

RDF: Resource Description Framework



CS-E4410 Semantic Web, 20.1.2021

Eero Hyvönen

Aalto University, Semantic Computing Research Group (SeCo) <http://seco.cs.aalto.fi>

University of Helsinki, HELDIG

<http://heldig.fi>

eero.hyvonen@aalto.fi

Learning Objectives

Understand why RDF data model is useful

- RDF is the foundation of the Semantic Web!

Learn the RDF data model basic principles

Learn the RDF language(s)

= how RDF graphs are serialized (represented as text)



Outline

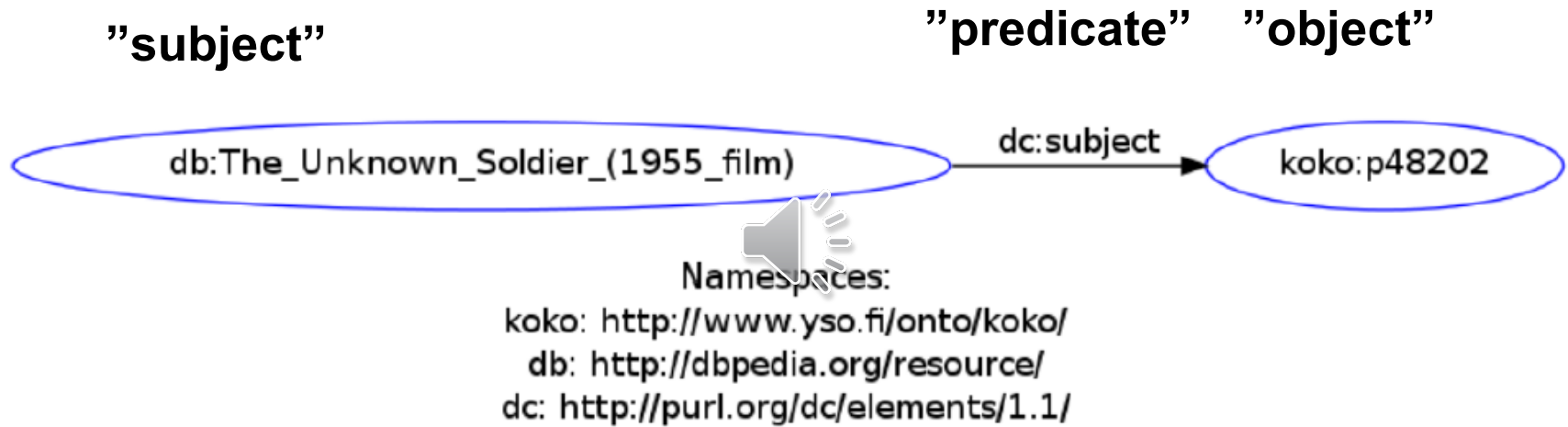
- RDF data model
- RDF syntax



RDF data model



Key idea: triple



RDF graph = set of RDF triples

- Two node types are used

- Literals (for data values)
 - Can be represented as boxes
 - Arcs can point to literals, but not start from them
- Resources
 - = IRI/URI web identifiers
 - Can be represented as oval circles
 - Arcs can point to resources or start from them, too.

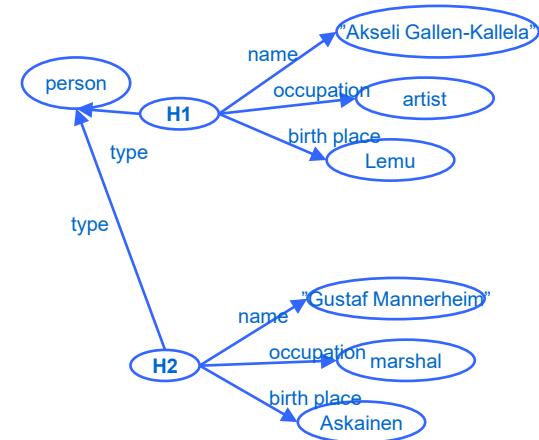


RDF data model and relational databases

- Information is often available as tables in relational databases or CSV files
- RDF is a set of triples
 - n-ary information can be represented as triples
- RDF is a data model: **directed named graph**

person	name	occupation	birth place	...
H1	Akseli Gallen-Kallela	artist	Lemu	
H2	Gustaf Mannerheim	marshal	Askainen	
...				

