RIF-BLD Reasoning with IRIS

Daniel Winkler, Reto Krummenacher, and Adrian Marte

Semantic Technology Institute (STI) Innsbruck,
University of Innsbruck, Austria
firstname.lastname@sti2.at

Abstract. IRIS is an open-source, Java-based Datalog reasoner that supports various evaluation strategies and extensions such as function symbols and equality in the rule conclusion. The main objective of IRIS is to serve as general purpose reasoner for the Semantic Web, which is realized by an implementation that focuses on compliance with existing W3C standards like the Rule Interchange Format (RIF), thereby supporting the XML Schema Definition datatypes. Furthermore, IRIS acts as base for extensions that allow for reasoning with the OWL 2 profiles EL and RL and the Web Service Modeling Language (WSML). In this demo, we present the IRIS reasoner and its applications, as well as the larger WSML2Reasoner framework with IRIS at its core. The basic IRIS implementation for RIF and OWL 2 reasoning is stable, however, further performance improvements and the implementation of effective parallelized rule-based reasoning strategies that are ongoing work will also be presented.

1 Introduction

The amount of data published on the Web is growing rapidly and with it the structured data related to it. Moreover, much original data is published in a structured form and exposed as so-called Linked Data.[1] Formal languages are used to describe and annotate such data, in order to allow machines to interpret it and reason about it; i.e., to check consistency, to answer queries, or to infer new otherwise uncovered knowledge. In response to the immense amount of data, formalisms are sought that allow for tractable and efficient reasoning algorithms. The difficulty with such languages is the trade-off between the requirements for expressivity and the usefulness for enabling tractable reasoning characteristics.

One such effort is the Rule Interchange Format (RIF), which was recently standardized as W3C recommendation. RIF aims at specifying a format that can be used for knowledge exchange between various rule systems based on a common syntax. More important with respect to rule-based reasoning are the semantic profiles that RIF defines. Profiles reflect particular use case requirements and yield purposeful balances between expressivity and computational complexity. The support for different profiles is not only reflected in RIF, but also in another recent standard by the W3C. The updated Web Ontology Language standard

[1] http://linkeddata.org/

OWL 2 provides dialects that are restricted in their semantic expressivity for the sake of better reasoning behavior; e.g., OWL EL or OWL RL profiles. The very same idea is at the basis of the WSML language family that we will encounter throughout this demo description, in particular the variants WSML-Rule and WSML-Flight. Defining such language variants helps in establishing formalisms that are expressive enough to be useful, while exhibiting reasoning characteristics that can also scale to the size of the Web, which is exactly the application area we address with our rule-based reasoner IRIS.

The Integrated Rule Inference System IRIS is realized with generality, flexibility and extensibility in mind.[2] Not at last for this reason, IRIS can be used as core engine for different reasoners that tackle diverse formalism ranging across various RIF, OWL 2 and WSML dialects; cf. subsequent sections. Effectively, IRIS is applied in different research projects for diverse tasks; e.g., semantic discovery, ranking and design-time service composition in the integrated project SOA4All and as general purpose reasoning plug-in in LarKC.[3] Being open-source and Java-based further increases the impact, and IRIS is leveraged in many other research projects and applications. So far, IRIS was downloaded more than 1500 times from sourceforge.net, not taking into account the accesses via the project's Maven repository.

The continuation of this paper is structured as follows. In Section 2 we present our rule-based reasoning engine IRIS, and show how it is used as reasoner for different profiles, mainly RIF-BLD. In the same section we present RIF4J, a Java object model for RIF rule bases. Section 3 provides a comprehensive presentation of a RIF-BLD demo application scenario, and yields some pointers towards the use of IRIS for reasoning about time and intervals via corresponding RIF built-ins. In Section 4, we discuss different extensions to IRIS that embed the reasoner in a larger framework in order to support different OWL 2 and WSML dialects. This allows for even more diverse support for reasoning on the Web. We conclude with Section 5, and present a brief discussion of future extensions and applications of IRIS.

