Semantic Web

Carlo Meghini

Istituto di Scienza e Tecnologie della Informazione
Consiglio Nazionale delle Ricerche – Pisa

2017-18

Today's Lecture: Resource Description Framework

A data model for the Semantic Web

Formal descriptions are one of the basic technologies for the Semantic Web (SW).

Formal descriptions are composed of statements, structured according to a data model.

What would be an adequate data model for the SW?

The SW has adopted a data modelling style that has been developed from the 80s, known as *semantic* or *object-oriented data modelling.*

Limitations of relational representations

We want to design an information system containing typical data about people, such as name, birthdate, birthplace, and the like.

Following a traditional database design approach we would come up very likely with a tabular representation of our data like this:

First Name	Last Name	Birth Place	Birth Date	Social_ID
"Mario"	"Rossi"	"Roma"	01-01-1987	"MRARSS87A01A789Y"
"Marina"	"Verdi"	"Milano"	25-10-1962	"MRNVRD62L65A111I"
...	...	...	...	...

Every column of the table is labelled with an *attribute, e.g.*, **First Name**

Every attribute has a *domain,* that is the set of values that the attribute can take; the domain of **First Name** is the set of alphabetic strings.

In every row, the value of the attribute is drawn from a set that is called the *domain* of the attribure. “Mario" is an alphabetic string, so it can be used as a value for the attribute **First Name** in any row of the table.

Social_ID is a very special attribute, it is called “primary key” because no two different rows can have the same value for the Social_ID.

Relational databases have been invented for managing efficiently large quantities of formatted data such as those found in business applications. They are not optimal for the features of the web:

- the web is distributed, and so we want the SW to be
- the SW needs an easy-to-use model for building descriptions . . .
- . . . that are deeply interconnected, much in the same way web pages are connected to one another.

In other words, we want a “semantic” data model, whose descriptions reflects reality in a direct and intuitive way.

In addition, we want descriptions that are interconnected and distributed.

In this sense, the relational data model poses two problems:

1. it is too machine-oriented for building *semantic* information systems. It describes tables, rows and columns, not reality.
2. it is not "webby" enough: it does not use IRIs, and it does not allow the navigation of information via HTTP.

Can it be extended to cope with these problems? Probably, but the result would be ugly, and would have to be re-implemented.

As such, the relational data model is not a good vehicle for the SW.

We must look into a different type of models, more semantic in nature: the so-called semantic data models.

These models mimic natural language as a modelling style, and therefore are much more intuitive than any other type of model.

Semantic modelling
Objects

Semantic data models aim at representing the world, and they view the world as consisting of *objects* and *relationships* between objects. Such relationships represent *facts.*

Let's see how by looking back at our example about people.

The first problem that we face is how to identify a person.

In the relational example, persons are identified by their Social_ID.

From a mathematical point of view this is a valid choice, because every person has a different Social_ID. However, from a semantic point of view the choice is not good:

- a Social_ID identifies people in the context of the public administration. As such, it not necessarily applies to every context.
- a person is a person, and their Social_ID is one of their properties. It happens to be a functional, injective property, however this is not a good reason to consider a person the same thing as their Social_ID.

In semantic data models identification is realized by assigning a *unique identifier* to every object that the user introduces into the database.

In the web, objects are called *resources* and every resource is identified by a special kind of alphanumeric string, called International Resource Identifier, abbreviated as "IRI".

RDF conforms to the web "way of modelling", so in RDF every object is a resource, identified by an IRI.

Persons are no exception, so in our example every person is identified by an IRI.

Moreover, IRIs of the HTTP protocol are preferable, because they are actionable, making it possible to associate web documents to IRIs and to retrieve these document via the HTTP protocol.

As we can see, RDF is conceived as a semantic data model and at the same time is strongly tied to the web architecture.

Relationships

Next, we have to decide how to represent the information about a person.

We want to represent the fact that a certain person is born in a certain place, for instance that Mario Rossi is born in Roma.

The birth place of Mario Rossi

Mario Rossi 

First Name	Last Name	Birth Place	Birth Date	Social_ID
"Mario"	"Rossi"	"Roma"	01-01-1987	"MRARSS87A01A789Y"
"Marina"	"Verdi"	"Milano"	25-10-1962	"MRNVRD62L65A111I"
...	...	...	...	...

In semantic data modeling, we understand the fact that a certain person is born in a certain place as a *relationship* between two resources:

Relationship: Mario Rossi was born in Roma

Resources: Mario Rossi, Roma

Type of relationship: was born in

Rank of the type: 2

Relationships can be of various types, each type coming with a rank:

Other examples:

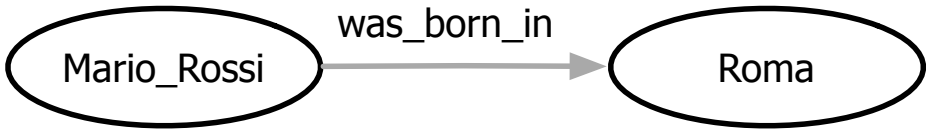
Relationship:	Pisa is in Toscana
Resources:	Pisa, Toscana
Type of relationship:	is (geographically) in
Rank of the type:	2

