Chapter 2
Technical Background

NIKOLAOS KONSTANTINOU
DIMITRIOS-EMMANUEL SPANOS

Outline

Introduction

RDF and RDF Schema

Description Logics

Querying RDF data with SPARQL

Mapping relational data with R2RML

Other technologies

Ontologies

Datasets

Introduction

Linked Data

- A set of technologies
- Focused on the web
- Use the Web as a storage and communication layer
- Provide meaning to web content

Semantics

- Added value when part of a larger context
  - Data modeled as a graph

Technologies fundamental to the web

HTTP – HyperText Transfer Protocol

An application protocol for the management and transfer of hypermedia documents in decentralized information systems

Defines a number of request types and the expected actions that a server should carry out when receiving such requests

Serves as a mechanism to

- Serialize resources as a stream of bytes
  - E.g. a photo of a person
- Retrieve descriptions about resources that cannot be sent over network
  - E.g. the person itself

URI – Uniform Resource Identifier

URL

- Identifies a document location
- Address of a document or other entity that can be found online

URI

- Provides a more generic means to identify anything that exists in the world

IRI

- Internationalized URI

IRI ⊇ URI ⊇ URL

HTML – HyperText Markup Language

A markup language for the composition and presentation of various types of content into web pages

- E.g. text, images, multimedia

Documents delivered through HTTP are usually expressed in HTML

- HTML5
  - Current recommendation
  - Additional markup tags
  - Extends support for multimedia and mathematical content

XML – eXtensible Markup Language

Allows for strict definition of the structure of information

- Markup tags

The RDF model also follows an XML syntax

Outline

Introduction

RDF and RDF Schema

Description Logics

Querying RDF data with SPARQL

Mapping relational data with R2RML

Other technologies

Ontologies

Datasets

Modeling Data Using RDF Graphs

Ontologies in the Semantic Web

- Model a system's knowledge
- Based on RDF
  - Model the perception of the world as a graph
  - OWL builds on top of RDF

RDF

- A data model for the Web
- A framework that allows representing knowledge
- Model knowledge as a directed and labeled graph

RDF (1)

The cornerstone of the Semantic Web

A common representation of web resources

First publication in 1999

Can represent data from other data models

- Makes it easy to integrate data from multiple heterogeneous sources

Main idea:

- Model every resource with respect to its relations (properties) to other web resources

RDF (2)

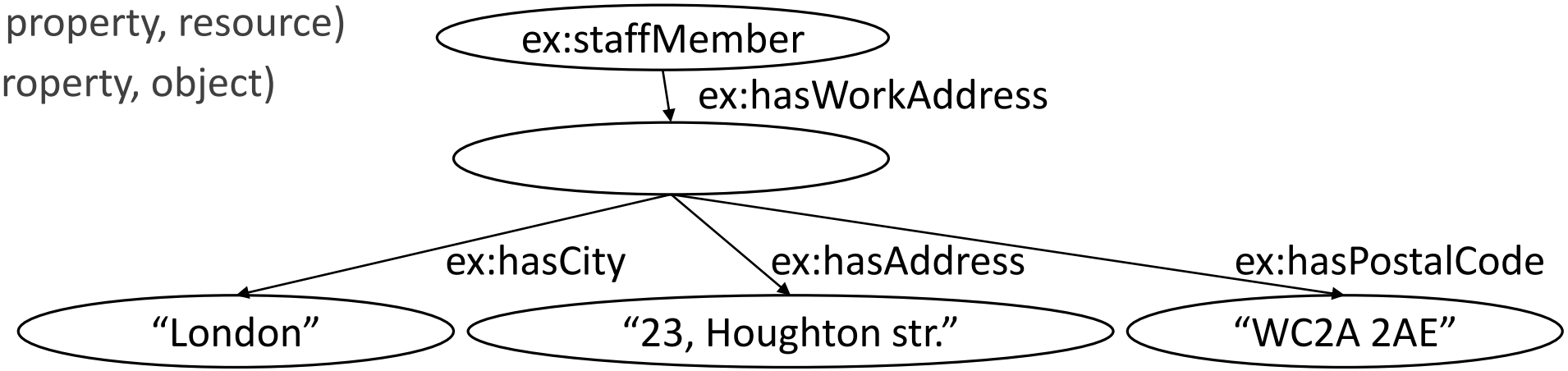
Relations form triples

- First term: subject
- Second term: property
- Third term: object

RDF statements contain triples, in the form:

(resource, property, resource)

or (subject, property, object)



Namespaces (1)

Initially designed to prevent confusion in XML names

Allow document elements to be uniquely identified

Declared in the beginning of XML documents

```
xmlns:prefix="location".
```

Namespaces (2)

Prefix	Namespace URI	Description
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	The built-in RDF vocabulary
rdfs	http://www.w3.org/2000/01/rdf-schema	RDFS puts an order to RDF
owl	http://www.w3.org/2002/07/owl	OWL terms
xsd	http://www.w3.org/2001/XMLSchema#	The RDF-compatible XML Schema datatypes
dc	http://purl.org/dc/elements/1.1/	The Dublin Core standard for digital object description
foaf	http://xmlns.com/foaf/0.1	The FOAF network
skos	http://www.w3.org/2004/02/skos/core#	Simple Knowledge Organization System
void	http://rdfs.org/ns/void#	Vocabulary of Interlinked Datasets
sioc	http://rdfs.org/sioc/ns#	Semantically-Interlinked Online Communities
cc	http://creativecommons.org/ns	Creative commons helps expressing licensing information
rdfa	http://www.w3.org/ns/rdfa	RDFa