2 IRIS as RIF Reasoner

The Rule Interchange Format (RIF) by the W3C RIF Working Group enables the semantic and syntactic description of rule systems, which can be further used to exchange axiomatic knowledge between systems.[4] It includes a framework for defining logic dialects, several concrete dialects, datatype definitions and built-in predicates and functions. In a first subsection (Section 2.1) we will shortly present the most relevant RIF dialects, and how our Datalog reasoning engine IRIS responds to them. [5]

[2] `http://www.iris-reasoner.org/`

[3] SOA4All: `http://www.soa4all.eu`, LarKC: `http://www.larkc.eu/`

[4] `http://www.w3.org/2005/rules/wiki/RIF_Working_Group`

[5] An online demonstrator of the pure Datalog implementation, not subject to this demonstration, is published at `http://www.iris-reasoner.org/demo`.

2.1 IRIS and RIF Dialects

The RIF Datatypes and Built-Ins (RIF-DTB, [12]) document specifies a list of datatypes, built-in functions and built-in predicates — mostly adapted from XPath functions [8] and XML Schema Datatypes (XSD, [11]) — that are expected to be supported by all RIF dialects. In this sense, RIF DTB defines the functional baseline for any RIF reasoner; unless explicitly defined differently by a given profile. IRIS has recently been updated to support the full range of built-in RIF functions and predicates, and all datatypes defined by XSD — going beyond the RIF standard — with the exception of the list-related ones.[6]

The RIF Core dialect is contained in the intersection of the production rules (RIF-PRD) and basic logic (RIF-BLD, [2]) profiles, thereby building a base for both of them. From a theoretical point of view, the expressivity of RIF Core corresponds to Datalog. Due to the very nature of IRIS, Datalog and thus the expressivity of RIF Core are captured by the implementation, furthermore various evaluation strategies and optimizations are implemented.

The RIF-BLD dialect corresponds, in terms of expressivity, to the language of definite Horn rules with equality and a standard first-order semantics. IRIS was extended to support the full range of language constructs defined in RIF-BLD, thereby investigating the impacts of equality in rule conclusions in detail. In specific, IRIS uses semi-naive bottom-up evaluation as default evaluation strategy for query answering [13]. In case of equality assertions, however, the strategy has to fall back to naive evaluation as pre-computed facts need to be re-evaluated in terms of equality. Improvements are subject to further investigations and none are yet reflected in the implementation.

RIF Core and RIF-BLD are also at the basis of the W3C recommendation on how to interpret combinations of RIF documents and RDF data, as well as RDFS and OWL ontologies [4]. Supporting RIF-BLD can thus also be seen as a prerequisite for the implementation of a fully-fledged rule-based reasoner for the (Semantic) Web.

2.2 RIF4J — An Object Model for RIF Rule Bases

While IRIS supports all the necessary logical constructs to offer a RIF-BLD reasoner, it still needed to parse and understand the RIF syntax. The RIF4J Java library provides the necessary extensions and offers a parser and object model for RIF rule bases.[7] Although RIF4J was conceptualized and developed as interface for RIF rule bases on top of IRIS, the object model is fully independent of any IRIS libraries. Consequently, RIF4J not only provides the required functionality to make IRIS a RIF-minded reasoning engine, but rather provides a general purpose set of technicalities for working with RIF construct.

Besides support for parsing RIF rule bases and for processing and modeling RIF rules in a Java object model, RIF4J offers an extension to serialize rule bases

