) or when all the nodes has status ***expanded***. The root node has at termination of the algorithm either ***sat*** or ***unsat*** set as status. The former indicates the the concepts to check are satisfiable w.r.t. the TBox Γ and RBox $\mathcal{R}$; unsatisfiable otherwise.
Preconditions:	All $\mathcal{SHI}$ OWL concepts and $\mathcal{SHI}$ TBox concepts are correctly mapped and stored in the RDF triple store, there are additionally metadata assisting construction of the and-or graph. There is a *root and-or graph node*, or simply *root node* v created for the $\mathcal{SHI}$ tableau which has the content $\Longrightarrow v.content ::= X \cup \Gamma$, i.e. the content of the root node is set to union of the TBox and all available concepts to check w.r.t. TBox Γ.
Postconditions:	Application of the tableau rules leads to insertion of new RDF triples and updates of existing RDF triples in the triple store. The and-or graph for the $\mathcal{SHI}$ tableau is built. The root node of the $\mathcal{SHI}$ tableau has either ***sat*** or ***unsat*** as status.
Invariance:	None
Exceptions:	None

Table 6: Use Case 5: Start run $\mathcal{SHI}$ Tableau Decision Algorithm

Use Case:	Output Tableau Decision Algorithm Result
Summary:	Reasoner outputs result after running the tableau decision algorithm.
Actor:	User
Description:	Reasoner stores time elapsed during parsing, and running the algorithm. It outputs the time values after the algorithm terminates.
Preconditions:	The tableau decision algorithm has finished and the status of the root node of the and-or graph built is set.
Postconditions:	Results are output
Invariance:	RDF triple store remains idle
Exceptions:	None

Table 7: Use Case 6: Output Tableau Decision Algorithm Result

achieved by modifying the generated anonymous node. Secondly, the anonymous node approach is a proved way in RDF to relate a subject to more than one objects without using an RDF list structure [8].

Listing 13: Node content indirection with references to concepts

```
@prefix reasoner:    <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
@prefix nodes:       <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
@prefix contenttype: <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/contentType
    />
@prefix concept:     <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/concept#>
@prefix reasoner-anonnode: <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner#>
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>


reasoner:and-or-graph-node-1 reasoner:hasContent  reasoner-anonnode:btnode-7e257c04
    -1063-44b9-8c3d-ba559011b69a

reasoner-anonnode:btnode-7e257c04-1063-44b9-8c3d-ba559011b69a reasoner:contentType
    contenttype:atomic

reasoner-anonnode:btnode-7e257c04-1063-44b9-8c3d-ba559011b69a reasoner:hasChild
    concept:C

reasoner-anonnode:btnode-7e257c04-1063-44b9-8c3d-ba559011b69a reasoner:hasChild
    concept:D

reasoner-anonnode:btnode-7e257c04-1063-44b9-8c3d-ba559011b69a reasoner:hasChild
    concept:F
```

Intuitively, atomic concepts or concept expression made up of atomic concepts which are content of the and-or graph node can simply be modeled as further RDF triples which have the URI of the generated anonymous node as subject; as shown in the example 13. Notice that in the listing examples given here, a prefix notation is used so that it is more neat to display the information. Beside the prefix notation, notice also that the RDF data model prescribes what is store in the place of the subject, predicate and object. To reiterate, an subject contains URI[9] (in the form of IRI), a predicate also and an object can contain URI, literal value, or data typed literal. Example 13 also shows that atomic concepts can be associated with the generated anonymous node with the predicate **reasoner:hasChild**. The contenttype:contentType predicate relates the

[8] considering the overhead of extra computation to process list structure, also the tableau algorithm does not prescribe list processing

[9] if the URI is written out completely, it must be enclosed between a pair of <>

anonymous node to a type of the node content for internal processing, a kind of metadata consequently for the node.

The other two attributes: *status* and *kind* of an and-or graph node can be modeled analogously with the respective predicates in an RDF triple. There is here no need for an indirection layer since the status and kind attributes have straightforward flat values. Example 14 shows the **reasoner:hasStatus** predicate.

Listing 14: Modeling node status for and-or graph nodes

```
@prefix reasoner:    <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
@prefix nodes:       <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
@prefix status:      <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodeStatus
    />
@prefix reasoner-anonnode: <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner#>
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>


reasoner:and-or-graph-node-1  reasoner:nodeStatus  status:unexpanded

reasoner:and-or-graph-node-2  reasoner:nodeStatus  status:expanded

reasoner:and-or-graph-node-3  reasoner:nodeStatus  status:unexpanded

reasoner:and-or-graph-node-4  reasoner:nodeStatus  status:unexpanded

reasoner:and-or-graph-node-5  reasoner:nodeStatus  status:unsat
```

With the **reasoner:nodeType** predicate shown in example 15, it relates an and-or graph node to its type: either and-node or or-node.

Listing 15: Modeling node type for and-or graph nodes

```
@prefix reasoner:    <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
@prefix nodes:       <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
@prefix status:      <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodeStatus
    />
@prefix type:        <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodeType/>
@prefix reasoner-anonnode: <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner#>
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>


reasoner:and-or-graph-node-1  reasoner:nodeStatus  status:unexpanded
reasoner:and-or-graph-node-1  reasoner:nodeType  type:and

reasoner:and-or-graph-node-2  reasoner:nodeStatus  status:expanded
reasoner:and-or-graph-node-2  reasoner:nodeType  type:and

reasoner:and-or-graph-node-3  reasoner:nodeStatus  status:unexpanded
reasoner:and-or-graph-node-3  reasoner:nodeType  type:or

reasoner:and-or-graph-node-4  reasoner:nodeStatus  status:unexpanded
reasoner:and-or-graph-node-4  reasoner:nodeType  type:or

reasoner:and-or-graph-node-5  reasoner:nodeStatus  status:unsat
reasoner:and-or-graph-node-5  reasoner:nodeType  type:and
```

Beside this design approach to model the necessary mentioned attributes for an and-or graph node, there are two other issues concerning additional metadata for an and-or graph node:

1. an additional *reasoner:contentType* predicate for an and-or graph node should be possible because it is possible for the node to refer to concept directly instead of via an indirect anonymous node as described previously. It this case, the and-or graph node takes the predicate and takes as object the URI directly referencing, for instance, an atomic concept;

2. it is also necessary to model an and-or graph node with a predicate relating it to, for instance a predecessor. The reason lies in the fact the the propose algorithm requires the propagation of both satisfiability or unsatisfiability information. With this in mind, it is obvious that it is necessary to navigate from one node to another one via some link. In RDF triple term, such property can be modeled as triples such as the example 16 shows. Here double-linked nodes are therefore in both directions *traversable* when one navigates form a predecessor to its successor or vice versa.

Listing 16: Modeling successor and predecessor links for and-or graph nodes

```
@prefix reasoner:    <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
@prefix nodes:       <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodes/>
@prefix status:      <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodeStatus
    />
@prefix type:        <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/nodeType/>
@prefix reasoner-anonnode: <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner#>
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>


reasoner:and-or-graph-node-1  reasoner:hasSuccessor   reasoner:and-or-graph-node-2
reasoner:and-or-graph-node-1  reasoner:hasPredecessor reasoner:and-or-graph-node-0

reasoner:and-or-graph-node-2  reasoner:hasSuccessor   reasoner:and-or-graph-node-7
reasoner:and-or-graph-node-2  reasoner:hasPredecessor reasoner:and-or-graph-node-1

reasoner:and-or-graph-node-3  reasoner:hasSuccessor   reasoner:and-or-graph-node-5
reasoner:and-or-graph-node-3  reasoner:hasPredecessor reasoner:and-or-graph-node-1

reasoner:and-or-graph-node-5  reasoner:hasSuccessor   reasoner:and-or-graph-node-4
reasoner:and-or-graph-node-5  reasoner:hasPredecessor reasoner:and-or-graph-node-1
```

The latter discussed double-links issue between two and-or graph nodes modeled as RDF predicate using *reasoner:hasSuccessor* and *reasoner:hasPredecessor* is especially intriguing because the propagation of satisfiability or unsatisfiability information of a node usually pops this piece of information *upwards* toward the root node. By navigating from an and-or graph node to its predecessor using RDF pattern matching of all the predecessors of the current node for instance is very convenient.

11.2. Normalization and Mapping to RDF

The OWL-DL language is a powerful ontology concept modeling language. It has many language constructors. To reiterate the necessity of introducing metadata in the process of normalization of the OWL concepts, i.e. transforming the *semantics* conveyed in a specific OWL language constructor[10] to a set of appropriate meaningful RDF triples which *reflect* faithfully to the intended semantics, thereby facilitating the processing of this information for the tableau represented as an and-or graph, can be a daunting task. Since the normalization process must be followed with a mapping procedure that relates to the corresponding OWL constructor and its associated concept a set of RDF triples that map, for instance, the value domain of the OWL concept to RDF triple; nevertheless, even the value itself should be properly mapped if it is typed. This is obvious if one considers that a typed value does not only have a *value* but also very often a semantics attached to the value for interpretation. To put it in another way a weight unit consists of a value as well as a unit for interpretation of the magnitude of the value of the weight unit, conveying a semantics of what is it about the value. The mapping process must be designed in such a way so that the set of RDF triples that is relevant to the semantics associated with one OWL constructor get attached to the main RDF triple which represents the mapped

[10] can be conceived as a language element, e.g. owl:Class, owl:inverseOf, etc.

constructor itself; often times it is necessary to introduce anonymous RDF node in order to indirectly attach a set of relevant RDF triple via the anonymous RDF node back to the main mapped triple, making the mapping thus complete.

With the mentioned rationale in mind, if a mapping process from OWL constructors to set of RDF triples is not faithful, there will be a false semantics introduced into the meaning of the OWL constructor and the concept which it defines. Firstly, one can start the normalization by adopting the following design approach start from a simple OWL constructor: owl:Class which, in its simplest form, simply introduces an *atomic* class concept with an associated name for the defined class concept.

Listing 17: A simple owl:Class definition in an OWL document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
     <!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
     <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
     <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#" >
     <!ENTITY owl  "http://www.w3.org/2002/07/owl#" >
]>
<rdf:RDF
  xmlns     ="&owl;"
  xmlns:owl ="&owl;"
  xmlns:rdf ="&rdf;"
  xmlns:rdfs="&rdfs;"
>
   <owl:Class rdf:ID="Human"/>
</rdf:RDF>
```

The normalization of an OWL constructor semantically always result in more than one RDF triple can be understood as follows, for instance, take the simple *owl:Class* constructor given in listing example 17 represented in the RDF/XML serialization format, the owl:Class is defined with the name *Human*. One way to map this to RDF triples that preserve the semantics as discussed previously could be the approach shown in listing example 18 in RDF data display format as found in the triple store.

Listing 18: First approach in class normalization

```
@prefix reasoner:    <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner#>
@prefix naming:      <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/naming/>
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#>
@prefix owl       <http://www.w3.org/2002/07/owl#>


reasoner:Human rdf:type    owl:Class
reasoner:Human naming:className  "Human"
```

The first approach normalizes the class constructor defining the class called *Human* into two RDF triples. The simple approach is appropriate because it captures what is defined with the OWL class constructor syntax. However, one can go from here further and try to amend this approach by adding another triple as shown in listing example 19.

Listing 19: Another approach in class normalization

```
@prefix reasoner:    <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner#>
@prefix naming:      <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/naming/>
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#>
@prefix owl       <http://www.w3.org/2002/07/owl#>
```

```
reasoner:Human rdf:type    owl:Class
reasoner:Human naming:className  "Human"
reasoner:Human naming:classNameTyped   "Human"^^xsd:string
```

The point here is saying additionally that the class *Human* has a name with string value. In fact, this is a little contrived here since a string value for the class name is not actually what is important in the original class constructor. However, this is not the end of the attempt to find out other design possibilities. Further investigation of the semantics of this simple class constructor reveals that there is presumably a "hidden" semantics within it: the simple OWL class constructor defines an *atomic* class concept. Therefore, it is also desirable to bring in this aspect of the semantics to the normalization. Listing example 20 shows the design approach of adding more *metadata* as RDF triples that not only preserve the semantics of the original OWL constructor but also enhance it with additional mapping information to facilitate subsequent processing.

Listing 20: An approach in class normalization together with mapped metadata

```
@prefix reasoner:    <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner#>
@prefix content:     <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/>
@prefix contenttype: <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/contentType
    />
@prefix naming:      <http://www.sts.tu-harburg.de/dl/gore/exp-owl-reasoner/naming/>
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#>
@prefix owl       <http://www.w3.org/2002/07/owl#>