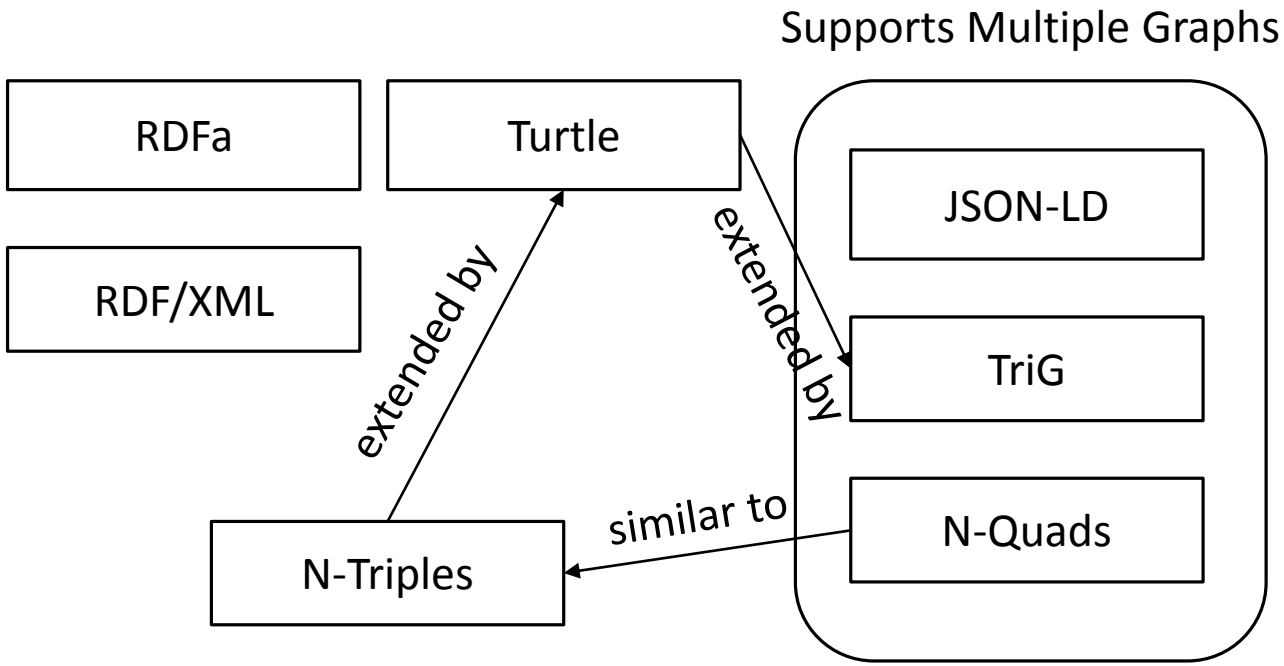
RDF Serialization

RDF is mainly destined for machine consumption

Several ways to express RDF graphs in machine-readable (serialization) formats

- N-Triples
- Turtle
- N-Quads
- TriG
- RDF/XML
- JSON-LD
- RDFa

Turtle family



N-Triples Serialization

```
<http://www.example.org/bradPitt> <http://www.example.org/isFatherOf>
<http://www.example.org/maddoxJoliePitt>.
<http://www.example.org/bradPitt> <http://xmlns.com/foaf/0.1/name> "Brad Pitt".
<http://www.example.org/bradPitt> <http://xmlns.com/foaf/0.1/based_near> :_x.
:_x <http://www.w3.org/2003/01/geo/wgs84_pos#lat> "34.1000".
:_x <http://www.w3.org/2003/01/geo/wgs84_pos#long> "118.3333".
```

Turtle Serialization

```
PREFIX ex: <http://www.example.org/>.
PREFIX foaf: <http://xmlns.com/foaf/0.1/>.
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>.

ex:bradPitt ex:isFatherOf ex:maddoxJoliePitt;
            foaf:name "Brad Pitt";
            foaf:based_near :_x.
:_x geo:lat "34.1000";
    geo:long "118.3333".
```

TriG Serialization

```
PREFIX ex: <http://www.example.org/>.
PREFIX foaf: <http://xmlns.com/foaf/0.1/>.
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>.

GRAPH <http://www.example.org/graphs/brad> {
ex:bradPitt ex:isFatherOf ex:maddoxJoliePitt;
            foaf:name "Brad Pitt";
            foaf:based_near :_x.
:_x geo:lat "34.1000";
    geo:long "118.3333".
}
```

XML/RDF Serialization

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:ex="http://www.example.org/"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
   <rdf:Description rdf:about=" http://www.example.org/bradPitt">
      <ex:isFatherOf rdf:resource=" http://www.example.org/maddoxJoliePitt"/>
      <foaf:name>Brad Pitt</foaf:name>
      <foaf:based_near rdf:nodeID="A0"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID="A0">
      <geo:lat>34.1000</geo:lat>
      <geo:long>118.3333</geo:long>
   </rdf:Description>
</rdf:RDF>
```

JSON-LD Serialization

```
{
 "@context":{
   "foaf": "http://xmlns.com/foaf/0.1/",
   "child": {
      "@id": "http://www.example.org/isFatherOf",
      "@type": "@id"
   },
   "name": "foaf:name",
   "location": "foaf:based_near",
   "geo": "http://www.w3.org/2003/01/geo/wgs84_pos#",
   "lat": "geo:lat",
   "long": "geo:long"
 },
 "@id": "http://www.example.org/bradPitt",
 "child": "http://www.example.org/maddoxJoliePitt",
 "name": "Brad Pitt",
 "location": {
   "lat": "34.1000",
   "long": "118.3333"
 }
}
```

RDFa Serialization

```
<html>
<head>
…
</head>
<body vocab="http://xmlns.com/foaf/0.1/">
   <div resource="http://www.example.org/bradPitt">
      <p>Famous American actor <span property="name">Brad Pitt</span>
eldest son is
 <a property="http://www.example.org/isFatheOf"
href="http://www.example.org/maddoxJoliePitt">Maddox Jolie-Pitt</a>.
      </p>
   </div>
</body>
</html>
```

The RDF Schema (1)

RDF

- A graph model
- Provides basic constructs for defining a graph structure

RDFS

- A semantic extension of RDF
- Provides mechanisms for assigning meaning to the RDF nodes and edges
- RDF Schema *is not* to RDF what XML Schema is to XML!

The RDF Schema (2)

Classes

- A definition of groups of resources
- Members of a class are called *instances* of this class

Class hierarchy

Property hierarchy

Property domain and range

RDFS specification also refers to the RDF namespace

Prefix	Namespace
rdfs	http://www.w3.org/2000/01/rdf-schema#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#

The RDF Schema (3)

A Class named Actor

