A Framework to Process and Exchange Logical Rules in Multiple Rule Languages

by

Ismail Akbari

Master's in Information Technology Engineering, Iran University of Science and Technology, Iran 2009

Bachelor in Computer Engineering, Tarbiat Moallem University, Iran, 2006

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

In the Graduate Academic Unit of Computer Science

Supervisors:	Yevgen Biletskiy, PhD, Electrical and Computer Engineering
	Weichang Du, PhD, Computer Science
Examining Board:	Monica Wachowicz, PhD, Geodesy and Geomatics Engineering
	Suprio Ray, PhD, Computer Science
	Harold Boley, PhD, Adjunct Prof. Computer Science
External Examiner:	Nick Bassiliades, PhD,
	Department of Informatics, Aristotle University of Thessaloniki

This dissertation is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

November, 2018

ABSTRACT

Web rule languages have been developed for the Web-based interchange of rules, in particular business rules, business policies, and any business or application logic that can be presented with rules.

The primary goal of this dissertation is to create methods for rule interchange between selected rule markup languages, as well as to develop a rule engine for a subset of W3C's RIF language. Logical rule interchange is the act of transforming rules presented in one rule language to another. The rule interchange is done from the Notation3, POSL, RuleML, SWRL rule languages to RIF-BLD.

In this dissertation, to enable rule interchange in the Semantic Web, a framework has been proposed. The framework contains different rule grammars, parsers, visualizers and translators as well as a rule engine. A grammar, parser and rule visualizer are developed for each of the N3, POSL and RIF-BLD languages. Also, rule translators from N3, POSL, SWRL, and RuleML to RIF-BLD are developed.

As a central component, a rule engine for the RIF-BLD language has been developed. The rule engine comprises both forward and backward reasoning. All the grammars, parsers, visualizers, translators and the rule engine are part of the framework. The translators and the rule engine have been separately evaluated with various use cases and with a case study on the independently provided Port Clearance Rules.

DEDICATION

I would like to dedicate this work to all wildlife conservationists around the globe who try hard to make this planet a better place for wild animals.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Professor Yevgen Biletskiy, for his support and guidance throughout my study and carrying out the research described in this dissertation. I am also grateful to my co-supervisor, Professor Weichang Du, for his help and support during my Ph.D. time. I would also like to thank Professors Harold Boley and Bruce Spencer for always answering my questions without any hesitation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A logical rule is a statement in the general form of “IF p THEN q” that represents a decision. Decisions can be made by human or machine. A business rule is a logical rule that defines some aspects of a business or business application. Business rules are rules that shape business behavior, process flow, business policy and decision making. These rules tell an organization what to do in details. Every business application has its own set of logical rules which can be hard-coded in the application code or are held separately as decision tables in formats such as a spreadsheet or simple text file.

Business rules play a crucial role in the process of any business decision-making initiative and are intended to influence, guide or support business decisions and behavior. From a system or application perspective, business rules are business logic defined in a formal manner. Business rules should be explicitly defined describing how a particular decision can be made without ending to an ambiguity. Logical rules in business (business rules) should also be easy to define and manage by regular business personnel rather than any business specialist. They should be easily understood by a broad range of business audience[1].

Business rules are often created, stored and maintained by business analysts in a user-friendly format such as Excel spreadsheets or even in natural language business documents. Logical rules in business often constitute a significant part of knowledge-based applications e.g. expert systems, e-business, e-learning, and other applications. Business rules are also stored in databases or hard-coded in business applications. The problem with

these two last forms is that anytime there is even a small need to update or maintain business rules, business analysts must inform database administrators or software team to update the database or software code respectively. This makes the adaptation or migration of existing systems to new requirements very difficult. Instead, business rules must be decoupled from application code and databases which make them easy to maintain by business analysts.

Some of the advantages of the proposed business rule approach in comparison to traditional systems are as follows: Separation of business logic from detailed software implementation makes maintaining rules easier with less burden on the business application. As a result, it decreases the cost of rule modification. Also, separation of business rules makes sharing them among different businesses or business applications easier [2]. Business rules can be expressed in natural languages, user-friendly formats (such as decision tables or trees), and programming languages.

Web rule languages have been developed for the Web-based interchange of rules, in particular business rules, business policies and any other business or application logic that can be presented in the form of rules. Web rule languages provide machine interpretation, automated processing, interchange and translation of rules (usually presented in XML-based syntax) on the Web. In general, Web rule languages come in two different formats. The first format is used by machines and has an XML-based or concrete syntax. These languages are Rule markup languages. The second format are rule languages with an abstract or presentation syntax, which is a human-friendly syntax [3]. In literature, there are several rule languages aiming at rule interchange, transforming and building a general rule markup language, including Rule Interchange Format (RIF) [4], Rule Markup

Language (RuleML) [5], and Semantic Web Rule Language (SWRL) [6]. There are other Semantic Web rule languages out there such as Positional-Slotted Language (POSL) [7] and Notation3 (N3) [8] which present business rules in an abstract syntax. Businesses can publish their business rules in any standard rule language based on their preferences. For a review of different rule markup languages and Semantic Web languages refer to [3].

1.1 Motivation

Logical rules are becoming ubiquitous in modern industry and are usually created, stored, and maintained by business analysts, knowledge engineers and software engineers in various formats. Classically, logical rules are logic constructs (e.g. "IF-THEN" type), and they are often represented using decision tables or decision trees. Technically, logical rules in business can be implemented using a programming language or using controlled English. There are many specific solutions for creation and maintenance of business rules. For instance, Microsoft Excel tables can be deployed as a user-friendly way to build decision tables. Systems like Drools [9] or Jess [10] provide platforms to build and maintain more complex business rules [11].

There are many logical rule languages as well as open source or commercial rule systems that implement business rules. Every rule language has its own syntax and semantics and every rule system has its own version of a rule language. This situation makes sharing of rules among rule systems and business owners, which use these rule systems, a difficult and time-consuming task. Generally, there can be two solutions to cope with this problem. The first solution is to have (for every pair of rule languages) a rule translator to translate rules from the first language to the second language and another translator for the opposite

way. A translator to transform rules from rule language A to rule language B is different from a translator to translate rules from rule language B to rule language A. Therefore, for exchanging rules between two rule languages, two translators are required. If there are n rule languages available, there is a need for $n \times (n - 1)$ translators to exchange rules among them. This does not seem to be efficient. The second solution is to have a generic rule language as an intermediary between other rule languages. Rules in any rule language can be translated to and from this rule language. RuleML and RIF are examples of rule languages that fall into this category.

The Rule Interchange Format (RIF) was introduced by W3C, beginning in 2005, to create a standard for exchanging rules among rule systems, in particular among Web rule languages and engines. RIF focuses on rule exchange rather than trying to develop a single one-fits-all rule language. Moreover, it can be used as a base language to define new Web rule languages. Before using RIF as an intermediary for interchanging rules of Web rule languages let's have a look at different categories of rule languages and their aspects and how RIF can help to interchange rules between them.

Rule languages can be broadly categorized as belonging to one of three different categories: (1) normative rules (or structural rules, or integrity constraints), (2) deductive rules (or deduction rules or database views or constructive rules), and (3) active or reactive rules. Deductive languages are further classified into logic programming (LP) and first-order logic (FOL). These languages are often evaluated top-down in response to a query. Reactive languages are further classified into production rules (PR) and event-condition-action rules (ECA). These languages are often evaluated bottom-up in response to changes in the knowledge base. Normative rules also known as integrity constraints (IC) [12], put

constraints on the data or logic of an application. Normative rules describe the disallowed inconsistencies rather than inferring new knowledge.

Knowing the dimensions or aspects of rule languages such as syntax, semantics, and limitations help to have a better understanding of different parts of exchanging rules between rule languages or rule systems. In Figure 1.1 the different discriminators (dimensions) of rule languages [13] are shown. Figure 1.1 does not have a comprehensive list of all the dimensions of rule languages. For a complete list of discriminators, you can refer to [13].

Rule discriminators include Syntactic, Syntactic-entailing-Semantic, Semantic, Pragmatic and discriminators for ECA rules. Every rule language may have a different collection of the above paradigms. Therefore, two different rule languages may share little in the way of syntax and semantics. Moreover, there can be big differences even between rule languages/systems within the same paradigm [14].

Every rule language including RIF dialects has two aspects: syntax and semantics. The syntax of a rule language needs to have a vocabulary, definition of terms and formulas to be able to create, present, and check that if a rule is structurally correct in that language. The semantics of a rule language show that whether a rule in a rule language is true or false with respect to an interpretation of the symbols in the language. Therefore, the first step in exchanging rules and make interoperability between rule languages possible is to know the type of the rule language (normative, deductive, and reactive), dimensions and limitations of the language, and finally the syntax and semantics of a rule language.

It is unavoidable that rule systems and languages need to communicate with each other (e.g., interchange rules) and thus problems in sharing rules arise. Due to the heterogeneity

of different languages, creating a generally accepted interchange format is not an easy task. RuleML, R2ML (REWERSE Rule Markup Language) [15] and RIF are proposed to markup and also facilitate rule interchange. RIF is a W3C standard whose goal is to construct a Web standard for exchanging rules between different rule languages. One of the advantages of mapping rules to one intermediate language such as RIF is that it can noticeably reduce the number of translators, especially when the number of rule languages is arising.

Rule interchange is required to be semantic preserving, therefore, the rules can be exchanged from one language to another without losing information. Specifically a rule in one language that is evaluated to a given truth value with respect to a given interpretation, when translated to a rule in a second language needs to be evaluated to the same truth value with respect to a similar interpretation. RIF can be used as a promising language to help exchange rules among other rule languages. However, there is one issue with respect to semantics which needs attention while developing translators to exchange rules among rule languages and that is the preservation of truth values. Since there are different categories of rule languages, RIF as one rule language is not enough, and there should be a specific language for exchanging rules in each category. Therefore, RIF has introduced dialects (RIF-Core, RIF-BLD, RIF-PRD, etc.). In other words, for every category of rule languages, there is a version of RIF that can help to exchange rules among languages in that category.

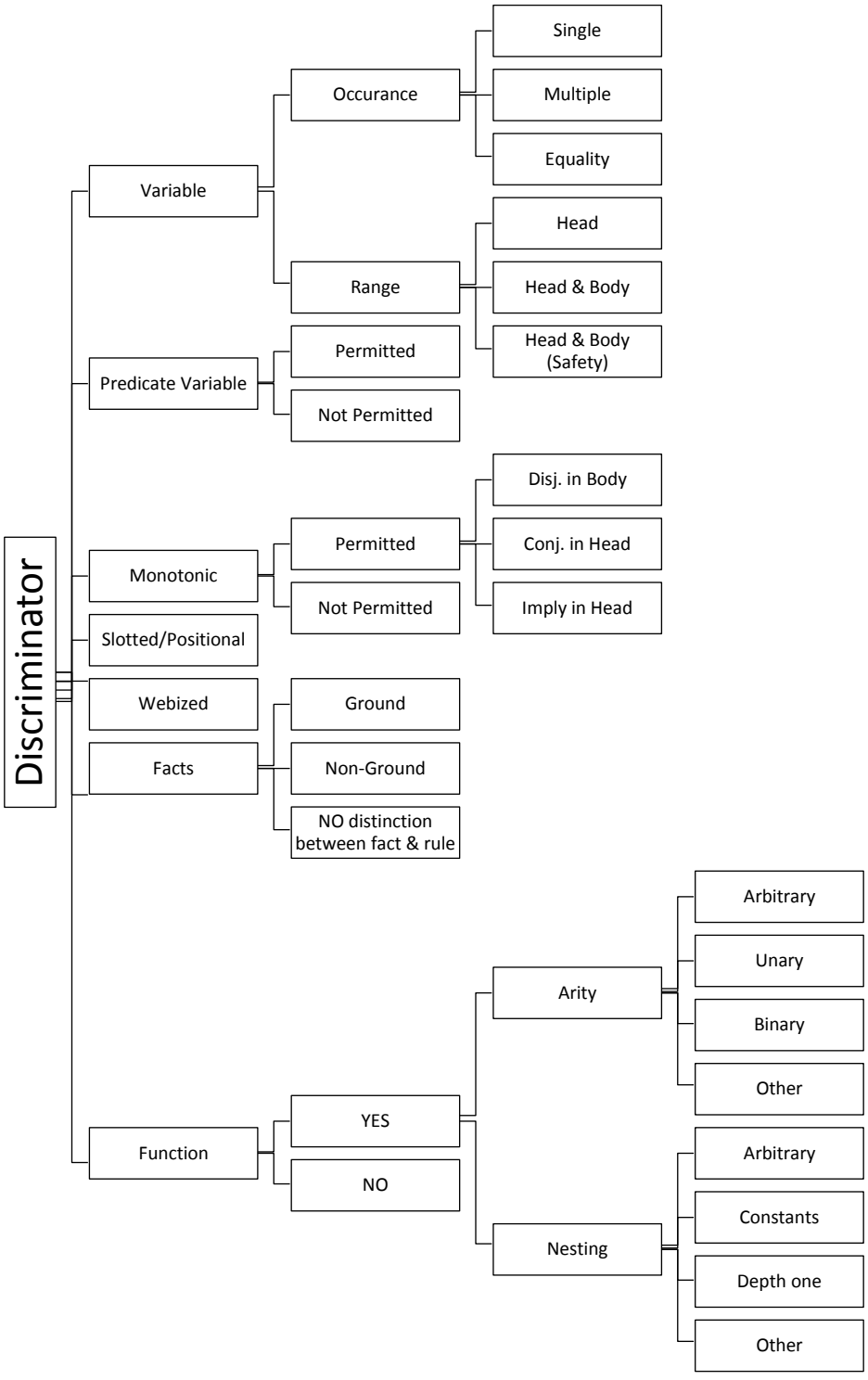

Figure 1.1. Discriminators (dimensions) of rule languages

All the challenges discussed above bring a need to find solutions that will ease the process of rule interchange between various logical rule languages. These challenges that need to be dealt with are briefly listed here:

- A rule language can belong to one of three completely different rule categories.
- Every rule language has its own syntax and semantics and every rule system has its own version of a rule language.
- There can be big differences even between rule languages/systems within the same rule language category.
- Syntax, semantics, limitations and dimensions of every rule language (Figure 1.1) is different from another rule language.
- There are various logical rule languages and rule knowledge bases available with minimum rule sharing capabilities.

These challenges are an obstacle in the way of rule interchange and sharing and make them a hard and time-consuming work. Moreover, the works that have been done in literature in the field of rule interchange and sharing between rule languages are very small. Also, there is no work done in rule interchange between N3, SWRL, RuleML and RIF-BLD. These reasons as well as the challenges mentioned above, have motivated undertaking this dissertation and formulating the dissertation goals and objectives to achieve these goals.

1.2 Objectives

The main objective of the present thesis is to make rule interchange and sharing between different rule markup languages possible. The thesis tries to overcome challenges that are in the way of rule interchange and sharing. It proposes solutions that can make rule

interchange a feasible and easier task. The objective includes translating rules from Notation3, POSL, SWRL, and Unary/Binary Datalog RuleML languages to RIF-BLD language. For each of these languages, a translator that can translate rules in that language to RIF-BLD language is needed. Another objective is this thesis is to build a RIF-BLD rule engine. Building a framework to gather rule translators and the rule engine in one place is also the objective.

There are situations in rule interchange that are not in the scope of this dissertation and the proposed framework will not cover them. However, these situations can be resolved as part of other researchers' work. For example, if a rule from the above rule languages does not fall into the deductive rules category, for example, if it is a production rule, then that rule translation is out of the scope of this thesis. In addition, negation in rules is not supported in this framework. If a rule base from a language includes negation, the framework does not cover its translation and ignores it. The framework is not limited to the mentioned languages, so other rule languages can be added to it as modules. The other objective of this dissertation is to develop a prototype rule engine for the RIF-BLD language. Since currently there is no rule engine available in the literature for RIF language, developing an engine would make a big impact in RIF language popularity and its application in various domains.

1.3 Research Challenges

Rule markup languages make publishing and transferring business rules easy and possible due to their XML-based syntax nature. There are several popular rule markup languages in the literature and they are successfully used by business owners and academics. However,

flexibility and usability of rule markup languages are limited due to the insufficient translation methods to make rule interchange and sharing possible between different business rule systems or rule engines. The scope of this thesis is to tackle problems facing sharing, translation, and interchange of following rule languages using RIF-BLD as a lever: Notation3, POSL, SWRL, and Datalog RuleML. Trying to overcome these challenges, reinforces the idea behind rule markup languages which is sharing, transformation, and reusability of rules on the Web. This work is going to find solutions for the following challenges:

- There is no Notation3 (N3) to RIF-BLD rule sharing solution in literature. Therefore, a translator should be developed from scratch to transform rules in Notation3 to RIF-BLD. The challenge will be developing a translator from scratch while considering both N3 and RIF-BLD syntax (alphabet, formulas, etc.), semantics, language limitations. Another challenge is that what dimensions of a rule language, N3 supports and how to convert them to RIF-BLD. Also, it should be investigated that if there are features of N3 that are not supported in RIF-BLD and cannot be translated without losing meanings. Translating rules in Notation3 to RIF opens more doors in rule sharing field and allows rule bases written in Notation3 be able to communicate with other rule base systems.
- There is no POSL to RIF-BLD translator in literature. All the translation considerations of N3 to RIF-BLD apply in POSL to RIF-BLD rule interchange as well. A translator needs to be developed from scratch. Since the N3 and POSL languages have a human-readable syntax, translating their rules to RIF-BLD first

requires an N3/POSL rule parser to identify different parts of a rule or fact in N3/POSL rule bases. Developing this parser is another challenge.

Having a translator to transform POSL rules to RIF-BLD can make the path of rule interchange smoother. In particular, when there are existing tools such as OpenL tablets [16] that can translate rules in the form of decision tables (such as spreadsheets and text files) to POSL [17].

- SWRL has both an XML-based syntax for machine interoperability and an abstract (human-readable) syntax. A translator is needed to convert SWRL rules to RIF-BLD rules. This translation can be either through SWRL XML-based syntax to RIF-BLD XML syntax or through SWRL presentation syntax to RIF-BLD presentation syntax. If the translation is through SWRL's abstract syntax, a parser would be needed as well. If the translation is through SWRL's XML syntax, an XSL stylesheet should be developed. In both cases, SWRL syntax, semantics and language limitations should be kept in mind (same as N3 and POSL) while translating SWRL rule to RIF-BLD.
- Datalog RuleML is a sublanguage of RuleML, and it is suitable to create function-free Horn logic knowledge bases and queries [18]. A translator in the form of XSL stylesheet needs to be developed from scratch to translate Datalog RuleML rules to RIF rules. Dlex [19] has been introduced as the common sublanguage of Datalog RuleML and RIF-Core. Dlex is restricted to positional arguments and non-conjunctive rule conclusions, and it allows equality plus externals (built-ins as defined in RIF-DTB [20]) in rule premises only [19].

- Another challenge is developing an inference engine for RIF-BLD language since there is no rule engine for RIF-BLD in the literature. Some of the challenges of developing any rule engine include the complexity of building a rule parser, a parser for built-in functions, backtracking algorithm design, unification algorithm design, memory management, optimization, etc.

A common challenge in translating from a rule markup language to another is preserving meanings during translation. In other words, result rules after translation should carry the same semantics as the origin rules. However, this task is not always feasible and in some cases, losing meanings after translating rules in one markup language to another is inevitable. The reason is that every rule language has its own aspects of rule logics, which may be different from another rule language (see Figure 1.1).

1.4 Research Contributions

The contributions of this dissertation are developing rule parser, visualizer, and translator for Notation3, POSL, SWRL, Datalog RuleML and RIF-BLD languages. Another contribution is developing a forward/backward reasoning engine for RIF-BLD language. Below lists these contributions in details:

1. A translator component from Notation3 to RIF-BLD is developed. The translator maps Notation3 rules in a rule base (e.g. in a .n3 file) to RIF-BLD presentation syntax. The converted rules then can be mapped from RIF-BLD presentation syntax to RIF-BLD XML syntax if desired which makes rule exchange through machine easy because of its XML-based nature. In addition to a translator, there is also an N3 rule parser that parses rules in N3 abstract syntax. This option gives the

opportunity to the user to use the framework to write, modify, deploy or parse Notation3 rules as well. The N3 parser can also visualize the input rules after parsing them. This option gives a graphical view of the rules to the user which can be beneficial in rule debugging.

2. A POSL component similar to the N3 component is developed and built in the framework. This component contains a POSL rule parser, visualizer, and translator. The POSL parser verifies that POSL rules to be syntactically correct. It also can be used for deploying POSL rules through the framework. The rule visualizer shows the input rules in a graphical view after parsing by the parser. The translator part of the component translates POSL rules to RIF-BLD rules. In literature, there is OOJDREW tool [21], which is a rule engine, and among other things, it can also parse and convert POSL rules to Object-Oriented RuleML.
3. A built-in XSL stylesheet in the framework aims at translating SWRL rules (in XML-based syntax). XSL stylesheet can be used separately from the framework too. It converts SWRL rules to RIF-BLD XML-based syntax. There is also another XSL stylesheet that translates latest version at the time of Datalog RuleML rules (version 1.02) to RIF-BLD rules. The XLS stylesheets are translators for XML-based SWRL and Datalog RuleML. These stylesheets are in their early stages and it should be noted that there are situations where SWRL or Datalog RuleML rules cannot be translated to RIF-BLD without losing meaning (semantics). Also, there are situations where the translation is out of the scope of this dissertation (e.g. negation). In these cases, the input rules still will be translated with ignoring the parts that cannot be translated. The SWRL and Datalog RuleML XSL translators

can be used (with some changes) as a starting point in transforming rules in these languages to other rule markup languages rather than RIF.

4. The final contribution in this dissertation is focused on RIF-BLD and is the proposed RIF-BLD rule engine. The RIF-BLD component of the framework includes a RIF-BLD parser, visualizer, translator and an inference engine. The RIF-BLD parser is used as a part of the rule engine as well. The parser can parse rules in RIF-BLD or RIF-Core presentation syntax. The RIF-BLD translator in the framework can translate RIF-BLD rules in presentation syntax to their equivalents in XML-based syntax. The proposed RIF-BLD rule engine has both forward and backward reasoning capabilities. It can answer queries as well as infer new knowledge from available rules. Inferencing new knowledge in based on universal and existential formulae in the rule base. The details of the rule engine implementation are described in the appropriate chapter in this dissertation.

1.5 Thesis Organization

In Chapter 2, a review of background research areas related to the works done in this dissertation is presented. Chapter 2 describes what a rule markup language is and some of the popular rule markup languages are explained. Moreover, a summary on a selection of Business Rule Engines (BREs) and Business Rule Management Systems (BRMS) is included in Chapter 2. ANTLR (Another Tool for Language Recognition) [22] is an API that is used in this thesis to help in developing parsers. A short introduction to ANTLR is brought in chapter 2 as well.

Chapter 3 has a review on the related works that have been done on business rule interchange, translation and sharing in recent years. Chapter 4 describes the framework and its components. It explains how the framework can be used to interchange rules between rule markup languages. Next following chapters describe in details the different parts of the framework and other works that have been done in this dissertation. Chapter 5 introduces the Notation3 component of the framework and how N3 rules can be translated to RIF-BLD rules. In chapter 6, a detailed description of the POSL module of the framework including POSL parser, visualizer and translator is presented. POSL parser can be used to write, edit, deploy and visualize POSL rules. POSL translator converts rules in POSL to RIF-BLD.

Chapter 7 describes how SWRL and Datalog RuleML rules (in XML-based syntax) are translated to rules in RIF-BLD serialization syntax using the XSLT transformation files. Chapter 8 is devoted to the RIF-BLD language. This chapter first describes in details RIF-BLD language and how the RIF-BLD parser is created. Then, the focus goes to RIF-BLD rule engine and its implementation details. The rule engine can answer queries as well as infer new knowledge from RIF-BLD rules. Chapter 8 also defines RIF-BLD translator which maps RIF-BLD rules from RIF-BLD human-readable syntax to its XML-based syntax.

Chapter 9 shows the details of the system design. It shows how to use the GUI of the framework's implementation, different modules of the framework (including the rule engine) and their functionalities. The framework includes rule parsers, visualizers, translators, as well as rule inference engine and query answering module which are

described in more details in chapter 9. Chapter 9 shows how the rule engine infers new knowledge or answer queries as well.

Chapter 10 contains a case study. The case study is based on a challenge from Decision Management (DM) community. Decision Management community is an initiative started in 2014 to facilitate sharing of knowledge concerning decision management. This case study is based on the DM 'challenge of March 2016' which consists of creating decision models from the structured text English of Port Clearance Rules.

In chapter 11 evaluation results are presented. The framework implementation is applied to various rule bases and the results are given in this chapter. Chapter 12 concludes the dissertation and gives an overview of the future works that can be done to make rule interchange easier. Appendices at the end of the dissertation show more implementation details or more use cases used in this thesis for business rule interchange.

Chapter 2

Background

A rule is a statement in the general form if “condition” then “action”. A business rule is a statement that provides detailed guidance about how a business policy or procedure can be translated into action. Business rules play a crucial role in the process of any business decision-making initiative and are intended to influence, guide or support business decisions and behavior. From a system or application perspective, business rules are business logics defined in a formal manner. Business rules should be explicitly defined describing how a particular decision can be made. Rule markup languages are the main approach for using rules on the Web. They allow rules to be published, exchanged and transferred on the Web. In general, rule markup languages come in two different syntaxes: XML-based syntax that is used by machines and human-readable syntax [3].

This chapter first reviews common rule markup languages and their features. Then, a summary of a selection of business rule engines and Business Rule Management Systems (BRMS) is described. ANTLR (Another Tool for Language Recognition) is a language recognizer and parser generator. A short introduction to ANTLR is brought in this chapter as well.

2.1 Web Rule Languages

Semantic Web stack has a layer for rule languages. On another hand, businesses and business applications use rules in any aspect of their decision-making processes. Web rule languages allow sharing and reuse of existing rules on the Web. Having rule markup

languages is the first step in rule sharing between business applications on the Web. Some of the anticipations from rule markup languages are expressiveness, rule interchange, rule integration, and rule language interoperability [23]. Authors in [3] state that some of the purposes of rule markup languages are to allow publication, interchange, and sharing of rules. Recently, there have been several efforts and standards aiming at rule interchange such as RuleML [5], RIF [4], SWRL [6], R2ML [15] and others.

Generally, a rule consists of consequence and antecedent, containing clauses combined through logical operators. If all the conditions (premises) in antecedent turn out true, then the consequence part of the rule will result in true (and actions may fire). A clause in antecedent or consequence is in the form of a subject-relation-object, where subject and object can be literals, variables, lists or other types of data structures. Relation is a function or predicate [23]. In addition to rules, axioms or facts can be part of a rule base. Based on Figure 1.1 in chapter 1 (resulted from [13] and [23]) some of the features of Web rule languages that can be used to differentiate them are:

- Variables: specify that if a rule language allows variables. If so, variable occurrence, equality, and range should be determined. The variable occurrence can be a single or multiple occurrences. Variable range defines that if variables are only allowed in rule body or in both rule body and rule head. Moreover, permitting variables in rule predicates should be determined.
- Monotonicity: is permitted or not. Monotonicity of entailment is a property of many logical systems that states that the hypotheses of any derived fact may be freely extended with additional assumptions.

- Slotted vs Positional: specifies that atoms in a rule are limited to positional atoms or slotted atoms are permitted too. Slotted atoms are n-ary atoms with a set of 'attribute=value' pairs.
- Webized: specifies that if a rule language supports URIs as object identifiers (OID) for Constants, Functions, Slots, Predicates and so on.
- Facts: in a rule language, facts can be ground (variable-free) facts or variables in facts are permitted.
- Functional Terms: specify that whether functional terms are allowed in a rule language. If so, function arity and nesting should be determined. Function arity can be unary, binary or other arities. Function nesting defines that if functions arguments can be in depth one or arbitrary nesting is allowed.
- Built-in functions: are functions that compute values of variables in rule clauses. Typical built-in functions include mathematical, Boolean, date-time and string functions.
- Logical quantifiers: specify the scope of possible quantity of variables in a rule formula. There are two types of quantifiers: universal and existential. Universal quantifiers specify that predicates in a rule formula can be satisfied for every member in the domain of variables. Existential quantifiers specify that predicates in a rule formula are valid for at least one member in the domain of variables (universe of discourse).
- Logical operators: Logical operators are used to build more complex rules and usually include conjunction (and), disjunction (or), negation (not) and implication (if-then).

- Data types: specify constraints on the possible values of literals. Common data types include string, numerical values, date/time, list and XML Schema data types.
- Rule set and rule name: rule set is used to group related rules together. Rule sets can extend the scope of a variable used in a rule to be used in another rule from the same rule set. Rule names are used to name a rule or rule set and can be beneficial in situations like retrieving a rule by its name (e.g. from an XML database).
- Production operations: specify actions that can be executed by a rule reasoner after the antecedent of a rule becomes valid. These actions are in addition to the consequent becoming valid. Actions in production rules can be rule or fact insertion, retraction or modification.
- Rule interchange format: allows different rule systems to interchange rules. Rule systems can export their rules using standards such as XML and JSON.

A detailed view of different aspects of rule markup languages is available at [3], [23] and [13]. In the following, some of the popular rule markup languages in Semantic Web are briefly described.

2.1.1 RuleML

The Rule Markup Language (RuleML) is a system of three families of XML-based rule languages including Deliberation, Reaction, and Consumer RuleML. RuleML has been introduced by RuleML Initiative [5]. It has a modular, hierarchical nature that covers different types of rules including derivation rules, integrity constraints, production rules and reaction rules. Figure 2.1 (borrowed from [5]) shows the RuleML family of languages. Deliberation RuleML is a language from RuleML family that covers declarative rules.

Unary/Binary Datalog RuleML (covered in the scope of this dissertation) is a function-free sub-language of Deliberation RuleML.

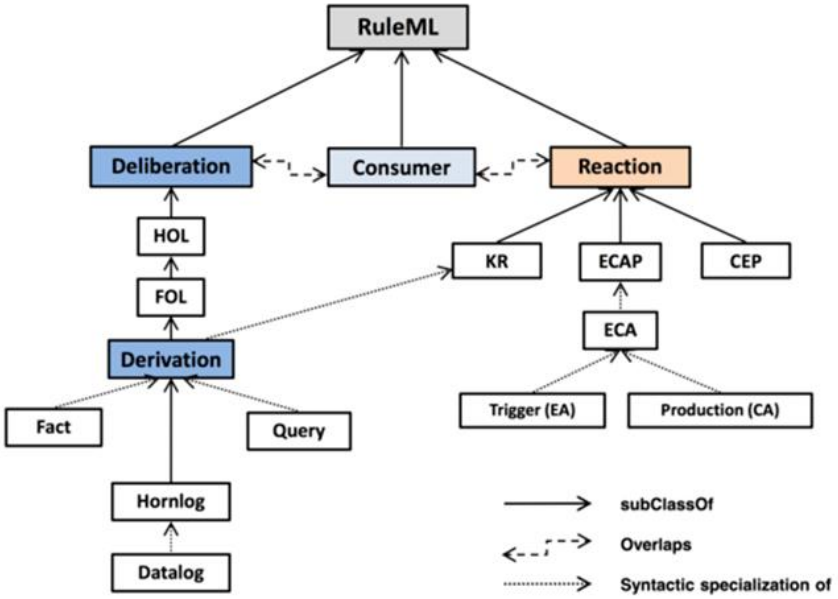

Figure 2.1. RuleML family of languages

Atomic formulae in Datalog RuleML are n-ary and each argument in an atomic formula can be positional or slotted (in OO Datalog). Datalog RuleML also supports (universal and existential) quantifiers, object identifiers, performatives (e.g. assert, retract), logical connectives and terms. Terms can be literal values, variables, individuals, etc. Hornlog RuleML expands Datalog with positional or slotted functional expressions as terms.

2.1.2 SWRL

Semantic Web Rule Language (SWRL) is defined as a combination of OWL DL and OWL Lite sublanguages of OWL with Unary/Binary Datalog RuleML. SWRL has both a human-readable syntax and an XML-based syntax. Rules in SWRL are in the form of an

antecedent-consequent pair. Antecedent (head) and consequent (body) of SWRL rules consist of a conjunction of one or more atoms. SWRL tends to be a reasoning layer built on top of OWL. All rules in SWRL are expressed in the form of OWL classes. SWRL provides seven types of atoms: Class atoms (owl:Class) e.g. Person(?p), individual property atoms (owl:ObjectProperty) e.g. hasBrother(?x, ?y), data valued property atoms (owl:DatatypeProperty) e.g. hasAge(?x, ?age), different individuals atoms e.g. differentFrom(?x, ?y), same individual atoms e.g. sameAs(?x, ?y), built-in atoms and data range atoms. SWRL's built-ins approach is based on the reuse of existing built-ins in XQuery and XPath. An SWRL's data range atom consists of a datatype name or a set of literals and a single argument representing a data value e.g. xsd:int(?x).

2.1.3 RIF

Rule Interchange Format (RIF) is a W3C standard to facilitate exchanging rule between rule systems. The idea of RIF is to provide mechanisms for rule systems to be able to map rules in their native language to RIF and back. This idea presents a standard method for rule translation and sharing. In order for rule interchange to be semantic-preserving, RIF defines dialects that support different categories of rules so two rule systems can talk trough a dialect which both support. RIF currently is focused on two types of dialects: logic-based dialects and dialects for rules with actions. RIF-BLD [24] is the RIF logic-based dialect and RIF-PRD [25] is a dialect for rules with actions. RIF-BLD and RIF-PRD share a common core dialect called RIF-Core [26]. RIF-DTB [23] (Datatypes and Built-in functions) is another dialect which as its name suggests comprising a set of data types and built-in functions. RIF also has RIF-FLD (Framework Logic Dialect) [27] that describes

mechanisms for specifying syntax and semantics of RIF logic-based dialects (RIF-Core and RIF-BLD). RIF-FLD also can be used as a template to design custom dialects.

2.1.4 Notation3

Notation3 language [8], also known as N3, is a human-readable serialization of RDF [28] models. Notation3 is much more compact and readable than XML-based RDF notation. It goes beyond RDF by supporting RDF-based rules. Notation3 extends RDF by bringing rule formulae, variables, implications, and predicates. A formula is a set of statements in N3. All statements in N3 are in the form of subject-predicate-object similar to nodes in RDF except that a node in N3 can also be a formula. Some of the elements of N3 are formula, variables, quantifiers, URIs, equality, datatypes, and list. Turtle [29] is a simplified RDF-only subset of N3.

2.1.5 POSL

Positional-Slotted Language (POSL) [7] is a Semantic Web rule language with a human-readable syntax that integrates Prolog's positional and F-logic slotted syntax for representing facts and rules. Positional notation is used for presenting ordered set of objects. The slotted notation is used for presenting unordered set of attribute-value pairs. Clauses (of rules and facts) in POSL are made of atoms which are relations with an optional list of parameters. List of parameters can be a combination of positional and slotted parameters. Positional parameters are separated by comma ',' and slotted parameters are separated by semicolon ';'. Positional arguments have a higher precedence than slotted ones. POSL supports variables, list, and Webizing (all language elements in POSL can be named using URIs in the style of N3 notation).

Table 2.1 shows some of the features of above-mentioned rule markup languages. Note that the features of these languages are not limited to those in Table 2.1. For a view of complete features of each of these languages, refer to their appropriate references mentioned in this chapter.

Table 2.1. Feature comparison of Web rule languages

Rule language	Logical Operators			Quantifiers (Universal, Existential)	Datatype Support	Rule Set/ Rule Name	Production Rules Version (Assert, Retract, Modify)	Syntax		Built-in Functions (Math., String, Boolean, Time, List URI)
	Conj.	Disj.	Neg.					Pres.	Serial.	
RuleML	✓	✓	✓	✓	✓	✓	✓	×	✓	✓
SWRL	✓	×	×	U	✓	RN	A	✓	✓	✓
POSL	✓	×	×	×	URIs to XML Schema Datatypes	RN	×	✓	×	×
Notation3	✓	×	×	✓	✓	×	A	✓	×	✓
RIF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

There are other rule markup languages and Semantic Web rule languages in the literature such as R2ML [15], Jena Rules [30], N-Triples [31], etc. For a comprehensive list of rule languages refer to [3], [23].

2.2 BRE and BRMS

A Business Rules Engine (BRE) is a software system that executes one or more business rules in a runtime environment. Business rule engines can provide the ability to define, validate, manage, deploy and infer business rules. There exists different types of rule engines: Forward chaining, backward chaining and hybrid chaining (which switches between forward and backward chaining). Forward chaining rule engines can be further

divided into inference rules and reaction or Event Condition Action (ECA) rules. In backward chaining, rule engines seek to resolve the facts to fit a particular goal (e.g. query answering). Table 2.2 compares a few number of business rule engines and their features.

Table 2.2. Comparison of Business Rules Engines

Name	Summary	JSR 94	RETE algorithm	Forward/ Backward chaining	Rule Language/ Platform	License	URL
Jess	Jess is a rule engine for the Java platform	Yes	Yes	Yes	Java	Academic /Comm.	http://www.jessrules.com/
Jena	Jena is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs.	No	Yes	Both	Java	Apache License 2.0	http://jena.apache.org
SweetRules	SweetRules is an integrated toolkit for semantic web rules, revolving around the RuleML	NO	NO	Both	Java	GNU LGPL	http://sweetrules.projects.semwebcentral.org/
JRuleEngine	a very simple open-source JSR-94 compliant rule engine written in Java	Yes	No	Forward	Java	LGPL	http://jruleengine.sourceforge.net/
Drools Expert	Drools Expert is a declarative, rule-based, coding environment	Yes	Yes	Both	Java	ASL	http://drools.jboss.org/drools-expert.htm

For a comprehensive list of rule engines, refer to [32]. A Business Rule Management System (BRMS) enables business policies and operational decisions associated with these policies to be defined, deployed and maintained separately from application code. BRMS systems provide features to maintain business rules by business users. Some of these features are as follows:

- Rule Repository: is a rule base to store business rules.
- Rule Editor: is (usually) a GUI to define, edit, and maintain business rules.
- Rule Engine: is the core component of a BRMS that enforces the rules.
- Vocabulary: Defines business terms and domain terminologies.

Other features of a BRMS can include rule visualization, decision trees, decision table, rule versioning, etc. Table 2.3 shows some of the popular BRMS tools and their features.

Table 2.3. Comparison of BRMS systems

Name	Comments	Components	Developer	License	URL
IBM ODM (JRules BRMS family +WebSphere Business Events)	IBM ODM provides an integrated business rules and events management platform to improve the quality of transaction and process-related decisions	Decision Center, Decision Server, Decision Server rules (Rule Designer), Decision Server Events (Event designer)	IBM	Comm.	http://www-03.ibm.com/software/products/en/odm
Drools	Drools is a BRMS with an inference based rules engine, using an enhanced implementation of the Rete algorithm.	Drools Guvnor (Business Rules Manager) Drools Expert (rule engine) jBPM (process/workflow) Drools Fusion (event processing/temporal reasoning) OptaPlanner (automated planning)	Red Hat	ASL	http://jboss.org/drools/
Corticon	Progress Corticon delivers automated business decisions. It separates decisions from processes, helping both business and IT users quickly create or reuse business rules	Business Rules Studio, Business Rules Server, Enterprise Data Connector	Corticon Technologies, Inc	Comm.	http://www.progress.com/products/corticon/
InRule	InRule is a .NET solution for authoring, managing, and executing business rules	irAuthor, irWord, irStudio, irVerify, irCatalog, irServer, irSOA, irSDK	InRule Technology	Comm.	www.inrule.com

2.3 ANTLR

ANother Tool for Language Recognition (ANTRL) [22] is a language recognizer and parser generator. ANTLR can read and process any structured text. Language implementers can use ANTLR to develop their domain-specific language (DSL). In general, ANTLR takes as input a grammar and generates a recognizer for the language defined by that grammar. The ANTLR grammar notation is similar to EBNF notation. Recognizer can be a lexer and/or a parser. A lexer is also called lexical analyzer, tokenizer or scanner. Although ANTLR itself is written in Java, it can output the required recognizer (lexer, parser) in Java, C#, Python, etc. ANTLR is a parser generator that uses LL(*) for parsing. An LL parser is a top-down parser that parses the input from left to right, performing leftmost derivation of the sentence. If we always expand the leftmost variable first, we would have a leftmost derivation. An LL parser is called an LL(k) parser if it uses k tokens of lookahead when parsing a sentence. In addition to lexer and parser, ANTLR can be used to generate a parse tree as well. A grammar in ANTLR 4 (latest version of ANTLR to date) has the general form shown in Table 2.4 [22].

Grammar name follows common identifier definition rules (e.g. in programming languages). The file name that contains the ANTLR grammar 'x' must be called 'x.g4'. The file name and grammar name must be the same. '.g4' is the extension of ANTLR grammar's file. /**…*/ and // are used for multiline and single line comments. Options, imports, token specifications and action names can be defined in any order and at most one of each can be defined.

Table 2.4. ANTLR grammar structure

```
/** Optional javadoc style comment */
grammar Name;
options {...}
import ... ;
tokens {...}
channels {...} // lexer only
@actionName {...}

rule1 // parser and lexer rules, possibly intermingled
...
ruleN
```

'import' imports other grammars. In other words, 'import' lets a sizable grammar to be broken into smaller logical and reusable chunks. 'tokens' is used to define token types needed by a grammar that there are no associated lexical rules defined for them in the grammar. Actions are used to inject code into the lexer or parser generated classes. 'rule1…ruleN' are the actual lexer and parser rules [22]. All these elements in an ANTLR grammar (Table 2.4) are optional except the grammar name and at least one rule. Rules have the following form:

```
ruleName : alternative1 | ... | alternativeN ;
```

Parser rule names must start with a lowercase letter and lexer rule name must start with a capital letter. Rules and rule alternatives can contain the native code of the target language (e.g. Java, Python, etc.). These codes will be executed when the parser visits that rule or alternative. Grammars can have a 'parser' or a 'lexer' prefix preceding the 'grammar' header keyword that defines a pure parser or lexer grammar respectively. Grammars

without the prefix are combined grammars and can contain both lexical and parser rules. For example, to make a grammar that only allows lexical rules use the following header:

```
lexer grammar myGrammar;
```

Only lexer grammars can contain custom channel specifications.

```
channels { WHITESPACE_CHANNEL, COMMENTS_CHANNEL }
```

Channels are used to keep white spaces and comments from the input stream. Channels can be used like enums in lexer rules. For example, the following lexer rule declares that blank, newline feed, and tab whitespaces should be kept in whitespace channel.

```
WS : [ \r\t\n]+ -> channel(WHITESPACE_CHANNEL) ;
```

For more information about ANTLR refer to [22]. To sum up, this chapter described rule markup languages including RuleML, SWRL, RIF, Notation3, and POSL. Then the section followed by a review of business rules engines and business rule management systems. Since ANLTR is used as a library to develop parsers for the rule languages in this dissertation, last part of this section briefly described ANTLR and its specifications.

Chapter 3

Related Work

In order for the Semantic Web to be successful and computer systems to be able to interpret data on the Semantic Web, these data or information must be encoded in a structured format (such as XML or a human-friendly format) that make rule interchange and sharing possible [33]. In addition, sets of rule inference engines must be constructed to allow automated reasoning over the Semantic Web rules. The need to reason about semantic data has created an increasing interest in rule markup languages and rule inference engines for these markup languages within the Semantic Web community [34]. This chapter describes related works to rule exchange/translation among rule markup languages. Moreover, it explains some of the popular rule inference engines that are reasoning engines for rule markup languages.