reasoner:Human rdf:type    owl:Class
reasoner:Human naming:className  "Human"
reasoner:Human content:contentReference   content:class-ctnode-2ec7c4bd-a9fa-4a22-a286
    -5ddbbbb369b7
content:class-ctnode-2ec7c4bd-a9fa-4a22-a286-5ddbbbb369b7   content:contentType
    contenttype:atomic
content:class-ctnode-2ec7c4bd-a9fa-4a22-a286-5ddbbbb369b7   content:ctermValue   "Human
    "^^xsd:string
```

The third example approach shows the introduction of an anonymous RDF node to link the mapped class URI to further set of RDF triples containing additional metadata via the *content:contentReference* predicate.

The examples shown are only of simple nature. But it is not difficult to grasp the idea of normalization and mapping semantics faithfully into RDF data. This simple design approach evaluation is meant to show and explain the possibilities in different design approaches. In order to adopt a consistent approach for all the other OWL constructors, the ones which are relevant to the design for supporting $\mathcal{ALC}$ and $\mathcal{SHI}$ concepts are chosen and shown in table 8. The OWL constructors which must be mapped to RDF triples indicating their *types* are shown with the corresponding namespaces which is stored in the object of RDF triples, linked to the subject via the *rdf:type* predicate as described in the previous examples.

OWL Constructors (subject)	**Linking Predicate** (predicate)	**Namespace** (object)
owl:Class	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
owl:onProperty	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#onProperty
owl:someValuesFrom	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#someValuesFrom
owl:allValuesFrom	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#allValuesFrom
owl:ObjectProperty	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#allValuesFrom
owl:DatatypeProperty	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
rdfs:domain	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2000/01/rdf-schema#domain
rdfs:range	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2000/01/rdf-schema#range
rdfs:subClassOf	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2000/01/rdf-schema#subClassOf
owl:equivalentClass	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#equivalentClass
rdfs:subPropertyOf	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2000/01/rdf-schema#subPropertyOf
owl:inverseOf	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#inverseOf
owl:equivalentProperty	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#equivalentProperty
owl:TransitiveProperty	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#TransitiveProperty
owl:Restriction	N/A	(restrictions treated together with owl:onProperty)
owl:intersectionOf	N/A	(normalized into *and* contentType node metadata)
owl:unionOf	N/A	(normalized into *or* contentType node metadata)
owl:complementOf	N/A	(normalized into *not* contentType node metadata)

Table 8: OWL language constructor[11] with corresponding mapped namespaces as RDF triples

OWL Constructors (subject)	**Linking Predicate** (predicate)	**Namespace URI** (object)
owl:Class	reasoner:contentType	contenttype:atomic
	reasoner:contentReference	reasoner:generated-uuid-content-ref-node
owl:ObjectProperty	reasoner:contentType	contenttype:ObjectProperty
	reasoner:contentReference	reasoner:generated-uuid-property-ref-node
	rdfs:domain	domain-concept-literal
	rdfs:range	range-concept-literal
owl:DatatypeProperty	reasoner:contentType	contenttype:DatatypeProperty
	reasoner:contentReference	reasoner:generated-uuid-property-ref-node
	rdfs:domain	domain-concept-literal
rdfs:subClassOf	rdfs:subClassOf	parent-class-literal
	reasoner:contentType	contenttype:subClass
	reasoner:hasSubClass	subclass-literal
	reasoner:hasSuperClass	parent-class-literal
owl:equivalentClass	reasoner:contentType	contenttype:equivalentClass
	reasoner:isEquivalentClassOf	equivalent-class-literal
	reasoner:hasEquivalentClass	mapped-class-literal
rdfs:subPropertyOf	reasoner:contentType	contenttype:subProperty
	reasoner:hasSuperProperty	parent-property-literal
	reasoner:hasSubProperty	mapped-property-literal

Table 9: Mapped **metadata** as RDF triples (Part1)
URI-prefixes: (reasoner:=http://www.sts.tu-harburg.de/dl/gore/reasoner/),(rdfs:=http://www.w3.org/2000/01/rdf-schema#), (contenttype:=http://www.sts.tu-harburg.de/dl/gore/reasoner/contentType),(owl:=http://www.w3.org/2002/07/owl#)

OWL Constructors (subject)	**Linking Predicate** (predicate)	**Namespace URI** (object)
owl:inverseOf	reasoner:contentType	contenttype:inverseProperty
	reasoner:isInversusPropertyOf	rectus-property-literal
	reasoner:hasInversusProperty	mapped-property-literal
owl:equivalentProperty	reasoner:contentType	contenttype:equivalentProperty
	reasoner:isEquivalentPropertyOf	equivalent-property-literal
	reasoner:hasEquivalentProperty	mapped-property-literal
owl:someValuesFrom	reasoner:contentType	contenttype:someValuesFromRestriction
owl:allValuesFrom	reasoner:contentType	contenttype:allValuesFromRestriction
owl:TransitiveProperty	reasoner:contentType	contenttype:TransitiveProperty
	reasoner:isTransitivePropertyOf	transitive-property-literal
	reasonre:hasTransitiveProperty	mapped-property-literal
owl:Restriction	reasoner:contentType	contenttype:restriction
	rdf:type	owl:anonymousClass (restriction is not a class)
owl:intersectionOf	reasoner:contentType	contenttype:and
	rdf:type	owl:anonymousClass (restriction is not a class)
owl:unionOf	reasoner:contentType	contenttype:or
	rdf:type	owl:anonymousClass (restriction is not a class)
owl:complementOf	reasoner:contentType	contenttype:not
	reasoner:contentType	contenttype:inverseClass
	reasoner:isComplementClassOf	rectus-class-literal
	reasoner:hasComplementClass	mapped-class-literal

Table 10: Mapped **metadata** as RDF triples (Part2)
URI-prefixes: (reasoner:=http://www.sts.tu-harburg.de/dl/gore/reasoner/),(rdfs:=http://www.w3.org/2000/01/rdf-schema#), (contenttype:=http://www.sts.tu-harburg.de/dl/gore/reasoner/contentType),(owl:=http://www.w3.org/2002/07/owl#)

The approach of mapping to the other metadata is shown for the selected constructors in table 9 and 10 that are relevant to the design, consult the source code for details.

11.3. Software Frameworks

There are a collection of triple stores currently available. Some of them being open source and others are commercially licensed. One of the design principle should be platform independence regarding specific triple store implementations. In the arena of open source implementations of triple store or middleware which can access triple stores, there are some projects such as the Sesame framework or Mulgara are themselves triple stores. Until recently, these triple stores support some different query languages for RDF data stored which means that no SPARQL support is available currently. The Jena framework is a modularized RDF frameworks which does not contain a triple store itself but can store RDF graphs in a relational database management system. It contains an API which supports standardized SPARQL query conforming to W3C recommendation. Therefore it provides the greatest degree of independence in terms of coupling to a triple store. The Jena SDB is a Jean framework component that can be connected to a database and support SPARQL query for the RDF triples directly. Many different databases by different vendors are supported and can be used.

From the perspective of modularized design, it is desired that only the necessary framework components are used when prototyping the reasoner. Jena provides some convenience also in the parsing of OWL documents to RDF triples. However, it is necessary to build a custom content handler to implement the normalization and mapping to required set of metadata as described in the previous section 11.2.

11.4. System Architecture

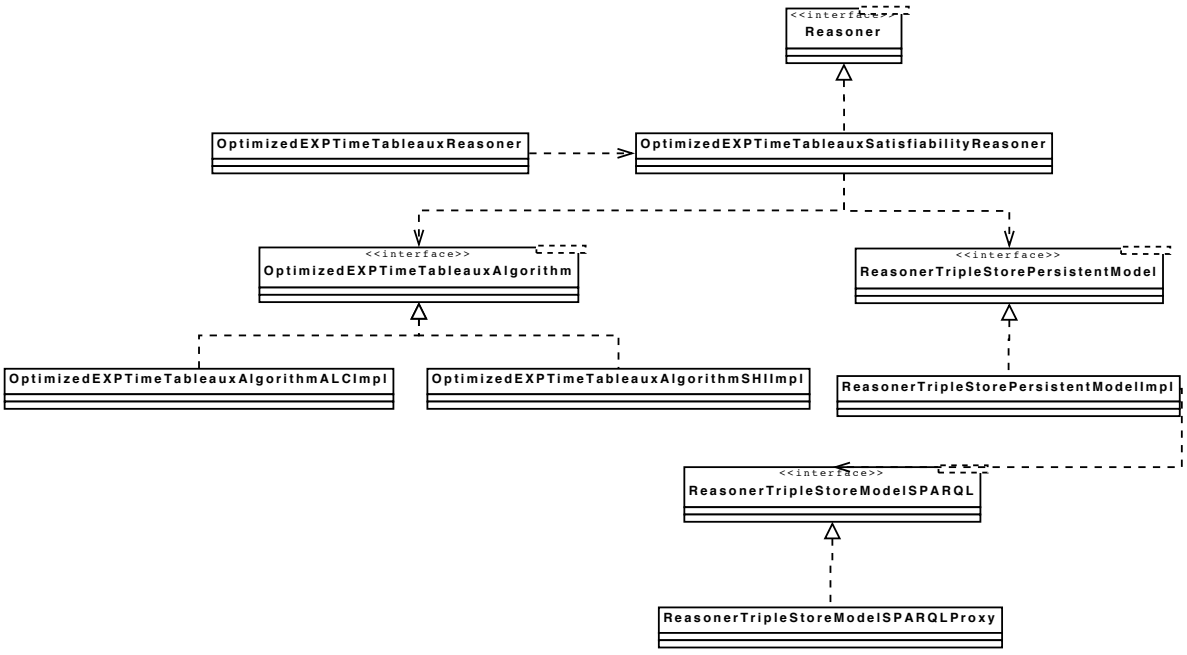

Figure 9: Class diagram of the reasoner

The class diagram 9 shows an overview of classes which the reasoner has. It is a layer ar-

chitecture consisting of mainly of a persistent layer containing implementation of the operations toward a triple store, thereby encapsulating the upper reasoner algorithm implementation layer from the details of triple store access and RDF triples manipulations. Furthermore, the persistent layer is designed to support SPARQL query, implementing the required software components which support SPARQL via an API on a query statement basis.

Part V.
Implementation

12. Implementation

This part of the thesis describes the implementation issues of the reasoner. It consists of an account on the topics of implementation of the parsing, normalization and mapping logic from OWL document to RDF triples. A custom content handler implements the required program logic and it is associated with the OWL parsing API of the Jena framework at run time. Another issue of implementation interest is set of SPARQL queries which implement the rules. These issues are discussed in the following. The appendix VII contains additional code excerpts.

12.1. RDF Framework Choice

Jena is the key API used for implementing the persistent layer for the reasoner. It provides a notion called a *Model*. a *Model* is the Java interface which abstracts away a collection of the triple store relevant methods. There are a series of sub-interfaces which extend the *Model* interface. One is called *InfModel* which is decidated interface for methods concerning inference reasoning; while another sub-interface is called *OntModel* which is concerned of ontology authoring and reasoning API. Since the reasoner relies on a persistent store, an interface or an implementation class that encapsulates trip *InfModel*, such for instance, the operations to open and close connection to a triple store are encapsulated in the class *ModelRDB*. This is the class that connects the Jena framework instance with a data source when used with a relational database. It provides static method to *create* an instance of a model. A *ModelRDB* inherits from another interface that it implements called *ModelCon* in Jena the methods to manipulate RDF triples such as *add* and *remove*. The application program logic of the reasoner uses the *ModelRDB* class in the following way: it opens a connection to a data source. It creates a new *ModelRDB* or open an existing one which represents the entity of the RDF data in the triple store with which it works with. Then the reasoner can proceed to add or remove triples using the mentioned method. The *ModelRDB* model hides away the data access implementation details from the reason to decouple them further from the program logic of the reasoner.

On the other hand, the Jena framework is not a triple store itself, although it can be used together with many different relational databases to form a triple store by using the relational database as a persistent layer. Jena uses a dedicated SDB component for accomplishing that. For SPARQL query language support, it uses the ARQ component which implements a series of interface that encapsulates methods to support SPARQL query. The main interfaces of this component that are relevant to the implementation of the reasoner are: *DataSource*, *QueryExecution*, *QuerySolution* and *ResultSet*. Furthermore, the class *Query* of the ARQ component is used by the reasoner to model a SPARQL query. This approach resembles a statement-based API; SPARQL queries are represented by instances of the *Query* class.

As a relational database backend, the reasoner uses PostgreSQL database. Theoretically, other relational database systems that are supported can be used with the Jena framework. For configuring a different backend, the *OptimizedEXPReasonerConfiguration* class can be altered to set

different database connection parameters and a different JDBC driver.

12.2. Overview of the Implementation

The software implementation of the reasoner uses the Java programming language. It consists of the following *packages*:

- de.tuhh.sts.dl.exptime.tableaux.reasoner ⟶ contains the main reasoner class *OptimizedEXPTimeTableauxReasoner*.
- de.tuhh.sts.dl.exptime.tableaux.reasoner.config ⟶ contains a class *OptimizedEXPReasonerConfiguration* that encapsulates the important configuration parameters.
- de.tuhh.sts.dl.exptime.tableaux.reasoner.config.logging ⟶ contains the class *OptimizedEXPReasonerLogger* which is responsible for logging message processing.
- de.tuhh.sts.dl.exptime.tableaux.reasoner.model.algorithms ⟶ contains the interface *OptimizedEXPTimeTableauxAlgorithm* which states the abstract methods encapsulating, for instance, invoking the tableau alogrithm and getting run time information. The $\mathcal{ALC}$ tableau decision algorithm is implemented by the class *OptimizedEXPTimeTableauxAlgorithmALCImpl* that implements the *OptimizedEXPTimeTableauxAlgorithm* interface. Furthermore the $\mathcal{SHI}$ tableau decision algorithm is implemented by the class *OptimizedEXPTimeTableauxAlgorithmSHIImpl*.
- de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db ⟶ the interface *ReasonerTripleStorePersistentModel* defines abstract methods which are implemented by the class *ReasonerTripleStorePersistentModelImpl*, for instance, to realize the tableau rules. The reason depends on the interface *ReasonerTripleStorePersistentModel* of this package also to manipulate the and-or graph modeled and store as RDF triples. This package abstracts away the persistent storage and data manipulation details from other program logic of the reasoner.
- de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql ⟶ the interface *ReasonerTripleStoreModelSPARQL* defines methods for creation of Jena *Query* instance as described previously. In fact the methods implementing the tableau rules of the class *ReasonerTripleStorePersistentModelImpl* in the previous package depends on the methods of this interface to substitute the SPARQL query statement with parameter values in order to obtain different queries. This package abstracts away the SPARQL query details from other program logic of the reasoner. The SPARQL queries containing placeholders for substitution are defined in the class *ReasonerTripleStoreModelSPARQLQueryConstants*. The other class *ReasonerTripleStoreModelSPARQLProxy* implements the interface *ReasonerTripleStoreModelSPARQL* the substitution logic on a query statement basis.
- de.tuhh.sts.dl.exptime.tableaux.reasoner.model.rdf ⟶ this package contains a custom OWL parser interface *DomainKnowledgeBaseParsable* and implementing class *DomainKnowledgeBaseOntologyParser* which hides away the details of parsing and validating the different OWL ontology documents specifying for instance the TBox, RBox, etc. Another class *TBoxOntologyContentHandler* implements SAX parser *ContentHandler* interface. The SAX parser interface provides an *event* based parsing mechanism. When compared to other parsers APIs such as those implementing the document object model (DOM) interface, the SAX parser event based mechanism is desirable in the implementation of the

reasoner content handler because mapping rules can be applied on the OWL constructors already while the parser has not yet stopped parsing; simultaneouly mapped RDF triples can be stored in the triple store. It encapsulates the main normalization and mapping logic from constructors to RDF triples. The class *DomainKnowledgeBaseOntologyParser* depends on this content handler implementation for a custom set of rules to transform the OWL constructors semantically.

- de.tuhh.sts.dl.exptime.tableaux.reasoner.subsumption $\longrightarrow$ this package contains the class *OptimizedEXPTimeTableauxSatisfiabilityReasoner* that defines the order to run the components of the reasoner. It combines the other components together by depending on the interface methods exported by the other interfaces of the mentioned packages.

12.3. Tableau Algorithms Implementation

The reasoner can handle both $\mathcal{ALC}$ and $\mathcal{SHI}$. For the purpose of describing the implementation for the $\mathcal{SHI}$ tableau algorithm, A comprehensive example of the Java source code is shown in listing example 21. This code snippet shows the *OptimizedEXPTimeTableauxAlgorithmSHI-Impl* class which implements the proposed $\mathcal{SHI}$ tableau decision algorithm as described in section 9.2. The abbreviated form is shown here with ... in this and the subsequent examples, indicating that irrelevant code sections are removed for brevity.

Listing 21: Implementation of the proposed $\mathcal{SHI}$ tableau algorithm

```
/**
 *
 */
package de.tuhh.sts.dl.exptime.tableaux.reasoner.model.algorithms;

import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.Set;

import org.apache.log4j.Logger;

import de.tuhh.sts.dl.exptime.tableaux.reasoner.config.
    OptimizedEXPReasonerConfiguration;
import de.tuhh.sts.dl.exptime.tableaux.reasoner.config.logging.
    OptimizedEXPReasonerLogger;
import de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.
    ReasonerTripleStorePersistentModel;
import de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.
    ReasonerTripleStorePersistentModel.TRIPLE_STORE_AND_OR_GRAPH_NODE_TYPE;
import de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.
    ReasonerTripleStorePersistentModel.TRIPLE_STORE_NODE_STATUS;
import de.tuhh.sts.dl.exptime.tableaux.reasoner.optimization.ReasonerConceptNodesEdge;
import de.tuhh.sts.dl.exptime.tableaux.reasoner.optimization.
    ReasonerConceptTermGraphNode;
import de.tuhh.sts.dl.exptime.tableaux.reasoner.optimization.
    ReasonerConceptTermOptimiedCache;
import de.tuhh.sts.dl.exptime.tableaux.reasoner.optimization.
    ReasonerConceptTermOptimizerFactory;

/**
 * @author patrick_un
 *
 */
public class OptimizedEXPTimeTableauxAlgorithmSHIImpl implements
    OptimizedEXPTimeTableauxAlgorithm {

   private final static Logger log = OptimizedEXPReasonerLogger.getLogger();

   private static ReasonerTripleStorePersistentModel tripleStore = null;

```

```
private ReasonerConceptTermOptimiedCache conceptTermCache = null;

private Set<ReasonerConceptNodesEdge> graphEdgesSet = null;

private Set<ReasonerConceptTermGraphNode> graphVerticesSet = null;

private ArrayList<Set<String>> resultDenominators = null;

private long tableauxAlgorithmRunTime = 0L;

/**
 * constructor
 */
public OptimizedEXPTimeTableauxAlgorithmSHIImpl(ReasonerTripleStorePersistentModel
    tripleStore) {
   super();
   setTripleStore(tripleStore);
   setResultDenominators(new ArrayList<Set<String>>());
   setGraphEdgesSet(new LinkedHashSet<ReasonerConceptNodesEdge>());
   setGraphVerticesSet(new LinkedHashSet<ReasonerConceptTermGraphNode>());
   setConceptTermCache(ReasonerConceptTermOptimizerFactory.createCache(
       getTripleStore()));
}

@Override
public void buildAndOrGraphRootNode() {
   getLog().info("create the and-or-graph root node for all the domain knowledge
       base boxes and the domain concepts in the triple store");
   getTripleStore().buildUnionOfTopLevelSHIConcepts();
}

/* (non-Javadoc)
 * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.algorithms.
     OptimizedEXPTimeTableauxAlgorithm#invokeGoreNguyenSatifiabilityAlgorithm()
 */
@Override
public String invokeGoreNguyenSatifiabilityAlgorithm() {
   String currentGraphNode  = null;
   int counter = 0;
   long executionInitTime = System.currentTimeMillis();
   long executionProgress = 0L;

   getLog().debug("**************** main loop SHI ****************");
   while ( (!getTripleStore().verifyNodesSatOrUnsat()) && ((currentGraphNode =
       getTripleStore().getUnexpandedConceptNode()) != null) ) {
      getLog().debug("**************** loop iteration " + counter + "
          ****************");
      getLog().info("**************** processing current and-or-graph node [" +
          currentGraphNode + "] ****************");
      if (getTripleStore().bottomRuleSHI(currentGraphNode)) {
         getLog().info("SHI button rule applicable");
         getLog().debug("******************************** SHI button rule applicable
             ********************************");
         getTripleStore().setNodeStatus(currentGraphNode, TRIPLE_STORE_NODE_STATUS.
             UNSAT);
         getLog().info("set node [" + currentGraphNode + "] to unsat");
      } else if (getTripleStore().andRuleSHI(currentGraphNode, getResultDenominators
          ())) {
         getLog().info("SHI and rule applicable");
         getLog().debug("******************************** SHI and rule applicable
             ********************************");
         getTripleStore().setNodeType(currentGraphNode,
             TRIPLE_STORE_AND_OR_GRAPH_NODE_TYPE.AND_NODE);
         getLog().info("set node [" + currentGraphNode + "] type to and");
      } else if (getTripleStore().restriction_H_NegativeRuleSHI(currentGraphNode,
          getResultDenominators())) {
         getLog().info("SHI H-rectus-rule applicable");
         getLog().debug("******************************** SHI H-rectus-rule
             applicable ********************************");
```

```
            getTripleStore().setNodeType(currentGraphNode,
                TRIPLE_STORE_AND_OR_GRAPH_NODE_TYPE.AND_NODE);
            getLog().info("set node [" + currentGraphNode + "] type to and");
         } else if (getTripleStore().restriction_H_PositiveRuleSHI(currentGraphNode,
             getResultDenominators())) {
            getLog().info("SHI H-inversus-rule applicable");
            getLog().debug("******************************** SHI H-+-rule applicable
                *********************************");
            getTripleStore().setNodeType(currentGraphNode,
                TRIPLE_STORE_AND_OR_GRAPH_NODE_TYPE.AND_NODE);
            getLog().info("set node [" + currentGraphNode + "] type to and");
         } else if (getTripleStore().orRuleSHI(currentGraphNode, getResultDenominators
             ())) {
            getLog().info("SHI or rule applicable");
            getLog().debug("******************************** SHI or rule applicable
                *********************************");
            getTripleStore().setNodeType(currentGraphNode,
                TRIPLE_STORE_AND_OR_GRAPH_NODE_TYPE.OR_NODE);
            getLog().info("set node [" + currentGraphNode + "] type to or");
         } else if (getTripleStore().restrictionCut_All_RuleSHI(currentGraphNode,
             getResultDenominators())) {
            getLog().info("SHI cut-all rule applicable");
            getLog().debug("******************************** SHI cut-all rule
                applicable *********************************");
            getTripleStore().setNodeType(currentGraphNode,
                TRIPLE_STORE_AND_OR_GRAPH_NODE_TYPE.OR_NODE);
            getLog().info("set node [" + currentGraphNode + "] type to or");
         } else if (getTripleStore().restrictionCut_B_RuleSHI(currentGraphNode,
             getResultDenominators())) {
            getLog().info("SHI cut-B rule applicable");
            getLog().debug("******************************** SHI cut-B rule applicable
                *********************************");
            getTripleStore().setNodeType(currentGraphNode,
                TRIPLE_STORE_AND_OR_GRAPH_NODE_TYPE.OR_NODE);
            getLog().info("set node [" + currentGraphNode + "] type to or");
         } else if (getTripleStore().restrictionCut_Five_RuleSHI(currentGraphNode,
             getResultDenominators())) {
            getLog().info("SHI cut-5 rule applicable");
            getLog().debug("******************************** SHI cut-5 rule applicable
                *********************************");
            getTripleStore().setNodeType(currentGraphNode,
                TRIPLE_STORE_AND_OR_GRAPH_NODE_TYPE.OR_NODE);
            getLog().info("set node [" + currentGraphNode + "] type to or");
         } else {
            getTripleStore().setNodeType(currentGraphNode,
                TRIPLE_STORE_AND_OR_GRAPH_NODE_TYPE.AND_NODE);
            if (getTripleStore().restriction_Transitive_RuleSHI(currentGraphNode,
                getResultDenominators())) {
               getLog().info("SHI restriction rule applicable");
            }
         }

         for (Set<String> denominatorContents : getResultDenominators()) {
            getTripleStore().constructSuccessor(currentGraphNode, denominatorContents);
         }
         getResultDenominators().clear();

         String currentNodeStatus = getTripleStore().getNodeStatus(currentGraphNode);
         if (currentNodeStatus.equals(OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_STATUS_UNSAT_NS) ||
               currentNodeStatus.equals(OptimizedEXPReasonerConfiguration.
                   W3C_OWL_REASONER_NODE_STATUS_SAT_NS)) {
            getTripleStore().propagateStatus(currentGraphNode);
         } else {
            if (!(currentNodeStatus.equals(OptimizedEXPReasonerConfiguration.
                W3C_OWL_REASONER_NODE_STATUS_UNSAT_NS))) {
               getTripleStore().setNodeStatus(currentGraphNode,
                   TRIPLE_STORE_NODE_STATUS.EXPANDED);
            }
```

```
            }

            counter++;
        }

        String rootNodeStatus = getTripleStore().getNodeStatus(
            OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS
            );
        if ((!rootNodeStatus.equals(OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_UNSAT_NS)) &&
            (!rootNodeStatus.equals(OptimizedEXPReasonerConfiguration.
                W3C_OWL_REASONER_NODE_STATUS_SAT_NS)) &&
            rootNodeStatus.equals(OptimizedEXPReasonerConfiguration.
                W3C_OWL_REASONER_NODE_STATUS_EXP_NS)) {
            getTripleStore().setNodeStatus(OptimizedEXPReasonerConfiguration.
                W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS, TRIPLE_STORE_NODE_STATUS.SAT)
                ;
        }
        executionProgress = System.currentTimeMillis();
        setTableauxAlgorithmRunTime((executionProgress - executionInitTime));
        String status = getTripleStore().getNodeStatus(OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS);
        return status;
    }

    ...
}
```

Step 1 of the $\mathcal{SHI}$ tableau algorithm description in section 9.2 requires the creation of a *root node* for the and-or graph structure before the main loop starts. The *root node* has its content set to $\tau.content := X \cup \Gamma$ and $\tau.status :=$ **unexpanded**, as shown in the implementation of the *buildUnionOfTopLevelSHIConcepts* method in listing example 22.

Listing 22: Implementation of root node creation program logic in the *ReasonerTripleStorePersistentModelImpl* class

```
/**
 *
 */
package de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db;


/**
 * @author patrick_un
 *
 */
public class ReasonerTripleStorePersistentModelImpl implements
    ReasonerTripleStorePersistentModel {

    ...