[6] RIF Lists, `xsd:ENTITIES`, `xsd:IDREFS`, `xsd:NMTOKENS`

[7] `http://sourceforge.net/projects/rif4j/`

Table 1. Demo Example: Temporal Delivery Conditions

```
Document(
 Prefix(cpt  <http://example.com/concepts#>)
 Prefix(func <http://www.w3.org/2007/rif-builtin-function#>)
 ...
Group (
 Forall ?item ?deliverydate ?scheduledate ?diffduration ?diffdays (
  cpt:reject(ex:John ?item) :-
   And(cpt:perishable(?item)
       cpt:delivered(?item ?deliverydate ex:John)
       cpt:scheduled(?item ?scheduledate)
       ?diffduration = External(func:subtract-dateTimes(
                                ?deliverydate ?scheduledate))
       ?diffdays = External(func:days-from-duration(?diffduration))
       External(pred:numeric-greater-than(?diffdays 10))))
 Forall ?item ( cpt:perishable(?item) :- cpt:grocery(?item))
 Forall ?item (
  cpt:scheduled(?item ?scheduledate) :-
   And(cpt:scheduled(?collection ?scheduledate)
       cpt:contains(?collection ?item)))
 cpt:orders(ex:John ex:Order)
 cpt:scheduled(ex:Order "2010-08-20T16:00:00"^^xs:dateTime)
 cpt:grocery(ex:Milk).
 cpt:software(ex:IRIS).
 ...
 cpt:contains(ex:Order ex:Milk)
 cpt:contains(ex:Order ex:IRIS)
 ...
 cpt:delivered(ex:IRIS "2010-08-20T12:32:52"^^xs:dateTime ex:John)
 cpt:delivered(ex:Milk "2010-10-21T08:00:01"^^xs:dateTime ex:John)
 ...))
```

to WSML logical expressions. In fact, WSML was the initial language stack which IRIS was built for [1]. The WSML language variants partly form supersets and/or subset of RIF-BLD, which allows for a simple syntactical mapping from RIF to WSML (cf. Section 4.2). To this end, by translating RIF rule bases to WSML logical expression, RIF4J realizes the RIF reasoner functionalities of IRIS on top of and by means of the well-tested and stable WSML2Reasoner framework, which is shortly presented in Section 4.

3 RIF-BLD Application Scenario

To illustrate how IRIS can be leveraged to reason with RIF-BLD rules, we present a hypothetical scenario based on the use case "Negotiating eBusiness Contracts Across Rule Platforms" in [10]. The core rule of the scenario is provided as "If

an item is perishable and it is delivered to John more than 10 days after the scheduled delivery date then the item will be rejected by him". This rule is formally given in Table 1 on lines 6–14, using RIF presentation syntax (cf. [2]).

According to the core rule of the example, John is negotiating a business contract for the shipment of different goods. John is very critical about the late delivery, and indicates that products, which arrive more than ten days after the scheduled date, will be rejected, if they are perishable. The rule base for our example moreover contains the claim that groceries are perishable (line 15), and that the scheduled delivery date of an item is inherited from the delivery date of the order (lines 16–19). In this weeks order (line 20), John requests, among many other things, that the IRIS software package and some milk shall be delivered to him (lines 22–27). According to the agree delivery schedule, the products should reach John by August 20, 2010 (line 21). The fact base states as well, on lines 28–30, when the different good were delivered to John.

The RIF-BLD reasoner can then be used for entailment checking against or querying over the specified rule base. For the purpose of this demo, there is a user interface publicly available at `http://iris.sti2.at/reasoners/rif-reasoner/`. By means of the following query: `cpt:reject(ex:John ?x)`, it is possible, for example, to check if John does reject any of the delivered items. In the given case of Table 1, John indeed does not accept the milk item, as it is eventually delivered significantly later than the agreed August 20, 2010.

4 Reasoner Extensions

This section brings IRIS into a larger context, and discusses the use of the Datalog reasoning engine as fundamental building block for the WSML2Reasoner framework, including the ELP reasoner Elly. Figure 1 reconciles the relevant software components. The two components on the left, RIF4J and WSMO4J,[8] represent the object models that are used to maintain the static data of the knowledge bases that can be reasoned over. Additionally, the OWL API, a development mostly driven by the University of Manchester,[9] provides a third data model component for the reasoning and manipulation of OWL ontologies. As such, the OWL API provides the reasoner interface that is implemented by Elly for supporting OWL 2 EL and RL profiles, as shortly discussed in Section 4.1. Subsequently, in Section 4.2, we present how the WSML2Reasoner framework serves as entry point for RIF and WSML reasoning by means of RIF4J/WSMO4J as object models, and IRIS and Elly as default reasoning engines.