OO jDREW [21], or the Object-Oriented Java Deductive Reasoning Engine for the Web is the reference implementation of the (Naf Hornlog) RuleML Web language. It is an Object-Oriented extension to jDREW. As a rule translator, OO jDREW can translate rules written in POSL language to RuleML. As a reasoning engine, OO jDREW is an engine for the RuleML and POSL [35] languages. OO jDREW is written in Java and it includes two modes: bottom-up (forward chaining of rules) and top-down (backward chaining of rules). OO jDREW supports positional Prolog style logic, object-oriented features from RuleML, slots, order-sorted types and oids [21].

SPARQL Inferencing Notation (SPIN) [36] is a SPARQL-based [37] rule and constraint language for the Semantic Web. SPIN provides capabilities that allow users to define their own SPARQL functions and query templates. SPIN utilizes SPARQL to facilitate the

definition of constraints and inference rules in ontologies. When applying the constraints on the ontology, SPIN can determine all the problematic data elements. By using SPARQL query template in SPIN, it is possible to define generic queries with high reusability [38].

Shapes Constraint Language (SHACL) [39] is a language for validating RDF graphs against a set of conditions. These conditions are provided as shapes and other constructs expressed in the form of an RDF graph. RDF graphs that are used in this manner are called shapes graphs in SHACL and the RDF graphs that are validated against a shapes graph are called data graphs [39]. SHACL includes features to express conditions that constrain the values that a property may have such as numeric ranges, string patterns and others.

DR-DEVICE [40] is a system for defeasible reasoning on the Web. Defeasible reasoning is a rule based approach for efficient reasoning with incomplete and inconsistent information. Such reasoning is useful, among others, for ontology integration where conflicting information arises naturally; and for the modeling of business rules and policies where rules with exceptions are often used [40]. DR-DEVICE is capable of reasoning RDF metadata over multiple Web sources using defeasible logic rules. This system is implemented on top of CLIPS production rule system and builds upon R-DEVICE, an earlier deductive rule system over RDF metadata.

DR-Prolog [41] is a system for defeasible reasoning with rules and ontologies on the Semantic Web. This system is a declarative system supporting 1) rules, facts and ontologies 2) Semantic Web standards such as RDFS, OWL and RuleML, and 3) monotonic and nonmonotonic rules, open and closed world assumption, and reasoning with inconsistencies. DR-Prolog is syntactically compatible with RuleML and is based on Prolog.

SweetJess [42] and SweetProlog [43] are two tools that translate rules into Jess and Prolog respectively. SweetJess translates rules in Situated Courteous Logic Programs (SCLP) knowledge representation, which are syntactically encoded in RuleML, into Jess rules. However, SweetJess can only reason over ontologies expressed in DAMLRuleML (a DAML+OIL like syntax for RuleML). SweetProlog translates OWL ontologies and rules expressed in OWLRuleML into a set of facts and rules in Prolog.

ROWL [44] is a system that enables users to express rules in RDF/XML syntax using an OWL ontology. It transforms rules in RDF/XML to rules in Jess using XSLT stylesheets. XSLT stylesheets in ROWL are used to transform OWL ontologies, instances, facts and rules into the CLIPS format. Then the facts and rules in CLIPS format are fed to the Jess engine that enables rule reasoning over them.

RIF2Jess [45] translates rules in RIF into Jess rules. It considers that RIF rules can be inferred by common Semantic Web rule engines such as Jess, Pychonko, and Euler. RIF2Jess purpose is to reason over RIF rules using Jess rule engine [46]. Jess is a rule engine based on RETE algorithm and is developed in Java. To achieve its goal, RIF2Jess separately translates RIF facts and rules in a RIF document, to Jess facts and rules. Then it asks Jess to reason over those rules and facts. RIF2Jess does not make a translation back. In other words, the inferred knowledge from the Jess engine will not be translated back to RIF.

Euler Yet another proof Engine (EYE) [47] is an inference engine supporting logic based proofs. It is available in various languages such as Java, Python, C#, JavaScript and Prolog. EYE is a backward chaining reasoning engine enhanced with Euler path detection mechanism and determines whether a given set of facts and rules supports a given

conclusion. Internally, EYE translates its supported rule language, Notation3 (N3) [8], to Prolog Coherent Logic intermediate code and runs it on YAP (Yet Another Prolog) [48] engine. EYE can provide useful information while reasoning, for example, proof explanation, debugging logs and warning logs [23].

Apache Jena [49] is an open source Java framework for building Semantic Web applications. The framework is composed of different APIs including an inference subsystem interacting together to process RDF data. The Jena inference subsystem is designed to allow a various range of reasoners to be added into Jena. Jena inference subsystem includes a transitive reasoner supporting simple taxonomy traversal, RDFS rule reasoner, OWL reasoner and a generic rule reasoner that supports user-defined rules written in the Jena's own format. Internally, there are two rule engines in Jena, a forward chaining RETE [50] engine, and a tabled backward Datalog engine. Built-in functions are also provided in Jena inference subsystem and can be extended by the user [23].

FuXi [51] is a forward and backward reasoning engine for the Semantic Web and Python. FuXi works as a companion to RDFLib, a Python library for working with RDF. FuXi uses RDF for its core knowledge representation of facts in a particular domain. The FuXi inference engine is developed based on RETE algorithm for pattern or object match problem. FuXi includes the following modules: FuXi.Horn is an API for managing an abstract logic programming syntax. Both FuXi.DLP and FuXi.RETE modules use this module for respectively creating the rulesets converted from OWL RDF expressions and creating a Horn ruleset from a parsed Notation 3 graph. FuXi.Syntax that uses InfixOWL library [52] for parsing and serializing ontologies into memory. FuXi.SPARQL module is the implementation for a backward chaining store and finally FuXi.LP module that is a

query answering method for recursive databases based on a meta-interpretation method called Backward Fixpoint Procedure. FuXi currently supports a limited subset of Notation3. It also supports negation, addition, and removal of rules. Backward chaining reasoning is also supported using SPARQL endpoints [23].

Jess (Java Expert System Shell) [46] is a rule engine and scripting language for the Java platform. Jess has the capacity to reason using the knowledge supplied in the form of declaration rules. Jess is based on the RETE algorithm and supports both forward and backward chaining. Rules in Jess can be written in two formats: Jess script language or XML. A program written in Jess may consist of facts, rules and (Java) objects. Jess was originally conceived as a Java clone of CLIPS, but nowadays it has many features that differentiate it from CLIPS.

This chapter described the related works in the field of rule interchange between rule languages such as RuleML, POSL, RIF, etc. It also had a review of some common rule engines developed for rule reasoning. However, there has not been any research done for rule interchange between N3, POSL, SWRL, Datalog RuleML and RIF in the literature. The goal of this dissertation is to facilitate rule interchange between the above rule languages. Also, there is no rule engine for the RIF language in the literature. As another goal of this dissertation, it introduces a rule engine for RIF. The methods, algorithms, prototypes introduced in this thesis for rule interchange between N3, POSL, SWRL, Datalog RuleML, and RIF as well for the RIF rule engine can be used as a start point by other researchers to work in rule interchange between these languages.

Chapter 4

The Framework

This chapter describes the proposed framework. Figure 4.1 shows this framework.

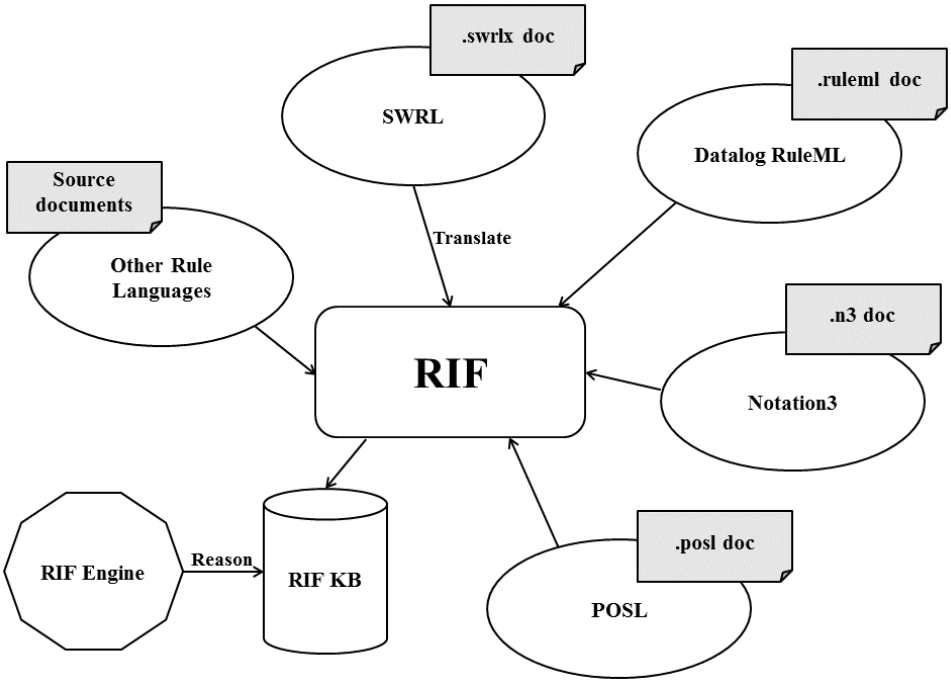

Figure 4.1. The proposed framework

The main objective of this framework which is rule integration between various rule languages is shown in Figure 4.1. Ovals show rule languages (i.e. source documents in rule languages). RIF language acts as an intermediary language. Rules in SWRL, Datalog RuleML, POSL, and Notation3 (N3) are transformed into RIF-BLD. Other rule languages can be added to the framework as well. Any rule language needs its own translator to transform its rules to RIF-BLD. Rules translated from N3, POSL, SWRL, and Datalog RuleML to RIF-BLD are stored in the RIF-BLD knowledge base. RIF-BLD engine in the framework is both forward chaining and backward chaining inference engine. It can answer

queries as well as reason over RIF-BLD rules and infer new knowledge. Figure 4.2 shows the framework in details. The framework has three major parts: the source rule languages (including their rule parser, visualizer, and translator), RIF language (including its knowledge base, rule parser, visualizer, and translator), and finally the RIF rule engine.

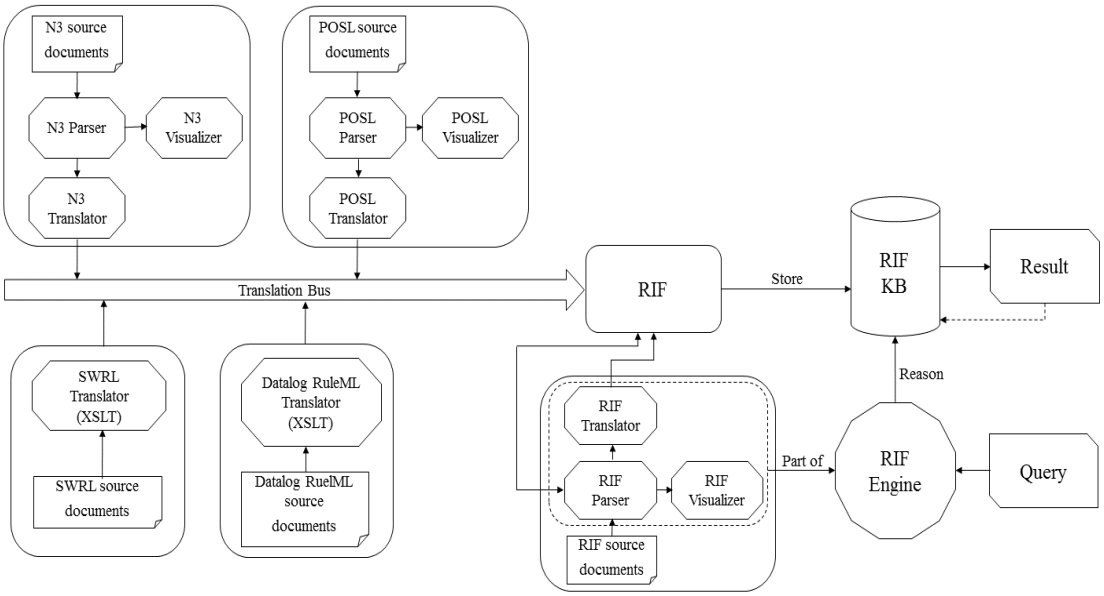

Figure 4.2. The proposed framework (in details)

N3 and POSL are human-readable rule languages. Therefore, to be able to translate their rules to RIF rules, an N3/POSL rule parser would be necessary. The parser preprocesses the input rules. It decomposes different parts of a rule or fact which are used later by the translator. The translator then translates rules and facts to their equivalents in RIF. SWRL and Datalog RuleML are XML-based rule languages. SWRL has a human-readable syntax as well. SWRL and Datalog RuleML rules are in XML format and they can be parsed with XML parsers. Also, since SWRL and Datalog RuleML are XML-based rule languages, their translators to RIF are XSL stylesheets. Translated rules (from N3, POSL, SWRL and

Datalog RuleML) are stored in the RIF knowledge base. N3 and POSL translated rules are stored in RIF abstract syntax and SWRL and Datalog RuleML translated rules are stored in RIF XML format.

RIF part of the framework contains RIF knowledge base, rule parser, visualizer, translator, and finally the RIF rule engine. RIF parser has three responsibilities: One, it can double check the translation result from other languages (e.g. N3 and POSL) to make sure there are no syntactical errors in the result. Two, it preprocesses RIF rules in human-readable syntax before translating them to RIF XML syntax by the RIF translator. Three, it is a part of the rule engine to parse RIF rules.

RIF translator translates the RIF rules in abstract syntax (human-readable) to presentation syntax (XML-based). The rule engine reasons over the RIF knowledge base. It also can answer queries in RIF. The inference result can be added back to the knowledge base. In the following the different parts of the framework are described in more details to make the concepts clearer. The rule visualizer in N3, POSL, and RIF can show the input rules in a tree format after being parsed by their parser. Different components of the framework are described as follows.

- N3/POSL: The N3/POSL rule parser reads the N3/POSL source document and then parses the rules and facts in the source document. N3/POSL visualizer generates a parse tree after the parser parses the input. The parser output as a parse tree is fed to the N3/POSL translator. The translator then traverses the parse tree and translates elements in the tree (nodes, edges, subtrees, etc.) to their equivalent elements in RIF. The N3/POSL translator result is RIF rules equivalent to N3/POSL rules from the source documents. The result is stored in the RIF KB.

- SWRL/Datalog RuleML: The SWRL/Datalog RuleML translator transforms the source documents (in XML format) to RIF XML. The translator in an XSLT stylesheet. For every rule/fact element in the SWRL/Datalog RuleML XML document a template should be defined to transform that element to its equivalent element in RIF-XML.
- RIF: The RIF parser reads the RIF-BLD source document and parses it. The RIF-BLD visualizer then generates a parse tree after the parser parses the document. The RIF-BLD translator takes the generated parse tree as input and translates elements in the tree (nodes, edges, subtrees, etc.) to their equivalent elements in RIF-XML. The parser can also parse the translated rules from N3 and POSL to make sure the translation results are syntactically correct. The RIF-BLD parser is a part of the RIF rule engine as well.
- RIF engine: The RIF-BLD engine works in two modes: query answering and inferring new knowledge. To infer new knowledge, the RIF-BLD parser first parses the knowledge base (e.g. the input RIF document). Then the rule engine uses backtracking techniques to infer new knowledge. The result (new knowledge) can be added to the knowledge base. The query answering part of the rule engine uses unification to match the variables in the query to the data in the knowledge base.

To show how the framework works, we use the following example. Consider, a knowledge base in N3 is fed to the framework. First, in the framework, the KB is parsed to find syntactical errors (if any) using the N3 parser. N3 KB has a human-readable syntax. If there are no syntactical errors, a Concrete Syntax Tree (CST) is built from the KB after being successfully parsed. The CST will then be fed to the N3 to RIF

translator. The translator converts the CST to a RIF-BLD knowledge base. RIF-BLD parser can now parse the translation result to make sure there no syntactical errors in it. If desired, the RIF-BLD forward reasoner can reason over the newly generated KB. This chapter described the proposed framework and its components. This framework is not limited to the current languages and other rule languages can be plugged into it. A software implementation of the framework is developed in this dissertation to make concepts clearer as well as to be available as a free rule interchange tool. The next chapters describe each component of the proposed framework in details.

Chapter 5

Notation 3 Translation

This chapter explains the N3 component of the framework proposed in this thesis. The contributions to the N3 language include an N3 grammar in ANTLR4 notation [22], a rule parser, rule visualizer and a rule translator. They are all part of the N3 component of the framework. Practical examples of using the N3 component are shown further in this chapter. The N3 translator is responsible for translating N3 rules into RIF rules. N3 parser parses the N3 input rules. If the input rules are syntactically correct, then they will be translated to RIF rules. This component can be used as a standalone N3 parser by business analysts and rule implementers whom their rules bases are in N3. They can write, edit and parse N3 rules using the parser. The rule visualizer can show N3 rules in a graphical tree format. Briefly speaking, the N3 component of the framework reads N3 rules from the input, parses them, shows them in a tree view and finally translates them to RIF rules.

5.1 Notation3 Parser

The N3 grammar is available in BNF notation in [53]. Appendix A shows this grammar. As the first step in N3 rule interchange, its grammar in [53] is sorted in a top-down manner (compare to randomly sort in [53]) then it is converted to ANTLR4 notation. N3 grammar in ANLTR4 notation is also available in Appendix A. ANTLR grammar notation is more complicated than EBNF notation (e.g., it can contain built-in action codes of the target language like Java).

Production rules in ANLTR grammar are divided into two categories: parser rules and lexer rules. Lexer rules specify token definitions. In other words, lexer rules show how input stream (of characters) can split up to a list of tokens. Parser rules are defined in a similar manner to lexer rules. The input to parser rules is token streams obtained from the lexer. In the following, the syntax to define a parser or lexer rule in ANTLR (as in is shown in section 2.3) is presented. A semicolon terminates rule definition:

```
ruleName : alternative1 | ... | alternativeN ;
```

In ANLTR, rule priority is top-down and for alternatives to a rule is from left to right. Rule priority is important especially in lexer rules when a part of input stream can be matched to more than one rule. In this case, priority is with the rule that is defined first. Consider the following example:

```
URI : PRTCL ‘://’  ;
PRTCL : [a-z] ;
```

Lexical rule ‘URI’ contains rule ‘PRTCL’. Therefore, rule ‘URI’ should be defined before rule ‘PRTCL’. In this case, if for example ‘http://’ is a part of the input stream, then it should be matched to rule ‘URI’ not rule ‘PRTCL’. However, if rule ‘PRTCL’ was defined before rule ‘URI’ then based on rule priority in ANTLR ‘http’ part of the input would match rule ‘PRTCL’ and the rest of input which is ‘://’ would cause a syntax error. For a more realistic example see ‘PREFIX’ and ‘BARENAME’ rule definitions of N3 grammar (in ANTLR notation) in Appendix A.

5.2 Notation3 Translator

The framework's implementation in this dissertation can show a graphical view of the N3 grammar in ANTLR notation. As an example, Figure 5.1 shows the production rule of 'n3_node' of N3 grammar in a graphical mode.

```
n3_node : n3_symbol | '{' n3_formulacontent '}' | N3_VARIABLE |
N3_NUMERICLITERAL | n3_literal | '[' n3_propertylist ']' |
'(' n3_pathlist ')' | '@this' ;
```

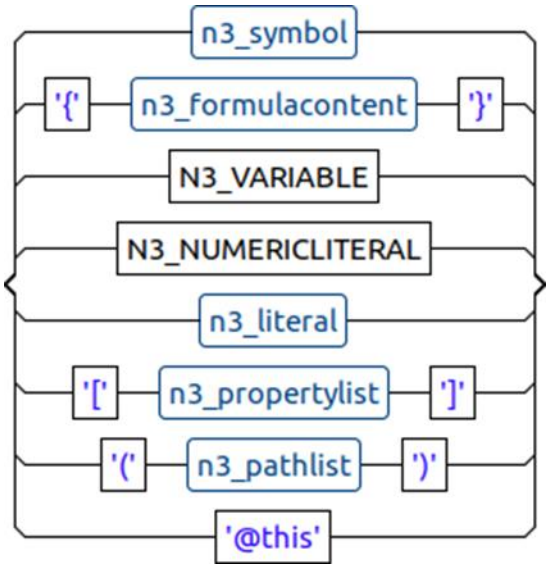

Figure 5.1. Graphical view of a Notation3 production rule

Where 'n3_node' is a name of a rule in the N3 grammar. Parallel lines mean alternatives. Names that start with a lowercase letter are parser rule identifiers and names that start with capital letters are lexer rule identifiers. Quoted values are literals. For a complete view of N3 grammar refer to [53] and Appendix A.

The examples 1 to 3 in Appendix A are used below to show how N3 parser and translator parts of the framework work. Example 2 and 3 are borrowed from [54].

Table 5.1. An example rule base in N3

```
#  Example 1
@base <http://www.example.org/> .
@prefix : <http://www.example.org/People#> .
:Joe :father :Fred.
:Fred :brother :Bob, :Rob.
:Alex :brother :Jack, John ; :sister :Sarah.
```

In Table 5.1, '#' sign indicates single line comment in N3 language, the '@base' directive sets the base URI to be used for the parsing of relative URIs and '@prefix' directive binds a prefix to a namespace URI. The namespace prefix may be empty, in which case the prefix qname starts with a colon. In this case, N3 to RIF translator uses 'EMPTY' as qname while translating N3 rules to RIF. Comma ',' shows the repetition of another object for the same subject and predicate. Semicolon ';' indicates the repetition of another predicate for the same subject. Facts in example 1 above say that:

```
Joe has father Fred.
Fred has brothers Bob and Rob.
Alex has brother Jack and John.
Alex has sister Sarah.
```

Figure 5.2 shows a part of the output tree of the N3 parser after parsing rules in example 1. For space preserving purposes, the whole tree is not shown here. The parser output is a Concrete Syntax Tree (CST). The CST shows that there is no syntactical error in the input rules. Otherwise, the CST tree will not be built successfully.

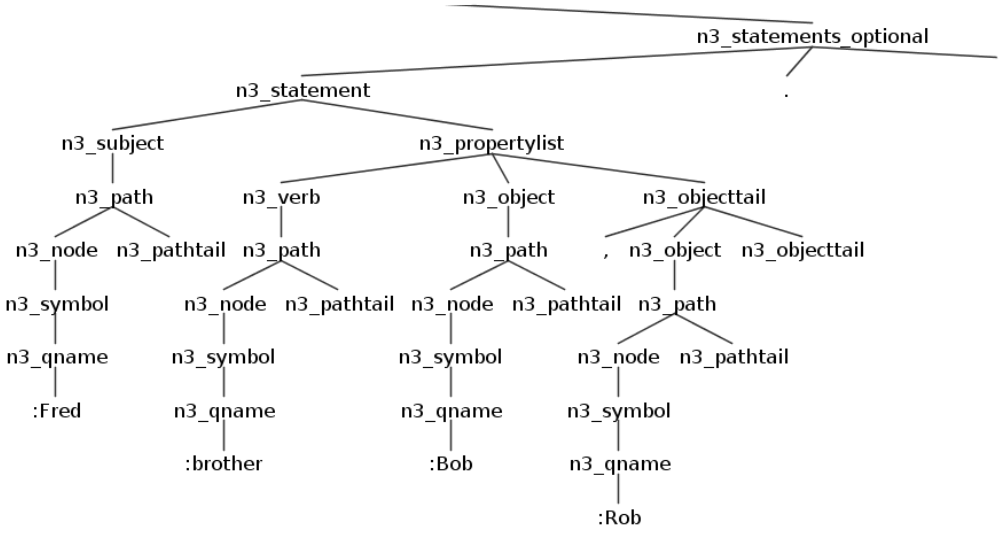

Figure 5.2. Concrete Syntax Tree of the rules in example 1

Table 5.2 shows the N3 translation result after parsing and translating N3 rules from example 1 to RIF rules.

Table 5.2. Translation result of N3 rules in example 1 to RIF rules

```
Document(
  Base(<http://www.example.org/>)
  Prefix(EMPTY  <http://www.example.org/People#>)
  Group(
    EMPTY:father(EMPTY:Joe EMPTY:Fred)
    EMPTY:brother(EMPTY:Fred EMPTY:Bob)
    EMPTY:brother(EMPTY:Fred EMPTY:Rob)
    EMPTY:brother(EMPTY:Alex EMPTY:Jack)
    EMPTY:brother(EMPTY:Alex John)
    EMPTY:sister(EMPTY:Alex EMPTY:Sarah)
  )
)
```

Table 5.3 shows another example of N3 rules (example 2 in Appendix A). Rules in this example are parsed and translated to RIF rules (using the N3 component of the framework).

Table 5.3. An example N3 rule base

```
#Example 2
# Processed by Id: cwm.py,v 1.29 2001/02/28 20:45:26 timbl Exp
# using base file:/afs/w3.org/pub/WWW/2000/10/swap/test/meet/
 # Notation3 generation by
# notation3.py,v 1.54 2001/02/12 18:38:31 timbl Exp
@base <http://www.example.org/>.
@prefix g: <http://www.another.example.org/geographical#> .
@prefix : <http://www.example.org/meeting_organization#> .
@prefix p: <http://www.example.org/personal_details#> .
 <http://meetings.example.com/cal#m1>
     :Location  [
           g:zip "02139";
           g:lat "14.124425";
           g:long "14.245" ];
      :chair <http://www.example.org/people#fred>;
      :homePage <http://meetings.example.com/m1/hp> .
 <http://meetings.example.com/m1/hp>     :policy
 <http://meetings.example.com/privacyPolicy> .
 <http://www.example.org/people#fred>     :attending
       <http://meetings.example.com/cal#m1>;
        p:GivenName "Fred";
        p:hasEmail <mailto:fred@example.com> .
 #ENDS
```

In N3, blank nodes are shown using '[' and ']' (see the location predicate in Table 5.3). The CST tree of the N3 rules in example 2 is not shown here due to preserving space. Table 5.4 shows the translation result of N3 rules (in Table 5.3) to their equivalent rules in RIF.

Table 5.4. Translation result of N3 rules in example 2 to RIF rules

```
Document(
  Base(<http://www.example.org/>)
  Prefix(g <http://www.another.example.org/geographical#>)
  Prefix(EMPTY  <http://www.example.org/meeting_organization#>)
  Prefix(p <http://www.example.org/personal_details#>)
 Group(
    EMPTY:Location(<http://meetings.example.com/cal#m1> ?x1)
    EMPTY:chair(<http://meetings.example.com/cal#m1> <http://www.example.org/people#fred>)
    EMPTY:homePage(<http://meetings.example.com/cal#m1> <http://meetings.example.com/m1/hp>)
    g:zip(?x1 "02139")
    g:lat(?x1 "14.124425")
    g:long(?x1 "14.245")
    EMPTY:policy(<http://meetings.example.com/m1/hp> <http://meetings.example.com/privacyPolicy>)
    EMPTY:attending(<http://www.example.org/people#fred> <http://meetings.example.com/cal#m1>)
    p:GivenName(<http://www.example.org/people#fred> "Fred")
    p:hasEmail(<http://www.example.org/people#fred> <mailto:fred@example.com>)
))
```

Empty prefix in N3 ':' is mapped to 'EMPTY' prefix in RIF. Translating blank nodes in N3 to RIF is a bit tricky. For each blank node, a new variable is defined to represent that blank node in the rule containing the node. Then each property in the blank node makes a

new rule in the target RIF using the defined variable. For example, see the following part of the first rule in Table 5.3.

```
<http://meetings.example.com/cal#m1>     :Location  [
        g:zip "02139";
        g:lat "14.124425";
        g:long "14.245" ];
```

The above rule is translated to the following rules in RIF using the N3 to RIF translator:

```
EMPTY:Location(<http://meetings.example.com/cal#m1> ?x1)
g:zip(?x1 "02139")
g:lat(?x1 "14.124425")
g:long(?x1 "14.245")
```

As mentioned earlier, N3 rule bases can have universal/existential rules as well. The example in Table 5.5 shows an existential N3 rule. The rule says that “if who1 has a father whose brother is who2, then who2 is who1‘s uncle”.

Table 5.5. An example N3 rule base

```
#Example 3
@prefix : <http://www.example.org/meeting_organization#> .
@forSome :who1, :who2.
{ :who1 :father [ :brother :who2 ] } => { :who1 :uncle :who2 }.
```

Table 5.6 shows the translation results of the N3 rule in Table 5.5 to RIF rules. Note the variable ‘?x1’ and conjunction symbol ‘And’ in Table 5.6.

Table 5.6. Translation result of N3 rules in example 3 to RIF

```
Document(
  Prefix(EMPTY  <http://www.example.org/meeting_organization#>)
  Group(
    Exists :who1 :who2  ?x1 (
      And(
        EMPTY:father(EMPTY:who1 ?x1)
        EMPTY:brother(?x1 EMPTY:who2)
      )
      :-   EMPTY:uncle(EMPTY:who1 EMPTY:who2)
)))
```

Some of the features of the N3 component of the framework are as follows:

- Detection of syntactical errors of N3 rules if any (using the parser).
- Creation of concrete syntax tree of N3 rules.
- Translating N3 directives, universal/existential rules, equality, formula, conjunctions in rule head or body, literals, language suffixes, functions, numerical values, etc. to their equivalents in RIF language.

This chapter introduced a parser and translator for N3 rules to RIF rules. Parser allows writing, editing and verifying N3 rules directly in the framework's implemented demo without a need for an external N3 tool. The parser can also create a concrete syntax tree view of the input N3 rules. The N3 translator converts N3 rules to RIF rules which is a valuable step in rule exchange, transport and sharing especially using RIF language as the target language. To verify the translation of N3 rules to RIF, a RIF parser (as another part

of the framework) can be used. RIF parser determines if the translated rules to RIF are syntactically correct. Details of the RIF parser is described in its own chapter.

Chapter 6

POSL Translation

This chapter explains the POSL component of the framework (proposed in this thesis). The contributions to the POSL language include a POSL grammar in ANTLR4 notation [22], a POSL rule parser, rule visualizer and a rule translator. They are all part of the POSL component of the framework. Practical examples of using the POSL component are shown further in this chapter.

6.1 POSL Parser

The POSL parser parses the POSL input rules. If the input rules are syntactically correct, then they will be translated to RIF rules. The POSL translator is responsible for translating POSL rules into RIF rules. The POSL parser can be used as a standalone tool by business analysts and rule implementers whom their rules bases are in POSL. The rule visualizer can show POSL rules in a graphical tree format. In a nutshell, the POSL component of the framework's implementation reads POSL rules from the input, parses them, shows them in a tree view and finally translates them into RIF rules.

Appendix B shows EBNF grammar of POSL language (from [35]). Note that it can be seen from the Appendix that the grammar is missing the definition of some of the production rules such as 'rel', 'role', 'type', and 'symbol'. A complete POSL grammar with ANTLR4 notation is developed based on the EBNF grammar from [35] and an outdated POSL grammar for older versions of ANTLR from [55]. To see this grammar, refer to Appendix B. Moreover, to see a graphical view of POSL grammar in ANTLR notation in HTML

format, refer to the framework's demo or its documentation. In the following the syntax of a production rule definition in ANTLR4 is shown:

```
ruleName : alternative1 | ... | alternativeN ;
```

A POSL rule is made of zero or more clauses. A clause can be a fact or a rule.

```
clause: atom (':-' atoms)? '.' ;
```

Figure 6.1 shows a graph view of the POSL clause definition above:

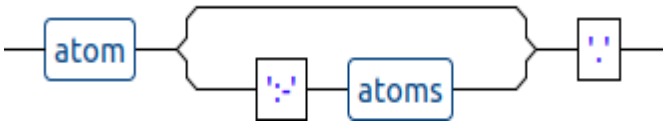

Figure 6.1. Definition of a POSL clause

Where 'clause' indicates the identifier of the production rule, quoted values are literals and semicolon indicates the end of the rule definition. 'atom' and 'atoms' are also rule name identifiers which their definitions are not shown here.

Rules and facts in POSL are made of atoms (as it can partly be seen from Figure 6.1) which are relations with a list of arguments. Atom arguments can be positional, slotted or a combination of positional and slotted arguments. A positional or slotted argument is made of terms which can be individuals, variables, complex terms (Cterm), Plex (a special case of complex terms) and Skolems. Originally, complex terms are what Prolog calls structures[7]. Cterm uses brackets '[…]' for arguments grouping. Cterm and Plex can have positional, slotted or positional/slotted arguments. For example, stakeholder[PeterMiller,SpeedShip] is a positional Cterm which describes a pair of

stakeholders. Similarly, a Plex (a constructor-less Cterm) version of the above Cterm is the Prolog-like list [PeterMiller,SpeedShip]. Table 6.1 shows different types of atoms (regarding their argument types) in POSL. Examples are borrowed from [35]:

Table 6.1. Atom arguments in POSL

Argument type	Description	Example
Positional	Ordered sequences of objects.	shipment(PC,47.5,BostonMoS,LondonSciM).
Slotted	Unordered sets of attribute-value pairs.	shipment(cargo->PC;price->47.5;source-> BostonMoS;dest->LondonSciM).
Positional + slotted	Combination of positional and slotted.	shipment(PC,47.5;source->BostonMoS;dest-> LondonSciM).
Positional + slotted Cterm	Positional/slotted complex term.	stakeholder[PeterMiller; Shipper-> SpeedShip]
Positional + slotted Plex	Positional/slotted constructor-less complex term (Plex)	[PeterMiller; Shipper-> SpeedShip]

In POSL a rule head is a single atom and a rule body can be a conjunction of atoms (comma-separated list of atoms). A fact is a single atom. The POSL component of the framework contains a POSL rule parser, visualizer, and a translator. POSL parser can parse and validated POSL rules. It also can be used as a pre-step of POSL translation to RIF to make sure the input POSL rules are syntactically correct before translating them to RIF rules. The POSL parser can be used as a standalone POSL rule editor as well. The following POSL rule example "example1.posl" (also available in Appendix B) shows a POSL rule which is a part of discount rule base from [55]. '%' and '/%...%/' are used to write single and multiple line comments in POSL.

```
% Example1: An example POSL rule

/% Rule: A discount of 5.0 percent is given to a ?customer

on a certain ?product, if that ?customer is premium and the ?product is a regular product. %/

discount(?customer, ?product, "5.0 percent") :-
premium(?customer), regular(?product).
```

The POSL parser can output an optional Concrete Syntax Tree (CST) that shows the input POSL rules in a graphical tree structure. If POSL rules have any syntax error, the rule with syntax error will be highlighted in red to show the error. Figure 6.2 shows the parse result of the POSL rule in example 1 as a CST tree.

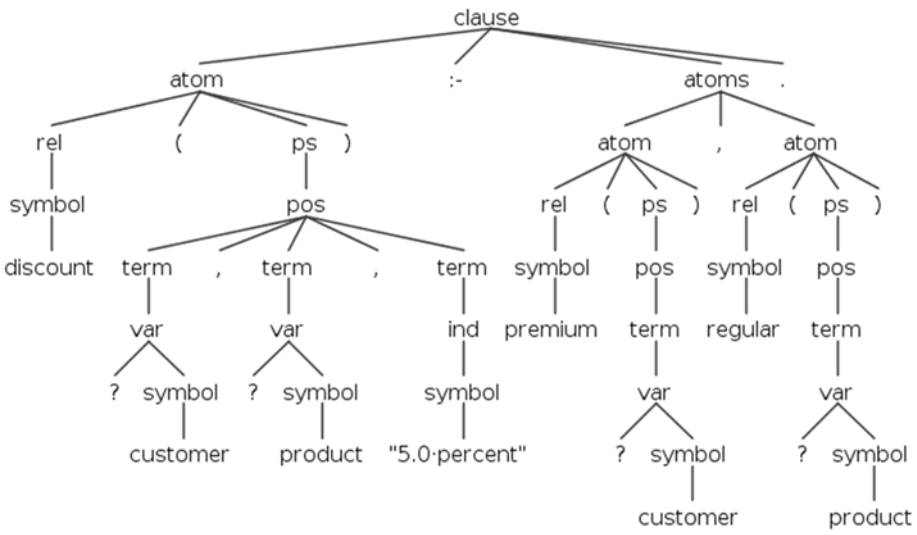

Figure 6.2. CST tree of POSL rule in example 1

6.2 POSL Translator

'Atoms' in POSL are mapped to 'uniterms' in RIF. The following production rule shows uniterm definition in RIF:

```
uniterm: const  '(' (term* | (NCName '->' term)*) ')' ;
```

Table 6.2 shows a summary of how different types of atoms (positional, slotted, positional/slotted, Cterm and Plex) in POSL are translated to uniterms in RIF:

Table 6.2. POSL atoms to RIF Uniterms translation

POSL Atom	RIF Uniterm
Positional	const '(' term* ‘)’
Slotted	const '((NCName '->' term)* ')' ;
Positional + slotted	First, it is converted to an equivalent slotted atom in POSL then it is translated to uniterm in RIF
Cterm	Cterm atoms are treated as regular atoms
Plex	First, it is converted to Cterm then the Cterm is translated to RIF

Since there is not an equivalence for a POSL’s positional/slotted atom in RIF, therefore the following strategy is used to translate it to a RIF uniterm: A positional/slotted atom in POSL is first converted to an all-slotted atom then it is converted to a uniterm in RIF. In other words, all positional arguments in a positional/slotted atom are encoded as a single slotted argument with the artificial constant ‘pos’ as the key and ‘[<positional arguments>]’ as the value in the ‘key -> value’ pair of a slot. See the following example:

```
shipment(PC,47.5;source->BostonMoS;dest->LondonSciM).
<~~encoding~~>
shipment(pos->[PC,47.5];source->BostonMoS;dest->LondonSciM).
```

In POSL Cterm, Individual, Variable, and Skolem can have an arbitrary type. In other words, they can optionally be followed by a colon ‘:’ and a type name. A skolem is defined

as being an underscore, optionally followed by a symbol for a named skolem. If a Cterm, individual or skolem has a type then in translating them to RIF, their type is converted to a class membership. Variables in RIF cannot have class membership. Therefore, if a variable in POSL has a type name, its type is ignored while translating it to a RIF variable.

For translating (positional, slotted and positional/slotted) Plex to uniterm in RIF first, it is converted to Cterm then the Cterm is translated to RIF. Atoms in POSL can be webized by using an OID (a URI possibly prefixed by a symbolic name) as a 'zeroth' argument separated from further arguments by an up-arrow infix '^' for example relation(oid^arg1 ... argN). An OID of a POSL atom is mapped to 'irimeta' in RIF. For more details about POSL to RIF rule translation, refer to the POSL component of the framework. Also, to get more information about POSL and RIF rule languages refer to [7] and [14] respectively. Appendix B shows a POSL rule base (borrowed from [55]) and its translation to RIF using POSL component of the framework.

This chapter introduced a parser and translator for POSL rules to RIF rules. Parser allows writing, editing and verifying POSL rules directly in the framework's implemented demo. The parser can also create a concrete syntax tree view of the input POSL rules. The POSL translator translates the positional, slotted, positional+slotted, Cterm and Plex in POSL to Uniterms in RIF. To verify the translation of POSL rules to RIF, a RIF parser (as another part of the framework) can be used.

Chapter 7

SWRL and RuleML Translation

This chapter explains the SWRL and RuleML translations to RIF. The contributions in this chapter include an SWRL to RIF-BLD and a RuleML to RIF-BLD translator. An XSL stylesheet in the framework aims at translating SWRL rules (in XML-based syntax) to RIF-BLD. The XSL stylesheet can be used separately from the framework too. There is also another XSL stylesheet that translates Unary/Binary Datalog RuleML to RIF-BLD. For translating RuleML to RIF-BLD, the latest version of Datalog RuleML to date (version 1.02) is used. Only RIF-Core will be needed for Datalog-level interchange. It should be noted that there are situations where SWRL or Datalog RuleML rules cannot be translated to RIF without losing semantics. Also, there are situations where the translation is out of the scope of this dissertation (e.g. translating negation). In these cases, the input rules still will be translated to RIF with ignoring parts that cannot be translated. SWRL and Datalog RuleML translators (XSL stylesheets) are in their early stages and do not cover all aspects of SWRL/Datalog RuleML. These XSLT stylesheets can be used (with some changes) as a start point in translating these rule languages to other rule markup languages.

7.1 SWRL Translation

Semantic Web Rule Language (SWRL) [6] is based on a combination of OWL DL and OWL Lite sublanguages of OWL [56] with the Unary/Binary Datalog sublanguage of RuleML. SWRL has both an XML-based syntax for machine interoperability and a human-readable syntax. Since there is no translator in literature to interchange rules between

SWRL and RIF, a translator would be valuable to convert SWRL rules to RIF rules. This translation can be either through SWRL XML-based syntax or presentation syntax. The translator proposed in this work in based on SWRL XML-based syntax.

Rules in SWRL are in the form of an antecedent-consequent pair. Antecedent (head) and consequent (body) of SWRL rules consist of a conjunction of one or more atoms. SWRL tends to be a reasoning layer built on top of OWL. All rules in SWRL are expressed in the form of OWL classes. SWRL provides seven types of atoms: Class atoms (owl:Class) e.g. Person(?p), individual property atoms (owl:ObjectProperty) e.g. hasBrother(?x, ?y), data valued property atoms (owl:DatatypeProperty) e.g. hasAge(?x, ?age), different individuals atoms e.g. differentFrom(?x, ?y), same individual atoms e.g. sameAs(?x, ?y), built-in atoms and data range atoms. SWRL's built-ins are based on the reuse of existing built-ins in XQuery and XPath. An SWRL's data range atom consists of a datatype name or a set of literals and a single argument representing a data value e.g. xsd:int(?x).

The XSL stylesheet file 'SWRL2RIF.xsl' is part of the framework and is responsible for translating SWRL rules to RIF-BLD rules. The details of translation are described as follows. Table 7.1 shows some of the main elements (XML tags) of SWRL documents.

Table 7.1. SWRL's main document elements

Element	Description
<swrlx:Ontology>	Root element in SWRL documents
<ruleml:imp>	Define rule
<ruleml:_head>	Rule head
<ruleml:_body>	Rule body
<ruleml:var>	Rule variable
<ruleml:_rlab>	A URI to optionally name a rule axiom
<owlx:Annotation>	Annotation

Table 7.2. SWRL element mapping to RIF

SWRL	RIF
<swrlx:Ontology>	<Document> <payload>
<ruleml:imp>	<sentence> <Forall> <xsl:call-template name="declare"/> <formula> <Implies>
<ruleml:_body>	<if> if <ruleml:_body> has more than one child (clause) they will be conjunct (AND) together.
<ruleml:_head>	<then> if <ruleml:_head> has more than one child they will be conjunct together.
<ruleml:_rlab>	Rule name is ignored in this work. It can be developed as future work
<owlx:Annotation>	Annotation is ignored in this work. It can be developed as future work.
<ruleml:var>	<declare> <Var>
<swrlx:classAtom>	Only named classes are supported here. <formula> <Member> <class> Or <formula> <Member> <instance>
<swrlx:individualPropertyAtom>	It is mapped to a (predicate) Atom in RIF.
<swrlx:datavaluedPropertyAtom>	Same as swrlx:individualPropertyAtom is mapped to <Atom> in RIF.
<swrlx:sameIndividualAtom>	It is mapped to Equal in RIF. <formula> <Equal> <left> <right>
<swrlx:differentIndividualsAtom>	Not supported.
<swrlx:builtinAtom>	Depend on the type of built-in, it is mapped to rif-builtin-predicate or rif-builtin-function.
<owlx:DataValue>	<Const>
<owlx:Individual>	<Const>

Table 7.2 shows some of the mapping considerations from SWRL rules to RIF. For details about XML-based SWRL rule documents, their XML elements, structures and so on, refer to XML Schema documents (XSD) of SWRL. Some of the features of SWRL rules are as follows.