    @Override
    public void buildUnionOfTopLevelSHIConcepts() {
        getLog().info("create the and-or-graph root node for all the domain knowledge
            base boxes and the domain concepts in the triple store");
        // insert the root graph node first into the triple store and set the necessary
            graph information for it
        add(OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS,
            OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS,
            OptimizedEXPReasonerConfiguration.
                W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_CONCEPT_NODE_NS);

        add(OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS,
            OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS,
            OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS);

        add(OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS,
            OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_NODE_STATUS_NO_SLASH_NS,
            OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_NODE_STATUS_UNEXP_NS);
```

```
      add(OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS,
         OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_NODE_TYPE_NO_SLASH_NS,
         OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_GRAPH_OR_TYPE_NODE_NS);

      add(OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS,
         OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_CONTENT_NO_SLASH_TYPE_NS,
         OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS);
      // retrieve a set of top level concept nodes dedicated to be either associated
          with tbox, rbox or concept box
      // for each of these top level concept nodes, create an edge (link) from the root
           graph node to the top level
      // concept node and add necessary hasContent, contentType information for the
          tableaux algorithm
      Query buildUnionOfTopLevelConceptsQuery = getQueryProxy().
          buildUnionOfTopLevelConcepts();
      QueryExecution queryExecSelectForBuildUnionOfTopLevelConcepts =
          QueryExecutionFactory.create(buildUnionOfTopLevelConceptsQuery,
          getTripleStoreRDBModel());
      ResultSet topLevelNodeResults = queryExecSelectForBuildUnionOfTopLevelConcepts.
          execSelect();
      while (topLevelNodeResults.hasNext()) {
         String topLevelNodeURI = topLevelNodeResults.nextSolution().getResource("?
             topLevelNode").getURI();
         if (topLevelNodeURI != null) {
            add(OptimizedEXPReasonerConfiguration.
                W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS,
               OptimizedEXPReasonerConfiguration.
                   W3C_OWL_REASONER_AND_OR_GRAPH_NODE_HAS_CONTENT_NS,
               topLevelNodeURI);
            add(OptimizedEXPReasonerConfiguration.
                W3C_OWL_REASONER_REASONING_UNION_OF_ALL_CONCEPTS_NS,
               OptimizedEXPReasonerConfiguration.
                   W3C_OWL_REASONER_AND_OR_GRAPH_NODE_HAS_MEMBER_NS,
               topLevelNodeURI);
            add(OptimizedEXPReasonerConfiguration.
                W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS,
               OptimizedEXPReasonerConfiguration.
                   W3C_OWL_REASONER_AND_OR_GRAPH_NODE_HAS_CHILD_NS,
               topLevelNodeURI);
            add(OptimizedEXPReasonerConfiguration.
                W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS,
               OptimizedEXPReasonerConfiguration.
                   W3C_OWL_REASONER_AND_OR_GRAPH_NODE_HAS_SUCCESSOR_NS,
               topLevelNodeURI);
            add(topLevelNodeURI,
               OptimizedEXPReasonerConfiguration.
                   W3C_OWL_REASONER_AND_OR_GRAPH_NODE_HAS_PREDECESSOR_NS,
               OptimizedEXPReasonerConfiguration.
                   W3C_OWL_REASONER_AND_OR_GRAPH_ROOT_NODE_NS);

            add(topLevelNodeURI,
               OptimizedEXPReasonerConfiguration.
                   W3C_OWL_REASONER_NODE_STATUS_NO_SLASH_NS,
               OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_NODE_STATUS_UNEXP_NS)
                   ;

            add(topLevelNodeURI,
               OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS,
               OptimizedEXPReasonerConfiguration.
                   W3C_OWL_REASONER_AND_OR_GRAPH_CONCEPT_NODE_NS);

            add(topLevelNodeURI,
               OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS,
               OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS);
```

```
            // for each of the top level nodes, search and link the top level node's
                child and
            // set the content type of the top level node with the content type of the
                child
            Query linkSHIUnionedTopLevelConceptsQuery = getQueryProxy().
                linkSHIUnionedTopLevelConcepts(topLevelNodeURI);
            QueryExecution queryExeclinkSHIUnionedTopLevelConcepts =
                QueryExecutionFactory.create(linkSHIUnionedTopLevelConceptsQuery,
                getTripleStoreRDBModel());
            ResultSet childResults = queryExeclinkSHIUnionedTopLevelConcepts.execSelect
                ();
            while (childResults.hasNext()) {
               QuerySolution childSolution = childResults.nextSolution();
               String childURI = childSolution.getResource("?topLevelNodeChild").getURI
                   ();
               String childContentType = childSolution.getResource("?
                   topLevelNodeChildContentType").getURI();
               if (childURI != null && childContentType != null) {
                  // link the current top level node to the child and set the content
                      type
                  add(topLevelNodeURI,
                     OptimizedEXPReasonerConfiguration.
                         W3C_OWL_REASONER_AND_OR_GRAPH_NODE_HAS_CHILD_NS,
                     childURI);

                  add(topLevelNodeURI,
                     OptimizedEXPReasonerConfiguration.
                         W3C_OWL_REASONER_NODE_CONTENT_NO_SLASH_TYPE_NS,
                     childContentType);

                  add(topLevelNodeURI,
                     OptimizedEXPReasonerConfiguration.
                         W3C_OWL_REASONER_AND_OR_GRAPH_NODE_HAS_CONTENT_NS,
                     childURI);

               }
            }
            queryExeclinkSHIUnionedTopLevelConcepts.close();
         }
      }
      queryExecSelectForBuildUnionOfTopLevelConcepts.close();
   }

   ...

}
```

The main loop of the $\mathcal{SHI}$ decision algorithm is implemented within the body of the *invokeGoreNguyenSatifiabilityAlgorithm* method. It starts with finding an and-or graph node that is not expanded yet. The condition for the main loop to continue is that the *root node* of the and-or graph is neither **sat** or *unsat* as described in the previous description of the tableau algorithm. The method *verifyNodesSatOrUnsat* checks this aspect. The other important premise for continue processing is either the entire tableau is not *closed* or all the nodes have not yet been expanded. The *getUnexpandedConceptNode* method queries the triple store for finding such unexpanded nodes.

The $\mathcal{SHI}$ tableau rules, which are implemented as SPARQL queries and hidden away from the discussion of the current class, are applied exactly in the order as given in the algorithm description. If certain rule applies to the current and-or graph node regarding to the node content, the and-or node will be assigned a certain node type and the status of the node will be set accordingly. Referring to the discussion of the algorithm in section 9.2, step g of the algorithm is implemented in the for-loop in line 120 of the listing, after the application of the tableau rules. It is a step of finding a *proxy* to current content of the node, creating a new node if

such proxy does not yet exist, otherwise an edge is added for the node to the proxy. This is an implementation of a global caching [**?**] [5] technique so that no redundant node is created for the content. From line 125 to 135, the steps h, i, j of the algorithm are implemented intuitively. The corresponding propagation of node status to predecessor is shown in listing example 23.

Listing 23: Implementation of *propagate* method

```
/**
 *
 */
package de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db;


/**
 * @author patrick_un
 *
 */
public class ReasonerTripleStorePersistentModelImpl implements
    ReasonerTripleStorePersistentModel {

   ...

   @Override
   public void propagateStatus(String currentNode) {
      getLog().info("propagate current node [" + currentNode + "] status upwards");
      Stack<String> parentStack = new Stack<String>();
      parentStack.push(currentNode);
      while (!parentStack.isEmpty()) {
         Query selectPredecessorSPARQLQuery = getQueryProxy().propagateStatus(
             parentStack.pop());
         QueryExecution queryExecSelectPredecessors = QueryExecutionFactory.create(
             selectPredecessorSPARQLQuery, getTripleStoreRDBModel());
         ResultSet predecessorResults = queryExecSelectPredecessors.execSelect();
         while (predecessorResults.hasNext()) {
            String parentNodeURI = predecessorResults.nextSolution().getResource("?
                parentNode").getURI();
            if (parentNodeURI != null) {
               getLog().info("found parent node [" + parentNodeURI + "]");
               if ((getNodeType(parentNodeURI).equals(OptimizedEXPReasonerConfiguration
                   .W3C_OWL_REASONER_GRAPH_OR_TYPE_NODE_NS) &&
                   testOneOfSuccessors(parentNodeURI, TRIPLE_STORE_NODE_STATUS.SAT))
                   ||
                  (getNodeType(parentNodeURI).equals(OptimizedEXPReasonerConfiguration.
                      W3C_OWL_REASONER_GRAPH_AND_TYPE_NODE_NS) &&
                   testAllOfSuccessors(parentNodeURI, TRIPLE_STORE_NODE_STATUS.SAT)))
               {

                  setNodeStatus(parentNodeURI, TRIPLE_STORE_NODE_STATUS.SAT);
                  parentStack.push(parentNodeURI);

               } else if ((getNodeType(parentNodeURI).equals(
                   OptimizedEXPReasonerConfiguration.
                   W3C_OWL_REASONER_GRAPH_OR_TYPE_NODE_NS) &&
                         testAllOfSuccessors(parentNodeURI, TRIPLE_STORE_NODE_STATUS.
                             UNSAT))
                         ||
                        (getNodeType(parentNodeURI).equals(
                            OptimizedEXPReasonerConfiguration.
                            W3C_OWL_REASONER_GRAPH_AND_TYPE_NODE_NS) &&
                        testOneOfSuccessors(parentNodeURI, TRIPLE_STORE_NODE_STATUS.
                            UNSAT)))
               {

                  setNodeStatus(parentNodeURI, TRIPLE_STORE_NODE_STATUS.UNSAT);
                  parentStack.push(parentNodeURI);

               }
            }
         }
```

```
            queryExecSelectPredecessors.close();
        }
    }

    ...

}
```

Here the *propagate* method uses an optimized technique that propagates both satisfiability and unsatisfiability upwards to the predecessor directly because once the status is set, it remains unchanged; therefore the status of the predecessor can be altered right away. It uses an optimized technique to traverse the nodes of the constructed and-or graph upwards using the *reasoner:hasPredecessor* predicate as described in previous listing example 16 of the stored RDF triples that model an and-or graph node. The propagation of status is not confined to direct predecessor only but also further upwards where appropriate.

When the main loop ends in step 3 of the described tableau algorithm, it states that if the *root node* has not yet been assigned a **sat** or *unsat* status, all the constructed and-or graph nodes must be checked and if the a certain node is not assigned an *unsat* status, it can be assigned a *sat* status. One way to optimized this looping step, instead of iterating through all constructed nodes, is to utilize the effect of the optimized implementation of the *propagate* method: since the status is not only propagated to the direct predecessor but the entire graph hierarchy upwards, the *root node* could sooner or later become a target for propagation in the execution of the main loop part. This means that step 3 can be written in a simplified way, instead of loop through all the and-or graph nodes $v \in V$, it just test that the *root node* is neither assigned the status **sat** nor *unsat*[12], and that it is **expanded**. One can be sure that because of the effect of the *propagate* method, the *root node* can be safely assigned a **sat** status. Therefore exempting the algorithm from looping through all the nodes once again for the purpose of setting status.

The series of method calls of the $\mathcal{SHI}$ tableau rules is delegated to the corresponding method implementations in the *ReasonerTripleStorePersistentModelImpl* class. For brevity reason, the details are not described here, one can consult the source code instead. In a nutshell an implementation of a method for a certain tableau rule usually proceeds in steps as a complex SPARQL query can be best handled by break it down to several steps as shown in listing example 24. The example shows the implementing the bottom rule ($\bot$) for the $\mathcal{ALC}$ tableau algorithm using this divide and conquer principle.

Listing 24: Implementation of the $\mathcal{ALC}$ bottom rule in the *ReasonerTripleStorePersistentModelImpl* class

```
/**
 *
 */
package de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db;


/**
 * @author patrick_un
 *
 */
public class ReasonerTripleStorePersistentModelImpl implements
    ReasonerTripleStorePersistentModel {

   ...

   @Override
   public boolean bottomRuleALC(String currentNode) {
```

[12] so far it is not assigned status in the main loop yet, otherwise it will bear a status of either **sat** or *unsat* already

```
getLog().info("applying bottom rule on node [" + currentNode + "]");
boolean result = false;
ArrayList<String> relatedConceptTermsList = null;

Query bottomRuleALCSelectProxyNodeToAnonymousConceptReferenceNodeQuery =
   getQueryProxy().bottomRuleALCSelectProxyNodeToAnonymousConceptReferenceNode(
       currentNode);
QueryExecution queryExecSelectProxyNode =
   QueryExecutionFactory.create(
       bottomRuleALCSelectProxyNodeToAnonymousConceptReferenceNodeQuery,
       getTripleStoreRDBModel());
ResultSet proxyResults = queryExecSelectProxyNode.execSelect();

while (proxyResults.hasNext()) {

   // process each found proxy node which is associated with a bridge (anonymous
       node) that concatenate all the concept terms
   // it is important to navigate to the bridge node via the found proxy node
   QuerySolution proxyBridgeQuerySolution = proxyResults.nextSolution();
   String proxyNodeURI = proxyBridgeQuerySolution.getResource("?
       anonymousProxyNode").getURI();
   String bridgeNodeURI = proxyBridgeQuerySolution.getResource("?
       buttomReferenceBridgeNode").getURI();

   if (proxyNodeURI != null && bridgeNodeURI != null) {

      // find a list of defined concept terms by navigating from the found bridge
          node
      // in fact we have to consider only concept terms that are tied together by
          AND
      // i.e. in owl term -> sub-elements of an owl:intersectionOf element in
      //the owl document
   Query bottomRuleALCSelectRelatedConceptsThroughReferenceBridgeNodeQuery =
   getQueryProxy().
       bottomRuleALCSelectRelatedConceptsThroughReferenceBridgeNodeQueryTemplate
       (bridgeNodeURI);
      QueryExecution queryExecSelectRelatedConceptsViaBridge =
         QueryExecutionFactory.create(
             bottomRuleALCSelectRelatedConceptsThroughReferenceBridgeNodeQuery,
             getTripleStoreRDBModel());
      ResultSet referencedRelatedConceptsResult =
          queryExecSelectRelatedConceptsViaBridge.execSelect();

      if (referencedRelatedConceptsResult.hasNext()) {
         relatedConceptTermsList = new ArrayList<String>();
         while (referencedRelatedConceptsResult.hasNext()) {
            // store the set of found related concept terms in an arraylist for
                further evaluation
            QuerySolution relatedConceptTermsSolution =
                referencedRelatedConceptsResult.nextSolution();
            String conceptTermURI = relatedConceptTermsSolution.getResource("?
                relevantDefinedAndConcepts").getURI();
            if (conceptTermURI != null) {
               relatedConceptTermsList.add(conceptTermURI);
            }
         }
      }
      queryExecSelectRelatedConceptsViaBridge.close();

      if (relatedConceptTermsList != null && (!relatedConceptTermsList.isEmpty())
          ) {
         // Evaluation of the found concept terms by building cartesian product
             of the list elements
         // copy the sublist (which does not contain the current observing
             concept term itself to
         // a new list, and then compare each one of the list element in the new
             list to the current
         // concept term itself for finding out clashes!
         for (String fixedCurrentCtermURI : relatedConceptTermsList) {
```

```
                    getLog().info("currently checking [" + fixedCurrentCtermURI + "] with
                         the other defined concepts for a clash");
                    ArrayList<String> subListToCheck = new ArrayList<String>();
                    // copy all concept terms except self to the tmp list
                    for (String elementURI : relatedConceptTermsList) {
70                     if (!elementURI.equals(fixedCurrentCtermURI)) {
                          subListToCheck.add(elementURI);
                       }
                    }
                    // for each of the elements in the subListToCheck, the fixed current
                         concept term is check
75                  // again the element concept term for clash
                    for (String ctermToURICheckForClash : subListToCheck) {
                       Query bottomRuleALCEvaluateConceptsQuery =
                          getQueryProxy().bottomRuleALCEvaluateConcepts(
                              fixedCurrentCtermURI, ctermToURICheckForClash);
                       QueryExecution queryExecAskEvaluateConcepts =
80                        QueryExecutionFactory.create(bottomRuleALCEvaluateConceptsQuery
                              , getTripleStoreRDBModel());
                       // if there is a match in the SPARQL ASK query, a clash is found!
                       boolean askResult = queryExecAskEvaluateConcepts.execAsk();
                       if (askResult) {
                          // there is a clash !
85                        getLog().info("the current concept term [" +
                              fixedCurrentCtermURI +
                                "] clashes with the concept term [" +
                                    ctermToURICheckForClash + "] from an intersection of
                                     the set of the concepts");
                          result = askResult;
                          queryExecAskEvaluateConcepts.close();
                          return result;
90                     }
                       queryExecAskEvaluateConcepts.close();
                    }
                 }
              }
95         }
      }

      getLog().info("botton rule ALC result: " + result + " >>> there is no clash among
           the concept terms");
      queryExecSelectProxyNode.close();
100   return result;
   }

   ...

105 }
```

The point is to find some indirection nodes (here called proxies) in the code excerpt that are navigable to the not contents; then perform check on possible clashes on the mapped RDF triples metadata. In the code excerpt shown, lines 60 to 95 indicate a possible way to check for clashes in the RDF metadata. The corresponding SPARQL query will be illustrated in the last section 12.4 of this part of the thesis.

12.4. Mapping Tableau Rules to SPARQL Queries

The *ReasonerTripleStoreModelSPARQLQueryConstants* class serves to bring together all the relevant SPARQL queries that are relevant to the implementation of the reasoner. The queries are written as string with many of them interspersed with placeholders throughout the SPARQL queries. This is an approach adopted so that the placeholder can be substituted with necessary parameters at run time. This kind of *template* substitution can be convenient at times when the program logic that implements the different tableau rules needs to use many different queries

during execution. Instead of writing down all possible RDF graph patterns, it is better to substitute at run time the appropriate parameter values, thus creating different instances of SPARQL *Query* to select matches. Example listing 25 shows this approach in action.

Listing 25: SPARQL query as *template* string to evaluate two atomic $\mathcal{ALC}$ concepts for clash

```
/**
 *
 */
package de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db;

/**
 * @author patrick_un
 *
 */
public final class ReasonerTripleStoreModelSPARQLQueryConstants {

   ...