```
ex:Actor rdf:type rdfs:Class.
```

- Where ex: http://www.example.org/

Class membership example

```
ex:bradPitt rdf:type ex:Actor.
```

Containment relationship

```
ex:Actor rdfs:subClassOf ex:MovieStaff.
```

Using reasoning, we can infer

```
ex:bradPitt rdf:type ex:MovieStaff.
```

- In practice, we could store inferred triples

The RDF Schema (4)

No restrictions on the form of the class hierarchy that can be defined in an ontology

- No strict tree-like hierarchy as regular taxonomies
- More complex graph-like structures are allowed
  - Classes may have more than one superclass

Allows hierarchies of properties to be defined

- Just like class hierarchies
- Introduced via the rdfs:subPropertyOf property

```
ex:participatesIn rdf:type rdf:Property.
ex:starsIn rdf:type rdf:Property.
ex:starsIn rdfs:subPropertyOf ex:participatesIn.
```

The RDF Schema (5)

```
ex:bradPitt ex:starsIn ex:worldWarZ.
```

- Inferred triple:

```
ex:bradPitt ex:participatesIn ex:worldWarZ.
```

Define the classes to which RDF properties can be applied

- rdfs:domain and rdfs:range

```
ex:starsIn rdfs:domain ex:Actor.
```

Then, the assertion

```
ex:georgeClooney ex:starsIn ex:idesOfMarch.
```

Entails

```
ex:georgeClooney rdf:type ex:Actor.
```

The RDF Schema (6)

Similarly

```
ex:starsIn rdfs:range ex:Movie.
```

Produces

```
ex:starsIn rdfs:range ex:Movie.
ex:worldWarZ rdf:type ex:Movie.
```

Domain and range axioms

- Are not constraints that data need to follow
- Function as rules that lead to the production of new triples and knowledge

The RDF Schema (7)

Example

```
ex:georgeClooney ex:starsIn ex:bradPitt.
```

- Infers:

```
ex:bradPitt rdf:type ex:Movie.
```

An individual entity can be a member of both ex:Actor and ex:Movie classes

Such restrictions are met in more expressive ontology languages, such as OWL

Reification (1)

Statements about other RDF statements

Ability to treat an RDF statement as an RDF resource

- Make assertions about that statement

```
<http://www.example.org/person/1> foaf:name "Brad Pitt " .
```

Becomes:

```
<http://www.example.org/statement/5> a rdf:Statement;
        rdf:subject <http://www.example.org/person/1>;
        rdf:predicate foaf:name;
        rdf:object "Brad Pitt".
```

Reification (2)

Useful when referring to an RDF triple in order to

- Describe properties that apply to it
  - e.g. provenance or trust
- E.g. assign a trust level of 0.8