- SWRL documents can contain OWL axioms.
- Multiple rules can be defined in an SWRL document.
- Rule variables are globally defined. In other words, variables used in every rule are defined (globally) before defining the rules.
- Each rule can have a name or ID.

In this work, elements of SWRL documents that are pure OWL elements are ignored and will not be translated to RIF. As mentioned earlier, there are seven types of atoms in SWRL rules: Class, Datarange, IndividualProperty, DatavaluedProperty, SameIndividual, DifferentIndividual and Builtin atoms. In translating class atoms of SWRL to RIF, only named classes are supported and six other types of classes (DataRestriction, ObjectRestriction, OneOf, UnionOf, IntersectionOf and ComplementOf which are types of OWL classes) are not supported.

Built-ins in SWRL will be mapped to RIF Atoms or Expressions. A built-in atom is mapped to a ‘rif-builtin-predicate’ or ‘rif-builtin-function’. RIF predicate is a RIF atom and RIF function is a RIF Expression. Currently, some of the SWRL Math built-ins (e.g. swrlb:add, swrlb:subtract, swrlb:multiply, and swrlb:divide) are mapped to RIF built-ins. The rest of built-ins will be added later as future work.

For more information regarding details of SWRL to RIF translation see XSL stylesheet ‘SWRL2RIF.xsl’. Example 1 (borrowed from [6]) in Appendix C shows a translation of an SWRL rule base to a RIF rule base using ‘SWRL2RIF.xsl’ translator.

7.2 RuleML Translation

The Rule Markup Language (RuleML) is a system of three families of XML-based rule languages including Deliberation, Reaction and Consumer RuleML introduced by RuleML Initiative [5]. RuleML has a modular, hierarchical nature that covers different types of rules including derivation rules, integrity constraints, production rules and reaction rules. Datalog RuleML is a function-free sub-language of Deliberation RuleML. Atomic formulae in Datalog RuleML are n-ary and each argument in an atomic formula can be positional or slotted (OO-Datalog). Datalog RuleML also supports universal and existential quantifiers, object identifiers, performatives (e.g. assert, retract), logical connectives and terms. Terms can be literal values, variables, individuals, etc. This dissertation covers Unary/Binary Datalog RuleML translation to RIF. Table 7.3 shows some of the mapping considerations from Unary/Binary Datalog RuleML rules to RIF. It should be noted that only RIF-Core will be needed for Datalog-level RuleML translation to RIF.

RuleML documents have a default namespace, and so as RIF documents. Therefore, while translating a RuleML document to a RIF document, RuleML default namespace is converted to ‘xmlns:ruleml’ namespace in RIF. ‘ruelml:Retract’ and ‘ruleml:Query’ are not supported in this work. Although, their translation can be developed as future work. ‘ruleml:Assert’ performative specifies that its content (optionally surrounded by a <formula> role) is asserted, making an 'implicit <Rulebase>' assumption [57].

Table 7.3. RuleML to RIF mappings

RuleML	RIF
<RuleML>	<Document> <payload> <xsl:apply-templates select="ruleml:Assert"/> <xsl:apply-templates select="ruleml:Retract"/> <xsl:apply-templates select="ruleml:Query"/> </payload>
<Atom>	<Atom> <xsl:call-template name="uniterm"/> </Atom>
<Implies>	<Implies> <xsl:apply-templates select="ruleml:if"/> <xsl:apply-templates select="ruleml:then"/> </Implies>
<And>	<And>
<Or>	<Or>
<Equivalent>	<Equal> <xsl:apply-templates select="ruleml:torso"/> </Equal>
<torso>	It is mapped to <left> or <right>. It depends on the <torso> position as a child inside its parent
<Forall>	<Forall> <xsl:call-template name="declare"/> <xsl:apply-templates select="*[name()!='Var']"/> </Forall>
<Entails>	Not supported.
<Rulebase>	<Group>
<op>	<op>
<Rel>	<Const>
<Ind>	<Const>
<slot>	<slot>
<Data>	<Const>
<Var>	<Var>
<Skolem>	<Const>

If all children of an atom in RuleML are slots, that atom is translated to uniterm{op, slot*} otherwise it is translated to uniterm{op, args} in RIF. Relax NG Compact syntax (RNC) format [58] of 'uniterm' production rule in RIF is as follows:

```
rif:uniterm={ IRIMETA?, op, (args | slot*) }
```

For more details on the translation of Unary/Binary Datalog RuleML to RIF see XSL stylesheet 'UBDatalogRuleML2RIF_BLD.xsl' in the framework. Appendix C shows two rule examples in a Unary/Binary Datalog RuleML documents (from [59]) and their translation to RIF using 'UBDatalogRuleML2RIF_BLD' translator. SWRL and RuleML to RIF XSL stylesheet translators are a promising start point in translating these languages to RIF and can be used to translate SWRL and Datalog RuleML to other rule languages (with modifications) as well.

This chapter focused on translation of XML-based SWRL and Datalog RuleML rules to RIF. Two separate XSL stylesheets were developed to perform the translations. This chapter described in details what features of SWRL and Datalog RuleML will be translated to RIF. For every feature, a template is defined in the XSL stylesheet to translate that feature of SWRL/Datalog RuleML to RIF.

Chapter 8

Rule Interchange Format (RIF)

The previous chapters described how to transform Notation3, POSL, SWRL, and Unary/Binary Datalog RuleML rule languages to RIF-BLD language. The contributions were an ANTLR grammar, rule parser, rule visualizer and a translator to RIF for each of the Notation3 and POSL languages. Also, an SWRL to RIF and RuleML to RIF translators (in the form of XSLT stylesheets) were other contributions.

This chapter is focused on RIF-BLD language and has four main contributions: Introducing an ANTLR grammar for RIF, a RIF parser, a RIF engine and a RIF translator. They are parts of the implemented prototype of the framework proposed in this dissertation. The RIF parser can be used as a standalone component to write, edit and verify rules in RIF-BLD. The parser is a part of the rule engine as well. The rule engine is both a top-down and bottom-up inference engine for RIF-BLD. The presented rule engine is the first rule engine developed for RIF language. The last but not least contribution of this chapter is a rule translator for RIF. The translator maps RIF rules in presentation syntax to equivalent rules in RIF XML-based syntax.

The Rule Interchange Format (RIF) was introduced by W3C, beginning in 2005, to create a standard for exchanging rules among rule systems, in particular among Web rule languages and engines. RIF focuses on rule exchange rather than trying to develop a single one-fits-all rule language [14]. Therefore, RIF has introduced dialects (e.g. RIF-Core, RIF-BLD, RIF-PRD etc.). In other words, for every category of rule languages, there is a specific version of RIF that can help to exchange rules among rule languages in that

category. Moreover, RIF can be used as a base to define new Web rule languages. Figure 8.1 shows the idea of RIF as an interchange language between different rule languages:

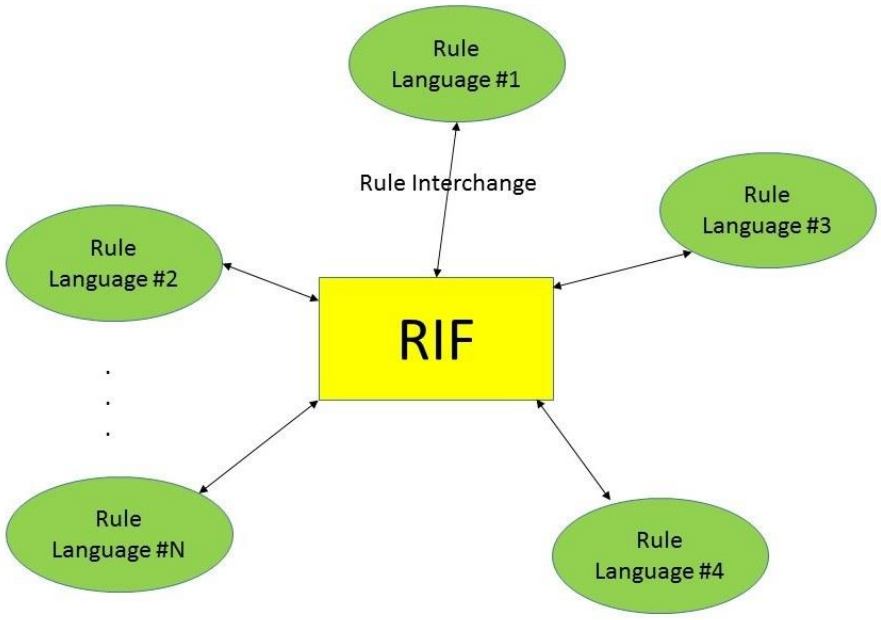

Figure 8.1. RIF idea for rule interchange

Before describing the RIF component of the framework in details, a short introduction to RIF-BLD language structure is presented. This introduction first explains what elements a generic Web rule language consists of (e.g. variables, terms, formulas, etc.). Then it describes those elements in RIF-BLD. In other words, it explains RIF-BLD syntactical structure. Knowing the syntax of a rule language is necessary for developing a parser, inference engine, translator, etc. for that language. Section 7.1 describes RIF-BLD syntactical structure and sections 7.2 to 7.4 define the RIF parser, engine, and translator respectively.

8.1 RIF-BLD

From a theoretical perspective, RIF-BLD corresponds to the language of definite Horn rules with equality and a standard first-order semantics [60]. RIF-BLD share certain characteristics with ISO Common Logic (ISO-CL) [61] which itself is an evolution of Knowledge Interchange Format (KIF) [62] and Conceptual graphs (GC) [63]. Some of these characteristics are XML (as normative syntax), IRIs (as identifiers), datatypes and built-ins. Unlike CL, RIF-BLD was designed to be a simple dialect with a limited expressiveness that lies within the intersection of first-order and logic-programming systems. This is why RIF-BLD does not support negation [24]. Since there are many rule languages with various features, it is not expected that most of the rule languages be translatable to RIF-BLD.

Every rule language including RIF dialects has two aspects: syntax and semantics. The syntax of a rule language needs to have an alphabet, definition of terms and formulas to be able to create, present, and check that whether a rule is structurally correct in that language. The semantics of a rule language is the study of meaning in formal and natural language using logic as an instrument. It shows that if a rule in that language is true or false. Semantics also show that how to infer new knowledge (facts and rules) from currently available facts and rules in a rule base. The first step in exchanging rules and make interoperability between rule languages possible is to know the type of the rule language (normative, deductive, and reactive), dimensions and limitations of the language, and finally the syntax and semantics of a rule language.

RIF-BLD has two different syntaxes: Presentation syntax and serialization syntax. The presentation syntax is a human-readable syntax. The serialization syntax uses XML for

presenting rules and can be obtained as a serialization of its presentation syntax [64]. The syntax (i.e. facts and rules) of every rule language consists of alphabet, terms, and formulae which themselves consist of constants, variables, built-ins, functions, connective symbols, quantifiers, and other auxiliary symbols (see Figure 8.2).

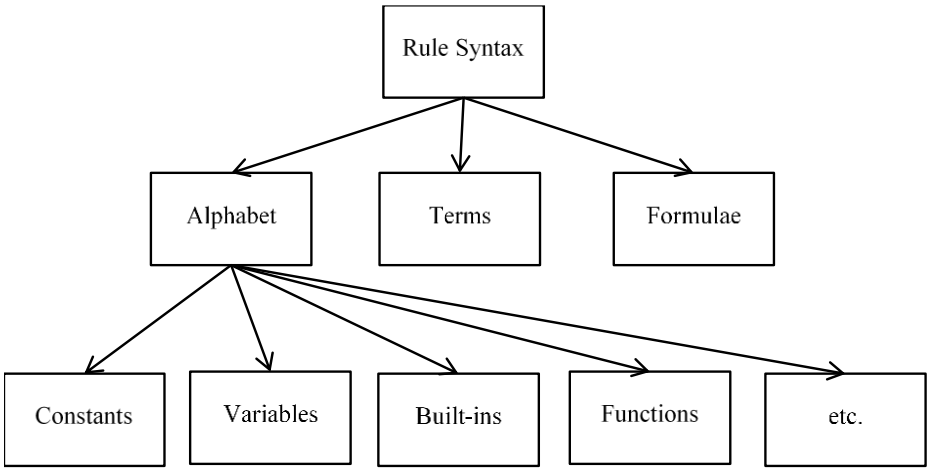

Figure 8.2. General syntactical structure of a rule language

The following list shows more details about the elements of a rule language that RIF-BLD covers as well [24]. The alphabet of RIF-BLD consists of the following symbols:

- Sets of constants, variables, and argument names
- connective symbols And, Or, and :-
- quantifiers Exists and Forall
- Symbols =, #, ##, ->, External, Import, Prefix, and Base
- Symbols ‘Group’ and ‘Document’
- Symbols for representing lists: List
- the auxiliary symbols (,), [,], <, >, and ^^

RIF-BLD defines several kinds of terms: constants and variables, positional terms, terms with named arguments, plus equality, membership, subclass, frame, and external terms[24]. In other words, terms in RIF-BLD are consist of:

- Constants and variables
- Positional terms
- Terms with named arguments e.g. t(s1->v1 ... sn->vn) where constant 't' represents a predicate or function. Terms with named arguments are introduced to support the exchange of rules in languages that permit predicate or function arguments to be named and/or unordered [24] such as slotted arguments in POSL [7].
- Lists. There are two types of lists: closed list in the form of List(t1 ... tm), and open list in the form of List(t1 ... tm | t). The open list is a list with a tail (or list rest).
- Equality (=), Membership (#), SubClass (##), Frame and Externally defined terms are the other types of terms supported by RIF-BLD.

Membership, subclass, and frame terms are used to describe objects and class hierarchies. External terms are used for representing built-in functions. Formulas in RIF-BLD are consists of:

- Atomic formula, which can be positional or named arguments terms, as well as Equality, membership, subclass, and frame terms.
- Condition formula (conjunction, disjunction and existential)
- Rule implication
- Universal rule/fact (Forall)
- Group (of rules and facts)

- Document formula

For more details of RIF-BLD formulas refer to [24].

8.2 RIF-BLD Parser

The RIF parser developed here is a parser for RIF-BLD language. This section describes how the parser works. Rule antecedent in RIF-BLD rules can be a conjunction, disjunction or existential of formulas. Rule consequent can be a conjunction of formulas. Disjunction and existential formulas are not allowed in rule consequent. From the RIF-BLD alphabet, terms, and formulas (see section 7.1) we can infer that what features of a rule language is available in RIF-BLD. Appendix D shows RIF-BLD grammar in EBNF notation. ANLTR 4 [22] notation of RIF-BLD grammar (developed here) is shown in Appendix D as well. Moreover, RIF-BLD grammar (in ANTLR notation) is available as a visual graph structure in the documents of the framework's implemented demo. The developed ANTLR 4 grammar of RIF-BLD does not exactly follow its EBNF grammar for the sake of building the parser. To see the RIF-BLD parser in action consider the following rule base (from [24]):

Table 8.1. An example rule in RIF-BLD

```
Document(
  Base(<http://example.com/people#>)
  Prefix(cpt <http://example.com/concepts#>)
  Prefix(bks <http://example.com/books#>)
  Group (
    Forall ?Buyer ?Item ?Seller (
       cpt:buy(?Buyer ?Item ?Seller) :- cpt:sell(?Seller ?Item ?Buyer)
    )
    cpt:sell(<John> bks:LeRif "Mary"^^rif:iri)
  ))
```

The example in Table 8.1 states:

In English:
• A buyer buys an item from a seller if the seller sells the item to the buyer. • John sells LeRif to Mary.

Figure 8.3 shows the parser output (as a tree) after parsing the rules in Table 8.1. Generating the parse tree as the parser output is optional. If there are any syntactical errors in the input rules, the parser will highlight them in red.

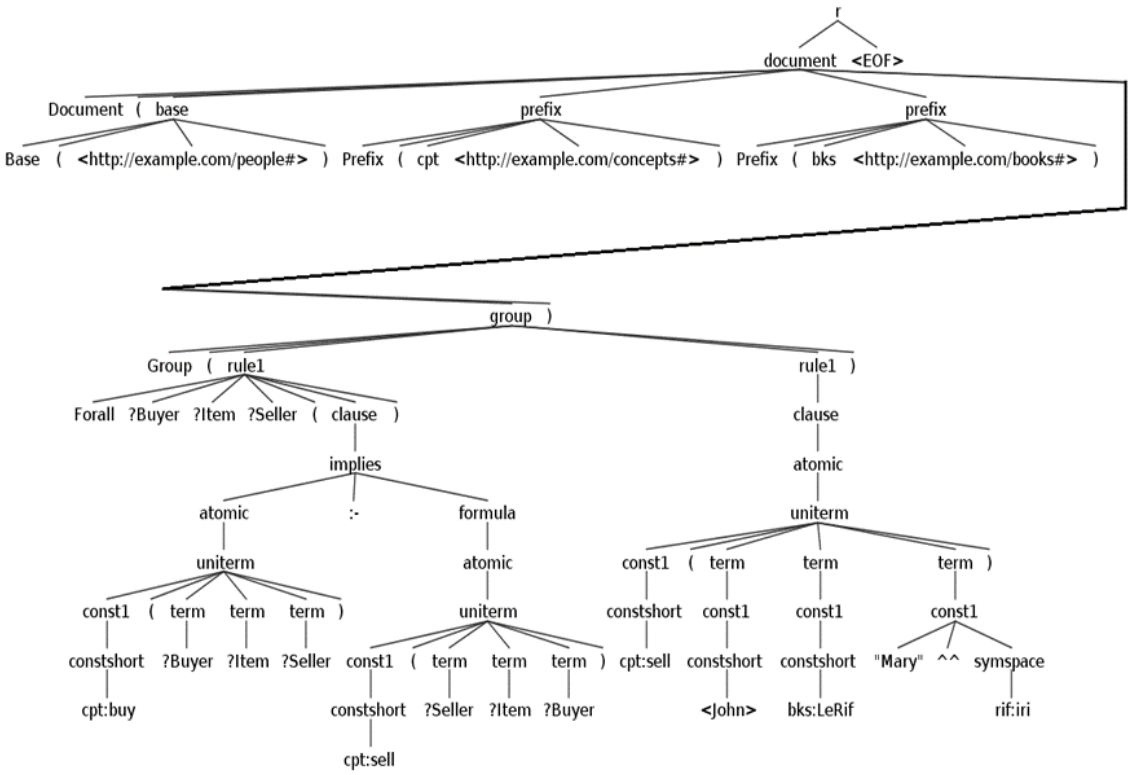

Figure 8.3. Parser output of rules in Table 8.1

8.3 RIF-BLD Rule Engine

This section describes the developed RIF rule engine. The first part of the rule engine (the parser) was explained above. This section goes into details of the inference part of the rule engine. Figure 8.4 shows the architecture diagram of the rule engine implementation.

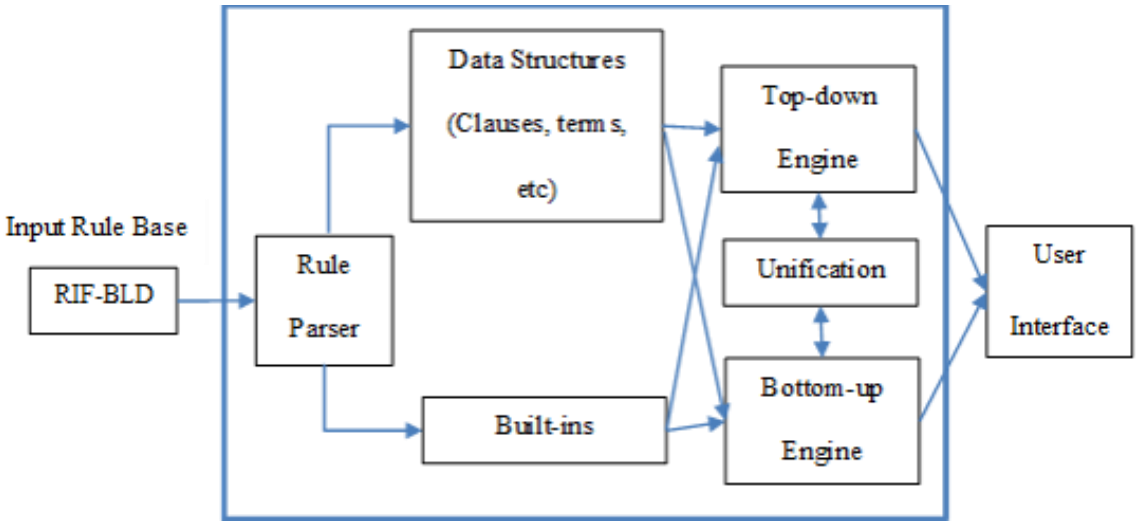

Figure 8.4. UML diagram of lexer and parser classes of RIF-BLD

This Rule engine is a Java based reasoning engine designed to support the features that are available in RIF-BLD. In order to implement the features presented in RIF-BLD a unification algorithm has been implemented that is capable of dealing clauses, terms, built-ins and others. The engine is developed as part of the framework and implements the major features of RIF-BLD. However, this rule engine is in its early stages and does not cover all features of RIF-BLD such as some built-ins and List. The current implementation is suitable for testing and executing small to medium size knowledge bases in RIF-BLD. To implement the rule engine, OO JDREW rule engine [21] has been used and followed as a reference especially when it comes to the implementation of the unification algorithm.

The key part of the rule engine is the unification algorithm that creates variable bindings and determines if two possibly two non-ground terms match and therefore can be used for a resolution step. The unification algorithm is described with the following steps. First, predicate symbols and arities must match. Second, for each pair of predicate arguments one of three conditions must be met: if both are constants then the constants must be the same, if one is a constant and one is a variable then the variable is bound to the constant, or if both are variables then one variable is bound to the other.

The following pseudo code shows the bottom-up algorithm of the rule engine at a high level of abstraction.

```
Input:  RIF BLD Knowledge Base

Output: Inferred New KNowledge

File BottomUpEngine(File RIFBLD_KnowledgeBase){

      Step 1: Parse_RIFBLD_KnowledgeBase();

      Step 2: Process_RIFBLD_KnowledgeBase();

}
```

The first step is to parse the KB to make sure there are no syntactical errors. Step 2 processes the KB to infer new knowledge using the forward reasoner.

```
Process_RIFBLD_KnowledgeBase(){

      ForwardReasoner();

      Subsumption();

      Unifier();

}
```

ForwardReasoner implements the forward reasoner (bottom-up) module. ForwardReasoner works by processing "new" facts; As each new fact is processed unificiation with all previously exisiting rules is attempted; if the unification is successful

one of two things happens: if the resolvent is a fact then it is added to the end of the new facts list; if the resolvent is a rule then it is processed (attempting unification with all processed facts) and is then added to the list of rules.

Subsumption is used for checking if one (newly selected) fact is subsumed by another fact that has already been processed. This class is used for checking if one (newly selected) fact is subsumed by another fact that has already been processed. Subsumption checking is a two step process; first the newly selected fact is ground (all variables are bound to newly created constants), once the fact has been made ground then subsumption is checked by attempting to unify the ground atom with other facts; if unification succeedes then the fact it was unified wtih subsumes the fact that was made ground.

Unifier is used to create a unifier that is used to test whether or not two terms are equal.

```
Unifier(){
      Unifier(DefiniteClause fact, DefiniteClause rule);
      resolvent();
      unify(Term term1, Term term2);
}
```

Unifier(fact, rule) method is used to create a unifier that is used to test whether a fact unifies with a rule. *resolvent* method will produce the resolvent of unifing a fact with a rule. *unify(term1, term2)* method is used to check if two terms unify with each other; and to perform any variable bindings that are necessary to make the terms unfiy.

The following pseudo code shows the top-down Algorithm (query answering module) of the rule engine at a high level of abstraction:

```
Input:  RIF BLD Knowledge Base and Query String
Output: Query Result
```

```
File TopDownEngine(File RIFBLD_KnowledgeBase, File Query){
      Step 1: Parse_RIFBLD_KnowledgeBase();
      Step 2: Parse_Query();
      Step 3: ProcessQuery();
}
```

Step 1 and 2 parse the KB and Query for syntactical errors and step 3 runs the Query against the KB.

```
ProcessQuery(){
      BackwardReasoner();
      Unifier(); applies variable bindings to the current Goal or
SubGoalList.
}
```

BackwardReasoner is the reasoner to find solution for Goals/GoalLists. *Unifier* applies variable bindings to the current Goal or SubGoalList.

```
BackwardReasoner(){
      DepthFirstSolutionIterator();
      Goal();
      GoalList();
}
```

DepthFirstSolutionIterator finds the next solution for query with depth first tree traversal. *Goal* contains an atomic formula to be solved. Each Goal object belongs to one GoalList and each Goal can have at most one SubGoalList attached to it at a time. The SubGoalList is a GoalList that forms the children of the Goal. A Goal is added to the GoalList if it is unifiable with the head atom of a Clause in the BackwardReasoner's input clause. A Goal

is fully solved if its SubGoalList is empty, or if all of the Goals in that SubGoalList are solved.

GoalList contains a list of Goals and is attached to its parent Goal. When a Goal in the GoalList is solved, the effect is to bind variables in the atomic formula. These bindings are immediately applied to the sibling Goals in this SubGoalList. Because a Goal may be attached to a failed SubGoalList, but other SubGoalLists are still available for it, it is important to be able undo the effects of this failed SubGoalList on the Goal, and consequently effects that have may been propagated throughout the BackwardReasoner.

```
Unifier(){
      Unified();
      applyToGoal();
      applyToGoalList();
}
```

Unifier performs two separate functions. It tells whether two clauses literals that occur in Goals/GoalLists clauses can unify, and if so, it builds the resulting Goals/GoalLists after the required substitution has been applied.

Unified is set to true if unification of the Goal and the first Goal in the subGoalList is successful, false otherwise. It should always be checked to see if it is set to true before calling the *applyToGoal* or *applyToGoalList* methods.

applyToGoal is used to apply the variable bindings to the current goal. *applyToGoalList* is used to apply variable bindings to the sub goal list.

Figure 8.5 shows the main classes of the rule engine implementation including rule translation and unifications classes.

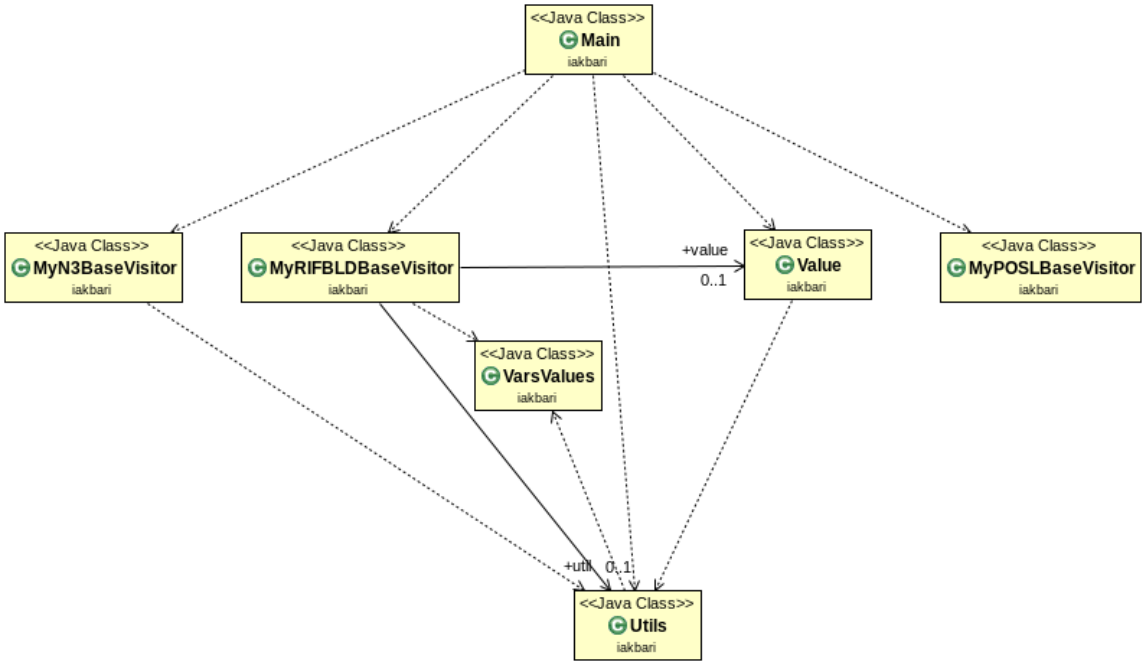

Figure 8.5. UML diagram of main classes of RIF-BLD rule engine

Figure 8.6 shows UML diagram of lexer and parser classes of RIF-BLD.

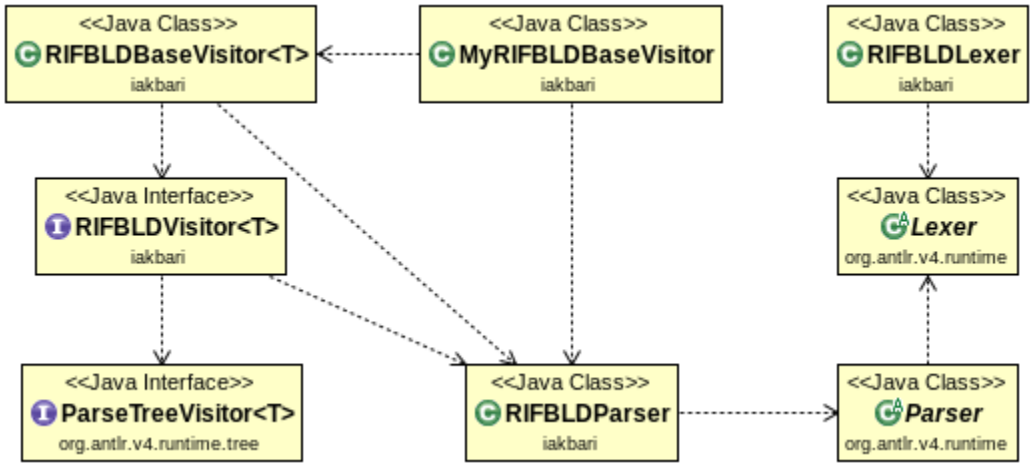

Figure 8.6. UML diagram of lexer and parser classes of RIF-BLD

The UML diagrams in Figure 8.5 and Figure 8.6 show the class relationships in the implemented rule engine. However, for simplicity class attributes and methods are not shown. UML diagrams of N3 and POSL parsers and translators are shown in Appendix F. Also, UML diagram for the classes to create the GUI of the framework's implementation are not shown here. They can be found in the demo's documentation. Table 8.2 shows the responsibilities of the more important classes (in Figure 8.5) of the rule engine:

Table 8.2. Description of classes in RIF rule engine

Class Name	Description
Main	Start point of the implemented framework
MyN3BaseVisitor	It translates N3 rules to RIF-BLD
MyPOSLBaseVisitor	It translates POSL rules to RIF-BLD
MyRIFBLDBaseVisitor	Start point of RIF-BLD rule inference
Utils	Class responsible for backtracking and unification
VarsValues	Aux class responsible for creating new data structures
Value	Aux class responsible for creating new data types

'Utils' class is the key class for implementing unification and backtracking of RIF-BLD rules. Design and implementation details of RIF-BLD rule engine will not be presented here. You can refer to the Javadoc documentation presented in the framework's demo for those details. To see how the RIF-BLD rule engine works in practice the following example is used. Table 8.3 shows this example. In the rule example in Table 8.3, there are two prefixes, a rule group, three rules and nine facts. This rule base is imported in the rule engine (through the GUI of the framework's implemented demo) and then the rule engine reasons over the rules inside the rule base.

Table 8.3. A RIF-BLD rule base example

```
Document(
   Prefix(ex <http://example.org/example#>)
   Prefix(xs <http://www.w3.org/2001/XMLSchema#>)
   Group (
      Forall ?x ?y  ?z  ( ?y=?z   :-  And ( ex:p(?x ?y)  ex:p(?x ?z) ) )
      Forall ?Customer (
          And (ex:discount(value -> 10 customer -> ?Customer) ex:sendGift(?Customer) )
                :-   Exists ?Customer (Or( ex:gold(customer -> ?Customer) ex:female(customer ->
                     ?Customer) ))
      )
       Forall ?Customer (
          ex:discount(customer -> ?Customer value -> 5) :- And ( ex:silver(customer -> ?Customer)
           ex:female(customer -> ?Customer) )
      )
      ex:gold(customer -> "Jane Doe")
      ex:female(customer -> "Jane Doe")
      ex:gold(customer -> "John Doe")
      ex:silver(customer -> "Sil Ver")
      ex:female(customer -> "Sil Ver")
      ex:p(ex:a ex:b)
      ex:p(ex:a ex:c)
       bronze("Alice Morgan")
      female("Alice Morgan")
))
```

Table 8.4 shows the results of running the rule engine on the rules in Table 8.3.

Table 8.4. Inferred rules from rule engine on the rules in Table 8.3

```
Document (
        Prefix (ex <http://example.org/example#> )
        Prefix (xs <http://www.w3.org/2001/XMLSchema#> )
        Group (
                Forall ?x ?y ?z (
                 ?y = ?z :- And ( ex:p ( ?x ?y ) ex:p ( ?x ?z ) ) )
                Forall ?Customer (
                 And ( ex:discount ( value -> 10 customer -> ?Customer ) ex:sendGift ( ?Customer ) )
                  :- Exists ?Customer ( Or ( ex:gold ( customer -> ?Customer ) ex:female ( customer -> ?Customer  ) ) ) )
                Forall ?Customer (
                 ex:discount ( customer -> ?Customer value -> 5 )
                 :- And ( ex:silver ( customer -> ?Customer ) ex:female ( customer -> ?Customer ) ) )
                ex:gold ( customer -> "Jane Doe" )
                ex:female ( customer -> "Jane Doe" )
                ex:gold ( customer -> "John Doe" )
                ex:silver ( customer -> "Sil Ver" )
                ex:female ( customer -> "Sil Ver" )
                ex:p ( ex:a ex:b )
                ex:p ( ex:a ex:c )
                bronze ( "Alice Morgan" )
                female ( "Alice Morgan" )
                ex:b = ex:c
                ex:discount ( value -> 10 customer -> "John Doe" )
                ex:discount ( value -> 10 customer -> "Jane Doe" )
                ex:discount ( value -> 10 customer -> "Sil Ver" )
                ex:sendGift ( "John Doe" )
                ex:sendGift ( "Jane Doe" )
                ex:sendGift ( "Sil Ver" )
                ex:discount ( customer -> "Sil Ver" value -> 5 )        ) )
```

The rule engine has inferred eight new facts from the initial rule base. These new facts are highlighted in bold in Table 8.4. The RIF-BLD rule engine is still under development and at this time it does not cover all features of the RIF-BLD language. Some of the features that are not covered by the rule engine are:

- Import
- Built-ins and data types
- List

These features can be developed as future work. The next section describes RIF-BLD translator which translates RIF rules in presentation syntax to XML-based syntax.

8.4 RIF-BLD Translator

The RIF-BLD translator maps RIF rules from presentation syntax to XML syntax. Figure 8.7 shows the translation process flow:

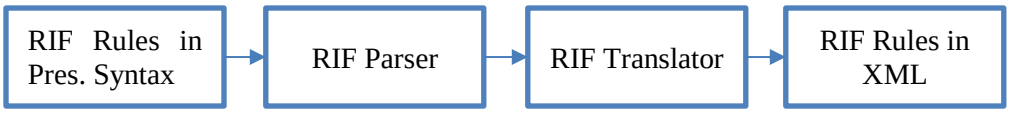

Figure 8.7. RIF translation process flow

RIF translator takes RIF rules as input (entered directly in the GUI of the framework's implementation or read from a RIF file) and translates them to their equivalent XML-based version. First, the RIF parser parses the input rules for any syntax error. If the rules were syntactically correct, then the parser builds a parse tree based on the input. The translator traverses the tree in depth-first search and translates the tree nodes to their equivalent XML tags.

This section defines the mapping, Xbld, from presentation syntax to XML syntax of RIF-BLD. Mapping are shown in the upcoming tables. Each row in the tables shows a mapping of a particular syntactic pattern in the presentation syntax. The right column of a row shows corresponding XML pattern. Italic symbols in presentation syntax represent meta variables and Xbld(metavar) in right column is a recursive application of Xbld to the presentation syntax represented by the 'metavar' metavariable. Question mark '?' in a pattern in the tables means optional. The mapping tables from presentation syntax to XML syntax of RIF-BLD are borrowed from [24].

8.4.1 Mapping of the Condition Language

Each row in the Table 8.5 indicates a translation Xbld(presentation)=XML. The function remove-outer-quotes used in the translation removes enclosing double quotes from a string. In the translation table, the positional and named-argument terms that occur in the context of atomic formulas are denoted by expressions of the form pred(...) and the terms that occur as individuals are denoted by expressions of the form func(...).

Table 8.5. Mapping of the RIF-BLD Condition Language

Presentation Syntax	XML Syntax

And (
  conjunct1
  . . .
  conjunctn
)
	<And>
	  <formula>χbld(conjunct1)</formula>
	  . . .
	  <formula>χbld(conjunctn)</formula>
	</And>

Or (
  disjunct1
  . . .
  disjunctn
)
	<Or>
	  <formula>χbld(disjunct1)</formula>
	  . . .
	  <formula>χbld(disjunctn)</formula>
	</Or>

Exists
  variable1
  . . .
  variablen (
  premise
)
	<Exists>
	  <declare>χbld(variable1)</declare>
	  . . .
	  <declare>χbld(variablen)</declare>
	  <formula>χbld(premise)</formula>
	</Exists>

External (atomexpr)	<External> <content>χbld(atomexpr)</content> </External>
pred ( )	<Atom> <op>χbld(pred)</op> </Atom>

pred (
  argument1
  . . .
  argumentm
)
	<Atom>
	  <op>χbld(pred)</op>
	  <args ordered="yes">
	  χbld(argument1)
	  . . .
	  χbld(argumentm)
	  </args>
	</Atom>

func ( )	<Expr> <op>χbld(func)</op> </Expr>

func (
  argument1
  . . .
  Argumentm
)
	<Expr>
	  <op>χbld(func)</op>
	  <args ordered="yes">
	  χbld(argument1)
	  . . .
	  χbld(argumentm)
	  </args>
	</Expr>

List (
  element1
  . . .
  elementn
)
	<List>
	  <items ordered="yes">
	  χbld(element1)
	  . . .
	  χbld(elementn)
	  </items>
	</List>

List (
  element1
  . . .
	<List>
	  <items ordered="yes">
	  χbld(element1)

elementn | remainder)	. . . χbld(elementn) </items> <rest>χbld(remainder)</rest> </List>
pred (name1 -> filler1 . . . namen -> fillern)	<Atom> <op>χbld(pred)</op> <slot ordered="yes"> <Name>χbld(name1)</Name> χbld(filler1) </slot> . . . <slot ordered="yes"> <Name>χbld(namen)</Name> χbld(fillern) </slot> </Atom>
func (name1 -> filler1 . . . namen -> fillern)	<Expr> <op>χbld(func)</op> <slot ordered="yes"> <Name>χbld(name1)</Name> χbld(filler1) </slot> . . . <slot ordered="yes"> <Name>χbld(namen)</Name> χbld(fillern) </slot> </Expr>
inst [key1 -> filler1 . . . keyn -> fillern]	<Frame> <object>χbld(inst)</object> <slot ordered="yes"> χbld(key1) χbld(filler1) </slot> . . . <slot ordered="yes"> χbld(keyn) χbld(fillern) </slot> </Frame>
inst # class	<Member> <instance>χbld(inst)</instance> <class>χbld(class)</class> </Member>
sub ## super	<Subclass> _{χbld(sub)} <super>χbld(super)</super> </Subclass>

left = right	<Equal> <left>χbld(left)</left> <right>χbld(right)</right> </Equal>
"unicodestring"^^symspace	<Const type="symspace">unicodestring</Const>
?name1	<Var>χbld(name1)</Var>
namei	remove-outer-quotes(namei)

8.4.2 Mapping of the Rule Language

The Xbld mapping from the presentation syntax to the XML syntax of the RIF-BLD Rule Language is specified by Table 8.6. While the Import directive is handled by the presentation-to-XML syntax mapping, the Prefix and Base directives are not. Each Prefix declaration becomes an ENTITY declaration within a DOCTYPE DTD attached to the RIF-BLD Document. The Base directive is mapped to the xml:base attribute in the XML Document tag.

Table 8.6. Mapping of the RIF-BLD Rule Language

Presentation Syntax	XML Syntax

Document(
  Import(loc1 prfl1?)
  . . .
  Import(locn prfln?)
  group?
)
	<Document>
	  <directive>
	  <Import>
	  <location>χbld(loc1)</location>
	  <profile>χbld(prfl1)</profile>?
	  </Import>
	  </directive>
	  . . .
	  <directive>
	  <Import>
	  <location>χbld(locn)</location>
	  <profile>χbld(prfln)</profile>?

	</Import> </directive> <payload>χbld(group)</payload>? </Document>
Group(clause1 . . . Clause)	<Group> <sentence>χbld(clause1)</sentence> . . . <sentence>χbld(clausen)</sentence> </Group>
Forall variable1 . . . variablen (rule)	<Forall> <declare>χbld(variable1)</declare> . . . <declare>χbld(variablen)</declare> <formula>χbld(rule)</formula> </Forall>
conclusion :- condition	<Implies> <if>χbld(condition)</if> <then>χbld(conclusion)</then> </Implies>

8.4.3 Mapping of Annotations

The χbld mapping from RIF-BLD annotations in the presentation syntax to the XML syntax is specified by Table 8.7. The metavariable Typetag in the presentation and XML syntaxes stands for any of the class names And, Or, External, Document, or Group, and Quantifier for Exists or Forall. The dollar sign, $, stands for any of the binary infix operator names #, ##, =, or :-, while Binop stands for their respective class names Member, Subclass, Equal, or Implies.

Table 8.7 Mapping of the RIF-BLD Annotations

Presentation Syntax	XML Syntax
(* iriconst? frameconj? *) *Typetag* (e1 . . . en)	<Typetag> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? e1' . . . en'

	</Typetag> where e_1', . . ., e_n' are defined by the equation χbld(Typetag(e1 . . . en)) = <Typetag>e1' . . . en'</Typetag>
(* iriconst? frameconj? *) *Quantifier* variable1 . . . variablen (formula)	<Quantifier> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? <declare>χbld(variable1)</declare> . . . <declare>χbld(variablen)</declare> <formula>χbld(formula)</formula> </Quantifier>
(* iriconst? frameconj? *) pred (argument1 . . . argumentn)	<Atom> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? <op>χbld(pred)</op> <args ordered="yes"> χbld(argument1) . . . χbld(argumentn) </args> </Atom>
(* iriconst? frameconj? *) func (argument1 . . . argumentn)	<Expr> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? <op>χbld(func)</op> <args ordered="yes"> χbld(argument1) . . . χbld(argumentn) </args> </Expr>
(* iriconst? frameconj? *) *List* (element1 . . . elementn)	<List> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? <items ordered="yes"> χbld(element1) . . . χbld(elementn) </items> </List>
(* iriconst? frameconj? *) *List* (element1	<List> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>?