Relationship:	Romeo gives a rose to Juliet
Resources:	Romeo, rose, Juliet
Type of relationship:	gives
Rank of the type:	3

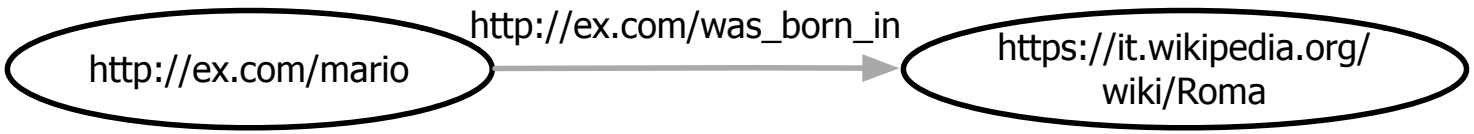
Semantic data modelling considers *exclusively* relationship types of rank 2, based on the assumption that they are the most common types, and that relationships of types with higher ranks can be reduced to equivalent relationships of types of rank 2.

In semantic modelling, a relationship type is represented by a *property* and relationships are represented as triples including the resources and the property.

Since there has never been a standardization of semantic data models, there is no unique format for representing these triples. Perhaps the most popular is the graphic format:



In RDF, properties are resources, so they are represented by IRIs like the connected objects. So in RDF the relationship between Mario Rossi and his birthplace is graphically represented as:



Graphics is not a good notation for machines, so we will see later that RDF has a concrete textual notation.

How does one choose which IRIs to use for their data?

If I want the knowledge on my web page to be accessed and used by a certain community, I must:

- use the language spoken by that community, and
- use the idioms that are well-understood in that community.

. If I aim for the widest possible community, I must

- pick the widest spoken language, possibly more than one, and
- avoid idiosyncratic expressions that would not be understood by most of the people.

The situation on the SW is exactly the same: data are created to be shared and understood by a certain community, so the IRIs for objects and for properties must be chosen accordingly.

Ontologies have a crucial role.

Let us now consider the birthdate, and how to represent the fact that Mario Rossi's birthdate is 01-01-1987. Following the same approach:

Relationship:	Mario Rossi was born on 01-01-1987
Resources:	Mario Rossi, 01-01-1987
Type of relationship:	was born on
Rank of relationship:	2

How to represent 01-01-1987?

Dates are abstract resources, so we could use a IRI for the date.

However, dates come from a fixed set that has been standardized, and there in no special fact that we may want to represent about a date. We only need to use a date as the value of a certain property.

The same applies to alphanumeric strings, numbers, months, *etc.*

For this reason RDF does not use IRIs for these kinds of objects, but a special type of identifiers called *literals.* "01-01-1987"

Literals

A literal consists of two or three elements:

1. a *lexical form*, a Unicode string
2. a *datatype IRI*, that is an IRI identifying a datatype that determines how the lexical form maps to a literal value
3. if and only if the datatype IRI is `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString` there is a non-empty language tag as defined by [BCP47] . In this case, the literal is a *language-tagged string.*

Every RDF serialization formats provides a notation for the combination of the two (or three) elements.

Notation for literals

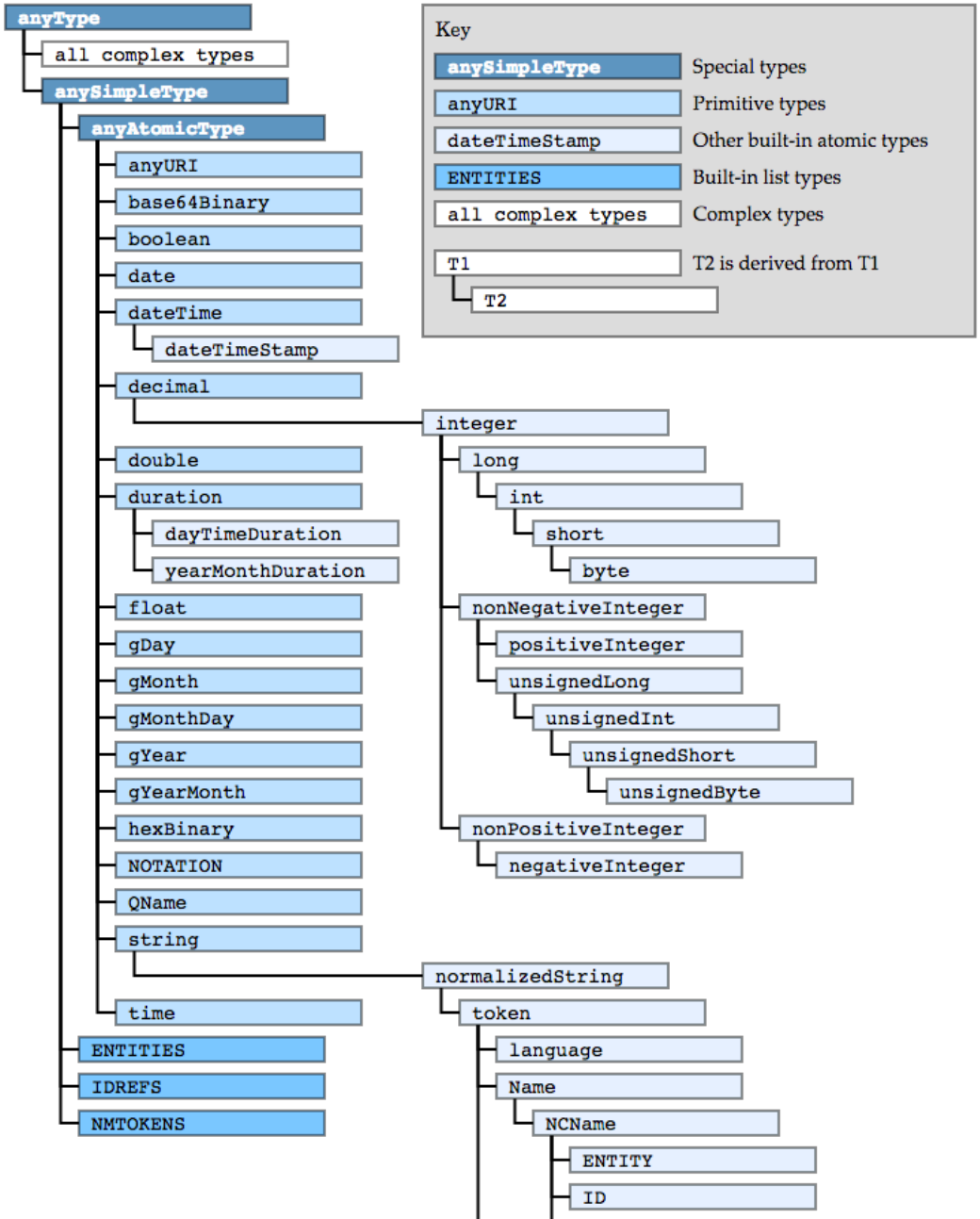
Examples in the Turtle notation:

[xsd is a prefix for http://www.w3.org/2001/XMLSchema#]

- "1"^^xsd:integer
- "3.14"^^xsd:decimal
- "true"^^xsd:boolean
- "2001-10-26T21:32:52"^^xsd:dateTime
- "carlo"^^xsd:string
- "carlo" (simple literal, xsd:string omitted)
- "carlo"@it (language-tagged string with xsd:langString omitted)

Every RDF implementation comes with a set of *recognized* datatypes, which are used to determine *if* a literal has a value and *what* is the value.

The implementation of RDF that we will use recognizes all common built-in XML datatypes, that is all those defined in the XML Schema specifications.


anyType
all complex types
anySimpleType
anyAtomicType
anyURI
base64Binary
boolean
date
dateTime
dateTimeStamp
decimal
integer
long
int
short
byte
nonNegativeInteger
positiveInteger
unsignedLong
unsignedInt
unsignedShort
unsignedByte
nonPositiveInteger
negativeInteger
double
duration
dayTimeDuration
yearMonthDuration
float
gDay
gMonth
gMonthDay
gYear
gYearMonth
hexBinary
NOTATION
QName
string
normalizedString
token
language
Name
NCName
ENTITY
ID
time
ENTITIES
IDREFS
NMTOKENS
Key
anySimpleType
Special types
anyURI
Primitive types
dateTimeStamp
Other built-in atomic types
ENTITIES
Built-in list types
all complex types
Complex types
T1
T2 is derived from T1
T2

Each such datatype can be uniquely addressed via a IRI Reference constructed as follows:

- the base IRI is the IRI of the XML Schema namespace, http://www.w3.org/2001/XMLSchema
- the fragment identifier is the name of the datatype

For example, the IRI of the int datatype of XML Schema is:

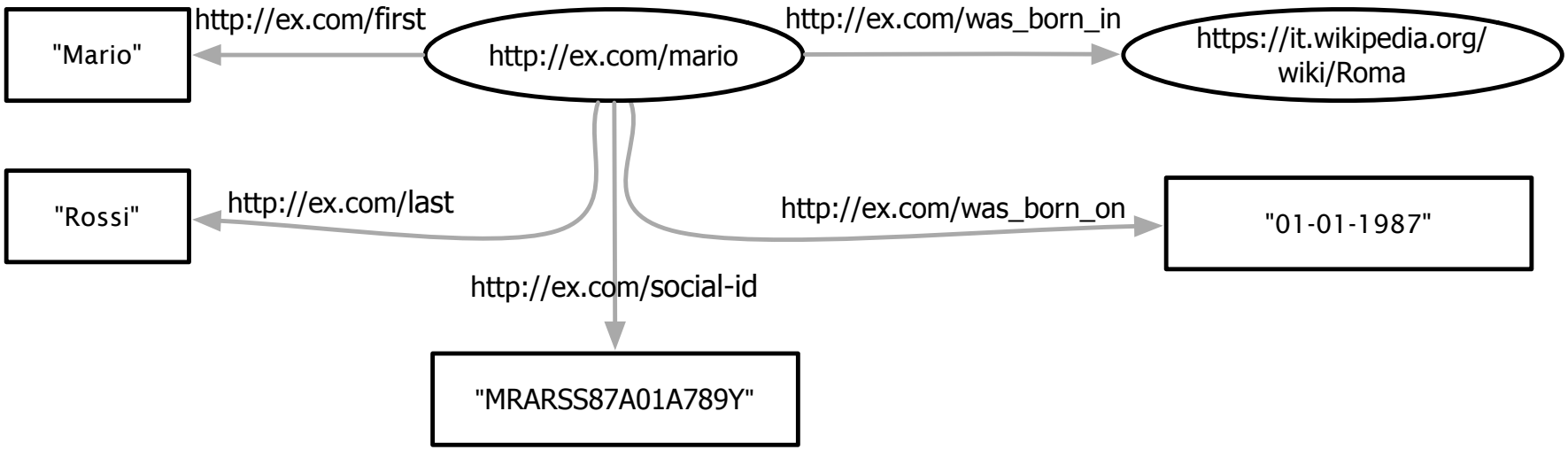
http://www.w3.org/2001/XMLSchema#int

In addition,

- HTML content can be used in RDF as a literal whose datatype is http://www.w3.org/1999/02/22-rdf-syntax-ns#HTML.
- XML content can be used in RDF as a literal whose datatype is http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral.

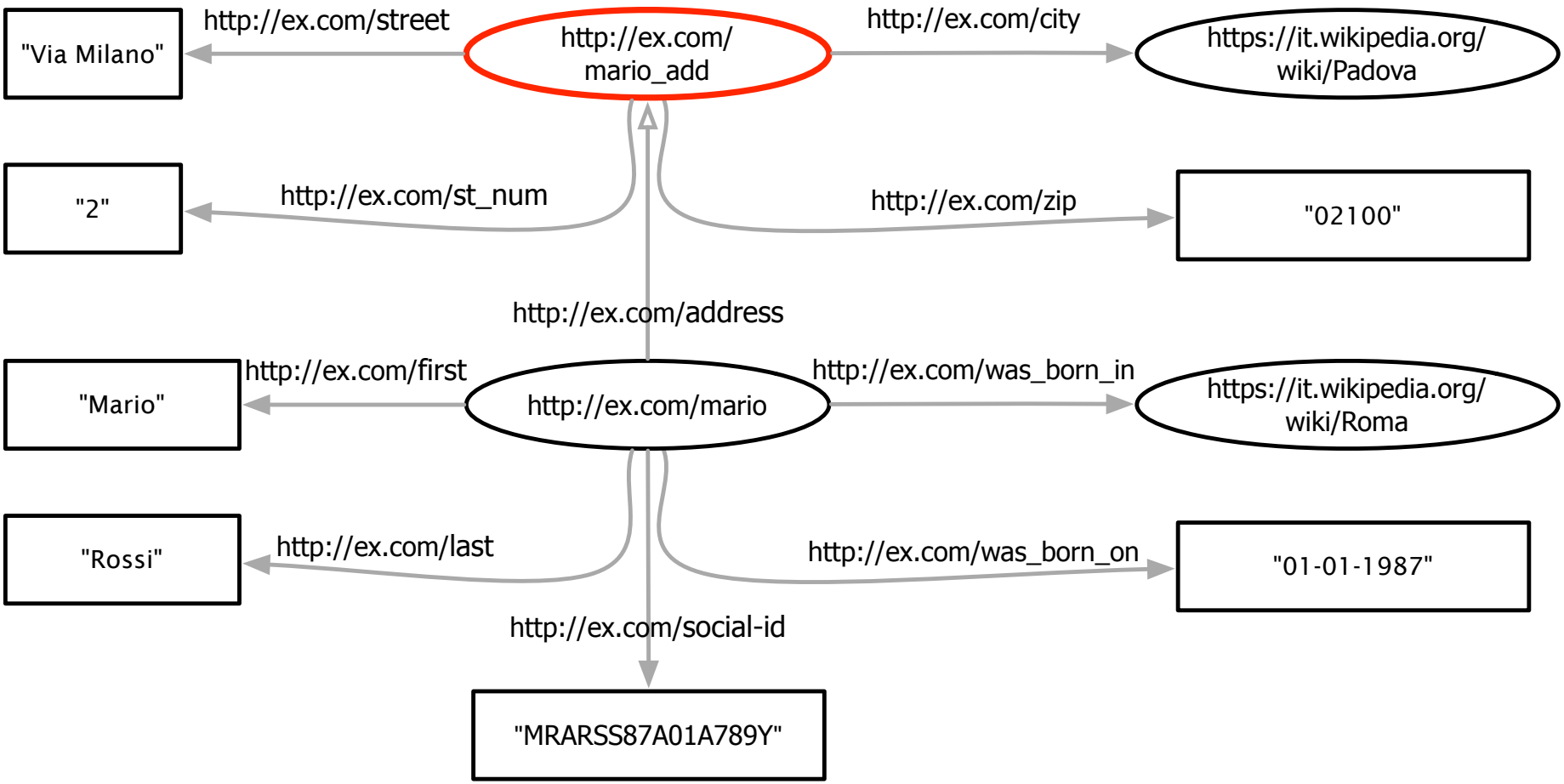
Semantic Networks

First Name	Last Name	Birth Place	Birth Date	Social_ID
"Mario"	"Rossi"	"Roma"	01-01-1987	"MRARSS87A01A789Y"



The semantic representation is called "Semantic Network" and is adopted as a modelling style by RDF.