   ////////////
   // bottom
   ////////////
   public static String bottomRuleALCEvaluateConceptsQueryTemplate =
      "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
          + "> " +
      "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
          + "> " +
      "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
          W3C_OWL_REASONER_BASE_NS + "> " +
      "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
          W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
      "ASK { " +
      "  {" +
      "  <@CONCEPT_TERM_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
          W3C_OWL_CLASS_NS + ">. " +
      "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:contentType <" +
          OptimizedEXPReasonerConfiguration.
          W3C_OWL_REASONER_NODE_CONTENT_TYPE_COMPLEMENTOF_INVERSUS_CLASS_NS + ">. " +
      "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:isComplementClassOf <
          @CONCEPT_TERM_TWO@>. " +
      "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:hasComplementClass <
          @CONCEPT_TERM_ONE@> " +
      "  } " +
      "UNION " +
      "  {" +
      "  <@CONCEPT_TERM_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
          W3C_OWL_CLASS_NS + ">. " +
      "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:contentType <" +
          OptimizedEXPReasonerConfiguration.
          W3C_OWL_REASONER_NODE_CONTENT_TYPE_NOT_TYPE_NS + ">. " +
      "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:isComplementClassOf <
          @CONCEPT_TERM_TWO@>. " +
      "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:hasComplementClass <
          @CONCEPT_TERM_ONE@> " +
      "  } " +
      "UNION " +
      "  {" +
      "  <@CONCEPT_TERM_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
          W3C_OWL_CLASS_NS + ">. " +
      "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:contentType <" +
          OptimizedEXPReasonerConfiguration.
          W3C_OWL_REASONER_NODE_CONTENT_TYPE_COMPLEMENTOF_INVERSUS_CLASS_NS + ">. " +
      "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:isComplementClassOf <
          @CONCEPT_TERM_ONE@>. " +
      "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:hasComplementClass <
          @CONCEPT_TERM_TWO@> " +
      "  } " +
      "UNION " +
```

```
        "  {" +
        "  <@CONCEPT_TERM_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_CLASS_NS + ">. " +
        "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_NOT_TYPE_NS + ">. " +
        "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:isComplementClassOf <
            @CONCEPT_TERM_ONE@>. " +
        "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:hasComplementClass <
            @CONCEPT_TERM_TWO@> " +
        "  } " +
        "}";

    ...

}
```

Here the placeholders for subject namespace URIs are delimited by a pair of @ signs. At run time parameters will be substituted in the dedicated places by the corresponding method in the *ReasonerTripleStoreModelSPARQLProxy* class.

With the substitution principle explained, the remaining implementation of the other tableau rules will be comprehensive with regard to this principle. For brevity purpose, this section does not list all the mapped SPARQL queries of the *ReasonerTripleStoreModelSPARQLQueryConstants* class implementing the tableau rules. Consult the appendix for a complete display of this class together with the corresponding the *ReasonerTripleStoreModelSPARQLProxy* class which contains the methods containing the substitution logic for the templates. The method names are intuitively chosen to correspond to the name of the template strings in the *ReasonerTripleStoreModelSPARQLQueryConstants* class.

Part VI.
Summary and Outlook

13. Summary

The reasoner prototype can be run in two modes, for satisfiability checking of $\mathcal{ALC}$ or $\mathcal{SHI}$ concepts w.r.t to the corresponding TBox. The implementation of the mode for the $\mathcal{ALC}$ algorithm requires an $\mathcal{ALC}$ OWL TBox and OWL documents concepts as input; whereas the $\mathcal{SHI}$ mode takes three input parameters: a $\mathcal{SHI}$ OWL TBox, an RBox and an OWL concepts documents as input. The reasoner is provided with a start script (as a batch file). Usage information is displayed if the reasoner is run without parameter arguments.

13.1. Evaluation

The run time performance of both the $\mathcal{ALC}$ and $\mathcal{SHI}$ tableau algorithms has been acceptable if one considers that every application of the tableau rules and updates of node information must be performed on persistent RDF data in a triple store, queried via an API. There are some conclusions to be drawn:

Firstly, using an event based OWL parser significantly reduce the amount of main memory required to parse the OWL input. If one considers the DOM parser API, an application will actually have to wait for the parser to finish parsing before it can proceed to access the in-memory data structure required. For the purpose of the prototype reasoner, this is using another parser would not significantly impede parsing performance provided that the OWL documents do not contain too many complex concept terms. However, it will become prohibitively slow if a practical reasoner should parse and import millions of concept terms using the DOM API. Another convenience of the event-based API is that it gives the possibility of navigating the OWL document (XML structure) on an element-to-element basis, providing flexibility in writing mapping rules that maps certain element to a set of RDF metadata during the parsing process. The metadata is added to the triple store simultaneously. Whilst using an in-memory structure such as a DOM-tree would mean first building up the tree, then navigate through the entire structure again in order to map the OWL constructors to metadata.

Secondly, because of the implementation of optimization techniques mentioned, such as global caching of existing and-or graph nodes and optimized propagation of node status, the run time for instance of the $\mathcal{ALC}$ tableau algorithm is on average in the magnitude of milliseconds. Of course, it has to be taken into account that the proposed algorithms have EXPTIME computational property. Nevertheless the introduction of additional tableau rules for the $\mathcal{SHI}$ algorithm leads to further slowdown of the algorithm. This is comprehensible because there are more tableau rules, i.e. SPARQL queries to execute.

Thirdly, by choosing the Jena framework to implement the reasoner for efficient clash recognition in a triple store, the prototype reasoner is decoupled from a specific type of triple store implementation because there are many different supported relational database backends which can be configured to use as persistent triple store together with Jean. The reasoner is therefore

to a certain degree independent of the platform on which RDF triples are stored and therefore immune to the caveats of a specific triple store implementation. Moreover, using the recommended standard query language SPARQL to query triple store will become the mainstream in the future because ever since the SPARQL query language for RDF has been promoted to an official W3C recommendation recently, there is a observable growing number of semantic web APIs and related software products supporting it.

13.2. Outlook

Although he prototype reasoner has incorporated the two modes to deal with $\mathcal{ALC}$ and $\mathcal{SHI}$ concepts, the run time performance can still be improved, despite the fact that the elapsed time for running the tableau algorithm one more expressive description logics concepts will unavoidably increase with increased complexity and expressiveness of the concepts involved.

There are some ways to further optimize the run time performance of the prototype reasoner:

1. fine tune the current SPARQL queries further,
2. selectively reduce some of the RDF data structures that models OWL constructor semantics in the triple store,
3. implement additional consistency-preserving optimization techniques of satisfiability checking on the and-or graph [13],
4. try out and evaluate the performance results of other triple stores that supports SPARQL.

The first issue to look for performance bottlenecks is to investigate and fine tune the SPARQL queries. Currently, because the reasoner has to include alternatives in the RDF graphs pattern matching, there are some queries that must take care of alternatives in pattern. The conventional solution to this issue is to use the UNION keyword in SPARQL. Since the adopted RDF data model is flexible in terms of how it models and map OWL constructors to the RDF triples to preserve semantics, there are understandably many alternatives to deal with when implementing the tableau rules. Fine tuning the SPARQL queries can be, for instance, to scrutinize some queries with many alternatives further to ascertain whether some alternatives can be removed. In this regard, the second possibility can be evaluated in tandem with the first one: instead of tackling the queries, one can try out some other possible semantics mapping alternatives in order to come to a different set of possibly simpler RDF data structure. Section 9.5.4 on *optimizing satisfiability testing* in chapter 9 of the description logics handbook describes some other optimization techniques such as using heuristic guided search and some recommendations on caching of satisfiability status, etc. It remains to be seen if these techniques can be used beneficially for further optimization. It makes sense to try out some other triple stores with the same reasoner prototype implementation to compare for performance difference. On the other hand, it can be a nice to have feature to implement thorough support for XML schema datatypes in the RDF data (types support in RDF literals). Currently, the reasoner deals with namespace URIs and string literals only for simplicity reason.

[13] see chapter 9 in the description logics handbook [**?**] [3]

References

[1] F. Baader and U. Sattler. Tableau algorithms for description logics. In R. Dyckhoff, editor, *Proceedings of the International Conference on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, volume 1847, pages 1–18, St Andrews, Scotland, UK, 2000. Springer-Verlag.

[2] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.

[3] Peter Patel-Schneider Franz Baarder, Diego Calvanese. *The Description Logics Handbook*. Cambridge University Press, The Edinburgh Building, Cambridge CB2 8RU, UK, 2 edition, August 2007.

[4] Rajeev Goré and Linh Anh Nguyen. Exptime tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In *TABLEAUX*, pages 133–148, 2007.

[5] Rajeev Goré and Linh Anh Nguyen. Optimised exptime tableaux for alc using sound global caching, propagation and cutoffs. Technical report, The Australian National University and NICTA, Institute of Informatics, University of Warsaw, 2007.

[6] Bastian Spanneberg. Prototypische implementierung eines triple-store-basierten alc-reasoners. Bachelorarbeit, University of Technology Hamburg, 2007.

Part VII.
SPARQL Query Interface and Implementation Classes

Listing 26: Class implementing the interface of SPARQL query processing *(abridged)*

```
/**
 *
 */
package de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql;

import java.util.Set;

import org.apache.log4j.Logger;

import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryFactory;

import de.tuhh.sts.dl.exptime.tableaux.reasoner.config.logging.
    OptimizedEXPReasonerLogger;

/**
 * @author patrick_un
 *
 */
public class ReasonerTripleStoreModelSPARQLProxy implements
    ReasonerTripleStoreModelSPARQL {

   private final static Logger log = OptimizedEXPReasonerLogger.getLogger();

   /**
    * constructor
    */
   public ReasonerTripleStoreModelSPARQLProxy() {
      super();
   }

   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
        ReasonerTripleStoreModelSPARQL#buildUnionOfTopLevelConcepts()
    */
   @Override
   public Query buildUnionOfTopLevelConcepts() {
      getLog().debug("generate an instance of SPARQL select query to retrieve all top
          level concept nodes");
      String selectForbuildAndOrGraphRootGraphNodeQueryStr =
          ReasonerTripleStoreModelSPARQLQueryConstants.
          selectForBuildAndOrGraphRootGraphNodeQuery;
      return QueryFactory.create(selectForbuildAndOrGraphRootGraphNodeQueryStr);
   }

   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
        ReasonerTripleStoreModelSPARQL#linkUnionedTopLevelConcepts(java.lang.String)
    */
   @Override
   public Query linkUnionedTopLevelConcepts(String topLevelNode) {
      getLog().debug("generate an instance of SPARQL select query to retrieve the
          children and children's content type of the level concept node [" +
          topLevelNode + "]");
      String selectChildForTopLevelUnionNodesQueryTemplateStr =
          ReasonerTripleStoreModelSPARQLQueryConstants.
          selectChildForTopLevelUnionNodesQueryTemplate;
```

```
      String topLevelNodeURIReplaceStr = new String(topLevelNode);
      String regexReplacementPattern = "@FIRST_LEVEL_GRAPH_NODES@";
      String [] strTokenArray = selectChildForTopLevelUnionNodesQueryTemplateStr.split(
          regexReplacementPattern);
      StringBuffer selectChildForTopLevelUnionNodesQueryBuffer = new StringBuffer(
          strTokenArray[0]);
      // all together 18 placeholders to substitute
      for ( int i = 1; i < 19; i++) {
         selectChildForTopLevelUnionNodesQueryBuffer.append(topLevelNodeURIReplaceStr);
         selectChildForTopLevelUnionNodesQueryBuffer.append(strTokenArray[i]);
      }
      return QueryFactory.create(selectChildForTopLevelUnionNodesQueryBuffer.toString()
          );
   }

   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
       ReasonerTripleStoreModelSPARQL#linkUnionedTopLevelConcepts(java.lang.String)
    */
   @Override
   public Query linkSHIUnionedTopLevelConcepts(String topLevelNode) {
      getLog().debug("generate an instance of SPARQL select query to retrieve the
          children and children's content type of the level concept node [" +
          topLevelNode + "]");
      // there are 54 placeholders to be substituted, better to use a loop stop
          condition is 55
      String selectSHIChildForTopLevelUnionNodesQueryTemplateStr =
          ReasonerTripleStoreModelSPARQLQueryConstants.
          selectSHIChildForTopLevelUnionNodesQueryTemplate;
      String topLevelNodeURIReplaceStr = new String(topLevelNode);
      String regexReplacementPattern = "@FIRST_LEVEL_GRAPH_NODES@";
      String [] strTokenArray = selectSHIChildForTopLevelUnionNodesQueryTemplateStr.
          split(regexReplacementPattern);
      StringBuffer selectSHIChildForTopLevelUnionNodesQueryBuffer = new StringBuffer(
          strTokenArray[0]);
      // all together 54 placeholders to substitute
      for (int i = 1; i < 55 ; i++) {
         selectSHIChildForTopLevelUnionNodesQueryBuffer.append(
             topLevelNodeURIReplaceStr);
         selectSHIChildForTopLevelUnionNodesQueryBuffer.append(strTokenArray[i]);
      }
      return QueryFactory.create(selectSHIChildForTopLevelUnionNodesQueryBuffer.
          toString());
   }


   ...

   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
       ReasonerTripleStoreModelSPARQL#rootGraphNodeSatOrUnsat()
    */
   @Override
   public Query rootGraphNodeSatOrUnsat() {
      getLog().debug("generate an instance of SPARQL ASK query for root node");
      return QueryFactory.create(ReasonerTripleStoreModelSPARQLQueryConstants.
          rootGraphNodeSatOrUnsatQuery);
   }

   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
       ReasonerTripleStoreModelSPARQL#getUnexpandedConceptNode()
    */
   @Override
   public Query getUnexpandedConceptNode() {
      getLog().debug("generate an instance of SPARQL SELECT query to retrieve
          unexpanded concept nodes");
      return QueryFactory.create(ReasonerTripleStoreModelSPARQLQueryConstants.
          selectUnexpandedConceptNodeQuery);
```

```
    }

    @Override
    public Query getAndOrGraphNodeStatus(String node) {
       getLog().debug("generate an instance of SPARQL SELECT query to get node status
           for [node: " + node + "]");
       String selectForGetNodeStatusQueryTemplateStr =
           ReasonerTripleStoreModelSPARQLQueryConstants.
           selectForGetNodeStatusQueryTemplate;
       String getNodeStatusQueryReplaceStr = new String(node);
       String regexReplacementPattern = "@NODE_TO_GET_STATUS@";
       String [] strTokenArray = selectForGetNodeStatusQueryTemplateStr.split(
           regexReplacementPattern);
       StringBuffer selectForGetNodeStatusQueryBuffer = new StringBuffer(strTokenArray
           [0]);
       selectForGetNodeStatusQueryBuffer.append(getNodeStatusQueryReplaceStr);
       selectForGetNodeStatusQueryBuffer.append(strTokenArray[1]);
       return QueryFactory.create(selectForGetNodeStatusQueryBuffer.toString());
    }

    @Override
    public Query setAndOrGraphNodeStatus(String node) {
       getLog().debug("generate an instance of SPARQL SELECT query to set node status
           for [node: " + node + "]");
       String selectForSetNodeStatusQueryTemplateStr =
           ReasonerTripleStoreModelSPARQLQueryConstants.
           selectForSetNodeStatusQueryTemplate;
       String setNodeStatusQueryReplaceStr = new String(node);
       String regexReplacementPattern = "@NODE_TO_SET_STATUS@";
       String [] strTokenArray = selectForSetNodeStatusQueryTemplateStr.split(
           regexReplacementPattern);
       StringBuffer selectForSetNodeStatusQueryBuffer = new StringBuffer(strTokenArray
           [0]);
       selectForSetNodeStatusQueryBuffer.append(setNodeStatusQueryReplaceStr);
       selectForSetNodeStatusQueryBuffer.append(strTokenArray[1]);
       return QueryFactory.create(selectForSetNodeStatusQueryBuffer.toString());
    }

    @Override
    public Query getAndOrGraphNodeType(String node) {
       getLog().debug("generate an instance of SPARQL SELECT query to get node type for
           [node: " + node + "]");
       String selectForGetNodeTypeQueryTemplateStr =
           ReasonerTripleStoreModelSPARQLQueryConstants.
           selectForGetNodeTypeQueryTemplate;
       String getNodeTypeQueryReplaceStr = new String(node);
       String regexReplacementPattern = "@NODE_TO_GET_NODE_TYPE@";
       String [] strTokenArray = selectForGetNodeTypeQueryTemplateStr.split(
           regexReplacementPattern);
       StringBuffer selectForGetNodeTypeQueryBuffer = new StringBuffer(strTokenArray[0])
           ;
       selectForGetNodeTypeQueryBuffer.append(getNodeTypeQueryReplaceStr);
       selectForGetNodeTypeQueryBuffer.append(strTokenArray[1]);
       selectForGetNodeTypeQueryBuffer.append(getNodeTypeQueryReplaceStr);
       selectForGetNodeTypeQueryBuffer.append(strTokenArray[2]);
       return QueryFactory.create(selectForGetNodeTypeQueryBuffer.toString());
    }

    @Override
    public Query setAndOrGraphNodeType(String node) {
       getLog().debug("generate an instance of SPARQL SELECT query to set node type for
           [node: " + node + "]");
       String selectForSetNodeTypeQueryTemplateStr =
           ReasonerTripleStoreModelSPARQLQueryConstants.
           selectForSetNodeTypeQueryTemplate;
       String setNodeTypeQueryReplaceStr = new String(node);
       String regexReplacementPattern = "@NODE_TO_SET_NODE_TYPE@";
       String [] strTokenArray = selectForSetNodeTypeQueryTemplateStr.split(
           regexReplacementPattern);
```

```
        StringBuffer selectForSetNodeTypeQueryBuffer = new StringBuffer(strTokenArray[0])
            ;
        selectForSetNodeTypeQueryBuffer.append(setNodeTypeQueryReplaceStr);
        selectForSetNodeTypeQueryBuffer.append(strTokenArray[1]);
        selectForSetNodeTypeQueryBuffer.append(setNodeTypeQueryReplaceStr);
        selectForSetNodeTypeQueryBuffer.append(strTokenArray[2]);
        return QueryFactory.create(selectForSetNodeTypeQueryBuffer.toString());
    }

