D1.2.2.1.1 Benchmarking the interoperability of ontology development tools using RDF(S) as interchange language APPENDIX

Raúl García-Castro (UPM)

with contributions from:
York Sure (UKARL)
Markus Zondler (UKARL)
Olivier Corby (INRIA)
Jesús Prieto-González (UPM)
Elena Paslaru Bontas (FU Berlin)
Lyndon Nixon (FU Berlin)
Malgorzata Mochol (FU Berlin)

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB
Deliverable D1.2.2.1.1 (WP 1.2 & WP2.1)

This document includes the appendix of deliverable D1.2.2.1.1.
Keyword list: benchmarking, benchmark suite, interoperability, RDF(S)

Document Identifier	KWEB/2006/D1.2.2.1.1/v1.5
Project	KWEB EU-IST-2004-507482
Version	v1.5
Date	3. August, 2006
State	final
Distribution	public

Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

University of Innsbruck (UIBK) - Coordinator
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

École Polytechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

France Telecom (FT)
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Institut National de Recherche en Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l'Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Contact person: Christoph Bussler
E-mail address: chris.bussler@deri.ie

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale
Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

University of Manchester (UoM)
Room 2.32. Kilburn Building, Department of Computer
Science, University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Sheffield (USFD)
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

Vrije Universiteit Brussel (VUB)
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

Centre for Research and Technology Hellas
École Polytechnique Fédérale de Lausanne
Free University of Bozen-Bolzano
Institut National de Recherche en Informatique et en Automatique
Learning Lab Lower Saxony
National University of Ireland Galway
Universidad Politécnica de Madrid
University of Karlsruhe
University of Manchester
University of Sheffield
University of Trento
Vrije Universiteit Amsterdam
Vrije Universiteit Brussel

Contents

Appendix A

RDF(S) detailed import results

A.1 KAON results

by YORK SURE AND MARKUS ZONDLER

A.1.1 Classes

Import of classes [*I01-02*]

KAON imports classes as concepts. It asks wether to insert comments in rdfs as labels or as documentation.

A.1.2 Metaclasses

Import of metaclasses [*I03-07*]

KAON can model metaclasses by using spanning instances of classes and declare these as types of other classes. KAON imports given benchmark files correctly.

A.1.3 Class hierarchies

Import of class hierarchies [*I08-10*]

KAON imports class hierarchies as concept hierarchies.

Import of class hierarchies with cycles [*I11-12*]

KAON doesn't allow to model cycles in class hierarchies and therefore present an error message while import process.

A.1.4 Classes related through properties

Import of classes with properties [*I13-18*]

KAON can model classes related through user defined properties. But when KAON imports given benchmark files just classes are imported and the information about properties gets lost.

A.1.5 Properties

Import of properties without domain and without range [*I19-20*]

KAON can model properties without domain or range and imports given benchmark files correctly.

Import of property hierarchies [*I21-25*]

KAON supports hierarchies in relations and imports given benchmark files correctly.

Import of properties with undefined resources as domain or range [*I26*]

KAON cannot model properties with undefined resources as domain or range. So when KAON imports given benchmark files nothing gets imported.

Import of properties with domain and without range [*I27-31*]

KAON can model properties with domain and without range and imports given benchmark files correctly.

Import of properties without domain and with range [*I32-38*]

KAON can model properties without domain and with range and imports given benchmark files correctly.

Import of properties with multiple domains [*I28,41,42,45,47*]

KAON can model properties with multiple domains and imports given benchmark files correctly.

Import of properties with multiple ranges [*I33,40,42*]

KAON can model properties with multiple ranges and imports given benchmark files correctly.

A.1.6 Instances

Import of instances of undefined resources [*I50*]

KAON cannot model just instances without corresponding classes.

Import of instances of one or multiple classes [*I51-53*]

KAON can model instances of one or multiple classes and imports given benchmark files correctly.

A.1.7 Instances and properties

Import of instances related through undefined properties [*I54-56*]

When KAON imports instances related through undefined object properties, it does not import the properties between the instances.

Import of instances related through defined properties [*I57-67*]

KAON can model every benchmark in this section and also imports given benchmark files correctly.

Import of instance properties whose range is *rdfs:Literal* [*I64*]

KAON interprets properties whose range is ”`http://www.w3.org/2000/01/rdf-schema#Literal`” as attributes.

Import of instance properties whose range is the XML Schema datatype ”string” [*I66,67*]

KAON creates new concept with id ”`http:www.w3.org/2001/XMLSchema#string`”.

A.1.8 Syntax and abbreviation

Import of resources with different URI reference syntaxes [*I68-71*]

KAON imports correctly resources with the different URI reference syntaxes.

Import of resources with different empty node syntaxes [*I72-73*]

KAON imports correctly resources with the different empty node syntaxes.

Import of resources with different multiple property syntaxes [*I74-75*]

KAON imports correctly resources with the different multiple properties shortened but throws an RDF exception while importing normal version of multiple properties.

Import of resources with different typed node syntaxes [*I76-77*]

KAON imports resources with the different typed node syntaxes correctly.

Import of resources with different string literal syntaxes [*I78-79*]

KAON imports correctly resources with the different string literal syntaxes.

Import of resources with different blank node syntaxes [*I80-81*]

KAON imports correctly resources with different blank node syntaxes.

Import of resources with *xml:lang* attributes [*I82*]

KAON does not import *xml:lang* attributes in properties.

A.2 Protégé results

by RAÚL GARCÍA-CASTRO

This section includes a detailed analysis of the results of evaluating the RDF(S) import capabilities of Protégé according to the different groups of benchmarks that have been carried out.

A.2.1 Classes

Import of classes [*I01-02*]

Protégé imports classes correctly.

A.2.2 Metaclasses

Import of metaclasses [*I03-07*]

Protégé imports metaclasses correctly except in the following case.

When Protégé imports a class that is instance of multiple metaclasses, it imports the class as instance of just one metaclass. This is not the expected result, as it should import the class as instance of all the metaclasses.

A.2.3 Class hierarchies

Import of class hierarchies [*I08-10*]

Protégé imports class hierarchies correctly.

Import of class hierarchies with cycles [*I11-12*]

Protégé cannot model cycles in class hierarchies. When Protégé finds a cycle in a class hierarchy it crashes, and does not import anything.

A.2.4 Classes related through properties

Import of classes with properties [*I13-18*]

Protégé does not allow to change its system classes when importing RDF(S) files. Therefore, when importing a class with a property, as it cannot create an own slot in the class' metaclass, it imports the property but does not define its domain and range.

A.2.5 Property hierarchies

Property hierarchies [*I21-23*]

Protégé imports property hierarchies as slot hierarchies.

Property hierarchies with cycles [*I24-25*]

Protégé cannot model cycles in property hierarchies. When Protégé finds a cycle in a property hierarchy, it crashes, and does not import anything.

A.2.6 Properties

Import of properties without domain and without range [*I19-20*]

Protégé imports properties without domain and without range as slots without domain and with a range of *Any*.

Import of properties with domain and without range [*I27-31*]

Protégé imports properties with domain and without range as slots with domain and with a range of *Any*.

Import of properties without domain and with range [*I32-38*]

Protégé imports properties without domain and with range as slots without domain and with range.

Import of properties with undefined resources as domain or range [*I26*]

When Protégé imports a property that has as domain (or range) a resource that is not defined, it creates the undefined resource as a class and creates the property with a domain (or range) of the class.

Import of properties with multiple domains [*I28,41-42,45,47*]

When Protégé imports a property that has multiple domains, it creates a slot with multiple domains.

This is not the expected result, as in Protégé multiple domains in slots are considered as the union of all the domains and in RDF(S) multiple domains in properties are considered the intersection of all the domains.

Import of properties with multiple ranges [*I33,40,42*]

When Protégé imports a property that has multiple ranges, it creates a slot with the range *Any,* as in Protégé multiple ranges in a property are considered as the union of all the ranges and in RDF(S) multiple ranges are considered as the intersection of all the ranges.

Import of properties with *rdfs:Class* as domain [*I30-31,35-36,48-49*]

Protégé does not allow to change its system classes. Therefore, when importing a property with domain *rdfs:Class*, it cannot create a template slot in :*STANDARD_CLASS* (Protégé's equivalent for *rdfs:Class)* for the property; it imports the property but does not define its domain.

Import of properties with *rdfs:Class* as range [*I35-36,48]*

When Protégé imports a property that has as range *rdfs:Class*, it imports *rdfs:Class* as the system class *:STANDARD_CLASS* (Protégé's equivalent for *rdfs:Class).*

Import of properties with *rdfs:Literal* as range [*I37-38,44-45,49,64-65*]

When Protégé imports a property that has as range *rdfs:Literal,* it creates the property as a slot with a range of Protégé's datatype *String*.

Import of properties with XML Schema datatype as range [*I46-47*]

When Protégé imports an ontology containing a property with range a XML Schema datatype, it creates the datatype as a class in the ontology with the *xsd* namespace.

Import of properties with a domain and a range [*I39,43*]

Protégé imports a property with a domain a class and a range another class, as a slot, even if the domain and range classes are the same.

A.2.7 Instances

Import of instances [*I51,53*]

Protégé imports instances correctly.

Import of instances of undefined resources [*I50*]

When Protégé imports an instance of a resource that is not defined, it does create the undefined resource as a class.

Import of instances of multiple classes [*I52*]

When Protégé imports an instance of multiple classes, it imports the instance as instance of just one class.

This is not the expected result, as it should import the instance as instance of all the classes.

A.2.8 Instances and properties

Import of instances related through undefined properties [*I54-55*]

When Protégé finds a property in an instance (an instance-property-object triple), and this property is not defined, it does not consider the property to belong to the domain class. Therefore, it creates the slot without domain and with a range of *Any*, and the property in the instance is lost. The range of the slot is *Any* even when the property relates an instance to a literal.

Import of instances related through properties [*I54-65*]

When Protégé imports instances of classes related through defined properties, it creates the instances and the properties correctly, even if the instances subject and object are instances of the same class, an instance has the same property with several other instances (either as subject or object), the instance is both the subject and object of the property, or the instance has the same property with several values.

Import of instances related through properties whose range is a XML Schema datatype [*I66-67*]

Protégé imports properties whose range is a XML Schema datatype as slots with range a class in the ontology with the *xsd* namespace. When Protégé finds a property in an instance with a value (an instance-property-value triple), it does not consider the value to be an instance of the datatype class. Therefore, the value of the property in the instance is lost.

A.2.9 Syntax and abbreviation

Import of resources with different URI reference syntaxes [*I68-71*]

Protégé imports correctly resources with the different URI reference syntaxes.

Import of resources with different empty node syntaxes [*I72-73]*

Protégé imports correctly resources with the empty node shortened syntax.

When Protégé imports an ontology containing a class definition with empty nodes unshortened it crashes, not importing anything, which is not the expected result.

Import of resources with different multiple property syntaxes [*I74-75*]

Protégé imports correctly resources with the different multiple property syntaxes.

Import of resources with different typed node syntaxes [*I76-77*]

Protégé imports correctly resources with the different typed node syntaxes.

