Future Internet

ILARIA TORRE

Ilaria.torre@unige.it

Principles of the Semantic Web

Tim Berners-Lee states:

- XML allows users to add *arbitrary structure* to their documents but says nothing about what the structures mean [2001]

To overcome this limit the idea is splitting:

1. the way to express statements about resources,

2. the concepts and relations that represent the domain knowledge

Principles of the Semantic Web

The solution proposed by Berners-Lee is adopting:

1) For the first point:

- a common simple domain-independent model for the expression of statements about resources
- a set of standard syntactic conventions for representing those statements

➔ Resource Description Framework

2) For the second point:

- shared vocabularies and ontologies to represent the domain knowledge

Recap: Semantic Web

At first, it can be defined as the transformation of the Web into a machine-understandable "machine-processable" version where documents are semantically annotated with formally defined metadata specified in vocabularies called ontologies.

More generally, the Semantic Web is a framework for enabling the exchange of data and the reuse of that data among applications, using formal descriptions of concepts and their relationships.

Theorized in 1998, the Semantic Web only in recent years is finding application in many domains.

Central components:

- The Resource Description Framework (RDF)
- The use of vocabularies and ontologies

RDF (Resource Description Framework)

- RDF (Resource Description Framework) is the **data model and language** proposed by W3C to express assertions, that is, to describe the metadata about a resource and its relationships to other resources.

- Everything is definable as a **resource**: a resource can be an object, a person, a group of people, a document, a web page, an email, an abstract concept.

RDF (Resource Description Framework)

- The Resource Description Framework (RDF) is a W3C recommendation that defines a language for describing resources.
- It was designed for describing Web resources such as Web pages. However, RDF does not require that resources be retrievable on the Web. RDF resources may be physical objects, abstract concepts, in fact anything that has identity.
- Thus, RDF defines a language for describing just about anything.

Brian McBride, 2004

RDF (Resource Description Framework)

RDF

- is the core of the SW (W3C Recommendation)
- is a <u>domain-independent</u> data model for the expression of statements and a language.

Its data model is a graph data model

Every statement can be expressed as a triple `<Subject,Predicate,Object>`

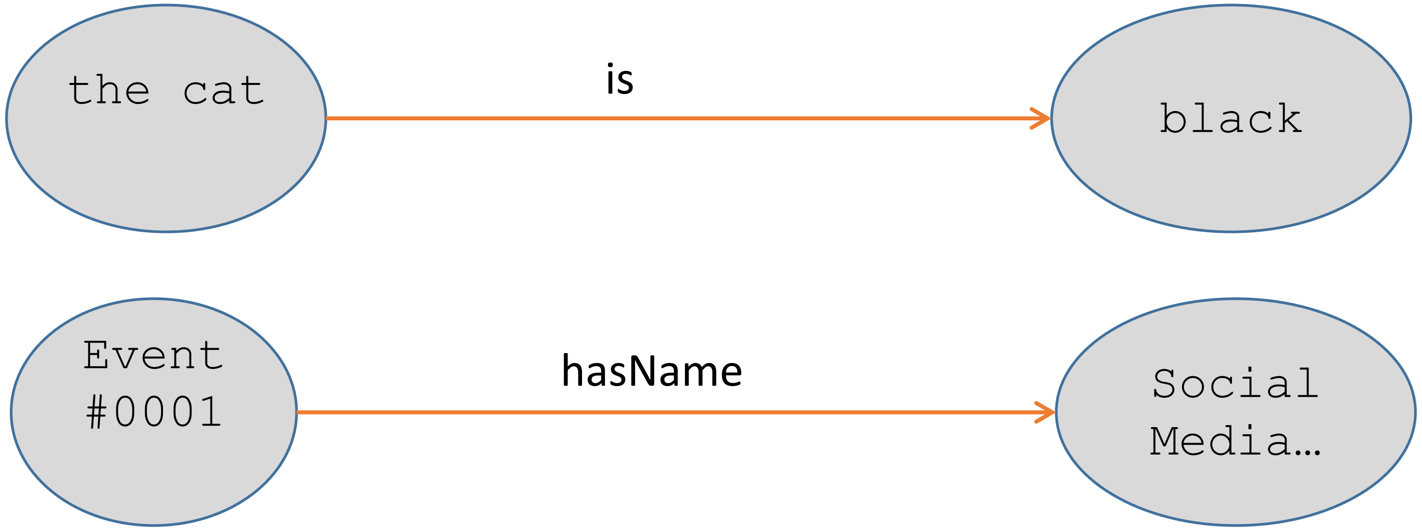
- each triple can be represented as a node-arc-node link;
- a set of such triples is called an RDF graph

Source https://www.w3.org/RDF/

RDF triples

SUBJECTS can be whatever resource that is being described

PREDICATES can be properties of a resource or relationships between resources

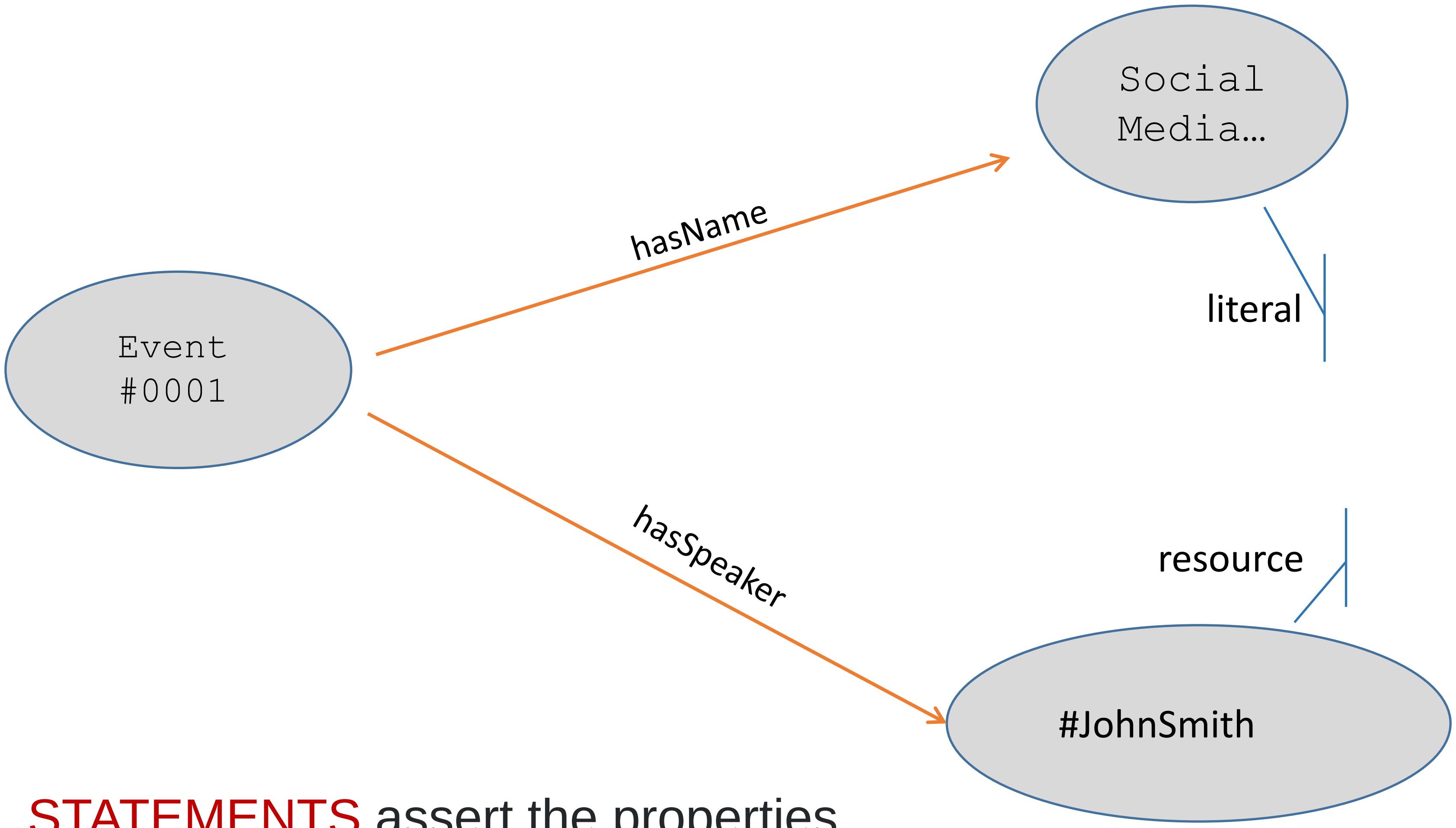


Conventionally, predicates (properties) start with lowercase. Often predicates are defined with verbs.

E.g. hasName, hasStartDate

In the following, we will use this convention

RDF graph



OBJECTS can be:

- a literal such as a number or string (e.g. Social Media). In this case the predicate represents a property of the subject (e.g. hasName) and the object is its value;
- a resource (e.g. #JohnSmith). In this case, the predicate represents a relation between resources

STATEMENTS assert the properties of resources in form of triples subject-predicate-object

Triples

PREDICATE

Event	hasName	hasStartDate	…………
Event#0001	Social Media…	2017-07-03	……….
Event#0002	Internet of Things	2017-09-11	…………
Event#0003	Machine Learning	2018-09-01	…………

SUBJECT

OBJECT

```
<subject,      predicate,       object>

<"Event#0001", hasName,       "Social Media">
<"Event#0001", hasStartDate,  "2017-07-03">
<"Event#0002", hasName,       "Internet of Things">
<"Event#0002", hasStartDate,  "2017-09-11">
<"Event#0003", hasName,       "Machine Learning">
<"Event#0003", hasStartDate,  "2018-09-01">
```

RDF language and data model

In summary:

RDF language

- An XML-based syntax for representing data
- A language for expressing subject-predicate-object statements (binary predicates)

  Statements assert the properties of resources in form of triples subject-predicate-object

RDF data model

- The data model in RDF is a graph data model
- An edge with two connecting nodes forms a triple

Ontologies

- In *philosophy* it is concerned with the study of being or existence.