subject	predicate	object
H1	type	person
H1	name	Akseli Gallen-Kallela
H1	occupation	artist
H1	birth place	Lemu
H2	type	person
H2	name	Gustaf Mannerheim
H2	occupation	marshal
H2	birth place	Askainen



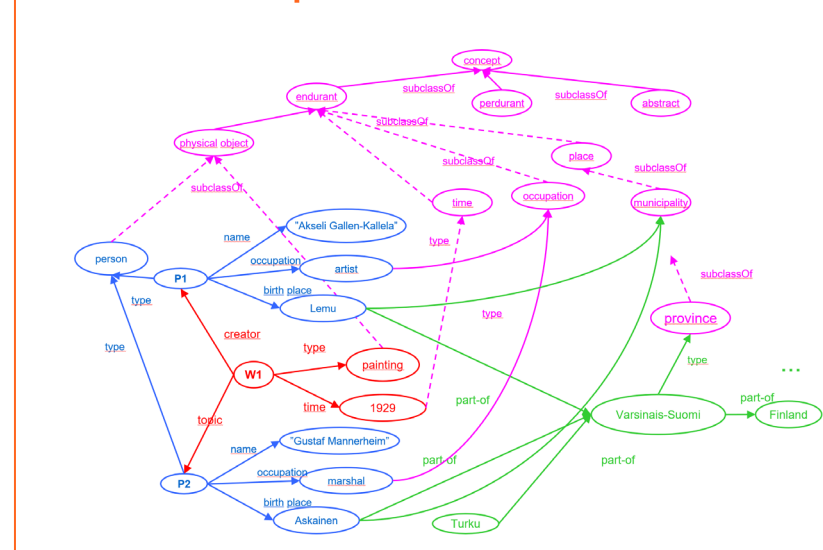
Why RDF graphs are useful?

- Very simple knowledge representation model
- Still very powerful model
- Semantics of graphs can be defined in logic
- Easy to combine several graphs
- - For linked data



Cf. example for the previous lecture:

Semantic RDF Graph Combines All Data: Web of Data



RDF Syntax (Language(s))

N-Triples and Turtle  Expressing Graphs

N-Triples

= straight-forward way for representing graphs

Triple set is serialized in the following form:

```
subject1 predicate1 object1 .  
subject2 predicate2 object2 .
```

...

IRIs are enclosed in angle brackets (<>):  <iri>:

```
<http://example.org/product2> .  
<http://www.w3.org/1999/02/22-rdf-syntax#type> .  
<http://example.org/computer> .
```

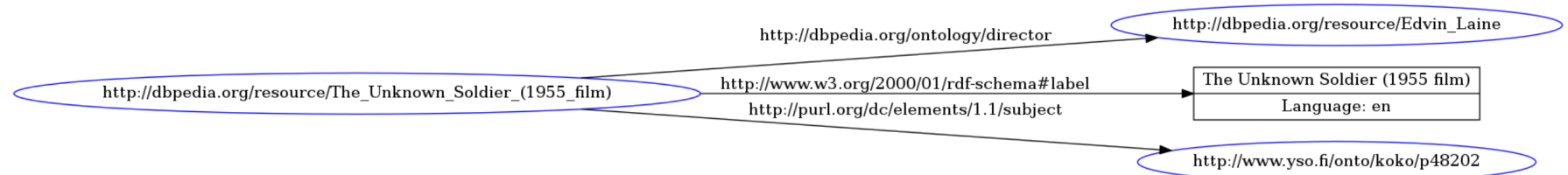
- For machines easy to read/write line by line
- For humans difficult to read/write due to redundancy

Example (from Wikipedia/DBpedia)

```
# Tuntemattoman sotilaan ohjasi Edvin Laine
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://dbpedia.org/ontology/director> # Ohjaaja-ominaisuus
<http://dbpedia.org/resource/Edvin_Laine> . # Edvin Laine

# Filmin nimi englanniksi
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://www.w3.org/2000/01/rdf-schema#label> # Nimike
"The Unknown Soldier (1955 film)"@en . # Literaaliarvo

# Tuntemattoman sotilaan aiheena on sota
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://purl.org/dc/elements/1.1/subject> #Aihe
<http://www.yso.fi/onto/koko/p48202> . # "Sota" KOKO:ssa
```



A Tool to Visualize RDF Graphs:

<http://www.lda.fi/service/rdf-grapher/>



RDF Grapher

RDF grapher is a web service for parsing RDF data and visualizing it as a graph.

The service is based on [Redland Raptor](#) and [Graphviz](#).