Import of resources with different string literal syntaxes [*I78-79]*

Protégé imports correctly resources with the different string literal syntaxes.

Import of resources with different blank node syntaxes [*I80-81*]

Protégé imports correctly resources with blank nodes in the shortened syntax.

When Protégé imports an ontology containing blank nodes with identifiers, each time the blank node appears it is imported in Protégé as a new node, which is not the expected result.

Import of resources with *xml:lang* attributes [*I82*]

Protégé does not import *xml:lang* attributes in properties.

A.3 WebODE results

by RAÚL GARCÍA-CASTRO

This section includes a detailed analysis of the results of evaluating the RDF(S) import capabilities of WebODE according to the different groups of benchmarks that have been carried out.

A.3.1 Classes

Import of classes [*I01-02]*

WebODE imports classes as concepts. If the class has a *rdfs:label* property, it inserts ”rdfs:label -> class_name” in the description of the concept.

A.3.2 Metaclasses

Import of metaclasses [*I03-07*]

WebODE cannot model metaclasses. Therefore, it imports metaclasses as concepts but does not import the fact that certain concepts are instances of others. If a metaclass is not defined as a class in the exported file, the metaclass is not imported.

A.3.3 Class hierarchies

Import of class hierarchies [*I08-10*]

WebODE imports class hierarchies as concept hierarchies. If the object of a *rdfs:subClassOf* property is not defined as a class in the file, the class is not imported and consequently neither the *rdfs:subClassOf* property.

Import of class hierarchies with cycles [*I11-12*]

WebODE cannot model cycles in class hierarchies. When WebODE finds a cycle in a class hierarchy, it does not import all the *rdfs:subClassOf* properties, in order to create a taxonomy without cycles.

A.3.4 Classes related through properties

Import of classes with properties [*I13-18*]

WebODE cannot model classes related through user defined properties. When WebODE imports two classes related through a property whose domain and range are some meta-class of the classes, it does not import the property.

A.3.5 Properties

Import of properties without domain and without range [*I19-20]*

WebODE cannot model properties without domain or range.

When WebODE imports a property without domain and without range, it creates *rdfs:Resource* as an imported term and creates the property as a relation with origin and destination *rdfs:Resource.*

Import of properties with domain and without range *[I27-31*]

WebODE cannot model properties without domain or range.

When WebODE imports a property with domain and without range, it creates *rdfs:Resource* as an imported term and creates the property as a relation with destination *rdfs:Resource.*

Import of properties without domain and with range [*I32-38]*

WebODE cannot model properties without domain or range.

When WebODE imports a property without domain and with range, it creates *rdfs:Resource* as an imported term. If the range is *rdfs:Literal* it creates the property as an instance attribute of *rdfs:Resource* and if not, it creates the property as a relation with *rdfs:Resource* as origin.

Import of property hierarchies [*I21-25*]

WebODE does not support hierarchies in relations. Therefore, when it imports a property hierarchy, it does not import the *rdfs:subPropertyOf* properties.

Import of properties with undefined resources as domain or range [*I26*]

When WebODE imports a property that has as domain (or range) a resource that is not defined, it creates the undefined resource as a concept and creates the property with a domain (or range) of the concept.

Import of properties with multiple domains [*I28,41-42,45,47*]

When WebODE imports a property that has multiple domains, it creates an anonymous concept as the origin of the relation and as subclass of one of the domain concepts.

This is not the expected result, as the anonymous concept should be created as subclass of all the domain concepts.

Import of properties with multiple ranges [*I33,40,42*]

When WebODE imports a property that has multiple ranges, it creates an anonymous concept *as* the destination of the relation and as subclass of one of the range concepts.

This is not the expected result, as the anonymous concept should be created as subclass of all the range concepts.

Import of properties with *rdfs:Class* as domain or range [*I30-31,35-36,48-49*]

When WebODE imports a property that has as domain or range *rdfs:Class*, it imports *rdfs:Class* as an imported term.

Import of properties with *rdfs:Literal* as range [*I37-38,44-45,49,64-65*]

When WebODE imports a property that has as range *rdfs:Literal*, it creates the property as an instance attribute with a type of WebODE's datatype *String*.

Import of properties with XML Schema datatype as range [*I46-47*]

When WebODE imports an ontology containing a property whose range is a XML Schema datatype, it considers the property as a relation between the concept and the datatype, being the datatype imported as an imported term.

This is not the expected result, as the property should be created as an instance attribute with type the XML Schema datatype.

Import of properties with a domain and a range [*I39,43*]

When WebODE imports a property whose domain is a class and whose range is another class, it creates the property as a relation, even if the domain and range classes are the same.

A.3.6 Instances

Import of instances [*I51,53*]

When WebODE imports an instance of a class, it creates an instance set for storing the instance and creates the instance correctly. If the instance has a *rdfs:label* property, it inserts ”rdfs:label -> instance_name” in the description of the instance.

Import of instances of undefined resources [*I50*]

When WebODE imports an instance of a resource that is not defined, it does not import the instance.

Import of instances of multiple classes [*I52*]

WebODE does not support instances of multiple classes.

When WebODE imports an instance of multiple classes, it imports the instance as instance of just one concept.

A.3.7 Instances and properties

Import of instances related through undefined properties [*I54-55*]

When WebODE imports instances related through undefined object properties, it does not import the properties between the instances.

Import of instances related through properties [*I54-65*]

When WebODE imports instances of classes related through defined properties, it creates the instances and the properties correctly, even if the instances subject and object are instances of the same class, an instance has the same property with several other instances (either as subject or object), the instance is both the subject and object of the property, or the instance has the same property with several values.

Import of instances related through properties whose range is a XML Schema datatype [I66-67]

When WebODE imports a property whose range is a XML Schema datatype and instances with values in the property, it considers WebODE's datatype *String* as the type of the instance attribute.

This is not the expected result, as the type of the instance attribute should be the XML Schema datatype.

A.3.8 Syntax and abbreviation

Import of resources with different URI reference syntaxes *[I68-71]*

WebODE imports correctly resources with the different URI reference syntaxes.

Import of resources with different empty node syntaxes [*I72-73*]

WebODE imports correctly resources with the different empty node syntaxes.

Import of resources with different multiple property syntaxes [*I74-75*]

WebODE imports correctly resources with the different multiple property syntaxes.

Import of resources with different typed node syntaxes [*I76-77*]

WebODE imports correctly resources with the different typed node syntaxes.

Import of resources with different string literal syntaxes [*I78-79*]

WebODE imports correctly resources with the different string literal syntaxes.

Import of resources with different blank node syntaxes [*I80-81*]

When WebODE imports an ontology containing blank nodes, it imports the blank node as an anonym instance of the corresponding concept and assigns to this instance the properties of the blank node.

Import of resources with *xml:lang* attributes [*I82*]

WebODE does not import *xml:lang* attributes in properties.

Appendix B

RDF(S) detailed export results

B.1 KAON results

by YORK SURE AND MARKUS ZONDLER

B.1.1 Classes

Export of classes [*E01-02*]

KAON exports concepts as classes correctly.

B.1.2 Metaclasses

Export of metaclasses [*E03-07*]

KAON can model metaclasses by using spanning instances.

B.1.3 Class hierarchies

Export of class hierarchies [*E08-12*]

KAON can model concept hierarchies and exports them as class hierarchies.

Export of class hierarchies with cycles [*E11-12*]

KAON doesn't allow to model concept hierarchies with cycles and, therefore, cannot export them.

B.1.4 Classes related through object or datatype properties

Export of classes with object properties [*E13-16*]

KAON cannot instantiate a property between classes if the definition of this property is supposed to be defined with a domain and a range of some metaclass of the classes. Due to some misunderstanding in the modelling process these benchmarks were marked as successfully passed by mistake.

Export of classes with datatype properties [*E17-18*]

KAON cannot model classes which have datatype properties with literals.

B.1.5 Datatype properties

Export of datatype properties without range [*E19-20,22-24*]

KAON seems to insert a range to *rdfs:Literal* automatically in the export process if there is no given range information for a property. If it imports these files again it ignores this range. But other applications seem to have some problems with this behaviour as it's not the expected one.

Export of datatype properties without domain and without range [*E19-20*]

KAON can model attributes not related to a concept or without type and, therefore, can export them. In the modeler it looks like kaon:Root is domain of such a property. But in an exported RDF file there is no further information about the domain.

Export of datatype properties with domain and without range [*E22-24*]

KAON can model attributes without type and, therefore, can export them.

Export of datatype properties without domain and with range [*E25-26*]

KAON can model instance attributes not related to a concept and, therefore, cannot export them.

Export of datatype properties with undefined resources as domain [*E21*]

KAON cannot model attributes of an undefined resource and, therefore, cannot export them. Due to some misunderstanding in the modelling process this benchmark was marked as by mistake successfully passed.

Export of datatype properties with multiple domains [*E23,28,30*]

KAON can model instance attributes of several concepts and, therefore, can export them.

Export of datatype properties with a domain and a range [*E27,29*]

KAON exports instance attributes as properties correctly.

Export of datatype properties whose range is *String* or *Integer* [*E25-30*]

KAON can model and exports instance attributes with type String or Integer. But this requires extra concepts to be modelled or imported from an external ontology with id `http://www.w3.org/2000/01/rdf-schema#string` or `http://www.w3.org/2000/01/rdf-schema#integer` and used as range.

Export of datatype property with XML Schema datatype integer as range and multiple domains[*E30*]

The exported benchmark file contains just one domain by mistake as it's possible to assign multiple domains to a property.

B.1.6 Object properties

Export of object properties without domain and without range [*E31-32*]

KAON can model relations without origin or destination and, therefore, can export them.

Export of object properties with domain and without range [*E34-36*]

KAON can model relations without destination and, therefore, can export them.

Export of object properties without domain and with range [*E37-39*]

KAON can model relations without origin and, therefore, can export them.

Export of object properties with undefined resources as domain or range [*E33*]

KAON cannot model relations with undefined resources as origin or destination and, therefore, cannot export them. Due to some misunderstanding in the modelling process these benchmarks were marked by mistake as successfully passed.

Export of object properties with multiple domains [*E35,42-43*]

KAON can model relations with multiple origins and, therefore, can export them.

Export of object properties with multiple ranges [*E38,41,43*]

KAON can model relations with multiple destinations and, therefore, can export them.

Export of object properties with a domain and a range [*E40,44*]

KAON models and exports relations as properties.

B.1.7 Instances

Export of instances of undefined resources [*E45*]

KAON cannot model instances of undefined resources and, therefore, cannot export them.

Export of instances [*E46-48*]

KAON models concepts and instances successfully and exports them correctly.

Export of instances related through undefined object properties [*E49-50*]

KAON cannot model instances related through undefined relations and, therefore, cannot export them.

Export of instances related through object properties [*E51-57*]

KAON exports instances related through relations as instances related by properties, even if the instances origin and destination are instance of the same class, an instance has the same relation with several other instances (either as origin or destination), or the instance is both the original and destination of the relation.