    ////////////////////////////////////////////////////////
    //
    // IMPLEMENTATION OF THE TABLEAUX RULES FOR ALC AND SHI
    //
    ////////////////////////////////////////////////////////
    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
         ReasonerTripleStoreModelSPARQL#
         bottomRuleALCSelectProxyNodeToAnonymousConceptReferenceNode(java.lang.String)
     */
    @Override
    public Query bottomRuleALCSelectProxyNodeToAnonymousConceptReferenceNode(String
        currentNode) {
        getLog().debug("generate an instance of SPARQL query for ALC bottom-rule (
            bottomRuleALCSelectProxyNodeToAnonymousConceptReferenceNode)");
        String bottomRuleALCSelectProxyNodeQueryTemplateStr =
           ReasonerTripleStoreModelSPARQLQueryConstants.
               bottomRuleALCSelectProxyNodeToAnonymousButtomReferenceNodeQueryTemplate;
        String currentNodeURIReplaceStr = new String(currentNode);
        String regexReplacementPattern = "@CURRENT_NODE@";
        String [] strTokenArray = bottomRuleALCSelectProxyNodeQueryTemplateStr.split(
            regexReplacementPattern);
        StringBuffer bottomRuleALCSelectProxyNodeQueryBuffer = new StringBuffer(
            strTokenArray[0]);
        bottomRuleALCSelectProxyNodeQueryBuffer.append(currentNodeURIReplaceStr);
        bottomRuleALCSelectProxyNodeQueryBuffer.append(strTokenArray[1]);
        bottomRuleALCSelectProxyNodeQueryBuffer.append(currentNodeURIReplaceStr);
        bottomRuleALCSelectProxyNodeQueryBuffer.append(strTokenArray[2]);
        return QueryFactory.create(bottomRuleALCSelectProxyNodeQueryBuffer.toString());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
         ReasonerTripleStoreModelSPARQL#
         bottomRuleALCSelectRelatedConceptsThroughReferenceBridgeNodeQueryTemplate(java
         .lang.String)
     */
    @Override
    public Query
        bottomRuleALCSelectRelatedConceptsThroughReferenceBridgeNodeQueryTemplate(
        String bridgeNode) {
        getLog().debug("generate an instance of SPARQL query for ALC bottom-rule (
            bottomRuleALCSelectRelatedConceptsThroughReferenceBridgeNodeQueryTemplate)")
            ;
        String bottomRuleALCSelectRelatedConceptsQueryTemplateStr =
           ReasonerTripleStoreModelSPARQLQueryConstants.
               bottomRuleALCSelectRelatedConceptsThroughReferenceBridgeNodeQueryTemplate
               ;
        String bridgeNodeURIReplaceStr = new String(bridgeNode);
        String regexReplacementPattern = "@BRIDGE_NODE@";
        String [] strTokenArray = bottomRuleALCSelectRelatedConceptsQueryTemplateStr.
            split(regexReplacementPattern);
        StringBuffer bottomRuleALCSelectRelatedConceptsQueryBuffer = new StringBuffer(
            strTokenArray[0]);
        bottomRuleALCSelectRelatedConceptsQueryBuffer.append(bridgeNodeURIReplaceStr);
        bottomRuleALCSelectRelatedConceptsQueryBuffer.append(strTokenArray[1]);
        bottomRuleALCSelectRelatedConceptsQueryBuffer.append(bridgeNodeURIReplaceStr);
        bottomRuleALCSelectRelatedConceptsQueryBuffer.append(strTokenArray[2]);
        bottomRuleALCSelectRelatedConceptsQueryBuffer.append(bridgeNodeURIReplaceStr);
        bottomRuleALCSelectRelatedConceptsQueryBuffer.append(strTokenArray[3]);
```

```
        bottomRuleALCSelectRelatedConceptsQueryBuffer.append(bridgeNodeURIReplaceStr);
        bottomRuleALCSelectRelatedConceptsQueryBuffer.append(strTokenArray[4]);
        return QueryFactory.create(bottomRuleALCSelectRelatedConceptsQueryBuffer.toString
            ());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
        ReasonerTripleStoreModelSPARQL#bottomRuleALCEvaluateConcepts(java.lang.String,
         java.lang.String)
     */
    @Override
    public Query bottomRuleALCEvaluateConcepts(String firstConceptToCheck, String
        anotherConceptToCheck) {
        getLog().debug("generate an instance of SPARQL query for ALC bottom-rule (
            bottomRuleALCEvaluateConcepts)");
        String bottomRuleALCEvaluateConceptsQueryTemplateStr =
            ReasonerTripleStoreModelSPARQLQueryConstants.
                bottomRuleALCEvaluateConceptsQueryTemplate;
        String firstConceptToCheckReplaceStr = new String(firstConceptToCheck);
        String anotherConceptToCheckReplaceStr = new String(anotherConceptToCheck);
        String firstConceptRegexReplacementPattern = "@CONCEPT_TERM_ONE@";
        String anotherConceptRegexReplacementPattern = "@CONCEPT_TERM_TWO@";
        // first split the with first concept term as regex pattern (13 fragments
            splitted)
        String [] strTokenArrayA = bottomRuleALCEvaluateConceptsQueryTemplateStr.split(
            firstConceptRegexReplacementPattern);
        StringBuffer tmpBufferA = new StringBuffer(strTokenArrayA[0]);
        for (int i = 1; i < 13 ;i++) {
            tmpBufferA.append(firstConceptToCheckReplaceStr);
            tmpBufferA.append(strTokenArrayA[i]);
        }

        // then split with the other concept term as regex pattern (also 13 fragments
            splitted)
        String [] strTokenArrayB = tmpBufferA.toString().split(
            anotherConceptRegexReplacementPattern);
        StringBuffer tmpBufferB = new StringBuffer(strTokenArrayB[0]);
        for (int j = 1; j < 13 ;j++) {
            tmpBufferB.append(anotherConceptToCheckReplaceStr);
            tmpBufferB.append(strTokenArrayB[j]);
        }

        // the query string is correctly substituted, copy to new buffer
        StringBuffer bottomRuleALCEvaluateConceptsQueryBuffer = new StringBuffer(
            tmpBufferB.toString());
        return QueryFactory.create(bottomRuleALCEvaluateConceptsQueryBuffer.toString());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
        ReasonerTripleStoreModelSPARQL#
        bottomRuleSHISelectProxyNodeToAnonymousConceptReferenceNode(java.lang.String)
     */
    @Override
    public Query bottomRuleSHISelectProxyNodeToAnonymousConceptReferenceNode(String
        currentNode) {
        getLog().debug("generate an instance of SPARQL query for SHI bottom-rule (
            bottomRuleALCSelectProxyNodeToAnonymousConceptReferenceNode)");
        String bottomRuleSHIlectProxyNodeQueryTemplateStr =
            ReasonerTripleStoreModelSPARQLQueryConstants.
                bottomRuleSHISelectProxyNodeToAnonymousButtomReferenceNodeQueryTemplate;
        String currentNodeURIReplaceStr = new String(currentNode);
        String regexReplacementPattern = "@CURRENT_NODE@";
        String [] strTokenArray = bottomRuleSHIlectProxyNodeQueryTemplateStr.split(
            regexReplacementPattern);
        StringBuffer bottomRuleSHISelectProxyNodeQueryBuffer = new StringBuffer(
            strTokenArray[0]);
        bottomRuleSHISelectProxyNodeQueryBuffer.append(currentNodeURIReplaceStr);
```

```
      bottomRuleSHISelectProxyNodeQueryBuffer.append(strTokenArray[1]);
      bottomRuleSHISelectProxyNodeQueryBuffer.append(currentNodeURIReplaceStr);
      bottomRuleSHISelectProxyNodeQueryBuffer.append(strTokenArray[2]);
      return QueryFactory.create(bottomRuleSHISelectProxyNodeQueryBuffer.toString());
   }

   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
        ReasonerTripleStoreModelSPARQL#
        bottomRuleSHISelectRelatedConceptsThroughReferenceBridgeNodeQueryTemplate(java
        .lang.String)
    */
   @Override
   public Query
       bottomRuleSHISelectRelatedConceptsThroughReferenceBridgeNodeQueryTemplate(
       String bridgeNode) {
      getLog().debug("generate an instance of SPARQL query for SHI bottom-rule (
          bottomRuleALCSelectRelatedConceptsThroughReferenceBridgeNodeQueryTemplate)")
          ;
      String bottomRuleSHISelectRelatedConceptsQueryTemplateStr =
         ReasonerTripleStoreModelSPARQLQueryConstants.
             bottomRuleSHISelectRelatedClassConceptsThroughReferenceBridgeNodeQueryTemplate
             ;
      String bridgeNodeURIReplaceStr = new String(bridgeNode);
      String regexReplacementPattern = "@BRIDGE_NODE@";
      String [] strTokenArray = bottomRuleSHISelectRelatedConceptsQueryTemplateStr.
          split(regexReplacementPattern);
      StringBuffer bottomRuleSHISelectRelatedConceptsQueryBuffer = new StringBuffer(
          strTokenArray[0]);
      bottomRuleSHISelectRelatedConceptsQueryBuffer.append(bridgeNodeURIReplaceStr);
      bottomRuleSHISelectRelatedConceptsQueryBuffer.append(strTokenArray[1]);
      bottomRuleSHISelectRelatedConceptsQueryBuffer.append(bridgeNodeURIReplaceStr);
      bottomRuleSHISelectRelatedConceptsQueryBuffer.append(strTokenArray[2]);
      bottomRuleSHISelectRelatedConceptsQueryBuffer.append(bridgeNodeURIReplaceStr);
      bottomRuleSHISelectRelatedConceptsQueryBuffer.append(strTokenArray[3]);
      bottomRuleSHISelectRelatedConceptsQueryBuffer.append(bridgeNodeURIReplaceStr);
      bottomRuleSHISelectRelatedConceptsQueryBuffer.append(strTokenArray[4]);
      return QueryFactory.create(bottomRuleSHISelectRelatedConceptsQueryBuffer.toString
          ());
   }

   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
        ReasonerTripleStoreModelSPARQL#bottomRuleSHIEvaluateConcepts(java.lang.String,
         java.lang.String)
    */
   @Override
   public Query bottomRuleSHIEvaluateConcepts(String firstClassConceptToCheck, String
       anotherClassConceptToCheck) {
//    getLog().debug("generate an instance of SPARQL query for SHI bottom-rule (
    bottomRuleALCEvaluateConcepts)");
      String bottomRuleSHIEvaluateConceptsQueryTemplateStr =
         ReasonerTripleStoreModelSPARQLQueryConstants.
             bottomRuleSHIEvaluateClassConceptsQueryTemplate;
      String firstClassConceptToCheckReplaceStr = new String(firstClassConceptToCheck);
      String anotherClassConceptToCheckReplaceStr = new String(
          anotherClassConceptToCheck);
      String firstClassConceptRegexReplacementPattern = "@CONCEPT_TERM_ONE@";
      String anotherClassConceptRegexReplacementPattern = "@CONCEPT_TERM_TWO@";
      // first split the with first concept term as regex pattern (13 fragments
          splitted)
      String [] strTokenArrayA = bottomRuleSHIEvaluateConceptsQueryTemplateStr.split(
          firstClassConceptRegexReplacementPattern);
      StringBuffer tmpBufferA = new StringBuffer(strTokenArrayA[0]);
      for (int i = 1; i < 13; i++) {
         tmpBufferA.append(firstClassConceptToCheckReplaceStr);
         tmpBufferA.append(strTokenArrayA[i]);
      }
```

```
      // then split with the other concept term as regex pattern (also 13 fragments
          splitted)
      String [] strTokenArrayB = tmpBufferA.toString().split(
          anotherClassConceptRegexReplacementPattern);
      StringBuffer tmpBufferB = new StringBuffer(strTokenArrayB[0]);
      for (int j = 1; j < 13; j++) {
         tmpBufferB.append(anotherClassConceptToCheckReplaceStr);
         tmpBufferB.append(strTokenArrayB[j]);
      }

      // the query string is correctly substituted, copy to new buffer
      StringBuffer bottomRuleSHIEvaluateConceptsQueryBuffer = new StringBuffer(
          tmpBufferB.toString());
      return QueryFactory.create(bottomRuleSHIEvaluateConceptsQueryBuffer.toString());
   }


   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
        ReasonerTripleStoreModelSPARQL#bottomRuleSHISelectReferencedPropertiesConcept(
        java.lang.String)
    */
   @Override
   public Query bottomRuleSHISelectReferencedPropertiesConcept(String currentNode) {
      getLog().debug("generate an instance of SPARQL query for SHI bottom-rule check
          properties (bottomRuleSHISelectReferencedPropertiesConcept)");
      // bottomRuleSHISelectTopReferencedPropertiesQueryTemplate 6 placeholders
      String bottomRuleSHISelectTopReferencedPropertiesQueryTemplateStr =
         ReasonerTripleStoreModelSPARQLQueryConstants.
             bottomRuleSHISelectTopReferencedPropertiesQueryTemplate;
      String currentNodeURIReplaceStr = new String(currentNode);
      String regexReplacementPattern = "@CURRENT_NODE@";
      String [] strTokenArray =
          bottomRuleSHISelectTopReferencedPropertiesQueryTemplateStr.split(
          regexReplacementPattern);
      StringBuffer bottomRuleSHISelectTopReferencedPropertiesQueryBuffer = new
          StringBuffer(strTokenArray[0]);
      for (int i = 1 ; i < 7; i++) {
         bottomRuleSHISelectTopReferencedPropertiesQueryBuffer.append(
             currentNodeURIReplaceStr);
         bottomRuleSHISelectTopReferencedPropertiesQueryBuffer.append(strTokenArray[i])
             ;
      }
      return QueryFactory.create(bottomRuleSHISelectTopReferencedPropertiesQueryBuffer.
          toString());
   }

   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
        ReasonerTripleStoreModelSPARQL#bottomRuleSHIEvaluatePropertyConcepts(java.lang
        .String, java.lang.String)
    */
   @Override
   public Query bottomRuleSHIEvaluatePropertyConcepts(String firstPropertyToCheck,
       String anotherPropertyToCheck) {
//    getLog().debug("generate an instance of SPARQL query for SHI bottom-rule check
    properties (bottomRuleSHIEvaluatePropertyConcepts)");
      String bottomRuleSHIEvaluatePropertyConceptsQueryTemplateStr =
         ReasonerTripleStoreModelSPARQLQueryConstants.
             bottomRuleSHIEvaluatePropertyConceptsQueryTemplate;

      String firstPropertyToCheckReplaceStr = new String(firstPropertyToCheck);
      String anotherPropertyToCheckReplaceStr = new String(anotherPropertyToCheck);
      String firstPropertyConceptRegexReplacementPattern = "@PROP_CONCEPT_TERM_ONE@";
      String anotherPropertyConceptRegexReplacementPattern = "@PROP_CONCEPT_TERM_TWO@";
      // first split the with first concept term as regex pattern (35 placeholders and
          36 fragments splitted)
      String [] strTokenArrayA = bottomRuleSHIEvaluatePropertyConceptsQueryTemplateStr.
          split(firstPropertyConceptRegexReplacementPattern);
```

```
      StringBuffer tmpBufferA = new StringBuffer(strTokenArrayA[0]);
      // 35 placeholders to substitute
      for (int i = 1; i < 36; i++) {
         tmpBufferA.append(firstPropertyToCheckReplaceStr);
         tmpBufferA.append(strTokenArrayA[i]);
      }

      // then split with the other concept term as regex pattern (35 placeholders and
          36 fragments splitted)
      String [] strTokenArrayB = tmpBufferA.toString().split(
          anotherPropertyConceptRegexReplacementPattern);
      StringBuffer tmpBufferB = new StringBuffer(strTokenArrayB[0]);
      for (int j = 1; j < 36; j++) {
         tmpBufferB.append(anotherPropertyToCheckReplaceStr);
         tmpBufferB.append(strTokenArrayB[j]);
      }

      // the query string is correctly substituted, copy to new buffer
      StringBuffer bottomRuleSHIEvaluatePropertyConceptsQueryBuffer = new StringBuffer(
          tmpBufferB.toString());
      return QueryFactory.create(bottomRuleSHIEvaluatePropertyConceptsQueryBuffer.
          toString());
   }

   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
       ReasonerTripleStoreModelSPARQL#andRuleALCSelectProxy(java.lang.String)
    */
   @Override
   public Query andRuleALCSelectProxy(String currentNode) {
      getLog().debug("generate an instance of SPARQL query for ALC and-rule to select
          content proxy linked to concepts");
      String andRuleALCSelectProxyQueryTemplateStr =
          ReasonerTripleStoreModelSPARQLQueryConstants.
          andRuleALCSelectProxyQueryTemplate;
      String currentNodeURIReplaceStr = new String(currentNode);
      String regexReplacementPattern = "@CURRENT_NODE@";
      String [] strTokenArray = andRuleALCSelectProxyQueryTemplateStr.split(
          regexReplacementPattern);
      StringBuffer andRuleALCSelectProxyQueryBuffer = new StringBuffer(strTokenArray
          [0]);
      for (int i = 1; i < 5; i++) {
         andRuleALCSelectProxyQueryBuffer.append(currentNodeURIReplaceStr);
         andRuleALCSelectProxyQueryBuffer.append(strTokenArray[i]);
      }
      return QueryFactory.create(andRuleALCSelectProxyQueryBuffer.toString());
   }

   /* (non-Javadoc)
    * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
       ReasonerTripleStoreModelSPARQL#andRuleALCSelectConcept(java.lang.String)
    */
   @Override
   public Query andRuleALCSelectConcept(String contentProxyNode) {
      getLog().debug("generate an instance of SPARQL query for ALC and-rule to select
          linked concepts");
      String andRuleALCSelectConceptQueryTemplateStr =
          ReasonerTripleStoreModelSPARQLQueryConstants.
          andRuleALCSelectConceptQueryTemplate;
      String contentProxyNodeURIReplaceStr = new String(contentProxyNode);
      String regexReplacementPattern = "@PROXY_CONTENT_NODE@";
      String [] strTokenArray = andRuleALCSelectConceptQueryTemplateStr.split(
          regexReplacementPattern);
      StringBuffer andRuleALCSelectConceptQueryBuffer = new StringBuffer(strTokenArray
          [0]);
      for (int i = 1; i < 13; i++) {
         andRuleALCSelectConceptQueryBuffer.append(contentProxyNodeURIReplaceStr);
         andRuleALCSelectConceptQueryBuffer.append(strTokenArray[i]);
      }
```