```
<http://www.example.org/statement/5> ex:hasTrust "0.8"^^xsd:float.
```

Classes (1)

rdfs:Resource

- All things described by RDF are instances of the class rdfs:Resource
- All classes are subclasses of this class, which is an instance of rdfs:Class

rdfs:Literal

- The class of all literals
- An instance of rdfs:Class
- Literals are represented as strings but they can be of any XSD datatype

Classes (2)

rdfs:langString

- The class of language-tagged string values
- It is a subclass of rdfs:Literal and an instance of rdfs:Datatype
- Example: "foo"@en

rdfs:Class

- The class of all classes, i.e. the class of all resources that are RDF classes

Classes (3)

rdfs:Datatype

- The class of all the data types
- An instance but also a subclass of rdfs:Class
- Every instance of rdfs:Datatype is also a subclass of rdfs:Literal

rdf:HTML

- The class of HTML literal values
- An instance of rdfs:Datatype
- A subclass of rdfs:Literal

Classes (4)

rdf:XMLLiteral

- The class of XML literal values
- An instance of rdfs:Datatype
- A subclass of rdfs:Literal

rdf:Property

- The class of all RDF properties

Properties (1)

rdfs:domain

- Declares the domain of a property *P*
- The class of all the resources that can appear as *S* in a triple (*S*, *P*, *O*)

rdfs:range

- Declares the range of a property *P*
- The class of all the resources that can appear as *O* in a triple (*S*, *P*, *O*)

Properties (2)

rdf:type

- A property that is used to state that a resource is an instance of a class

rdfs:label

- A property that provides a human-readable version of a resource's name

Properties (3)

rdfs:comment

- A property that provides a human-readable description of a resource

rdfs:subClassOf

- Corresponds a class to one of its superclasses
- A class can have more than one superclasses

rdfs:subPropertyOf

- Corresponds a property to one of its superproperties
- A property can have more than one superproperties

Container Classes and Properties (1)

rdfs:Container

- Superclass of all classes that can contain instances such as rdf:Bag, rdf:Seq and rdf:Alt

rdfs:member

- Superproperty to all the properties that declare that a resource belongs to a class that can contain instances (container)

rdfs:ContainerMembershipProperty

- A property that is used in declaring that a resource is a member of a container
- Every instance is a subproperty of rdfs:member

Container Classes and Properties (2)

rdf:Bag

- The class of unordered containers
- A subclass of rdfs:Container

rdf:Seq

- The class of ordered containers
- A subclass of rdfs:Container

rdf:Alt

- The class of containers of alternatives
- A subclass of rdfs:Container

Collections (1)

rdf:List

- The class of RDF Lists
- An instance of rdfs:Class
- Can be used to build descriptions of lists and other list-like structures

rdf:nil

- An instance of rdf:List that is an empty rdf:List

Collections (2)

rdf:first

- The first item in the subject RDF list
- An instance of rdf:Property

rdf:rest

- The rest of the subject RDF list after the first item
- An instance of rdf:Property

rdf:_1, rdf:_2, rdf:_3, etc.

- A sub-property of rdfs:member
- An instance of the class rdfs:ContainerMembershipProperty

Reification (1)

rdf:Statement

- The class of RDF statements
- An instance of rdfs:Class

rdf:subject

- The subject of the subject RDF statement
- An instance of rdf:Property

Reification (2)

rdf:predicate

- The predicate of the subject RDF statement
- An instance of rdf:Property

rdf:object

- The object of the subject RDF statement
- An instance of rdf:Property

Utility Properties

rdf:value

- Idiomatic property used for describing structured values
- An instance of rdf:Property

rdfs:seeAlso

- A property that indicates a resource that might provide additional information about the subject resource

rdfs:isDefinedBy

- A property that indicates a resource defining the subject resource
- The defining resource may be an RDF vocabulary in which the subject resource is described

Outline

Introduction

RDF and RDF Schema

Description Logics

Querying RDF data with SPARQL

Mapping relational data with R2RML

Other technologies

Ontologies

Datasets

Ontologies Based on Description Logics

Language roots are in Description Logics (DL)

DAML, OIL, and DAML + OIL

OWL and OWL 2 latest results in this direction

Formal semantics

Syntactically compatible to the RDF serializations

- Ontologies in OWL can be queried in the same approach as in RDF graphs

Description Logics (1)

Offers the language for the description and manipulation of independent individuals, roles, and concepts

There can be many DL languages

- Describe the world using formulas

Formulas constructed using

- Sets of concepts, roles, individuals
- Constructors, e.g. intersection $\wedge$, union $\vee$, exists $\exists$, for each $\forall$

Description Logics (2)

Variables in DL can represent arbitrary world objects

E.g. a DL formula can declare that a number *x*, greater than zero exists:

- $\exists x{:}greaterThan(x; 0)$

Adding more constructors to the basic DL language increases expressiveness

- Possible to describe more complex concepts

Description Logics (3)

E.g. concept conjunction ($C \cup D$)

- *Parent* = *Father* ∪ *Mother*

Expressiveness used in describing the world varies according to the subset of the language that is used

Differentiation affects

- World description capabilities
- Behavior and performance of processing algorithms

The Web Ontology Language (1)

OWL language is directly related to DL

- Different “flavors” correspond to different DL language subsets

Successor of DAML+OIL

- Creation of the OWL language officially begins with the initiation of the DAML project
- DAML, combined to OIL led to the creation of DAML + OIL
  - An extension of RDFS
- OWL is the successor of DAML + OIL

W3C recommendation, currently in version 2

The Web Ontology Language (2)

Designed in order to allow applications to process the information content itself instead of simply presenting the information

The goal is to provide a schema that will be compatible both to the Semantic Web and the World Wide Web architecture

Makes information more machine- and human- processable

The Web Ontology Language (3)

First version comprised three flavors

- Lite, DL, Full

OWL Lite

- Designed keeping in mind that it had to resemble RDFS

OWL DL

- Guaranteed that all reasoning procedures are finite and return a result

OWL Full

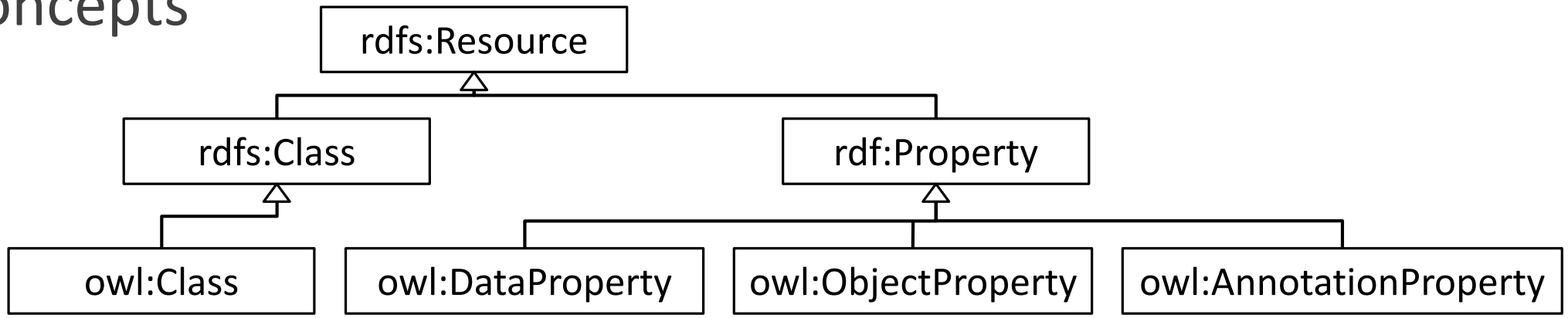
- The whole wealth and expressiveness of the language
- Reasoning is not guaranteed to be finite
  - Even for small declaration sets

The Web Ontology Language (4)

An OWL ontology may also comprise declarations from RDF and RDFS

- E.g. rdfs:subClassOf, rdfs:range, rdf:resource

OWL uses them and relies on them in order to model its concepts



OWL 2

Very similar overall structure to the first version of OWL

Backwards compatible

- Every OWL 1 ontology is also an OWL 2 ontology

OWL 2 Additional Features (1)

Additional property and qualified cardinality constructors

- Minimum, maximum or exact qualified cardinality restrictions, object and data properties
  - E.g. ObjectMinCardinality, DataMaxCardinality

Property chains

- Properties as a composition of other properties
  - E.g. ObjectPropertyChain in SubObjectPropertyOf

OWL 2 Additional Features (2)

Extended datatype support

- Support the definition of subsets of datatypes (e.g. integers and strings)
  - E.g. state that every person has an age, which is of type integer, and restrict the range of that datatype value

Simple metamodeling

- Relaxed separation between the names of, e.g. classes and individuals
  - Allow different uses of the same term

OWL 2 Additional Features (3)

Extended annotations

- Allow annotations of axioms
  - E.g. who asserted an axiom or when

Extra syntactic sugar

- Easier to write common patterns
  - Without changing language expressiveness, semantics, or complexity

OWL 2 Profiles (1)

OWL 2 EL

- Polynomial time algorithms for all standard reasoning tasks
- Suitable for very large ontologies
- Trades expressive power for performance guarantees

OWL 2 Profiles (2)

OWL 2 QL

- Conjunctive queries are answered in LOGSPACE using standard relational database technology
- Suitable for
  - Relatively lightweight ontologies with large numbers of individuals
  - Useful or necessary to access the data via relational queries (e.g. SQL)

OWL 2 Profiles (3)

OWL 2 RL

- Enables polynomial time reasoning algorithms using rule-extended database technologies operating directly on RDF triples
- Suitable for
  - Relatively lightweight ontologies with large numbers of individuals
  - Necessary to operate directly on data in the form of RDF triples

Manchester OWL syntax

A user-friendly syntax for OWL 2 descriptions