B.1.8 Instances and datatype properties

Export of instances related through undefined datatype properties [*E58*]

KAON cannot model instances with undefined instance attributes and, therefore, cannot export them.

Export of instances with datatype properties [*E59-60*]

KAON can model and export class, instance and property.

Export of instances with datatype properties [*E61-62*]

KAON cannot create these properties for an instance and therefore cannot export them.

B.1.9 URI character restrictions

Export of concepts and properties whose names start with a character that is not a letter or ’ ’ [*E63*]

KAON exports concepts and properties whose names start with a character that is not a letter or ’ ’.

Export of concepts and properties with spaces in their names [*E64*]

KAON encodes spaces as ” ” when exporting concepts and properties with spaces in their names. It also adds extra label element to element definitions.

Export of concepts and properties with URI reserved characters in their names (’;’, ’/’, ’?’, ’:’, ’@’, ’&’, ’=’, ’+’, ’$’, ’,’) [*E65*]

KAON exports concepts and properties with URI reserved characters in their names. It also adds extra label element to element definitions.

Export of concepts and properties with XML delimiter characters in their names (’¡’, ’¿’, ’#’, ’%’, ’”’) [*E66*]

KAON exports concepts and properties with the character ’”’ in their names. It also adds extra label element to element definitions.

B.2 Protégé results

by RAÚL GARCÍA-CASTRO

This section includes a detailed analysis of the results of evaluating the RDF(S) export capabilities of Protégé according to the different groups of benchmarks that have been carried out.

B.2.1 Classes

Export of classes [*E01-02*]

Protégé exports classes as subclass of *rdfs:Resource*, it also inserts a *rdfs:label* property into the class with the name of the class.

B.2.2 Metaclasses

Export of metaclasses [*E03-07*]

Protégé exports metaclasses as classes that are subclass of *rdfs:Class*.

If a class is instance of several metaclasses, it exports the class as instance of just one of the metaclasses. This is not the expected result, as the class should be exported as instance of all the metaclasses.

B.2.3 Class hierarchies

Export of class hierarchies [*E08-12]*

Protégé exports class hierarchies correctly.

Export of class hierarchies with cycles [*E11-12*]

Protégé cannot model class hierarchies with cycles and, therefore, cannot export them.

B.2.4 Classes related through object or datatype properties

Export of classes with object properties [*E13-16*]

Protégé cannot model classes related through undefined template slots and, therefore, cannot export them. The template slot is supposed to be defined with a domain and a range of some metaclass of the classes.

Export of classes with datatype properties [*E17-18*]

Protégé cannot model classes related through undefined template slots and, therefore, cannot export them. The template slot is supposed to be defined with a domain and a range of some metaclass of the classes.

B.2.5 Datatype properties

Export of datatype properties without domain and without range [*E19-20*]

Protégé exports template slots without domain and without range as properties without domain and without range.

Export of datatype properties with domain and without range [*E22-24*]

Protégé exports template slots without range as properties without range.

Export of datatype properties without domain and with range [*E25-26*]

Protégé exports template slots without domain as properties without domain.

Export of datatype properties with undefined resources as domain [*E21*]

Protégé cannot model template slots of an undefined resource and, therefore, cannot export them.

Export of datatype properties with multiple domains [*E23,28,30*]

Protégé exports a template slot with multiple domains as a property with multiple domains.

This is not the expected result, as in Protégé multiple domains in template slots are considered as the union of all the domains and in RDF(S) multiple domains in properties are considered the intersection of all the domains.

Export of datatype properties with a domain and a range [*E27,29*]

Protégé exports template slots as properties, it also inserts a *rdfs:label* property into the property with the name of the template slot.

Export of datatype properties whose range is *String [E25-28*]

Protégé exports template slots whose range is *String* as properties with range *rdfs:Literal*.

Export of datatype properties with XML Schema datatype as range [*E29-30*]

Protégé cannot model datatype properties with XML Schema datatype as range and, therefore, cannot export them.

B.2.6 Object properties

Export of object properties without domain and without range [*E31-32*]

Protégé exports template slots without domain and with a range of *Instance* with no allowed classes as properties without domain and with range *rdfs:Resource.*

Export of object properties with domain and without range [*E34-36*]

Protégé exports template slots with a range of *Instance* and with no allowed classes as properties with range *rdfs:Resource.*

Export of object properties without domain and with range [*E37-39*]

Protégé exports template slots without domain as properties without domain.

Export of object properties with undefined resources as domain or range [*E33*]

Protégé cannot model template slots with undefined resources as domain or range and, therefore, cannot export them.

Export of object properties with multiple domains [*E35,42-43*]

Protégé exports a template slot with multiple domains as a property with multiple domains.

This is not the expected result, as in Protégé multiple domains in template slots are considered as the union of all the domains and in RDF(S) multiple domains in properties are considered the intersection of all the domains.

Export of object properties with multiple ranges [*E38,41,43*]

Protégé exports a template slot with multiple ranges as a property with a range of *rdfs:Resource.*

Export of object properties with a domain and a range [*E40,44*]

Protégé exports template slots as properties, it also inserts a *rdfs:label* property into the property with the name of the template slot.

B.2.7 Instances

Export of instances [*E46,48*]

Protégé exports instances correctly, it also inserts a *rdfs:label* property into the instance with the name of the instance.

Export of instances of undefined resources [*E45*]

Protégé cannot model instances of undefined resources and, therefore, cannot export them.

Export of instances of multiple classes [*E47*]

Protégé exports an instance of multiple classes as an instance of just one of the classes.

This is not the expected result, as the instance should be exported as instance of all the classes.

B.2.8 Instances and object properties

Export of instances related through object properties [*E51-57*]

Protégé exports instances related through template slots as instances related through properties, even if the instances domain and range are instances of the same class, an instance has the same template slot with several other instances (either as domain or range), or the instance is both the domain and range of the template slot.

Export of instances related through undefined object properties [*E49-50*]

Protégé cannot model instances related through undefined template slots and, therefore, cannot export them.

B.2.9 Instances and datatype properties

Export of instances with datatype properties [*E59-60*]

Protégé exports instances with template slots as instances with properties, even if an instance has several values in the template slot.

Export of instances with datatype properties with XML Schema datatype as range

[*E61-62*]

Protégé cannot model template slots with XML Schema datatype as range and, therefore, cannot export them.

Export of instances related through undefined datatype properties [*E58*]

Protégé cannot model instances with undefined template slots and, therefore, cannot export them.

B.2.10 URI character restrictions

Export of classes and properties whose names start with a character that is not a letter or '_' [*E63*]

Protégé exports classes and template slots whose names start with a character that is not a letter or "_" by inserting "_" as the first character of the name.

Export of classes and properties with spaces in their names [*E64*]

Protégé encodes spaces as "_" when exporting classes and template slots with spaces in their names.

Export of classes and properties with URI reserved characters in their names (';', '/', '?', ':', '@', '&', '=', '+', '$', ',') [*E65*]

Protégé encodes "/", "=", and "$" as "_" when exporting classes and template slots with these characters in their names.

Export of classes and properties with XML delimiter characters in their names (’$<$’, ’$>$’, ’#’, ’%’, ’”’) [*E66*]

Protégé encodes XML delimiter characters as ”_” when exporting classes and template slots with these characters in their names.

B.3 WebODE results

by RAÚL GARCÍA-CASTRO

This section includes a detailed analysis of the results of evaluating the RDF(S) export capabilities of WebODE according to the different groups of benchmarks that have been carried out.

B.3.1 Classes

Export of classes [*E01-02*]

WebODE exports concepts as classes, it also inserts a *rdfs:label* property into the class with the name of the concept.

B.3.2 Metaclasses

Export of metaclasses [*E03-07*]

WebODE cannot model metaclasses and, therefore, cannot export them.

B.3.3 Class hierarchies

Export of class hierarchies [*E08-12*]

WebODE exports concept hierarchies as class hierarchies.

Export of class hierarchies with cycles [*E11-12*]

WebODE cannot model concept hierarchies with cycles and, therefore, cannot export them.

B.3.4 Classes related through object or datatype properties

Export of classes with object properties [*E13-16*]

WebODE cannot model concepts related through relations and, therefore, cannot export them. The relation is supposed to be defined with an origin and a destination of some metaconcept of the concepts.

Export of classes with datatype properties [*E17-18*]

WebODE cannot model a concept related through instance attributes and, therefore, cannot export them. The instance attributes are supposed to be defined in a metaconcept of the concept.

B.3.5 Datatype properties

Export of datatype properties without domain and without range [*E19-20*]

WebODE cannot model instance attributes not related to a concept or without type and, therefore, cannot export them.

Export of datatype properties with domain and without range [*E22-24*]

WebODE cannot model instance attributes without type and, therefore, cannot export them.

Export of datatype properties without domain and with range [*E25-26*]

WebODE cannot model instance attributes not related to a concept and, therefore, cannot export them.

Export of datatype properties with undefined resources as domain [*E21*]

WebODE cannot model instance attributes of an undefined resource and, therefore, cannot export them.

Export of datatype properties with multiple domains [*E23,28,30*]

WebODE cannot model instance attributes of several concepts and, therefore, cannot export them.

Export of datatype properties with a domain and a range [*E27,29*]

WebODE exports instance attributes as properties, it also inserts a *rdfs:label* property into the property with the name of the instance attribute.

WebODE exports all its own datatypes as *rdfs:Literal*.

Export of datatype properties whose range is *String [E25-28*]

WebODE exports instance attributes with type *String* as properties with range *rdfs:Literal*.

Export of datatype properties with XML Schema datatype as range [*E29-30*]

WebODE exports instance attributes with type an XML Schema datatype as properties with range the XML Schema datatype.

B.3.6 Object properties

Export of object properties without domain and without range [*E31-32*]

WebODE cannot model relations without origin or destination and, therefore, cannot export them.

Export of object properties with domain and without range [*E34-36*]

WebODE cannot model relations without destination and, therefore, cannot export them.

Export of object properties without domain and with range [*E37-39*]

WebODE cannot model relations without origin and, therefore, cannot export them.

Export of object properties with undefined resources as domain or range [*E33*]

WebODE cannot model relations with undefined resources as origin or destination and, therefore, cannot export them.

Export of object properties with multiple domains [*E35,42-43*]

WebODE cannot model relations with multiple origins and, therefore, cannot export them.

Export of object properties with multiple ranges [*E38,41,43*]

WebODE cannot model relations with multiple destinations and, therefore, cannot export them.

Export of object properties with a domain and a range [*E40,44*]

WebODE exports relations as properties, it also inserts a *rdfs:label* property into the property with the name of the relation.

B.3.7 Instances

Export of instance sets [*E46,48*]

WebODE does not export instance sets in ontologies.

Export of instances [*E46,48*]

WebODE exports instances correctly, it also inserts a *rdfs:label* property into the instance with the name of the instance.

Export of instances of undefined resources [*E45*]

WebODE cannot model instances of undefined resources and, therefore, cannot export them.

Export of instances of multiple classes [*E47*]

WebODE cannot model instances of multiple classes and, therefore, cannot export them.