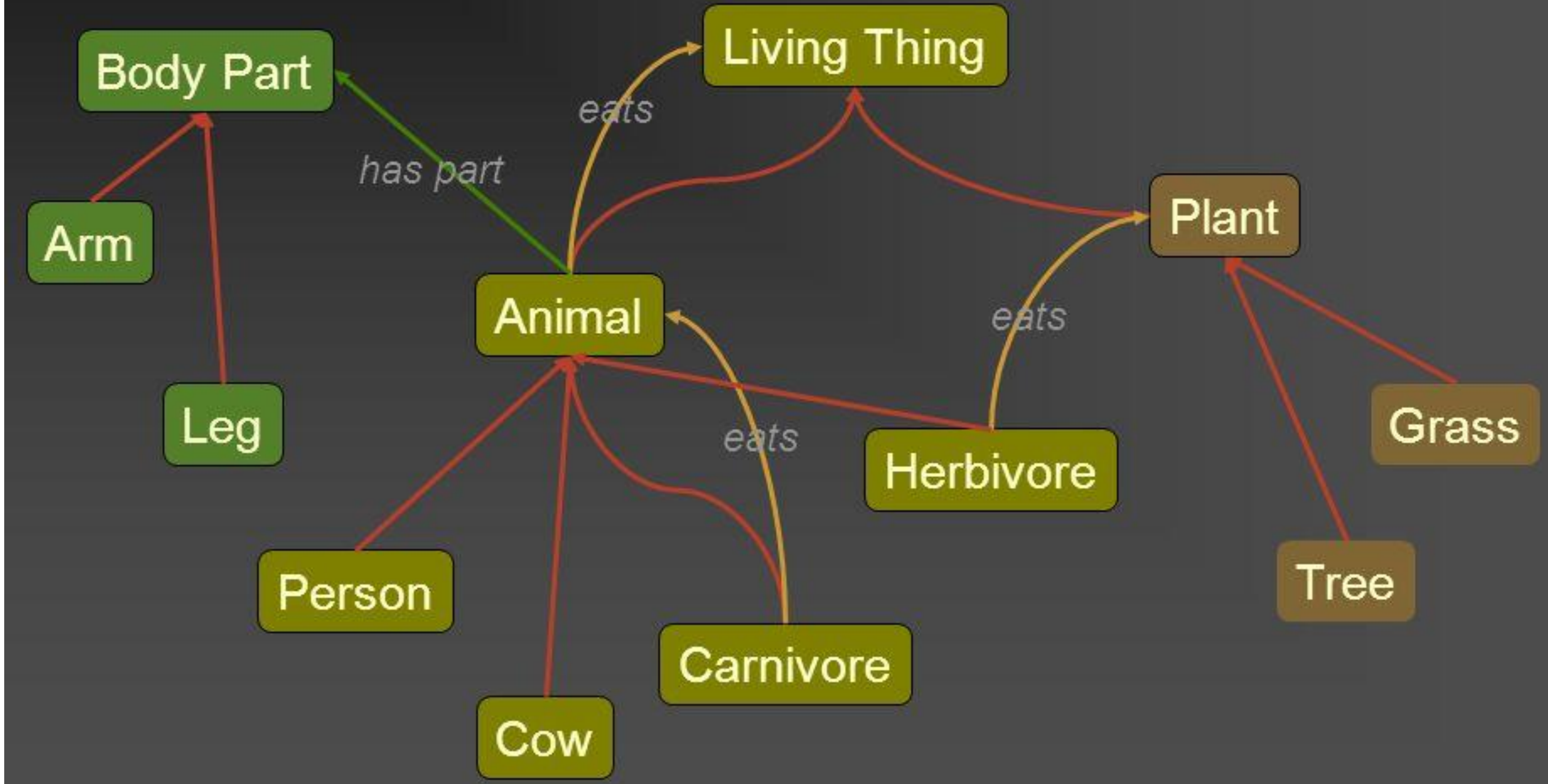
- In *computer and information science*, ontology denotes an artifact that is designed for a purpose: the modeling of knowledge about some domain.

A well-established definition is:

An ontology is a formal, explicit specification of a shared conceptualization (Gruber 1993, Studer et al 1998)

Source: http://tomgruber.org/writing/ontology-definition-2007.htm

A simple ontology: Animals



Published by Gilbert Parrish

Ontologies

"Explicit and formal representation of a shared conceptualization"

"conceptualization": an abstract model of some phenamenon, which identifies the relevant concepts

"explicit": the types of concepts used and the restrictions must be explicitly defined

"formal" must be "machine-understanstandable"

"Shared" an ontology captures a consensual knowledge, not restricted to a few individuals

Ontologies

An ontology formally describes the domain of a discourse and consists of:

- **a finite list of terms denoting concepts** (i.e., object classs) of the domain

- **Relationships**: Class hierarchies, properties, value restrictions, logical and disjunction relationships.

Ontologies

Hierarchical relationships are those typical of **taxonomies**

Ontologies: Purposes

- **Interoperability**: sharing a common understanding of the structure of information in a group
- **reuse** of domain knowledge, so as not to duplicate efforts in the development of software systems
- **separation** of declarative domain knowledge from procedural knowledge
- **communication** between SW agents and between SW and humans, providing common terminology
- **Make** domain assumptions **explicit**

Ontologies: methodology

Designing an ontology means:

- Define classes
- Organize classes in a taxonomic hierarchy (subclass-superclass)
- Define class properties and class relationships, and describe the permissible values for each class
- Assign values to properties for all instances that you create.

Ontologies: methodology

- Ontology must reflect the reality of the world it seeks to shape
- There is **no a priori** correct method for modeling a domain of knowledge
- The best solution depends on the intended application and its future extensions
- The process of developing an ontology is an **iterative** process

Ontologies: methodology

- **Concepts** usually correspond to nouns used in the knowledge domain

- **Properties** can correspond to adjectives:
  - "Intrinsic" properties (the taste of a wine)
  - "Extrinsic" properties (the name of a wine)
  - Constituent parts (the parts of the body)

- **Relationships** with other parties (owner of ...): verbs

Ontologies: methodology

Properties ("slots") can have several restrictions ("facets"):

- **Cardinality**: How many values can I have in a slot
- **Value type**: data type
- **Domain**: which classes does this property apply to?
- **Range**: allowed classes and values for properties that can be defined on instances.

Possible approaches

- **Top-down**:

  start by defining the most general concepts and specialize these definitions into more specific concepts.

- **Bottom-up**:

  start by defining the most specific classes, and try to group them into more abstract conceptual groups.

Ontologies: methodology

- **Steps**

1. Determine the **domain** and **purpose** of the ontology
2. Consider **reusing** existing ontologies
3. **Enumerate** the main terms of the ontology
4. Define **classs** and the **hierarchy** between classs
5. Define Class **Properties**
6. Define **constraints** on properties
7. Create the **instances**

Noy, McGuinness, "**Ontology Development 101: A Guide to Creating Your First Ontology** ",2001

Lab: try and practice

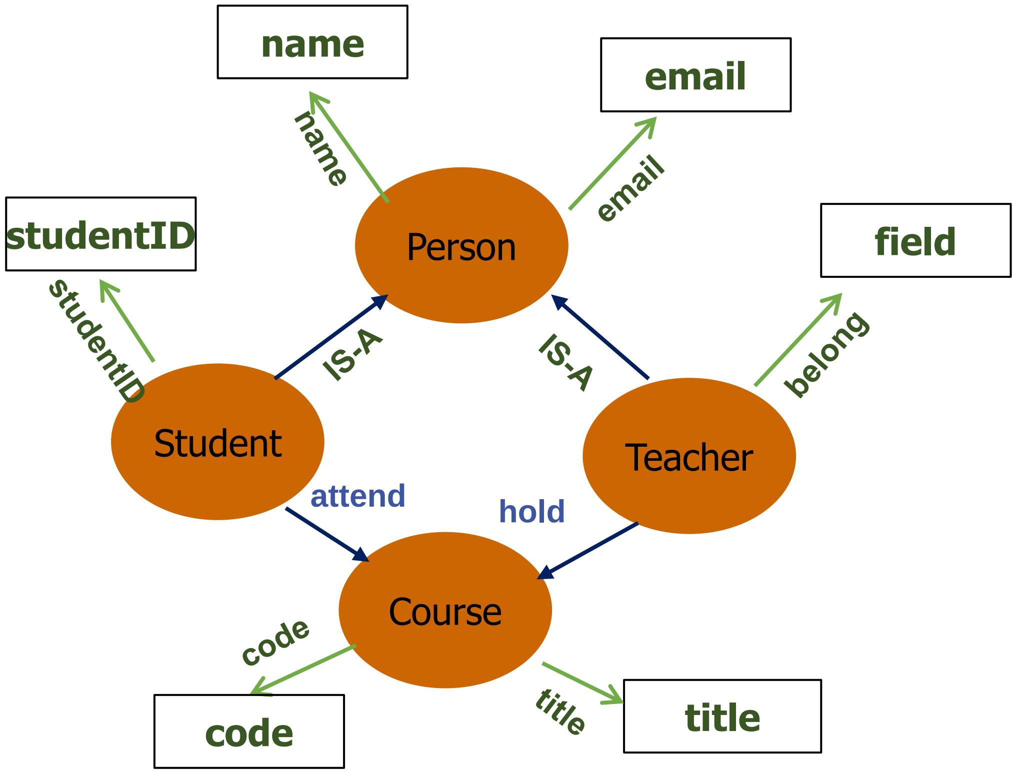
- Conceptual definition of an ontology for the University domain
  - Classses
  - Data properties
  - Object properties
- Use Protégé to build the ontology
  - Specify the ontology URI/IRI
  - Define classes and properties (T-box)
  - Save in a serialization language
  - Add instances and assertions about such instances (A-box)
  - Save in a serialization language

RDF data model

Concept/class

Data properties

Object properties



Predicate (Subject, Object/data)