. . . elementn | remainder)	<items ordered="yes"> χbld(element1) . . . χbld(elementn) </items> <rest>χbld(remainder)</rest> </List>
(* iriconst? frameconj? *) pred (name1 -> filler1 . . . namen -> fillern)	<Atom> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? <op>χbld(pred)</op> <slot ordered="yes"> <Name>χbld(name1)</Name> χbld(filler1) </slot> . . . <slot ordered="yes"> <Name>χbld(namen)</Name> χbld(fillern) </slot> </Atom>
(* iriconst? frameconj? *) func (name1 -> filler1 . . . namen -> fillern)	<Expr> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? <op>χbld(func)</op> <slot ordered="yes"> <Name>χbld(name1)</Name> χbld(filler1) </slot> . . . <slot ordered="yes"> <Name>χbld(namen)</Name> χbld(fillern) </slot> </Expr>
(* iriconst? frameconj? *) inst [key1 -> filler1 . . . keyn -> fillern]	<Frame> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? <object>χbld(inst)</object> <slot ordered="yes"> χbld(key1) χbld(filler1) </slot> . . . <slot ordered="yes"> χbld(keyn) χbld(fillern) </slot>

	</Frame>
(* iriconst? frameconj? *) e1 $ e2	<Binop> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? e1' e2' </Binop> where Binop, e1', e2' are defined by the equation χbld(e1 $ e2) = <Binop>e1' e2'</Binop>
(* iriconst? frameconj? *) unicodestring^^symspace	<Const type="symspace"> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? unicodestring </Const>
(* iriconst? frameconj? *) ?name1	<Var> <id>χbld(iriconst)</id>? <meta>χbld(frameconj)</meta>? χbld(name1) </Var>

The following shows some of the mapping considerations from presentation syntax to XML-based syntax (based on [24]):

- The positional and named-argument terms that occur in the context of atomic formulas (which are denoted as predicates) are translated to <Atom> and the terms that occur as individuals (and are denoted as functions) are translated to <Expr>.
- Each Prefix declaration becomes an ENTITY declaration within a DOCTYPE DTD attached to the RIF-BLD document. The Base directive is mapped to the 'xml:base' attribute in the XML Document tag.
- Translation is complete and covers the whole RIF-BLD language elements including 'And', 'Or', 'Exists', 'External', 'predication', 'function', 'List',

‘Frame’, ‘Member’, ‘Subclass’, ‘Equal’, ‘Constant’, ‘Variable’, ‘Import’, ‘Group’, ‘Forall’, ‘implication’ and ‘Annotation’ (irimeta).

To see more details of the translation, refer to the documentation of the framework’s implemented demo and [24]. For space preserving purpose, a simple RIF rule document is used here as an example to show how the translation works in practice. Table 8.8 shows this example and Table 8.9 shows its translation result in XML.

Table 8.8. A simple RIF rule document example

```
Document(
    Import(<http://example.org/mygraph> <http://www.w3.org/2007/rif-import-profile#OWL-
            DL-annotation>)
    Group (
        ?list = List( List(?x | ?y) | ?z )        )
)
```

Table 8.9. XML equivalent of the RIF document in Table 8.8

```
xml=<?xml version="1.0" encoding="UTF-8"?>
<Document
  xmlns="http://www.w3.org/2007/rif#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema#">
 <directive>
  <Import>
   <location>http://example.org/mygraph</location>
   <profile>http://www.w3.org/2007/rif-import-profile#OWL-DL-
annotation</profile>
  </Import>
 <directive>
 <payload>
 <Group>
 <sentence>
  <Equal>
   <left>
     <Var>list </Var>
```

```
    </left>
    <right>
     <List>
      <items ordered="yes">
       <List>
        <items ordered="yes">
         <Var>x</Var>
        </items>
        <rest>
         <Var>y</Var>
        </rest>
       </List>
      </items>
      <rest>
       <Var>z</Var>
      </rest>
     </List>
    </right>
   </Equal>
  </sentence>
</Group>
</payload>
</Document>
```

This chapter described RIF component of the framework in this thesis. This component includes a rule parser, visualizer, translator and more important a rule engine for RIF language. The parser parses the RIF-BLD rule documents (in human-readable syntax) and the visualizer builds and shows a tree structure of the parsed rules by the parser. A tree presentation of input facts and rules gives a more tangible view of them and makes rule writing and debugging an easier task. The RIF translator maps RIF rules in human-readable syntax to RIF XML. Finally, the rule engine is a top-down and bottom-up reasoning engine for RIF-BLD.

Chapter 9

Rule Translator System

9.1 System Design

This chapter describes how to use the implemented framework, its modules and their functionalities. The implemented framework is developed in Java and is available as a GitHub repository at [65]. It can be downloaded as a Java Web Start (JWS) file. JWS files have a '.jnlp' file extension. A JWS file is an XML formatted file and can be edited with any text editor. To run a JWS file you often receive a Java security message. To see how to solve this security issue, refer to Appendix G. Figure 9.1 shows a snapshot of the implemented demo's main GUI.

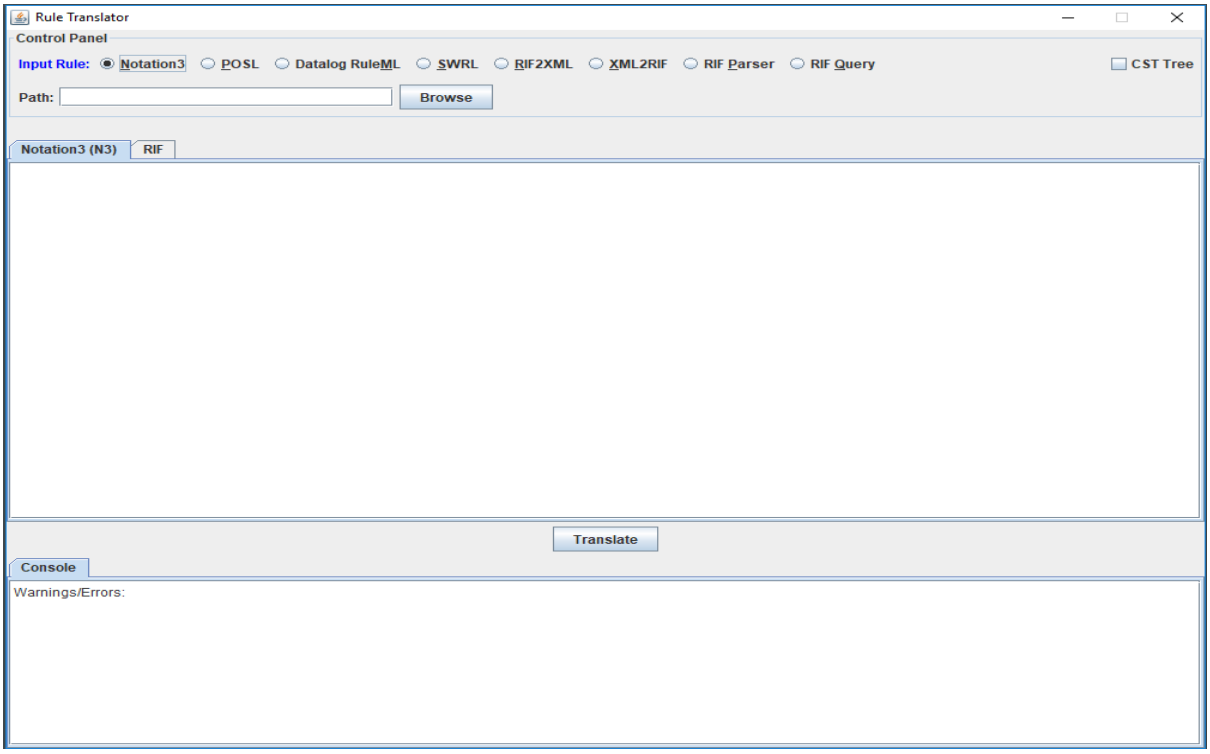

Figure 9.1. The main user interface of the framework's implemented demo

As it can be seen from Figure 9.1, the framework's demo has three main parts: control panel at top, source, and target languages in the middle and the error/warning messages box at the bottom. In control panel, the main task is selected which can be parsing, translation or querying the KB. The latter two include parsing as well. Options are: Notation3, POSL, Datalog RuleML, SWRL, RIF2XML, XML2RIF, RIF parser and RIF Query engine. Table 9.1 shows what each option does:

Table 9.1. Input languages and their responsibility in the framework's demo

Input Rule	Description
Notation3	It translates rules in N3 language to RIF-BLD language
POSL	It translates rules in POSL language to RIF-BLD language
RuleML	It translates rules in RuleML language to RIF-BLD language using XSLT stylesheets
SWRL	It translates rules in SWRL language to RIF-BLD language using XSLT stylesheets
RIF2XML	It translates rules in RIF-BLD presentation syntax to RIF-BLD XML-based syntax
XML2RIF	It translates rules in RIF-BLD XML syntax to RIF-BLD presentation syntax
RIF Query	This option queries the RIF knowledge base (rules and facts)

Rules from any of the input languages can be read in three different ways: they can be read from a URL, browsed from local machine or can be typed directly in source language text box. If the CST option is enabled and also checked, the appropriate parser (Notation3, POSL or RIF) can show the input rules in a concrete syntax tree diagram after parsing the rules. Since RuleML and SWRL do not have a parser, the CST tree option will not be enabled for them. RuleML and SWRL translators are defined in XSLT stylesheets. To use the implemented demo, follow these steps:

- Select the input language
- Check the CST option if wanted.

- Select the rules from a URL path, local machine or enter them directly in the source language text box
- Hit the translate button

By hitting the ‘Translate’ button, the framework’s demo first reads the input rules and shows them in the source language text box. Then it parses the rules, displays the parse tree (if selected) and puts the translation (to RIF-BLD) result in the target language text box.

To use the rule engine, follow the same steps. The rule engine parses the input rules (in RIF-BLD), infers new knowledge, and shows the result in the target language text box. The results include both the original rules plus the new inferred knowledge. If the CST option is selected, the rule engine shows the original rules in a tree structure and original plus inferred knowledge in another tree structure.

Figure 9.2 shows an example of translating POSL to RFI-BLD using the implemented demo with the CST tree option selected.

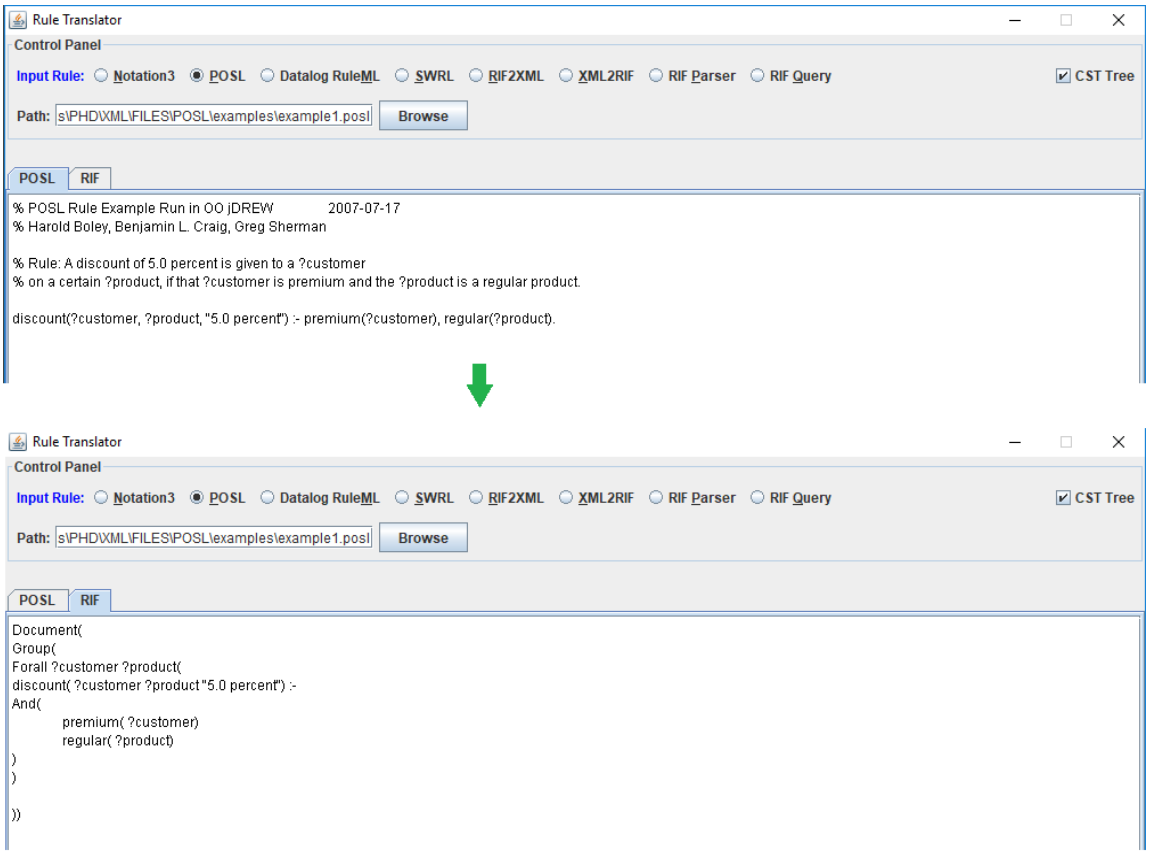

Figure 9.2. POSL to RIF-BLD translation using the implemented framework

Figure 9.3 shows the parse tree output of the POSL rules in Figure 9.2.

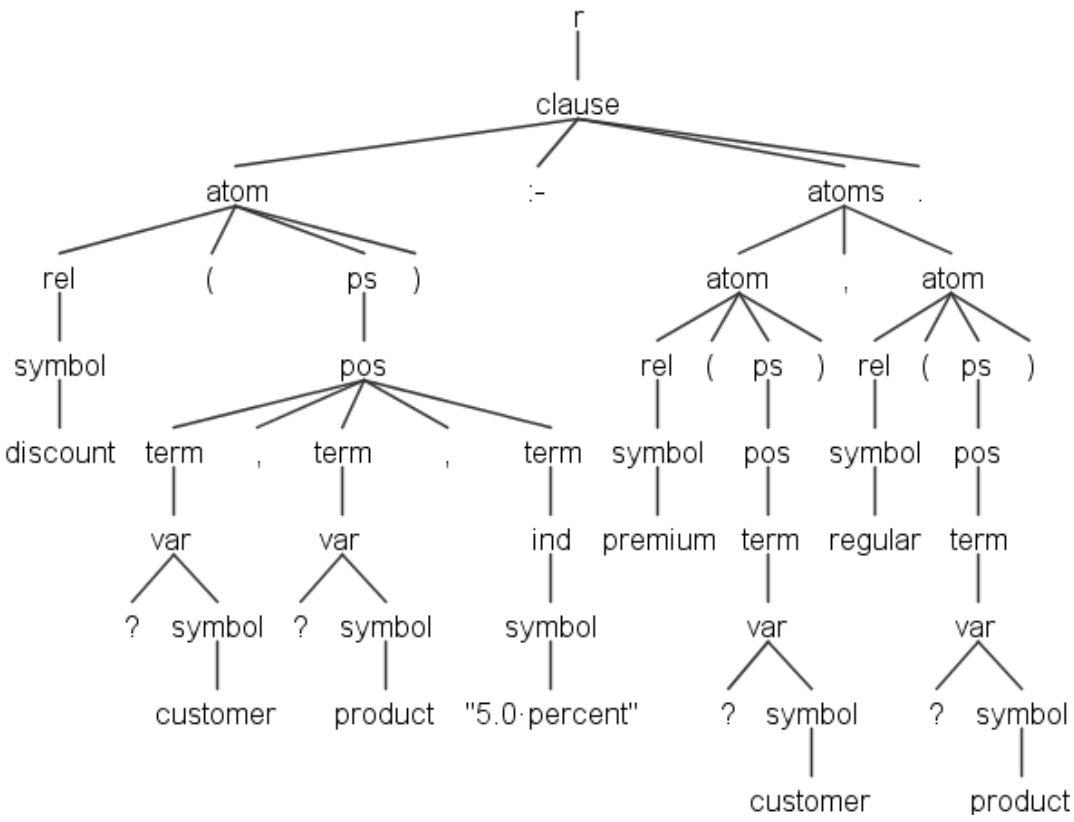

Figure 9.3. Concrete syntax Tree of the POSL rules in Figure 9.2

Figure 9.4 shows an example of using the rule engine in the framework's demo. The rule engine reasons over the input rules and puts the inferred knowledge in the target text box.

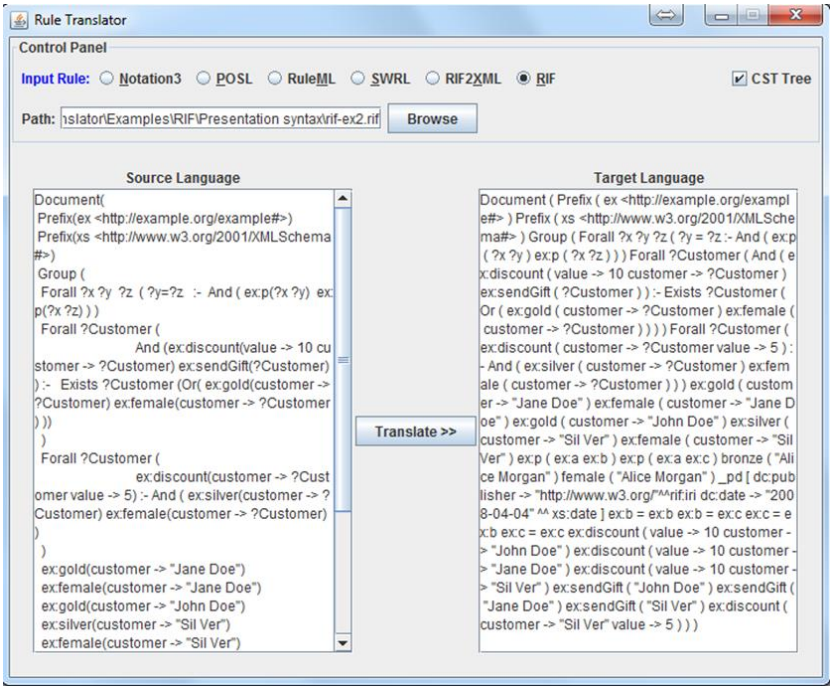

Figure 9.4. Using RIF rule engine in the implemented framework

Figure 9.5 shows a part of the CST tree of the new inferred knowledge. The original tab in Figure 9.5 can show the tree of the input rules.

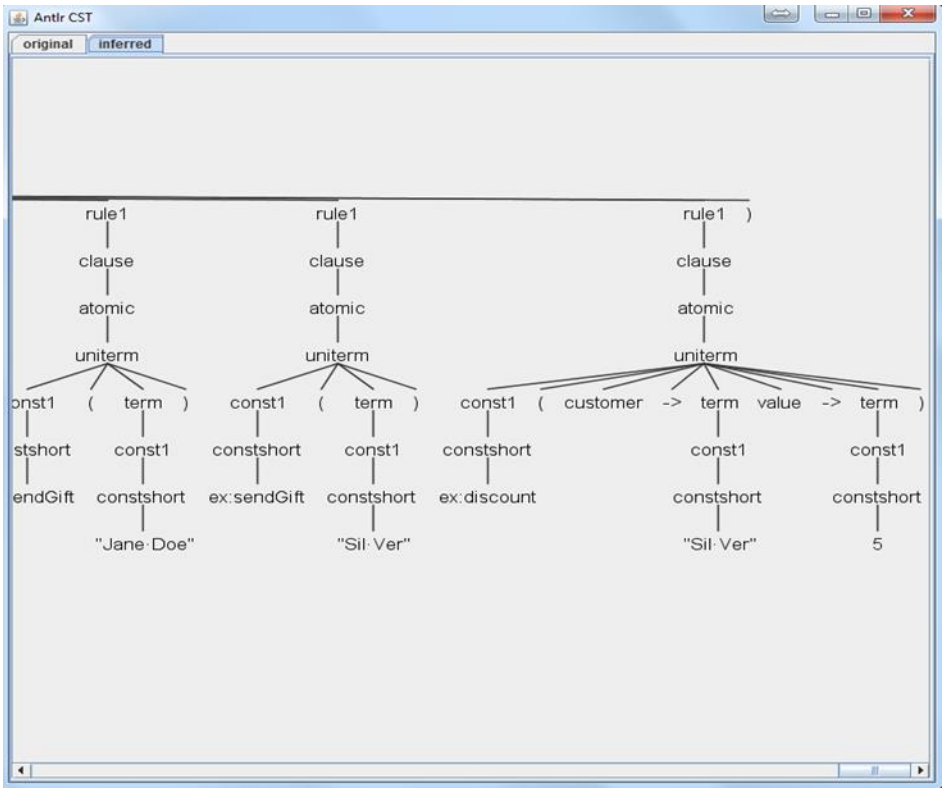

Figure 9.5. Rule engine inferred rules as a CST tree

To see more examples of rule translation (in N3, POSL, RuleML, SWRL, RIF languages) to RIF-BLD using the implemented demo refer to [65]. Next chapter presents experimental results and future works.

Chapter 10

Case Study

This chapter describes a case study of rules interchange. The case study is based on a challenge from Decision Management (DM) community. Decision Management community [66] is an initiative started in 2014 to facilitate sharing of knowledge concerning decision management. DM community provides a monthly challenge regarding decision modeling problems. Every challenge consists of a problem to be solved using business rules and decision management systems [66]. This case study is based on the DM 'challenge of March 2016' which consists of creating decision models from the structured text English of Port Clearance Rules.

10.1 Introduction to Port Clearance Rules and queries to the rules

Port Clearance Rules challenge targets a decision model that is capable to decide if a ship can enter a Dutch port on a certain date. The problem is inspired by the international Ship and Port Facility Security Code which originally was used as use case in the Game of Rules by the business Rules Platform Netherlands [67]. There are ten rules in this challenge. The English of each rule is moderately controlled, some having a structured 'if' part.

Here are the rules:

1. The hold of a ship must be considered clean if the hold does not contain remainders of cargo.
2. An unloaded ship may only enter a Dutch port if the ship complies with the requirements of the Inspection for unloaded ships.

3. A ship must comply with the requirements of the Inspection for unloaded ships if the ship complies with all of the following: a) the ship meets the safety requirements for unloaded ships; b) the ship has a certificate of registry that is valid.
4. A ship must be categorized as large if the total length of the ship is at least 80 meters.
5. A ship's hold contains remainders of cargo if the residual cargo measurement is higher than 0.5 mg dry weight per cm^2.
6. A ship only meets the safety requirements for unloaded ships if the ship complies with at least one of the following: a) the ship meets the safety requirements for small unloaded ships; b) the ship meets the safety requirements for large unloaded ships.
7. A ship only meets the safety requirements for large unloaded ships if the ship complies with all of the following: a) the ship is categorized as large; b) the hold of the ship is clean; c) the hold of the ship is double hulled.
8. A ship only meets the safety requirements for small unloaded ships if the ship complies with all of the following: a) the ship is categorized as small; b) the hold of the ship is clean.
9. A ship must be categorized as small if the total length of the ship is less than 80 meters.
10. A ship's certificate of registry must be considered valid if the date up to which the registration is valid of the certificate of registry is after the current date.

Figure 10.1 shows a general graph presentation of the above rules. An outgoing arrow shows what rules have to meet to satisfy its parent rule.

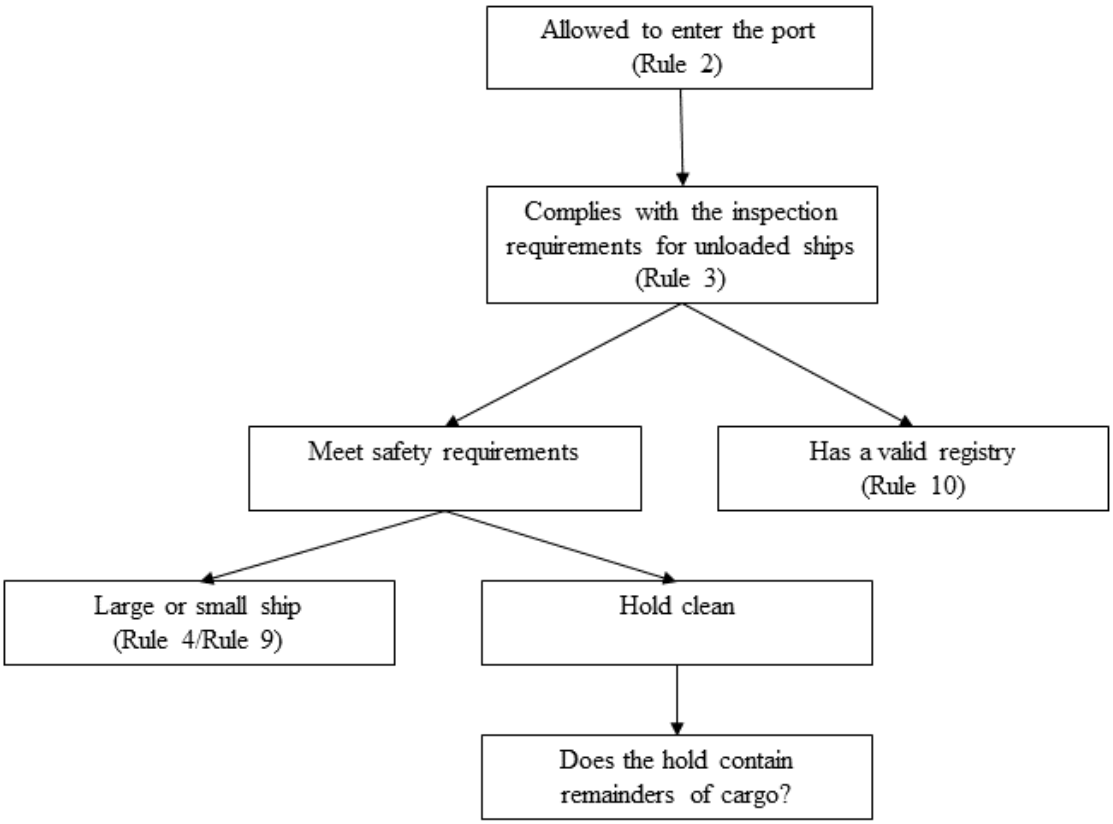

Figure 10.1. Graph presentation of the Port Clearance Rules

This chapter presents the Port Clearance Rules in N3, POSL, SWRL, and RuleML. Then they are translated to RIF and queried using RIF rule engine (each in a subsection of this chapter). Since the DM Challenge has introduced only ship rules, authors of [68] have developed ship facts for systematic testing of rules using PSOATransRun [69]. The present work also uses the facts introduced in [68] to test translations. Since the facts in [68] are in PSOA RuleML language [70], they are first translated to N3, POSL, SWRL and RuleML before using them. Therefore, in each section of this chapter, first Port Clearance Rules and facts are formalized in N3, POSL, SWRL or RuleML accordingly. Then, the acquired knowledge base (rules and facts) is translated to RIF language and finally queried using

RIF engine to test the formalization and translation process. At the end of this chapter, the formalization, translation and query results are compared in these languages.

The formalization process is done on 10 port clearance rules and 14 facts for 4 languages (N3, POSL, SWRL and RuleML). The translation process is done on 24 rules and facts from the above 4 languages to RIF. The translation is also done on 14 facts from PSOA RuleML to each of the N3, POSL, SWRL and RuleML languages since these facts defined in [68] are in PSOA RuleML. As it can be seen, since the formalization and translation processes generate a large amount of data, these data is not shown here and is available at appendix E.

10.2 N3 to RIF translation and reasoning

This section defines the Port Clearance Rules and facts in N3 language. The rules and facts can be defined with or without using frames in N3. For example, Rule #8 can be written in three different formats as shown below. Rule #8 in format one is a simple conjunction of atoms. Format two uses frames and format three uses nested frames. Lines start with '#' are comments and not part of the actual rules/facts.

```
#Rule 8: A ship only meets the safety requirements for small unloaded ships if the
ship #complies with all of the following:
#a) the ship is categorized as small;
#b) the hold of the ship is clean.
#Rule 8 (includes disjunct of original Rule 6)

#format one
       { ?s :MeetsSafetyRequirementsUnloaded log:Truth . }
       <=
       {
              ?s rdf:Type :ship .
              ?s  :size :small .
              ?s   :hold ?h1 .
                                  ?h1 rdf:Type :shipHold .
                                  ?h1  :status :clean .
       }.
```

```
#format two
       { ?s :MeetsSafetyRequirementsUnloaded log:Truth . }
       <=
       {
              ?s rdf:Type :ship ;
                 :size :small ;
                 :hold ?h1 .
                                ?h1  rdf:Type :shipHold ;
                                    :status :clean  .
       }.

#format three
       { ?s :MeetsSafetyRequirementsUnloaded log:Truth . }
       <=
       {
              ?s rdf:Type :ship ;
                 :size :small ;
                 :hold [ rdf:Type :shipHold ;  :status :clean] .
       }.
```

It is a choice of rule expert how to define the rules. The present work uses nested frames when possible (i.e. when language supports) to formalize the rules and facts. Appendix E shows the formalization of Port Clearance Rules in N3. As mentioned earlier, this formalization is not shown here to save space.

To be able to query the KB, some facts are also needed in addition to the rules. Since the "March 2016 DM Community Challenge" has introduced only ship rules (Port Clearance Rules), the ship facts introduced in [68] are borrowed here. These facts are written in PSOA RuleML [70]. Each ship fact is a frame having three slots, "registryExpirationDate", "totalLength", and "hold". The value of the "hold" slot itself is an embedded frame with three slots, "rdf:Type" (with "shipHold" as its value), "residualCargoMeasurement" and "hull". The embedded "hold" frame can be written in a non-frame fashion if wanted as well. Each ship fact comes with a comment on whether the ship instance should be allowed to enter a Dutch port, as of 2018-01-21, as well as the main reason [68]. The ship facts are defined in N3 and shown here. There are fifteen facts: Fourteen ship facts and a 'Date' fact.

‘Ship1’ fact is shown below. The complete definition of all the fifteen ship facts are presented in Appendix E.

```
# Ship 1 - No, registry has expired
:ship1 rdf:Type :ship ;
            :registryExpirationDate "2017-01-01"  ;
     :totalLength 20 ;
     :hold h1 .
h1  rdf:Type :ShipHold ;
     :residualCargoMeasurement 0.2 ;
     :hull :single  .
```

Now that both Port Clearance Rules and facts are defined in N3, they are translated to RIF using the framework. The translation of rule #8 and ship1 fact are presented below. The complete translation of all the port clearance rules and facts are shown in Appendix E. Since nested frames are not supported directly in RIF, the N3 rules and facts (in Appendix E) that are defined using nested frames, are unnested and slotributed when translating to RIF using Unnesting and Slotribution methods in [68].

```
Document(
     Base(<https://dmcommunity.org/>)
     Prefix(local  <https://dmcommunity.org/challenge/challenge-march-2016#>)
     Prefix(log <http://www.w3.org/2000/10/swap/log#>)
     Prefix(math <http://www.w3.org/2000/10/swap/math#>)
     Prefix(time <http://www.w3.org/2000/10/swap/time#>)
     Prefix(rdf <http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
     Group(

          Forall  ?s  ?x1 (
       local:MeetsSafetyRequirementsUnloaded( ?s )
       :-
       And(
          local:ship( ?s )
          local:size( ?s local:small )
          local:hold( ?s ?x1 )
          local:shipHold( ?x1 )
          local:status( ?x1 local:clean )
       )
         )

       local:ship( local:ship1 )
       local:registryExpirationDate( local:ship1 "2017-01-01" )
       local:totalLength( local:ship1 20 )
       local:hold( local:ship1 h1 )
       local:ShipHold( h1 )
```

```
            local:residualCargoMeasurement( h1 0.2 )
            local:hull( h1 local:single )

    )
)
```

The obtained knowledge base in RIF can be queried now. The following snapshots show the queries run on the KB. These queries are borrowed from [68]. Since the queries in [68] are in PSOA syntax, they are first defined in RIF before using them. The query translation, query run and query answers are obtained by the developed RIF engine. These queries include:

- ground queries using the top-level predicate MayEnterDutchPortUnloaded
- A non-ground query to :MayEnterDutchPortUnloaded
- Query over the KB in a bottom-up traversal manner. We start with the object-centered-query.
- Next query asks whether the hold ?h of ship7 is clean.
- Query that asks whether ship7 is a large ship.
- An extended query that asks whether ship7 is a large ship and its hold is clean and double hulled.
- We proceed to the relational-query portion of this traversal. Query that asks whether ship7 meets the safety requirements.
- Query that asks whether ship7 has a valid certificate.
- Query that asks whether ship7 complies with the requirements of the inspection for unloaded ships. It can be proved by Rule 3 based on the previous two (sub) queries.

- And finally, some non-ground queries that can extract interesting content from the KB.

In the following you see some of these queries. The ground queries that answer Port Clearance questions such as the top-level predicate "MayEnterDutchPortUnloaded" applied to specific ship instances, e.g. to ship1 and ship7, are shown below (these queries are borrowed from [68]):

```
local:MayEnterDutchPortUnloaded(local:ship1)
Yes

local:MayEnterDutchPortUnloaded(local:ship7)
No
```

A non-ground query to MayEnterDutchPortUnloaded, using output variable ?w to deduce ships that may enter a Dutch port along with its answers are shown below. This is a top-down deductive query.

```
local:MayEnterDutchPortUnloaded(?w)

?w=<http://psoa.ruleml.org/usecases/PortClearance#ship14>
?w=<http://psoa.ruleml.org/usecases/PortClearance#ship2>
?w=<http://psoa.ruleml.org/usecases/PortClearance#ship12>
?w=<http://psoa.ruleml.org/usecases/PortClearance#ship7>
?w=<http://psoa.ruleml.org/usecases/PortClearance#ship4>
```

The following queries are explored over the KB in a bottom-up traversal manner. We start with the object-centered-query portion of this traversal. The following query asks whether h7 is a ship hold and it is clean, which can be proved by Rule 1&5 and the h7 frame embedded inside the ship7 fact.

```
local:status( h7 local:clean )
Yes
```

Next query asks whether the hold ?h of ship7 is clean. This can be proved by first binding ?h to the hold h7 of :ship7 and then check whether h7 is clean using the previous (sub)query.

```
local:ship( local:ship7 )
local:hold( local:ship7 ?h )
local:status( ?h local:clean )
```

The next query asks whether ship7 is a large ship, which can be proved by Rule 4 using its totalLength in the ship7 fact.

```
local:size( ship7 local:large )
Yes
```

An extended query then asks whether ship7 is a large ship and its hold is clean and double hulled. The size and hold status of ship7 have been proved by the previous two (sub) queries while the hold-hull information can be proved directly through the ship7 fact [68].

```
local:size( ship7 local:large )
local:hold( local:ship7 ?h )
local:status( ?h local:clean )
local:hull( ?h local:double )

?h=<http://psoa.ruleml.org/usecases/PortClearance#h7>
```

We now proceed to the relational-query portion of this traversal. The following query asks whether ship7 meets the safety requirements. It can be proven by Rule 7 and the previous (sub) query [68].

```
MeetsSafetyRequirementsUnloaded(ship7)
Yes
```

The next query asks whether ship7 has a valid certificate, which can be proved by Rule 10 based on its registryExpirationDate slot in a fact.

```
HasValidCertificate(ship7)
Yes)
```

The next query of this traversal asks whether ship7 complies with the requirements of the inspection for unloaded ships. It can be proved by Rule 3 based on the previous two (sub) queries.

```
CompliesInspectionRequirementsUnloaded(ship7)
Yes
```

The top-level query MayEnterDutchPortUnloaded(ship7) can now be proved by Rule 2 and the previous (sub)query. Next, some non-ground queries that can extract interesting content from the KB. The following query asks for the size of ship1 [68].

```
local:size( ship1 ?z )
?z=<http://psoa.ruleml.org/usecases/PortClearance#small>
```

The subsequent query asks for any large ship whose hold is clean.

```
local:size( ?s local:large )
local:hold( ?s ?h )
local:status( ?h local:clean )

?s=<http://psoa.ruleml.org/usecases/PortClearance#ship7>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship13>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship10>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship14>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship6>
```

The next query asks for any ship that is large and has a valid certificate.

```
local:HasValidCertificate(?s)
local:size( ?s local:large )

?s=<http://psoa.ruleml.org/usecases/PortClearance#ship5>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship14>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship6>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship9>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship13>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship7>
```

The final query asks for any ship that is small and meets the safety requirements for unloaded ships.

```
local:MeetsSafetyRequirementsUnloaded(?s)
local:size( ?s local:small )

?s=<http://psoa.ruleml.org/usecases/PortClearance#ship4>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship1>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship12>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship2>
```

10.3 POSL to RIF translation and reasoning

This section defines the Port Clearance rules and facts in POSL. The obtained KB is then translated to RIF and queried using the same queries in 10.2 (which are borrowed from

[68]). In POSL, same as in N3 (in section 10.2), Port Clearance Rules and facts can be written using a conjunction of unit atoms or using (nested) frames. In comparison to N3, frames in POSL should be used as a predicate's (atom's) arguments.

Moreover, in POSL, a variable cannot be used as a frame's name or a slot's key (key -> value). Therefore, this should be considered while defining non-ground rules using frames. For example, see rule #8 and ship1 fact below. The complete formalization of Port Clearance rules and facts defined in POSL are shown in Appendix E.

```
%Rule 8: A ship only meets the safety requirements for small unloaded ships if the ship complies
with all of the following:
%a) the ship is categorized as small;
%b) the hold of the ship is clean.
%Rule 8 (includes disjunct of original Rule 6)

MeetsSafetyRequirementsUnloaded(?s) :-
size(?s:ship , "small") , hold(?s:ship , ?h:shipHold) ,
status(?h:shipHold , "clean").

%Ship facts (No or Yes refer to answers for queries  with :ship1, :ship2, ... as arguments)
/% the two above facts about ship1 (and the other ships here) can also be written as below: %/

ship(ship1[
          registryExpirationDate -> "2017-01-01"  ;
       totalLength -> 20 ;
       hold->h1[
              residualCargoMeasurement -> 0.2 ;
              hull -> "single" ]
]).
ShipHold(h1).
```

The translation of Port Clearance Rules and facts from POSL to RIF is done using the developed framework and is presented in Appendix E. the translation of rule #8 and ship1 fact are shown below. Since nested frames are not supported directly in RIF, rules and facts in POSL are unnested and slotributed when translating to RIF using Unnesting and Slotribution methods in [68].

```
Document(
  Group(

       Forall ?s ?h(
           MeetsSafetyRequirementsUnloaded( ?s) :-
            And(
          size( ship(?s) "small")
          hold( ship(?s) shipHold(?h))
          status( shipHold(?h) "clean")
             )
       )

      ship(ship1)
      registryExpirationDate(ship1 "2017-01-01")
      totalLength (ship1 20)
      hold(ship1 h1)
      shipHold(h1)

      residualCargoMeasurement (h1 0.2)
      hull(h1 "single")

  )
)
```

Now that the Port Clearance rules and facts in POSL are translated to RIF, the obtained knowledge base can be queried. The same queries in 10.2 are used here as well. The following shows the queries #3 and #14 of fourteen queries run on the KB. Appendix E shows the complete list of these queries.

```
//Query 3
local:MayEnterDutchPortUnloaded(?w)

//Query 14
local:MeetsSafetyRequirementsUnloaded(?s)
local:size( ?s local:small )
```

The query answers are obtained by the developed RIF engine. Since the queries return the same answers using RIF engine as in section 10.2, these answers are not shown here to save space.

10.4 SWRL to RIF translation and reasoning

This section defines the Port Clearance Rules and facts in SWRL XML. The obtained KB is then translated to RIF XML. Finally, it is translated from RIF XML to RIF presentation

syntax to be able to query the KB using the RIF engine. The KB is queried using the same queries in 10.2 and 10.3.

A predicate in SWRL can be defined using <swrlx:individualPropertyAtom> or <swrlx:datavaluedPropertyAtom> and the predicate arguments can be defined as the children of these tags. To define a type or class membership, <owlx:Class> can be used. Frames and nested frames are also supported in SWRL. <owlx:Individual> and <owlx:ObjectPropertyValue> can be used to define frames and nested frames. The following shows rule #8 and ship1 fact formalization in SWRL. To see a complete definition of Port Clearance rules and facts refer to Appendix E.

```
<-- Rule 8 --->

<ruleml:imp>
  <ruleml:_rlab ruleml:href="#rule8"/>
  <ruleml:_body>
      <swrlx:classAtom>
        <owlx:Class owlx:name="ship" />
        <ruleml:var>s</ruleml:var>
    </swrlx:classAtom>
    <swrlx:datavaluedPropertyAtom  swrlx:property="size">
      <ruleml:var>s</ruleml:var>
      <owlx:DataValue
owlx:datatype="&xsd;string">small</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
     <swrlx:classAtom>
        <owlx:Class owlx:name="shipHold" />
        <ruleml:var>h</ruleml:var>
    </swrlx:classAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hold">
      <ruleml:var>s</ruleml:var>
      <ruleml:var>h</ruleml:var>
      </swrlx:individualPropertyAtom>
      <swrlx:datavaluedPropertyAtom  swrlx:property="status">

        <ruleml:var>h</ruleml:var>
        <owlx:DataValue
owlx:datatype="&xsd;string">clean</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
```

```
    <swrlx:individualPropertyAtom
swrlx:property="MeetsSafetyRequirementsUnloaded">
      <ruleml:var>s</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>

<!—ship1 fact -->

<!--
# Ship 1 - No, registry has expired
:ship1 rdf:Type :ship ;
          :registryExpirationDate "2017-01-01"  ;
        :totalLength "20" ;
        :hold [
              rdf:Type :ShipHold ;
              :residualCargoMeasurement "0.2" ;
              :hull :single ] .
-->
<owlx:Individual owlx:name="ship1">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue
owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2017-
01-01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue
owlx:datatype="&xsd;int">20</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue
owlx:property="residualCargoMeasurement">
        <owlx:DataValue
owlx:datatype="&xsd;double">0.2</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue
owlx:datatype="&xsd;string">single</owlx:DataValue>
      </owlx:DataPropertyValue>

    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>
```

The snapshot below shows rule #8 and ship1 fact translated from SWRL to RIF XML. Appendix E contains the complete translation of Port Clearance Rules and facts from SWRL to RIF XML.

```
<-- Rule 8 --->

<Implies>
  <if>
       <And>
              <formula>
                     <formula>
                            <Member>
                                   <class>
                                          <Const
type="http://www.w3.org/2007/rif#iri">ship</Const>
                                   </class>
                                   <instance>
                                          <Var>s</Var>
                                   </instance>
                            </Member>
                     </formula>
              </formula>
              <formula>
                     <Atom>
                            <op>
                                   <Const
type="http://www.w3.org/2007/rif#iri">size</Const>
                            </op>
                            <args ordered="yes">
                                   <Var>s</Var>
                                   <Const
type="http://www.w3.org/2001/XMLSchema#string">small</Const>
                            </args>
                     </Atom>
              </formula>
              <formula>
                     <formula>
                            <Member>
                                   <class>
                                          <Const
type="http://www.w3.org/2007/rif#iri">shipHold</Const>
                                   </class>

                                   <instance>
                                          <Var>h</Var>
                                   </instance>
                            </Member>
                     </formula>
              </formula>
```

```
            <formula>
                  <Atom>
                        <op>
                              <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                        </op>
                        <args ordered="yes">
                              <Var>s</Var>
                              <Var>h</Var>
                        </args>
                  </Atom>
            </formula>
            <formula>
                  <Atom>
                        <op>
                              <Const
type="http://www.w3.org/2007/rif#iri">status</Const>
                        </op>
                        <args ordered="yes">
                              <Var>h</Var>
                              <Const
type="http://www.w3.org/2001/XMLSchema#string">clean</Const>
                        </args>
                  </Atom>
            </formula>
      </And>
  </if>
  <then>
      <Atom>
            <op>
                  <Const
type="http://www.w3.org/2007/rif#iri">MeetsSafetyRequirementsUnloaded</Const
>
            </op>
            <args ordered="yes">
                  <Var>s</Var>
            </args>
      </Atom>
  </then>
</Implies>

<!–ship1 fact -->

<Atom>
      <op>
            <Const type="http://www.w3.org/2007/rif#iri">Ship</Const>

      </op>
      <slot ordered="yes">
            <Const type="http://www.w3.org/2007/rif#iri">ship1</Const>
            <slot ordered="yes">
                  <Const
```

```
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                    <Const type="http://www.w3.org/2007/rif#iri">2017-01-
01</Const>
              </slot>
              <slot ordered="yes">
                    <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                    <Const type="http://www.w3.org/2007/rif#iri">20</Const>
              </slot>
              <slot ordered="yes">
                    <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                    <Const type="http://www.w3.org/2007/rif#iri">h1</Const>
                    <slot ordered="yes">
                          <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                          <Const
type="http://www.w3.org/2007/rif#iri">0.2</Const>
                    </slot>
                    <slot ordered="yes">
                          <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                          <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                    </slot>
              </slot>
       </slot>
</Atom>
```

After defining Port Clearance rules and facts in SWRL and translated to RIF XML, they are translated one more time from RIF XML to RIF presentation syntax using the XML2RIF component of the developed framework. After these steps, the obtained KB can be queried using the RIF engine. To prevent redundancy and save space, the KB in RIF presentation is not shown. It can be seen in section 10.2 if needed. The same queries in 10.2 and 10.3 are used here as well. Since the query answers obtained by the RIF engine are the same as in sections 10.2 and 10.3, they are not shown here. These answers can be found in Appendix E.