```
Class: VegetarianPizza
EquivalentTo:
Pizza and
not (hasTopping some FishTopping) and
not (hasTopping some MeatTopping)

DisjointWith:
NonVegetarianPizza
```

Outline

Introduction

RDF and RDF Schema

Description Logics

Querying RDF data with SPARQL

Mapping relational data with R2RML

Other technologies

Ontologies

Datasets

Querying the Semantic Web with SPARQL

SPARQL

- Is for RDF what SQL is for relational databases

SPARQL family of recommendations

- SPARQL Update
- SPARQL 1.1 Protocol
- Graph Store HTTP Protocol
- SPARQL 1.1 entailment regimes
- Specifications about the serialization of query results
  - In JSON, CSV and TSV, and XML

SELECT Queries (1)

Return a set of bindings of variables to RDF terms

Binding

- A mapping from a variable to a URI or an RDF literal

Triple pattern

- Resembles an RDF triple
- May also contain variables in one or more positions

SELECT Queries (2)

An RDF triple *matches* a triple pattern

- There is an appropriate substitution of variables with RDF terms that makes the triple pattern and the RDF triple equivalent
- Example:

```
ex:bradPitt rdf:type ex:Actor.
```

matches

```
?x rdf:type ex:Actor
```

SELECT Queries (3)

A set of triple patterns build a *basic graph pattern*

- The heart of a SPARQL SELECT query

```
SELECT ?x ?date
WHERE {
   ?x rdf:type ex:Actor .
   ?x ex:bornIn ?date
}
```

SELECT Queries (4)

Graph example

```
ex:bradPitt rdf:type ex:Actor;
            ex:bornIn "1963"^^xsd:gYear.
ex:angelinaJolie rdf:type ex:Actor;
                 ex:bornIn "1975"^^xsd:gYear.
ex:jenniferAniston rdf:type ex:Actor.
```

SPARQL query solution

?x	?date
ex:bradPitt	"1963"^^xsd:gYear
ex:angelinaJolie	"1975"^^xsd:gYear

The OPTIONAL keyword

Resource ex:jenniferAniston not included in the results

- No RDF triple matches the second triple pattern

Use of the OPTIONAL keyword

- Retrieves all actors and actresses and get their birthdate only if it is specified in the RDF graph

```
SELECT ?x ?date
WHERE {
      ?x rdf:type ex:Actor .
      OPTIONAL {?x ex:bornIn ?date}
}
```

?x	?date
ex:bradPitt	"1963"^^xsd:gYear
ex:angelinaJolie	"1975"^^xsd:gYear
ex:jenniferAniston	

The UNION Keyword

Specify alternative graph patterns

```
SELECT ?x
WHERE {
    {?x rdf:type ex:Actor}
    UNION
    {?x rdf:type ex:Director}
}
```

Search for resources that are of type ex:Actor or ex:Director, including cases where a resource may belong to both classes

The FILTER Keyword

Set a condition that limits the result of a SPARQL query

```
SELECT ?x ?date
WHERE{
   ?x rdf:type ex:Actor .
   ?x ex:bornIn ?date .
   FILTER(?date < "1975"^^xsd:gYear)
}
```

Return all actors that were known to be born before 1975 and their corresponding birth date

ORDER BY and LIMIT Keywords

```
SELECT ?x ?date
WHERE {
   ?x rdf:type ex:Actor .
   ?x ex:bornIn ?date .
}
ORDER BY DESC(?date)
LIMIT 10
```

CONSTRUCT Queries

Generate an RDF graph based on a graph template using the solutions of a SELECT query

```
CONSTRUCT {?x rdf:type ex:HighlyPaidActor}
WHERE {
   ?x rdf:type ex:Actor .
   ?x ex:hasSalary ?salary .
   FILTER (?salary > 1000000)
}
```

ASK Queries

Respond with a boolean

Whether a graph pattern is satisfied by a subgraph of the considered RDF graph

```
ASK {
   ?x rdf:type ex:Actor .
   ?x foaf:name "Cate Blanchett" .
}
```

Return true if the RDF graph contains an actor named "Cate Blanchett"

DESCRIBE Queries

Return a set of RDF statements that contain data about a given resource

Exact structure of the returned RDF graph depends on the specific SPARQL engine