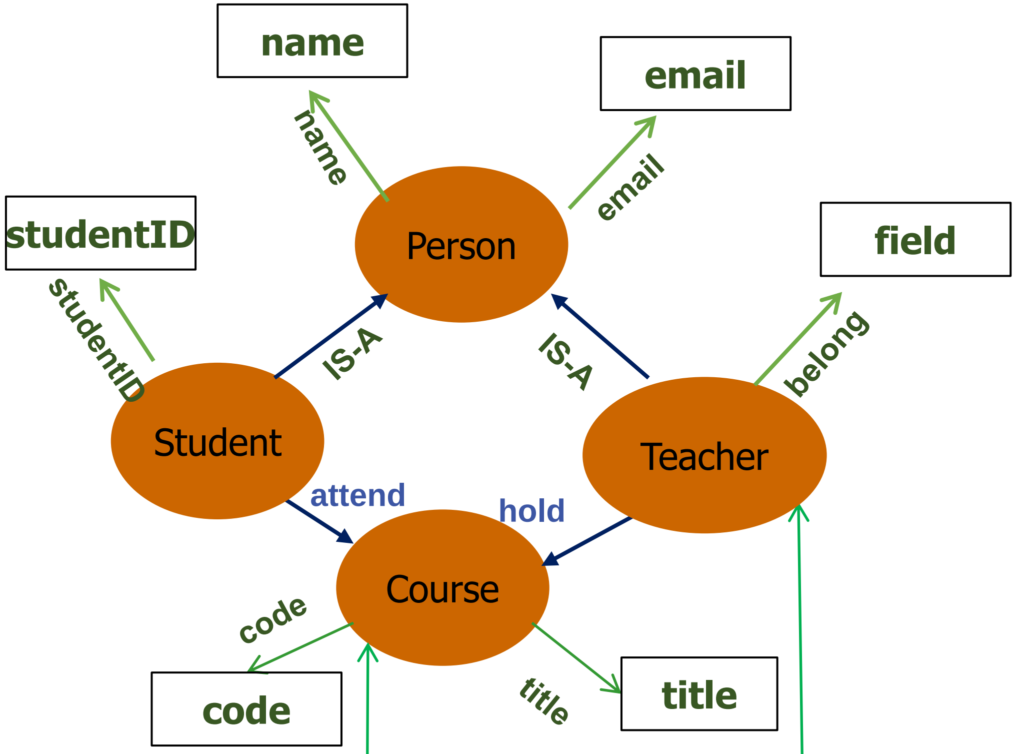
dataProperty (Subject, data)

objectProperty (Subject, Object)

code (Course, code)

hold (Teacher, Course)

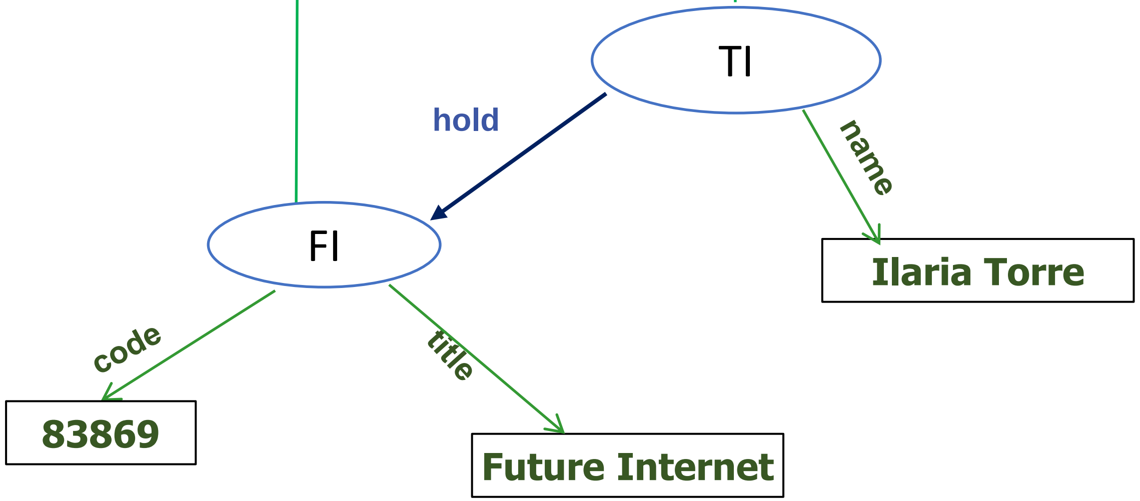
T-Box (terminology box):
Ontology Schema



hold (Teacher, Course)

code (Course, code)
title(Course, title)

A-Box (assertion box)



hold (TI,FI)
code (FI, 83869)
Title (FI, 'Future Internet')
....

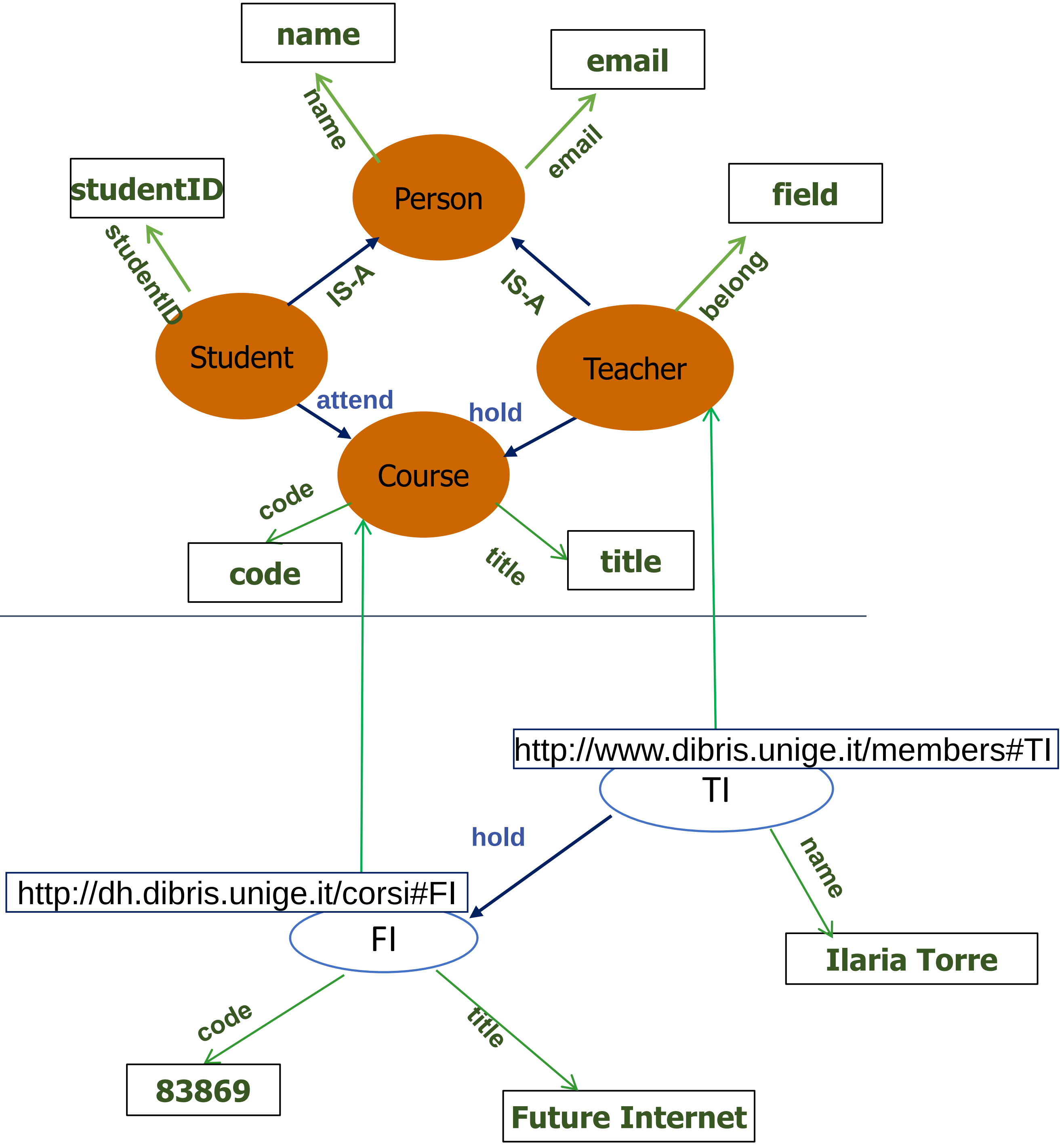
T-Box (terminology box):
Ontology Schema

hold (Teacher, Course)

code (Course, code)
title(Course, title)

A-Box (assertion box)

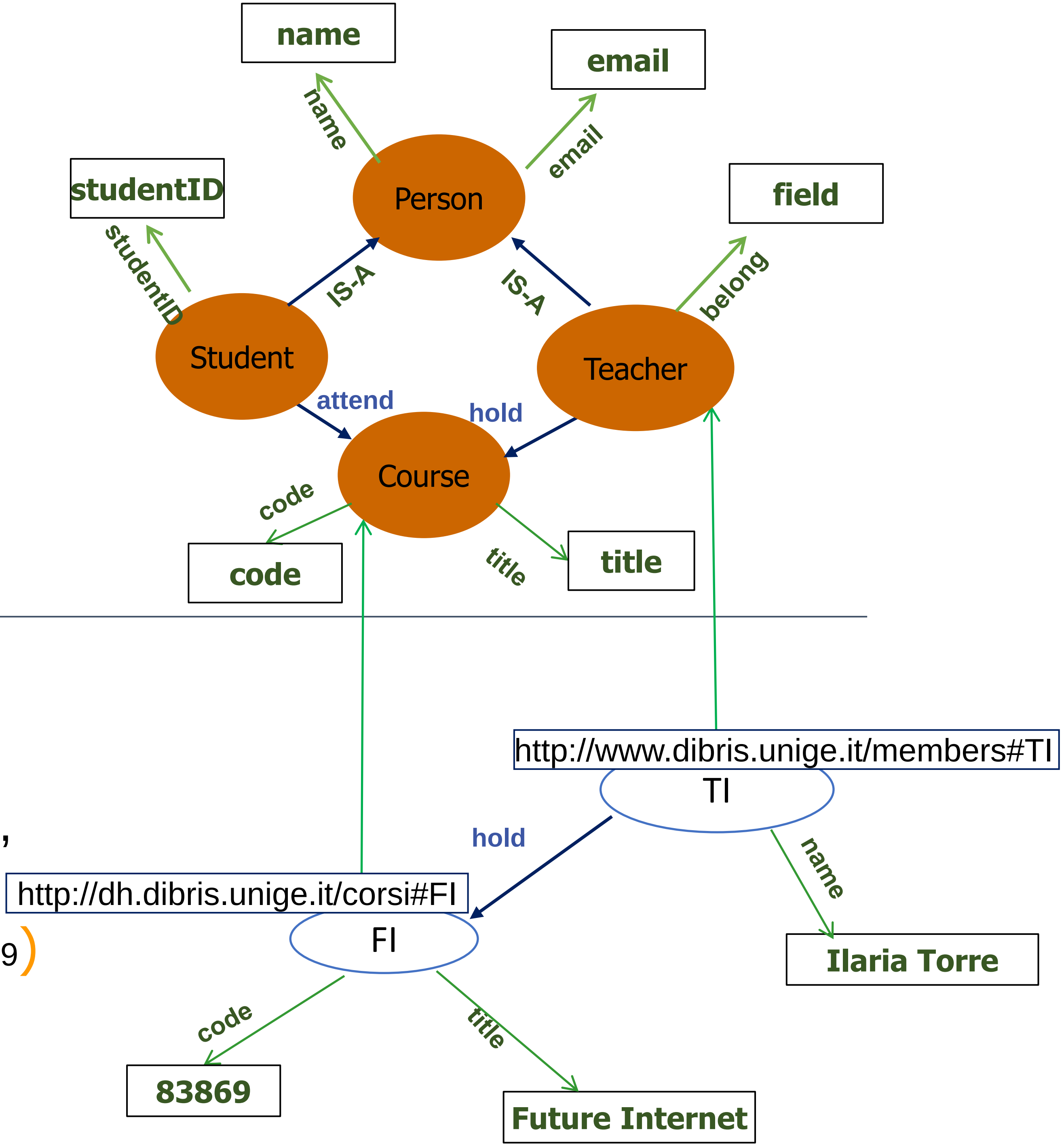
hold (TI,FI)
code (FI, 83869)
Title (FI, ‘Future Internet’)
….