Now we can add Mario Rossi's address to our network.


"Via Milano"
http://ex.com/street
http://ex.com/
mario_add
http://ex.com/city
https://it.wikipedia.org/
wiki/Padova
"2"
http://ex.com/st_num
http://ex.com/zip
"02100"
http://ex.com/address
"Mario"
http://ex.com/first
http://ex.com/mario
http://ex.com/was_born_in
https://it.wikipedia.org/
wiki/Roma
"Rossi"
http://ex.com/last
http://ex.com/was_born_on
"01-01-1987"
http://ex.com/social-id
"MRARSS87A01A789Y"

Comparing the semantic representation with the relational representation:

- Every (non-trivial) resource is identified by a IRI, so that we do not confuse the resource with one of its property.
- HTTP IRIs are actionable, so one can "click on" each of them and see what it represents. Humans will see HTML pages, artificial agents will obtain semantic representations. Data are linked between them the same way web pages are.
- The network is naturally distributed, *e.g.*, the description of Rome will be stored in the Wikipedia database, and can be navigated using the same mechanism that humans use to navigate the web.
- The structure of the data is reduced to a minimum, so "integrating" the data in two networks is a lot simpler than integrating two relational tables.

Next, we will see a data model for RDF, based on these principles.

Statements

The basic syntactic construct of RDF is the *triple,* that is an expression composed of three elements.

There are two kinds of triples: ground and non-ground triples.

A *ground RDF triple* consists of:

- the subject, which is an IRI
- the predicate, which is an IRI
- the object, which is an IRI or a literal

An RDF triple asserts the statement that some relationship holds between the resources denoted by the subject and object.

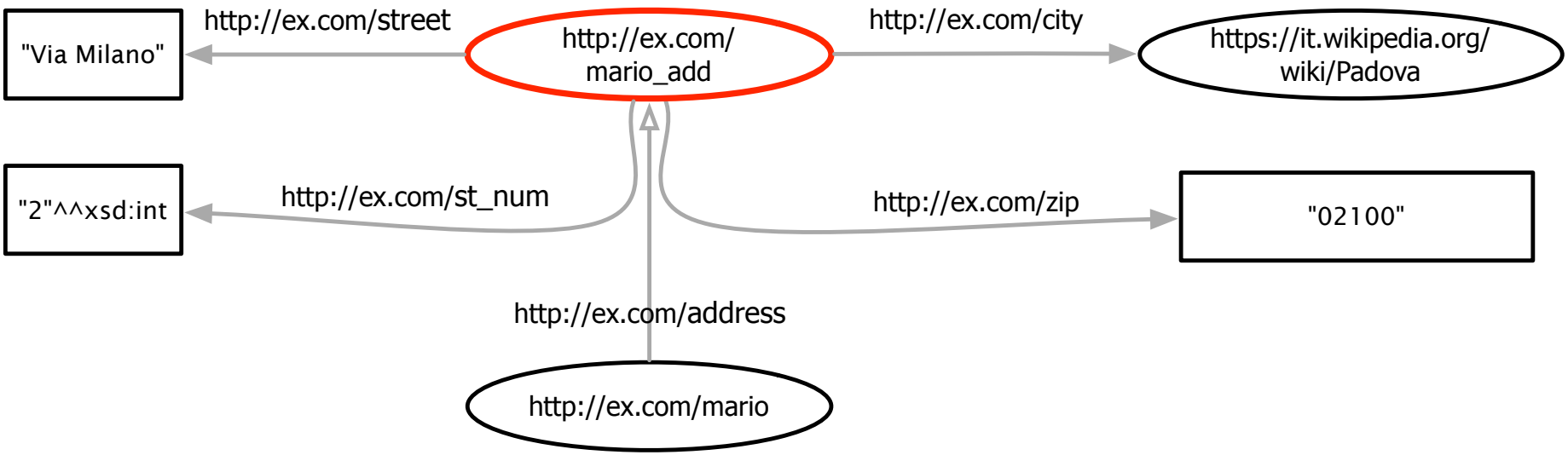
The type of the relationship is indicated by the predicate.

The predicate itself is an IRI and denotes a property, that is, a resource that can be thought of as a binary relation.

Blank Nodes

Sometimes, it is useful to represent knowledge about resources that are not so important as to require a specific IRI for them.