10.5 RuleML to RIF translation and reasoning

This section defines the Port Clearance rules and facts in Binary Datalog RuleML. The obtained KB is in XML format. It is translated to RIF XML and then to RIF presentation syntax using the “RuleML to RIF” and “RIF XML to RIF presentation” components of the framework. After that, the KB is queried using the same queries used in the previous sections of this chapter.

Port Clearance rules and facts in RuleML can be formalized using a conjunction of unit atoms or using (nested) frames. For example, the following definition uses nested frames to define ship1 fact:

```
<Atom>
  <op>
       <Rel>Ship</Rel>
  </op>
  <slot>
       <Ind type="dm:ship">ship1</Ind>
       <Plex>
              <slot>
                     <Ind type="xsd:string">registryExpirationDate</Ind>
                     <Ind type="xsd:dateTime">2017-01-01</Ind>
              </slot>
              <slot>
                     <Ind type="xsd:string">totalLength</Ind>
                     <Data xsi:type="xs:int">20</Data>
              </slot>
              <slot>
                     <Ind type="xsd:string">hold</Ind>
                     <Plex>
                            <Ind type="dm:shipHold">h1</Ind>
                            <slot>
                                   <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                   <Ind type="xsd:double">0.2</Ind>
                            </slot>
                            <slot>
                                   <Ind type="xsd:string">hull</Ind>
                                   <Ind type="xsd:string">single</Ind>
                            </slot>
                     </Plex>
```

```
            </slot>
      </Plex>
  </slot>
</Atom>
```

As it is shown in the fact above, frames and nested frames in Datalog RuleML can be defined using <slot> and <Plex> tags. In comparison to N3, frames in datalog RuleML (similar to POSL) should be used as a predicate's (atom's) argument. The following shows the formalization of rule #8.

```
<!-- %Rule 8: A ship only meets the safety requirements for small unloaded
      ships if the ship complies with all of the following: %a) the ship is
categorized
      as small; %b) the hold of the ship is clean. %Rule 8 (includes
disjunct of
      original Rule 6) MeetsSafetyRequirementsUnloaded(?s) :- size(?s:ship
, "small")
      , hold(?s:ship , ?h:shipHold) , status(?h:shipHold , "clean"). -->

<Implies>
  <then>
      <Atom>
            <op>
                  <Rel>MeetsSafetyRequirementsUnloaded</Rel>
            </op>
            <Var>s</Var>
      </Atom>
  </then>
  <if>
      <And>
            <Atom>
                  <op>
                        <Rel>size</Rel>
                  </op>
                  <Var type="dm:ship">s</Var>

                  <Ind type="xsd:string">small</Ind>
            </Atom>
            <Atom>
```

```
                    <op>
                          <Rel>hold</Rel>
                    </op>
                    <Var type="dm:ship">s</Var>
                    <Var type="dm:shipHold">h</Var>
              </Atom>
              <Atom>
                    <op>
                          <Rel>status</Rel>
                    </op>
                    <Var type="dm:shipHold">h</Var>
                    <Ind type="xsd:string">clean</Ind>
              </Atom>
        </And>
  </if>
</Implies>
```

After formalization of Port Clearance rules and facts in RuleML they are translated to RIF XML and then to RIF presentation syntax which both translations can be seen in Appendix E. To prevent redundancy and save space the result KB (in RIF XML and presentation syntax) are not presented here.

10.6 Comparisons

To define the Port Clearance Rules and facts in each of the N3, POSL, SWRL and RuleML languages as well as translating them to RIF, their limitations and differences have been observed in this section. Since there are no facts defined in the Port Clearance Rules challenge, facts from [68] are borrowed to then be able to query the obtained KB. This section compares the four languages above based on how they can formalize Port Clearance Rules and facts and show their differences and limitations. This comparison is mostly evolve around how or if each of the languages above presents: Truth values, predicates (unary, binary, n-ary), nested predicates, frames, nested frames, built-ins and etc.

N3:

As it is known and mentioned before, every statement in N3 is in the form of subject-predicate-object. Therefore, to show a unary predicate in the form “pred(arg)” in N3, a third argument is needed. In this case, “true” value is used to fill the object place in subject-predicate-object statement. Hence, the unary predicate “pred(arg)” is defined as “arg-pred-true”. For example, see definition of rule 2 of Port Clearance rules in section 10.2 below:

```
{ ?s :MayEnterDutchPortUnloaded log:Truth } <=
{ ?s :CompliesInspectionRequirementsUnloaded log:Truth } .
```

Binary predicates pred(arg1, arg2), however, can be presented as ‘arg1 pred arg2.’ triple. N-ary predicates can be defined using nested statements.

Predicate arguments in N3 can be atoms, other predicates and frames. Frames and nested frames are also supported in N3. Comma and semicolon are used to define a frame. Comma ‘,’ shows the repetition of another object for the same subject and predicate. Semicolon ‘;’ indicates the repetition of another predicate for the same subject. The following example shows a frame definition:

```
:Alex :brother :Jack, John ; :sister :Sarah.
```

Which is a formalization of the frame below (in a hypothetical language):

```
Alex[ brother -> jack ; brother -> John ; sister -> Sarah ]
```

Frames can also be defined using blank nodes in N3. Blank nodes are shown using brackets ‘[‘ and ‘]’. You can refer to chapter 5 for an example.

The definitions of built-in functions and truth values (truth and false) in N3 are not clear. There are no defined library or formal definitions for them. However, “log” and “math” are used as libraries in [8] to present truth values and some mathematic functions. These

prefixes are also used here in Port Clearance Rules formalization when needed (see the example above). Type or class membership is also supported in N3 and is treated as unary predicate. For example, in formalizing Port Clearance Rules and facts ‘rdf:Type’ is used to define the type of a ship. Table 10.1 shows the N3 features compare to other four languages.

POSL:

Every clause in POSL is an atom or conjunction of atoms. Atoms in POSL are predicates with a mixture of positional and slotted arguments. Therefore, unary, binary and n-ary predicates are supported in POSL as were in N3. Since every clause in POSL is in the form of predicates, therefore, a frame cannot be defined standalone (e.g. as a ground fact). It can be defined as an argument of a predicate (atom). In this case, the predicate’s name can be a type or class membership. For example, ship1 fact from [68] can be defined as an argument for the ‘ship’ predicate. See the snapshot below. Ship1 is defined using nested frames.

```
ship(ship1[
          registryExpirationDate -> "2017-01-01"  ;
       totalLength -> 20 ;
       hold->h1[
           type -> ShipHold ;
              residualCargoMeasurement -> 0.2 ;
              hull -> "single" ]
]).
```

In N3, the same fact for ship1 is defined as follows. As it can be seen this fact (in the form of nested frames) is standalone compare to its definition in POSL.

```
:ship1 rdf:Type :ship ;
          :registryExpirationDate "2017-01-01"  ;
          :totalLength 20 ;
          :hold h1 .
     h1 rdf:Type :ShipHold ;
          :residualCargoMeasurement 0.2 ;
          :hull :single  .
```

In addition, in POSL, a variable cannot be used as a frame name or a slot key (key -> value). Therefore, this was considered while defining non-ground rules using frames (of Port Clearance Rules and facts). It seems that there is no formal library definition in POSL. Also, POSL does not have built-in libraries (e.g. numeric/string functions). However, it does support XML Schema data types. See table 10.1.

SWRL:

Rules in SWRL are in the form of an antecedent-consequent pair. Predicates (unary, binary) are supported in SWRL. A predicate in SWRL can be defined using <swrlx:individualPropertyAtom> or <swrlx:datavaluedPropertyAtom> and the predicate arguments can be defined as the children of these tags. Therefore, definition of Port Clearance rules and facts using predicates can be easily done in SWRL XML.

To define a type or class membership, <owlx:Class> is used.

Built-in functions are also supported in SWRL using the <swrlx:builtinAtom> tag and "swrlx:builtin" library. From above, we can conclude that SWRL supports predicates, frames, and built-ins and therefore has no trouble in defining Port clearance rules.

RuleML:

Unary, binary and n-ary predicates are supported in Datalog RuleML and are defined using atoms. Predicate arguments can be variables and individuals. Frames and nested frames are

also supported in Datalog RuleML and can be defined using <slot> and <Plex> tags. Therefore, ships and ship hulls frames in Port Clearance facts can be defined using these tags. In comparison to N3, frames in Datalog RuleML (similar to POSL) should be used as a predicate's argument and cannot be defined stand alone.

It should be noted that since nested frames are supported in N3 but not in RIF, this should be considered while translating Port Clearance Rules and facts from N3 to RIF. Therefore, in this work, ship facts that are defined using nested frames in N3, are first unnested and slotributed when translating to RIF using Unnesting and Slotribution methods in [68].

Table 10.1 compares the formalization of Port Clearance Rules in N3, POSL, SWRL, and RuleML as well as their translation to RIF. Table 10.1 shows rule language elements that are supported in each language when defining Port Clearance rules and facts and their translation to RIF.

Table 10.1. Comparison of rule languages in regard to Port Clearance rules and facts

	N3	POSL	SWRL	RuleML	RIF
Unary predicate	Not directly	Yes	Yes	Yes	Yes
Binary/n-ary predicate	Indirectly	Yes	Binary	Yes	Yes
Predicate arguments	Atom, predicate, frame	Positional, slotted, positional + slotted	Yes	Yes	Yes
Frame support	Yes	As predicate argument	No	As predicate argument	Yes
Nested frame	Yes	As predicate argument	No	As predicate argument	As predicate argument
Truth values	not clear	not clear	From OWL	Yes	Yes
Built-in functions	Partly	XML Schema data types	swrlx:builtin	XML Schema data types	Yes
Class membership or type	rdf:Type	Yes	Yes	Type attribute	Yes

As it can be seen, Port Clearance Rules and facts can be formalized using any of the above five languages with putting their features and limitations into consideration. Also, there are

situations where the formalization in one language cannot be directly translated to RIF. There are situations where the rules or facts should be simplified using methods such as Unnesting and Slotribution (in [68]) first, before being able to translate them to their equivalents in RIF.

A language such as RuleML has XML syntax does not have a presentation syntax. In these cases the Port Clearance Rules and facts are first translated to RIF XML and then from RIF XML to RIF presentation syntax using XSLT libraries defined in the framework.

Every rule language has its own strengths and limitations. Rule Interchange Format (RIF) is a W3C standard to facilitate exchanging rules between rule systems. The idea of RIF is to provide mechanisms for rule systems to be able to map rules in their native language to RIF and back. This idea presents a standard method for rule translation and sharing. The main goal of this thesis is to help researches to feel and better understand challenges that arise for rule language interchange task. This chapter uses the Decision Management community challenge 2016 as a case study to show limitations, restrictions and other aspects of rule interchange when formalizing or translating knowledge bases from N3, POSL, SWRL and RuleML languages to RIF language.

Chapter 11

Evaluation

Each rule translator in the framework has been applied to different use cases in its rule language format (N3, POSL, SWRL, etc.). Since there is not an accumulated rule repository for most of these rule languages, the author has gathered up different use cases for each language. The main place to find use cases and rule examples for these languages (SWRL, POSL, N3, RuleML and RIF) is their official website [6-8,18,14].

The following tables in this chapter show the rule translation results using the implemented framework. The evaluation criterion is the correctness and completeness of the translation. The translation results have been empirically validated. All of the expected answers were produced and no extra answer were produced that was not expected. There will be situations where the translation cannot be complete. For example, where the translation cannot be done because the target language (RIF-BLD) does not support a specific feature from the source language without losing information e.g. 'swrlx:differentIndividualsAtom' in SWRL (that means two objects are different), is not available in RIF-BLD.

Each table below shows the result of translating one of the framework languages to RIF-BLD language. In each table, the '# of rules/facts' column shows the number of rules and facts in that rule base excluding the metadata (e.g. prefix, base, comments, etc.). Also, the nested rules or facts are counted as one rule or fact.

Table 11.1 shows the results of applying the Notation3 rule translator to various N3 rules. More examples of Notation3 rules can be found on the Web (e.g. [8]). Also, for more examples of translating Notation3 rules to RIF using the N3 translator refer to [65].

Table 11.1. Notation 3 rule translation results

Rule Base	# of rules/facts	Translated	Rule Base URL
White.n3	8	OK	https://www.w3.org/2000/10/swap/Examples.html
gedcom-facts.n3	37	OK	
rules12.n3	4	OK	
Rules13.n3	6	OK	

Table 11.2 show the results of applying the POSL translator to different POSL rule bases.

Table 11.2. POSL rule translation results

Rule Base	# of rules/facts	Translated	Rule Base URL
RuleML2007.posl	42	OK	https://github.com/OOjDREW/oojdrew-website
paperSubmital.pols	18	OK	
programChairAgent2.posl	7	OK	
Beverage.posl	20	OK	

Table 11.3 shows the results of applying the SWRL translator to some SWRL rules. SWRL rules used are in XML-based format. The SWRL translator is an XSLT stylesheet. SWRL stylesheet translator is called ‘SWRL2RIF.xsl’ and it is a prototype and it does not fully support SWRL translation. It can be used as a starting point to translate SWRL rules (in XML format) to rules RIF-BLD language. Rules and facts in SWRL XML are always surrounded by ‘<swrlx:Ontology>’ tag since SWRL rules can be part of OWL ontologies in OWL.

Table 11.3. SWRL rule translation results

Rule Base	# of rules/facts	Translated	Rule Base URL
example5.1-3.swlrx	1	OK	https://www.w3.org/Submission/SWRL/
example5.1-4.swlrx	12	?	
example5.1-6.swlrx	1	OK	

Question mark '?' in Table 11.3 indicates that some parts of the rules/facts in the 'example5.1-4' have been ignored since the SWRL translator cannot fully translate SWRL rules to RIF-BLD without losing information.

Table 11.4 show the translation results of Unary/Binary Datalog RuleML to RIF. The translator is called 'UBDatalogRuleML2RIF_BLD.xsl' and is an XSLT stylesheet. This RuleML translator, same as SWRL stylesheet translator, is a prototype and it does not fully support RuleML translation.

Table 11.4. Unary/Binary Datalog RuleML translation results

Rule Base	# of rules/facts	Translated	Rule Base URL
recommend.ruleml	2	OK	http://deliberation.ruleml.org/1.02/exa/
equivalent.ruleml	2	OK	
reify-bindatagroundfact.ruleml	4	OK	

Table 11.5 shows the experimental results of the developed rule engine. The rule engine is applied to different rule categories in RIF. The use cases in Table 11.5 are from W3C RIF working group at [71].

Table 11.5. Rule Engine reasoning results on various RIF rules

Rule title	Rule category	Reasoning Results	RIF2XML Results
Frames	RIF-Core	OK	OK
Positional Arguments	RIF-Core	OK	OK
Class Membership	RIF-BLD	OK	OK
Equality in conclusion	RIF-BLD	OK	OK
Named Arguments	RIF-BLD	OK	OK

The rule engine infers the expected new knowledge (e.g. new facts, and rule consequents) from these rule bases. The rule engine can show the results directly in the GUI of the implemented demo, output them to a file and/or display them in a tree structure format. The rule engine currently does not support reasoning over List, Import and built-ins in RIF. They are part of the future work.

One more component of the framework is the RIF to XML translator which maps RIF-BLD rules in presentation syntax to their equivalent rules in XML-based syntax. The RIF2XML translator has been applied to the same rule examples as the RIF engine. The last column of the Table 11.5 shows this translator evaluation results and it indicates that the translator is sound and complete. For space preserving purposes, the actual translation results to XML are not shown here. They can be found at [65]. To see more use cases and evaluation of N3, POSL, SWRL, RuleML translation to RIF-BLD, RIF-BLD presentation syntax to XML-based syntax translation, as well as the rule engine evaluation refer to [65]. It should be mentioned here that the developed prototype of the framework is in its early stages and still improvements and debugging are required.

Chapter 12

Conclusions and Future Works

12.1 Conclusions

Web rule languages have been introduced for the Web-based interchange of business policies, business rules and any business or application logic that can be presented in the form of rules. Web rule languages provide machine interpretation, automated processing, interchange and translation of rules (usually presented in XML-based syntax) on the Web. Every rule language has its own syntax and semantics and every rule system has its own version of a rule language. This situation makes sharing of rules among rule systems and business owners, which use those rule systems, a cumbersome and time-consuming task.

Therefore, the first step in rule interchange and to make interoperability between rule languages possible is to know the type of the rule language (normative, deductive, and reactive), dimensions and limitations of the language, and finally the syntax and semantics of a rule language. The Rule Interchange Format (RIF) is a standard for exchanging rules among rule systems, in particular among Web rule languages and engines. RIF focuses on rule exchange rather than trying to develop a single one-fits-all rule language. Rule mappings are required to be semantics-preserving, therefore the rules can be exchanged from one language to another without losing meanings.

This dissertation proposes a framework that makes rule interchange between some of the popular rule markup languages possible. The framework uses RIF-BLD language as an intermediary to facilitate rule interchange. The framework attempts to interchange rules

between Notation3 (N3), POSL, SWRL, Unary/Binary Datalog RuleML and RIF-BLD languages. A rule editor, parser, visualizer, and translator are developed for N3, POSL and RIF-BLD languages. The N3 and POSL translators exchange rules between N3/POSL and RIF-BLD. The RIF-BLD translator maps RIF-BLD rules in presentation syntax to XML-based syntax. SWRL and RuleML translators in the framework are XLST stylesheets. They translate SWRL/RuleML XML-based rules to RIF-BLD rules. The framework also comprises a RIF-BLD rule engine. The developed rule engine is both a forward-chaining as well as a backward chaining inference engine. It can answer queries as well as infer new knowledge from the rule base. The RIF rule engine presented in this dissertation is the first rule engine developed for the RIF language. Having an inference engine for the RIF that its main goal is to facilitate rule interchange and sharing is a big step in this path.

All the rule translators plus the RIF rule engine included in the framework are parts of the contributions made in this dissertation. It should be mention that all the components of the framework are in their early stages and need improvements. The framework's prototype development is still in progress.

12.2 Future work

The framework purpose is to facilitate the rule interchange between Notation3, POSL, SWRL, RuleML and RIF rule languages. As future works, some updates need to be done on the framework. The SWRL and Unary/Binary Datalog RuleML stylesheet translators need to be updated. These translators are in their early phase and do not completely cover SWRL and RuleML to RIF translation. There are situations specifically in SWRL language that the translation is not feasible. Currently, the rule engine does not support List, Import,

and Built-ins in RIF-BLD. Adding these functionalities to the rule engine are parts of the future works. The rule engine does not have any memory management or optimization. These functionalities can be added to it. All the translators may be error prone and need debugging. Also, error handling is weak and needs some tuning. Finally, adding a syntax highlighter (for human-readable rule languages in the framework e.g. N3 and POSL) and XML tag coloring will make rule editing, writing, and debugging easier and more user-friendly. These functionalities are also a part of the future works.

Bibliography

[1] R. G. Ross and G. Ronald, *Principles of the business rule approach*. Addison-Wesley Professional, 2003.

[2] B. Von Halle and L. Goldberg, "The Business Rule Revolution. Happy About," *Silicon Val.*, 2006.

[3] A. Paschke and H. Boley, "Rule markup languages and semantic web rule languages," *Rule Markup Lang. Semant. Web Rule Lang.*, pp. 1–24, 2010.

[4] Kifer, Michael. "RIF: The Rule Interchange Format." Encyclopedia of Social Network Analysis and Mining (2014): 1586-1588.

[5] Athan, Tara, Harold Boley, and Adrian Paschke. "RuleML 1.02: Deliberation, Reaction and Consumer Families." Challenge+ DC@ RuleML. 2015.

[6] SWRL, A. "Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, 21 May 2004." Accessed February (2016).

[7] H. Boley, "POSL: an integrated positional-slotted language for Semantic Web knowledge," *RuleML Work. Draft. May*, 2004.

[8] T. Berners-Lee and D. Connolly, "Notation3 (N3): A readable RDF syntax," *W3C*, 2011. [Online]. Available: http://www.w3.org/TeamSubmission/n3/.

[9] Salatino, Mauricio, Mariano De Maio, and Esteban Aliverti. Mastering JBoss Drools 6. Packt Publishing Ltd, 2016.

[10] E. Friedman-Hill and others, "Jess, the rule engine for the java platform." 2008.

[11] Y. Biletskiy, "A Framework for Web-based Interoperation among Business Rules.," in *CSWS*, 2013, pp. 24–27.

[12] "Rule Language Dialects." [Online]. Available: http://www.w3.org/2005/rules/wg/wiki/RuleLanguageDialects.html. [Accessed: 29-Mar-2017].

[13] "Classification of Rules." [Online]. Available: http://www.w3.org/2005/rules/wg/wiki/Classification_of_Rules.html. [Accessed: 01-Jan-2017].

[14] M. Kifer and H. Boley, "RIF Overview (Second Edition)," *W3C*, 2013. [Online]. Available: http://www.w3.org/2013/pdf/NOTE-rif-overview-20130205.pdf.

[15] G. Wagner, A. Giurca, and S. Lukichev, "R2ml: A general approach for marking up rules," in *Principles and practices of Semantic Web reasoning, Dagstuhl seminar proceedings*, 2005, vol. 5371, p. 47.

[16] O. Tablets, "Business Friendly Rules." 2012.

[17] Y. Biletskiy and G. R. Ranganathan, "An Invertebrate Semantic/Software Application Development Framework for knowledge-based systems," *Knowledge-Based Syst.*, vol. 21, no. 5, pp. 371–376, 2008.

[18] H. Boley and T. Athan, "RuleML Primer, August 2012," *Available online a t http//ruleml. org/papers/Primer/RuleMLPrimer2012-08-09/RuleMLPrimer-p0-2012-08-09. html, Visit. Oct. 19th*, 2012.

[19] H. Boley, "RIF RuleML Rosetta Ring: Round-Tripping the Dlex Subset of Datalog RuleML and RIF-Core," in *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, 2009, pp. 29–42.

[20] A. Polleres, H. Boley, and M. Kifer, "RIF datatypes and built-ins 1.0," *W3C Work.*

Draft. http//www. w3. org/2005/rules/wiki/DTB, 2009.
[21] M. Ball, H. Boley, D. Hirtle, J. Mei, and B. Spencer, "The OO jDREW reference implementation of RuleML," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2005, vol. 3791 LNCS, pp. 218–223.
[22] T. Parr, *The Definite ANTLR 4 Reference*. 2013.
[23] T. Rattanasawad, K. R. Saikaew, M. Buranarach, and T. Supnithi, "A review and comparison of rule languages and rule-based inference engines for the Semantic Web," in *2013 International Computer Science and Engineering Conference, ICSEC 2013*, 2013, pp. 1–6.
[24] H. Boley and M. Kifer, "RIF basic logic dialect," *W3C Work. Draft (July 2009)*, 2007.
[25] C. de Sainte Marie, G. Hallmark, and A. Paschke, "RIF production rule dialect," *W3C Recomm. (June 22, 2010)*, 2009.
[26] H. Boley, G. Hallmark, M. Kifer, A. Paschke, A. Polleres, and D. Reynolds, "RIF Core Dialect (Second Edition)," *W3C*, 2013. [Online]. Available: http://www.w3.org/TR/rif-core/.
[27] H. Boley and M. Kifer, "RIF Framework for logic dialects," *Work. Draft. W3C (July 3, 2009)*, 2009.
[28] G. Klyne and J. J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax," *W3C Recomm.*, vol. 10, no. October, pp. 1--20, 2004.
[29] Prud'hommeaux, Eric, et al. "Turtle–terse rdf triple language." Candidate Recommendation, W3C (2013).
[30] D. Reynolds, "Jena rules," in *Proceedings of the 2006 Jena user conference*, 2006.
[31] D. Beckett, "RDF 1.1 N-Triples," *W3C Recomm.*, 2014.
[32] S. Singh and R. Karwayun, "A comparative study of inference engines," in *ITNG2010 - 7th International Conference on Information Technology: New Generations*, 2010, pp. 53–57.
[33] T. I. M. Berners-Lee, J. Hendler, and O. R. a Lassila, *The Semantic Web*, vol. 284, no. 5. 2001.
[34] M. Ball, "Oo jdrew: Design and implementation of a reasoning engine for the semantic web," *CS4997 Honours Thesis Proj. Report, UNB*, vol. 6, 2005.
[35] H. Boley, "Integrating positional and slotted knowledge on the Semantic Web," in *Journal of Emerging Technologies in Web Intelligence*, 2010, vol. 2, no. 4, pp. 343–353.
[36] Knublauch, H., J. A. Hendler, and K. Idehen. "Spin sparql inferencing notation." Retrieved Feb 8 (2009): 2013.
[37] Harris, Steve, Andy Seaborne, and Eric Prud'hommeaux. "SPARQL 1.1 query language." W3C recommendation 21.10 (2013).
[38] Fürber, Christian, and Martin Hepp. "Using SPARQL and SPIN for data quality management on the semantic web." International Conference on Business Information Systems. Springer, Berlin, Heidelberg, 2010.
[39] Knublauch, Holger, and Arthur Ryman. "Shapes constraint language (SHACL)." W3C Candidate Recommendation 11 (2017): 8.
[40] Bassiliades, Nick, Grigoris Antoniou, and Ioannis Vlahavas. "A defeasible logic

reasoner for the semantic web." International Journal on Semantic Web and Information Systems (IJSWIS) 2.1 (2006): 1-41.
[41] Antoniou, Grigoris, and Antonis Bikakis. "Dr-prolog: A system for defeasible reasoning with rules and ontologies on the semantic web." IEEE transactions on knowledge and data engineering 19.2 (2007).
[42] B. N. Grosof, M. D. Gandhe, T. W. Finin, and others, “SweetJess: Translating DAMLRuleML to JESS.,” in *RuleML*, 2002.
[43] L. Laera, V. Tamma, T. Bench-Capon, and G. Semeraro, “SweetProlog: A system to integrate ontologies and rules,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2004, vol. 3323 LNCS, pp. 188–193.
[44] F. Gandon, M. Sheshagiri, and N. M. Sadeh, “ROWL: Rule language in OWL and translation engine for JESS,” *Mob. Commer. Lab.*, 2004.
[45] W. Chen, D. T. Ouyang, and Y. X. Ye, “RIF2Jess: Inferencing RIF rules via translation to Jess rules,” in *Proceedings - 2009 International Conference on Computational Intelligence and Software Engineering, CiSE 2009*, 2009.
[46] E. F. Hill, *Jess in Action: Java Rule-Based Systems*. 2003.
[47] J. De Roo, “Euler yet another proof engine.” 2013.
[48] V. S. Costa, R. Rocha, and L. Damas, “The YAP Prolog System,” *Theory Pract. Log. Program.*, vol. 12, no. 1–2, pp. 5–34, 2012.
[49] Jena, Apache. "A free and open source Java framework for building Semantic Web and Linked Data applications." Available online: jena. apache. org/(accessed on 28 April 2015) (2015).
[50] C. L. Forgy, “Rete: A fast algorithm for the many pattern/many object pattern match problem,” *Artif. Intell.*, vol. 19, no. 1, pp. 17–37, 1982.
[51] C. Ogbuji and others, “FuXi 1.4: A Python-based, bi-directional logical reasoning system for the semantic web,” *URL https//code.google.com/p/fuxi/*, 2015.
[52] C. Ogbuji, “InfixOWL: An idiomatic interface for OWL,” in *CEUR Workshop Proceedings*, 2009, vol. 432.
[53] T. Berners-Lee, “Notation 3 Logic,” *Www.W3.Org*. 2005.
[54] T. Berners-Lee, D. Connolly, and S. Hawke, “Primer: Getting into rdf & semantic web using n3.” 2005.
[55] https://github.com/OOjDREW. [Accessed: 29-Mar-2017].
[56] Bechhofer, Sean. "OWL: Web ontology language." Encyclopedia of database systems. Springer, Boston, MA, 2009. 2008-2009.
[57] D. Hirtle, T. Dema, D. D. Duong, and L. T. T. Thuy, “Glossary of RuleML 0.91.” .
[58] E. der Vlist, *Relax Ng*. “ O’Reilly Media, Inc.,” 2003.
[59] https://github.com/RuleML/ruleml-website. [Accessed: 29-Mar-2017].
[60] C.-L. Chang and R. C.-T. Lee, *Symbolic logic and mechanical theorem proving*. Academic press, 2014.
[61] H. Delugach, “ISO common logic. Standard, ISO.” 2008.
[62] M. R. Genesereth and R. E. Fikes, “Knowledge Interchange Format, Version 3.0 Reference Manual,” *Interchange*, no. Logic-92-1, pp. 1–68, 1992.
[63] J. F. Sowa, “Conceptual graphs: Information processing in mind and machine”, *Reading, Addison Wesley*, vol. 234, 1984.

[64] “Rulesystem Arrangement Framework.” [Online]. Available: http://www.w3.org/2005/rules/wg/wiki/Rulesystem_Arrangement_Framework.html. [Accessed: 29-Mar-2017].
[65] I. Akbari, “Rule Language Translation,” 2017. [Online]. Available: https://github.com/ismailakbari/Rule-Translator [Accessed: 20-Mar-2017].
[66] Decision Management Community, https://dmcommunity.wordpress.com/home/, 2017.
[67] S. Spreeuwenberg, B. Charlotte, and Z. Martijn. Het spel met de regels. Business Rules Platform Nederland, The Netherlands, 2013.
[68] Zou, G., Boley, H.: Port Clearance Rules in PSOA RuleML: From Controlled English Regulation to Object-Relational Logic. In: Proceedings of the RuleML+RR 2017 Challenge. Volume 1875., CEUR (July 2017).
[69] http://wiki.ruleml.org/index.php/PSOA_RuleML#PSOATransRun
[70] Boley, Harold. "PSOA RuleML: integrated object-relational data and rules." Reasoning Web International Summer School. Springer International Publishing, 2015.
[71] Paschke, Adrian, et al. "Rif use cases and requirements." W3C Working Draft (2008).

Appendix A

Notation3 Grammars and Use Cases

Table A.1 is the Notation3 grammar in EBNF notation. Literal characters are quoted. void means an empty production.

Table A.1. Notation3 Grammar in EBNF Notation

```
document ::=          statements_optional EOF
statements_optional ::=      statement  "."  statements_optional  |     void
statement ::=         declaration
              |       existential
              |       simpleStatement
              |       universal
declaration ::=               "@base"  explicituri
              |        "@keywords"  barename_csl
              |        "@prefix"  prefix explicituri
existential ::=               "@forSome"  symbol_csl
simpleStatement ::=   subject propertylist
universal ::=         "@forAll"  symbol_csl
explicituri ::=       <[^>]*>
barename_csl ::=      barename barename_csl_tail   |  void
symbol_csl ::=        symbol symbol_csl_tail    |  void
subject ::=           expression
propertylist ::=              predicate object objecttail propertylisttail    |  void

barename_csl_tail ::= ","  barename barename_csl_tail    |void
symbol ::=            explicituri       |   qname
symbol_csl_tail ::=   ","  symbol symbol_csl_tail  |   void
expression ::=        pathitem pathtail
predicate ::=         "<="  |  "=" | "=>" | "@a" | "@has"  expression
                                |               "@is"  expression  "@of"
              |       expression
object ::=            expression
objecttail ::=        ","  object objecttail      |void
propertylisttail ::=  ";"  propertylist     |         void
pathitem ::=          "("  pathlist  ")"  |  "["  propertylist  "]"      |       "{"
formulacontent  "}"
              |       boolean       |   literal |  numericliteral | quickvariable |
symbol
pathtail ::=          "!"  expression |      "^"  expression  |   void
pathlist ::=          expression pathlist    |    void
formulacontent ::=    statementlist
boolean ::=           "@false"       | "@true"
literal ::=           string dtlang
numericliteral ::=    decimal |      double |      integer |      rational


statementlist ::=             statement statementtail | void
dtlang ::=            "@"  langcode |        "^^"  symbol    |    void
statementtail ::=             "."  statementlist |  void
```

```
prefix ::=    ([A-Z_a-z#x00c0-#x00d6#x00d8-#x00f6#x00f8-#x02ff#x0370-#x037d#x037f-
#x1fff#x200c-#x200d#x2070-#x218f#x2c00-#x2fef#x3001-#xd7ff#xf900-#xfdcf#xfdf0-
#xfffd#x00010000-#x000effff][\-0-9A-Z_a-z#x00b7#x00c0-#x00d6#x00d8-#x00f6#x00f8-
#x037d#x037f-#x1fff#x200c-#x200d#x203f-#x2040#x2070-#x218f#x2c00-#x2fef#x3001-
#xd7ff#xf900-#xfdcf#xfdf0-#xfffd#x00010000-#x000effff]*)?:
barename ::=  [A-Z_a-z#x00c0-#x00d6#x00d8-#x00f6#x00f8-#x02ff#x0370-#x037d#x037f-
#x1fff#x200c-#x200d#x2070-#x218f#x2c00-#x2fef#x3001-#xd7ff#xf900-#xfdcf#xfdf0-
#xfffd#x00010000-#x000effff][\-0-9A-Z_a-z#x00b7#x00c0-#x00d6#x00d8-#x00f6#x00f8-
#x037d#x037f-#x1fff#x200c-#x200d#x203f-#x2040#x2070-#x218f#x2c00-#x2fef#x3001-
#xd7ff#xf900-#xfdcf#xfdf0-#xfffd#x00010000-#x000effff]*
qname ::=     (([A-Z_a-z#x00c0-#x00d6#x00d8-#x00f6#x00f8-#x02ff#x0370-#x037d#x037f-
#x1fff#x200c-#x200d#x2070-#x218f#x2c00-#x2fef#x3001-#xd7ff#xf900-#xfdcf#xfdf0-
#xfffd#x00010000-#x000effff][\-0-9A-Z_a-z#x00b7#x00c0-#x00d6#x00d8-#x00f6#x00f8-
#x037d#x037f-#x1fff#x200c-#x200d#x203f-#x2040#x2070-#x218f#x2c00-#x2fef#x3001-
#xd7ff#xf900-#xfdcf#xfdf0-#xfffd#x00010000-#x000effff]*)?:)?[A-Z_a-z#x00c0-
#x00d6#x00d8-#x00f6#x00f8-#x02ff#x0370-#x037d#x037f-#x1fff#x200c-#x200d#x2070-
#x218f#x2c00-#x2fef#x3001-#xd7ff#xf900-#xfdcf#xfdf0-#xfffd#x00010000-#x000effff][\-0-
9A-Z_a-z#x00b7#x00c0-#x00d6#x00d8-#x00f6#x00f8-#x037d#x037f-#x1fff#x200c-
#x200d#x203f-#x2040#x2070-#x218f#x2c00-#x2fef#x3001-#xd7ff#xf900-#xfdcf#xfdf0-
#xfffd#x00010000-#x000effff]*
quickvariable ::=     \?[A-Z_a-z#x00c0-#x00d6#x00d8-#x00f6#x00f8-#x02ff#x0370-
#x037d#x037f-#x1fff#x200c-#x200d#x2070-#x218f#x2c00-#x2fef#x3001-#xd7ff#xf900-
#xfdcf#xfdf0-#xfffd#x00010000-#x000effff][\-0-9A-Z_a-z#x00b7#x00c0-#x00d6#x00d8-
#x00f6#x00f8-#x037d#x037f-#x1fff#x200c-#x200d#x203f-#x2040#x2070-#x218f#x2c00-
#x2fef#x3001-#xd7ff#xf900-#xfdcf#xfdf0-#xfffd#x00010000-#x000effff]*
string ::=    ("""[^"\\]*(?:(?:\\.|"(?!""))[^"\\]*)*""")|("[^"\\]*(?:\\.[^"\\]*)*")
decimal ::=   [-+]?[0-9]+(\.[0-9]+)?
double ::=    [-+]?[0-9]+(\.[0-9]+)?([eE][-+]?[0-9]+)
integer ::=   [-+]?[0-9]+
rational ::=  integer  "/"  unsignedint
langcode ::=  [a-z]+(-[a-z0-9]+)*
unsignedint ::=       [0-9]+
```

Table A.2 shows the N3 grammar in ANTLR4 notation equivalent to the EBNF notation of N3 in Table A.1.

‘@header’ symbol is a code snippet in ANTLR that is used for implementation purposes. Single and multiline comments in ANLTR start with // and /** respectively (as like other programming languages). In an ANLTR grammar, identifiers of production rules (also called parser rules) start with a lowercase letter and identifiers of Terminals (lexer rules) start with a capital letter. Keyword ‘skip’ in a rule ignores that rule while parsing the input stream (e.g. ignoring comments). Part 1 of the grammar defines the header section of N3

rules. Part 2 shows the rules to define N3 statements and part 3 includes the token definitions:

Table A.2. Notation3 Grammar in ANTRL4 Notation

```
/**
 * Definition of N3 grammar
 */
grammar N3;           // grammar name

// implementation consideration
@header{
       package iakbari;
}
//define N3 rules
r: g0 g1 g2 n3_statements_optional;

//PART 1 HEADERS: include DECLARATION, UNIVERSAL AND EXISTENTIAL
/* rule alternatives can be empty which is similar to void in EBNF notation e.g. in g0
production rule*/

g0:           |              n3_declaration g0    ;
n3_declaration:        '@prefix' PREFIX N3_EXPLICITURI '.'  |             '@keywords'
g8
                       |      '@base' N3_EXPLICITURI '.'  ;
g8:                '.'  |     BARENAME g11         ;
g11:                    '.'  |       ',' BARENAME g11     ;
g1:                    |      n3_universal g1             ;
n3_universal:                  '@forAll' g6                              ;
g6:      '.'    |      n3_symbol g9                   ;
g9:      '.'    |       ',' n3_symbol g9                     ;
g2:                    |       n3_existential g2                    ;
n3_existential:                          '@forSome' g7                     ;
g7:      '.'    |       n3_symbol g10               ;
g10:     '.'    |       ',' n3_symbol g10           ;



//PART 2 N3_STATEMENTS_OPTIONAL

n3_statements_optional:       |       n3_statement '.' n3_statements_optional    ;

n3_statement:   n3_subject n3_propertylist
       ;
n3_subject:     n3_path
       ;
n3_path:        n3_node n3_pathtail
       ;
n3_node:        n3_symbol             |   '{' n3_formulacontent '}'     |
N3_VARIABLE
                                      |   N3_NUMERICLITERAL        |   n3_literal
                                      |   '[' n3_propertylist ']'  |   '(' n3_pathlist
')'
```

```
                                                |   '@this'
;
n3_pathtail:          |   '!' n3_path                      |   '^' n3_path
        ;
n3_formulacontent:          |   g3 g4 g5 n3_statementlist
        ;
g3:                                         |   n3_declaration g3
        ;
g4:                                   |   n3_universal g4
        ;
g5:                                   |   n3_existential g5
;
n3_statementlist:                     |   n3_statement n3_statementtail
        ;
n3_statementtail:           |   '.' n3_statementlist
        ;
n3_propertylist:            |   n3_verb n3_object n3_objecttail n3_propertylisttail
        ;
n3_verb:        n3_path             |   '@has' n3_path   |   '@is' n3_path '@of'   |
'@a'
                                            |   '='           |   '=>'  |   '<='
        ;
n3_object:                     n3_path      ;
n3_objecttail:        |    ',' n3_object n3_objecttail
        ;
n3_propertylisttail:          |    ';' n3_propertylist
        ;
n3_pathlist:          |   n3_path n3_pathlist ;

//PART 3: TOKENS/Terminals/Lexer rules

n3_symbol:      N3_EXPLICITURI |  n3_qname  ; // N3_QNAME is replaced by PREFIX?
BARENAME
n3_qname: PREFIX? BARENAME   ;
n3_literal:     N3_STRING n3_dtlang?        ;
n3_dtlang:      N3_LANGCODE     |   '^^' n3_symbol         ;
//NUMBERS
N3_NUMERICLITERAL: DECIMAL  |     DOUBLE     |     INTEGER    |   RATIONAL;
RATIONAL: INTEGER  '/'  UNSIGNEDINT                                        ;
DECIMAL: INTEGER '.' [0-9]*   ;



DOUBLE: INTEGER ('.'UNSIGNEDINT)? ([eE][-+]?INTEGER)       ;
INTEGER: [-+]? UNSIGNEDINT                                                 ;
UNSIGNEDINT: [0-9]+
;

/**Regular expression of N3_LANGCODE is:  [a-z]+('-'[a-z0-9]+)*
However, because it has collision with BARENAME I changed it to the following: */

N3_LANGCODE:  '@' [a-z][a-z]('-'[a-z0-9]+)*                                ;
N3_STRING:  ('"""')(.|[\s\S])*? ('"""')   |    ('"')(.|[\s\S])*? ('"')          ;

N3_VARIABLE : '?' BARENAME                          ;

//PREFIX: [a-z]':' ;
```

```
PREFIX: (BARENAME)? ':'                                          ;       //should be before
BARENAME
BARENAME:  [A-Z_a-z\u00c0-\u00d6\u00d8-\u00f6\u00f8-\u02ff\u0370-\u037d\u037f-
\u1fff\u200c-\u200d\u2070-\u218f\u2c00-\u2fef\u3001-\ud7ff\uf900-\ufdcf\ufdf0-
\ufffd\U00010000-\U000effff] [\\-0-9A-Z_a-z\u00b7\u00c0-\u00d6\u00d8-\u00f6\u00f8-
\u037d\u037f-\u1fff\u200c-\u200d\u203f-\u2040\u2070-\u218f\u2c00-\u2fef\u3001-
\ud7ff\uf900-\ufdcf\ufdf0-\ufffd\U00010000-\U000effff]* ;

N3_EXPLICITURI: '<' .*? '>'                                      ;
WS: [ \t\r\n]+ -> skip                                                   ;       //
skip spaces, tabs, newlines
//comment
COMMENT: '#' .*? ('\n' | '\r') -> skip                           ;
```

The following examples 1 to 3 in tables A.3 to A.5, are used for the purpose of testing N3 parser and translator in the dissertation respectively.