```
        return QueryFactory.create(andRuleALCSelectConceptQueryBuffer.toString());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
         ReasonerTripleStoreModelSPARQL#andRuleALCSelectAllNotAndContentProxyNodes(java
         .lang.String)
     */
    @Override
    public Query andRuleALCSelectAllNotAndContentProxyNodes(String currentNode) {
        getLog().debug("generate an instance of SPARQL query for ALC and-rule to select
            content proxy which is not and content typed");
        String andRuleALCSelectAllNotAndConceptProxyNodesQueryTemplateStr =
            ReasonerTripleStoreModelSPARQLQueryConstants.
            andRuleALCSelectAllNotAndConceptProxyNodesQueryTemplate;
        String currentNodeURIReplaceStr = new String(currentNode);
        String regexReplacementPattern = "@CURRENT_NODE@";
        String [] strTokenArray =
            andRuleALCSelectAllNotAndConceptProxyNodesQueryTemplateStr.split(
            regexReplacementPattern);
        StringBuffer andRuleALCSelectAllNotAndConceptProxyNodesQueryBuffer = new
            StringBuffer(strTokenArray[0]);
        andRuleALCSelectAllNotAndConceptProxyNodesQueryBuffer.append(
            currentNodeURIReplaceStr);
        andRuleALCSelectAllNotAndConceptProxyNodesQueryBuffer.append(strTokenArray[1]);
        andRuleALCSelectAllNotAndConceptProxyNodesQueryBuffer.append(
            currentNodeURIReplaceStr);
        andRuleALCSelectAllNotAndConceptProxyNodesQueryBuffer.append(strTokenArray[2]);
        return QueryFactory.create(andRuleALCSelectAllNotAndConceptProxyNodesQueryBuffer.
            toString());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
         ReasonerTripleStoreModelSPARQL#andRuleSHISelectProxy(java.lang.String)
     */
    @Override
    public Query andRuleSHISelectProxy(String currentNode) {
        getLog().debug("generate an instance of SPARQL query for SHI and-rule to select
            content proxy linked to concepts");
        String andRuleSHISelectProxyQueryTemplateStr =
            ReasonerTripleStoreModelSPARQLQueryConstants.
            andRuleSHISelectProxyQueryTemplate;
        String currentNodeURIReplaceStr = new String(currentNode);
        String regexReplacementPattern = "@CURRENT_NODE@";
        String [] strTokenArray = andRuleSHISelectProxyQueryTemplateStr.split(
            regexReplacementPattern);
        StringBuffer andRuleSHISelectProxyQueryBuffer = new StringBuffer(strTokenArray
            [0]);
        for (int i = 1; i < 6; i++) {
           andRuleSHISelectProxyQueryBuffer.append(currentNodeURIReplaceStr);
           andRuleSHISelectProxyQueryBuffer.append(strTokenArray[i]);
        }
        return QueryFactory.create(andRuleSHISelectProxyQueryBuffer.toString());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
         ReasonerTripleStoreModelSPARQL#andRuleSHISelectConcept(java.lang.String)
     */
    @Override
    public Query andRuleSHISelectConcept(String contentProxyNode) {
        getLog().debug("generate an instance of SPARQL query for SHI and-rule to select
            linked concepts");
        String andRuleSHISelectConceptQueryTemplateStr =
            ReasonerTripleStoreModelSPARQLQueryConstants.
            andRuleSHISelectConceptQueryTemplate;
        String contentProxyNodeURIReplaceStr = new String(contentProxyNode);
        String regexReplacementPattern = "@PROXY_CONTENT_NODE@";
```

```
        String [] strTokenArray = andRuleSHISelectConceptQueryTemplateStr.split(
            regexReplacementPattern);
        StringBuffer andRuleSHISelectConceptQueryBuffer = new StringBuffer(strTokenArray
            [0]);
        for (int i = 1; i < 19; i++) {
           andRuleSHISelectConceptQueryBuffer.append(contentProxyNodeURIReplaceStr);
           andRuleSHISelectConceptQueryBuffer.append(strTokenArray[i]);
        }
        return QueryFactory.create(andRuleSHISelectConceptQueryBuffer.toString());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
         ReasonerTripleStoreModelSPARQL#andRuleSHISelectAllNotAndContentProxyNodes(java
         .lang.String)
     */
    @Override
    public Query andRuleSHISelectAllNotAndContentProxyNodes(String currentNode) {
        getLog().debug("generate an instance of SPARQL query for SHI and-rule to select
            content proxy which is not and content typed");
        String andRuleSHISelectAllNotAndConceptProxyNodesQueryTemplateStr =
            ReasonerTripleStoreModelSPARQLQueryConstants.
            andRuleSHISelectAllNotAndConceptProxyNodesQueryTemplate;
        String currentNodeURIReplaceStr = new String(currentNode);
        String regexReplacementPattern = "@CURRENT_NODE@";
        String [] strTokenArray =
            andRuleSHISelectAllNotAndConceptProxyNodesQueryTemplateStr.split(
            regexReplacementPattern);
        StringBuffer andRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer = new
            StringBuffer(strTokenArray[0]);
        andRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer.append(
            currentNodeURIReplaceStr);
        andRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer.append(strTokenArray[1]);
        andRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer.append(
            currentNodeURIReplaceStr);
        andRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer.append(strTokenArray[2]);
        return QueryFactory.create(andRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer.
            toString());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
         ReasonerTripleStoreModelSPARQL#orRuleALCSelectProxy(java.lang.String)
     */
    @Override
    public Query orRuleALCSelectProxy(String currentNode) {
        getLog().debug("generate an instance of SPARQL query for ALC-or-rule on node [" +
             currentNode + "]");
        String orRuleALCSelectProxyQueryTemplateStr =
            ReasonerTripleStoreModelSPARQLQueryConstants.
            orRuleALCSelectProxyQueryTemplate;
        String currentNodeURIReplaceStr = new String(currentNode);
        String regexReplacementPattern = "@CURRENT_NODE@";
        String [] strTokenArray = orRuleALCSelectProxyQueryTemplateStr.split(
            regexReplacementPattern);
        StringBuffer orRuleALCSelectProxyQueryBuffer = new StringBuffer(strTokenArray[0])
            ;
        for (int i = 1; i < 5; i++) {
           orRuleALCSelectProxyQueryBuffer.append(currentNodeURIReplaceStr);
           orRuleALCSelectProxyQueryBuffer.append(strTokenArray[i]);
        }
        return QueryFactory.create(orRuleALCSelectProxyQueryBuffer.toString());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
         ReasonerTripleStoreModelSPARQL#orRuleALCSelectConcept(java.lang.String)
     */
    @Override
```

```
public Query orRuleALCSelectConcept(String contentProxyNode) {
   getLog().debug("generate an instance of SPARQL query for content proxy linked to
       or binding concepts nodes");
   String orRuleALCSelectConceptQueryTemplateStr =
       ReasonerTripleStoreModelSPARQLQueryConstants.
       orRuleALCSelectConceptQueryTemplate;
   String contentProxyURIReplaceStr = new String(contentProxyNode);
   String regexReplacementPattern = "@PROXY_CONTENT_NODE@";
   String [] strTokenArray = orRuleALCSelectConceptQueryTemplateStr.split(
       regexReplacementPattern);
   StringBuffer orRuleALCSelectConceptQueryBuffer = new StringBuffer(strTokenArray
       [0]);
   for (int i = 1; i < 13; i++) {
      orRuleALCSelectConceptQueryBuffer.append(contentProxyURIReplaceStr);
      orRuleALCSelectConceptQueryBuffer.append(strTokenArray[i]);
   }
   return QueryFactory.create(orRuleALCSelectConceptQueryBuffer.toString());
}

/* (non-Javadoc)
 * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
     ReasonerTripleStoreModelSPARQL#orRuleALCSelectAllNotOrContentProxyNodes(java.
     lang.String)
 */
@Override
public Query orRuleALCSelectAllNotOrContentProxyNodes(String currentNode) {
   getLog().debug("generate an instance of SPARQL query for retrieving all not or
       concepts nodes except self");
   String orRuleALCSelectAllNotOrConceptProxyNodesQueryTemplateStr =
       ReasonerTripleStoreModelSPARQLQueryConstants.
       orRuleALCSelectAllNotOrConceptProxyNodesQueryTemplate;
   String currentNodeURIReplaceStr = new String(currentNode);
   String regexReplacementPattern1 = "@CURRENT_NODE@";
   String [] strTokenArray =
       orRuleALCSelectAllNotOrConceptProxyNodesQueryTemplateStr.split(
       regexReplacementPattern1);
   StringBuffer orRuleALCSelectAllNotOrConceptProxyNodesQueryBuffer = new
       StringBuffer(strTokenArray[0]);
   orRuleALCSelectAllNotOrConceptProxyNodesQueryBuffer.append(
       currentNodeURIReplaceStr);
   orRuleALCSelectAllNotOrConceptProxyNodesQueryBuffer.append(strTokenArray[1]);
   orRuleALCSelectAllNotOrConceptProxyNodesQueryBuffer.append(
       currentNodeURIReplaceStr);
   orRuleALCSelectAllNotOrConceptProxyNodesQueryBuffer.append(strTokenArray[2]);
   return QueryFactory.create(orRuleALCSelectAllNotOrConceptProxyNodesQueryBuffer.
       toString());
}

/* (non-Javadoc)
 * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
     ReasonerTripleStoreModelSPARQL#orRuleSHISelectProxy(java.lang.String)
 */
@Override
public Query orRuleSHISelectProxy(String currentNode) {
   getLog().debug("generate an instance of SPARQL query for SHI or-rule to select
       content proxy linked to or concepts");
   String orRuleSHISelectProxyQueryTemplateStr =
       ReasonerTripleStoreModelSPARQLQueryConstants.
       orRuleSHISelectProxyQueryTemplate;
   String currentNodeURIReplaceStr = new String(currentNode);
   String regexReplacementPattern = "@CURRENT_NODE@";
   String [] strTokenArray = orRuleSHISelectProxyQueryTemplateStr.split(
       regexReplacementPattern);
   StringBuffer orRuleSHISelectProxyQueryBuffer = new StringBuffer(strTokenArray[0])
       ;
   for (int i = 1; i < 6; i++) {
      orRuleSHISelectProxyQueryBuffer.append(currentNodeURIReplaceStr);
      orRuleSHISelectProxyQueryBuffer.append(strTokenArray[i]);
   }
```

```
        return QueryFactory.create(orRuleSHISelectProxyQueryBuffer.toString());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
         ReasonerTripleStoreModelSPARQL#orRuleSHISelectConcept(java.lang.String)
     */
    @Override
    public Query orRuleSHISelectConcept(String contentProxyNode) {
        getLog().debug("generate an instance of SPARQL query for SHI or-rule to select
            linked concepts");
        // orRuleSHISelectConceptQueryTemplate
        String orRuleSHISelectConceptQueryTemplateStr =
            ReasonerTripleStoreModelSPARQLQueryConstants.
            orRuleSHISelectConceptQueryTemplate;
        String contentProxyNodeURIReplaceStr = new String(contentProxyNode);
        String regexReplacementPattern = "@PROXY_CONTENT_NODE@";
        String [] strTokenArray = orRuleSHISelectConceptQueryTemplateStr.split(
            regexReplacementPattern);
        StringBuffer orRuleSHISelectConceptQueryBuffer = new StringBuffer(strTokenArray
            [0]);
        for (int i = 1; i < 19; i++) {
           orRuleSHISelectConceptQueryBuffer.append(contentProxyNodeURIReplaceStr);
           orRuleSHISelectConceptQueryBuffer.append(strTokenArray[i]);
        }
        return QueryFactory.create(orRuleSHISelectConceptQueryBuffer.toString());
    }

    /* (non-Javadoc)
     * @see de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql.
         ReasonerTripleStoreModelSPARQL#orRuleSHISelectAllNotOrContentProxyNodes(java.
         lang.String)
     */
    @Override
    public Query orRuleSHISelectAllNotOrContentProxyNodes(String currentNode) {
        getLog().debug("generate an instance of SPARQL query for SHI or-rule to select
            content proxy which is not or content typed");
        // orRuleSHISelectAllNotOrConceptProxyNodesQueryTemplate
        String orRuleSHISelectAllNotAndConceptProxyNodesQueryTemplateStr =
            ReasonerTripleStoreModelSPARQLQueryConstants.
            orRuleSHISelectAllNotOrConceptProxyNodesQueryTemplate;
        String currentNodeURIReplaceStr = new String(currentNode);
        String regexReplacementPattern = "@CURRENT_NODE@";
        String [] strTokenArray =
            orRuleSHISelectAllNotAndConceptProxyNodesQueryTemplateStr.split(
            regexReplacementPattern);
        StringBuffer orRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer = new
            StringBuffer(strTokenArray[0]);
        orRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer.append(
            currentNodeURIReplaceStr);
        orRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer.append(strTokenArray[1]);
        orRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer.append(
            currentNodeURIReplaceStr);
        orRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer.append(strTokenArray[2]);
        return QueryFactory.create(orRuleSHISelectAllNotAndConceptProxyNodesQueryBuffer.
            toString());
    }

    ...

}
```

Listing 27: Class containing SPARQL query templates for both $\mathcal{ALC}$ and $\mathcal{SHI}$ concepts *(abridged)*

```
/**
 *
 */
package de.tuhh.sts.dl.exptime.tableaux.reasoner.model.db.sparql;

import de.tuhh.sts.dl.exptime.tableaux.reasoner.config.
    OptimizedEXPReasonerConfiguration;

/**
 * @author patrick_un
 *
 */
public final class ReasonerTripleStoreModelSPARQLQueryConstants {

   ////////////////////////////////////////////////////////////
   // ALL TEMPLATE PLACEHOLDER IS MARKED AND DELIMITED WITH @@
   // THE CONTENT TOGETHER WITH THE DELIMITERS MUST BE REPLACED
   // BEFORE THE QUERY STRING IS USED
   ////////////////////////////////////////////////////////////

   ////////////////////////////////////////////////////////////////////
   //
   // query root and-or-graph node and get unexpanded nodes
   //
   ////////////////////////////////////////////////////////////////////
   public static String rootGraphNodeSatOrUnsatQuery =
      "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
          + "> " +
      "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
          + "> " +
      "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
          W3C_OWL_REASONER_BASE_NS + "> " +
      "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
          W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
      "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
          W3C_OWL_REASONER_NODES_NS + "> " +
      "ASK {" +
      "  {" +
      "  nodes:and-or-graph-root-node exptableauxreasoner:nodeStatus nodeStatus:sat. "
          +
      "  nodes:and-or-graph-root-node rdf:type exptableauxreasoner:graphNode " +
      "  } " +
      "UNION" +
      "  { " +
      "  nodes:and-or-graph-root-node exptableauxreasoner:nodeStatus nodeStatus:unsat.
          " +
      "  nodes:and-or-graph-root-node rdf:type exptableauxreasoner:graphNode " +
      "  } " +
      "}";

   public static String selectUnexpandedConceptNodeQuery =
      "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
          + "> " +
      "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
          + "> " +
      "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
          W3C_OWL_REASONER_BASE_NS + "> " +
      "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
          W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
      "SELECT ?node " +
      "WHERE { " +
      "  ?node exptableauxreasoner:nodeStatus nodeStatus:unexpanded. " +
      "  ?node <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS + ">
          exptableauxreasoner:graphNode. " +
      "}";

   ////////////////////////////////////////////////////////////////////////////////
   // graph construction, helper queries and queries for manipulating structures
```

```
55  ////////////////////////////////////////////////////////////////////////////////

    ... (abridged) ...

    ////////////////////////////////////////////////////////////////////
60  //
    //    A L C DL-expressiveness
    //
    ////////////////////////////////////////////////////////////////////

65  ... (abridged) ...

    ////////////////////////////////////////////////////////////////////////
    //
    //    S H I DL-expressiveness
70  //
    ////////////////////////////////////////////////////////////////////////

    public static String selectSHIChildForTopLevelUnionNodesQueryTemplate =
       ////////////////////////////////////////////////////////////////////////////
75     // should incorporate ObjectProperty, DatatypeProperty and subProperty, etc.
       ////////////////////////////////////////////////////////////////////////////
       "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
           + "> " +
       "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
           + "> " +
       "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
           W3C_OWL_REASONER_BASE_NS + "> " +
80     "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
           W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
       "PREFIX nodeContentType:   <" + OptimizedEXPReasonerConfiguration.
           W3C_OWL_REASONER_NODE_CONTENT_TYPE_NS + ">" +
       "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
           W3C_OWL_REASONER_NODES_NS + "> " +
       "SELECT ?topLevelNodeChild ?topLevelNodeChildContentType " +
       "WHERE { " +
85     ////////////////////////
       // owl:Class concepts
       ////////////////////////
       "  { " +
       "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
90     "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type <" + OptimizedEXPReasonerConfiguration.
           W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
       "  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
            " +
       "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
       "  ?topLevelNodeChild exptableauxreasoner:contentType ?
           topLevelNodeChildContentType. " +
       "  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
           W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
95     "  } " +
       "UNION " +
       "  { " +
       "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
       "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type <" + OptimizedEXPReasonerConfiguration.
           W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
100    "  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
            " +
       "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
       "  ?topLevelNodeChild exptableauxreasoner:contentType ?
           topLevelNodeChildContentType. " +
       "  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
           W3C_OWL_CLASS_NS + ">. " +
       "  } " +
105    "UNION " +
       "  { " +
       "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
       "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type <" + OptimizedEXPReasonerConfiguration.
           W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
```

```
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
    topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
"  } " +
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
    topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_CLASS_NS + ">. " +
"  } " +
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
"  } " +
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_CLASS_NS + ">. " +
"  } " +
////////////////////
// owl:ObjectProperty
////////////////////
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_OBJECTPROPERTY_NS + ">. " +
"  } " +
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
```

```
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_OBJECTPROPERTY_NS + ">. " +
"  } " +
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
    topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_OBJECTPROPERTY_NS + ">. " +
"  } " +
////////////////////
// owl:DatatypeProperty
////////////////////
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_DATATYPEPROPERTY_NS + ">. " +
"  } " +
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_DATATYPEPROPERTY_NS + ">. " +
"  } " +
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
    topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_DATATYPEPROPERTY_NS + ">. " +
"  } " +
//////////////////////
// owl:TransitiveProperty
//////////////////////
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_TRANSITIVEPROPERTY_NS + ">. " +
"  } " +
"UNION " +
```

```
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_TRANSITIVEPROPERTY_NS + ">. " +
"  } " +
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
    topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_OWL_TRANSITIVEPROPERTY_NS + ">. " +
"  } " +
////////////////////
// rdfs:subPropertyOf
////////////////////
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_RDFS_SUBPROPERTYOF_NS + ">. " +
"  } " +
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_RDFS_SUBPROPERTYOF_NS + ">. " +
"  } " +
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
    topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
    topLevelNodeChildContentType. " +
"  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
    W3C_RDFS_SUBPROPERTYOF_NS + ">. " +
"  } " +
/////////////////
// owl:inverseOf
/////////////////
"UNION " +
"  { " +
"  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
"  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
     " +
"  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
"  ?topLevelNodeChild exptableauxreasoner:contentType ?
```

```
            topLevelNodeChildContentType. " +
        "  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  } " +
        "UNION " +
        "  { " +
        "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
        "  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
            " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
        "  ?topLevelNodeChild exptableauxreasoner:contentType ?
            topLevelNodeChildContentType. " +
        "  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  } " +
        "UNION " +
        "  { " +
        "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
        "  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
            " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            topLevelNodeChild. " +
        "  ?topLevelNodeChild exptableauxreasoner:contentType ?
            topLevelNodeChildContentType. " +
        "  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  } " +
        ////////////////////////////
        // owl:equivalentProperty
        ////////////////////////////
        "UNION " +
        "  { " +
        "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
        "  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
            " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
        "  ?topLevelNodeChild exptableauxreasoner:contentType ?
            topLevelNodeChildContentType. " +
        "  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_EQUIVALENTPROPERTY_NS + ">. " +
        "  } " +
        "UNION " +
        "  { " +
        "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
        "  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
            " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?topLevelNodeChild. " +
        "  ?topLevelNodeChild exptableauxreasoner:contentType ?
            topLevelNodeChildContentType. " +
        "  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_EQUIVALENTPROPERTY_NS + ">. " +
        "  } " +
        "UNION " +
        "  { " +
        "  <@FIRST_LEVEL_GRAPH_NODES@> rdf:type exptableauxreasoner:graphNode. " +
        "  <@FIRST_LEVEL_GRAPH_NODES@> exptableauxreasoner:hasContent ?topLevelNodeChild.
            " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            topLevelNodeChild. " +
        "  ?topLevelNodeChild exptableauxreasoner:contentType ?
            topLevelNodeChildContentType. " +
        "  ?topLevelNodeChild rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_EQUIVALENTPROPERTY_NS + ">. " +
        "  } " +
        "}";