- *e.g.*, Mario Rossi's address

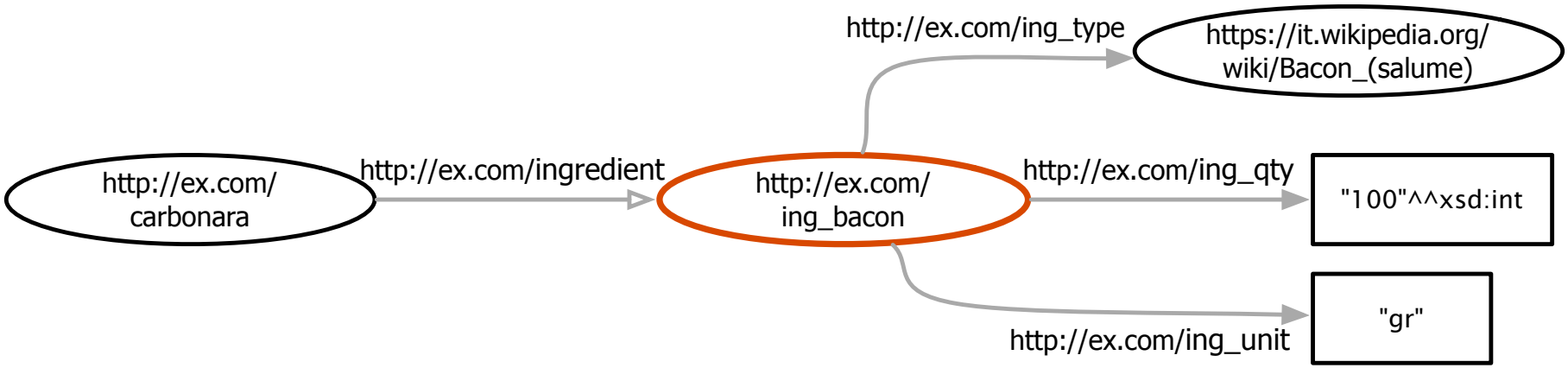


Another common case is when we have a property with rank higher than 2, *e.g.*,

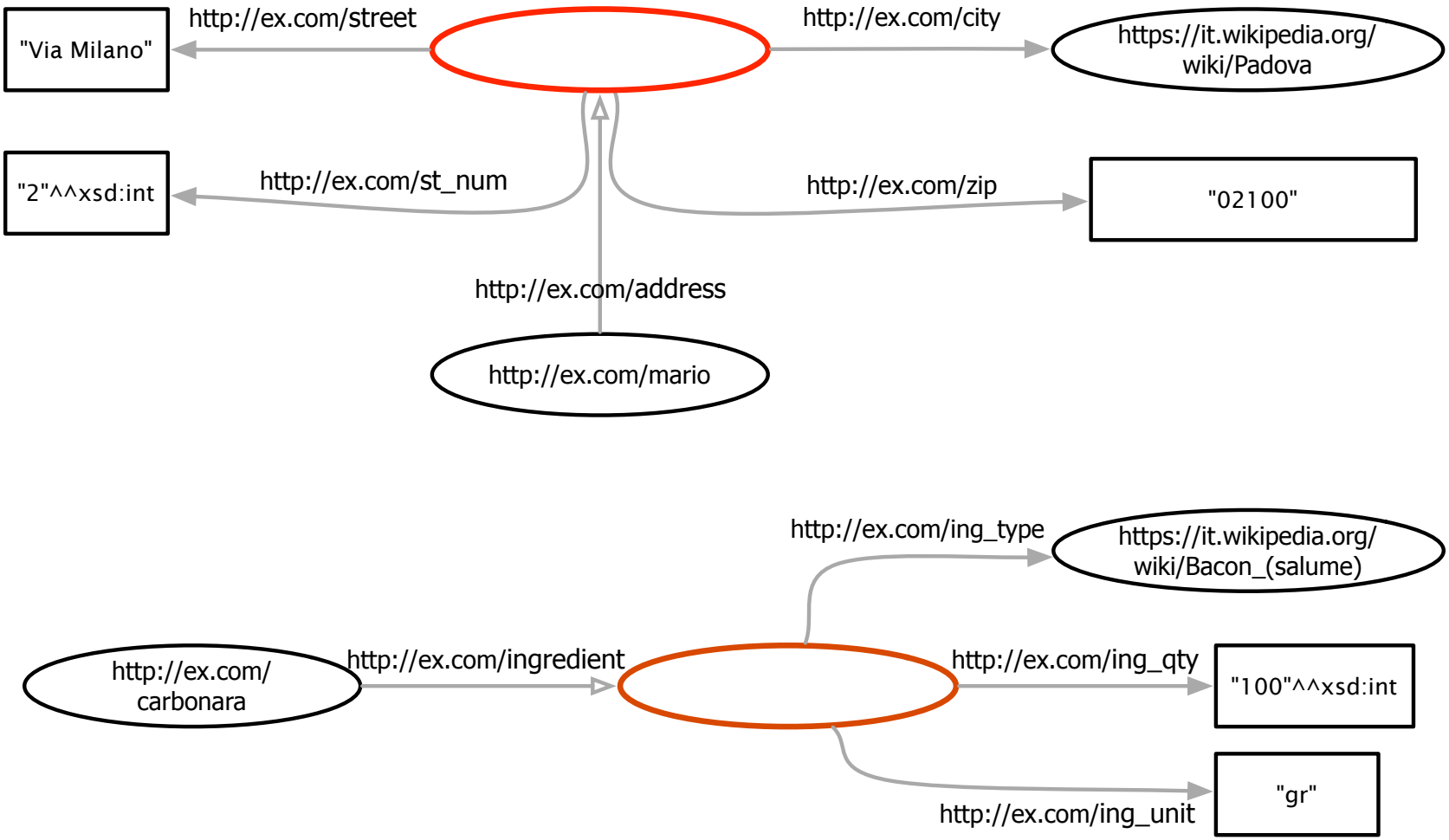
- Carbonara for 4 persons needs 100 grams of bacon

The property that links a recipy to one of its ingredients has rank 4, since it involves recipy, ingredient, value, measure unit.