```
DESCRIBE <http://www.example.org/bradPitt> .
```

Outline

Introduction

RDF and RDF Schema

Description Logics

Querying RDF data with SPARQL

Mapping relational data with R2RML

Other technologies

Ontologies

Datasets

Mapping Relational Data to RDF (1)

Lack of RDF data volumes ⟶ Slow uptake of the Semantic Web vision

Large part of available data is stored in relational databases

Several methods and tools were developed for the translation of relational data to RDF

- Each one with its own set of features and, unfortunately, its own mapping language

The need for the reuse of RDB-to-RDF mappings led to the creation of R2RML

- A standard formalism by W3C for expressing such mappings

Mapping Relational Data to RDF (2)

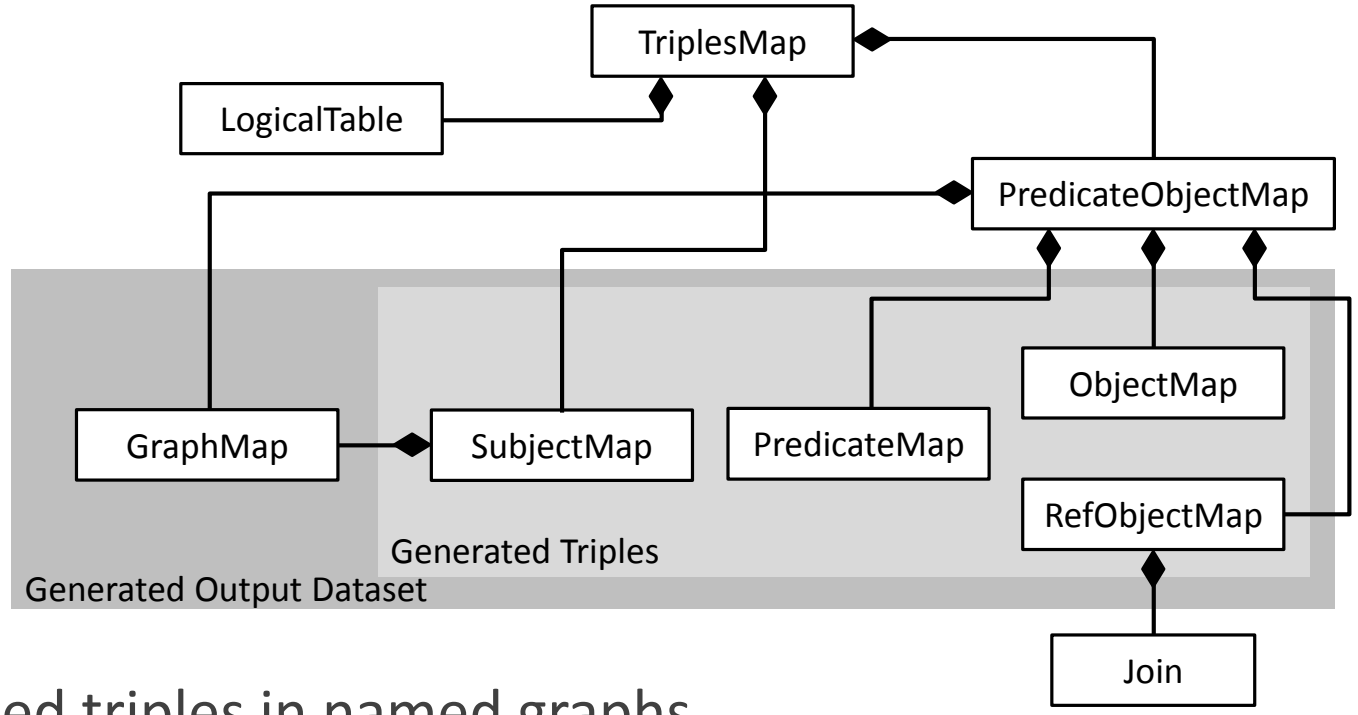
R2RML: RDB to RDF Mapping Language

- Provides a vocabulary for the definition of RDF views over relational schemas
- Is database vendor-agnostic

An R2RML mapping is an RDF graph: an *R2RML mapping graph*

- In Turtle syntax

R2RML Overview



More features

- Organization of generated triples in named graphs
- Definition of blank nodes
- Specification of a generated literal's language

R2RML (1)

An R2RML mapping
- Associates relational views with functions that generate RDF terms

Logical tables
- Relations or (custom) views defined in the relational schema

Term maps
- RDF generating functions
- Distinguished according to the position of the RDF term in the generated triple
  - Subject, predicate, and object maps

R2RML (2)

Triples maps

- Functions that map relational data to a set of RDF triples
- Groups of term maps
- R2RML mappings contain one or more triples maps

Graph maps

- Generated RDF triples can be organized into named graphs

R2RML Example (1)

A relational instance

FILM

ID	*Title*	*Year*	*Director*
1	The Hunger Games	2012	1

DIRECTOR

ID	*Name*	*BirthYear*
1	Gary Ross	1956

ACTOR

ID	*Name*	*BirthYear*	*BirthLocation*
1	Jennifer Lawrence	1990	Louisville, KY
2	Josh Hutcherson	1992	Union, KY

FILM2ACTOR

FilmID	*ActorID*
1	1
1	2

R2RML Example (2)

An R2RML triples map

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://www.example.org/>.
@prefix dc: <http://purl.org/dc/terms/>.

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "FILM" ];
    rr:subjectMap [
        rr:template "http://data.example.org/film/{ID}";
        rr:class ex:Movie;
    ];
    rr:predicateObjectMap [
        rr:predicate dc:title;
        rr:objectMap [ rr:column "Title" ];
    ];
    rr:predicateObjectMap [
        rr:predicate ex:releasedIn;
        rr:objectMap [ rr:column "Year";
                       rr:datatype xsd:gYear;];
].
```

R2RML Example (3)

The result (in Turtle)

```
<http://data.example.org/film/1> a ex:Movie;
                    dc:title "The Hunger Games";
                    ex:releasedIn "2012"^^xsd:gYear.
```

Note the reuse of terms from external ontologies

- E.g. dc:title from Dublin Core

R2RML Example (4)

The R2RML mapping graph of the example

- Specifies the generation of a set of RDF triples for every row of the FILM relation
- Is the simplest possible
  - Contains only one triples map

Every triples map must have exactly

- One logical table
- One subject map
- One or more predicate-object maps

R2RML Example (5)

A logical table represents an SQL result set

- Each row gives rise to an RDF triple
  - Or quad in the general case, when named graphs are used

A subject map is simply a term map that produces the subject of an RDF triple

R2RML Example (6)

Term maps

- Template-valued (rr:template)
  - The subject map of the example
- Constant-valued (rr:predicate)
  - The predicate map of the example
- Column-valued (rr:column)
  - The object map of the example

Every generated resource will be an instance of ex:Movie

- Use of rr:class

R2RML Example (7)