Supported RDF serialization formats: Turtle, RDF/XML, RDF/JSON, N-Triples, TriG, and N-Quads.

Supported image formats: PNG, SVG, PDF, PS, EPS, GIF, and JPG.

Usage:

http://www.lda.fi/service/rdf-grapher?rdf=DATA_OR_URI&from=FORMAT&to=FORMAT

GET/POST parameters:

rdf RDF data or URI
from input serialization format (ttl, xml, json, nt, trig, nq), default: ttl
to output image format (png, svg, pdf, ps, eps, gif, jpg), default: png

Examples:

<http://www.lda.fi/service/rdf-grapher?rdf=<http://example.com/s>+<http://example.com/p>+<http://example.com/o>+.&from=ttl&to=png>

<http://www.lda.fi/service/rdf-grapher?rdf=http://dbpedia.org/resource/Helsinki&from=xml&to=png>

Try the service:

RDF data or URI:

```
# Tuntemattoman sotilaan ohjasi Edvin Laine
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://dbpedia.org/ontology/director> # Ohjaaja-ominaisuus
<http://dbpedia.org/resource/Edvin_Laine> . # Edvin Laine
# Filmin nimi englanniksi
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://www.w3.org/2000/01/rdf-schema#label> # Nimi
"The Unknown Soldier (1955 film)"@en . # Literaaliarvo
# Tuntemattoman sotilaan aiheena on sota
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://purl.org/dc/elements/1.1/subject> #Aihe
<http://www.yso.fi/onto/koko/p48202> . # "Sota" KOKO:ssa
```

From format:

To format:

Send form as HTTP POST (needed for large RDF data):



Turtle simplifies N-triples: Examples

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix db: <http://dbpedia.org/resource/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix koko: <http://www.yso.fi/onto/koko/> .
```

Using namespace prefixes
as short hand notation for
long identifiers

```
# Tuntemattoman sotilaan ohjasi Edvin Laine
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
dbo:director # Ohjaaja-ominaisuus DBpedian ontologiassa
db:Edvin_Laine . # Edvin Laineen resurssi

# Filmin nimi englanniksi
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
rdfs:label # Ominaisuus label kertoo nimikkeen
"The Unknown Soldier (1955 film)"@en . # Literaaliarvo

# Tuntemattoman sotilaan aiheena on sota
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
dc:subject #Aiheen kertova ominaisuus
koko:p48202 . # Käsité "sota" KOKO-ontologiassa
```

```
# Tuntemattoman sotilaan ohjasi Edvin Laine
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://dbpedia.org/ontology/director> # Ohjaaja-ominaisuus
<http://dbpedia.org/resource/Edvin_Laine> . # Edvin Laine

# Filmin nimi englanniksi
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://www.w3.org/2000/01/rdf-schema#label> # Nimike
"The Unknown Soldier (1955 film)"@en . # Literaaliarvo

# Tuntemattoman sotilaan aiheena on sota
<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
<http://purl.org/dc/elements/1.1/subject> #Aihe
<http://www.yso.fi/onto/koko/p48202> . # "Sota" KOKO:ssa
```

Data in simpler Turtle notation

Original data in N-triples

Representing multiple property values & several properties of a resource

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix db: <http://dbpedia.org/resource/> .

<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
  rdfs:label "Tuntematon sotilas (1955 filmi)"@fi,
            "The Unknown Soldier (1955 film)"@en .
```

Multiple property values

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix db: <http://dbpedia.org/resource/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix koko: <http://www.yso.fi/onto/koko/> .

<http://dbpedia.org/resource/The_Unknown_Soldier_(1955_film)>
  dbo:director db:Edvin_Laine ;
  rdfs:label "The Unknown Soldier (1955 film)"@en ;
  dc:subject koko:p48202 .
```



Several properties

```

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix wgs: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix : <http://paikat.fi> .

```

```

:p7 rdfs:label "Helsinki" ;
  dc:location [
    rdf:type :Point;
    wgs:lat 60.17;
    wgs:long 24.94
  ] .

```



Nesting blank nodes



Namespaces:
 rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 rdfs: <http://www.w3.org/2000/01/rdf-schema#>
 dc: <http://purl.org/dc/elements/1.1/>
 wgs: http://www.w3.org/2003/01/geo/wgs84_pos#
 http://paikat.fi

Turtle – more syntactic sugar



Example

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .

<#green-goblin>
  rel:enemyOf <#spiderman> ;
  a foaf:Person ; # in the context of the Marvel universe
  foaf:name "Green Goblin" .

<#spiderman>
  rel:enemyOf <#green-goblin> ;
  a foaf:Person ;
  foaf:name "Spiderman", "Человек-паук"@ru .
```