4.1 Web Ontology Language (OWL) 2 EL and RL Reasoning

ELP has been presented as a hybrid between Logic Programming (LP) and Description Logics (DL) that combines the tractable DLs $\mathcal{EL}^{++}$ and DLP in a novel

[8] WSMO4J is an API and a reference implementation for WSML constructs.
[9] `http://owlapi.sourceforge.net`

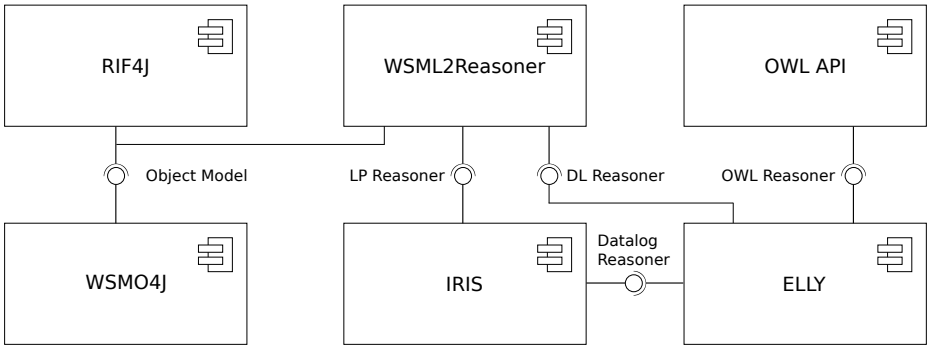

Fig. 1. Reasoner architecture and data model components

way [7]. These two formalisms serve as logical foundation for the OWL 2 profiles EL and RL, which are thus fully captured by the semantics of ELP [9]. Since ELP does not only define the semantics of the language, but also provides a tractable reasoning algorithm that translates ELP rule bases into Datalog rule bases, a corresponding extension to IRIS could be implemented.

Elly is a reasoner for entailment and satisfiability checking of ELP rule bases.[10] Its core package implements an object model for ELP rule bases and a reasoner based on the translation algorithm to Datalog [14]. Elly is implemented on top of IRIS and reuses, but abstracts, all of the built-in datatypes and predicates of IRIS; as such that Elly can be considered an extension to IRIS. Elly, like all other software presented in this paper, is released under the LGPL 2.1 license.[11].

As mentioned above, ELP subsumes the semantics of the OWL 2 profiles EL and RL. To this end, the ELP reasoner Elly, in integration with the parsers, object models and reasoning interfaces of the OWL API, becomes a fully-fledged OWL 2 EL and RL reasoner.[12]

4.2 Web Service Modeling Language (WSML) Reasoning

The Web Service Modeling Language WSML is a formal language for the specification of ontologies and the modeling of Semantic Web services descriptions [3]. Several different WSML language variants exist, which are based upon different logical formalisms. More recently, WSML has been released in version 2.0 such that it provides an alignment of the LP-based variants with RIF and tractability of the WSML-DL dialect due to alignment of the semantics to ELP [7].

WSML2Reasoner is a highly modular framework that combines various validation, normalization and transformation algorithms that enable the translation

[10] `http://elly.sourceforge.net/`
[11] `http://www.gnu.org/licenses/lgpl.txt`
[12] Notably, Elly is listed on `http://www.w3.org/2007/OWL/wiki/Implementations` as reasoner for the OWL 2 EL and RL profiles.

of ontology descriptions in WSML to a generic syntax, which is interpretable by underlying reasoning engines.[13] A first task of WSML2Reasoner is the provisioning of translation modules for the mapping of WSML variants onto Datalog (with certain extensions) or ELP. Although designed to be extensible for other reasoners, the default release of WSML2Reasoner is shipped with IRIS and Elly only, as this offers the most comprehensive support for the semantics of the respective variants and all built-ins required by RIF-DTB. All in all, WSML2Reasoner yields a complete reasoning infrastructure for the WSML language family.