T-Box (terminology box): Ontology Schema

hold (Teacher, Course)

code (Course, code)
title(Course, title)



A-Box (assertion box)

hold (http://www.dibris.unige.it/members#TI, http://dh.dibris.unige.it/corsi#FI)

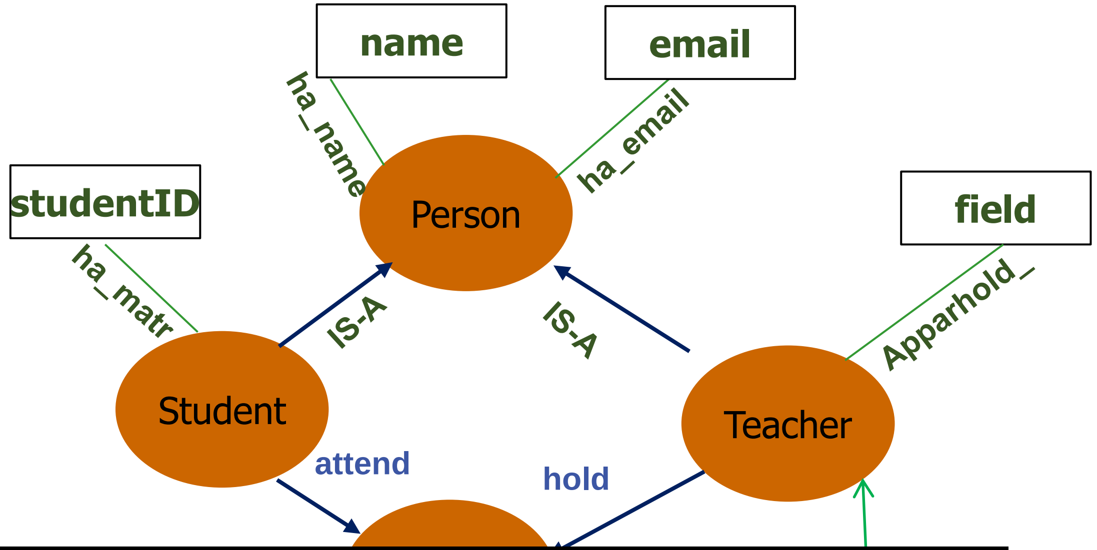
code (http://dh.dibris.unige.it/corsi#FI, 83869)

....

T-Box (terminology box):
Ontology Schema

hold (Teacher, Course)

code (Course, code)

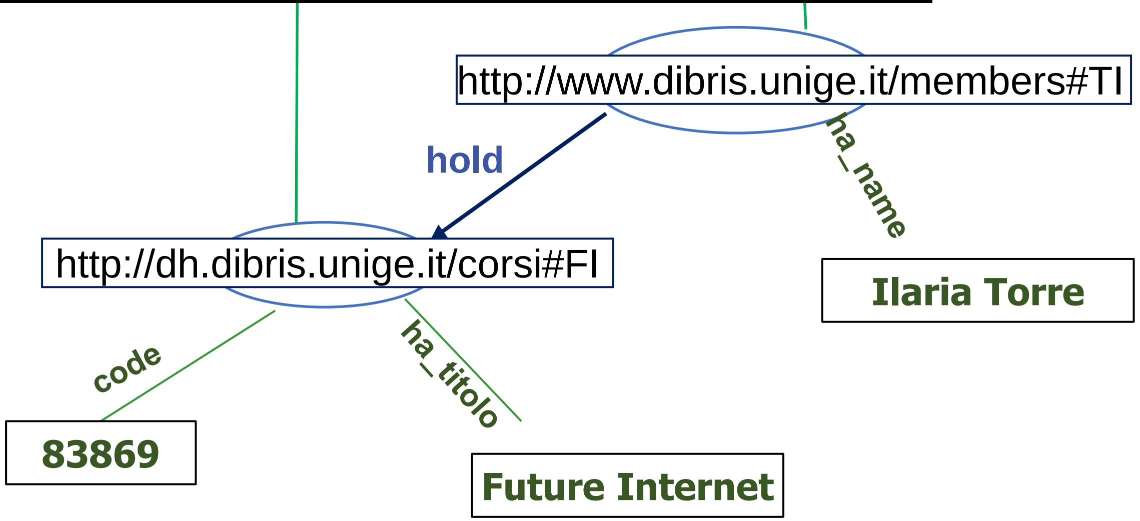


Note: Classes and properties are also identified with a URI/IRI: they are those of the vocabulary/ontology (T-Box) used to annotate resources. In this example, the URI of the hold property in the A-Box will be the same URI that identifies the hold property in the vocabulary, i.e. in the T-Box

hold (http://www.dibris.unige.it/members#TI,
http://dh.dibris.unige.it/corsi#FI)

code (http://dh.dibris.unige.it/corsi#FI, 83869)

....



Use the ontology editor Protégé to build the ontology of the University domain

https://protege.stanford.edu/

Vedere “Guida su modellazione di ontologie e utilizzo di Protege" su AulaWeb

Ontology: Rule I

When to introduce a new class?

Subclasses usually have:

- additional properties that the superclass does not possess,
- different restrictions than those of the superclass,
- participate in relationships other than the superclass

Ontology: Rule II

Create a class or a property?

- If a feature is important in the domain, and you think that objects with different values for the feature are objects of different types, then you should create a new class.
- Otherwise, the feature is more simply represented by a different value of the property.

Ontologie: Rule III

Analyze the "children" of the hierarchy

- "Children" are classes that are direct subclasses of the same class. All children in the hierarchy (except at most the children of the root) must have the same level of generality

When are they too many and when are they too few?

- If a class has only one direct subclass, there may be a modeling problem: the ontology is not complete
- If there are more than 10-12 subclasses, intermediate classes are probably needed

Ontologie: Rule IV

Classes and class names

- It is important to distinguish between classes and their names
- Classes represent concepts in the domain, not the words used to express those concepts
- Synonyms do not constitute different classes

Avoid Class Loops

- If A is a subclass of B and B is a subclass of A, this is equivalent to saying that A and B are the same class

Ontologie: Rule V

Ensure that the class hierarchy is correct

- The class hierarchy represents the "is-a" relationship
- A subclass of a class represents a Concept that is a "possible type of" with respect to the Concept represented by the superclass
- Hierarchical relationships are transitive:

  if B is a subclass of A and C is a subclass of B,

  then C is a subclass of A.

Problems with ontologies

- Interoperability between ontologies (ontology mapping problem)
- Usability/reusability tradeoff
- Time consuming

Development of ontologies

There are development tools to make the creation of ontologies less time-consuming and resource-intensive. E.g.,

Protègè
OntoEdit

Ontologies

Schema ontology/vocabulary defines:

- the concepts (domain classs) and their properties
- relationships between the concepts
- formal constraints that are relevant for that domain

T-Box
(terminology box)

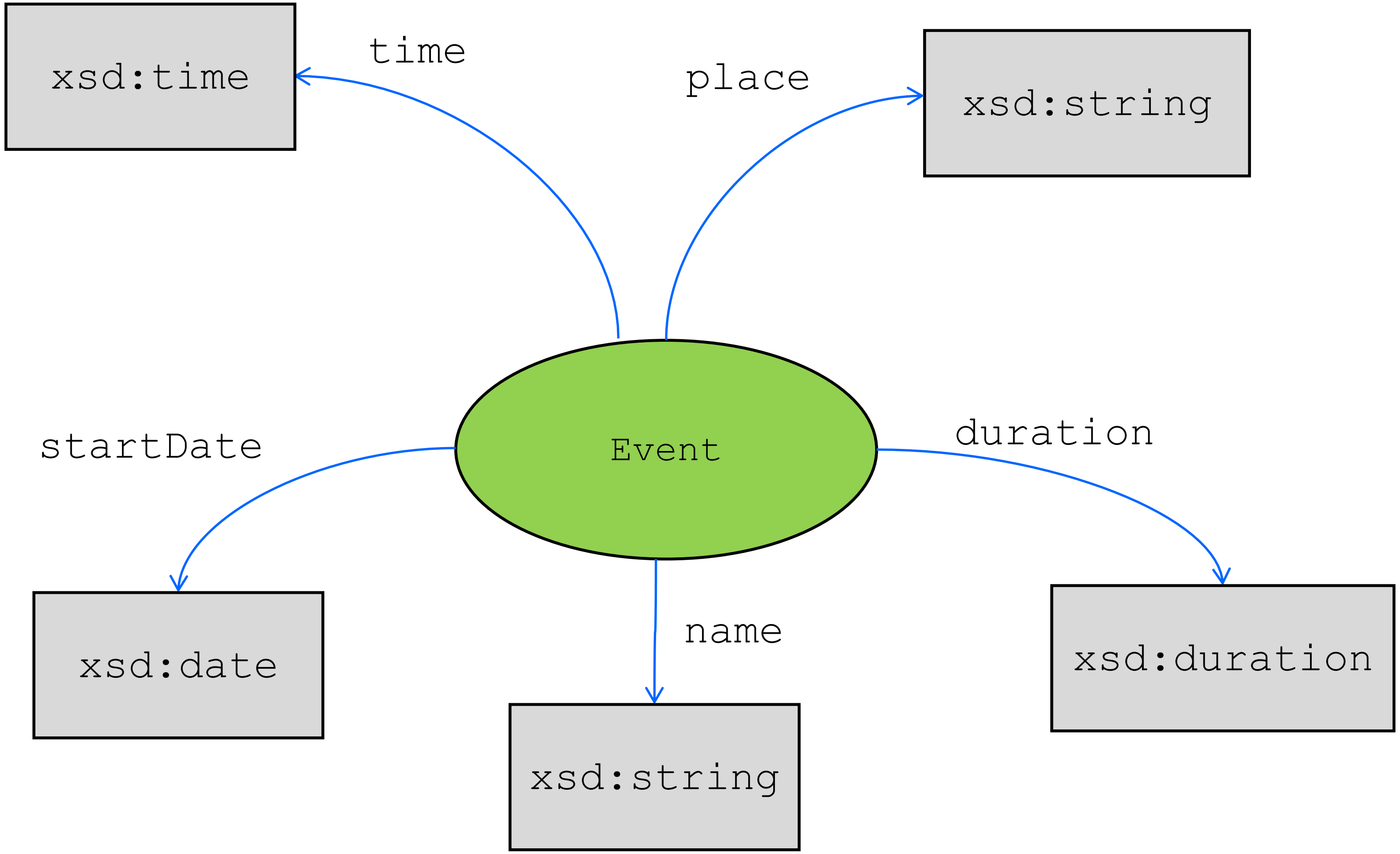
Ontology instances:

- the individual instances of classs
- their relationships

A-Box
(assertions box)

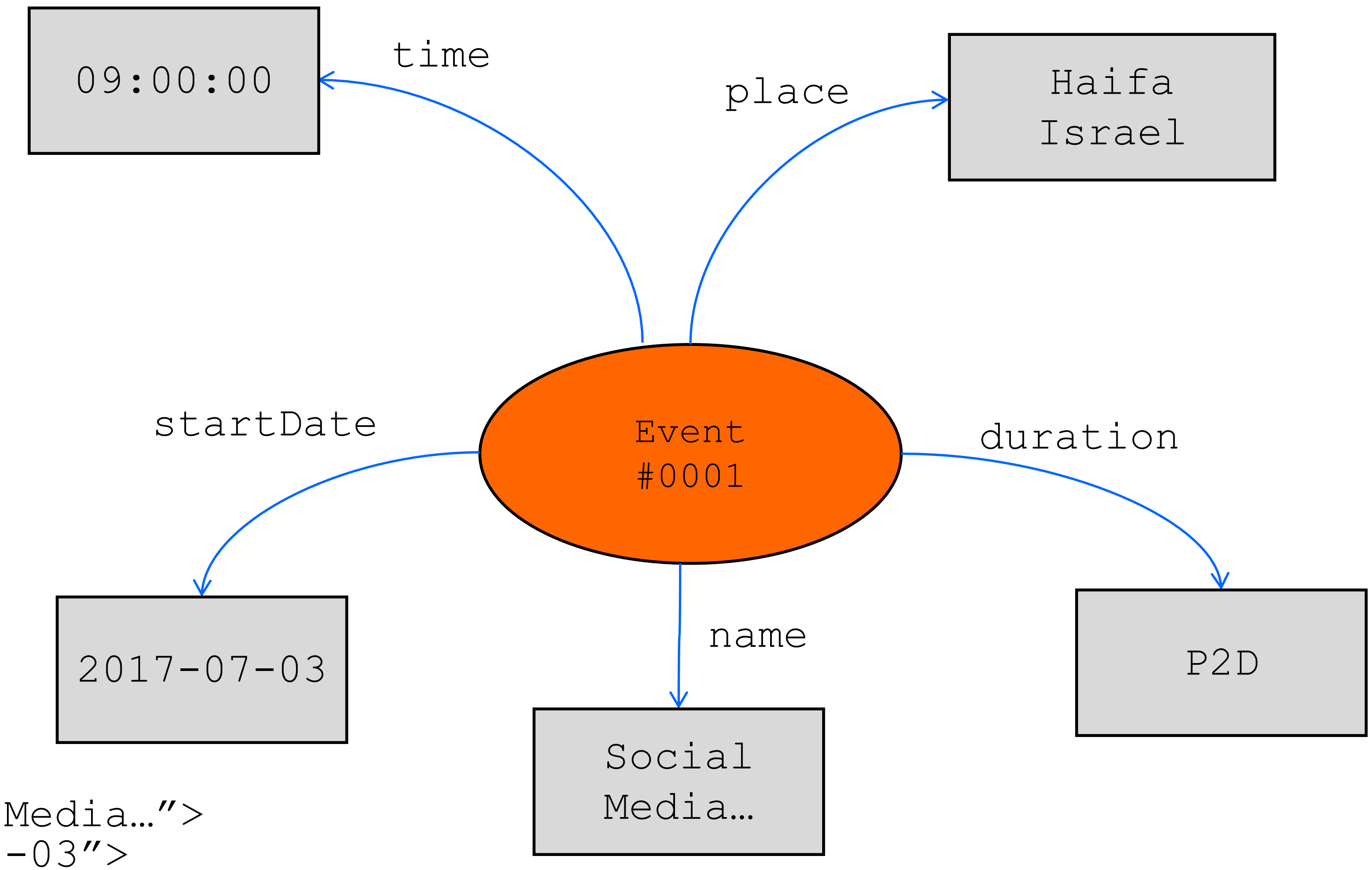
Example: definition of the Event Class for a new ontology that models events