Table A.3. Notation3 rules example 1

```
#  Example 1
#  Notation3 generation by
#       notation3.py,v 1.54 2001/02/12 18:38:31 timbl Exp

@base <http://www.example.org/> .

:Joe :father :Fred.
:Fred :brother :Bob, :Rob.
:Alex :brother :Jack, John ; :sister :Sarah.
```

Table A.4. Notation3 rules example 2

```
#Example 2
#Processed by Id: cwm.py,v 1.29 2001/02/28 20:45:26 timbl Exp
      #    using base file:/afs/w3.org/pub/WWW/2000/10/swap/test/meet/

#  Notation3 generation by
#       notation3.py,v 1.54 2001/02/12 18:38:31 timbl Exp

    @base <http://www.example.org/>.
    @prefix g: <http://www.another.example.org/geographical#> .
    @prefix : <http://www.example.org/meeting_organization#> .
    @prefix p: <http://www.example.org/personal_details#> .

   <http://meetings.example.com/cal#m1>     :Location  [
          g:zip "02139";
          g:lat "14.124425";
          g:long "14.245" ];
       :chair <http://www.example.org/people#fred>;
       :homePage <http://meetings.example.com/m1/hp> .
```

```
<http://meetings.example.com/m1/hp>     :policy <http://meetings.example.com/privacyPolicy> .

 <http://www.example.org/people#fred>     :attending <http://meetings.example.com/cal#m1>;
        p:GivenName "Fred";
        p:hasEmail <mailto:fred@example.com> .

 #ENDS
```

Table A.5. Notation3 rules example 3

```
#Example 3
@prefix : <http://www.example.org/meeting_organization#> .
@forSome :who1, :who2.
{ :who1 :father [ :brother :who2 ] } => { :who1 :uncle :who2 }.
```

Appendix B

POSL Grammars and Use Cases

Table B.1 shows the EBNF grammar of POSL language as provided in POSL website. It should be mention that the grammar is missing the definition of some of the production rules such as ‘rel’, ‘role’, ‘type’, and ‘symbol’.

Table B.1. POSL grammar (from POSL website)

```
rulebase ::= (clause | signature)* .
clause ::= atom (IMP atoms)? PERIOD .
signature ::= atom ASTERISK .
atoms ::= atom (COMMA atom)* .
atom ::= rel LPAREN oid? cont RPAREN .
oid ::= term HAT .
cont ::= SEMI? term? | SEMI COMMA | ps .
ps ::=
(pos prest? | prest) (SEMI set)? srest?
| set srest pstrail?
| (set pstrail?)? srest? .
pstrail ::= (SEMI pos prest? | prest) (SEMI set)? .
prest ::= PIPE (var | posplex) .
srest ::= BANG (var | setplex) .
posplex ::= LBRACK pos? prest? RBRACK .
setplex ::= LBRACK (SEMI term? | term SEMI set)? srest? RBRACK .
pos ::= COMMA term? | term (COMMA term)+ .
set ::= term (SEMI term)* .
term ::= slot | unkeyed .
slot ::= role ARROW unkeyed .
unkeyed ::=
ind
| var
| skolem
| structure
| plex .
structure ::= ctor LBRACK cont RBRACK (COLON type)? .
plex ::= LBRACK cont RBRACK .
ind ::= (symbol uri? | uri) (COLON type)? .
var ::= QMARK symbol? (COLON type)? .
skolem ::= USCORE symbol? (COLON type)? .
ctor ::= rel ::= role ::= type ::= symbol .
```

Table B.2 shows the POSL grammar in ANTLR4 notation developed as a part of this dissertation.

Table B.2. POSL grammar in ANTLR4 notation

```
/**
 * Definition of POSL grammar
 */
grammar POSL;
@header{
       package iakbari;
}

//rulebase:
r:        (clause)* EOF;
clause:     atom (IMP atoms)? PERIOD;
atoms:     atom (COMMA atom)*;
atom:  rel LPAREN oid? (ps)?  RPAREN ;
ps:       pos (prest)? (SEMI slots)? (srest)?
          | slots     ((prest (SEMI slots)?)? (srest)?   | SEMI pos (prest)?
(SEMI slots)? (srest)?   )
          | prest (SEMI slots)? (srest)?    | srest      | atom      ;
oid:      term HAT;
prest:     PIPE (var | posplex);
srest:     BANG (var | slotplex);
posplex:     LBRACK (pos)? (prest)? RBRACK;
slotplex:     LBRACK (slots)? (srest)? RBRACK;
pos:     term (COMMA term)*;
slots     slot (SEMI slot)*;
slot:     role ARROW term;
term:       ind     | var     | cterm     | skolem     | plex     ;
cterm:   ctor LBRACK (ps)? RBRACK (COLON type)?;
plex:     LBRACK (ps)? RBRACK;
ctor:     symbol;
rel:       symbol;
role:     symbol;
type:     symbol;
ind:     (symbol (uri)? | uri) (COLON type)?;
skolem:     USCORE (symbol)? (COLON type)?;
var:     QMARK (symbol)? (COLON type)?;
symbol:        SYMBOL | QSYMBOL;
uri:     URI;

PIPE:   '|';
BANG:   '!';
HAT:    '^';
COLON:  ':';
SEMI:   ';';
LBRACK: '[';
RBRACK: ']';
LPAREN: '(';
RPAREN: ')';
QMARK: '?';
```

```
COMMA: ',';
PERIOD: '.';
LBRACE: '{';
RBRACE: '}';
USCORE: '_';

IMP: ':-';
ARROW: '->'; // the symbol that separates a role name from the value of a slot
URI:   '<' ('a'..'z'|'A'..'Z'|'0'..'9'|'_'|':'|'/'|'.'|'?'|'&'|'%'|'#'|'-')+
'>';
SYMBOL:  ('-')? ('a'..'z'|'A'..'Z'|'0'..'9'|'$')
('a'..'z'|'A'..'Z'|'0'..'9'|'_'|'.'|'$')*;
QSYMBOL: '\"' (~('\"'))* '\"';
COMMENT:
       '%' (~('\n'))*  -> skip ;
    //'%' (~('\n'))* { $skip };
// Single-line comments are denoted by % - the remainder of the line is ignored
MLCOMMENT:
       '/%' (~('%'))* '%/' -> skip ;
    //'/%' (~('%'))* '%/' { $skip };
// Multi line comments are denoted by /% .... %/ - the comment body cannot
contain the % character
WS:    (' ' | '\t'    | '\r' '\n'    | '\n'     )   -> skip       ;
```

Table B.3 and Table B.4 show two rule examples in POSL that are used to test the POSL translator to RIF.

Table B.3. POSL rules example 1

```
% An Example POSL rule

/% Rule: A discount of 5.0 percent is given to a ?customer

    on a certain ?product, if that ?customer is premium and the ?product is a regular product. %/

discount(?customer, ?product, "5.0 percent") :- premium(?customer), regular(?product).
```

Table B.4. POSL rules example 2

```
% POSL Rule Example Run in OO jDREW              2007-07-17

% Harold Boley, Benjamin L. Craig, Greg Sherman

% Rule: A discount of 5.0 percent is given to a ?customer
```

```
% on a certain ?product, if that ?customer is premium and the ?product is a regular product.

discount(?customer, ?product, "5.0 percent") :- premium(?customer), regular(?product).

% Rule: A discount of 7.5 percent is given to a ?customer

% on a certain ?product, if that ?customer is premium and the ?product is a luxury product.

discount(?customer, ?product, "7.5 percent") :- premium(?customer), luxury(?product).

% Rule: A ?customer is premium if their spending is "min 5000 euro" in the previous year.

premium(?customer) :- spending(?customer, "min 5000 euro", "previous year").

% Fact: A Porsche is a luxury product.

luxury(Porsche).

% Fact: A Honda is a regular product.

regular(Honda).

% Fact: "Peter Miller"'s spending was "min 5000 euro" in the previous year.

spending("Peter Miller", "min 5000 euro", "previous year").

spending("Peter Miller", "min 7000 euro", "previous year").
```

Table B.5 shows the RIF equivalents of POSL rules example 2 in Table B.4.

Table B.5. RIF rules translated from POSL rules in Table B.4

```
Document(
Group(
Forall ?customer ?product(
discount( ?customer ?product "5.0 percent") :-
And(
        premium( ?customer)
        regular( ?product)
)
)
Forall ?customer ?product(
discount( ?customer ?product "7.5 percent") :-
And(
        premium( ?customer)
        luxury( ?product)
)
)
```

```
Forall ?customer(
premium( ?customer) :- spending( ?customer "min 5000 euro" "previous year")
)
luxury( Porsche)
regular( Honda)
spending( "Peter Miller" "min 5000 euro" "previous year")
spending( "Peter Miller" "min 7000 euro" "previous year")

))
```

Appendix C

Use Cases of SWRL and Datalog RuleML Translation to RIF

Table C.1 shows a SWRL rule example which is translated to its equivalent rule RIF as shown in Table C.2. The translation is done using ‘SWRL2RIF.xsl’ stylesheet.

Table C.1. SWRL rules example

```
<?xml version='1.0' ?>
<?xml-stylesheet type="text/xsl" href="SWRL2RIF.xsl"?>

<swrlx:Ontology
  xmlns:owlx="http://www.w3.org/2003/05/owl-xml"
  xmlns:swrlx="http://www.w3.org/2003/11/swrlx"
  xmlns:ruleml="http://www.w3.org/2003/11/ruleml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2003/11/swrlx
http://www.ruleml.org/swrl/xsd/swrlx.xsd">
  <owlx:Annotation>
    <owlx:Documentation>SWRL Example 5.1-2</owlx:Documentation>
  </owlx:Annotation>
  <owlx:VersionInfo>$Id: example5.1-2.swrlx,v 1.1 2004/05/21 18:46:39 vivien Exp
$</owlx:VersionInfo>

<ruleml:var>x1</ruleml:var>
<ruleml:var>x2</ruleml:var>
<ruleml:var>x3</ruleml:var>

<ruleml:imp>
  <ruleml:_rlab ruleml:href="#example1"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom  swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hasSibling">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hasSex">
      <ruleml:var>x3</ruleml:var>
      <owlx:Individual owlx:name="#male" />
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
```

```
</ruleml:imp>
</swrlx:Ontology>
```

Table C.2. SWRL Translation results in RIF using SWRL2RIF translator

```
<?xml version="1.0" encoding="utf-8"?>
<Document
 xmlns="http://www.w3.org/2007/rif#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xs="http://www.w3.org/2001/XMLSchema#">
   <payload>
      <Group>
         <_rlab>ignored.</_rlab>
         <sentence>
            <Forall>
               <declare>
                  <Var>x2</Var>
               </declare>
               <declare>
                  <Var>x1</Var>
               </declare>
               <declare>
                  <Var>x3</Var>
               </declare>
               <formula>
                  <Implies>
                     <if>
                        <And>
                           <Atom>
                              <op>hasParent</op>
                              <args ordered="yes">
                                 <Var>x1</Var>
                                 <Var>x2</Var>
                              </args>
                           </Atom>
                           <Atom>
                              <op>hasSibling</op>
                              <args ordered="yes">
                                 <Var>x2</Var>
                                 <Var>x3</Var>
                              </args>
                           </Atom>
                           <Atom>
                              <op>hasSex</op>
                              <args ordered="yes">
                                 <Var>x3</Var>
                                 <Const
type="http://www.w3.org/2007/rif#iri">#male</Const>
                              </args>
                           </Atom>
                        </And>
                     </if>
```

```
                    <then>
                       <Atom>
                          <op>hasUncle</op>
                          <args ordered="yes">
                             <Var>x1</Var>
                             <Var>x3</Var>
                          </args>
                       </Atom>
                    </then>
                 </Implies>
              </formula>
           </Forall>
        </sentence>
     </Group>
  </payload>
  <Annotation>ignored.</Annotation>
  <VersionInfo>ignored.</VersionInfo>
</Documnet>
```

Table C.3 shows a rule example in RuleML which is translated to its equivalent rule in RIF as shown in Table C.4.

The translation is done using ‘UBDatalogRuleML2RIF_BLD.xsl’ stylesheet.

Table C.3. Rule example in Unary/Binary Datalog RuleML

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model
href="http://deliberation.ruleml.org/1.02/relaxng/bindatalog_relaxed.rnc"?>
<?xml-stylesheet type="text/xsl" href="UBDatalogRuleML2RIF_BLD.xsl"?>
<RuleML
xmlns="http://ruleml.org/spec"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ruleml.org/spec
http://deliberation.ruleml.org/1.02/xsd/bindatalog.xsd">

<Assert mapClosure="universal">

<Implies>
  <if>
    <Atom>
      <op>
        <Rel>want to review</Rel>
      </op>
      <Var>you</Var>
      <Ind>rule principles</Ind>
    </Atom>
  </if>
  <then>
    <Atom>
```

```
      <op>
        <Rel>may look at</Rel>
      </op>
      <Var>you</Var>
      <Ind iri="http://www.cs.brandeis.edu/...">Rule-Based Systems</Ind>
    </Atom>
  </then>
</Implies>

  <Atom>
    <op>
      <Rel>want to review</Rel>
    </op>
    <Ind>fred</Ind>
    <Ind>rule principles</Ind>
  </Atom>

</Assert>
</RuleML>
```

Table C.4. Translation result of rules in Table C.3 to RIF

```
<?xml version="1.0" encoding="utf-8"?>
<Document xmlns="http://www.w3.org/2007/rif#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema#"
xmlns:ruleml="http://ruleml.org/spec">
   <payload>
      <Atom>
         <op>
            <Const type="http://www.w3.org/2007/rif#iri">want to review</Const>
         </op>
         <args ordered="yes">
            <Const type="http://www.w3.org/2007/rif#iri">fred</Const>
            <Const type="http://www.w3.org/2007/rif#iri">rule principles</Const>
         </args>
      </Atom>
      <Implies>
         <if>
            <Atom>
               <op>
                  <Const type="http://www.w3.org/2007/rif#iri">want to
review</Const>
               </op>
               <args ordered="yes">
                  <Var>you</Var>
                  <Const type="http://www.w3.org/2007/rif#iri">rule
principles</Const>
               </args>
            </Atom>
         </if>
```

```
         <then>
            <Atom>
               <op>
                  <Const type="http://www.w3.org/2007/rif#iri">may look at</Const>
               </op>
               <args ordered="yes">
                  <Var>you</Var>
                  <Const type="http://www.cs.brandeis.edu/...">Rule-Based
Systems</Const>
               </args>
            </Atom>
         </then>
      </Implies>
   </payload>
</Document>
```

Appendix D

RIF Grammars in EBNF and ANTRL4 notations

Table D.1 shows the RIF-BLD grammar in EBNF notation. The grammar is borrowed from RIF W3C website.

Table D.1. RIF grammar in EBNF notation

```
//Rule Language
  Document       ::= IRIMETA? 'Document' '(' Base? Prefix* Import* Group? ')'
  Base                ::= 'Base' '(' ANGLEBRACKIRI ')'
  Prefix              ::= 'Prefix' '(' NCName ANGLEBRACKIRI ')'
  Import            ::= IRIMETA? 'Import' '(' LOCATOR PROFILE? ')'
  Group             ::= IRIMETA? 'Group' '(' (RULE | Group)* ')'
  RULE              ::= (IRIMETA? 'Forall' Var+ '(' CLAUSE ')') | CLAUSE
  CLAUSE         ::= Implies | ATOMIC
  Implies           ::= IRIMETA? (ATOMIC | 'And' '(' ATOMIC* ')') ':-' FORMULA
  LOCATOR    ::= ANGLEBRACKIRI
  PROFILE       ::= ANGLEBRACKIRI

//Condition Language:
 FORMULA        ::= IRIMETA? 'And' '(' FORMULA* ')' |   IRIMETA? 'Or' '(' FORMULA* ')'
|
                                  IRIMETA? 'Exists' Var+ '(' FORMULA ')' |   ATOMIC |
                                  IRIMETA? 'External' '(' Atom ')'
  ATOMIC          ::=  IRIMETA? (Atom | Equal | Member | Subclass | Frame)
  Atom                ::= UNITERM
  UNITERM        ::= Const '(' (TERM* | (Name '->' TERM)*) ')'
  Equal                ::= TERM '=' TERM
  Member             ::= TERM '#' TERM
  Subclass            ::= TERM '##' TERM
  Frame               ::= TERM '[' (TERM '->' TERM)* ']'
  TERM               ::= IRIMETA? (Const | Var | Expr | List | 'External' '(' Expr
')')
  Expr                ::= UNITERM
  List                  ::= 'List' '(' TERM* ')' | 'List' '(' TERM+ '|' TERM ')'
  Const               ::= '"' UNICODESTRING '"^^' SYMSPACE | CONSTSHORT
  Var                  ::= '?' Name
  Name               ::= NCName | '"' UNICODESTRING '"'
  SYMSPACE    ::= ANGLEBRACKIRI | CURIE
//Annotations:
  IRIMETA        ::= '(*' IRICONST? (Frame | 'And' '(' Frame* ')')? '*)'

//The rest
  ANGLEBRACKIRI        ::= IRI_REF
  IRI_REF                  ::=  '<' ([^<>"{}|^`\]-[#x00-#x20])* '>'
  NCName                   ::=  NCNameStartChar NCNameChar*     /* An XML Name,
minus the ":" */
  NCNameStartChar     :=  NameStartChar - ':'
```

```
  NCNameChar                ::=   NameChar - ':'
  NameStartChar       ::=   ":" | [A-Z] | "_" | [a-z] | [#xC0-#xD6] | [#xD8-#xF6] |
[#xF8-#x2FF] | [#x370-#x37D]
                                  | [#x37F-#x1FFF] | [#x200C-#x200D] | [#x2070-
#x218F] | [#x2C00-#x2FEF]
                                  | [#x3001-#xD7FF] | [#xF900-#xFDCF] | [#xFDF0-
#xFFFD] | [#x10000-#xEFFFF]
  NameChar          ::=   NameStartChar | "-" | "." | [0-9] | #xB7 | [#x0300-#x036F] |
[#x203F-#x2040]
  UNICODESTRING       ::=  '"'(.|[\s\S])*? '"' //I used production of string
(N3_STRING) from N3 grammar
  CONSTSHORT ::=   ANGLEBRACKIRI              // shortcut for "..."^^rif:iri
                  | CURIE                                                    //
shortcut for "..."^^rif:iri
                  | '"' UNICODESTRING '"'                        // shortcut for
"..."^^xs:string
                  | NumericLiteral                                        // shortcut
for "..."^^xs:integer,xs:decimal,xs:double
                  | '_' NCName                                             // shortcut
for "..."^^rif:local
                  | '"' UNICODESTRING '"' '@' langtag     // shortcut for
"...@..."^^rdf:PlainLiteral
  CURIE                    ::=  PNAME_LN | PNAME_NS
  PNAME_LN          ::=  PNAME_NS PN_LOCAL
  PNAME_NS          ::=  PN_PREFIX? ':'
  PN_LOCAL          ::=  ( PN_CHARS_U | [0-9] ) ((PN_CHARS|'.')* PN_CHARS)?
  PN_PREFIX         ::=  PN_CHARS_BASE ((PN_CHARS|'.')* PN_CHARS)?
  PN_CHARS_U       ::=  PN_CHARS_BASE | '_'
  PN_CHARS          ::=  PN_CHARS_U | '-' | [0-9] | #x00B7 | [#x0300-#x036F] | [#x203F-
#x2040]
  PN_CHARS_BASE       ::=  [A-Z] | [a-z] | [#x00C0-#x00D6] | [#x00D8-#x00F6] |
[#x00F8-#x02FF]
                                              | [#x0370-#x037D] | [#x037F-#x1FFF]
| [#x200C-#x200D]
                                              | [#x2070-#x218F] | [#x2C00-#x2FEF]
| [#x3001-#xD7FF]
                                              | [#xF900-#xFDCF] | [#xFDF0-#xFFFD]
| [#x10000-#xEFFFF]
  NumericLiteral                  ::=  NumericLiteralUnsigned | NumericLiteralPositive
|   NumericLiteralNegative
  NumericLiteralUnsigned       ::=  INTEGER |      DECIMAL |      DOUBLE
  NumericLiteralPositive       ::=  INTEGER_POSITIVE |    DECIMAL_POSITIVE |
       DOUBLE_POSITIVE
  NumericLiteralNegative       ::=  INTEGER_NEGATIVE |    DECIMAL_NEGATIVE |
       DOUBLE_NEGATIVE
  INTEGER                     ::=  [0-9]+
  DECIMAL                       ::=  [0-9]+ '.' [0-9]* | '.' [0-9]+
  DOUBLE                         ::=  [0-9]+ '.' [0-9]* EXPONENT | '.' ([0-9])+
EXPONENT | ([0-9])+ EXPONENT
  INTEGER_POSITIVE     ::=   '+' INTEGER
  DECIMAL_POSITIVE     ::=   '+' DECIMAL
  DOUBLE_POSITIVE      ::=   '+' DOUBLE
  INTEGER_NEGATIVE     ::=   '-' INTEGER
  DECIMAL_NEGATIVE     ::=   '-' DECIMAL
  DOUBLE_NEGATIVE      ::=   '-' DOUBLE
  EXPONENT             ::=   [eE] [+-]? [0-9]+
  langtag              ::=   [a-z][a-z]('-'[a-z0-9]+)* //langtag is simplified here
```

Table D.2 shows the RIF-BLD grammar in ANTLR 4 notation developed as a part of this dissertation. Comments in ANTLR grammar is similar to comments in Java or C++ language (// for single line and /*…*/ for multiline comments). '@header' header part of the grammar if for Java packaging purposes. 'fragment' word before a production rule means that the is part of a bigger rule and output of the rule is not a token but a part of another token.

Table D.2. RIF grammar in ANTRL4 notation

```
/**
 * Define RIFBLD grammar
 */
grammar RIFBLD;

//-------------------------------------------------------------------
//Header

@header{
       package iakbari;
}

//-------------------------------------------------------------------

//Rule Language:
r: document EOF;
  document :           irimeta? 'Document' '(' base? prefix* import1* group? ')';
  base          :           'Base'  '(' ANGLEBRACKIRI ')' ;
  prefix        :          'Prefix'  '(' NCName ANGLEBRACKIRI ')'  ;
  import1     :         irimeta? 'Import' '(' angle? ')'  ;
  angle      :          ANGLEBRACKIRI ;
  group        :          irimeta? 'Group' '(' (rule1 | group)* ')'      ;
  rule1         :          (irimeta? 'Forall' Var+ '(' clause ')')
                | clause    ;
  clause        :          implies | atomic        ;
  implies      :          irimeta? (atomic | 'And' '(' atomic* ')') ':-' formula;

//-------------------------------------------------------------------
//Condition Language:
  formula     :          irimeta? 'And' '(' formula* ')'
                | irimeta? 'Or' '(' formula* ')'
                | irimeta? 'Exists' Var+ '(' formula ')'
                | atomic
                                | irimeta? 'External' '(' uniterm ')'
;



  atomic      :         irimeta? ( equal | member | subclass | uniterm | frame) ;
```

```
  equal        :           term '=' term    ;
  member     :           term '#' term       ;
  subclass     :           term '##' term    ;
  irimeta      :           '(*' IRICONST? (frame | 'And' '(' frame* ')')? '*)'  ;
  frame        :           term '[' (term '->' term)* ']' ;
  term       :           irimeta? ( const1 | Var | uniterm | list | 'External' '('
uniterm ')')   ;
  list             :            'List' '(' term* ')' | 'List' '(' term+ '|' term
')'     ;
  uniterm      :           const1 ('(' (term* | (NCName '->' term)*) ')') ;

  //-------------------------LEXER RULES ----------------------------------------
  Var            :          '?' (NCName | (PN_CHARS_BASE)*)
;
  IRICONST:           '"'  ('http' | 'https'|'ftp'|'file') ':' [-a-zA-Z0-
9+&@#/%?=~_|!:,.;]*
                                [-a-zA-Z0-9+&@#/%=~_|] '"^^' 'rif:iri' ;

  //-----------------------------------------------------------------
  constshort      :       ANGLEBRACKIRI              // shortcut for
"..."^^rif:iri
                              | CURIE                       // shortcut for
"..."^^rif:iri
                              | UNICODE
                              | NCName                      // shortcut for
"..."^^rif:local
                              | NumericLiteral        ;     // shortcut for
"..."^^xs:integer,xs:decimal,xs:double
  UNICODE    :      UNICODESTRING ('@' ([a-z][a-z]('-'[a-z0-9]+)*))?;
  UNICODESTRING    :         '"'(.|[\s\S])*? '"' ;      // I used N3_STRING from
N3 grammar as UNICODESTRING
  symspace             :          ANGLEBRACKIRI | CURIE ;

  //ANGLEBRACKIRI      :           IRI_REF  ;
  //IRI_REF                        :           '<' ([^<>"{}|^`\]-[#x00-#x20])*
'>'     ;
  //ANGLEBRACKIRI     :          '<' ~([<>"{}|^`\]|[#x00-#x20])* '>'     ;
  ANGLEBRACKIRI        :           '<'
~('<'|'>'|'\"'|'{'|'}'|'^'|'`'|'\\'|[\u0000-\u0020])* '>';
  NCName                     :          (PN_CHARS_BASE | '_') (PN_CHARS_BASE |
'_' | '-'
                                                  | [0-9] | [#x00B7] |
[#x0300-#x036F] | [#x203F-#x2040] )*;
  //CURIE                          :          PNAME_LN | PNAME_NS ;
  //PNAME_LN                :          PNAME_NS PN_LOCAL ;
  //CURIE                          :          PNAME_NS PN_LOCAL? ;
  CURIE                      :          (PN_CHARS_BASE ((PN_CHARS_BASE | '_' | '-'
| [0-9]
                                                  | [#x00B7] | [#x0300-#x036F]
| [#x203F-#x2040]|'.')*
                                                  (PN_CHARS_BASE | '_' | '-' |
[0-9] | [#x00B7] | [#x0300-#x036F]
                                                  | [#x203F-#x2040]))?)? ':'
(( PN_CHARS_BASE | '_' | [0-9] )
                                                  ((PN_CHARS_BASE | '_' | '-'
|
```

```
[0-9] | [#x00B7]
                                                    | [#x0300-#x036F] | [#x203F-
#x2040]|'.')* (PN_CHARS_BASE
                                                    | '_' | '-' | [0-9] |
[#x00B7] | [#x0300-#x036F] | [#x203F-#x2040]))?)? ;

  //PN_CHARS               :          PN_CHARS_U | '-' | [0-9] | [#x00B7] |
[#x0300-#x036F] | [#x203F-#x2040] ;
  //PN_CHARS_U            :          PN_CHARS_BASE | '_'  ;
  // PN_CHARS_BASE is same as BARENAME in N3
  fragment PN_CHARS_BASE   :        [A-Z] | [a-z] | [#x00C0-#x00D6] | [#x00D8-
#x00F6]
                                                        | [#x00F8-#x02FF] |
[#x0370-#x037D] | [#x037F-#x1FFF]
                                                        | [#x200C-#x200D] |
[#x2070-#x218F] | [#x2C00-#x2FEF]
                                                        | [#x3001-#xD7FF] |
[#xF900-#xFDCF] | [#xFDF0-#xFFFD]
                                                        | [#x10000-#xEFFFF]    ;

 //I changed the following grammar  to the NumericLiteral borrowed from  N3
Grammar
 NumericLiteral              :              DECIMAL  |  DOUBLE  |     INTEGER
;  //   | RATIONAL;

//RATIONAL : INTEGER  '/'  UNSIGNEDINT ;
fragment DECIMAL : INTEGER '.' [0-9]* ;
fragment DOUBLE : INTEGER ('.' UNSIGNEDINT)? ([eE][-+]?INTEGER) ;
fragment INTEGER : [-+]? UNSIGNEDINT ;
fragment UNSIGNEDINT : [0-9]+;

//------------------------------------------------------------------
//whitespace and comments.
WS : [ \t\r\n]+ -> skip ; // skip spaces, tabs, newlines

//comment
//COMMENT : '//' .*? ('\n' | '\r' | '\n\r') -> skip ;
//MLCOMMENT:   '/*' (~('*'))* '*/' -> skip ;
```

Appendix E

Port Clearance Rules and Facts Case Study

This Appendix contains the complete formalization and translation of Port Clearance Rules and facts of chapter 10. The case study is based on a challenge from Decision Management (DM) community. Every challenge consists of a problem to be solved using business rules and decision management systems. This case study is based on the DM 'challenge of March 2016' which consists of creating decision models from the structured text English of Port Clearance Rules.

The full formalization and translation of port clearance rules and facts in chapter 10 for N3, POSL, SWRL, RuleML and RIF are shown below.

N3:

The following shows the formalization of Port Clearance Rules in N3 language:

```
# Rule 2 : An unloaded ship may only enter a Dutch port if the ship complies with the requirements of
the Inspection for unloaded ships.


{ ?s :MayEnterDutchPortUnloaded log:Truth } <= { ?s :CompliesInspectionRequirementsUnloaded
log:Truth } .

# Rule 3: A ship must comply with the requirements of the Inspection for unloaded ships if the ship
complies with all of the following:
# a) the ship meets the safety requirements for unloaded ships;
# b) the ship has a certificate of registry that is valid.


{           ?s :CompliesInspectionRequirementsUnloaded log:Truth }
<=
 { ?s :HasValidCertificate log:Truth . ?s :MeetsSafetyRequirementsUnloaded log:Truth .  } .

# Rule 10: A ship's certificate of registry must be considered valid if the date up to which the
registration is valid of the certificate of registry is after the current date.

           { ?s :HasValidCertificate log:Truth}
            <=
```

```
{
                ?s rdf:Type :ship .
                ?s :registryExpirationDate ?e .
                ?d time:currentDate ?cd .
                ?e math:greaterThan ?cd .
        } .

#Rule 8: A ship only meets the safety requirements for small unloaded ships if the ship complies with all
of the following:
#a) the ship is categorized as small;
#b) the hold of the ship is clean.
#Rule 8 (includes disjunct of original Rule 6)

        { ?s :MeetsSafetyRequirementsUnloaded log:Truth . }
        <=
        {
                ?s rdf:Type :ship ;
                  :size :small ;
                  :hold [ rdf:Type :shipHold ;  :status :clean] .
        }.

#Rule 7: A ship only meets the safety requirements for large unloaded ships if the ship complies with all
of the following:
# a) the ship is categorized as large;
# b) the hold of the ship is clean;
# c) the hold of the ship is double hulled.

#Rule 7 (includes disjunct of original Rule 6)


        { ?s :MeetsSafetyRequirementsUnloaded log:Truth }
        <=
        {
                ?s rdf:Type :ship ;
                  :size :large ;
                  :hold [ rdf:Type :shipHold ; :status :clean ; :hull :double ].
        }.

#Rule 9: A ship must be categorized as small if the total length of the ship is less than 80 meters.


        { ?s :size :small }
        <=
        {
                ?s rdf:Type :ship.
                ?s :totalLength ?l .
                ?l math:lessThan 80 .
        }.

#Rule 4: A ship must be categorized as large if the total length of the ship is at least 80 meters.
```

```
{ ?s :size :large }
        <=
        {
                ?s rdf:Type :ship.
                ?s :totalLength ?l .
                ?l math:greaterThan 80 .
        }.


# Rule 1&5 (combines Rule 1 and Rule 5)
# Rule 1: The hold of a ship must be considered clean if the hold does not contain remainders of cargo.
# Rule 5: A ship's hold contains remainders of cargo if the residual cargo measurement is higher than
0.5 mg dry weight per cm2.
# Negation is eliminated by propagation into Rule 5's condition, where the negated math:greaterThan
call is simplified to a math:lessEq call


        {?h :status :clean }
        <=
        {
                ?h rdf:Type :shipHold.
                ?h :residualCargoMeasurement ?c .
                ?c math:LessEq 0.5 .
        }.
```

The following shows the definition of port clearance facts in N3. There are fifteen facts including fourteen ship facts and a 'Date' fact:

```
                        #-------------------------------------facts----------------------

#Ship facts (No or Yes refer to answers for queries  with :ship1, :ship2, ... as arguments)


today time:currentDate "2018-01-21" .

# Ship 1 - No, registry has expired
:ship1 rdf:Type :ship ;
            :registryExpirationDate "2017-01-01"  ;
      :totalLength 20 ;
      :hold h1 .
           h1  rdf:Type :ShipHold ;
                     :residualCargoMeasurement 0.2 ;
                     :hull :single  .


# Ship 2 - Yes, registry is valid
:ship2  rdf:Type :ship ;
              :registryExpirationDate "2020-01-01"  ;
              :totalLength 20 ;
                     :hold h2 .
           h2         rdf:Type :ShipHold ;
                     :residualCargoMeasurement 0.2 ;
                     :hull :single  .


# Distinction for :residualCargoMeasurement
#Ship 3 - No, hold not clean
:ship3  rdf:Type :ship ;
                     :registryExpirationDate "2020-01-01"  ;
                     :totalLength 70 ;
                     :hold h3 .
           h3                  rdf:Type :ShipHold ;
                               :residualCargoMeasurement 0.6 ;
                               :hull :single  .

# Ship 4 - Yes, hold clean (qualitatively the same as for Ship 2)
:ship4  rdf:Type :ship ;
              :registryExpirationDate "2020-01-01"  ;
              :totalLength 70 ;
                     :hold h4 .
           h4                  rdf:Type :ShipHold ;
                               :residualCargoMeasurement 0.4 ;
                               :hull :single .


#Distinctions for :residualCargoMeasurement and :hull
#Ship 5 - No, hold not clean
:ship5  rdf:Type :ship ;
              :registryExpirationDate "2020-01-01"  ;
              :totalLength 90 ;
```

```
                :hold h5 .
        h5              rdf:Type :ShipHold ;
                        :residualCargoMeasurement 0.6 ;
                        :hull :double  .

#Ship 6 - No, size large yet hold single-hulled
:ship6  rdf:Type :ship ;
            :registryExpirationDate "2020-01-01"  ;
            :totalLength 90 ;
                :hold h6 .
        h6              rdf:Type :ShipHold ;
                        :residualCargoMeasurement 0.4 ;
                        :hull :single  .

#Ship 7 - Yes, hold clean and double-hulled
:ship7  rdf:Type :ship ;
            :registryExpirationDate "2020-01-01"  ;
            :totalLength 90 ;
                :hold h7 .
        h7              rdf:Type :ShipHold ;
                        :residualCargoMeasurement 0.4 ;
                        :hull :double .


#Facts with multiple reasons for No or Yes
#Three reasons for No
#Ship 8 - No, registry expired, hold not clean, and size large yet hold single-hulled
:ship8  rdf:Type :ship ;
            :registryExpirationDate "2017-01-01"  ;
            :totalLength 90 ;
                :hold h8 .
        h8              rdf:Type :ShipHold ;
                        :residualCargoMeasurement 0.9 ;
                        :hull :single  .

#Two reasons for No
#Ship 9 - No, hold not clean and size large yet hold single-hulled
:ship9  rdf:Type :ship ;
            :registryExpirationDate "2020-01-01"  ;
            :totalLength 90 ;
                :hold h9 .
        h9              rdf:Type :ShipHold ;
                        :residualCargoMeasurement 0.9 ;
                        :hull :single .

#Ship 10 - No, registry expired and size large yet hold single-hulled
:ship10  rdf:Type :ship ;
            :registryExpirationDate "2017-01-01"  ;
            :totalLength 90 ;
                :hold h10 .
        h10             rdf:Type :ShipHold ;
                        :residualCargoMeasurement 0.2 ;
```

```
                        :hull :single .

#Ship 11 - No, registry expired and hold not clean
:ship11  rdf:Type :ship ;
            :registryExpirationDate "2017-01-01"  ;
            :totalLength 90 ;
                    :hold h11 .
         h11                rdf:Type :ShipHold ;
                            :residualCargoMeasurement 0.9 ;
                            :hull :double .

#Two reasons for Yes
#Ship 12 - Yes, size small nevertheless hold double-hulled
:ship12  rdf:Type :ship ;
            :registryExpirationDate "2020-01-01"  ;
            :totalLength 60 ;
                    :hold h12 .
         h12                rdf:Type :ShipHold ;
                            :residualCargoMeasurement 0.1 ;
                            :hull :double .

#Facts probing special cases
#Ship 13 - No, large ship must have some (a double) hull
:ship13  rdf:Type :ship ;
            :registryExpirationDate "2020-01-01"  ;
            :totalLength 120 ;
                    :hold h13 .
         h13                rdf:Type :ShipHold ;
                            :residualCargoMeasurement 0.2 .

#Ship 14 - Yes, date, length, and measurement are at the threshold
:ship14  rdf:Type :ship ;
            :registryExpirationDate "2018-01-21"  ;
            :totalLength 80 ;
                    :hold h14 .
         h14                rdf:Type :ShipHold ;
                            :residualCargoMeasurement 0.5 ;
                            :hull :double  .
```

After defining Port Clearance Rules and facts in N3, they are translated to RIF using the framework developed in this thesis. The following shows these rules and facts in RIF after being translated using the framework:

```
Document(
Base(<https://dmcommunity.org/>)
Prefix(local  <https://dmcommunity.org/challenge/challenge-march-2016#>)
Prefix(log <http://www.w3.org/2000/10/swap/log#>)
Prefix(math <http://www.w3.org/2000/10/swap/math#>)
Prefix(time <http://www.w3.org/2000/10/swap/time#>)
Prefix(rdf <http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Group(

Forall  ?s (
          local:MayEnterDutchPortUnloaded( ?s )
          :-
          local:CompliesInspectionRequirementsUnloaded( ?s )
)

Forall  ?s (
          local:CompliesInspectionRequirementsUnloaded( ?s )
          :-
          And(
                    local:HasValidCertificate( ?s )
                    local:MeetsSafetyRequirementsUnloaded( ?s )
          )
)

Forall  ?cd  ?s  ?d  ?e (
          local:HasValidCertificate( ?s )
          :-
          And(
                    local:ship( ?s )
                    local:registryExpirationDate( ?s ?e )
                    time:currentDate( ?d ?cd )
                    math:greaterThan( ?e ?cd )
          )
)

Forall  ?s  ?x1 (
          local:MeetsSafetyRequirementsUnloaded( ?s )
          :-
          And(
                    local:ship( ?s )
                    local:size( ?s local:small )
                    local:hold( ?s ?x1 )
                    local:shipHold( ?x1 )
                    local:status( ?x1 local:clean )
          )
)

Forall  ?s  ?x2 (
          local:MeetsSafetyRequirementsUnloaded( ?s )
          :-
          And(
                    local:ship( ?s )
```

```
                local:size( ?s local:large )
                local:hold( ?s ?x2 )
                local:shipHold( ?x2 )
                local:status( ?x2 local:clean )
                local:hull( ?x2 local:double )
        )
)

Forall  ?s  ?l (
        local:size( ?s local:small )
        :-
        And(
                local:ship( ?s )
                local:totalLength( ?s ?l )
                math:lessThan( ?l 80 )
        )
)

Forall  ?s  ?l (
        local:size( ?s local:large )
        :-
        And(
                local:ship( ?s )
                local:totalLength( ?s ?l )
                math:greaterThan( ?l 80 )
        )
)

Forall  ?c  ?h (
        local:status( ?h local:clean )
        :-
        And(
                local:shipHold( ?h )
                local:residualCargoMeasurement( ?h ?c )
                math:LessEq( ?c 0.5 )
        )
)



time:currentDate( today "2018-01-21" )


local:ship( local:ship1 )
local:registryExpirationDate( local:ship1 "2017-01-01" )
local:totalLength( local:ship1 20 )
local:hold( local:ship1 h1 )
local:ShipHold( h1 )
local:residualCargoMeasurement( h1 0.2 )
local:hull( h1 local:single )
```

```
local:ship( local:ship2 )
local:registryExpirationDate( local:ship2 "2020-01-01" )
local:totalLength( local:ship2 20 )
local:hold( local:ship2 h2 )
local:ShipHold( h2 )
local:residualCargoMeasurement( h2 0.2 )
local:hull( h2 local:single )


local:ship( local:ship3 )
local:registryExpirationDate( local:ship3 "2020-01-01" )
local:totalLength( local:ship3 70 )
local:hold( local:ship3 h3 )
local:ShipHold( h3 )
local:residualCargoMeasurement( h3 0.6 )
local:hull( h3 local:single )


local:ship( local:ship4 )
local:registryExpirationDate( local:ship4 "2020-01-01" )
local:totalLength( local:ship4 70 )
local:hold( local:ship4 h4 )
local:ShipHold( h4 )
local:residualCargoMeasurement( h4 0.4 )
local:hull( h4 local:single )


local:ship( local:ship5 )
local:registryExpirationDate( local:ship5 "2020-01-01" )
local:totalLength( local:ship5 90 )
local:hold( local:ship5 h5 )
local:ShipHold( h5 )
local:residualCargoMeasurement( h5 0.6 )
local:hull( h5 local:double )


local:ship( local:ship6 )
local:registryExpirationDate( local:ship6 "2020-01-01" )
local:totalLength( local:ship6 90 )
local:hold( local:ship6 h6 )
local:ShipHold( h6 )
local:residualCargoMeasurement( h6 0.4 )
local:hull( h6 local:single )


local:ship( local:ship7 )
local:registryExpirationDate( local:ship7 "2020-01-01" )
local:totalLength( local:ship7 90 )
local:hold( local:ship7 h7 )
local:ShipHold( h7 )
local:residualCargoMeasurement( h7 0.4 )
local:hull( h7 local:double )
```

```
local:ship( local:ship8 )
local:registryExpirationDate( local:ship8 "2017-01-01" )
local:totalLength( local:ship8 90 )
local:hold( local:ship8 h8 )
local:ShipHold( h8 )
local:residualCargoMeasurement( h8 0.9 )
local:hull( h8 local:single )


local:ship( local:ship9 )
local:registryExpirationDate( local:ship9 "2020-01-01" )
local:totalLength( local:ship9 90 )
local:hold( local:ship9 h9 )
local:ShipHold( h9 )
local:residualCargoMeasurement( h9 0.9 )
local:hull( h9 local:single )


local:ship( local:ship10 )
local:registryExpirationDate( local:ship10 "2017-01-01" )
local:totalLength( local:ship10 90 )
local:hold( local:ship10 h10 )
local:ShipHold( h10 )
local:residualCargoMeasurement( h10 0.2 )
local:hull( h10 local:single )


local:ship( local:ship11 )
local:registryExpirationDate( local:ship11 "2017-01-01" )
local:totalLength( local:ship11 90 )
local:hold( local:ship11 h11 )
local:ShipHold( h11 )
local:residualCargoMeasurement( h11 0.9 )
local:hull( h11 local:double )


local:ship( local:ship12 )
local:registryExpirationDate( local:ship12 "2020-01-01" )
local:totalLength( local:ship12 60 )
local:hold( local:ship12 h12 )
local:ShipHold( h12 )
local:residualCargoMeasurement( h12 0.1 )
local:hull( h12 local:double )


local:ship( local:ship13 )
local:registryExpirationDate( local:ship13 "2020-01-01" )
local:totalLength( local:ship13 120 )
local:hold( local:ship13 h13 )
local:ShipHold( h13 )
local:residualCargoMeasurement( h13 0.2 )
```

```
local:ship( local:ship14 )
local:registryExpirationDate( local:ship14 "2018-01-21" )
local:totalLength( local:ship14 80 )
local:hold( local:ship14 h14 )
local:ShipHold( h14 )
local:residualCargoMeasurement( h14 0.5 )
local:hull( h14 local:double )

))
```

POSL:

The following defines the Port Clearance Rules in POSL language:

```
% Rule 2 : An unloaded ship may only enter a Dutch port if the ship complies with the requirements of the Inspection for unloaded ships.