    ////////////
    // bottom
    ////////////
```

```
public static String
    bottomRuleSHISelectProxyNodeToAnonymousButtomReferenceNodeQueryTemplate =
   "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
       + "> " +
   "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
       + "> " +
   "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
       W3C_OWL_REASONER_BASE_NS + "> " +
   "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
       W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
   "SELECT ?anonymousProxyNode ?buttomReferenceBridgeNode " +
   "WHERE { " +
   "  {" +
   "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?anonymousProxyNode. " +
   "  ?anonymousProxyNode exptableauxreasoner:contentType <" +
       OptimizedEXPReasonerConfiguration.
       W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
   "  ?anonymousProxyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
       W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
   "  ?anonymousProxyNode exptableauxreasoner:hasChild ?buttomReferenceBridgeNode "
       +
   "  } " +
   "UNION " +
   "  {" +
   "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?anonymousProxyNode. " +
   "  ?anonymousProxyNode exptableauxreasoner:contentType <" +
       OptimizedEXPReasonerConfiguration.
       W3C_OWL_REASONER_NODE_CONTENT_TYPE_ATOMIC_TYPE_NS + ">. " +
   "  ?anonymousProxyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
       W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
   "  ?anonymousProxyNode exptableauxreasoner:hasChild ?buttomReferenceBridgeNode "
       +
   "  } " +
   "}";

public static String
    bottomRuleSHISelectRelatedClassConceptsThroughReferenceBridgeNodeQueryTemplate
    =
   "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
       + "> " +
   "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
       + "> " +
   "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
       W3C_OWL_REASONER_BASE_NS + "> " +
   "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
       W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
   "SELECT ?relevantDefinedAndConcepts " +
   "WHERE { " +
   "  {" +
   "  <@BRIDGE_NODE@> exptableauxreasoner:contentType <" +
       OptimizedEXPReasonerConfiguration.
       W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
   "  <@BRIDGE_NODE@> exptableauxreasoner:hasChild ?relevantDefinedAndConcepts. " +
   "  } " +
   "UNION " +
   "  {" +
   "  <@BRIDGE_NODE@> exptableauxreasoner:contentType <" +
       OptimizedEXPReasonerConfiguration.
       W3C_OWL_REASONER_NODE_CONTENT_TYPE_ATOMIC_TYPE_NS + ">. " +
   "  <@BRIDGE_NODE@> exptableauxreasoner:hasChild ?relevantDefinedAndConcepts. " +
   "  } " +
   "} " +
   "ORDER BY ASC(?relevantDefinedAndConcepts) ";

public static String bottomRuleSHIEvaluateClassConceptsQueryTemplate =
   "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
       + "> " +
   "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
       + "> " +
```

```
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "ASK { " +
        "  {" +
        "  <@CONCEPT_TERM_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_CLASS_NS + ">. " +
        "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_COMPLEMENTOF_INVERSUS_CLASS_NS + ">. " +
        "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:isComplementClassOf <
            @CONCEPT_TERM_TWO@>. " +
        "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:hasComplementClass <
            @CONCEPT_TERM_ONE@> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CONCEPT_TERM_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_CLASS_NS + ">. " +
        "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_NOT_TYPE_NS + ">. " +
        "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:isComplementClassOf <
            @CONCEPT_TERM_TWO@>. " +
        "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:hasComplementClass <
            @CONCEPT_TERM_ONE@> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CONCEPT_TERM_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_CLASS_NS + ">. " +
        "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_COMPLEMENTOF_INVERSUS_CLASS_NS + ">. " +
        "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:isComplementClassOf <
            @CONCEPT_TERM_ONE@>. " +
        "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:hasComplementClass <
            @CONCEPT_TERM_TWO@> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CONCEPT_TERM_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_CLASS_NS + ">. " +
        "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_NOT_TYPE_NS + ">. " +
        "  <@CONCEPT_TERM_TWO@> exptableauxreasoner:isComplementClassOf <
            @CONCEPT_TERM_ONE@>. " +
        "  <@CONCEPT_TERM_ONE@> exptableauxreasoner:hasComplementClass <
            @CONCEPT_TERM_TWO@> " +
        "  } " +
        "}";

    public static String bottomRuleSHISelectTopReferencedPropertiesQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "SELECT ?propertyNode " +
        "WHERE { " +
        ////////////// handle different properties ////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?propertyNode. " +
        "  ?propertyNode exptableauxreasoner:contentType <" +
```

```
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OBJECT_PROPERTY_NS + ">. " +
        "  ?propertyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_OBJECTPROPERTY_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?propertyNode. " +
        "  ?propertyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_DATATYPE_PROPERTY_NS + ">. " +
        "  ?propertyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_DATATYPEPROPERTY_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?propertyNode. " +
        "  ?propertyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_TRANSITIVE_PROPERTY_NS + ">. " +
        "  ?propertyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_TRANSITIVEPROPERTY_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?propertyNode. " +
        "  ?propertyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_SUBPROPERTY_NS + ">. " +
        "  ?propertyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_RDFS_SUBPROPERTYOF_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?propertyNode. " +
        "  ?propertyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  ?propertyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_INVERSEOF_INVERSUS_PROPERTY_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?propertyNode. " +
        "  ?propertyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_EQUIVALENTPROPERTY_NS + ">. " +
        "  ?propertyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_EQUIVALENTPROPERTY_NS + "> " +
        "  } " +
        "} " +
        "ORDER BY ASC(?propertyNode) ";

    public static String bottomRuleSHIEvaluatePropertyConceptsQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "ASK { " +
        ////////////////////////////////
        // ask match for ObjectProperty
        ////////////////////////////////
        "  {" +
        "  ?someProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
```

```
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?someProxyNode exptableauxreasoner:hasChild <@PROP_CONCEPT_TERM_ONE@>. " +
        "  <@PROP_CONCEPT_TERM_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_OBJECTPROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:isInversusPropertyOf <
            @PROP_CONCEPT_TERM_TWO@>. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:hasInversusProperty <
            @PROP_CONCEPT_TERM_ONE@> " +
        "  } " +
        "UNION " +
        "  {" +
        "  ?someProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?someProxyNode exptableauxreasoner:hasChild <@PROP_CONCEPT_TERM_TWO@>. " +
        "  <@PROP_CONCEPT_TERM_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_OBJECTPROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:isInversusPropertyOf <
            @PROP_CONCEPT_TERM_ONE@>. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:hasInversusProperty <
            @PROP_CONCEPT_TERM_TWO@> " +
        "  } " +
        "UNION " +
        ////////////////////////////////
        // DatatypeProperty
        ////////////////////////////////
        "  {" +
        "  ?someProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?someProxyNode exptableauxreasoner:hasChild <@PROP_CONCEPT_TERM_ONE@>. " +
        "  <@PROP_CONCEPT_TERM_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_DATATYPEPROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:isInversusPropertyOf <
            @PROP_CONCEPT_TERM_TWO@>. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:hasInversusProperty <
            @PROP_CONCEPT_TERM_ONE@> " +
        "  } " +
        "UNION " +
        "  {" +
        "  ?someProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?someProxyNode exptableauxreasoner:hasChild <@PROP_CONCEPT_TERM_TWO@>. " +
        "  <@PROP_CONCEPT_TERM_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_DATATYPEPROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:isInversusPropertyOf <
            @PROP_CONCEPT_TERM_ONE@>. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:hasInversusProperty <
            @PROP_CONCEPT_TERM_TWO@> " +
        "  } " +
        "UNION " +
        ////////////////////////////////
        // TransitiveProperty
        ////////////////////////////////
        "  {" +
        "  ?someProxyNode exptableauxreasoner:contentType <" +
```

```
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?someProxyNode exptableauxreasoner:hasChild <@PROP_CONCEPT_TERM_ONE@>. " +
        "  <@PROP_CONCEPT_TERM_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_TRANSITIVEPROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:isInversusPropertyOf <
            @PROP_CONCEPT_TERM_TWO@>. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:hasInversusProperty <
            @PROP_CONCEPT_TERM_ONE@> " +
        "  } " +
        "UNION " +
        "  {" +
        "  ?someProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?someProxyNode exptableauxreasoner:hasChild <@PROP_CONCEPT_TERM_TWO@>. " +
        "  <@PROP_CONCEPT_TERM_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_TRANSITIVEPROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:isInversusPropertyOf <
            @PROP_CONCEPT_TERM_ONE@>. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:hasInversusProperty <
            @PROP_CONCEPT_TERM_TWO@> " +
        "  } " +
        "UNION " +
        ////////////////////////////////
        // subPropertyOf
        ////////////////////////////////
        "  {" +
        "  ?someProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?someProxyNode exptableauxreasoner:hasChild <@PROP_CONCEPT_TERM_ONE@>. " +
        "  <@PROP_CONCEPT_TERM_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_RDFS_SUBPROPERTYOF_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:isInversusPropertyOf <
            @PROP_CONCEPT_TERM_TWO@>. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:hasInversusProperty <
            @PROP_CONCEPT_TERM_ONE@> " +
        "  } " +
        "UNION " +
        "  {" +
        "  ?someProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?someProxyNode exptableauxreasoner:hasChild <@PROP_CONCEPT_TERM_TWO@>. " +
        "  <@PROP_CONCEPT_TERM_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_RDFS_SUBPROPERTYOF_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:isInversusPropertyOf <
            @PROP_CONCEPT_TERM_ONE@>. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:hasInversusProperty <
            @PROP_CONCEPT_TERM_TWO@> " +
        "  } " +
        "UNION " +
        ////////////////////////////////
        // EquivalentProperty
        ////////////////////////////////
        "  {" +
```

```
        "  ?someProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?someProxyNode exptableauxreasoner:hasChild <@PROP_CONCEPT_TERM_ONE@>. " +
        "  <@PROP_CONCEPT_TERM_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_EQUIVALENTPROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:isInversusPropertyOf <
            @PROP_CONCEPT_TERM_TWO@>. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:hasInversusProperty <
            @PROP_CONCEPT_TERM_ONE@> " +
        "  } " +
        "UNION " +
        "  {" +
        "  ?someProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?someProxyNode exptableauxreasoner:hasChild <@PROP_CONCEPT_TERM_TWO@>. " +
        "  <@PROP_CONCEPT_TERM_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_EQUIVALENTPROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_INVERSEOF_INVERSUS_PROPERTY_NS + ">. " +
        "  <@PROP_CONCEPT_TERM_TWO@> exptableauxreasoner:isInversusPropertyOf <
            @PROP_CONCEPT_TERM_ONE@>. " +
        "  <@PROP_CONCEPT_TERM_ONE@> exptableauxreasoner:hasInversusProperty <
            @PROP_CONCEPT_TERM_TWO@> " +
        "  } " +
        "}";

    ////////////
    // and
    ////////////
    public static String andRuleSHISelectProxyQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT DISTINCT ?andProxyNode ?andContentProxyNode " +
        "WHERE { " +
        ////////////////////////// handle possible 1st level proxy
            ////////////////////////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?andProxyNode. " +
        "   ?andProxyNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS +
             "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_CLASS_NS + ">. " +
        "  ?andProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?andProxyNode exptableauxreasoner:hasChild ?andContentProxyNode " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?andProxyNode. " +
        "   ?andProxyNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS +
             "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS + ">.
            " +
        "  ?andProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?andProxyNode exptableauxreasoner:hasChild ?andContentProxyNode " +
```

```
        "  } " +
        "UNION " +
        ////////////////////////// handle possible 2nd level proxy
            ////////////////////////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?andBTProxyNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?andBTProxyNode. " +
        "  ?andBTProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "   ?andBTProxyNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS
             + "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS +
            ">. " +
        "  ?andBTProxyNode exptableauxreasoner:hasChild ?andProxyNode. " +
        "  ?andProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?andProxyNode exptableauxreasoner:hasChild ?andContentProxyNode.  " +
        "FILTER ( str(?andBTProxyNode) != str(?andContentProxyNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?andBTProxyNode. " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            andBTProxyNode. " +
        "  ?andBTProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "   ?andBTProxyNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS
             + "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS +
            ">. " +
        "  ?andBTProxyNode exptableauxreasoner:hasChild ?andProxyNode. " +
        "  ?andProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?andProxyNode exptableauxreasoner:hasChild ?andContentProxyNode.  " +
        "FILTER ( str(?andBTProxyNode) != str(?andContentProxyNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?andBTProxyNode. " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?andBTProxyNode. " +
        "  ?andBTProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "   ?andBTProxyNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS
             + "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS +
            ">. " +
        "  ?andBTProxyNode exptableauxreasoner:hasChild ?andProxyNode. " +
        "  ?andProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?andProxyNode exptableauxreasoner:hasChild ?andContentProxyNode.  " +
        "FILTER ( str(?andBTProxyNode) != str(?andContentProxyNode)) " +
        "  } " +
        "}";

    public static String andRuleSHISelectConceptQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT ?andConcept " +
```

```
        "WHERE { " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?andConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_CLASS_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?andConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_RDFS_SUBCLASSOF_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?andConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_EQUIVALENTCLASS_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?andConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_OBJECTPROPERTY_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?andConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_DATATYPEPROPERTY_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?andConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_EQUIVALENTPROPERTY_NS + "> " +
        "  } " +
        "}";

    public static String andRuleSHISelectAllNotAndConceptProxyNodesQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT ?andProxyNode ?andContentProxyNode " +
        "WHERE { " +
```

```
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?andProxyNode. " +
        "   ?andProxyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_CLASS_NS + ">. " +
        "  ?andProxyNode exptableauxreasoner:contentType ?contentType. " +
        "  ?andProxyNode exptableauxreasoner:hasChild ?andContentProxyNode. " +
        "FILTER ( str(?contentType) != \"" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + "\" ) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?andProxyNode. " +
        "   ?andProxyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "  ?andProxyNode exptableauxreasoner:contentType ?contentType. " +
        "  ?andProxyNode exptableauxreasoner:hasChild ?andContentProxyNode. " +
        "FILTER ( str(?contentType) != \"" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + "\" ) " +
        "  } " +
        "}";

    ////////////
    // or
    ////////////
    public static String orRuleSHISelectProxyQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT DISTINCT ?orProxyNode ?orContentProxyNode " +
        "WHERE { " +
        ////////////////////////// handle possible 1st level proxy
            ////////////////////////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?orProxyNode. " +
        "   ?orProxyNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS +
            "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_CLASS_NS + ">. " +
        "  ?orProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?orProxyNode exptableauxreasoner:hasChild ?orContentProxyNode " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?orProxyNode. " +
        "   ?orProxyNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS +
            "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS + ">. "
             +
        "  ?orProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?orProxyNode exptableauxreasoner:hasChild ?orContentProxyNode " +
        "  } " +
        "UNION " +
        ////////////////////////// handle possible 2nd level proxy
            ////////////////////////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?orBTProxyNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?orBTProxyNode. " +
        "  ?orBTProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "   ?orBTProxyNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS
```

```
            + "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS + ">.
             " +
        "  ?orBTProxyNode exptableauxreasoner:hasChild ?orProxyNode. " +
        "  ?orProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?orProxyNode exptableauxreasoner:hasChild ?orContentProxyNode.  " +
        "FILTER ( str(?orBTProxyNode) != str(?orContentProxyNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?orBTProxyNode. " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            orBTProxyNode. " +
        "  ?orBTProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "   ?orBTProxyNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS
            + "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS + ">.
             " +
        "  ?orBTProxyNode exptableauxreasoner:hasChild ?orProxyNode. " +
        "  ?orProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?orProxyNode exptableauxreasoner:hasChild ?orContentProxyNode.  " +
        "FILTER ( str(?orBTProxyNode) != str(?orContentProxyNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?orBTProxyNode. " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?orBTProxyNode. " +
        "  ?orBTProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "   ?orBTProxyNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS
            + "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS + ">.
             " +
        "  ?orBTProxyNode exptableauxreasoner:hasChild ?orProxyNode. " +
        "  ?orProxyNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?orProxyNode exptableauxreasoner:hasChild ?orContentProxyNode.  " +
        "FILTER ( str(?orBTProxyNode) != str(?orContentProxyNode)) " +
        "  } " +
        "}";

    public static String orRuleSHISelectConceptQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:              <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT ?orConcept " +
        "WHERE { " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?orConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_CLASS_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
```

```
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?orConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_RDFS_SUBCLASSOF_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?orConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_EQUIVALENTCLASS_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?orConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_OBJECTPROPERTY_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?orConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_DATATYPEPROPERTY_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  <@PROXY_CONTENT_NODE@> exptableauxreasoner:hasChild ?orConcept. " +
        "  <@PROXY_CONTENT_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_EQUIVALENTPROPERTY_NS + "> " +
        "  } " +
        "}";

    public static String orRuleSHISelectAllNotOrConceptProxyNodesQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT ?orProxyNode ?orContentProxyNode " +
        "WHERE { " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?orProxyNode. " +
        "   ?orProxyNode rdf:type <" + OptimizedEXPReasonerConfiguration.W3C_OWL_CLASS_NS
             + ">. " +
        "  ?orProxyNode exptableauxreasoner:contentType ?contentType. " +
        "  ?orProxyNode exptableauxreasoner:hasChild ?orContentProxyNode. " +
        "FILTER ( str(?contentType) != \"" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + "\" ) " +
        "  } " +
        "UNION " +
        "  {" +
```

```
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?orProxyNode. " +
        "   ?orProxyNode rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "  ?orProxyNode exptableauxreasoner:contentType ?contentType. " +
        "  ?orProxyNode exptableauxreasoner:hasChild ?orContentProxyNode. " +
        "FILTER ( str(?contentType) != \"" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + "\" ) " +
        "  } " +
        "}";

    ////////////
    // H rules
    ////////////
    public static String restriction_H_PositiveRuleSHISelect1LevelIndirectionProxyQuery
        =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT DISTINCT ?restrictionContent  " +
        "WHERE { " +
        //////////////////// 1 LEVEL INDIRECTION || AND CONTENT PROXY FOR THE 3 BOXES
            ////////////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            restrictionProxyBTNode. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        "UNION " +
        //////////////////////1 LEVEL INDIRECTION || OR CONTENT PROXY FOR THE 3 BOXES
            ////////////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
```

```
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            restrictionProxyBTNode. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        " } " +
        "ORDER BY ASC (?restrictionContent) ";

    public static String restriction_H_PositiveRuleSHISelect2LevelsIndirectionProxyQuery
        =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT DISTINCT ?secondLevelIndirectionProxy " +
        "WHERE { " +
        //////////////////////// 2 LEVELS INDIRECTION || AND-AND CONTENT PROXY FOLLOWED BY
            OR-OR CONTENT PROXY FOR THE 3 BOXES ////////////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
```