Subject maps

- Generate subjects

Predicate-object maps

- Generate pairs of predicates and objects
- Contain at least one predicate map
- Contain at least one object map

Typed Literals

- Use of rr:datatype
- The second object map of the example specifies that the datatype of the RDF literal that will be generated will be xsd:gYear

Foreign Keys (1)

Add another predicate-object map that operates on the DIRECTOR relation

Specify the generation of three RDF triples per row of the DIRECTOR relation

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://www.example.org/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

<#TriplesMap2>
    rr:logicalTable [ rr:tableName "DIRECTOR" ];
    rr:subjectMap [
        rr:template "http://data.example.org/director/{ID}";
        rr:class ex:Director;
    ];
    rr:predicateObjectMap [
        rr:predicate foaf:name;
        rr:objectMap [ rr:column "Name" ];
    ];
    rr:predicateObjectMap [
        rr:predicate ex:bornIn;
        rr:objectMap [ rr:column "BirthYear";
                       rr:datatype xsd:gYear;];
    ].
```

Foreign Keys (2)

Add another predicate-object map, the referencing object map

Link entities described in separate tables

Exploit the foreign key relationship of FILM.Director and DIRECTOR.ID

```
<#TriplesMap1>
    rr:logicalTable [ rr:tableName "FILM" ];
    rr:subjectMap [
        rr:template "http://data.example.org/film/{ID}";
        rr:class ex:Movie;
    ];
    rr:predicateObjectMap [
        rr:predicate dc:title;
        rr:objectMap [ rr:column "Title" ];
    ];
    rr:predicateObjectMap [
        rr:predicate ex:releasedIn;
        rr:objectMap [ rr:column "Year";
                       rr:datatype xsd:gYear;];

    rr:predicateObjectMap[
        rr:predicate ex:directedBy;
        rr:objectMap[
                rr:parentTriplesMap <#TriplesMap2>;
                rr:joinCondition[
                            rr:child "Director";
                            rr:parent "ID";
                                ];
                        ];
    ].
```

Foreign Keys (3)

TriplesMap1 now contains a *referencing object map*

- Responsible for the generation of the object of a triple
- Referencing object maps are a special case of object maps
- Follows the generation rules of the subject map of another triples map
  - Called *parent triples* map
- Join conditions are also specified in order to select the appropriate row of the logical table of the *parent triples* map

Foreign Keys (4)

Not necessary for this foreign key relationship to be explicitly defined as a constraint in the relational schema

R2RML is flexible enough to allow the linkage of any column set among different relations

TriplesMap1 result

```
<http://data.example.org/film/1> ex:directedBy <http://data.example.org/director/1>.
```

TriplesMap2 result

```
<http://data.example.org/director/1> a ex:Director;
    foaf:name "Gary Ross";
    ex:bornIn "1956"^^xsd:gYear.
```

Custom Views (1)

R2RML view defined via the rr:sqlQuery property

Used just as the native logical tables

Could also have been defined as an SQL view in the underlying database system

- Not always feasible/desirable

Produced result set must not contain columns with the same name

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://www.example.org/>.
@prefix dc: <http://purl.org/dc/terms/>.

<#TriplesMap3>

    rr:logicalTable [ rr:sqlQuery """
                      SELECT ACTOR.ID AS ActorId, ACTOR.Name AS
ActorName, ACTOR.BirthYear AS ActorBirth, FILM.ID AS FilmId FROM
ACTOR, FILM, FILM2ACTOR WHERE FILM.ID=FILM2ACTOR.FilmID AND
ACTOR.ID=FILM2ACTOR.ActorID; """];

    rr:subjectMap [
        rr:template "http://data.example.org/actor/{ActorId}";
        rr:class ex:Actor;
    ];
    rr:predicateObjectMap [
        rr:predicate foaf:name;
        rr:objectMap [ rr:column "ActorName" ];
    ];
    rr:predicateObjectMap [
        rr:predicate ex:bornIn;
        rr:objectMap [ rr:column "ActorBirth";
                       rr:datatype xsd:gYear;];
    ].
    rr:predicateObjectMap [
        rr:predicate ex:starsIn;
        rr:objectMap [ rr:template
"http://data.example.org/film/{ID}";
                     ];
    ].
```

Custom Views (2)

TriplesMap3 generates the following triples

```
<http://data.example.org/actor/1> a ex:Actor;
    foaf:name "Jennifer Lawrence";
    ex:bornIn "1990"^^xsd:gYear;
  ex:starsIn<http://data.example.org/film/1>.
<http://data.example.org/actor/2> a ex:Actor;
    foaf:name "Josh Hutcherson";
    ex:bornIn "1992"^^xsd:gYear;
    ex:starsIn <http://data.example.org/film/1>.
```

Direct Mapping (1)

A default strategy for translating a relational instance to an RDF graph

A trivial transformation strategy

A recommendation by W3C since 2012

Defines a standard URI generation policy for all kinds of relations

Direct Mapping (2)

Maps every relation to a new ontology class and every relation row to an instance of the respective class

The generated RDF graph essentially mirrors the relational structure

Can be useful as a starting point for mapping authors who can then augment and customize it accordingly using R2RML

Outline

Introduction

RDF and RDF Schema

Description Logics

Querying RDF data with SPARQL

Mapping relational data with R2RML

Other technologies

Ontologies

Datasets

POWDER – Protocol for Web Description Resources (1)

XML-based protocol

Allows provision of information about RDF resources

A POWDER document contains

- Its own provenance information
  - Information about its creator, when it was created, etc.
- A list of description resources

POWDER – Protocol for Web Description Resources (2)

Description resource

- Contains a set of property-value pairs in RDF
- For a given resource or group of resources

Allows for quick annotation of large amounts of content with metadata

Ideal for scenarios involving personalized delivery of content, trust control and semantic annotation

RIF – Rule Interchange Format

An extensible framework for the representation of rule dialects

A W3C recommendation

A family of specifications

- Three rule dialects
  - RIF-BLD (Basic Logic Dialect)
  - RIF-Core and
  - RIF-PRD (Production Rule Dialect)
- RIF-FLD (Framework for Logic Dialects)
  - Allows the definition of other rule dialects
  - Specifies the interface of rules expressed in a logic-based RIF dialect with RDF and OWL