MayEnterDutchPortUnloaded(?s)  :-  CompliesInspectionRequirementsUnloaded(?s)  .


% Rule 3: A ship must comply with the requirements of the Inspection for unloaded ships if the ship complies with all of the following:
% a) the ship meets the safety requirements for unloaded ships;
% b) the ship has a certificate of registry that is valid.

CompliesInspectionRequirementsUnloaded(?s)          :-
            HasValidCertificate(?s) , MeetsSafetyRequirementsUnloaded(?s).


% Rule 10: A ship's certificate of registry must be considered valid if the date up to which the registration is valid of the certificate of registry is after the current date.

HasValidCertificate(?s)  :-
  registryExpirationDate(?s:ship, ?e) ,          currentDate(?cd) ,          greaterThanOrEqual(?e, ?cd)  .


%Rule 8: A ship only meets the safety requirements for small unloaded ships if the ship complies with all of the following:
%a) the ship is categorized as small;
%b) the hold of the ship is clean.
%Rule 8 (includes disjunct of original Rule 6)

MeetsSafetyRequirementsUnloaded(?s)        :-
size(?s:ship , "small") , hold(?s:ship , ?h:shipHold) , status(?h:shipHold , "clean").
```

```
%Rule 7: A ship only meets the safety requirements for large unloaded ships if the ship complies with
all of the following:
% a) the ship is categorized as large;
% b) the hold of the ship is clean;
% c) the hold of the ship is "double" hulled.
%Rule 7 (includes disjunct of original Rule 6)

MeetsSafetyRequirementsUnloaded(?s) :-
size(?s:ship ,"large") , hold(?s:ship , ?h:shipHold) , status(?h:shipHold , "clean") , hull(?h:shipHold ,
"double") .


%Rule 9: A ship must be categorized as small if the total length of the ship is less than 80 meters.




size(?s:ship , "small")        :-
            totalLength(?s:ship , ?l) , lessThan(?l , 80) .

%Rule 4: A ship must be categorized as large if the total length of the ship is at least 80 meters.

size(?s:ship , "large")        :-
            totalLength(?s:ship , ?l) , greaterThanOrEqual(?l , 80) .

% Rule 1&5 (combines Rule 1 and Rule 5)
% Rule 1: The hold of a ship must be considered clean if the hold does not contain remainders of cargo.
% Rule 5: A ship's hold contains remainders of cargo if the residual cargo measurement is higher than
0.5 mg dry weight per cm2.
% Negation is eliminated by propagation into Rule 5's condition, where the negated
math:greaterThanOrEqual call is simplified to a math:lessEq call

 status(?h:shipHold , "clean")          :-
            residualCargoMeasurement(?h:shipHold , ?c) , LessEq(?c , 0.5 ) .



%--------------------------------------facts-----------------------
%Ship facts (No or Yes refer to answers for queries  with :ship1, :ship2, ... as arguments)


/% the two above facts about ship1 (and the other ships here) can also be written as below: %/
ship(ship1[
              registryExpirationDate -> "2017-01-01"  ;
      totalLength -> 20 ;
      hold->h1[
                        residualCargoMeasurement -> 0.2 ;
                        hull -> "single" ]
]).
ShipHold(h1).
```

```
% Ship 2 - Yes, registry is valid
ship(ship2[
            registryExpirationDate -> "2020-01-01"  ;
      totalLength -> 20 ;
      hold->h2[
                  residualCargoMeasurement -> 0.2 ;
                  hull -> "single" ]
]).
ShipHold(h2).

% Distinction for :residualCargoMeasurement
ship(ship3[
            registryExpirationDate -> "2020-01-01"  ;
      totalLength -> 70 ;
      hold->h3[
                  residualCargoMeasurement -> 0.6 ;
                  hull -> "single" ]


]).
ShipHold(h3).

% Ship 4 - Yes, hold clean (qualitatively the same as for Ship 2)
ship(ship4[
            registryExpirationDate -> "2020-01-01"  ;
      totalLength -> 70 ;
      hold->h4[
                  residualCargoMeasurement -> 0.4 ;
                  hull -> "single" ]
]).
ShipHold(h4).

%Distinctions for :residualCargoMeasurement and :hull
ship(ship5[
            registryExpirationDate -> "2020-01-01"  ;
      totalLength -> 90 ;
      hold->h5[
                  residualCargoMeasurement -> 0.6 ;
                  hull -> "double" ]
]).
ShipHold(h5).


%Ship 6 - No, size large yet hold single-hulled
ship(ship6[
            registryExpirationDate -> "2020-01-01"  ;
      totalLength -> 90 ;
      hold->h6[
                  residualCargoMeasurement -> 0.4 ;
                  hull -> "single" ]
]).
ShipHold(h6).
```

```
%Ship 7 - Yes, hold clean and double-hulled
ship(ship7[
            registryExpirationDate -> "2020-01-01"  ;
      totalLength -> 90 ;
      hold->h7[
                  residualCargoMeasurement -> 0.4 ;
                  hull -> "double" ]
]).
ShipHold(h7).

%Facts with multiple reasons for No or Yes
%Three reasons for No
%Ship 8 - No, registry expired, hold not clean, and size large yet hold single-hulled
ship(ship8[
            registryExpirationDate -> "2017-01-01"  ;
      totalLength -> 90 ;
      hold->h8[


                  residualCargoMeasurement -> 0.9 ;
                  hull -> "single" ]
]).
ShipHold(h8).

%Two reasons for No
%Ship 9 - No, hold not clean and size large yet hold single-hulled
ship(ship9[
            registryExpirationDate -> "2020-01-01"  ;
      totalLength -> 90 ;
      hold->h9[
                  residualCargoMeasurement -> 0.9 ;
                  hull -> "single" ]
]).
ShipHold(h9).

%Ship 10 - No, registry expired and size large yet hold single-hulled
ship(ship10[
            registryExpirationDate -> "2017-01-01"  ;
      totalLength -> 90 ;
      hold->h10[
                  residualCargoMeasurement -> 0.2 ;
                  hull -> "single" ]
]).
ShipHold(h10).


%Ship 11 - No, registry expired and hold not clean
ship(ship11[
            registryExpirationDate -> "2017-01-01"  ;
      totalLength -> 90 ;
      hold->h11[
                  residualCargoMeasurement -> 0.9 ;
```

```
                    hull -> "double" ]
]).
ShipHold(h11).


%Two reasons for Yes
%Ship 12 - Yes, size small nevertheless hold double-hulled
ship(ship12[
            registryExpirationDate -> "2020-01-01"  ;
      totalLength -> 60 ;
      hold->h12[
                    residualCargoMeasurement -> 0.1 ;
                    hull -> "double" ]
]).
ShipHold(h12).

%Facts probing special cases
%Ship 13 - No, large ship must have some (a double) hull
ship(ship13[
            registryExpirationDate -> "2020-01-01"  ;
      totalLength -> 120 ;
      hold->h13[
                    residualCargoMeasurement -> 0.2
            ]
]).
ShipHold(h13).


%Ship 14 - Yes, date, length, and measurement are at the threshold
ship(ship14[
            registryExpirationDate -> "2018-01-21"  ;
      totalLength -> 80 ;
      hold->h14[
                    residualCargoMeasurement -> 0.5 ;
                    hull -> "double" ]
]).
ShipHold(h14).
```

The following shows the Port Clearance Rules and facts in RIF after being translated from POSL using the framework:

```
Document(
Group(
	Forall ?s(
		MayEnterDutchPortUnloaded( ?s) :- CompliesInspectionRequirementsUnloaded( ?s)
	)

	Forall ?s(
		CompliesInspectionRequirementsUnloaded( ?s) :-
	And(
		HasValidCertificate( ?s)
		MeetsSafetyRequirementsUnloaded( ?s)
		)
	)

Forall ?cd ?s ?e(
HasValidCertificate( ?s) :-
And(
	registryExpirationDate( ship(?s) ?e)
	currentDate( ?cd)
	greaterThanOrEqual( ?e ?cd)
)
)


Forall ?s ?h(
MeetsSafetyRequirementsUnloaded( ?s) :-
And(
	size( ship(?s) "small")
	hold( ship(?s) shipHold(?h))
	status( shipHold(?h) "clean")
)
)
Forall ?s ?h(
MeetsSafetyRequirementsUnloaded( ?s) :-
And(
	size( ship(?s) "large")
	hold( ship(?s) shipHold(?h))
	status( shipHold(?h) "clean")
	hull( shipHold(?h) "double")
)
)
Forall ?s ?l(
size( ship(?s) "small") :-
And(
	totalLength( ship(?s) ?l)
	lessThan( ?l 80)
)
)
Forall ?s ?l(
size( ship(?s) "large") :-
And(
	totalLength( ship(?s) ?l)
```

```
        greaterThanOrEqual( ?l 80)
)
)
Forall ?c ?h(
status( shipHold(?h) "clean") :-
And(
        residualCargoMeasurement( shipHold(?h) ?c)
        LessEq( ?c 0.5)
)
)

ship(ship1)
registryExpirationDate(ship1 "2017-01-01")
totalLength (ship1 20)
hold(ship1 h1)
shipHold(h1)
residualCargoMeasurement (h1 0.2)
hull(h1 "single")


ship(ship2)
registryExpirationDate(ship2 "2020-01-01")
totalLength (ship2 20)
hold(ship2 h2)


shipHold(h2)
residualCargoMeasurement (h2 0.2)
hull(h2 "single")



ship(ship3)
registryExpirationDate(ship3 "2020-01-01")
totalLength (ship3 70)
hold(ship3 h3)
shipHold(h3)
residualCargoMeasurement (h3 0.6)
hull(h3 "single")



ship(ship4)
registryExpirationDate(ship4 "2020-01-01")
totalLength (ship4 70)
hold(ship4 h4)
shipHold(h4)
residualCargoMeasurement (h4 0.4)
hull(h4 "single")
```

```
ship(ship5)
registryExpirationDate(ship5 "2020-01-01")
totalLength (ship5 90)
hold(ship5 h5)
shipHold(h5)
residualCargoMeasurement (h5 0.6)
hull(h5 "double")


ship(ship6)
registryExpirationDate(ship6 "2020-01-01")
totalLength (ship6 90)
hold(ship6 h6)
shipHold(h6)
residualCargoMeasurement (h6 0.4)
hull(h6 "single")



ship(ship7)
registryExpirationDate(ship7 "2020-01-01")
totalLength (ship7 90)
hold(ship7 h7)
shipHold(h7)
residualCargoMeasurement (h7 0.4)
hull(h7 "double")


ship(ship8)
registryExpirationDate(ship8 "2017-01-01")
totalLength (ship8 90)
hold(ship8 h8)
shipHold(h8)
residualCargoMeasurement (h8 0.9)
hull(h8 "single")


ship(ship9)
registryExpirationDate(ship9 "2020-01-01")
totalLength (ship9 90)
hold(ship9 h9)
shipHold(h9)
residualCargoMeasurement (h9 0.9)
hull(h9 "single")


ship(ship10)
registryExpirationDate(ship10 "2017-01-01")
totalLength (ship10 90)
hold(ship10 h10)
shipHold(h10)
residualCargoMeasurement (h10 0.2)
```

```
hull(h10 "single")

ship(ship11)
registryExpirationDate(ship11 "2017-01-01")
totalLength (ship11 90)
hold(ship11 h11)
shipHold(h11)
residualCargoMeasurement (h11 0.9)
hull(h11 "double")


ship(ship12)
registryExpirationDate(ship12 "2020-01-01")
totalLength (ship12 60)
hold(ship12 h12)
shipHold(h12)
residualCargoMeasurement (h12 0.1)
hull(h12 "double")


ship(ship13)
registryExpirationDate(ship13 "2020-01-01")
totalLength (ship13 120)
hold(ship13 h13)

shipHold(h13)
residualCargoMeasurement (h13 0.2)

ship(ship14)
registryExpirationDate(ship14 "2018-01-21")
totalLength (ship14 80)
hold(ship14 h14)
shipHold(h14)
residualCargoMeasurement (h14 0.5)
hull(h14 "double")

))
```

Now that the Port Clearance rules and facts in POSL are translated to RIF, the obtained knowledge base can be queried. These queries are shown below:

```
//Query 1
local:MayEnterDutchPortUnloaded(local:ship1)

//Query 2
local:MayEnterDutchPortUnloaded(local:ship7)
```

```
//Query 3
local:MayEnterDutchPortUnloaded(?w)

//Query 4
local:status( h7 local:clean )

//Query 5
local:ship( local:ship7 )
local:hold( local:ship7 ?h )
local:status( ?h local:clean )

//Query 6
local:size( ship7 local:large )


//Query 7
local:size( ship7 local:large )
local:hold( local:ship7 ?h )
local:status( ?h local:clean )
local:hull( ?h local:double )

//Query 8
MeetsSafetyRequirementsUnloaded(ship7)

//Query 9
HasValidCertificate(ship7)

//Query 10
CompliesInspectionRequirementsUnloaded(ship7)

//Query 11
local:size( ship1 ?z )

//Query 12
local:size( ?s local:large )
local:hold( ?s ?h )
local:status( ?h local:clean )

//Query 13
local:HasValidCertificate(?s)
local:size( ?s local:large )

//Query 14
local:MeetsSafetyRequirementsUnloaded(?s)
local:size( ?s local:small )
```

SWRL:

The following defines the Port Clearance Rules and facts in SWRL language:

```
<?xml version='1.0' ?>
<!DOCTYPE swrlx:Ontology [
       <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
       <!ENTITY swrlb  "http://www.w3.org/2003/11/swrlb#" >
    <!ENTITY dm "https://dmcommunity.org/challenge/challenge-march-2016#">
]>

<?xml-stylesheet type="text/xsl" href="SWRL2RIF.xsl"?>
<swrlx:Ontology
  xmlns:owlx="http://www.w3.org/2003/05/owl-xml"
  xmlns:swrlx="http://www.w3.org/2003/11/swrlx"
  xmlns:ruleml="http://www.w3.org/2003/11/ruleml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2003/11/swrlx
http://www.ruleml.org/swrl/xsd/swrlx.xsd"
>
  <owlx:Annotation>
    <owlx:Documentation>Port Clearance Rules in SWRL</owlx:Documentation>
  </owlx:Annotation>
  <owlx:VersionInfo>Version 1.0, February 2018</owlx:VersionInfo>
  <!--
  Rule 2 : An unloaded ship may only enter a Dutch port if the ship complies with the
requirements of the Inspection for unloaded ships.
  Comments show the rules in POSL languages.
  MayEnterDutchPortUnloaded(?s)  :-  CompliesInspectionRequirementsUnloaded(?s)  .
  -->

<ruleml:var>s</ruleml:var>

<ruleml:imp>
  <ruleml:_rlab ruleml:href="#Rule2"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom
swrlx:property="CompliesInspectionRequirementsUnloaded">
      <ruleml:var>s</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="MayEnterDutchPortUnloaded">
      <ruleml:var>s</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>

<!--
% Rule 3: A ship must comply with the requirements of the Inspection for unloaded
ships if the ship complies with all of the following:
% a) the ship meets the safety requirements for unloaded ships;
% b) the ship has a certificate of registry that is valid.
CompliesInspectionRequirementsUnloaded(?s)  :-
       HasValidCertificate(?s) , MeetsSafetyRequirementsUnloaded(?s).

-->

<ruleml:imp>
  <ruleml:_rlab ruleml:href="#rule3"/>
  <ruleml:_body>
```

```
    <swrlx:individualPropertyAtom  swrlx:property="HasValidCertificate">
      <ruleml:var>s</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="MeetsSafetyRequirementsUnloaded">
      <ruleml:var>s</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom
swrlx:property="CompliesInspectionRequirementsUnloaded">
      <ruleml:var>s</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>


<!--
% Rule 10: A ship's certificate of registry must be considered valid if the date up
to which the registration is valid of the certificate of registry is after the
current date.
HasValidCertificate(?s)  :-
       registryExpirationDate(?s:ship, ?e) ,       currentDate(?cd) ,
       greaterThan(?e, ?cd)  .

-->

<ruleml:imp>
  <ruleml:_rlab ruleml:href="#rule10"/>
  <ruleml:_body>
        <swrlx:classAtom>
      <owlx:Class owlx:name="ship" />
      <ruleml:var>s</ruleml:var>
    </swrlx:classAtom>
    <swrlx:individualPropertyAtom  swrlx:property="registryExpirationDate">
                <ruleml:var>s</ruleml:var>
          <ruleml:var>e</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="currentDate">
      <ruleml:var>cd</ruleml:var>
    </swrlx:individualPropertyAtom>
       <swrlx:builtinAtom  swrlx:builtin="&swrlb;greaterThanOrEqual">
              <ruleml:var>cd</ruleml:var>
              <ruleml:var>e</ruleml:var>
       </swrlx:builtinAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="HasValidCertificate">
      <ruleml:var>s</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>


<!--
%Rule 8: A ship only meets the safety requirements for small unloaded ships if the
ship complies with all of the following:
%a) the ship is categorized as small;
```

```
%b) the hold of the ship is clean.
%Rule 8 (includes disjunct of original Rule 6)
MeetsSafetyRequirementsUnloaded(?s) :-
size(?s:ship , "small") , hold(?s:ship , ?h:shipHold) , status(?h:shipHold ,
"clean").

-->


<ruleml:imp>
  <ruleml:_rlab ruleml:href="#rule8"/>
  <ruleml:_body>
        <swrlx:classAtom>
      <owlx:Class owlx:name="ship" />
      <ruleml:var>s</ruleml:var>
    </swrlx:classAtom>
       <swrlx:datavaluedPropertyAtom  swrlx:property="size">
      <ruleml:var>s</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;string">small</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
       <swrlx:classAtom>
      <owlx:Class owlx:name="shipHold" />
      <ruleml:var>h</ruleml:var>
    </swrlx:classAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hold">
      <ruleml:var>s</ruleml:var>
          <ruleml:var>h</ruleml:var>
    </swrlx:individualPropertyAtom>
       <swrlx:datavaluedPropertyAtom  swrlx:property="status">
      <ruleml:var>h</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;string">clean</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="MeetsSafetyRequirementsUnloaded">
      <ruleml:var>s</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>

<!--
%Rule 7: A ship only meets the safety requirements for large unloaded ships if the
ship complies with all of the following:
% a) the ship is categorized as large;
% b) the hold of the ship is clean;
% c) the hold of the ship is double hulled.
%Rule 7 (includes disjunct of original Rule 6)

%MeetsSafetyRequirementsUnloaded(?s) :-
%size(?s:ship , "large") , hold(?s , ?h:shipHold -> [status -> "clean" ; hull  ->
"double" ]).

% rule 7 can also be written as following after Unnesting and Slotribution (see PSOA
formalization of Port Clearance Rules)
MeetsSafetyRequirementsUnloaded(?s) :-
size(?s:ship ,"large") , hold(?s:ship , ?h:shipHold) , status(?h:shipHold , "clean")
, hull(?h:shipHold , "double") .
```

```
-->


<ruleml:imp>
  <ruleml:_rlab ruleml:href="#rule7"/>
  <ruleml:_body>
        <swrlx:classAtom>
      <owlx:Class owlx:name="ship" />
      <ruleml:var>s</ruleml:var>
    </swrlx:classAtom>
       <swrlx:datavaluedPropertyAtom  swrlx:property="size">
      <ruleml:var>s</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;string">large</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
       <swrlx:classAtom>
      <owlx:Class owlx:name="shipHold" />
      <ruleml:var>h</ruleml:var>
    </swrlx:classAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hold">
      <ruleml:var>s</ruleml:var>
          <ruleml:var>h</ruleml:var>
    </swrlx:individualPropertyAtom>
       <swrlx:datavaluedPropertyAtom  swrlx:property="status">
      <ruleml:var>h</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;string">clean</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
       <swrlx:datavaluedPropertyAtom  swrlx:property="hull">
      <ruleml:var>h</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;string">double</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="MeetsSafetyRequirementsUnloaded">
      <ruleml:var>s</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>

<!--

%Rule 9: A ship must be categorized as small if the total length of the ship is less
than 80 meters.
size(?s:ship , "small")      :-
       totalLength(?s:ship , ?l) , lessThan(?l , 80) .

-->

<ruleml:imp>
  <ruleml:_rlab ruleml:href="#rule9"/>
  <ruleml:_body>
        <swrlx:classAtom>
      <owlx:Class owlx:name="ship" />
      <ruleml:var>s</ruleml:var>
    </swrlx:classAtom>
    <swrlx:individualPropertyAtom  swrlx:property="totalLength">
      <ruleml:var>s</ruleml:var>
          <ruleml:var>l</ruleml:var>
    </swrlx:individualPropertyAtom>
```

```
       <swrlx:builtinAtom  swrlx:builtin="&swrlb;lessThan">
      <ruleml:var>l</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;int">80</owlx:DataValue>
    </swrlx:builtinAtom>
  </ruleml:_body>
  <ruleml:_head>
   <swrlx:datavaluedPropertyAtom  swrlx:property="size">
      <ruleml:var>s</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;string">small</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
  </ruleml:_head>
</ruleml:imp>

<!--
%Rule 4: A ship must be categorized as large if the total length of the ship is at
least 80 meters.
size(?s:ship , "large")      :-
       totalLength(?s:ship , ?l) , greaterThan(?l , 80) .
-->

<ruleml:imp>
  <ruleml:_rlab ruleml:href="#rule4"/>
  <ruleml:_body>
        <swrlx:classAtom>
      <owlx:Class owlx:name="ship" />
      <ruleml:var>s</ruleml:var>
    </swrlx:classAtom>
    <swrlx:individualPropertyAtom  swrlx:property="totalLength">
      <ruleml:var>s</ruleml:var>
          <ruleml:var>l</ruleml:var>
    </swrlx:individualPropertyAtom>
       <swrlx:builtinAtom  swrlx:builtin="&swrlb;greaterThanOrEqual">
      <ruleml:var>l</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;int">80</owlx:DataValue>
    </swrlx:builtinAtom>
  </ruleml:_body>
  <ruleml:_head>
   <swrlx:datavaluedPropertyAtom  swrlx:property="size">
      <ruleml:var>s</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;string">large</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
  </ruleml:_head>
</ruleml:imp>

<!--

% Rule 1&5 (combines Rule 1 and Rule 5)
% Rule 1: The hold of a ship must be considered clean if the hold does not contain
remainders of cargo.
% Rule 5: A ship's hold contains remainders of cargo if the residual cargo
measurement is higher than 0.5 mg dry weight per cm2.
% Negation is eliminated by propagation into Rule 5's condition, where the negated
math:greaterThan call is simplified to a math:lessEq call
 status(?h:shipHold , "clean")      :-
       residualCargoMeasurement(?h:shipHold , ?c) , LessEq(?c , 0.5 ) .

-->
```

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#rule9"/>
  <ruleml:_body>
        <swrlx:classAtom>
      <owlx:Class owlx:name="shipHold" />
      <ruleml:var>h</ruleml:var>
    </swrlx:classAtom>
    <swrlx:individualPropertyAtom  swrlx:property="residualCargoMeasurement">
      <ruleml:var>h</ruleml:var>
          <ruleml:var>c</ruleml:var>
    </swrlx:individualPropertyAtom>
       <swrlx:builtinAtom  swrlx:builtin="&swrlb;lessThanOrEqual">
      <ruleml:var>c</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;double">0.5</owlx:DataValue>
    </swrlx:builtinAtom>
  </ruleml:_body>
  <ruleml:_head>
   <swrlx:datavaluedPropertyAtom  swrlx:property="status">
      <ruleml:var>h</ruleml:var>
      <owlx:DataValue owlx:datatype="&xsd;string">clean</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
  </ruleml:_head>
</ruleml:imp>


<!--            facts                 -->
<!--Ship facts (No or Yes refer to answers for queries  with :ship1, :ship2, ... as
arguments) -->

<!-- today time:currentDate "2018-01-21" . -->
<!--
 <swrlx:datavaluedPropertyAtom  swrlx:property="currentDate">
       <ruleml:var>today</ruleml:var>
        <owlx:DataValue owlx:datatype="&xsd;dateTime">2018-01-21</owlx:DataValue>
</swrlx:datavaluedPropertyAtom>
-->
<owlx:DataPropertyValue owlx:property="currentDate">
    <owlx:DataValue owlx:datatype="&xsd;dateTime">2018-01-21</owlx:DataValue>
</owlx:DataPropertyValue>

<!--
# Ship 1 - No, registry has expired
:ship1 rdf:Type :ship ;
          :registryExpirationDate "2017-01-01"  ;
       :totalLength "20" ;
       :hold [
              rdf:Type :ShipHold ;
              :residualCargoMeasurement "0.2" ;
              :hull :single ] .
-->
<owlx:Individual owlx:name="ship1">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
```

```
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2017-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">20</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.2</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">single</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>


<!--
# Ship 2 - Yes, registry is valid
:ship2  rdf:Type :ship ;
           :registryExpirationDate "2020-01-01"  ;
           :totalLength "20" ;
              :hold [
              rdf:Type :ShipHold ;
              :residualCargoMeasurement "0.2" ;
              :hull :single ] .
-->
<owlx:Individual owlx:name="ship2">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">202-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">20</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.2</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">single</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>
```

```
<!--
# Distinction for :residualCargoMeasurement
#Ship 3 - No, hold not clean
:ship3  rdf:Type :ship ;
               :registryExpirationDate "2020-01-01"  ;
               :totalLength "70" ;
               :hold [
                      rdf:Type :ShipHold ;
                      :residualCargoMeasurement "0.6" ;
                      :hull :single ] .
-->
<owlx:Individual owlx:name="ship3">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2020-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">70</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.6</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">single</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

<!--
# Ship 4 - Yes, hold clean (qualitatively the same as for Ship 2)
:ship4  rdf:Type :ship ;
           :registryExpirationDate "2020-01-01"  ;
           :totalLength "70" ;
              :hold [
                      rdf:Type :ShipHold ;
                      :residualCargoMeasurement "0.4" ;
                      :hull :single ] .
-->
<owlx:Individual owlx:name="ship4">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2020-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
```

```
      <owlx:DataPropertyValue owlx:property="totalLength">
            <owlx:DataValue owlx:datatype="&xsd;int">70</owlx:DataValue>
      </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.4</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">single</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

<!--
#Distinctions for :residualCargoMeasurement and :hull
#Ship 5 - No, hold not clean
:ship5  rdf:Type :ship ;
           :registryExpirationDate "2020-01-01"  ;
           :totalLength "90" ;
              :hold [
                     rdf:Type :ShipHold ;
                     :residualCargoMeasurement "0.6" ;
                     :hull :double ] .
-->
<owlx:Individual owlx:name="ship5">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
            <owlx:DataValue owlx:datatype="&xsd;dateTime">2020-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
            <owlx:DataValue owlx:datatype="&xsd;int">90</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.6</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">double</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

<!--
#Ship 6 - No, size large yet hold single-hulled
:ship6  rdf:Type :ship ;
           :registryExpirationDate "2020-01-01"  ;
```

```
           :totalLength "90" ;
              :hold [
                     rdf:Type :ShipHold ;
                     :residualCargoMeasurement "0.4" ;
                     :hull :single ] .
-->
<owlx:Individual owlx:name="ship6">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2020-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">90</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
       <owlx:DataValue owlx:datatype="&xsd;double">0.4</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
       <owlx:DataValue owlx:datatype="&xsd;string">single</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

<!--
#Ship 7 - Yes, hold clean and double-hulled
:ship7  rdf:Type :ship ;
           :registryExpirationDate "2020-01-01"  ;
           :totalLength "90" ;
              :hold [
                     rdf:Type :ShipHold ;
                     :residualCargoMeasurement "0.4" ;
                     :hull :double ] .
-->
<owlx:Individual owlx:name="ship7">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2020-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">90</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
```

```
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.4</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">double</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

<!--
#Facts with multiple reasons for No or Yes
#Three reasons for No
#Ship 8 - No, registry expired, hold not clean, and size large yet hold single-hulled
:ship8  rdf:Type :ship ;
           :registryExpirationDate "2017-01-01"  ;
           :totalLength "90" ;
              :hold [
                     rdf:Type :ShipHold ;
                     :residualCargoMeasurement "0.9" ;
                     :hull :single ] .
-->
<owlx:Individual owlx:name="ship8">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2017-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">90</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.9</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">single</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

<!--
#Two reasons for No
#Ship 9 - No, hold not clean and size large yet hold single-hulled
:ship9  rdf:Type :ship ;
           :registryExpirationDate "2020-01-01"  ;
           :totalLength "90" ;
              :hold [
                     rdf:Type :ShipHold ;
```

```
                    :residualCargoMeasurement "0.9" ;
                    :hull :single ] .
-->
<owlx:Individual owlx:name="ship9">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2020-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">90</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
       <owlx:DataValue owlx:datatype="&xsd;double">0.9</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
       <owlx:DataValue owlx:datatype="&xsd;string">single</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

<!--
#Ship 10 - No, registry expired and size large yet hold single-hulled
:ship10  rdf:Type :ship ;
           :registryExpirationDate "2017-01-01"  ;
           :totalLength "90" ;
              :hold [
                     rdf:Type :ShipHold ;
                     :residualCargoMeasurement "0.2" ;
                     :hull :single ] .
-->
<owlx:Individual owlx:name="ship10">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2017-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">90</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
       <owlx:DataValue owlx:datatype="&xsd;double">0.2</owlx:DataValue>
```

```
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">single</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

<!--
#Ship 11 - No, registry expired and hold not clean
:ship11  rdf:Type :ship ;
           :registryExpirationDate "2017-01-01"  ;
           :totalLength "90" ;
              :hold [
                     rdf:Type :ShipHold ;
                     :residualCargoMeasurement "0.9" ;
                     :hull :double ] .
-->
<owlx:Individual owlx:name="ship11">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2017-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">90</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.9</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">double</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

<!--
#Two reasons for Yes
#Ship 12 - Yes, size small nevertheless hold double-hulled
:ship12  rdf:Type :ship ;
           :registryExpirationDate "2020-01-01"  ;
           :totalLength "60" ;
              :hold [
                     rdf:Type :ShipHold ;
                     :residualCargoMeasurement "0.1" ;
                     :hull :double ] .
-->
<owlx:Individual owlx:name="ship12">
   <owlx:type owlx:name="&dm;ship" />
```

```
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2020-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">60</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.1</owlx:DataValue>
      </owlx:DataPropertyValue>
      <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">double</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

<!--
#Facts probing special cases
#Ship 13 - No, large ship must have some (a double) hull
:ship13  rdf:Type :ship ;
           :registryExpirationDate "2020-01-01"  ;
           :totalLength "120" ;
              :hold [
                     rdf:Type :ShipHold ;
                     :residualCargoMeasurement "0.2"
                     ] .
-->
<owlx:Individual owlx:name="ship13">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2020-01-
01</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">120</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.2</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>
```

```
<!--
#Ship 14 - Yes, date, length, and measurement are at the threshold
:ship14  rdf:Type :ship ;
           :registryExpirationDate "2018-01-21"  ;
           :totalLength "80" ;
              :hold [
                     rdf:Type :ShipHold ;
                     :residualCargoMeasurement "0.5" ;
                     :hull :double ] .
-->
<owlx:Individual owlx:name="ship14">
   <owlx:type owlx:name="&dm;ship" />
   <!--<swrlx:classAtom>
      <owlx:Class owlx:name="&dm;ship" />
      <ruleml:var>ship1</ruleml:var>
    </swrlx:classAtom> -->
       <owlx:DataPropertyValue owlx:property="registryExpirationDate">
              <owlx:DataValue owlx:datatype="&xsd;dateTime">2018-01-
21</owlx:DataValue>
       </owlx:DataPropertyValue>
       <owlx:DataPropertyValue owlx:property="totalLength">
              <owlx:DataValue owlx:datatype="&xsd;int">80</owlx:DataValue>
       </owlx:DataPropertyValue>
  <owlx:ObjectPropertyValue owlx:property="hold">
  <owlx:type owlx:name="&dm;shipHold" />
    <owlx:Individual>
      <owlx:DataPropertyValue owlx:property="residualCargoMeasurement">
        <owlx:DataValue owlx:datatype="&xsd;double">0.5</owlx:DataValue>
      </owlx:DataPropertyValue>
         <owlx:DataPropertyValue owlx:property="hull">
        <owlx:DataValue owlx:datatype="&xsd;string">double</owlx:DataValue>
      </owlx:DataPropertyValue>
    </owlx:Individual>
  </owlx:ObjectPropertyValue>
</owlx:Individual>

</swrlx:Ontology>
```

The following shows the Port Clearance Rules and facts translated from SWRL to RIF XML using the framework:

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="http://www.w3.org/2007/rif#"
       xmlns:ruleml="http://ruleml.org/spec"
       xmlns:xs="http://www.w3.org/2001/XMLSchema#"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
       <payload>
              <Group>
                     <sentence>
                            <Group>
                                   <Implies>
                                          <if>
```

```
                                                <Atom>
                                                       <op>
                                                              <Const
type="http://www.w3.org/2007/rif#iri">CompliesInspectionRequirementsUnloaded
                                                              </Const>
                                                       </op>
                                                       <args ordered="yes">
                                                              <Var>s</Var>
                                                       </args>
                                                </Atom>
                                         </if>
                                         <then>
                                                <Atom>
                                                       <op>
                                                              <Const
type="http://www.w3.org/2007/rif#iri">MayEnterDutchPortUnloaded</Const>
                                                       </op>
                                                       <args ordered="yes">
                                                              <Var>s</Var>
                                                       </args>
                                                </Atom>
                                         </then>
                                  </Implies>
                                  <Implies>
                                         <if>
                                                <And>
                                                       <Atom>
                                                              <op>
                                                                     <Const
type="http://www.w3.org/2007/rif#iri">HasValidCertificate</Const>
                                                              </op>
                                                              <args
ordered="yes">
                                                                     <Var>s</Var>
                                                              </args>
                                                       </Atom>
                                                       <Atom>
                                                              <op>
                                                                     <Const
type="http://www.w3.org/2007/rif#iri">MeetsSafetyRequirementsUnloaded</Const>
                                                              </op>
                                                              <args
ordered="yes">
                                                                     <Var>s</Var>
                                                              </args>
                                                       </Atom>
                                                </And>
                                         </if>
                                         <then>
                                                <Atom>
                                                       <op>
                                                              <Const
type="http://www.w3.org/2007/rif#iri">CompliesInspectionRequirementsUnloaded
                                                              </Const>
                                                       </op>
                                                       <args ordered="yes">
                                                              <Var>s</Var>
                                                       </args>
                                                </Atom>
```

```
                                        </then>
                                </Implies>
                                <Implies>
                                        <if>
                                                <And>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                                </op>
                                                                <args
ordered="yes">
                                                                        <Var>s</Var>
                                                                        <Var>e</Var>
                                                                </args>
                                                        </Atom>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">currentDate</Const>
                                                                </op>
                                                                <args
ordered="yes">

        <Var>cd</Var>
                                                                </args>
                                                        </Atom>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">greaterThan</Const>
                                                                </op>
                                                                <args
ordered="yes">
                                                                        <Var>e</Var>

        <Var>cd</Var>
                                                                </args>
                                                        </Atom>
                                                </And>
                                        </if>
                                        <then>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">HasValidCertificate</Const>
                                                        </op>
                                                        <args ordered="yes">
                                                                <Var>s</Var>
                                                        </args>
                                                </Atom>
                                        </then>
                                </Implies>
                                <Implies>
                                        <if>
                                                <And>
                                                        <Atom>
                                                                <op>
```

```
                                                                <Const
type="http://www.w3.org/2007/rif#iri">size</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">small</Const>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
                                                                <Var>h</Var>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">status</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>h</Var>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">clean</Const>
                                                        </args>
                                                </Atom>
                                        </And>
                                </if>
                                <then>
                                        <Atom>
                                                <op>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">MeetsSafetyRequirementsUnloaded</Const>
                                                </op>
                                                <args ordered="yes">
                                                        <Var>s</Var>
                                                </args>
                                        </Atom>
                                </then>
                        </Implies>
                        <Implies>
                                <if>
                                        <And>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">size</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
```

```
                                                                <Const
type="http://www.w3.org/2007/rif#iri">large</Const>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
                                                                <Var>h</Var>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">status</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>h</Var>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">clean</Const>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>h</Var>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                        </args>
                                                </Atom>
                                        </And>
                                </if>
                                <then>
                                        <Atom>
                                                <op>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">MeetsSafetyRequirementsUnloaded</Const>
                                                </op>
                                                <args ordered="yes">
                                                        <Var>s</Var>
                                                </args>
                                        </Atom>
                                </then>
                        </Implies>
                        <Implies>
                                <if>
                                        <And>
                                                <Atom>
                                                        <op>
```

```
                                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
                                                                <Var>l</Var>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">lessThan</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>l</Var>

        <Var>80</Var>
                                                        </args>
                                                </Atom>
                                        </And>
                                </if>
                                <then>
                                        <Atom>
                                                <op>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">size</Const>
                                                </op>
                                                <args ordered="yes">
                                                        <Var>s</Var>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">small</Const>
                                                </args>
                                        </Atom>
                                </then>
                        </Implies>
                        <Implies>
                                <if>
                                        <And>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
                                                                <Var>l</Var>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">greaterThanOrEqual</Const>
                                                        </op>
                                                        <args
ordered="yes">
```

```
                                                                <Var>l</Var>
        <Var>80</Var>
                                                        </args>
                                                </Atom>
                                        </And>
                                </if>
                                <then>
                                        <Atom>
                                                <op>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">size</Const>
                                                </op>
                                                <args ordered="yes">
                                                        <Var>s</Var>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">large</Const>
                                                </args>
                                        </Atom>
                                </then>
                        </Implies>
                        <Implies>
                                <if>
                                        <And>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>h</Var>
                                                                <Var>c</Var>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">lessThanOrEqual</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>c</Var>
        <Var>0.5</Var>
                                                        </args>
                                                </Atom>
                                        </And>
                                </if>
                                <then>
                                        <Atom>
                                                <op>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">status</Const>
                                                </op>
                                                <args ordered="yes">
```

```
                                                                <Var>h</Var>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">clean</Const>
                                                        </args>
                                                </Atom>
                                        </then>
                                </Implies>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">currentDate</Const>
                                        </op>
                                        <args ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">2018-01-21</Const>
                                        </args>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship1</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2017-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">20</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h1</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.2</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
```

```
                                                    <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                                </op>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">ship2</Const>
                                                    <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                    </slot>
                                                    <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">20</Const>
                                                    </slot>
                                                    <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h2</Const>
                                                        <slot ordered="yes">
                                                            <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                            <Const
type="http://www.w3.org/2007/rif#iri">0.2</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                            <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                            <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                    </slot>
                                                </slot>
                                            </Atom>
                                            <Atom>
                                                <op>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                                </op>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">ship3</Const>
                                                    <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                    </slot>
                                                    <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">70</Const>
                                                    </slot>
                                                    <slot ordered="yes">
```

```
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h3</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.6</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship4</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">70</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h4</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.4</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
```

```
                                    <Atom>
                                        <op>
                                            <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                            <Const
type="http://www.w3.org/2007/rif#iri">ship5</Const>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">h5</Const>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">0.6</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                </slot>
                                            </slot>
                                        </slot>
                                    </Atom>
                                    <Atom>
                                        <op>
                                            <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                            <Const
type="http://www.w3.org/2007/rif#iri">ship6</Const>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
```

```
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h6</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.4</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship7</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h7</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.4</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                        </slot>
                                                </slot>
```

```
                                        </slot>
                                    </Atom>
                                    <Atom>
                                        <op>
                                            <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                            <Const
type="http://www.w3.org/2007/rif#iri">ship8</Const>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">h8</Const>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">0.9</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                </slot>
                                            </slot>
                                        </slot>
                                    </Atom>
                                    <Atom>
                                        <op>
                                            <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                            <Const
type="http://www.w3.org/2007/rif#iri">ship9</Const>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
```

```
                                                        <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h9</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.9</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship10</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2017-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h10</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.2</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
```

```
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship11</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2017-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h11</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.9</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship12</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
```

```
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">60</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h12</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.1</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship13</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">120</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h13</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.2</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
```