T-Box



Example: an instance of the Event Class

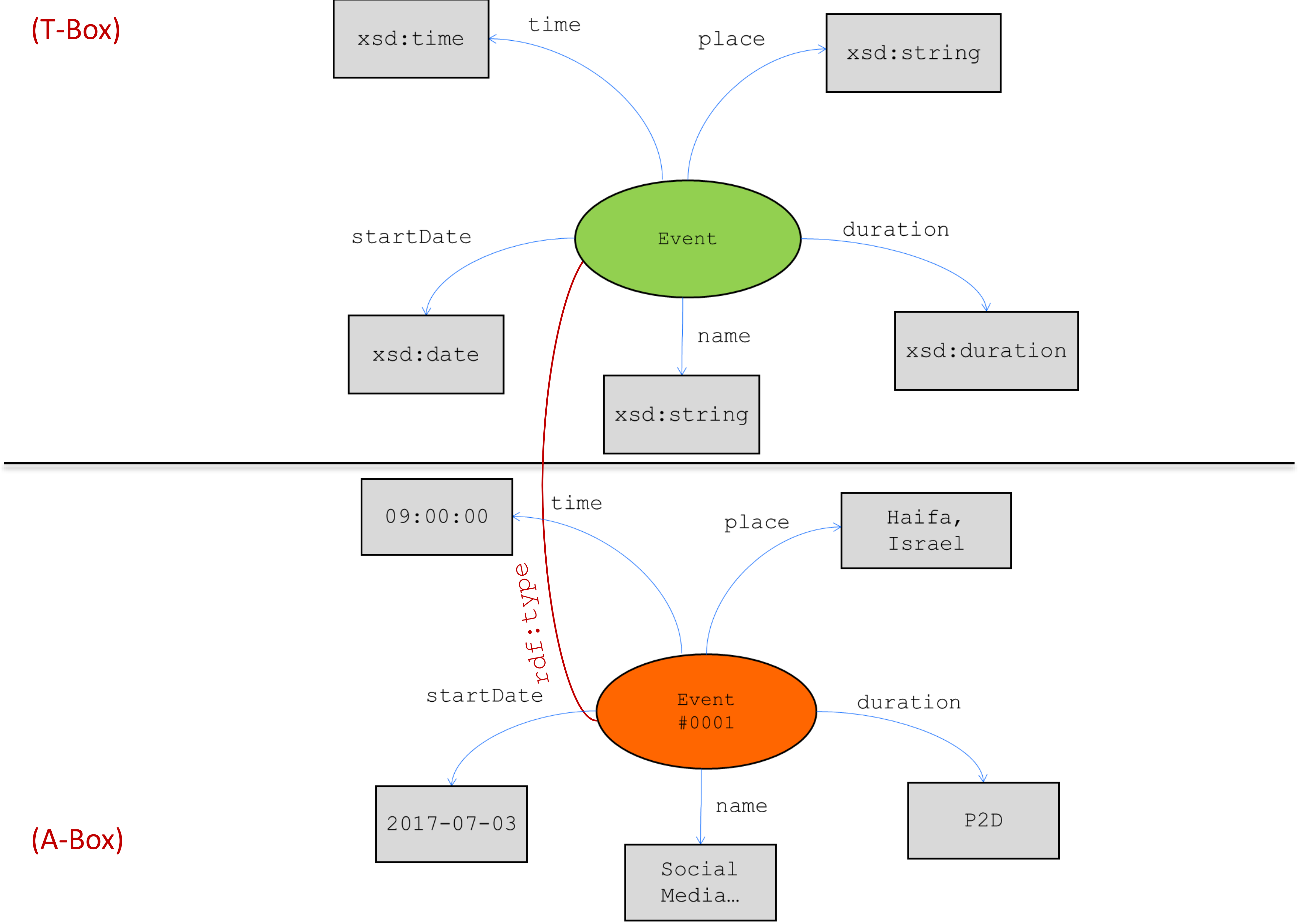
A-Box



Each arc is a statement (assertion) about #0001 instance of the Event Class

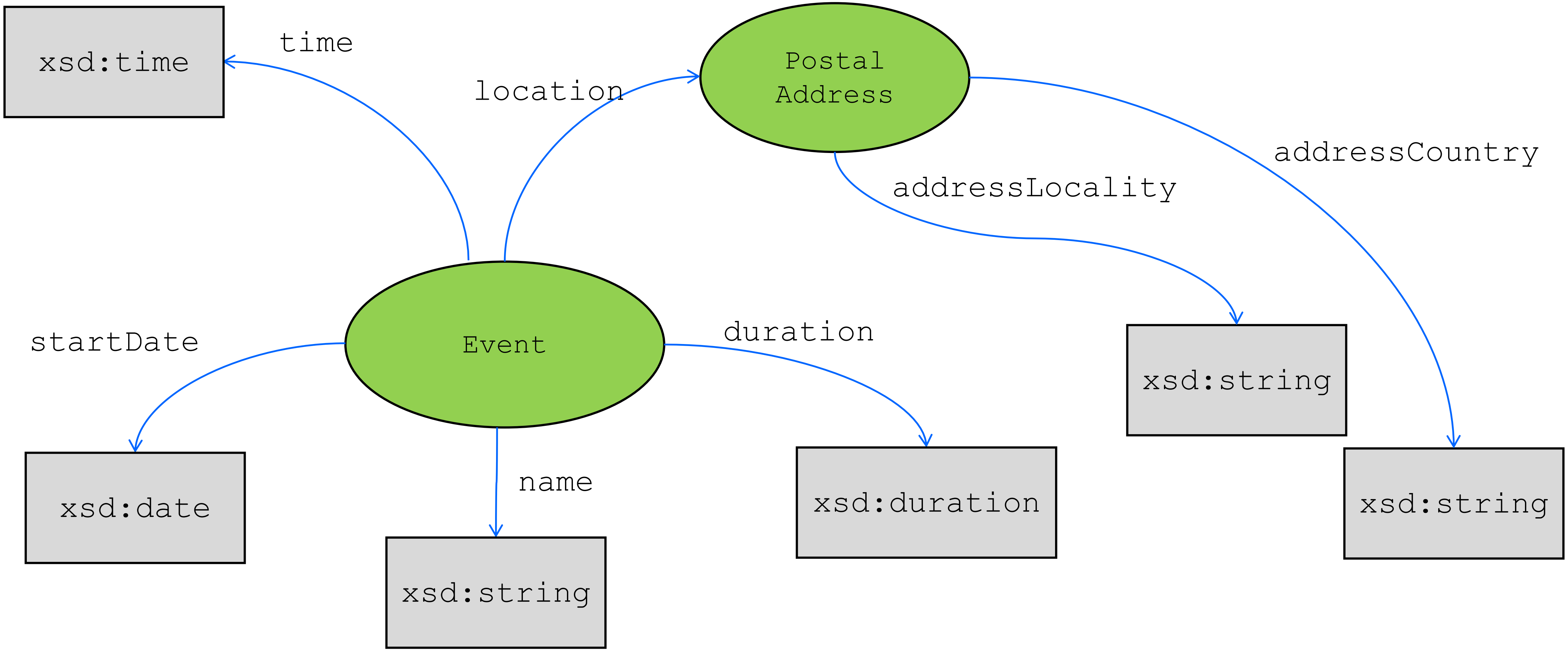
```
<"#0001", name,       "Social Media…">
<"#0001", startDate,  "2017-07-03">
<"#0002", name,       "Internet of Things">
………………
```

Example

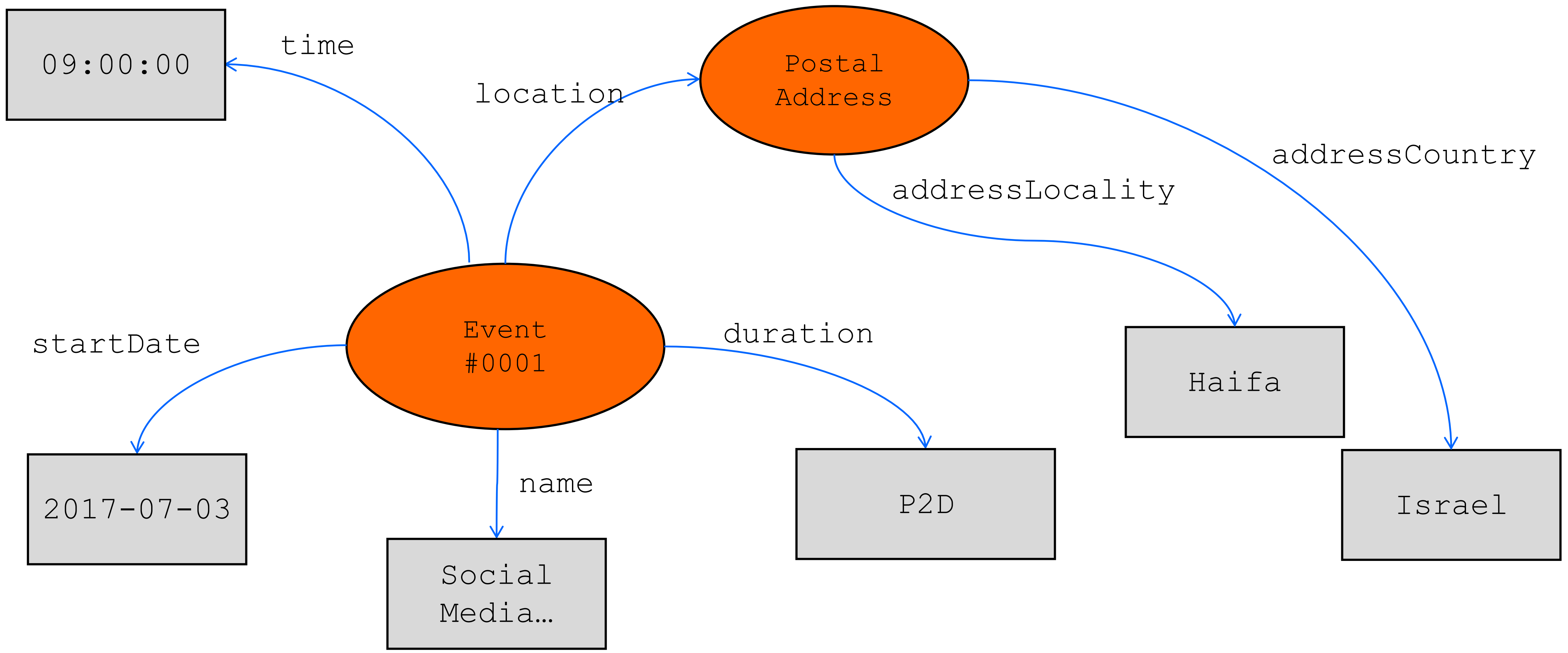


(T-Box)
xsd:time
time
place
xsd:string
Event
startDate
duration
xsd:date
name
xsd:duration
xsd:string
09:00:00
time
place
Haifa, Israel
rdf:type
startDate
Event #0001
duration
2017-07-03
name
P2D
Social Media…
(A-Box)

Let's add relations



An instance of our model



RDF Schema: adding semantics to RDF

- **RDF** is a domain-independent language: it provides mechanisms for making assertions

- **RDF Schema** provides a data-modelling vocabulary for RDF data
  - it is a semantic extension of RDF
  - It provides a mechanism for describing groups of related resources and the relationships between these resources

- RDFS is a simple vocabulary/ontology modeling language
  - A richer vocabulary/ontology language is OWL

Source: https://www.w3.org/TR/rdf-schema/

RDF Schema

RDFS allows to express:

- classs (*rdfs:Class)* and properties (*rdfs:Property),*
- class hierarchies and Inheritance (*rdfs:subClassOf*) ,
- property hierarchies (*rdfs:subPropertyOf*) ,
- domain (*rdfs:domain*) and range (*rdfs:range*) restrictions for properties

classs and instances: semantic annotation

Individual objects that belong to a class are referred to as instances of that class (*rdf:type*).

Source: https://www.w3.org/TR/rdf-schema/

Web Ontology Language (OWL)

- The W3C OWL language is designed to represent richer knowledge than RDF/S.
- RDF/S is useful to describe the concepts and their relationships, but does not solve all possible requirements
- OWL provides constructs for stating logical expressions such as: *Equality, Property Characteristics, Property Restrictions, Restricted Cardinality, Annotation Properties, Versioning*, etc.
- Moreover it allows to describe data in terms of set operations (such as UnionOf)

Source: https://www.w3.org/OWL/

Web Ontology Language (OWL)

- The current version, OWL 2 adds further features:
  - property chains;
  - richer datatypes, data ranges;
  - qualified cardinality restrictions;
  - asymmetric, reflexive, and disjoint properties;
  - enhanced annotation capabilities

Source: https://www.w3.org/OWL/

Identification of resources

- In RDF everything is a resource (also classs and properties are resources)
- Each resource should be identified using a unique ID
- The ID used to identify resources is the URI/IRI

URI (Uniform Resource Identifier) identifies whatever kind of resources. It is based on the URL structure, but is used as an ID.

URL (Uniform Resource Locator) is a specific type of URI which both identifies a resource and localizes it.

IRI (Internationalized Resource Identifier) while URIs are limited to a subset of the ASCII character set, IRIs may contain characters from Unicode.

fragments can be appended to a URI/URL/IRI

- For example, the property "hasName" can be defined as:

  URI: http://www.dibris.unige.it/ontology.owl#hasName

fragments

Using URIs containing hash fragments

e.g. http://example.com/things/thing1#realthing

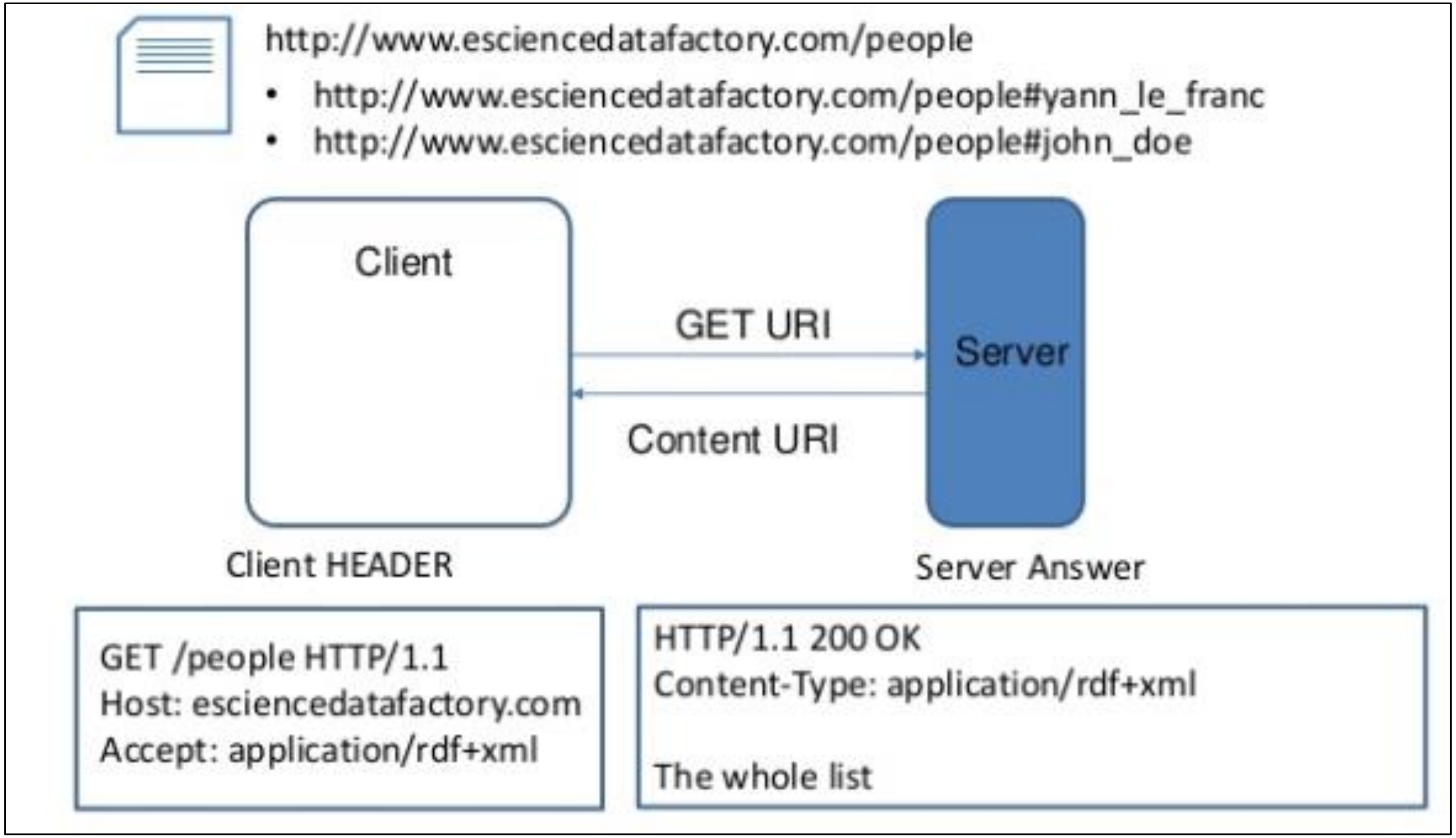
meaning the server can return a document at