B.3.8 Instances and object properties

Export of instances related through object properties [*E51-57*]

WebODE exports instances related through relations as instances related through properties, even if the instances origin and destination are instances of the same concept, an instance has the same relation with several other instances (either as origin or destination), or the instance is both the origin and destination of the relation.

Export of instances related through undefined object properties [*E49-50*]

WebODE cannot model instances related through undefined relations and, therefore, cannot export them.

B.3.9 Instances and datatype properties

Export of instances with datatype properties [*E59-62*]

WebODE exports instances with instance attributes as instances with properties, even if an instance has several values in the instance attribute, or the instance attribute has an XML Schema datatype as type.

Export of instances related through undefined datatype properties [*E58*]

WebODE cannot model instances with undefined instance attributes and, therefore, cannot export them.

B.3.10 URI character restrictions

Export of classes and properties whose names start with a character that is not a letter or '_' [*E63*]

WebODE exports concepts and properties whose names start with a character that is not a letter or '_'. The resulting file is validated by the W3C RDF Validator, but the RDF produced is not standard.

Export of classes and properties with spaces in their names [*E64*]

WebODE encodes spaces as "_" when exporting concepts and properties with spaces in their names.

Export of classes and properties with URI reserved characters in their names (';', '/', '?', ':', '@', '&', '=', '+', '$', ',') [*E65*]

WebODE exports concepts and properties with URI reserved characters in their names. The resulting file is validated by the W3C RDF Validator, but the RDF produced is not standard.

Export of classes and properties with XML delimiter characters in their names (’$<$’, ’$>$’, ’#’, ’%’, ’”’) [*E66*]

WebODE cannot model concepts or properties with the character ’ ” ’ in their names and, therefore, cannot export them.

Appendix C

RDF(S) detailed interoperability results

This chapter includes a detailed analysis of the results of the interoperability experiments for each tool.

The analysis performed for each tool includes the interoperability results when the tool is the destination of the interchange and the other tools are the origin of the interchange. Therefore, for consulting the interoperability results between an origin tool and a destination one, the section to take into account is that of the tool that is the destination of the interchange.

The sections devoted to each tool include comments about the tool results classified according to the benchmark groups present in the RDF(S) Interoperability Benchmark Suite. These comments are of two types. general comments that affect all the tools and tool-specific comments that override the general comments for the mentioned tools.

The benchmarks related to a certain comment are identified according to the structure *<Tool>:<Type><Range>*, where:

- *<Tool>* is optional and is the name of the tool that the comment refers to.
- *<Type>* is ”E”, ”I” or ”In” indicating that the benchmarks are from the export, import or interoperability benchmark suites respectively.
- *<Range>* is composed of benchmark numbers or benchmarks ranges *(num1 -num2)* separated by commas.

For example, ”Protégé:E01,E08” and ”In:17-19” are benchmark identifications.

In the analysis of the results, the terms *send* and *receive* are used to define the actions involved in the interchange of an ontology. Therefore, when reading ”Tool X receives classes from Tool Y”, the intended meaning is ”Tool X imports classes from RDF(S) that have been exported before by Tool Y”.

Part (or all) of the benchmark executions have been performed manually. Therefore the results may contain inconsistencies or errors, either when executing the benchmark or when modelling the described ontologies. Nevertheless, having all the results available can help to detect this kind of errors and inconsistencies. In the cases where some inconsistency has been detected, a sentence such as ”Is this a mistake when modelling the benchmark?” or similar is included.

When a tool has not provided results for a certain benchmark, a sentence such as ”There are no results for Tool X.” or similar is included.

When an origin tool is not able of modelling the ontology described in a benchmark and, therefore, to interchange that ontology with another destination tool, a sentence such as ”Tool X cannot receive some component from Tool Y as the latter cannot model them.” or similar is included in the destination tool results.

Sometimes, the symbol ♠ is placed next to an origin tool name, meaning that in some or all the cases the interchange between that origin tool and the destination one does not produce its expected result.

C.1 KAON results

by YORK SURE AND MARKUS ZONDLER

C.1.1 Classes

Interchange of classes [*In01-02*]

KAON receives correctly classes from the other tools.

Corese. Classes are created correctly.

KAON. Classes are created correctly.

Protege. Classes are created correctly. *rdfs:Resource* is created as an imported term and the classes are created as subclass of *rdfs:Resource*, as Protégé exports classes as subclass of *rdfs:Resource*.

WebODE. Classes are created correctly.

C.1.2 Metaclasses

Interchange of metaclasses [*In03-07*]

KAON can model metaclasses by using spanning instances of classes and declare these as types of other classes.

Corese. Classes are created correctly. Additionally a class *rdfs:Class* is created which also occurs in exported RDF file of Corese.

KAON. Classes are created correctly.

Protege. In [*In03,05*] classes are created correctly and a *rdfs:Class* is created as an imported term Classes in [*In04,06,07*] are not created correctly

C.1.3 Class hierarchies

Interchange of class hierarchies [*In08-10*]

KAON receives correctly class hierarchies from other tools.

Corese. Class hierarchies are created correctly.

KAON. Class hierarchies are created correctly.

Protege. Class hierarchies are created correctly.

WebODE. Class hierarchies are created correctly.

Interchange of class hierarchies with cycles [*In11-12*]

KAON cannot model cycles in class hierarchies.

Corese. KOAN executes the import and also shows a graph which looks like it's correct. But clicking on a concept doesn't reveal any information about it.

KAON. KAON cannot receive cycles in class hierarchies from KAON as the latter cannot model them.

Protege. KAON cannot receive cycles in class hierarchies from Protégé as the latter cannot model them.

WebODE. KAON cannot receive cycles in class hierarchies from WebODE as the latter cannot model them.

C.1.4 Classes related through object or datatype properties

Interchange of classes with object properties [*In13-16*]

KAON cannot instantiate a property between classes if the definition of this property is supposed to be defined with a domain and a range of some metaclass of the classes. Due to some misunderstanding in the modelling process these benchmarks were marked as successfully passed by mistake.

Corese. The object properties are not created.

KAON. KAON cannot receive classes related through object properties from KAON as the latter cannot model them.

Protege. KAON cannot receive classes related through object properties from Protégé as the latter cannot model them.

WebODE. KAON cannot receive classes related through object properties from WebODE as the latter cannot model them.

Interchange of classes with datatype properties [*In17-18*]

KAON cannot model classes which have datatype properties with literals.

Corese. The datatype properties are not created.

KAON. KAON cannot receive classes related through datatype properties from KAON as the latter cannot model them.

Protege. KAON cannot receive classes related through datatype properties from Protégé as the latter cannot model them.

WebODE. KAON cannot receive classes related through datatype properties from WebODE as the latter cannot model them.

C.1.5 Datatype properties

Interchange of datatype properties without domain and without range [*In19-20*]

KAON can model datatype properties without domain and without range. In the modeler it looks like kaon:Root is domain of such a property. But in an exported RDF file there is no further information about the domain.

Corese. Datatype properties are created with a new concept as range representing the datatype.

KAON. Datatype properties are created correctly.

Protege. Datatype properties are created correctly.

WebODE. KAON cannot receive datatype properties without domain and without range from WebODE as the latter cannot model them.

Interchange of datatype properties with undefined resources as domain [*In21*]

KAON cannot model attributes of an undefined resource and, therefore, cannot export them. Due to some misunderstanding in the modelling process this benchmark was marked as by mistake successfully passed.

Corese. Datatype properties are created with a new concept as range representing the datatype.

KAON. KAON cannot receive datatype properties with undefined resources as domain from KAON as the latter cannot model them.

Protege. KAON cannot receive datatype properties with undefined resources as domain from Protégé as the latter cannot model them.

WebODE. KAON cannot receive datatype properties with undefined resources as domain from WebODE as the latter cannot model them.

Interchange of datatype properties with domain and without range [*In22-24*]

KAON can model datatype properties with domain and without range.

Corese. Datatype properties and classes are created correctly. Extra concepts are created additionally as parent classes of domain class.

KAON. Datatype properties and classes are created correctly.

Protege. Datatype properties and classes are created correctly.

WebODE. KAON cannot receive datatype properties with domain and without range from WebODE as the latter cannot model them.

Interchange of datatype properties without domain and with range [*In25-26*]

KAON can model datatype properties without domain and with range.

Corese. Datatype properties and classes are created correctly. Extra concepts are created additionally as parent classes of domain class.

KAON. Datatype properties and classes are created correctly.

Protege. KAON cannot import datatype properties and classes correctly. Information about range gets lost.

WebODE. KAON cannot receive datatype properties without domain and with range from WebODE as the latter cannot model them.

Interchange of datatype properties with multiple domains [*In23,28,30*]

KAON can model datatype properties with multiple domains.

Corese. Datatype properties and classes are created correctly.

KAON. Datatype properties and classes are created correctly.

Protege. Datatype properties and classes are created correctly in [*In23,28*]. For [*In30*] KAON cannot receive datatype properties with multiple domains from Protégé as the latter cannot model them.

WebODE. KAON cannot receive datatype properties with multiple domains from WebODE as the latter cannot model them.

Interchange of datatype properties whose range is *String* [*In25-28*]

Corese. KAON creates properties and classes correctly. It adds extra concept for datatype ”String” which is subconcept of another extra concept ”Datatype”.

KAON. Datatype properties and classes are created correctly.

Protege. KAON cannot import datatype properties and classes correctly. Information about range gets lost.

WebODE. For [*In25,26,28*] KAON cannot receive datatype properties fromWebODE as the latter cannot model them. Benchmark [*In27*] fails because information about range gets lost.

Interchange of datatype properties with XML Schema datatype as range [*In29-30*]

KAON imports correctly a datatype property that has an XML Schema datatype as range.

Corese. KAON creates properties and classes correctly. It adds extra concept for datatype ”String” which is subconcept of another extra concept ”Datatype”.

KAON. Datatype properties and classes are created correctly.

Protege. KAON cannot receive datatype properties with XML Schema datatype as range from Protégé as the latter cannot model them.

WebODE. For [*In29*] the datatype property with XML Schema datatype as range is created correctly. For [*In30*] KAON cannot receive datatype properties with XML Schema datatype as range from WebODE as the latter cannot model them.

C.1.6 Object properties

Interchange of object properties without domain and without range [*In31-32*]

KAON can model object properties without domain or range.

Corese. The object properties are created with a range of a class. Is this a mistake when modelling the benchmark?

KAON. The object properties are created correctly.

Protege. The object properties are created correctly.

WebODE. KAON cannot receive object properties without domain and without range from WebODE as the latter cannot model them.

Interchange of object properties with undefined resources as domain or range [*In33*]

KAON cannot model relations with undefined resources as origin or destination and, therefore, cannot export them. Due to some misunderstanding in the modelling process the export benchmarks were marked by mistake as successfully passed.

Corese. KAON imports nothing from RDF file.

KAON. KAON cannot receive object properties with undefined resources as domain or range from KAON as the latter cannot model them.

Protege. KAON cannot receive object properties with undefined resources as domain or range from Protégé as the latter cannot model them.