RDF data model in more detail:

- Literals (for data values)
- Resources (and their identifiers)
- Statements (triples)
- Graphs
- Datasets and quads



Literals

for representing data values



Literals

Literal is data encoded as a string

- "Suomi", "Last waltz in Paris"

Literal value can be accompanied with a XML language tag:

- "Suomi"@fi, "Last waltz in Paris"@en

Literal value can be accompanied with a datatype (XML Schema)

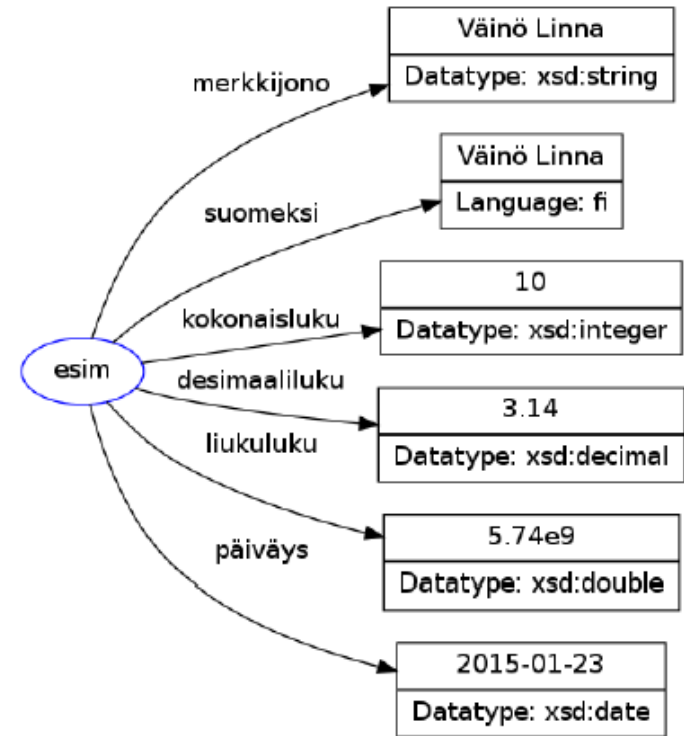
- "-5"^^xsd:integer, "4.2E9"^^xsd:double
- Abbreviated numeric literals: -5, 4.2E9
- Default datatype: "Suomi" = "Suomi"^^xsd:string

Visualized typically as a rectangle in an RDF graph

3.14
Datatype: xsd:decimal

Example

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
@prefix : <http://koe.fi> .  
  