```
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
890         "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
             restrictionProxyChildCTNode. " +
            "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
             OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
            "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
             secondLevelIndirectionProxy.  " +
            "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
            "  } " +
895         "UNION " +
            "  {" +
            "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
            "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
             restrictionProxyBTNode. " +
            "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
             OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
900         "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
             restrictionProxyChildCTNode. " +
            "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
             OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
            "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
             secondLevelIndirectionProxy.  " +
            "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
            "  } " +
905         "UNION " +
            "  {" +
            "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
            "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
             restrictionProxyBTNode. " +
            "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
             OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
910         "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
             restrictionProxyChildCTNode. " +
            "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
             OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
            "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
             secondLevelIndirectionProxy.  " +
            "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
            "  } " +
915         "UNION " +
            "  {" +
            "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
            "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
             . " +
            "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
             OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
920         "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
             restrictionProxyChildCTNode. " +
            "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
             OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
            "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
             secondLevelIndirectionProxy.  " +
            "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
            "  } " +
925         "UNION " +
            "  {" +
            "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
            "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
             . " +
            "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
             OptimizedEXPReasonerConfiguration.
             W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
```

```
930     "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
935     "UNION " +
        ////////////////////// 2 LEVELS INDIRECTION || AND-OR CONTENT PROXY FOLLOWED BY
            OR-AND CONTENT PROXY FOR THE 3 BOXES ////////////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
940     "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
945     "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
950     "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
955     "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            restrictionProxyBTNode. " +
960     "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
965     "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            restrictionProxyBTNode. " +
970     "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
```

```
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
        " } " +
        "ORDER BY ASC (?secondLevelIndirectionProxy) ";

    public static String
        restriction_H_PositiveRuleSHISelectonPropertyAllValuesFromQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT ?onProperty ?allValuesFrom " +
        "WHERE { " +
        "  {" +
        "  <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_RDF_SYNTAX_TYPE_NS + "> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
```

```
            W3C_OWL_ONPROPERTY_NS + "> ?onProperty. " +
        "   <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ALLVALUESFROM_NS + "> ?allValuesFrom " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_RDF_SYNTAX_TYPE_NS + "> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_CLASS_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onProperty. " +
        "   <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ALLVALUESFROM_NS + "> ?allValuesFrom " +
        "  } " +
        "}  " +
        "ORDER BY ASC (?onProperty) ";

    public static String
        restriction_H_PositiveRuleSHIAskonPropertyIsSubPropertyQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:              <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "ASK {" +
        "  {" +
        "  <@PROPERTY_ONE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_RDFS_SUBPROPERTYOF_NS + "> <@PROPERTY_TWO@>. " +
        "  <@PROPERTY_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_SUBPROPERTY_NS + "> " +
        "  } " +
        "UNION" +
        "  { " +
        "  <@PROPERTY_TWO@> <" + OptimizedEXPReasonerConfiguration.
            W3C_RDFS_SUBPROPERTYOF_NS + "> <@PROPERTY_ONE@>. " +
        "  <@PROPERTY_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_SUBPROPERTY_NS + "> " +
        "  } " +
        "}";

    public static String
        restriction_H_NegativeRuleSHIAskonPropertyIsSubPropertyQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:              <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "ASK {" +
        "  {" +
        "  <@PROPERTY_ONE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_RDFS_SUBPROPERTYOF_NS + "> <@PROPERTY_TWO@>. " +
        "  <@PROPERTY_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_SUBPROPERTY_NS + ">. " +
```

```
        "  <@ASSOCIATED_ALLVALUESFROM_CLASS_CONCEPT@> exptableauxreasoner:contentType ?
            contentTypeToCheck.  " +
        "FILTER ( str(?contentTypeToCheck) != \"" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_COMPLEMENTOF_INVERSUS_CLASS_NS + "\" ) "
            +
        "  } " +
        "UNION" +
        "  { " +
        "  <@PROPERTY_TWO@> <" + OptimizedEXPReasonerConfiguration.
            W3C_RDFS_SUBPROPERTYOF_NS + "> <@PROPERTY_ONE@>. " +
        "  <@PROPERTY_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_SUBPROPERTY_NS + ">. " +
        "  <@ASSOCIATED_ALLVALUESFROM_CLASS_CONCEPT@> exptableauxreasoner:contentType ?
            contentTypeToCheck.  " +
        "FILTER ( str(?contentTypeToCheck) != \"" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_COMPLEMENTOF_INVERSUS_CLASS_NS + "\" ) "
            +
        "  } " +
        " } ";

    ///////////////
    // cut rules
    ///////////////
    public static String restrictionCut_All_RuleSHIAskQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "ASK {" +
        "  {" +
        "    {" +
        "  <@PROXY_BRIDGE_NODE_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_ONE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onPropertyOne. " +
        "   <@PROXY_BRIDGE_NODE_ONE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ALLVALUESFROM_NS + "> ?allValuesFromOne. " +
        "  ?rectusClassOne exptableauxreasoner:hasComplementClass ?allValuesFromOne " +
        "    } " +
        "    { " +
        "  <@PROXY_BRIDGE_NODE_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_TWO@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onPropertyTwo. " +
        "   <@PROXY_BRIDGE_NODE_TWO@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_SOMEVALUESFROM_NS + "> ?someValuesFromTwo. " +
        "  ?rectusClassTwo exptableauxreasoner:hasComplementClass ?someValuesFromTwo " +
        "    } " +
        "  FILTER ( sameTerm(?onPropertyOne, ?onPropertyTwo) && " +
        "           (( (str(?rectusClassOne) = str(?someValuesFromTwo)) || (str(?
            rectusClassTwo) = str(?allValuesFromOne)) ))   ) " +
        "  } " +
        "} ";

    public static String restrictionCut_B_RuleSHIAskQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
```

```
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "ASK {" +
        "  {" +
        "    {" +
        "  <@PROXY_BRIDGE_NODE_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_ONE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onPropertyOne. " +
        "   <@PROXY_BRIDGE_NODE_ONE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ALLVALUESFROM_NS + "> ?allValuesFromOne. " +
        "  ?rectusPropertyOne exptableauxreasoner:hasInversusProperty ?onPropertyOne " +
        "    } " +
        "    { " +
        "  <@PROXY_BRIDGE_NODE_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_TWO@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onPropertyTwo. " +
        "   <@PROXY_BRIDGE_NODE_TWO@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_SOMEVALUESFROM_NS + "> ?someValuesFromTwo. " +
        "  ?rectusPropertyTwo exptableauxreasoner:hasInversusProperty ?onPropertyTwo " +
        "    } " +
        "  FILTER ( sameTerm(?allValuesFromOne, ?someValuesFromTwo) && " +
        "           (( (str(?rectusPropertyOne) = str(?onPropertyTwo)) || (str(?
            rectusPropertyTwo) = str(?onPropertyOne)) ))   ) " +
        "  } " +
        "} ";

    public static String restrictionCut_Five_RuleSHIAskQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "ASK {" +
        "  {" +
        "    {" +
        "  <@PROXY_BRIDGE_NODE_ONE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_ONE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_ONE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onPropertyOne. " +
        "   <@PROXY_BRIDGE_NODE_ONE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ALLVALUESFROM_NS + "> ?allValuesFromOne. " +
        "  ?rectusClassOne exptableauxreasoner:hasComplementClass ?allValuesFromOne. " +
        "  ?rectusPropertyOne exptableauxreasoner:hasInversusProperty ?onPropertyOne " +
        "    } " +
        "    { " +
        "  <@PROXY_BRIDGE_NODE_TWO@> rdf:type <" + OptimizedEXPReasonerConfiguration.
```

```
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_TWO@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "   <@PROXY_BRIDGE_NODE_TWO@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onPropertyTwo. " +
        "   <@PROXY_BRIDGE_NODE_TWO@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_SOMEVALUESFROM_NS + "> ?someValuesFromTwo. " +
        "  ?rectusClassTwo exptableauxreasoner:hasComplementClass ?someValuesFromTwo. " +
        "  ?rectusPropertyTwo exptableauxreasoner:hasInversusProperty ?onPropertyTwo " +
        "    } " +
        "  FILTER (  sameTerm(?onPropertyOne, ?onPropertyTwo) && sameTerm(?
            allValuesFromOne, ?someValuesFromTwo) && " +
        "          (  ( (str(?rectusClassOne) = str(?someValuesFromTwo)) || (str(?
            rectusClassTwo) = str(?allValuesFromOne)) )    ||" +
        "            ( (str(?rectusPropertyOne) = str(?onPropertyTwo)) || (str(?
            rectusPropertyTwo) = str(?onPropertyOne)) )   )    ) " +
        "  } " +
        "} ";

    ////////////////////////////////////////////
    // restrict (incl. transitive properties)
    ////////////////////////////////////////////
    public static String
        restriction_Transitiveness_RuleSHI_Existential_Select1LevelProxyQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT DISTINCT ?restrictionContent  " +
        "WHERE { " +
        ////////////////////// 1 LEVEL INDIRECTION || AND CONTENT PROXY FOR THE 3 BOXES
            ////////////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "   ?restrictionContent exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            restrictionProxyBTNode. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "   ?restrictionContent exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
```

```
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "   ?restrictionContent exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        "UNION " +
        ///////////////////////1 LEVEL INDIRECTION || OR CONTENT PROXY FOR THE 3 BOXES
            ////////////////////////
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "   ?restrictionContent exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            restrictionProxyBTNode. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "   ?restrictionContent exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?restrictionContent.  " +
        "   ?restrictionContent exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionContent)) " +
        "  } " +
        " } " +
        "ORDER BY ASC (?restrictionContent) ";

    public static String
        restriction_Transitiveness_RuleSHI_Existential_Select2LevelProxyQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
```

```
        W3C_OWL_REASONER_NODES_NS + "> " +
    "SELECT DISTINCT ?secondLevelIndirectionProxy " +
    "WHERE { " +
    //////////////////// 2 LEVELS INDIRECTION || AND-AND CONTENT PROXY FOLLOWED BY
        OR-OR CONTENT PROXY FOR THE 3 BOXES ///////////////////////
    "  {" +
    "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
    "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
        . " +
    "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
        OptimizedEXPReasonerConfiguration.
        W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
    "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
        restrictionProxyChildCTNode. " +
    "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
        OptimizedEXPReasonerConfiguration.
        W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
    "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
        secondLevelIndirectionProxy.  " +
    "   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
        OptimizedEXPReasonerConfiguration.
        W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
    "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
    "  } " +
    "UNION " +
    "  {" +
    "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
    "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
        . " +
    "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
        OptimizedEXPReasonerConfiguration.
        W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
    "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
        restrictionProxyChildCTNode. " +
    "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
        OptimizedEXPReasonerConfiguration.
        W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
    "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
        secondLevelIndirectionProxy.  " +
    "   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
        OptimizedEXPReasonerConfiguration.
        W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
    "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
    "  } " +
    "UNION " +
    "  {" +
    "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
    "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
        restrictionProxyBTNode. " +
    "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
        OptimizedEXPReasonerConfiguration.
        W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
    "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
        restrictionProxyChildCTNode. " +
    "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
        OptimizedEXPReasonerConfiguration.
        W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
    "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
        secondLevelIndirectionProxy.  " +
    "   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
        OptimizedEXPReasonerConfiguration.
        W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
    "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
    "  } " +
    "UNION " +
    "  {" +
    "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
    "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
        restrictionProxyBTNode. " +
```

```
"  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
    W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
"  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
    restrictionProxyChildCTNode. " +
"  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
    W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
"  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
    secondLevelIndirectionProxy.  " +
"   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
    W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
"FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
"  } " +
"UNION " +
"  {" +
"  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
"  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
    . " +
"  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
    W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
"  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
    restrictionProxyChildCTNode. " +
"  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
    W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
"  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
    secondLevelIndirectionProxy.  " +
"   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
    W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
"FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
"  } " +
"UNION " +
"  {" +
"  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
"  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
    . " +
"  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
    W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
"  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
    restrictionProxyChildCTNode. " +
"  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
    W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
"  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
    secondLevelIndirectionProxy.  " +
"   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
    W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
"FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
"  } " +
"UNION " +
////////////////////// 2 LEVELS INDIRECTION || AND-OR CONTENT PROXY FOLLOWED BY
    OR-AND CONTENT PROXY FOR THE 3 BOXES ////////////////////////
"  {" +
"  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
"  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
    . " +
"  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
    W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
"  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
    restrictionProxyChildCTNode. " +
"  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
    OptimizedEXPReasonerConfiguration.
```

```
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:TBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            restrictionProxyBTNode. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:DomainConceptBox exptableauxreasoner:hasMember ?
            restrictionProxyBTNode. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
        "UNION " +
```

```
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
        "UNION " +
        "  {" +
        "  <@CURRENT_NODE@> exptableauxreasoner:hasContent ?restrictionProxyBTNode. " +
        "  exptableauxreasoner:RBox exptableauxreasoner:hasMember ?restrictionProxyBTNode
            . " +
        "  ?restrictionProxyBTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_OR_TYPE_NS + ">. " +
        "  ?restrictionProxyBTNode exptableauxreasoner:hasChild ?
            restrictionProxyChildCTNode. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_AND_TYPE_NS + ">. " +
        "  ?restrictionProxyChildCTNode exptableauxreasoner:hasChild ?
            secondLevelIndirectionProxy.  " +
        "   ?secondLevelIndirectionProxy exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "FILTER ( str(?restrictionProxyBTNode) != str(?restrictionProxyChildCTNode)) " +
        "  } " +
        " } " +
        "ORDER BY ASC (?secondLevelIndirectionProxy) ";

    public static String restriction_Transitiveness_RuleSHI_Existential_
SelectOnPropertySomeValuesFromTransitiveQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT DISTINCT ?someValuesFromClassOne " +
        "WHERE { " +
        "  {" +
        "    {" +
        "     <@PROXY_BRIDGE_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "     <@PROXY_BRIDGE_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "     <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onPropertyOne. " +
        "     <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_SOMEVALUESFROM_NS + "> ?someValuesFromClassOne. " +
        "     ?onPropertyOne exptableauxreasoner:contentType ?propertyOneContentType. " +
        "     ?onPropertyOne rdf:type ?propertyOneRDFType " +
```

```
        "    } " +
        "    { " +
        "     <@PROXY_BRIDGE_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "     <@PROXY_BRIDGE_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "     <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onPropertyTwo. " +
        "     <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_SOMEVALUESFROM_NS + "> ?someValuesFromClassTwo. " +
        "     ?onPropertyTwo exptableauxreasoner:contentType ?propertyTwoContentType. " +
        "     ?onPropertyTwo rdf:type ?propertyTwoRDFType " +
        "    } " +
        "    FILTER (" +
        "           sameTerm(?someValuesFromClassOne, ?someValuesFromClassTwo) && " +
        "           (  (str(?propertyOneContentType) = \"" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_TRANSITIVE_PROPERTY_NS + "\") && " +
        "              (str(?propertyOneRDFType) = \"" +
            OptimizedEXPReasonerConfiguration.W3C_OWL_TRANSITIVEPROPERTY_NS + "\")   )
            || " +
        "           (  (str(?propertyTwoContentType) = \"" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_TRANSITIVE_PROPERTY_NS + "\") && " +
        "              (str(?propertyTwoRDFType) = \"" +
            OptimizedEXPReasonerConfiguration.W3C_OWL_TRANSITIVEPROPERTY_NS + "\")   ) "
             +
        "         )" +
        "  } " +
        "} ";

    public static String restriction_Transitiveness_RuleSHI_Universal_
SelectOnPropertyAllValuesFromTransitiveQueryTemplate =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT DISTINCT ?allValuesFromClassOne " +
        "WHERE { " +
        "  {" +
        "    {" +
        "     <@PROXY_BRIDGE_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "     <@PROXY_BRIDGE_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "     <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onPropertyOne. " +
        "     <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ALLVALUESFROM_NS + "> ?allValuesFromClassOne. " +
        "     ?onPropertyOne exptableauxreasoner:contentType ?propertyOneContentType. " +
        "     ?onPropertyOne rdf:type ?propertyOneRDFType " +
        "    } " +
        "    { " +
        "     <@PROXY_BRIDGE_NODE@> rdf:type <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ANONYMOUS_CLASS_NS + ">. " +
        "     <@PROXY_BRIDGE_NODE@> exptableauxreasoner:contentType <" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_RESTRICTION_NS + ">. " +
        "     <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_ONPROPERTY_NS + "> ?onPropertyTwo. " +
        "     <@PROXY_BRIDGE_NODE@> <" + OptimizedEXPReasonerConfiguration.
```

```
            W3C_OWL_ALLVALUESFROM_NS + "> ?allValuesFromClassTwo. " +
        "     ?onPropertyTwo exptableauxreasoner:contentType ?propertyTwoContentType. " +
        "     ?onPropertyTwo rdf:type ?propertyTwoRDFType " +
        "    } " +
        "    FILTER (" +
        "           sameTerm(?allValuesFromClassOne, ?allValuesFromClassTwo) && " +
        "           (  (str(?propertyOneContentType) = \"" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_TRANSITIVE_PROPERTY_NS + "\") && " +
        "               (str(?propertyOneRDFType) = \"" +
            OptimizedEXPReasonerConfiguration.W3C_OWL_TRANSITIVEPROPERTY_NS + "\")   )
            || " +
        "           (  (str(?propertyTwoContentType) = \"" +
            OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_CONTENT_TYPE_TRANSITIVE_PROPERTY_NS + "\") && " +
        "               (str(?propertyTwoRDFType) = \"" +
            OptimizedEXPReasonerConfiguration.W3C_OWL_TRANSITIVEPROPERTY_NS + "\")   ) "
             +
        "         )" +
        "  } " +
        "} ";

    public static String restrictionTransitionalRuleSHISelectAllOtherTBoxMembers =
        "PREFIX rdf:            <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_NS
            + "> " +
        "PREFIX rdfs:           <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SCHEMA_NS
            + "> " +
        "PREFIX exptableauxreasoner:<" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_BASE_NS + "> " +
        "PREFIX nodeStatus:     <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODE_STATUS_NS + "> " +
        "PREFIX nodes:             <" + OptimizedEXPReasonerConfiguration.
            W3C_OWL_REASONER_NODES_NS + "> " +
        "SELECT ?tboxMemberNode " +
        "WHERE { " +
        "  {" +
        "  <" + OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_REASONING_TBOX_NS + ">
             exptableauxreasoner:hasMember ?tboxMemberNode. " +
        "  ?tboxMemberNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS
             + "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_CLASS_NS + "> " +
        "  } " +
        "UNION " +
        "  {" +
        "  <" + OptimizedEXPReasonerConfiguration.W3C_OWL_REASONER_REASONING_TBOX_NS + ">
             exptableauxreasoner:hasMember ?tboxMemberNode. " +
        "  ?tboxMemberNode <" + OptimizedEXPReasonerConfiguration.W3C_RDF_SYNTAX_TYPE_NS
             + "> <" + OptimizedEXPReasonerConfiguration.W3C_OWL_ANONYMOUS_CLASS_NS + ">
             " +
        "  } " +
        "}";

}
```