To represent this information in a semantic network, we need to introduce a resource that does not really represent something relevant in our application domain



In these cases, RDF allows one to use *blank nodes* to denote these resources.



Blank nodes are abstract objects, disjoint from IRIs and literals.

Unlike IRIs and literals, blank nodes do not identify specific resources.

Statements involving blank nodes say that something with the given relationships exists, without explicitly naming it.

Blank nodes make knowledge vague and this vagueness imposes a computational price.

Blank nodes can be used either as subject or as object in a triple, but not as predicate.

A triple involving a blank node is said to be *non-ground.*

In its most general form, an RDF triple consists of:

- the subject, which is an IRI or a blank node
- the predicate, which is an IRI
- the object, which is an IRI, a literal or a blank node

IRIs, literals and blank nodes are collectively known as RDF terms.

RDF: A web-based language for semantic networks

RDF term:

- Literals: "*lexical-form*"^^*datatype* or "*lexical-form*"@*language-tag*
- IRIs, possibly HTTP(S)
- Blank nodes: abstract objects, disjoint from IRIs and literals.

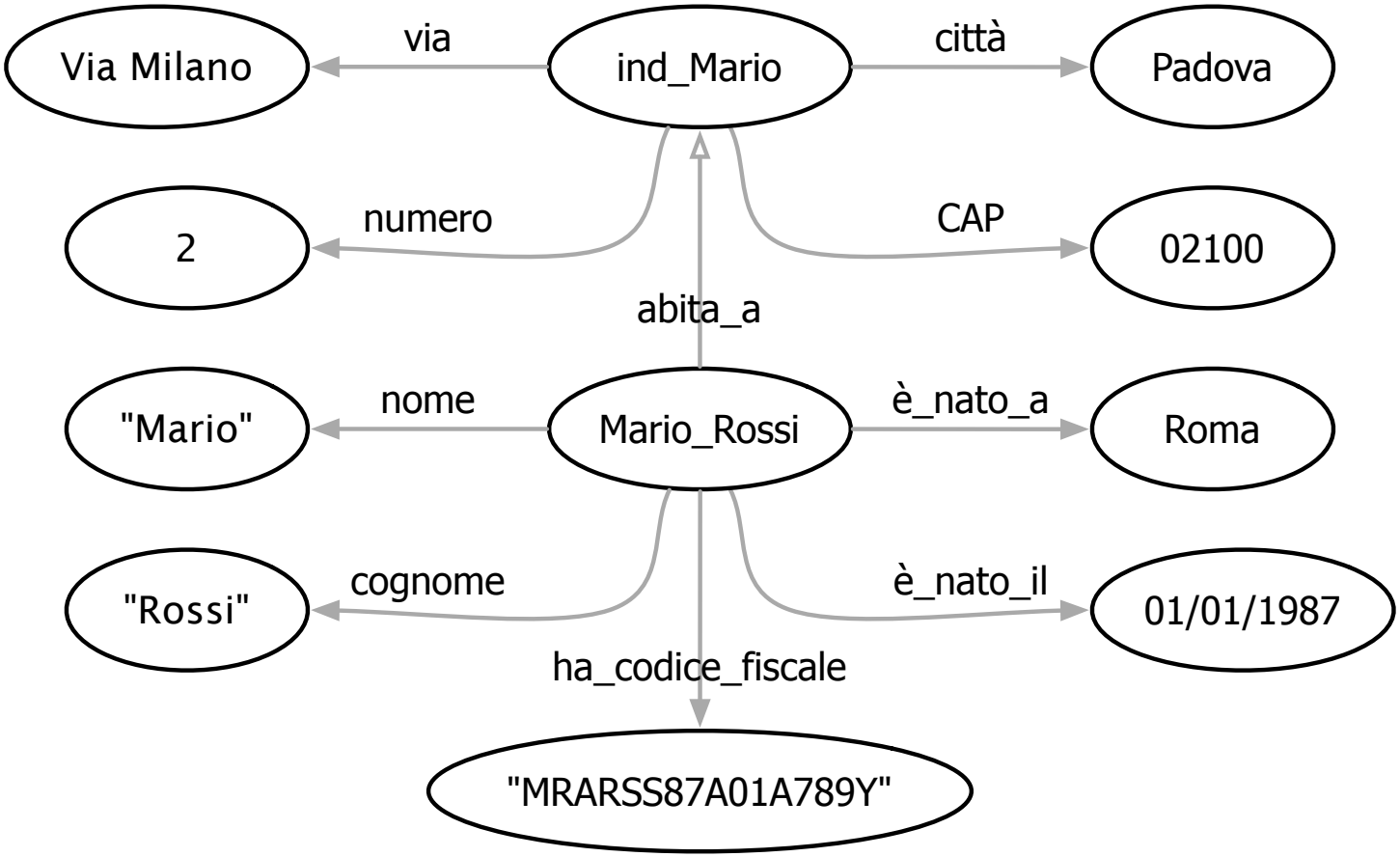
RDF triple (s p o):