WebODE. KAON cannot receive object properties with undefined resources as domain or range from WebODE as the latter cannot model them.

Interchange of object properties with domain and without range [*In34-36*]

KAON can model object properties with domain and without range.

Corese. The object properties are created correctly.

KAON. The object properties are created correctly.

Protege. The object properties are created correctly.

WebODE. KAON cannot receive object properties with domain and without range from WebODE as the latter cannot model them.

Interchange of object properties without domain and with range [*In37-39*]

KAON can model object properties without domain and with range.

Corese. The object properties are created correctly.

KAON. The object properties are created correctly.

Protege. The object properties are created correctly.

WebODE. KAON cannot receive object properties without domain and with range from WebODE as the latter cannot model them.

Interchange of object properties with multiple domains [*In35,42-43*]

Corese. The object properties are created correctly.

KAON. The object properties are created correctly.

Protege. The object properties are created correctly.

WebODE. KAON cannot receive object properties with multiple domains fromWebODE as the latter cannot model them.

Interchange of object properties with multiple ranges [*In38,41,43*]

Corese. The object properties are created correctly.

KAON. The object properties are created correctly.

Protege. The object properties are created correctly.

WebODE. KAON cannot receive object properties withmultiple domains fromWebODE as the latter cannot model them.

Interchange of object properties with a domain and a range [*In40,44*]

Corese. The object property is created correctly.

KAON. The object property is created correctly.

Protege. The object property is created correctly.

WebODE. The object property is created correctly.

C.1.7 Instances

Interchange of instances of undefined resources [*In45*]

When KAON imports an instance of a resource that is not defined, it does not import the instance.

Corese. The instance and the undefined resource are not created.

KAON. KAON cannot receive instances of undefined resources from KAON as the latter cannot model them.

Protege. KAON cannot receive instances of undefined resources from Protégé as the latter cannot model them.

WebODE. KAON cannot receive instances of undefined resources from WebODE as the latter cannot model them.

Interchange of instances [*In46,48*]

When KAON imports an instance of a class, it creates the instance correctly.

Corese. The instance is created correctly.

KAON. The instance is created correctly.

Protege. The instance is created correctly.

WebODE. The instance is created correctly.

Interchange of instances of multiple classes [*In47*]

KAON does support instances of multiple classes.

Corese. The instances are imported correctly.

KAON. The instances are imported correctly.

Protege. The instances are imported correctly.

WebODE. KAON cannot receive instances of multiple classes from WebODE as the latter cannot model them.

C.1.8 Instances and object properties

Interchange of instances related through undefined object properties [*In49-50*]

When KAON imports instances related through undefined object properties, it does not import object properties between these instances.

Corese. The object properties are not created.

KAON. KAON cannot receive instances related through undefined object properties from KAON as the latter cannot model them.

Protege. KAON cannot receive instances related through undefined object properties from Protégé as the latter cannot model them.

WebODE. KAON cannot receive instances related through undefined object properties from WebODE as the latter cannot model them.

Interchange of instances related through object properties [*In51-57*]

Corese. The instances and the object properties are created correctly.

KAON. The instances and the object properties are created correctly.

Protege. The instances and the object properties are created correctly.

WebODE. The instances and the object properties are created correctly.

C.1.9 Instances and datatype properties

Interchange of instances related through undefined datatype properties [*In58*]

When KAON imports instances related through undefined datatype properties, it does not import the datatype properties between the instances.

Corese. The datatype properties are not created. The instance is not created as the class is undefined in the exported file.

KAON. KAON cannot receive instances related through undefined datatype properties from KAON as the latter cannot model them.

Protege. KAON cannot receive instances related through undefined datatype properties from Protégé as the latter cannot model them.

WebODE. KAON cannot receive instances related through undefined datatype properties from WebODE as the latter cannot model them.

Interchange of instances related through datatype properties [*In59-62*]

Corese. The instances, the properties and the values are created correctly for [*In59-61*]. KAON cannot receive instances related through datatype properties with a XML Schema datatype as range from Corese as the latter cannot model them.

KAON. The instances, the properties and the values are created correctly for [*In59-60*], even if an instance has the same property with several values. For [*In61-62*] KAON cannot receive instances related through datatype properties with a XML Schema datatype as range from KAON as the latter cannot model them.

Protege. For [*In59-60*] information about range of properties is not imported. For [*In61-62*] KAON cannot receive instances related through datatype properties with a XML Schema datatype as range from Protégé as the latter cannot model them.

WebODE. The instances, the properties and the values are created correctly, even if an instance has the same property with several values or the datatype has an XML Schema datatype as range.

C.1.10 URI character restrictions

Interchange of concepts and properties whose names start with a character that is not a letter or ' ' [*In63*]

KAON receives correctly concepts and properties whose names start with a character that is not a letter or ' '.

Corese. There are no results for Corese.

KAON. The concepts and properties are created correctly.

Protege. Information about range of property is not imported.

WebODE. The concepts and properties are created correctly.

Interchange of concepts and properties with spaces in their names [*In64*]

Corese. There are no results for Corese.

KAON. The concepts and properties are created correctly.

Protege. Information about range of property is not imported.

WebODE. The concepts and properties are created correctly.

Interchange of concepts and properties with URI reserved characters in their names (';', '/', '?', ':', '@', '&', '=', '+', '$', ',') [*In65*]

When KAON receives concepts and properties with URI reserved characters in their names, the names of the concepts and properties are created correctly.

Corese. There are no results for Corese.

KAON. The concepts and properties are created correctly.

Protege. Information about range of property is not imported.

WebODE. The concepts and properties are created correctly.

Interchange of concepts and properties with XML delimiter characters in their names ('<', '>', '#', '%', '"') [*In66*]

When KAON receives concepts and properties with URI reserved characters in their names, the names of the concepts and properties are changed.

Corese. There are no results for Corese.

KAON. The concepts and properties are created correctly.

Protege. Information about range of property is not imported.

WebODE. KAON cannot receive concepts and properties with XML delimiter characters in their names from WebODE as the latter cannot model them.

C.2 Protégé results

by RAÚL GARCÍA-CASTRO

This section includes a detailed analysis of the results of evaluating the RDF(S) interoperability capabilities of Protégé according to the different groups of benchmarks that have been carried out.

C.2.1 Classes

Interchange of classes [*In01-02*]

Protégé receives correctly classes from the other tools.

Corese. Classes are created correctly.

KAON. Classes are created correctly.

Protege. Classes are created correctly.

WebODE. Classes are created correctly.

C.2.2 Metaclasses

Interchange of metaclasses [*In03-07*]

Protégé does not import instances of multiple resources, it only imports one of the *rdf:type* properties.

When Protégé receives from other tools classes that are instance of metaclasses, it imports the classes as instance of the metaclasses except when they are defined in the file as instance of other class (such as *rdfs:Class)([Corese:In03-07],[KAON:In03-07])*.

Corese. ♠ The metaclasses are created as classes and the *rdf:type* properties between the classes are lost because classes are defined in the file as instance of the metaclass and of *rdfs:Class*.

KAON. ♠ The metaclasses are created as classes and the *rdf:type* properties between the classes are lost because classes are defined in the file as instance of the metaclass and of *rdfs:Class*.

Protege. ♠ The metaclasses and the classes are created correctly except when a class is instance of multiple metaclasses *([Protégé:In04])*.

WebODE. Protégé cannot receive metaclasses from WebODE as the latter cannot model them.

C.2.3 Class hierarchies

Interchange of class hierarchies [*In08-10*]

Protégé receives correctly class hierarchies from other tools. If the resource object of *rdfs:subClassOf* property is not defined as a class in the file *([Corese:In09-10]),* the resource object is created as a class.

Corese. Class hierarchies are created correctly.

KAON. Class hierarchies are created correctly.

Protege. Class hierarchies are created correctly.

WebODE. Class hierarchies are created correctly.

Interchange of class hierarchies with cycles [*In11-12*]

Protégé cannot model cycles in class hierarchies. When Protégé finds a cycle in a class hierarchy it crashes, and does not import anything.

Corese. ♠Protégé crashes when receiving class hierarchies with cycles from Corese, and does not import anything.

KAON. Protégé cannot receive cycles in class hierarchies from KAON as the latter cannot model them.

Protege. Protégé cannot receive cycles in class hierarchies from Protégé as the latter cannot model them.

WebODE. Protégé cannot receive cycles in class hierarchies from WebODE as the latter cannot model them.

C.2.4 Classes related through object or datatype properties

Interchange of lasses with object properties [*In13-16*]

Protégé cannot model classes related through user defined object properties. When Protégé imports two classes related through an object property whose domain and range are some metaclass of the classes, it does not import either the property between the classes or the domain and the range of the property.

Corese. Neither the object properties between the classes nor the domain and the range of the property are created.

KAON. Protégé cannot receive classes related through object properties from KAON as the latter cannot model them.

Protege. Protégé cannot receive classes related through object properties from Protégé as the latter cannot model them.

WebODE. Protégé cannot receive classes related through object properties from WebODE as the latter cannot model them.

Interchange of classes with datatype properties [*In17-18*]

Protégé cannot model classes related through user defined datatype properties. When Protégé imports a class with a datatype property whose domain is some metaclass of the class, it does not import either the property between the class and the literal or the domain of the property.

Corese. ♠ When importing from Corese, Protégé crashes not importing anything as the files have a *rdf:datatype* attribute in the property.

KAON. Protégé cannot receive classes related through datatype properties from KAON as the latter cannot model them.

Protege. Protégé cannot receive classes related through datatype properties from Protégé as the latter cannot model them.

WebODE. Protégé cannot receive classes related through datatype properties from WebODE as the latter cannot model them.

C.2.5 Datatype properties

Interchange of datatype properties without domain and without range [*In19-20*]

Protégé imports correctly datatype properties without domain and without range.

Corese. ♠Protégé crashes and does not import anything as Corese includes the definition of *xsd:integer* as a *rdfs:Datatype* in the file.

KAON. ♠The datatype property is created with a range of *String*. Is this a mistake when modelling the benchmark?

Protege. Datatype properties without domain and without range are created correctly.

WebODE. Protégé cannot receive datatype properties without domain and without range from WebODE as the latter cannot model them.

Interchange of datatype properties with domain and without range [*In22-24*]

Protégé imports correctly datatype properties with domain and without range.

Corese. Datatype properties with domain and without range are created correctly.

KAON. ♠The datatype property is created with a range of *String*. Is this a mistake when modelling the benchmark?

Protege. Datatype properties with domain and without range are created correctly.

WebODE. Protégé cannot receive datatype properties with domain and without range from WebODE as the latter cannot model them.

Interchange of datatype properties without domain and with range [*In25-26*]

Protégé imports correctly datatype properties without domain and with range.

Corese. ♠Protégé crashes and does not import anything as Corese includes the definition of *xsd:integer* as a *rdfs:Datatype* in the file.

KAON. Datatype properties without domain and with range are created correctly.

Protege. Datatype properties without domain and with range are created correctly.

WebODE. Protégé cannot receive datatype properties without domain and with range from WebODE as the latter cannot model them.