Online demonstrators for WSML-based reasoning are also available:

- WSML-DL v2.0, `http://iris.sti2.at/reasoners/wsml-dl-reasoner/`
- WSML-Rule v2.0, `http://iris.sti2.at/reasoners/wsml-rule-reasoner/`

5 Conclusions and Future Work

Going beyond the initial purpose of being a Datalog reasoner for WSML, IRIS – together with WSML2Reasoner and other presented software components – evolved to a comprehensive reasoning infrastructure for the (Semantic) Web. Important in this respect is a strict conformance to existing Web standards and different knowledge base representation formalisms, such as RIF, OWL and WSML.

In this paper we presented the current status of IRIS, and emphasized on the application and demonstration of IRIS as RIF-BLD reasoner. Although stable and already used in various projects, there is still further work to be done. To conclude the paper, we shortly address the two most relevant and currently ongoing improvements. IRIS currently relies on an in-memory Java implementation, and hence, performance-wise, cannot keep up to comparable reasoners that leverage database bindings; in particular when dealing with big and growing knowledge bases. To counter-act this limitation, work has started to persist facts in a relational database system. The reasoning algorithms would still be run on an in-memory object model, and hence, subsequent work will deal with moving the reasoning tasks (closer) to the database query engine. The envisaged result would be a feature-rich Datalog reasoner for the Web enhanced with the speed and scalability of a relational database engine.

A second enhancement currently under investigation in the LarKC project is the development of parallelized Datalog rule bases leveraging techniques such as data and rule partitioning [6] and map-reduce-style programming models [5]. The LarKC project develops a platform for massive distributed incomplete reasoning that shall remove the scalability barriers of currently existing reasoning systems for the Semantic Web.

Acknowledgment: The work on IRIS is supported by the EU FP7 IP SOA4All, and the e-Infrastructures project SEALS.

[13] `http://tools.sti2.at/wsml2reasoner/`

References

1. Bishop, B., Fischer, F.: IRIS - Integrated Rule Inference System. In: 1st Workshop on Advancing Reasoning on the Web: Scalability and Commonsense (June 2008)
2. Boley, H., Kifer, M.: RIF Basic Logic Dialect. W3C Recommendation (June 2010)
3. de Bruijn, J., Lausen, H., Polleres, A., Fensel, D.: The Web Service Modeling Language WSML: An Overview. In: 3rd European Semantic Web Conference. pp. 590–604 (June 2006)
4. de Bruijn, J.: RIF RDF and OWL Compatibility. W3C Recommendation (June 2010)
5. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM 51(1), 107–113 (January 2008)
6. Ganguly, S., Silberschatz, A., Tsur, S.: A Framework for the Parallel Processing of Datalog Queries. ACM SIGMOD Record 19(2), 143–152 (June 1990)
7. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable Rules for OWL 2. In: 7th Int'l Semantic Web Conference. pp. 649–664 (October 2008)
8. Malhotra, A., Melton, J., Walsh, N.: XQuery 1.0 and XPath 2.0 Functions and Operators. W3C Recommendation (January 2007)
9. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C Recommendation (October 2009)
10. Paschke, A., Hirtle, D., Ginsberg, A., Patranjan, P.L., McCabe, F.: RIF Use Cases and Requirements. W3C Working Draft (December 2008)
11. Peterson, D., Gao, S., Malhotra, A., Sperberg-McQueen, C.M., Thompson, H.S.: W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. W3C Working Draft (December 2009)
12. Polleres, A., Boley, H., Kifer, M.: RIF Datatypes and Built-Ins 1.0. W3C Recommendation (June 2010)
13. Ullman, J.D.: Principles of Database and Knowledge-Base Systems: Volume II: The New Technologies. W. H. Freeman & Co. (1990)
14. Winkler, D.: ELLY - An ELP Reasoner. Master Thesis, Semantic Technology Institute (STI) Innsbruck, University of Innsbruck (April 2010)