http://example.com/things/thing1

that describes

http://example.com/things/thing1#realthing.

fragments


http://www.esciencedatafactory.com/people
• http://www.esciencedatafactory.com/people#yann_le_franc
• http://www.esciencedatafactory.com/people#john_doe
Client
GET URI
Server
Content URI
Client HEADER
Server Answer
GET /people HTTP/1.1
Host: esciencedatafactory.com
Accept: application/rdf+xml
HTTP/1.1 200 OK
Content-Type: application/rdf+xml
The whole list

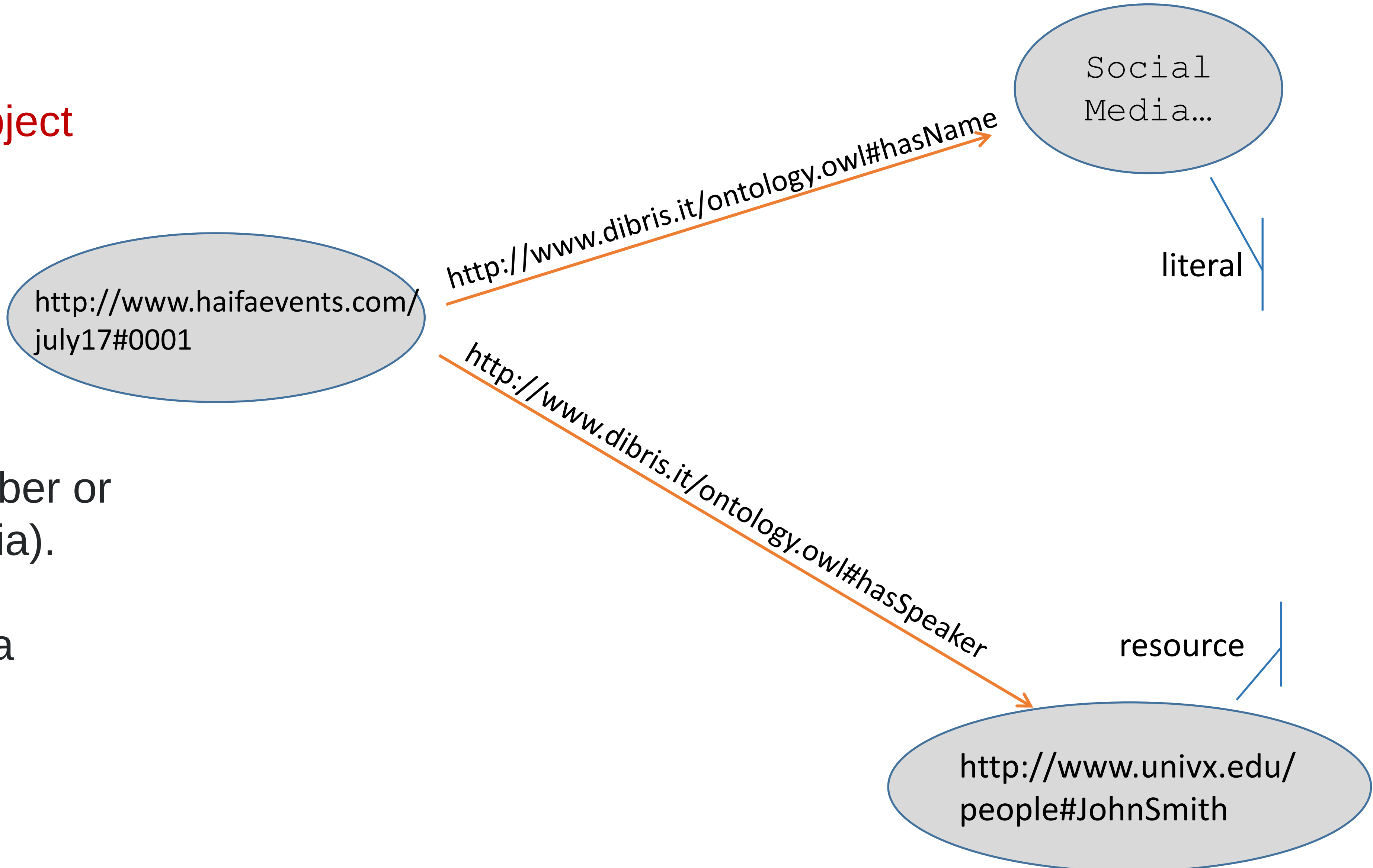
RDF graph

URIs are used for the subject and the predicate.

The object can be

- a literal such as a number or string (e.g. Social Media).

- a URI which specifies a resource.



RDF graph serialization

RDF graphs can be serialized

- into sequences of triples
- with a number of different formats including:
  - RDF/XML
  - N3
  - N-Triples
  - JSON-LD
  - RDFa
  - Microdata

```
<“#0001”, name,        “Social Media…”>
<“#0001”, startDate,   “2017-07-03”>
<“#0001”, time,        “09:00:00”>
<“#0001”, duration,    “P2D”>
……………
```

- Several conversion tools
  - E.g. http://www.easyrdf.org/converter
- Also available as APIs and Rest APIs
  - E.g. http://rdf-translator.appspot.com/

RDF/XML

```
<?xml version="1.0"?>

<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:exo="http://www.exampleOntol.org#">

      <rdf:Description rdf:about="Event#0001"
           rdf:type="http://www.exampleOntol.org/Event">
            <exo:name>Social Media and Data Science</exo:name>
            <exo:startDate>2017-07-03</exo:startDate>
            <exo:duration>P2D</exo:duration>
            <exo:time>09:00:00</exo:time>
            <exo:place>Haifa Israel</exo:place>
      </rdf:Description>
</rdf:RDF>
```

JSON-LD

```
{
  "@context": {
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "exo": "http://www.exampleOntol.org#">
  },
  "@id": "Event#0001",
  "@type": "http://www.exampleOntol.org/Event",
  "exo:name": "Social Media and Data Science",
  "exo:startDate": "2017-07-03",
  "exo:duration": "P2D",
  "exo:time": "09:00:00",
  "exo:place": "Haifa Israel"
}
```

Microdata (HTML5)

```
<div>
  <div itemscope itemtype="http://www.exampleOntol.org/Measurement/Event"
  itemid="Event#0001" >
    <meta itemprop="name" content="Social Media and Data Science" />
    <meta itemprop="startDate" content="2017-07-03" />
    <meta itemprop="duration" content="P2D" />
    <meta itemprop="time" content="09:00:00" />
    <meta itemprop="place" content="Haifa Israel" />
  </div>
</div>
```

Owl serialization

- Also for OWL, a concrete syntax, that is a serialization, is needed in order to store OWL 2 ontologies and to exchange them among tools and applications.

- The primary exchange syntax for OWL 2 is RDF/XML [RDF Syntax]; this is indeed the only syntax that must be supported by all OWL 2 tools.

Namespace

Namespace = a list of names (to be used for various purposes, e.g. within vocabularies and ontologies), identified by URIs/IRIs

E.g.:

`http://www.w3.org/2001/XMLSchema`

`http://www.biblio.org/voc`

If I want to use multiple namespaces in a document, it is useful to associate a prefix to the URI/IRI:

prefix = "*namespaceURI*"

es.

```
xs="http://www.w3.org/2001/XMLSchema"
biblio="http://www.biblio.org/voc”
dbpedia="http://dbpedia.org/resource/"
schema="https://schema.org/"
```

Namespace

```
biblio="http://www.biblio.org/voc”
schema="https://schema.org/"
```

Once the URI and its prefix have been declared, I can refer to the vocabulary terms using the contracted form

eg. `biblio:author`

`schema:Person`

This contracted form of URI:resource is called CURIE (Compact URI)

Namespace

It is interpreted as follows:

`biblio:author`

author is a resource within the biblio namespace

that has URI http://www.biblio.org/voc

`schema:Person`

Person is a resource within the schema namespace

that has URI https://schema.org

VOCABULARIES

Popular vocabularies are available at
http://www.w3.org/2011/rdfa-context/rdfa-1.1 (old URL)
https://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/CommonVocabularies

Prefix	URI	Description
cc	http://creativecommons.org/ns#	ccREL
ctag	http://commontag.org/ns#	Common Tag Ontology
dc	http://purl.org/dc/terms/	Dublin Core Metadata Terms
dcterms	http://purl.org/dc/terms/	Dublin Core Metadata Terms
dc11	http://purl.org/dc/elements/1.1/	Dublin Core Metadata Element Set, Version 1.1
foaf	http://xmlns.com/foaf/0.1/	FOAF
gr	http://purl.org/goodrelations/v1#	GoodRelations Ontology
ical	http://www.w3.org/2002/12/cal/icaltzd#	iCalendar terms in RDF
og	http://ogp.me/ns#	Facebook's Open Graph protocol
rev	http://purl.org/stuff/rev#	RDF Review Vocabulary
sioc	http://rdfs.org/sioc/ns#	SIOC Core Ontology
v	http://rdf.data-vocabulary.org/#	Google Rich Snippets' Vocabularies
vcard	http://www.w3.org/2006/vcard/ns#	vCard in RDF
schema	http://schema.org/	The Schema.org vocabulary