Interchange of datatype properties with undefined resources as domain [*In21*]

When Protégé imports a datatype property that has as domain a resource that is not defined, it creates the undefined resource as a class.

Corese. ♠Protégé crashes and does not import anything as Corese includes the definition of *xsd:integer* as a *rdfs:Datatype* in the file.

KAON. Protégé cannot receive datatype properties with undefined resources as domain from KAON as the latter cannot model them.

Protege. Protégé cannot receive datatype properties with undefined resources as domain from Protégé as the latter cannot model them.

WebODE. Protégé cannot receive datatype properties with undefined resources as domain from WebODE as the latter cannot model them.

Interchange of datatype properties with multiple domains [*In23,28,30*]

When Protégé imports a datatype property that has multiple domains, it creates a slot with multiple domains. This is not the expected result, as in Protégé multiple domains in slots are considered as the union of all the domains and in RDF(S) multiple domains in properties are considered the intersection of all the domains.

Corese. ♠ Multiple domains in a property are created as multiple domains in slots.

KAON. ♠ Multiple domains in a property are created as multiple domains in slots.

Protege. ♠ Multiple domains in a property are created as multiple domains in slots.

WebODE. Protégé cannot receive datatype properties with multiple domains from WebODE as the latter cannot model them.

Interchange of datatype properties whose range is *String* [*In25-28*]

When Protégé imports a datatype property whose range is *String*, it creates the datatype property with a range of Protégé's datatype *String*.

Corese. ♠ Corese includes the definition of *xsd:integer* as a *rdfs:Datatype* in the file. When Protégé finds this datatype definition it crashes, and does not import anything.

KAON. *xsd:string* is created as a class. The range of the property is created as *xsd:string*.

Protege. The datatype property is created correctly.

WebODE. The datatype property is created correctly.

Interchange of datatype properties whose range is a XML Schema datatype [*In29-30*]

When Protégé imports a datatype property whose range is a XML Schema datatype, it creates the datatype as a class in the ontology with the *xsd* namespace.

Corese. ♠ Corese includes the definition of *xsd:integer* as a *rdfs:Datatype* in the file. When Protégé finds this datatype definition it crashes, and does not import anything.

KAON. ♠ *xsd:string* and *xsd:integer* are created as a class. The range of the property is the XML Schema datatype class.

The KAON exported ontology in ([*KAON:In30*]) models a datatype property with only one domain instead of a datatype property with multiple domains. Is this a mistake when modelling the benchmark?

Protege. Protégé cannot receive datatype properties with XML Schema datatypes from Protégé as the latter cannot model them.

WebODE. The XML Schema datatype is created as a class. The range of the property is the XML Schema datatype class.

C.2.6 Object properties

Interchange of object properties without domain and without range [*In31-32*]

When Protégé imports an object property without domain and without range, it creates the object property with a range of *Any*.

Corese. ♠ The object property is created with a range of a class. Is this a mistake when modelling the benchmark?

KAON. The object property is created with a range of *Any*.

Protege. The object property is created with a range of *Instance*.

WebODE. Protégé cannot receive object properties without domain and without range from WebODE as the latter cannot model them.

Interchange of object properties with domain and without range [*In34-36*]

Protégé imports properties with domain and without range as slots with domain and whose range is *Any*.

Corese. ♠ The object property is created with a range of *Any*.

When importing from Corese in *([Corese:In35])*, Protégé crashes not importing anything as the file has a *rdf:datatype* attribute in the property.

When importing from Corese in *([Corese:In36])*, Protégé creates the object properties with a class as range. Is this a mistake when modelling the benchmark?

KAON. The object property is created with a range of *Any*.

Protege. The object property is created with a range of *Instance*.

WebODE. Protégé cannot receive object properties with domain and without range from WebODE as the latter cannot model them.

Interchange of object properties without domain and with range [*In37-39*]

Protégé imports properties without domain and with range correctly.

Corese. ♠ The object property is created correctly.

When importing from Corese in *([Corese:In38])*, Protégé crashes not importing anything as the file has a *rdf:datatype* attribute in the property.

KAON. The object property is created correctly.

Protege. The object property is created correctly.

WebODE. Protégé cannot receive object properties without domain and with range from WebODE as the latter cannot model them.

Interchange of object properties with undefined resources as domain or range [*In33*]

When Protégé imports an object property that has as domain (or range) a resource that is not defined, it creates the undefined resource as a class and creates the property with a domain (or range) of the class.

Corese. The undefined resource is created as a class.

The object property is created with domain (or range) the class.

KAON. Protégé cannot receive object properties with undefined resources as domain or range from KAON as the latter cannot model them.

Protege. Protégé cannot receive object properties with undefined resources as domain or range from Protégé as the latter cannot model them.

WebODE. Protégé cannot receive object properties with undefined resources as domain or range from WebODE as the latter cannot model them.

Interchange of object properties with multiple domains [*In35,42-43*]

When Protégé imports a property that has multiple domains, it creates a slot with multiple domains. This is not the expected result, as in Protégé multiple domains in slots are considered as the union of all the domains and in RDF(S) multiple domains in properties are considered the intersection of all the domains.

Corese. ♠ When importing from Corese, Protégé crashes not importing anything as the file has a *rdf:datatype* attribute in the property.

KAON. ♠Protégé creates a slot with multiple domains. This is not the expected result, as in Protégé multiple domains in slots are considered as the union of all the domains and in RDF(S) multiple domains in properties are considered the intersection of all the domains.

Protege. Protégé creates a slot with multiple domains.

WebODE. Protégé cannot receive object properties with multiple domains from WebODE as the latter cannot model them.

Interchange of object properties with multiple ranges [*In38,41,43*]

When Protégé imports a property that has multiple ranges, it creates a slot with the range *Any*, as in Protégé multiple ranges in a property are considered as the union of all the ranges and in RDF(S) multiple ranges are considered as the intersection of all the ranges.

Corese. ♠ When importing from Corese, Protégé crashes not importing anything as the file has a *rdf:datatype* attribute in the property.

KAON. Protégé creates a slot with the range *Any*.

Protege. Protégé creates a slot with the range *Instance*.

WebODE. Protégé cannot receive object properties with multiple ranges from WebODE as the latter cannot model them.

Interchange of object properties with a domain and a range [*In40,44*]

When Protégé imports an object property whose domain is a class and whose range is another class, it creates the object property correctly, even if the domain and range classes are the same.

Corese. ♠ The object property is created correctly.

When importing from Corese in *([Corese:In44])*, Protégé crashes not importing anything as the file has a *rdf:datatype* attribute in the property.

KAON. The object property is created correctly.

Protege. The object property is created correctly.

WebODE. The object property is created correctly.

C.2.7 Instances

Interchange of instances [*In46,48*]

When Protégé imports an instance of a class, it creates the instance correctly.

Corese. ♠ The instance is created correctly.

When importing from Corese in *([Corese:In48])*, Protégé crashes not importing anything as the file has a *rdf:datatype* attribute in the property.

KAON. The instance is created correctly.

Protege. The instance is created correctly.

WebODE. The instance is created correctly.

Interchange of instances of undefined resources [*In45*]

When Protégé imports an instance of a resource that is not defined, it creates the undefined resource as a class.

Corese. The undefined resource is created as a class.

KAON. Protégé cannot receive instances of undefined resources from KAON as the latter cannot model them.

Protege. Protégé cannot receive instances of undefined resources from Protégé as the latter cannot model them.

WebODE. Protégé cannot receive instances of undefined resources from WebODE as the latter cannot model them.

Interchange of instances of multiple classes [*In47*]

When Protégé imports an instance of multiple classes, it imports the instance as instance of just one class. This is not the expected result, as it should import the instance as instance of all the classes.

Corese. ♠ The instance is imported as instance of just one class.

KAON. ♠ The instance is imported as instance of just one class.

Protege. ♠ The instance is imported as instance of just one class.

WebODE. Protégé cannot receive instances of multiple classes from WebODE as the latter cannot model them.

C.2.8 Instances and object properties

Interchange of instances related through object properties [*In51-57*]

When Protégé imports instances of classes related through defined object properties, it creates the instances and the properties correctly, even if the instances subject and object are instances of the same class, an instance has the same property with several other instances (either as subject or object), the instance is both the subject and object of the property, or the instance has the same property with several values.

Corese. ♠ When importing from Corese, Protégé crashes not importing anything as the files have a *rdf:datatype* attribute in the property.

KAON. The instances and the object properties are created correctly.

Protege. The instances and the object properties are created correctly.

WebODE. The instances and the object properties are created correctly.

Interchange of instances related through undefined object properties [*In49-50*]

When Protégé finds an object property in an instance (an instance-property-object triple), and this property is not defined, it does not consider the property to belong to the domain class. Therefore, it creates the slot without domain and with a range of *Any*, and the property in the instance is lost.

Corese. ♠ The object properties between the instances are not created. The object property is created without domain and with a range of *Any*.

When importing from Corese in *([Corese:In49]),* Protégé crashes not importing anything as the file has a *rdf:datatype* attribute in the property.

KAON. Protégé cannot receive instances related through undefined object properties from KAON as the latter cannot model them.

Protege. Protégé cannot receive instances related through undefined object properties from Protégé as the latter cannot model them.

WebODE. Protégé cannot receive instances related through undefined object properties from WebODE as the latter cannot model them.

C.2.9 Instances and datatype properties

Interchange of instances with by datatype properties [*In59-62*]

When Protégé imports instances of classes related through defined datatype properties, it creates the instances and the properties correctly, even if the instance has the same property with several values.

Corese. ♠ When importing from Corese, Protégé crashes not importing anything as the file has a *rdf:datatype* attribute in the property.

KAON. The instances, the properties and the values are created correctly, even if an instance has the same property with several values.

Protégé cannot receive instances related through datatype properties with a XML Schema datatype as range from KAON as the latter cannot model them.

Protege. The instances, the properties and the values are created correctly, even if an instance has the same property with several values.

Protégé cannot receive instances related through datatype properties with a XML Schema datatype as range from Protégé as the latter cannot model them.

WebODE. The instances, the properties and the values are created correctly, even if an instance has the same property with several values.

Protégé inserts *xsd:string* as a class. The values of property are not created.

Interchange of instances related through undefined datatype properties [*In58*]

When Protégé finds a datatype property in an instance (an instance-property-object triple), and this property is not defined, it does not consider the property to belong to the domain class. Therefore, it creates the slot without domain and with a range of *Any*, and the property in the instance is lost.

Corese. ♠ When importing from Corese, Protégé crashes not importing anything as the file has a *rdf:datatype* attribute in the property.

KAON. Protégé cannot receive instances related through undefined datatype properties from KAON as the latter cannot model them.

Protege. Protégé cannot receive instances related through undefined datatype properties from Protégé as the latter cannot model them.

WebODE. Protégé cannot receive instances related through undefined datatype properties from WebODE as the latter cannot model them.

C.2.10 URI character restrictions

Interchange of concepts and properties whose names start with a character that is not a letter or ’_’ [*In63*]

Protégé receives correctly concepts and properties whose names start with a character that is not a letter or ’_’.