SPIN – SPARQL Inferencing Notation

RIF uptake not as massive as expected

- Complexity, lack of supporting tools

SPIN

- SPARQL-based
- Production rules

Links RDFS and OWL classes with constraint checks and inference rules

- Similar to class behavior in object-oriented languages

GRDDL – Gleaning Resource Descriptions from Dialects of Languages

A W3C recommendation

Mechanisms that specify how to construct RDF representations

- Can be applied to XHTML and XML documents
- Defines XSLT transformations that generate RDF/XML

Is the standard way of converting an XML document to RDF

Familiarity with XSLT required

LDP – Linked Data Platform

Best practices and guidelines for organizing and accessing LDP resources

- Via HTTP RESTful services
- Resources can be RDF or non-RDF

A W3C recommendation

- Specifies the expected behavior of a Linked Data server when it receives HTTP requests
  - Creation, deletion, update
- Defines the concept of a Linked Data Platform Container
  - A collection of resources (usually homogeneous)

LDP specification expected to establish a standard behavior for Linked Data systems

Example: Apache Marmotta

Outline

Introduction

RDF and RDF Schema

Description Logics

Querying RDF data with SPARQL

Mapping relational data with R2RML

Other technologies

Ontologies

Datasets

Ontologies and Datasets (1)

Standards and technologies (e.g. RDF and OWL)

- Machine-consumable serializations
- Syntactical groundwork
- Assignment of meaning to resources and their relationships

Vocabularies

- (Re)used in order for the described information to be commonly, unambiguously interpreted and understood
- Reused by various data producers
- Make data semantically interoperable

Ontologies and Datasets (2)

Datasets

- Collections of facts involving specific individual entities and ontologies, which provide an abstract, high-level, terminological description of a domain, containing axioms that hold for every individual

A great number of ontologies exist nowadays

- Subject domains: e.g. geographical, business, life sciences, literature, media

Ontology search engines and specialized directories

Schemapedia

Watson

Swoogle

Linked Open Vocabularies (LOV)

- The most accurate and comprehensive source of ontologies used in the Linked Data cloud
- Vocabularies described by appropriate metadata and classified to domain spaces
- Full-text search enabled at vocabulary and element level

DC – Dublin Core (1)

A minimal metadata element set

Mainly used for the description of web resources

DC vocabulary

- Available as an RDFS ontology
- Contains classes and properties
  - E.g. agent, bibliographic resource, creator, title, publisher, description

DC – Dublin Core (2)

DC ontology partitioned in two sets

- Simple DC
  - Contains 15 core properties
- Qualified DC
  - Contains all the classes and the rest of the properties (specializations of those in simple DC)

FOAF – Friend-Of-A-Friend

One of the first lightweight ontologies being developed since the first years of the Semantic Web

Describes human social networks

Classes such as Person, Agent or Organization

Properties denoting personal details and relationships with other persons and entities

SKOS – Simple Knowledge Organization System (1)

A fundamental vocabulary in the Linked Data ecosystem

A W3C recommendation since 2009

Contains terms for organizing knowledge

- In the form of taxonomies, thesauri, and concept hierarchies

Defines concept schemes as sets of concepts

- Concept schemes can relate to each other through "broader/narrower than" or equivalence relationships

SKOS – Simple Knowledge Organization System (2)

Defined as an OWL Full ontology

- Has its own semantics
  - Its own set of entailment rules distinct from those of RDFS and OWL

Widely used in the library and information science domain

VoID – Vocabulary of Interlinked Datasets

Metadata for RDF datasets

- General metadata
  - E.g. Name, description, creator
- Information on the ways that this dataset can be accessed
  - E.g. as a data dump, via a SPARQL endpoint
- Structural information
  - E.g. the set of vocabularies that are used or the pattern of a typical URI
- Information on the links
  - E.g. the number and target datasets

Aid users and machines to decide whether a dataset is suitable for their needs

SIOC – Semantically-Interlinked Online Communities

An OWL ontology

Describes online communities

- E.g. forums, blogs, mailing lists

Main classes

- E.g. forum, post, event, group, user

Properties

- Attributes of those classes
- E.g. the topic or the number of views of a post

Good Relations

Describes online commercial offerings

- E.g. Product and business descriptions, pricing and delivery methods

Great impact in real-life applications

- Adoption by several online retailers and search engines

Search engines able to interpret product metadata expressed in the Good Relations vocabulary

- Offer results better tailored to the needs of end users

Outline

Introduction

RDF and RDF Schema

Description Logics

Querying RDF data with SPARQL

Mapping relational data with R2RML

Other technologies

Ontologies

Datasets

Datasets

Several “thematic neighborhoods” in the Linked Data cloud

- E.g. government, media, geography, life sciences

Single-domain and cross-domain datasets

DBpedia (1)

The RDF version of Wikipedia

- Perhaps the most popular RDF dataset in the LOD cloud
- A large number of incoming links

A cross-domain dataset

- Assigns a URI to every resource described by a Wikipedia article
- Produces structured information by mining information from Wikipedia infoboxes

DBpedia (2)

A multilingual dataset

- Transforms various language editions of Wikipedia

The DBpedia dataset offered as data dump and through a SPARQL endpoint

A “live” version available, updated whenever a Wikipedia page is updated

Freebase

Openly-licensed, structured dataset, also available as an RDF graph

Tens of millions of concepts, types and properties

Can be edited by anyone

Gathers information from several structured sources and free text

Offers an API for developers

Linked to DBpedia through incoming and outgoing links

GeoNames

An open geographical database

Millions of geographical places

Can be edited by anyone

Geographical information as the context of descriptions and facts

Omnipresent in the Linked Data cloud

Several incoming links from other datasets

Lexvo

Central dataset for the linguistic Linked Data Cloud

Provides identifiers for thousands of languages

URIs for terms in every language

Identifiers for language characters