:esim  
  :merkkijono "Väinö Linna"^^xsd:string ;  
  :suomeksi "Väinö Linna"@fi ;  
  :kokonaisluku "10"^^xsd:integer ;  
  :desimaaliluku "3.14"^^xsd:decimal ;  
  :liukuluku "5.74e9"^^xsd:double ;  
  :päiväys "2015-01-23"^^xsd:date .
```



Namespaces:
xsd: http://www.w3.org/2001/XMLSchema#
http://koe.fi

Resources and their identifiers

for identifying resources globally

URL


URL: Uniform Resource Locator

- An identifier that also describes its network location for the HTTP protocol
- When one writes a URL in a browser one gets an HTML page in return
- <http://www.aalto.fi/fi/research/>
- <http://www.ask.com/web?qsrc=1&o=0&l=dir&q=Capital+of+Finland&qo=serpSearchTopBox>
- <http://urn.fi/urn:isbn:978-952-10-4171-6>



URI

URI: Uniform Resource Identifier

- Identifier that conforms syntactically to some official **URI scheme** on the Web
 - *E.g., http, https, ftp, mailto, urn, oid, xmpp, ...*
- URL is one of the URI schemes but there are tens of others, too
 - *A URI is not necessarily a web address*
- <http://dbpedia.org/resource/Helsinki>
 - *Identifier for the concept of Helsinki in DBPedia*
- <mailto:eero.hyvonen@aalto.fi> Send email
- *Identifier for an email address embedded in HTML code*

URN

URN: Uniform Resource Name

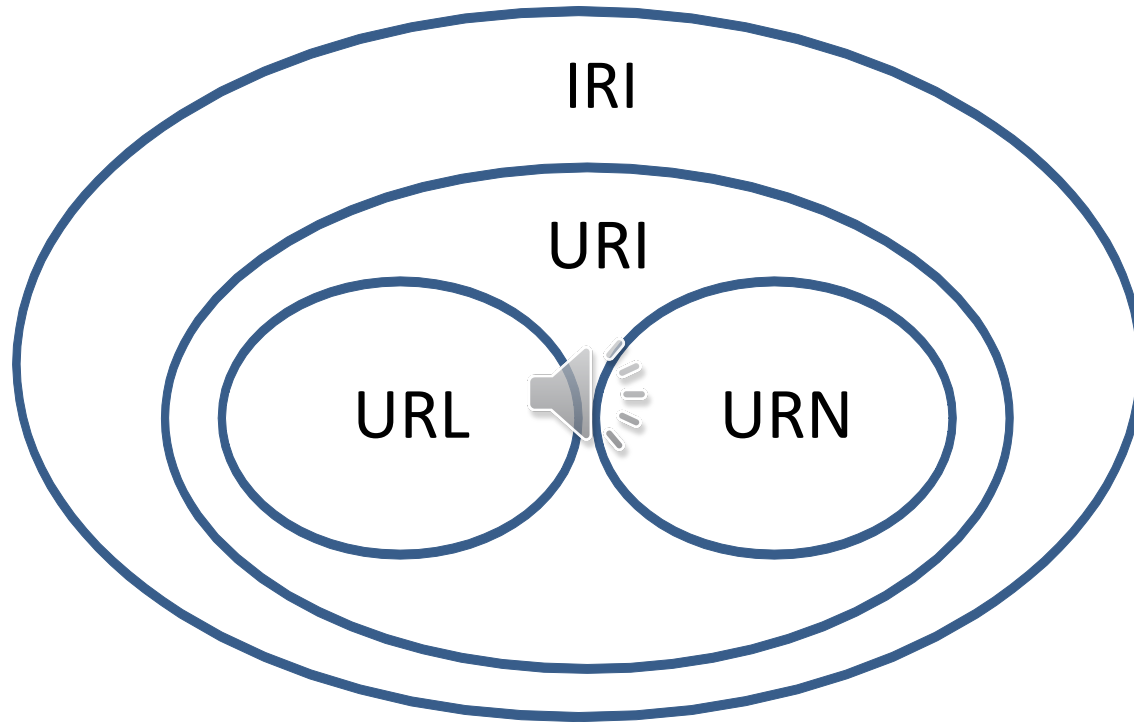
- An URI scheme for identifiers that only specifies a name but not address (like in HTTP)
- <urn:isbn:978-952-10-4171-6>

IRI

IRI: Internationalized Resource Identifier

- Generalization of URI based on Unicode character set
- URL encoding not needed
- http://dbpedia.org/resource/Väinö_Linna





URI schemes

- Particular syntactic types of URIs with an agreed interpretation
- Standardized by IANA Internet Assigned Numbers Authority
 - *Tens of URI schemes are available:*
 - ftp, http, mailto, urn, oid, xmpp, ...
- Semantic Web advocates the use of HTTP URI/IRIs (URLs)
 - *HTTP URIs not only identify things but are addresses, too*
 - *Type URI in a browser and you get useful info back!*

IRI/URI syntax

3. Syntax Components

The generic URI syntax consists of a hierarchical sequence of components referred to as the scheme, authority, path, query, and fragment.

```
URI          = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

hier-part    = "//" authority path-abempty
              / path-absolute
              / path-rootless
              / path-empty
```

The scheme and path components are required, though the path may be empty (no characters). When authority is present, the path must either be empty or begin with a slash ("/") character. When authority is not present, the path cannot begin with two slash characters ("//"). These restrictions result in five different ABNF rules for a path (Section 3.3), only one of which will match any given URI reference.

The following are two example URIs and their component parts:

```
foo://example.com:8042/over/there?name=ferret#nose
  \  /  \  /  \  /  \  /  \  /  \  /  \  /
  |      |      |      |      |
scheme authority path query fragment