Corese. There are no results for Corese.

KAON. The concepts and properties are created correctly.

Protege. ♠ The names of the concepts and properties are changed. The illegal character is replaced with a ’_’.

WebODE. The concepts and properties are created correctly.

Interchange of concepts and properties with spaces in their names [*In64*]

When Protégé receives concepts and properties with spaces in their names, the names of the concepts and properties are changed. The spaces are replaced with a '_'.

Corese. There are no results for Corese.

KAON. ♠ The names of the concepts and properties are changed. The space character is replaced with a'_'.

Protege. ♠ The names of the concepts and properties are changed. The space character is replaced with a '_'.

WebODE. ♠ The names of the concepts and properties are changed. The space character is replaced with a '_'.

Interchange of concepts and properties with URI reserved characters in their names (';', '/', '?', ':', '@', '&', '=', '+', '$', ',') [*In65*]

When Protégé receives concepts and properties with URI reserved characters in their names, the names of the concepts and properties are created correctly.

Corese. There are no results for Corese.

KAON. ♠ The names of the concepts and properties are changed. The reserved character is replaced with a '_'.

Protege. ♠ The names of the concepts and properties are changed. The reserved character is replaced with a '_' .

WebODE. The concepts and properties are created correctly.

Interchange of concepts and properties with XML delimiter characters in their names ('<', '>', '#', '%', '"')[*In66*]

When Protégé receives concepts and properties with URI reserved characters in their names, the names of the concepts and properties are changed.

Corese. There are no results for Corese.

KAON. ♠ The names of the concepts and properties are changed. The delimiter character is encoded.

Protege. ♠ The names of the concepts and properties are changed. The delimiter character is replaced with a ’_’. This causes ’concept<1’ and ’concept>1’ to be considered the same, that is, ’concept_1’.

WebODE. Protégé cannot receive concepts and properties with XML delimiter characters in their names from WebODE as the latter cannot model them.

C.3 WebODE results

by RAÚL GARCÍA-CASTRO

This section includes a detailed analysis of the results of evaluating the RDF(S) interoperability capabilities of WebODE according to the different groups of benchmarks that have been carried out.

C.3.1 Classes

Interchange of classes [*In01-02*]

WebODE receives correctly classes from the other tools. If the class has a *rdfs:label* property ([*Protégé:In01-02*], [*WebODE:In01-02*]), it inserts ”rdfs:label -> class_name” in the description of the class.

Corese. Classes are created correctly.

KAON. Classes are created correctly.

Protege. Classes are created correctly.

”rdfs:label -> class_name” is inserted in the description of the class.

rdfs:Resource is created as an imported term and the classes are created as subclass of *rdfs:Resource*, as Protégé exports classes as subclass of *rdfs:Resource*.

WebODE. Classes are created correctly.

”rdfs:label -> class_name” is inserted in the description of the class.

C.3.2 Metaclasses

Interchange of metaclasses [*In03-07*]

WebODE cannot model metaclasses. Therefore, when receiving metaclasses from other tools it imports the metaclasses as classes and loses the *rdf:type* properties between classes. If a metaclass is not defined as a class in the exported file ([*Corese:In04], [Protégé:In06-*

07]), the metaclass is not imported. If the class is not defined as a class in the exported file *([Protégé:In03-05]),* the class is imported as an instance.

Corese. ♠ The metaclasses are created as classes and the *rdf:type* properties between the classes are lost.

If one class is instance of several metaclasses ([*Corese:In04*]) the metaclasses are not created, as the metaclasses are not defined as classes in the exported file.

KAON. The metaclasses are created as classes and the *rdf:type* properties between the classes are lost.

Protege. ♠ The metaclasses are created as classes and the *rdf:type* properties between the classes are lost.

Classes are created as instances, as they are not defined as classes in the exported file.

rdfs:Class is created as an imported term and the classes are created as subclass of *rdfs:Class*, as Protégé exports metaclasses as subclass of *rdfs:Class*.

rdfs:subClassOf is created as an imported term.

If there is a cycle of *rdf:type* properties between classes *([Protégé:In06-07])* the metaclasses are not created, as they are not defined as classes in the exported file.

WebODE. WebODE cannot receive metaclasses from WebODE as the latter cannot model them.

C.3.3 Class hierarchies

Interchange of class hierarchies [*In08-10*]

WebODE receives correctly class hierarchies from other tools. If the object of a *rdfs:subClassOf* property is not defined as a class in the file *([Corese:In09-10])*, the class is not imported and consequently neither the *rdfs:subClassOf* property.

Corese. ♠ Linear class hierarchies are created correctly *([Corese:In08])*.

If class hierarchies are non-linear *([Corese:In09-10])* the parent classes and the subclass properties are not created, as the object of the *rdfs:subClassOf* properties is not defined as a class in the file.

KAON. Class hierarchies are created correctly.

Protege. Class hierarchies are created correctly.

WebODE. Class hierarchies are created correctly.

Interchange of class hierarchies with cycles [*In11-12*]

WebODE cannot model cycles in class hierarchies. When WebODE finds a cycle in a class hierarchy, it creates a class and an imported term with the same name as the object of the *rdfs:subClassOf* property that causes the cycle and creates the subclass with the imported term.

Corese. ♠ If class hierarchies form cycles, for each subclass property that causes the cycle a class and an imported term are created with the same name as the parent class, and the subclass is created with the imported term as parent.

KAON. WebODE cannot receive cycles in class hierarchies from KAON as the latter cannot model them.

Protege. WebODE cannot receive cycles in class hierarchies from Protégé as the latter cannot model them.

WebODE. WebODE cannot receive cycles in class hierarchies from WebODE as the latter cannot model them.

C.3.4 Classes related through object or datatype properties

Interchange of classes with object properties [*In13-16*]

WebODE cannot model classes related through user defined object properties. When WebODE imports two classes related through an object property whose domain and range are some metaclass of the classes, it does not import the property.

Corese. The object properties are not created.

KAON. WebODE cannot receive classes related through object properties from KAON as the latter cannot model them.

Protege. WebODE cannot receive classes related through object properties from Protégé as the latter cannot model them.

WebODE. WebODE cannot receive classes related through object properties from WebODE as the latter cannot model them.

Interchange of classes with datatype properties [*In17-18*]

WebODE cannot model classes related through user defined datatype properties. When WebODE imports a class with a datatype property whose domain is some metaclass of the class, it does not import the property.

Corese. The datatype properties are not created.

KAON. WebODE cannot receive classes related through datatype properties from KAON as the latter cannot model them.

Protege. WebODE cannot receive classes related through datatype properties from Protégé as the latter cannot model them.

WebODE. WebODE cannot receive classes related through datatype properties from WebODE as the latter cannot model them.

C.3.5 Datatype properties

Interchange of datatype properties without domain and without range [*In19-20*]

WebODE cannot model datatype properties without domain or range.

When WebODE imports a datatype property without domain and without range, it creates *rdfs:Resource* as an imported term and creates the datatype property as an object property with a domain and range of *rdfs:Resource*.

Corese. *rdfs:Resource* is created as an imported term.

The datatype property is created with a domain of *rdfs:Resource* and a range of *xsd:string*. Is this a mistake when modelling the benchmark?

KAON. *rdfs:Resource* is created as an imported term.

The datatype property is created with a domain of *rdfs:Resource* and a range of *String*. Is this a mistake when modelling the benchmark?

Protege. *rdfs:Resource* is created as an imported term.

The datatype property is created as an object property with a domain and range of *rdfs:Resource*.

WebODE. WebODE cannot receive datatype properties without domain and without range from WebODE as the latter cannot model them.

Interchange of datatype properties with domain and without range [*In22-24*]

WebODE cannot model datatype properties without domain or range.

When WebODE imports a datatype property with domain and without range, it creates *rdfs:Resource* as an imported term and creates the datatype property as an object property with a range of *rdfs:Resource*.

Corese. *rdfs:Resource* is created as an imported term.

The datatype property is created as an object property with a range of *rdfs:Resource*.

KAON. ♠The datatype property is created with a range of *String*. Is this a mistake when modelling the benchmark?

Protege. *rdfs:Resource* is created as an imported term.

The datatype property is created as an object property with a range of *rdfs:Resource*.

WebODE. WebODE cannot receive datatype properties with domain and without range from WebODE as the latter cannot model them.

Interchange of datatype properties without domain and with range [*In25-26*]

WebODE cannot model datatype properties without domain or range.

When WebODE imports a datatype property without domain and with range, it creates *rdfs:Resource* as an imported term and creates the datatype property with a domain of *rdfs:Resource*.

Corese. ♠Nothing is created, as WebODE crashes when receiving from Corese the String range defined in the file as a datatype without namespace.

KAON. *rdfs:Resource* is created as an imported term.

The datatype property is created with a domain of *rdfs:Resource*.

Protege. *rdfs:Resource* is created as an imported term.

The datatype property is created with a domain of *rdfs:Resource*.

WebODE. WebODE cannot receive datatype properties with domain and without range from WebODE as the latter cannot model them.

Interchange of datatype properties with undefined resources as domain [*In21*]

When WebODE imports a datatype property that has as domain a resource that is not defined, it creates the undefined resource as a class and creates the datatype property with a domain of the class.

Corese. The undefined resource is created as a class.

The datatype property is created with domain the class and range *xsd:string*.

KAON. WebODE cannot receive datatype properties with undefined resources as domain from KAON as the latter cannot model them.

Protege. WebODE cannot receive datatype properties with undefined resources as domain from Protégé as the latter cannot model them.

WebODE. WebODE cannot receive datatype properties with undefined resources as domain from WebODE as the latter cannot model them.

Interchange of datatype properties with multiple domains [*In23,28,30*]

When WebODE imports a datatype property that has multiple domains, it creates an anonymous concept as the domain of the datatype property and as subclass of one of the domain classes.

Corese. ♠ An anonymous concept is created as subclass of one of the domain classes.

The datatype property is created with the anonymous concept as domain.

KAON. ♠ An anonymous concept is created as subclass of one of the domain classes.

The datatype property is created with the anonymous concept as domain.

Protege. ♠ An anonymous concept is created as subclass of one of the domain classes.

The datatype property is created with the anonymous concept as domain.

WebODE. WebODE cannot receive datatype properties with multiple domains from WebODE as the latter cannot model them.

Interchange of datatype properties whose range is *String [In25-28]*

When WebODE imports datatype properties whose range is *String*, it creates the datatype property with a range of WebODE's *String* datatype.

Corese. ♠Nothing is created, as WebODE crashes when receiving from Corese the *String* range defined in the file as a datatype without namespace.

KAON. *xsd:string* is created as an imported term.

string is created as an imported term.

The datatype property is created as both an object property and a datatype property with a range of *xsd:string*, as the range is defined as a class in the file.

Protege. The datatype property is created correctly.

WebODE. The datatype property is created correctly.

Interchange of datatype properties with XML Schema datatype as range [*In29-30*]

WebODE imports correctly a datatype property that has an XML Schema datatype as range.