```
                                    <Atom>
                                        <op>
                                            <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                            <Const
type="http://www.w3.org/2007/rif#iri">ship14</Const>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">2018-01-21</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">80</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">h14</Const>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">0.5</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                </slot>
                                            </slot>
                                        </slot>
                                    </Atom>
                                </Group>
                            </sentence>
                        </Group>
                    </payload>
                </Document>
```

RuleML:

The following defines the Port Clearance Rules and facts in RuleML language:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model
href="http://deliberation.ruleml.org/1.02/relaxng/bindatalog_relaxed.rnc"?>
<!--<?xml-model href="http://deliberation.ruleml.org/1.02/xsd/bindatalog.xsd"
       type="application/xml" schematypens="http://www.w3.org/2001/XMLSchema"?> -->
<?xml-stylesheet type="text/xsl" href="UBDatalogRuleML2RIF_BLD.xsl"?>
<RuleML xmlns="http://ruleml.org/spec"
       xmlns:this="http://example.org/bindatalog.ruleml"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:dm="https://dmcommunity.org/challenge/challenge-march-2016#"
       xsi:schemaLocation="http://ruleml.org/spec
http://deliberation.ruleml.org/1.02/xsd/bindatalog.xsd">

       <Assert mapClosure="universal">
              <!--<oid><Ind>test rulebase</Ind></oid> -->
              <Rulebase>

                     <!-- Rule 2 : An unloaded ship may only enter a Dutch port if
the ship
                            complies with the requirements of the Inspection for
unloaded ships. comments
                            show the rules in POSL languages.
MayEnterDutchPortUnloaded(?s) :- CompliesInspectionRequirementsUnloaded(?s)
                            . -->

                     <Implies>
                            <then>
                                   <Atom>
                                          <op>

       <Rel>MayEnterDutchPortUnloaded</Rel>
                                          </op>
                                          <Var>s</Var>
                                   </Atom>
                            </then>
                            <if>
                                   <Atom>
                                          <op>

       <Rel>CompliesInspectionRequirementsUnloaded</Rel>
                                          </op>
                                          <Var>s</Var>
                                   </Atom>
                            </if>
                     </Implies>

                     <!-- % Rule 3: A ship must comply with the requirements of the
Inspection
                            for unloaded ships if the ship complies with all of the
following: % a) the
                            ship meets the safety requirements for unloaded ships;
% b) the ship has
                            a certificate of registry that is valid.
CompliesInspectionRequirementsUnloaded(?s)
                            :- HasValidCertificate(?s) ,
MeetsSafetyRequirementsUnloaded(?s). -->
```

```
                    <Implies>
                           <then>
                                  <Atom>
                                         <op>

       <Rel>CompliesInspectionRequirementsUnloaded</Rel>
                                         </op>
                                         <Var>s</Var>
                                  </Atom>
                           </then>
                           <if>
                                  <And>
                                         <Atom>
                                                <op>

       <Rel>HasValidCertificate</Rel>
                                                </op>
                                                <Var>s</Var>
                                         </Atom>
                                         <Atom>
                                                <op>

       <Rel>MeetsSafetyRequirementsUnloaded</Rel>
                                                </op>
                                                <Var>s</Var>
                                         </Atom>
                                  </And>
                           </if>
                    </Implies>



                    <!-- % Rule 10: A ship’s certificate of registry must be
considered valid
                           if the date up to which the registration is valid of
the certificate of registry
                           is after the current date. HasValidCertificate(?s) :-
registryExpirationDate(?s:ship,
                           ?e) , currentDate(?cd) , greaterThan(?e, ?cd) . -->

                    <Implies>
                           <then>
                                  <Atom>
                                         <op>
                                                <Rel>HasValidCertificate</Rel>
                                         </op>
                                         <Var>s</Var>
                                  </Atom>
                           </then>
                           <if>
                                  <And>
                                         <Atom>
                                                <op>

       <Rel>registryExpirationDate</Rel>
                                                </op>
                                                <Var type="dm:ship">s</Var>
                                                <Var type="xsd:dateTime">e</Var>
```

```
                                        </Atom>
                                        <Atom>
                                                <op>
                                                        <Rel>currentDate</Rel>
                                                </op>
                                                <Var type="xsd:dateTime">cd</Var>
                                        </Atom>
                                        <Atom>
                                                <op>
                                                        <Rel>greaterThan</Rel>
                                                </op>
                                                <Var type="xsd:dateTime">e</Var>
                                                <Var type="xsd:dateTime">cd</Var>
                                        </Atom>
                                </And>
                        </if>
                </Implies>

                <!-- %Rule 8: A ship only meets the safety requirements for
small unloaded
                        ships if the ship complies with all of the following:
%a) the ship is categorized
                        as small; %b) the hold of the ship is clean. %Rule 8
(includes disjunct of
                        original Rule 6) MeetsSafetyRequirementsUnloaded(?s) :-
size(?s:ship , "small")
                        , hold(?s:ship , ?h:shipHold) , status(?h:shipHold ,
"clean"). -->

                <Implies>
                        <then>
                                <Atom>
                                        <op>

        <Rel>MeetsSafetyRequirementsUnloaded</Rel>
                                        </op>
                                        <Var>s</Var>
                                </Atom>
                        </then>
                        <if>
                                <And>
                                        <Atom>
                                                <op>
                                                        <Rel>size</Rel>
                                                </op>
                                                <Var type="dm:ship">s</Var>
                                                <Ind type="xsd:string">small</Ind>
                                        </Atom>
                                        <Atom>
                                                <op>
                                                        <Rel>hold</Rel>
                                                </op>
                                                <Var type="dm:ship">s</Var>
                                                <Var type="dm:shipHold">h</Var>
                                        </Atom>
                                        <Atom>
                                                <op>
                                                        <Rel>status</Rel>
                                                </op>
```

```
                                                <Var type="dm:shipHold">h</Var>
                                                <Ind type="xsd:string">clean</Ind>
                                        </Atom>
                                </And>
                        </if>
                </Implies>


                <!-- %Rule 7: A ship only meets the safety requirements for
large unloaded
                        ships if the ship complies with all of the following: %
a) the ship is categorized
                        as large; % b) the hold of the ship is clean; % c) the
hold of the ship is
                        double hulled. %Rule 7 (includes disjunct of original
Rule 6) %MeetsSafetyRequirementsUnloaded(?s)
                        :- %size(?s:ship , "large") , hold(?s , ?h:shipHold ->
[status -> "clean"
                        ; hull -> "double" ]). % rule 7 can also be written as
following after Unnesting
                        and Slotribution (see PSOA formalization of Port
Clearance Rules) MeetsSafetyRequirementsUnloaded(?s)
                        :- size(?s:ship ,"large") , hold(?s:ship , ?h:shipHold)
, status(?h:shipHold
                        , "clean") , hull(?h:shipHold , "double") . -->

                <Implies>
                        <then>
                                <Atom>
                                        <op>

        <Rel>MeetsSafetyRequirementsUnloaded</Rel>
                                        </op>
                                        <Var>s</Var>
                                </Atom>
                        </then>
                        <if>
                                <And>
                                        <Atom>
                                                <op>
                                                        <Rel>size</Rel>
                                                </op>
                                                <Var type="dm:ship">s</Var>
                                                <Ind type="xsd:string">large</Ind>
                                        </Atom>
                                        <Atom>
                                                <op>
                                                        <Rel>hold</Rel>
                                                </op>
                                                <Var type="dm:ship">s</Var>
                                                <Var type="dm:shipHold">h</Var>
                                        </Atom>
                                        <Atom>
                                                <op>
                                                        <Rel>status</Rel>
                                                </op>
                                                <Var type="dm:shipHold">h</Var>
                                                <Ind type="xsd:string">clean</Ind>
                                        </Atom>
```

```
                                        <Atom>
                                                <op>
                                                        <Rel>hull</Rel>
                                                </op>
                                                <Var type="dm:shipHold">h</Var>
                                                <Ind
type="xsd:string">double</Ind>
                                        </Atom>
                                </And>
                        </if>
                </Implies>


                <!-- %Rule 9: A ship must be categorized as small if the total
length
                        of the ship is less than 80 meters. size(?s:ship ,
"small") :- totalLength(?s:ship
                        , ?l) , lessThan(?l , 80) . -->

                <Implies>
                        <then>
                                <Atom>
                                        <op>
                                                <Rel>size</Rel>
                                        </op>
                                        <Var type="dm:ship">s</Var>
                                        <Ind type="xsd:string">small</Ind>
                                </Atom>
                        </then>
                        <if>
                                <And>
                                        <Atom>
                                                <op>
                                                        <Rel>totalLength</Rel>
                                                </op>
                                                <Var type="dm:ship">s</Var>
                                                <Var type="xsd:int">l</Var>
                                        </Atom>
                                        <Atom>
                                                <op>
                                                        <Rel>lessThan</Rel>
                                                </op>
                                                <Var type="xsd:int">l</Var>
                                                <Var type="xsd:int">80</Var>
                                        </Atom>
                                </And>
                        </if>
                </Implies>


                <!-- %Rule 4: A ship must be categorized as large if the total
length
                        of the ship is at least 80 meters. size(?s:ship ,
"large") :- totalLength(?s:ship
                        , ?l) , greaterThan(?l , 80) . -->


                <Implies>
                        <then>
```

```
                                        <Atom>
                                                <op>
                                                        <Rel>size</Rel>
                                                </op>
                                                <Var type="dm:ship">s</Var>
                                                <Ind type="xsd:string">large</Ind>
                                        </Atom>
                                </then>
                                <if>
                                        <And>
                                                <Atom>
                                                        <op>
                                                                <Rel>totalLength</Rel>
                                                        </op>
                                                        <Var type="dm:ship">s</Var>
                                                        <Var type="xsd:int">l</Var>
                                                </Atom>
                                                <Atom>
                                                        <op>

        <Rel>greaterThanOrEqual</Rel>

                                                        </op>
                                                        <Var type="xsd:int">l</Var>
                                                        <Var type="xsd:int">80</Var>
                                                </Atom>
                                        </And>
                                </if>
                        </Implies>


                        <!-- % Rule 1&5 (combines Rule 1 and Rule 5) % Rule 1: The hold
of a ship
                                must be considered clean if the hold does not contain
remainders of cargo.
                                % Rule 5: A ship's hold contains remainders of cargo if
the residual cargo
                                measurement is higher than 0.5 mg dry weight per cm2. %
Negation is eliminated
                                by propagation into Rule 5's condition, where the
negated math:greaterThan
                                call is simplified to a math:lessEq call
status(?h:shipHold , "clean") :-
                                residualCargoMeasurement(?h:shipHold , ?c) , LessEq(?c
, 0.5 ) . -->

                        <Implies>
                                <then>
                                        <Atom>
                                                <op>
                                                        <Rel>status</Rel>
                                                </op>
                                                <Var type="dm:shipHold">h</Var>
                                                <Ind type="xsd:string">clean</Ind>
                                        </Atom>
                                </then>
                                <if>
                                        <And>
                                                <Atom>
                                                        <op>
```

```
        <Rel>residualCargoMeasurement</Rel>
                                                        </op>
                                                        <Var type="dm:shipHold">h</Var>
                                                        <Var type="xsd:double">c</Var>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Rel>lessThanOrEqual</Rel>
                                                        </op>
                                                        <Var type="xsd:double">c</Var>
                                                        <Var type="xsd:double">0.5</Var>
                                                </Atom>
                                        </And>
                                </if>
                        </Implies>

                        <!--
################################facts#################################
                                #Ship facts (No or Yes refer to answers for queries
with :ship1, :ship2,
                                ... as arguments) -->

                        <!-- :currentDate(phys:date(2017 5 6)) % Uncomment for fixed
date (reproducibility) -->
                        <Atom closure="universal">
                                <Rel>currentDate</Rel>
                                <Data xsi:type="xs:dateTime">2018-01-21</Data>
                        </Atom>

                        <!-- % Facts covering all cases with qualitative slot-filler
distinctions
                                % Explanatory comments for Yes answers focus on the
most relevant slots %
                                Distinction for :registryExpirationDate % Ship 1 - No,
registry has expired
                                :ship1#:Ship(:registryExpirationDate->phys:date(2017 5
1) :totalLength->20
                                :hold->:h1#:ShipHold(:residualCargoMeasurement->0.2
:hull->:single)) -->

                        <Atom>
                                <op>
                                        <Rel>Ship</Rel>
                                </op>
                                <slot>
                                        <Ind type="dm:ship">ship1</Ind>
                                        <Plex>
                                                <!--<slot><Ind
type="xsd:string">registryExpirationDate</Ind><Data
                                                        xsi:type="xs:dateTime">2017-01-
01</Data></slot> -->
                                                <slot>
                                                        <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                        <Ind type="xsd:dateTime">2017-01-
01</Ind>
                                                </slot>
                                                <slot>
```

```
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">20</Data>
                                        </slot>
                                        <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                        <Ind
type="dm:shipHold">h1</Ind>
                                                        <slot>
                                                                <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                <Ind
type="xsd:double">0.2</Ind>
                                                        </slot>
                                                        <slot>
                                                                <Ind
type="xsd:string">hull</Ind>
                                                                <Ind
type="xsd:string">single</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
                        </slot>
                </Atom>

                <!-- % Ship 2 - Yes, registry is valid
:ship2#:Ship(:registryExpirationDate->phys:date(2017
                        10 1) :totalLength->20 :hold-
>:h2#:ShipHold(:residualCargoMeasurement->0.2
                        :hull->:single)) -->

                <Atom>
                        <op>
                                <Rel>Ship</Rel>
                        </op>
                        <slot>
                                <Ind type="dm:ship">ship2</Ind>
                                <Plex>
                                        <!--<slot><Ind
type="xsd:string">registryExpirationDate</Ind><Data
                                                xsi:type="xs:dateTime">2017-01-
01</Data></slot> -->
                                        <slot>
                                                <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                <Ind type="xsd:dateTime">2020-01-
01</Ind>
                                        </slot>
                                        <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">20</Data>
                                        </slot>
                                        <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
```

```
                                                        <Ind
type="dm:shipHold">h2</Ind>
                                                        <slot>
                                                                <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                <Ind
type="xsd:double">0.2</Ind>
                                                        </slot>
                                                        <slot>
                                                                <Ind
type="xsd:string">hull</Ind>
                                                                <Ind
type="xsd:string">single</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
                        </slot>
                </Atom>

                <!-- % Distinction for :residualCargoMeasurement % Ship 3 - No,
hold not
                        clean :ship3#:Ship(:registryExpirationDate-
>phys:date(2020 1 1) :totalLength->70
                        :hold->:h3#:ShipHold(:residualCargoMeasurement->0.6
:hull->:single)) -->

                <Atom>
                        <op>
                                <Rel>Ship</Rel>
                        </op>
                        <slot>
                                <Ind type="dm:ship">ship3</Ind>
                                <Plex>
                                        <slot>
                                                <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                <Ind type="xsd:dateTime">2020-01-
01</Ind>
                                        </slot>
                                        <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">70</Data>
                                        </slot>
                                        <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                        <Ind
type="dm:shipHold">h3</Ind>
                                                        <slot>
                                                                <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                <Ind
type="xsd:double">0.6</Ind>
                                                        </slot>
                                                        <slot>
                                                                <Ind
type="xsd:string">hull</Ind>
```

```
                                                                <Ind
type="xsd:string">single</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
                        </slot>
                </Atom>

                <!-- % Ship 4 - Yes, hold clean (qualitatively the same as for
Ship 2)
                        :ship4#:Ship(:registryExpirationDate->phys:date(2020 1
1 ) :totalLength->70
                        :hold->:h4#:ShipHold(:residualCargoMeasurement->0.4
:hull->:single)) -->

                <Atom>
                        <op>
                                <Rel>Ship</Rel>
                        </op>
                        <slot>
                                <Ind type="dm:ship">ship4</Ind>
                                <Plex>
                                        <slot>
                                                <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                <Ind type="xsd:dateTime">2020-01-
01</Ind>
                                        </slot>
                                        <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">70</Data>
                                        </slot>
                                        <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                        <Ind
type="dm:shipHold">h4</Ind>
                                                        <slot>
                                                                <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                <Ind
type="xsd:double">0.4</Ind>
                                                        </slot>
                                                        <slot>
                                                                <Ind
type="xsd:string">hull</Ind>
                                                                <Ind
type="xsd:string">single</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
                        </slot>
                </Atom>

                <!-- % Distinctions for :residualCargoMeasurement and :hull %
Ship 5 -
```

```
                        No, hold not clean
:ship5#:Ship(:registryExpirationDate->phys:date(2020 1
                        1) :totalLength->90 :hold-
>:h5#:ShipHold(:residualCargoMeasurement->0.6 :hull->:double)) -->

                <Atom>
                        <op>
                                <Rel>Ship</Rel>
                        </op>
                        <slot>
                                <Ind type="dm:ship">ship5</Ind>
                                <Plex>
                                        <slot>
                                                <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                <Ind type="xsd:dateTime">2020-01-
01</Ind>
                                        </slot>
                                        <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">90</Data>
                                        </slot>
                                        <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                        <Ind
type="dm:shipHold">h5</Ind>
                                                        <slot>
                                                                <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                <Ind
type="xsd:double">0.6</Ind>
                                                        </slot>
                                                        <slot>
                                                                <Ind
type="xsd:string">hull</Ind>
                                                                <Ind
type="xsd:string">double</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
                        </slot>
                </Atom>

                <!-- % Ship 6 - No, size large yet hold single-hulled
:ship6#:Ship(:registryExpirationDate->phys:date(2020
                        1 1) :totalLength->90 :hold-
>:h6#:ShipHold(:residualCargoMeasurement->0.4
                        :hull->:single)) -->

                <Atom>
                        <op>
                                <Rel>Ship</Rel>
                        </op>
                        <slot>
                                <Ind type="dm:ship">ship6</Ind>
                                <Plex>
```

```
                                        <slot>
                                                <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                <Ind type="xsd:dateTime">2020-01-
01</Ind>
                                        </slot>
                                        <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">90</Data>
                                        </slot>
                                        <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                        <Ind
type="dm:shipHold">h6</Ind>
                                                        <slot>
                                                                <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                <Ind
type="xsd:double">0.4</Ind>
                                                        </slot>
                                                        <slot>
                                                                <Ind
type="xsd:string">hull</Ind>
                                                                <Ind
type="xsd:string">single</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
                        </slot>
                </Atom>

                <!-- % Ship 7 - Yes, hold clean and double-hulled
:ship7#:Ship(:registryExpirationDate->phys:date(2020
                        1 1) :totalLength->90 :hold-
>:h7#:ShipHold(:residualCargoMeasurement->0.4
                        :hull->:double) -->

                <Atom>
                        <op>
                                <Rel>Ship</Rel>
                        </op>
                        <slot>
                                <Ind type="dm:ship">ship7</Ind>
                                <Plex>
                                        <slot>
                                                <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                <Ind type="xsd:dateTime">2020-01-
01</Ind>
                                        </slot>
                                        <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">90</Data>
                                        </slot>
                                        <slot>
```

```
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                       <Ind
type="dm:shipHold">h7</Ind>
                                                       <slot>
                                                              <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                              <Ind
type="xsd:double">0.4</Ind>
                                                       </slot>
                                                       <slot>
                                                              <Ind
type="xsd:string">hull</Ind>
                                                              <Ind
type="xsd:string">double</Ind>
                                                       </slot>
                                                </Plex>
                                         </slot>
                                  </Plex>
                           </slot>
                    </Atom>

                    <!-- % Facts with multiple reasons for No or Yes % Three
reasons for No
                           % Ship 8 - No, registry expired, hold not clean, and
size large yet hold
                           single-hulled :ship8#:Ship(:registryExpirationDate-
>phys:date(2017 1 1) :totalLength->90
                           :hold->:h8#:ShipHold(:residualCargoMeasurement->0.9
:hull->:single)) -->

                    <Atom>
                           <op>
                                  <Rel>Ship</Rel>
                           </op>
                           <slot>
                                  <Ind type="dm:ship">ship8</Ind>
                                  <Plex>
                                         <slot>
                                                <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                <Ind type="xsd:dateTime">2020-01-
01</Ind>
                                         </slot>
                                         <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">90</Data>
                                         </slot>
                                         <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                       <Ind
type="dm:shipHold">h8</Ind>
                                                       <slot>
                                                              <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                              <Ind
type="xsd:double">0.9</Ind>
```

```
                                                        </slot>
                                                        <slot>
                                                                <Ind
type="xsd:string">hull</Ind>
                                                                <Ind
type="xsd:string">single</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
                        </slot>
                </Atom>

                <!-- % Two reasons for No % Ship 9 - No, hold not clean and
size large
                        yet hold single-hulled
:ship9#:Ship(:registryExpirationDate->phys:date(2018
                        1 1) :totalLength->90 :hold-
>:h9#:ShipHold(:residualCargoMeasurement->0.9
                        :hull->:single)) -->

                <Atom>
                        <op>
                                <Rel>Ship</Rel>
                        </op>
                        <slot>
                                <Ind type="dm:ship">ship9</Ind>
                                <Plex>
                                        <slot>
                                                <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                <Ind type="xsd:dateTime">2020-01-
01</Ind>
                                        </slot>
                                        <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">90</Data>
                                        </slot>
                                        <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                        <Ind
type="dm:shipHold">h9</Ind>
                                                        <slot>
                                                                <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                <Ind
type="xsd:double">0.9</Ind>
                                                        </slot>
                                                        <slot>
                                                                <Ind
type="xsd:string">hull</Ind>
                                                                <Ind
type="xsd:string">single</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
```

```
                        </slot>
                    </Atom>

                    <!-- % Ship 10 - No, registry expired and size large yet hold
single-hulled
                        :ship10#:Ship(:registryExpirationDate->phys:date(2017 1
1) :totalLength->90
                        :hold->:h10#:ShipHold(:residualCargoMeasurement->0.2
:hull->:single)) -->

                    <Atom>
                        <op>
                            <Rel>Ship</Rel>
                        </op>
                        <slot>
                            <Ind type="dm:ship">ship10</Ind>
                            <Plex>
                                <slot>
                                    <Ind
type="xsd:string">registryExpirationDate</Ind>
                                    <Ind type="xsd:dateTime">2017-01-
01</Ind>
                                </slot>
                                <slot>
                                    <Ind
type="xsd:string">totalLength</Ind>
                                    <Data xsi:type="xs:int">90</Data>
                                </slot>
                                <slot>
                                    <Ind type="xsd:string">hold</Ind>
                                    <Plex>
                                        <Ind
type="dm:shipHold">h10</Ind>
                                        <slot>
                                            <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                            <Ind
type="xsd:double">0.2</Ind>
                                        </slot>
                                        <slot>
                                            <Ind
type="xsd:string">hull</Ind>
                                            <Ind
type="xsd:string">single</Ind>
                                        </slot>
                                    </Plex>
                                </slot>
                            </Plex>
                        </slot>
                    </Atom>

                    <!-- % Ship 11 - No, registry expired and hold not clean
:ship11#:Ship(:registryExpirationDate->phys:date(2017
                        1 1) :totalLength->90 :hold-
>:h11#:ShipHold(:residualCargoMeasurement->0.9
                        :hull->:double)) -->

                    <Atom>
                        <op>
```

```
                                        <Rel>Ship</Rel>
                                </op>
                                <slot>
                                        <Ind type="dm:ship">ship11</Ind>
                                        <Plex>
                                                <slot>
                                                        <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                        <Ind type="xsd:dateTime">2017-01-
01</Ind>
                                                </slot>
                                                <slot>
                                                        <Ind
type="xsd:string">totalLength</Ind>
                                                        <Data xsi:type="xs:int">90</Data>
                                                </slot>
                                                <slot>
                                                        <Ind type="xsd:string">hold</Ind>
                                                        <Plex>
                                                                <Ind
type="dm:shipHold">h11</Ind>
                                                                <slot>
                                                                        <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                        <Ind
type="xsd:double">0.9</Ind>
                                                                </slot>
                                                                <slot>
                                                                        <Ind
type="xsd:string">hull</Ind>
                                                                        <Ind
type="xsd:string">double</Ind>
                                                                </slot>
                                                        </Plex>
                                                </slot>
                                        </Plex>
                                </slot>
                        </Atom>

                        <!-- % Two reasons for Yes % Ship 12 - Yes, size small
nevertheless hold
                                double-hulled :ship12#:Ship(:registryExpirationDate-
>phys:date(2020 1 1)
                                :totalLength->60 :hold-
>:h12#:ShipHold(:residualCargoMeasurement->0.1 :hull->:double)) -->

                        <Atom>
                                <op>
                                        <Rel>Ship</Rel>
                                </op>
                                <slot>
                                        <Ind type="dm:ship">ship12</Ind>
                                        <Plex>
                                                <slot>
                                                        <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                        <Ind type="xsd:dateTime">2020-01-
01</Ind>
                                                </slot>
```

```
                                        <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">60</Data>
                                        </slot>
                                        <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                        <Ind
type="dm:shipHold">h12</Ind>
                                                        <slot>
                                                                <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                <Ind
type="xsd:double">0.1</Ind>
                                                        </slot>
                                                        <slot>
                                                                <Ind
type="xsd:string">hull</Ind>
                                                                <Ind
type="xsd:string">double</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
                        </slot>
                </Atom>

                <!-- % Facts probing special cases % Ship 13 - No, large ship
must have
                        some (a double) hull
:ship13#:Ship(:registryExpirationDate->phys:date(2020
                        1 1) :totalLength->120 :hold-
>:h13#:ShipHold(:residualCargoMeasurement->0.2)) -->

                <Atom>
                        <op>
                                <Rel>Ship</Rel>
                        </op>
                        <slot>
                                <Ind type="dm:ship">ship13</Ind>
                                <Plex>
                                        <slot>
                                                <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                <Ind type="xsd:dateTime">2020-01-
01</Ind>
                                        </slot>
                                        <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">120</Data>
                                        </slot>
                                        <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                        <Ind
type="dm:shipHold">h13</Ind>
                                                        <slot>
```

```
                                                                <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                <Ind
type="xsd:double">0.2</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
                        </slot>
                </Atom>

                <!-- % Ship 14 - Yes, date, length, and measurement are at the
threshold
                        :ship14#:Ship(:registryExpirationDate->phys:date(2017 5
7) :totalLength->80
                        :hold->:h14#:ShipHold(:residualCargoMeasurement->0.5
:hull->:double)) -->

                <Atom>
                        <op>
                                <Rel>Ship</Rel>
                        </op>
                        <slot>
                                <Ind type="dm:ship">ship14</Ind>
                                <Plex>
                                        <slot>
                                                <Ind
type="xsd:string">registryExpirationDate</Ind>
                                                <Ind type="xsd:dateTime">2018-01-
21</Ind>
                                        </slot>
                                        <slot>
                                                <Ind
type="xsd:string">totalLength</Ind>
                                                <Data xsi:type="xs:int">80</Data>
                                        </slot>
                                        <slot>
                                                <Ind type="xsd:string">hold</Ind>
                                                <Plex>
                                                        <Ind
type="dm:shipHold">h14</Ind>
                                                        <slot>
                                                                <Ind
type="xsd:string">residualCargoMeasurement</Ind>
                                                                <Ind
type="xsd:double">0.5</Ind>
                                                        </slot>
                                                        <slot>
                                                                <Ind
type="xsd:string">hull</Ind>
                                                                <Ind
type="xsd:string">double</Ind>
                                                        </slot>
                                                </Plex>
                                        </slot>
                                </Plex>
                        </slot>
                </Atom>
```

```
            </Rulebase>
        </Assert>
</RuleML>
```

After formalization of Port Clearance rules and facts in RuleML they are translated to RIF XML using the RuleML to RIF component of the framework. The translation result is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="http://www.w3.org/2007/rif#"
        xmlns:ruleml="http://ruleml.org/spec"
        xmlns:xs="http://www.w3.org/2001/XMLSchema#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <payload>
                <Group>
                        <sentence>
                                <Group>
                                        <Implies>
                                                <if>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">CompliesInspectionRequirementsUnloaded
                                                                        </Const>
                                                                </op>
                                                                <args ordered="yes">
                                                                        <Var>s</Var>
                                                                </args>
                                                        </Atom>
                                                </if>
                                                <then>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">MayEnterDutchPortUnloaded</Const>
                                                                </op>
                                                                <args ordered="yes">
                                                                        <Var>s</Var>
                                                                </args>
                                                        </Atom>
                                                </then>
                                        </Implies>
                                        <Implies>
                                                <if>
                                                        <And>
                                                                <Atom>
                                                                        <op>
                                                                                <Const
type="http://www.w3.org/2007/rif#iri">HasValidCertificate</Const>
                                                                        </op>
                                                                        <args
ordered="yes">
```

```
                                                                <Var>s</Var>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">MeetsSafetyRequirementsUnloaded</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
                                                        </args>
                                                </Atom>
                                        </And>
                                </if>
                                <then>
                                        <Atom>
                                                <op>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">CompliesInspectionRequirementsUnloaded
                                                        </Const>
                                                </op>
                                                <args ordered="yes">
                                                        <Var>s</Var>
                                                </args>
                                        </Atom>
                                </then>
                        </Implies>
                        <Implies>
                                <if>
                                        <And>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
                                                                <Var>e</Var>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">currentDate</Const>
                                                        </op>
                                                        <args
ordered="yes">

                <Var>cd</Var>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">greaterThan</Const>
                                                        </op>
```

```
                                                        <args
ordered="yes">
                                                                <Var>e</Var>

        <Var>cd</Var>
                                                        </args>
                                                </Atom>
                                        </And>
                                </if>
                                <then>
                                        <Atom>
                                                <op>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">HasValidCertificate</Const>
                                                </op>
                                                <args ordered="yes">
                                                        <Var>s</Var>
                                                </args>
                                        </Atom>
                                </then>
                        </Implies>
                        <Implies>
                                <if>
                                        <And>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">size</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">small</Const>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
                                                                <Var>h</Var>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">status</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>h</Var>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">clean</Const>
                                                        </args>
                                                </Atom>
```

```
                                                </And>
                                        </if>
                                        <then>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">MeetsSafetyRequirementsUnloaded</Const>
                                                        </op>
                                                        <args ordered="yes">
                                                                <Var>s</Var>
                                                        </args>
                                                </Atom>
                                        </then>
                                </Implies>
                                <Implies>
                                        <if>
                                                <And>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">size</Const>
                                                                </op>
                                                                <args
ordered="yes">
                                                                        <Var>s</Var>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">large</Const>
                                                                </args>
                                                        </Atom>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                                </op>
                                                                <args
ordered="yes">
                                                                        <Var>s</Var>
                                                                        <Var>h</Var>
                                                                </args>
                                                        </Atom>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">status</Const>
                                                                </op>
                                                                <args
ordered="yes">
                                                                        <Var>h</Var>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">clean</Const>
                                                                </args>
                                                        </Atom>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                </op>
                                                                <args
ordered="yes">
```

```
                                                                <Var>h</Var>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                        </args>
                                                </Atom>
                                        </And>
                                </if>
                                <then>
                                        <Atom>
                                                <op>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">MeetsSafetyRequirementsUnloaded</Const>
                                                </op>
                                                <args ordered="yes">
                                                        <Var>s</Var>
                                                </args>
                                        </Atom>
                                </then>
                        </Implies>
                        <Implies>
                                <if>
                                        <And>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>s</Var>
                                                                <Var>l</Var>
                                                        </args>
                                                </Atom>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">lessThan</Const>
                                                        </op>
                                                        <args
ordered="yes">
                                                                <Var>l</Var>

                <Var>80</Var>
                                                        </args>
                                                </Atom>
                                        </And>
                                </if>
                                <then>
                                        <Atom>
                                                <op>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">size</Const>
                                                </op>
                                                <args ordered="yes">
                                                        <Var>s</Var>
                                                        <Const

type="http://www.w3.org/2007/rif#iri">small</Const>
                                                </args>
                                        </Atom>
```

```
                                        </then>
                                </Implies>
                                <Implies>
                                        <if>
                                                <And>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                                </op>
                                                                <args
ordered="yes">
                                                                        <Var>s</Var>
                                                                        <Var>l</Var>
                                                                </args>
                                                        </Atom>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">greaterThanOrEqual</Const>
                                                                </op>
                                                                <args
ordered="yes">
                                                                        <Var>l</Var>

        <Var>80</Var>
                                                                </args>
                                                        </Atom>
                                                </And>
                                        </if>
                                        <then>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">size</Const>
                                                        </op>
                                                        <args ordered="yes">
                                                                <Var>s</Var>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">large</Const>
                                                        </args>
                                                </Atom>
                                        </then>
                                </Implies>
                                <Implies>
                                        <if>
                                                <And>
                                                        <Atom>
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                </op>
                                                                <args
ordered="yes">
                                                                        <Var>h</Var>
                                                                        <Var>c</Var>
                                                                </args>
                                                        </Atom>
                                                        <Atom>
```

```
                                                                <op>
                                                                        <Const
type="http://www.w3.org/2007/rif#iri">lessThanOrEqual</Const>
                                                                </op>
                                                                <args
ordered="yes">
                                                                        <Var>c</Var>

        <Var>0.5</Var>
                                                                </args>
                                                        </Atom>
                                                </And>
                                        </if>
                                        <then>
                                                <Atom>
                                                        <op>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">status</Const>
                                                        </op>
                                                        <args ordered="yes">
                                                                <Var>h</Var>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">clean</Const>
                                                        </args>
                                                </Atom>
                                        </then>
                                </Implies>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">currentDate</Const>
                                        </op>
                                        <args ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">2018-01-21</Const>
                                        </args>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship1</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2017-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">20</Const>
                                                </slot>
                                                <slot ordered="yes">
```

```
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h1</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.2</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship2</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">20</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h2</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.2</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
```

```
                                    <Atom>
                                        <op>
                                            <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                            <Const
type="http://www.w3.org/2007/rif#iri">ship3</Const>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">70</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">h3</Const>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">0.6</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                </slot>
                                            </slot>
                                        </slot>
                                    </Atom>
                                    <Atom>
                                        <op>
                                            <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                            <Const
type="http://www.w3.org/2007/rif#iri">ship4</Const>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">70</Const>
```

```
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h4</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.4</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship5</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h5</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.6</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                        </slot>
                                                </slot>
```

```
                                        </slot>
                                    </Atom>
                                    <Atom>
                                        <op>
                                            <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                            <Const
type="http://www.w3.org/2007/rif#iri">ship6</Const>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">h6</Const>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">0.4</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                    <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                    <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                </slot>
                                            </slot>
                                        </slot>
                                    </Atom>
                                    <Atom>
                                        <op>
                                            <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                            <Const
type="http://www.w3.org/2007/rif#iri">ship7</Const>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                            </slot>
                                            <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
```

```
                                                        <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h7</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.4</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship8</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h8</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.9</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
```

```
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship9</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h9</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.9</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship10</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2017-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
```

```
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h10</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.2</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">single</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship11</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2017-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">90</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h11</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.9</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
```

```
                                                                <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship12</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">60</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h12</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.1</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship13</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2020-01-01</Const>
```

```
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">120</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h13</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.2</Const>
                                                        </slot>
                                                </slot>
                                        </slot>
                                </Atom>
                                <Atom>
                                        <op>
                                                <Const
type="http://www.w3.org/2007/rif#iri">Ship</Const>
                                        </op>
                                        <slot ordered="yes">
                                                <Const
type="http://www.w3.org/2007/rif#iri">ship14</Const>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">registryExpirationDate</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">2018-01-21</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">totalLength</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">80</Const>
                                                </slot>
                                                <slot ordered="yes">
                                                        <Const
type="http://www.w3.org/2007/rif#iri">hold</Const>
                                                        <Const
type="http://www.w3.org/2007/rif#iri">h14</Const>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">residualCargoMeasurement</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">0.5</Const>
                                                        </slot>
                                                        <slot ordered="yes">
                                                                <Const
type="http://www.w3.org/2007/rif#iri">hull</Const>
                                                                <Const
type="http://www.w3.org/2007/rif#iri">double</Const>
                                                        </slot>
                                                </slot>
```

```
                                        </slot>
                                    </Atom>
                                </Group>
                            </sentence>
                        </Group>
                </payload>
</Document>
```

Appendix F

UML diagrams of Notation3 and POSL translators

Figure F.1 shows the UML diagram of Notation3 (N3) parser and translator.

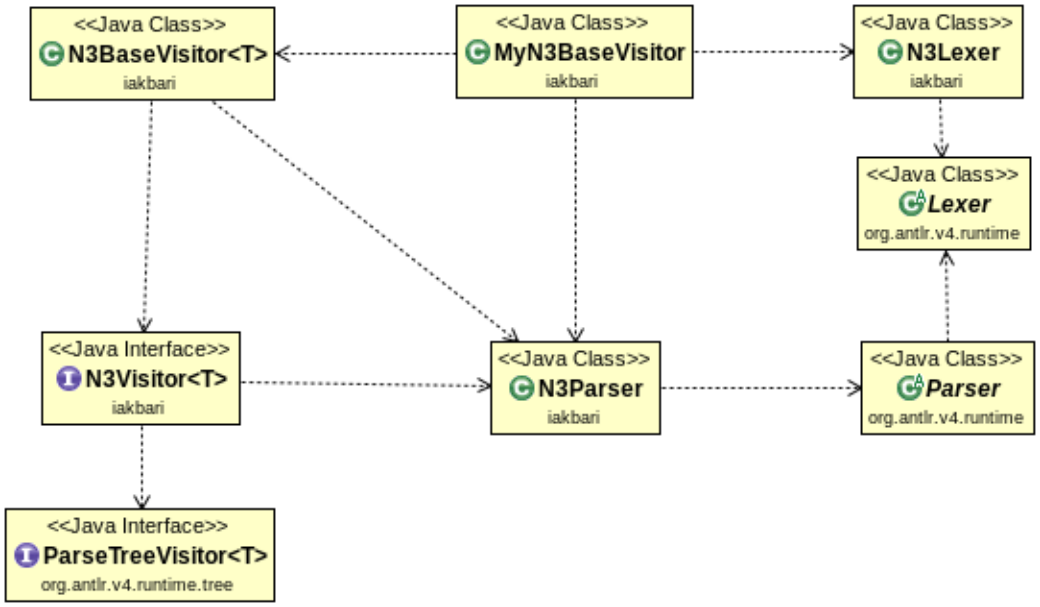

Figure F.1. Notation3 translator UML diagram

'MyN3BaseVisitor' class (in Figure F.1) translates N3 rules to RIFBLD. Figure F.2 shows the UML diagram of POSL parser and translator.

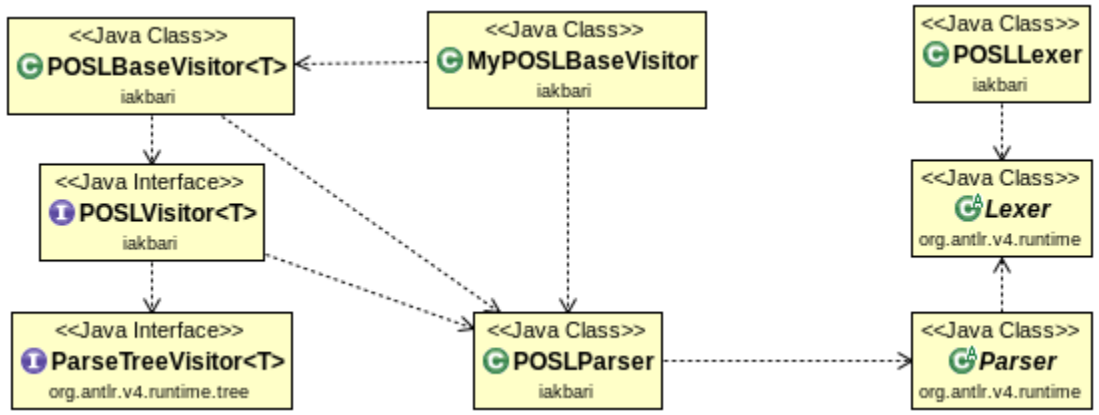

Figure F.2. POSL translator UML diagram

‘MyPOSLBaseVisitor’ translates POSL rules to RIFBLD rules. To see details of N3 and POSL implementations of parsers and translators to RIF-BLD refer to the Javadoc documentation in the framework.

Appendix G

Java Web Start

The implemented framework is developed in Java and is available as a GitHub repository at [59]. It can be downloaded as a Java Web Start (JWS) file. JWS files have a '.jnlp' file extension. A JWS file is an XML formatted file and can be edited with any text editor. After downloading the framework from [59] and to run it for the first time, you will receive a Java security message as shown in Figure G.1.

Figure G.1. Java security message for running Java Web Start Applications

This appendix shows how to cope with this issue. To unblock the JWS application, the location URI as shown in the security message in figure G.1 must be added to Java exception site list. Figure G.2 show how to add the location URI which is "http://www2.unb.ca" to the Java exception list.

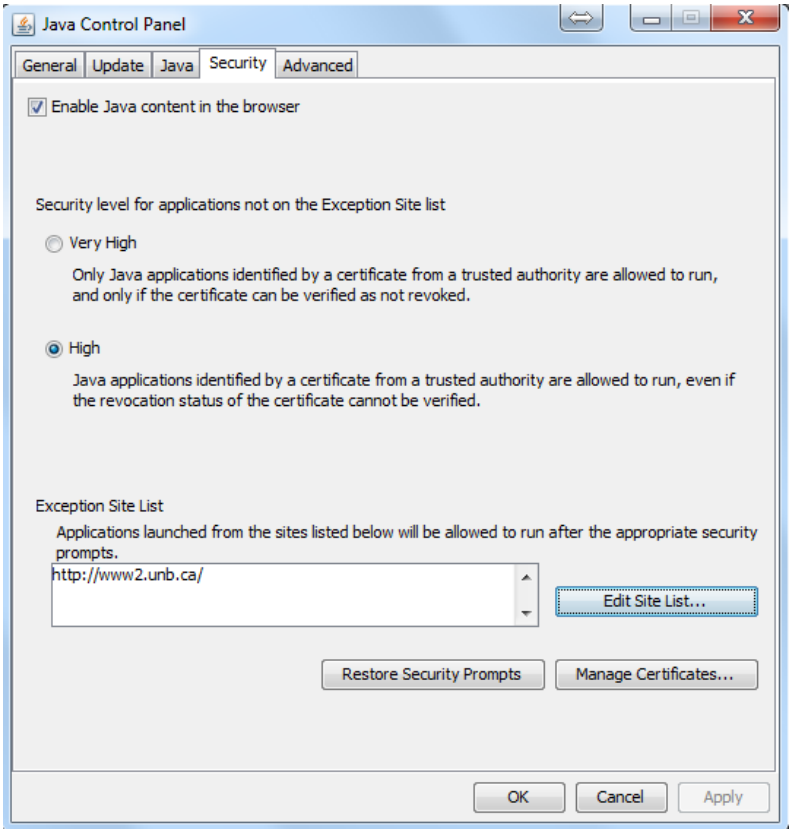

Figure G.2. Add JWS URI to Java exception site list

After adding the JWS URI to Java exception site list you can run the JWS application file. This time you will receive the following Java security warning (Figure G.3). By accepting this warning message, you can run the framework.

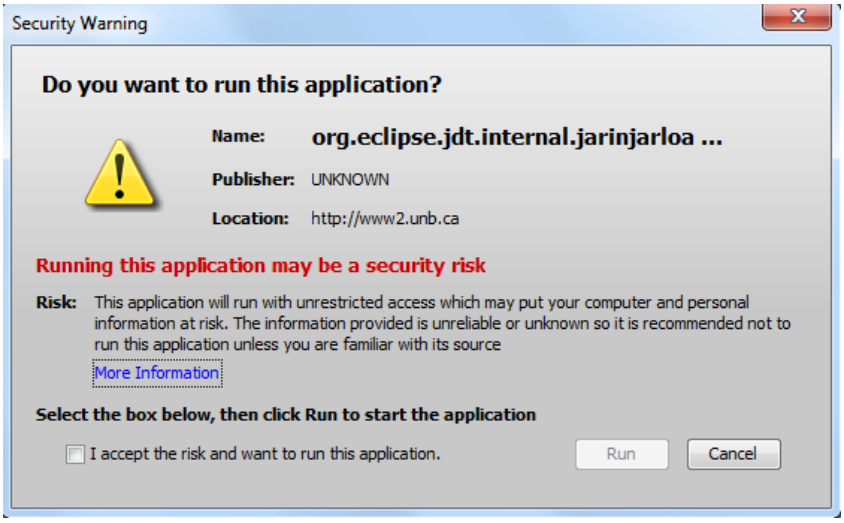

Figure G.3. Java Web Start security message

The warning in Figure G.3 is a common warning while trying to run Java Web Start applications from a personal computer.

Curriculum Vitae

Candidate's full name: Ismail Akbari

Universities attended (with dates and degrees obtained):

University of New Brunswick, Fredericton, NB, Canada.

Ph.D. Candidate, Faculty of Computer Science, started September 2012.

Master of Information Technology Engineering, Iran University of Science and Technology, Iran, 2009.

Bachelor of Computer Engineering, Teacher Training University, Iran, 2006.

Publications:

Biletskiy, Yevgen, J. Anthony Brown, Girish R. Ranganathan, Ebrahim Bagheri, and Ismail Akbari. "Building a business domain meta-ontology for information pre-processing." *Information Processing Letters* (2018).

Ranganathan, Girish R., Yevgen Biletskiy, and Ismail Akbari. "Semi-Automatic Rule Learning Method Enabling Information Extraction for Ontology Population." *Iranian Journal of Science and Technology, Transactions of Electrical Engineering* 40.2-4 (2016): 103-115.

Akbari, Ismail, Yevgen Biletskiy, and Weichang Du. "Practical Measurements for Quality of Ontology Matching Applying to the OAEI Dataset." *Mexican International Conference on Artificial Intelligence*. Springer International Publishing, 2015.

Akbari, Ismail, et al. "Visualizing SWRL Rules: From Unary/Binary Datalog and PSOA RuleML to Graphviz and Grailog." *4th Canadian Semantic Web Symposium (CSWS 2013)*. 2013.

Zeighami, V., R. Akbari, and K. Ziarati. "Development of a method based on particle swarm optimization to solve resource constrained project scheduling problem." *Scientia Iranica. Transaction E, Industrial Engineering* 20, no. 6 (2013): 2123.

Akbari, Reza, Vahid Zeighami, Koorush Ziarati, and Ismail Akbari. "Development of an Efficient Hybrid Method for Motif Discovery in DNA Sequences." *AUT Journal of Electrical Engineering* 44, no. 1 (2012): 63-75.

Akbari, Ismail, and Mohammad Fathian. "A novel algorithm for ontology matching." *Journal of Information Science* 36, no. 3 (2010): 324-334.

Akbari, Ismail, Mohammad Fathian, and Kambiz Badie. "An improved MLMA+ and its application in ontology matching." In *Innovative technologies in intelligent systems and industrial applications, 2009. CITISIA 2009*, pp. 56-60. IEEE, 2009.