- s, the *subject,* an IRI or a blank node
- p, the *predicate,* an IRI
- o, the *object,* an IRI, a literal or a blank node

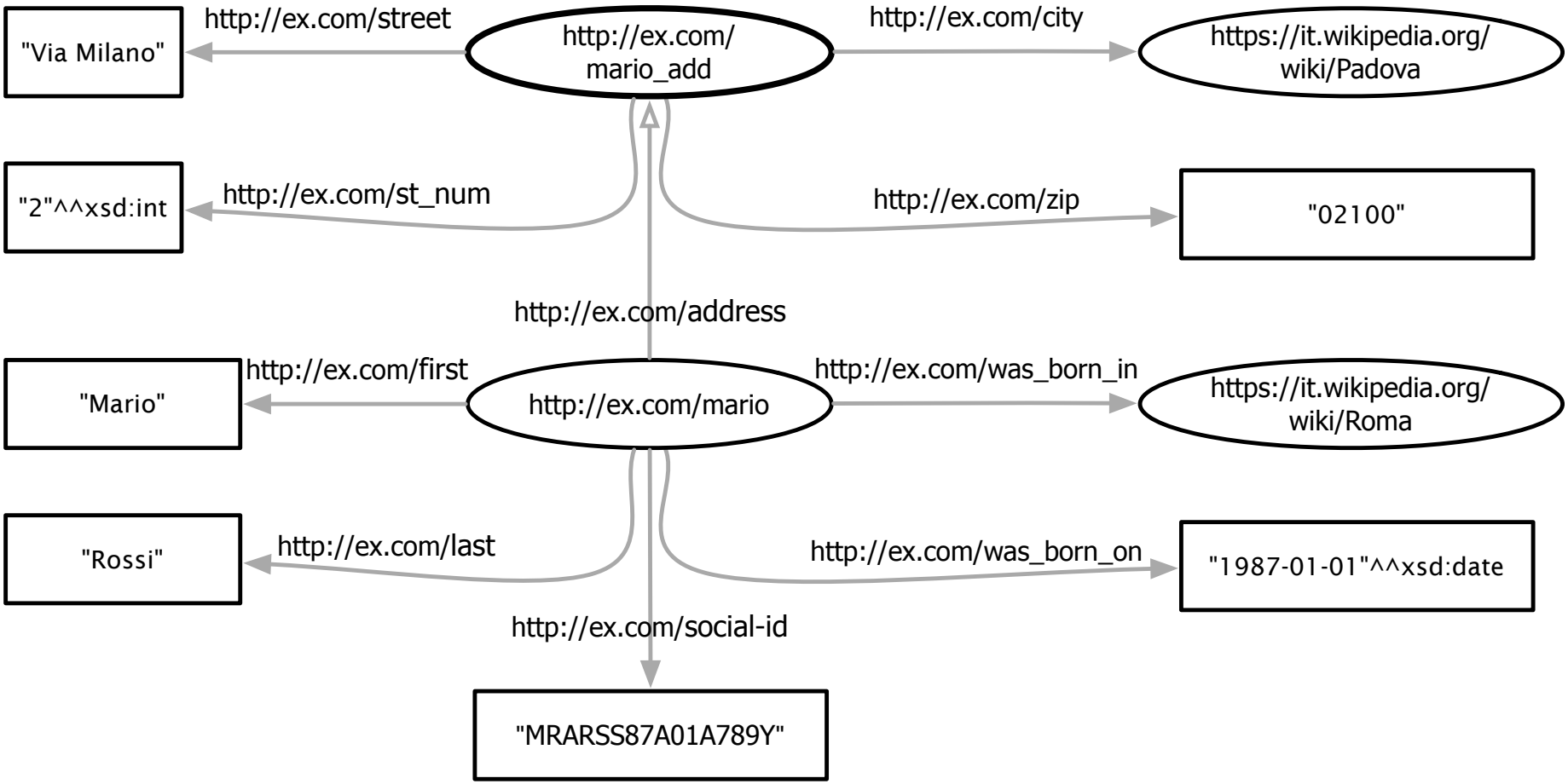
RDF graph: a set of RDF triples.

RDF dataset: a collection of RDF graphs, comprising: one default graph with no name and zero or more named graphs.

A semantic network about Mario Rossi:



An RDF semantic network:



RDF networks are serialized in various formats for automatic processing:

- Turtle

RDF has a formal semantics inspired by mathematical logic and based on the notion of interpretation, which captures the idea of a *possible world.*

Based on interpretation, a definition of entailment is given: a graph G entails a graph E if E is true in all worlds in which G is true.

Entailment allows defining a mechanism for extracting information from semantic networks, based on the SPARQL Query Language.

What's the FOAF name of each known resource and how many friends does it have?

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name (COUNT(?friend) AS ?count)
WHERE {
      ?person foaf:name ?name .
      ?person foaf:knows ?friend }
GROUP BY ?person ?name
```

Useful Readings

- Guus Schreiber, Yves Raimond. RDF 1.1 Primer. https://www.w3.org/TR/rdf11-primer/
- Richard Cyganiak, David Wood, Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. http://www.w3.org/TR/rdf11-concepts/