Corese. The datatype property with XML Schema datatype as range is created correctly.

KAON. ♠ The XML Schema datatype is created as an imported term.

string and *integer* are created as imported terms.

The datatype property is created as both an object property and a datatype property with a range of the XML Schema datatype, as the range is defined as a class in the file.

The KAON exported ontology in *([KAON:In30])* models a datatype property with only one domain instead of a datatype property with multiple domains. Is this a mistake when modelling the benchmark?

Protege. WebODE cannot receive datatype properties with XML Schema datatype as range from Protégé as the latter cannot model them.

WebODE. The datatype property with XML Schema datatype as range is created correctly.

C.3.6 Object properties

Interchange of object properties without domain and without range [*In31-32]*

WebODE cannot model object properties without domain or range.

When WebODE imports an object property without domain and without range, it creates *rdfs:Resource* as an imported term and creates the object property with a domain and range of *rdfs:Resource*.

Corese. ♠ *rdfs:Resource* is created as an imported term.

The object property is created with a domain of *rdfs:Resource* and a range of a class. Is this a mistake when modelling the benchmark?

KAON. *rdfs:Resource* is created as an imported term.

The object property is created with a domain and range of *rdfs:Resource*.

Protege. *rdfs:Resource* is created as an imported term.

The object property is created with a domain and range of *rdfs:Resource*.

WebODE. WebODE cannot receive object properties without domain and without range from WebODE as the latter cannot model them.

Interchange of object properties with domain and without range [*In34-36]*

WebODE cannot model object properties without domain or range.

When WebODE imports an object property with domain and without range, it creates *rdfs:Resource* as an imported term and creates the object property with a range of *rdfs:Resource*.

Corese. ♠ *rdfs:Resource* is created as an imported term.

The object property is created with a range of *rdfs:Resource*.

When importing from Corese in *([Corese:In36])*, WebODE creates the object properties with a concept as range. Is this a mistake when modelling the benchmark?

KAON. *rdfs:Resource* is created as an imported term.

The object property is created with a range of *rdfs:Resource*.

Protege. *rdfs:Resource* is created as an imported term. The object property is created with a range of *rdfs:Resource*.

WebODE. WebODE cannot receive object properties with domain and without range from WebODE as the latter cannot model them.

Interchange of object properties without domain and with range [*In37-39*]

WebODE cannot model object properties without domain or range.

When WebODE imports a object property without domain and with range, it creates *rdfs:Resource* as an imported term and creates the object property with a domain of *rdfs:Resource*.

Corese. *rdfs:Resource* is created as an imported term.

The object property is created with a domain of *rdfs:Resource*.

KAON. *rdfs:Resource* is created as an imported term.

The object property is created with a domain of *rdfs:Resource*.

Protege. *rdfs:Resource* is created as an imported term.

The object property is created with a domain of *rdfs:Resource*.

WebODE. WebODE cannot receive object properties without domain and with range from WebODE as the latter cannot model them.

Interchange of object properties with undefined resources as domain or range [*In33*]

When WebODE imports an object property that has as domain (or range) a resource that is not defined, it creates the undefined resource as a class and creates the object property with a domain (or range) of the class.

Corese. The undefined resource is created as a class.

The object property is created with domain (or range) the class.

KAON. WebODE cannot receive object properties with undefined resources as domain or range from KAON as the latter cannot model them.

Protege. WebODE cannot receive object properties with undefined resources as domain or range from Protégé as the latter cannot model them.

WebODE. WebODE cannot receive object properties with undefined resources as domain or range from WebODE as the latter cannot model them.

Interchange of object properties with multiple domains [*In35,42-43*]

When WebODE imports an object property that has multiple domains, it creates an anonymous concept as the domain of the object property and as subclass of one of the domain classes.

Corese. ♠ An anonymous concept is created as subclass of one of the domain classes. The object property is created with the anonymous concept as domain.

KAON. ♠ An anonymous concept is created as subclass of one of the domain classes. The object property is created with the anonymous concept as domain.

Protege. ♠ An anonymous concept is created as subclass of one of the domain classes. The object property is created with the anonymous concept as domain.

WebODE. WebODE cannot receive object properties with multiple domains from WebODE as the latter cannot model them.

Interchange of object properties with multiple ranges [*In38,41,43*]

When WebODE imports an object property that has multiple ranges, it creates an anonymous concept as the range of the object property and as subclass of one of the range classes.

Corese. ♠ An anonymous concept is created as subclass of one of the range classes.

The object property is created with the anonymous concept as range.

KAON. ♠ An anonymous concept is created as subclass of one of the range classes.

The object property is created with the anonymous concept as range.

Protege. *rdfs:Resource* is created as an imported term.

The object property is created with *rdfs:Resource* as range.

WebODE. WebODE cannot receive object properties with multiple ranges from WebODE as the latter cannot model them.

Interchange of object properties with a domain and a range [*In40,44*]

When WebODE imports an object property with a domain a class and a range another class, it creates the object property correctly, even if the domain and range classes are the same.

Corese. The object property is created correctly.

KAON. The object property is created correctly.

Protege. The object property is created correctly.

WebODE. The object property is created correctly.

C.3.7 Instances

Interchange of instances [*In46,48*]

When WebODE imports an instance of a class, it creates the instance correctly. If the instance has a *rdfs:label* property *([Protégé:In46,48],[WebODE:In46,48]),* it inserts ”rdfs:label -> instance_name” in the description of the instance.

Corese. The instance is created correctly.

KAON. The instance is created correctly.

Protege. The instance is created correctly.

”rdfs:label -> instance_name” is inserted in the description of the instance.

WebODE. The instance is created correctly.

”rdfs:label -> instance_name” is inserted in the description of the instance.

Interchange of instances of undefined resources [*In45*]

When WebODE imports an instance of a resource that is not defined, it does not import the instance.

Corese. The instance and the undefined resource are not created.

KAON. WebODE cannot receive instances of undefined resources from KAON as the latter cannot model them.

Protege. WebODE cannot receive instances of undefined resources from Protégé as the latter cannot model them.

WebODE. WebODE cannot receive instances of undefined resources from WebODE as the latter cannot model them.

Interchange of instances of multiple classes [*In47]*

WebODE does not support instances of multiple classes.

When WebODE imports an instance of multiple classes, it imports the instance as instance of just one class.

Corese. The instance is imported as instance of just one class.

KAON. The instance is imported as instance of just one class.

Protege. The instance is imported as instance of just one class.

WebODE. WebODE cannot receive instances of multiple classes from WebODE as the latter cannot model them.

C.3.8 Instances and object properties

Interchange of instances related through object properties [*In51-57*]

When WebODE imports instances of classes related through defined object properties, it creates the instances and the properties correctly, even if the instances subject and object are instances of the same class, an instance has the same property with several other instances (either as subject or object), or the instance is both the subject and object of the property.

Corese. The instances and the object properties are created correctly.

KAON. The instances and the object properties are created correctly.

Protege. The instances and the object properties are created correctly.

WebODE. The instances and the object properties are created correctly.

Interchange of instances related through undefined object properties [*In49-50*]

When WebODE imports instances related through undefined object properties, it does not import the object properties between the instances.

Corese. The object properties are not created.

KAON. WebODE cannot receive instances related through undefined object properties from KAON as the latter cannot model them.

Protege. WebODE cannot receive instances related through undefined object properties from Protégé as the latter cannot model them.

WebODE. WebODE cannot receive instances related through undefined object properties from WebODE as the latter cannot model them.

C.3.9 Instances and datatype properties

Interchange of instances related through datatype properties [*In59-62*]

When WebODE imports instances of classes related through defined datatype properties, it creates the instances, the properties and the values correctly, even if an instance has the same property with several values or the datatype has an XML Schema datatype as range.

Corese. The instances, the properties and the values are created correctly, even if an instance has the same property with several values or the datatype has an XML Schema datatype as range.

KAON. The instances, the properties and the values are created correctly, even if an instance has the same property with several values.

WebODE cannot receive instances related through datatype properties with a XML Schema datatype as range from KAON as the latter cannot model them.

Protege. The instances, the properties and the values are created correctly, even if an instance has the same property with several values.

WebODE cannot receive instances related through datatype properties with a XML Schema datatype as range from Protégé as the latter cannot model them.

WebODE. The instances, the properties and the values are created correctly, even if an instance has the same property with several values or the datatype has an XML Schema datatype as range.

Interchange of instances related through undefined datatype properties [*In58*]

When WebODE imports instances related through undefined datatype properties, it does not import the datatype properties between the instances.

Corese. ♠ The datatype properties are not created.

The instance is not created as the class is undefined in the exported file. Is this a mistake when modelling the benchmark?

KAON. WebODE cannot receive instances related through undefined datatype properties from KAON as the latter cannot model them.

Protege. WebODE cannot receive instances related through undefined datatype properties from Protégé as the latter cannot model them.

WebODE. WebODE cannot receive instances related through undefined datatype properties from WebODE as the latter cannot model them.

C.3.10 URI character restrictions

Interchange of concepts and properties whose names start with a character that is not a letter or ’_’ [*In63*]

WebODE receives correctly concepts and properties whose names start with a character that is not a letter or ’_’.

Corese. There are no results for Corese.

KAON. The concepts and properties are created correctly.

Protege. ♠ The names of the concepts and properties are changed. The illegal character is replaced with a ’_’.

WebODE. The concepts and properties are created correctly.

Interchange of concepts and properties with spaces in their names [*In64*]

When WebODE receives concepts and properties with spaces in their names, the names of the concepts and properties are changed. The spaces are replaced with a ’_’.

Corese. There are no results for Corese.

KAON. ♠ The names of the concepts and properties are changed. The space character is replaced with a ’_’.

Protege. ♠ The names of the concepts and properties are changed. The space character is replaced with a ’_’.

WebODE. ♠ The names of the concepts and properties are changed. The space character is replaced with a ’_’.

Interchange of concepts and properties with URI reserved characters in their names (’;’, ’/’, ’?’, ’:’, ’@’, ’&’, ’=’, ’+’, ’$’, ’,’) [*In65*]

When WebODE receives concepts and properties with URI reserved characters in their names, the names of the concepts and properties are created correctly.

Corese. There are no results for Corese.

KAON. ♠ The names of the concepts and properties are changed. The reserved character is replaced with a ’_’.

Protege. ♠ The names of the concepts and properties are changed. The reserved character is replaced with a ’_’.

WebODE. The concepts and properties are created correctly.

Interchange of concepts and properties with XML delimiter characters in their names (’<’, ’>’, ’#’, ’%’, ’”’) [*In66*]

When WebODE receives concepts and properties with URI reserved characters in their names, the names of the concepts and properties are changed.

Corese. There are no results for Corese.

KAON. ♠ The names of the concepts and properties are changed. The delimiter character is encoded.

Protege. ♠ The names of the concepts and properties are changed. The delimiter character is replaced with a '_'. This causes 'concept<1' and 'concept>1' to be considered the same, that is, 'concept_1'.

WebODE. WebODE cannot receive concepts and properties with XML delimiter characters in their names from WebODE as the latter cannot model them.