VOCABULARIES

Prefix	URI	Description
cc	http://creativecommons.org/ns#	ccREL
ctag	http://commontag.org/ns#	Common Tag Ontology
dc	http://purl.org/dc/terms/	Dublin Core Metadata Terms
dcterms	http://purl.org/dc/terms/	Dublin Core Metadata Terms
dc11	http://purl.org/dc/elements/1.1/	Dublin Core Metadata Element Set, Version 1.1
foaf	http://xmlns.com/foaf/0.1/	FOAF
gr	http://purl.org/goodrelations/v1#	GoodRelations Ontology
ical	http://www.w3.org/2002/12/cal/icaltzd#	iCalendar terms in RDF
og	http://ogp.me/ns#	Facebook's Open Graph protocol
rev	http://purl.org/stuff/rev#	RDF Review Vocabulary
sioc	http://rdfs.org/sioc/ns#	SIOC Core Ontology
v	http://rdf.data-vocabulary.org/#	Google Rich Snippets' Vocabularies
vcard	http://www.w3.org/2006/vcard/ns#	vCard in RDF
schema	http://schema.org/	The Schema.org vocabulary

Schema.org

- Schema.org is a vocabulary that provides webmasters with a wide set of terms they can use to mark up their pages in ways that can be understood by the major search engines.

- Markup vocabularies include: Person, Event, Organization, Product, Review, Review-aggregate, Breadcrumb, Offer, Offer-aggregate, etc.

- Major search engine operators like Google, Microsoft and Yahoo! rely on this markup to improve search results pages.

  "Authors are encouraged to re-use existing vocabularies, as this makes content re-use easier"

Property	Expected Type	Description
Properties from Person		
additionalName	Text	An additional name for a Person, can be used for a middle name.
address	PostalAddress or Text	Physical address of the item.
affiliation	Organization	An organization that this person is affiliated with. For example, a school/university, a club, or a team.
alumniOf	EducationalOrganization or Organization	An organization that the person is an alumni of. Inverse property: alumni.
award	Text	An award won by or for this item. Supersedes awards.
birthDate	Date	Date of birth.
birthPlace	Place	The place where the person was born.
brand	Brand or Organization	The brand(s) associated with a product or service, or the brand(s) maintained by an organization or business person.
children	Person	A child of the person.
colleague	Person or URL	A colleague of the person. Supersedes colleagues.

Schema.org

- Until version 7, the canonical machine representation of schema.org was in RDFa

- Starting from version 8, machine-readable representations of schemas are available at this page: https://schema.org/docs/developers.html

- Basically, it includes:
  - set of **types**, arranged in a **multiple inheritance hierarchy**
  - a set of **properties**

- It is used by over 10 million websites

source: http://schema.org

VOCABULARIES

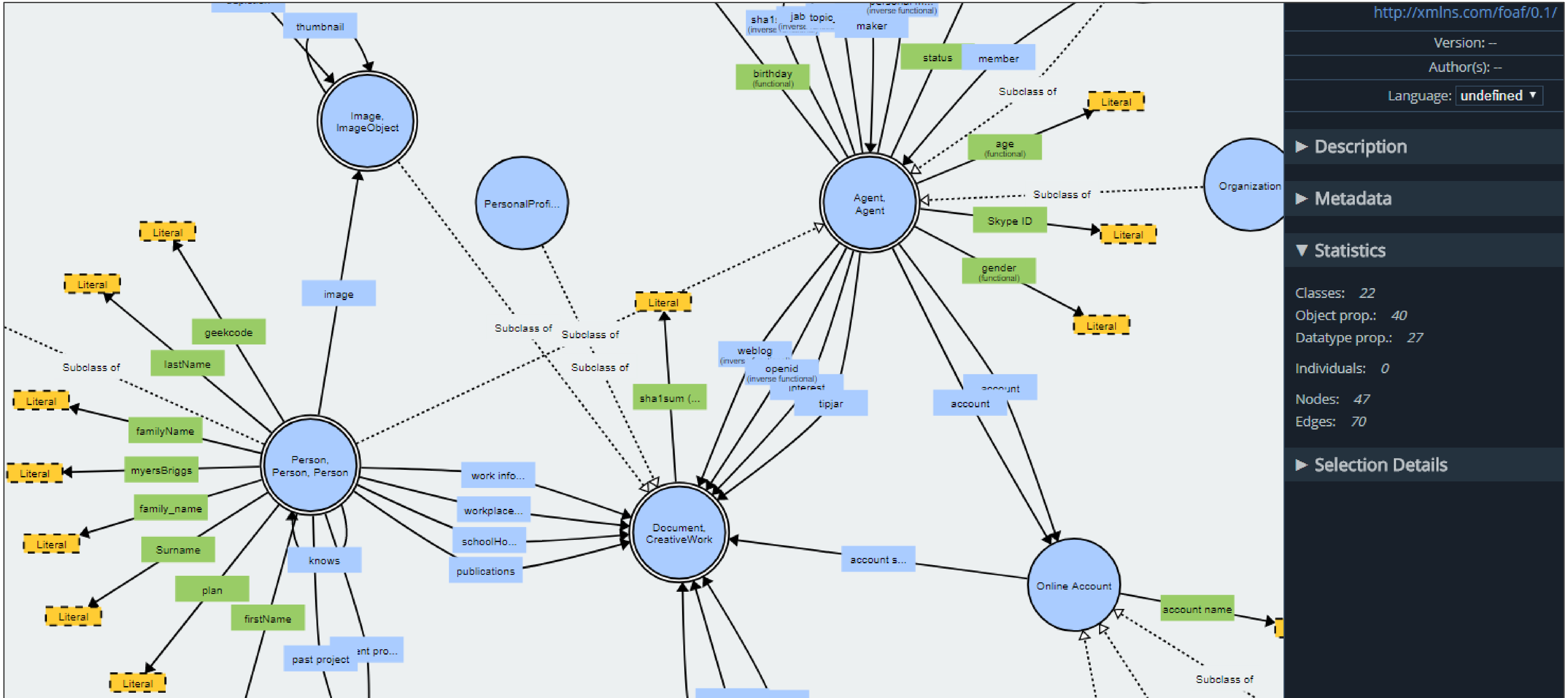
Prefix	URI	Description
cc	http://creativecommons.org/ns#	ccREL
ctag	http://commontag.org/ns#	Common Tag Ontology
dc	http://purl.org/dc/terms/	Dublin Core Metadata Terms
dcterms	http://purl.org/dc/terms/	Dublin Core Metadata Terms
dc11	http://purl.org/dc/elements/1.1/	Dublin Core Metadata Element Set, Version 1.1
foaf	http://xmlns.com/foaf/0.1/	FOAF
gr	http://purl.org/goodrelations/v1#	GoodRelations Ontology
ical	http://www.w3.org/2002/12/cal/icaltzd#	iCalendar terms in RDF
og	http://ogp.me/ns#	Facebook's Open Graph protocol
rev	http://purl.org/stuff/rev#	RDF Review Vocabulary
sioc	http://rdfs.org/sioc/ns#	SIOC Core Ontology
v	http://rdf.data-vocabulary.org/#	Google Rich Snippets' Vocabularies
vcard	http://www.w3.org/2006/vcard/ns#	vCard in RDF
schema	http://schema.org/	The Schema.org vocabulary

FOAF (Friend of A Friend Ontology)

- FOAF is an RDF/OWL ontology describing people, their activities (e.g., pictures, calendars, weblogs) and their relations to other people and objects
- Widely used in social (semantic) systems
- Properties for user identification (mailbox hash, open ID, etc.)
- Mechanism for autodiscovery

Source: http://xmlns.com/foaf/spec/

FOAF (Friend of A Friend Ontology)



VOCABULARIES

Prefix	URI	Description
cc	http://creativecommons.org/ns#	ccREL
ctag	http://commontag.org/ns#	Common Tag Ontology
dc	http://purl.org/dc/terms/	Dublin Core Metadata Terms
dcterms	http://purl.org/dc/terms/	Dublin Core Metadata Terms
dc11	http://purl.org/dc/elements/1.1/	Dublin Core Metadata Element Set, Version 1.1
foaf	http://xmlns.com/foaf/0.1/	FOAF
gr	http://purl.org/goodrelations/v1#	GoodRelations Ontology
ical	http://www.w3.org/2002/12/cal/icaltzd#	iCalendar terms in RDF
og	http://ogp.me/ns#	Facebook's Open Graph protocol
rev	http://purl.org/stuff/rev#	RDF Review Vocabulary
sioc	http://rdfs.org/sioc/ns#	SIOC Core Ontology
v	http://rdf.data-vocabulary.org/#	Google Rich Snippets' Vocabularies
vcard	http://www.w3.org/2006/vcard/ns#	vCard in RDF
schema	http://schema.org/	The Schema.org vocabulary

The Open Graph Protocol

https://ogp.me/

The Open Graph protocol was originally created at **Facebook**.

The vocabulary defines a set of PROPERTIES that allow to describe an object (such as a web page).

If used to annotate a webpage, it enables the web page to become a rich object in a social graph. For instance, this is used on Facebook to allow any web page to have the same functionality as any other object on Facebook.

The Open Graph Protocol

The RDF Schema can be found at: https://ogp.me/ns/ogp.me.rdf

Prefix og: http://ogp.me/ns#

Main Properties

og:title - The title of your object as it should appear within the graph, e.g., "The Rock".
og:type - The type of your object, e.g., "video.movie". Depending on the type you specify, other properties may also be required.
og:image - An image URL which should represent your object within the graph.
og:url - The canonical URL of your object that will be used as its permanent ID in the graph, e.g., "https://www.imdb.com/title/tt0117500/".
og:locale - The locale these tags are marked up in. Of the format language_TERRITORY. Default is en_US
og:description - A one to two sentence description of your object.
og:site_name - If your object is part of a larger web site, the name which should be displayed for the overall site. e.g., "IMDb".
og:video - A URL to a video file that complements this object.