  \  /  \  /
  |      |
urn:example:animal:ferret:nose
```

IRI: IETF RFC 3987

”Blank nodes” without a known identifier

Blank nodes arise from nested Turtle descriptions

- Can be represented in Turtle using brackets [...]
- Blank node IRI is not given by the user but by the system when there is no need for assigning an IRI for making external references
- Blank nodes used only locally and internally in creating RDF graphs

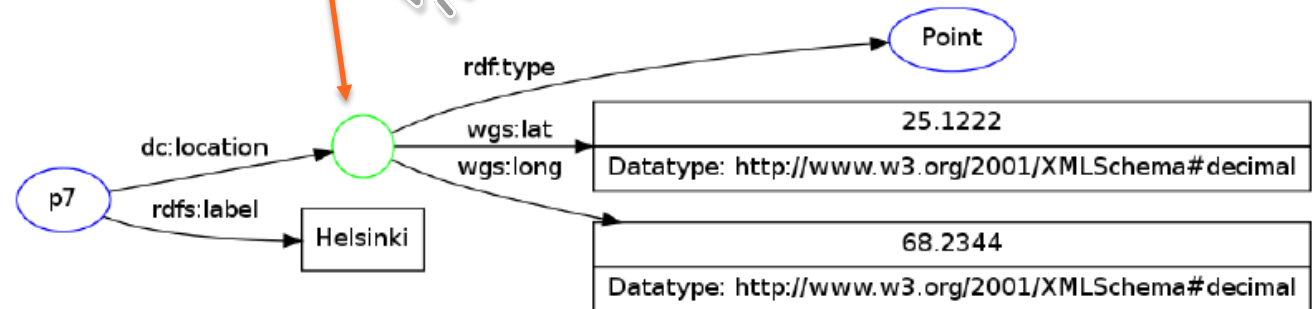
Blank nodes are also called “bnodes” or “anonymous nodes”

Blank nodes are written in form `_:identifier`

Example

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix wgs: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix : <http://paikat.fi> .

:p7 rdfs:label "Helsinki" ;
  dc:location [
    rdf:type :Point;
    wgs:lat 25.1222;
    wgs:long 68.2344
  ] .
```



Namespaces:
rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
rdfs: <http://www.w3.org/2000/01/rdf-schema#>
dc: <http://purl.org/dc/elements/1.1/>
wgs: http://www.w3.org/2003/01/geo/wgs84_pos#
<http://paikat.fi>

Statements

asserting information



Statement

Statement asserts a relationship (property) between two resources

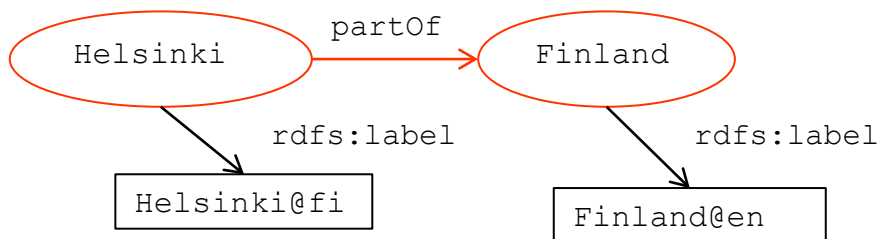
- E.g., "Helsinki is part of Finland"

Statement is represented as a triple

- `<resource, property, property_value>`
`<subject, predicate, object>`



RDF graph = set of statements



Subject	Predicate	Object
1. Helsinki	partOf	Finland
2. Helsinki	rdfs:label	"Helsinki"@fi
3. Finland	rdfs:label	"Finland"@en

Statement characteristics

Subject is an IRI or blank node

Predicate is an IRI (blank node is not reasonable predicate)

Object is an IRI, literal, or blank node

- Literals are used only as property values



Are binary predicates enough for knowledge representation?

RDF uses only binary predicates (statements/triples)

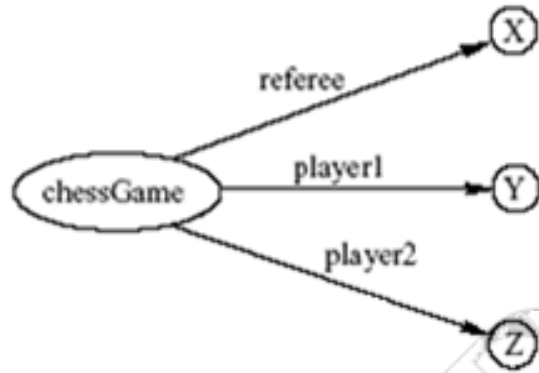
- Often we use predicates with more than 2 arguments

Example problem: referee(X, Y, Z)

- **X** is the referee in a chess game between players **Y** and **Z**

N-ary predicates can always be represented by binary ones:

- a new auxiliary resource **chessGame**
- new binary predicates for arguments: **ref**, **player1**, and **player2**



RDF graphs

set of triples

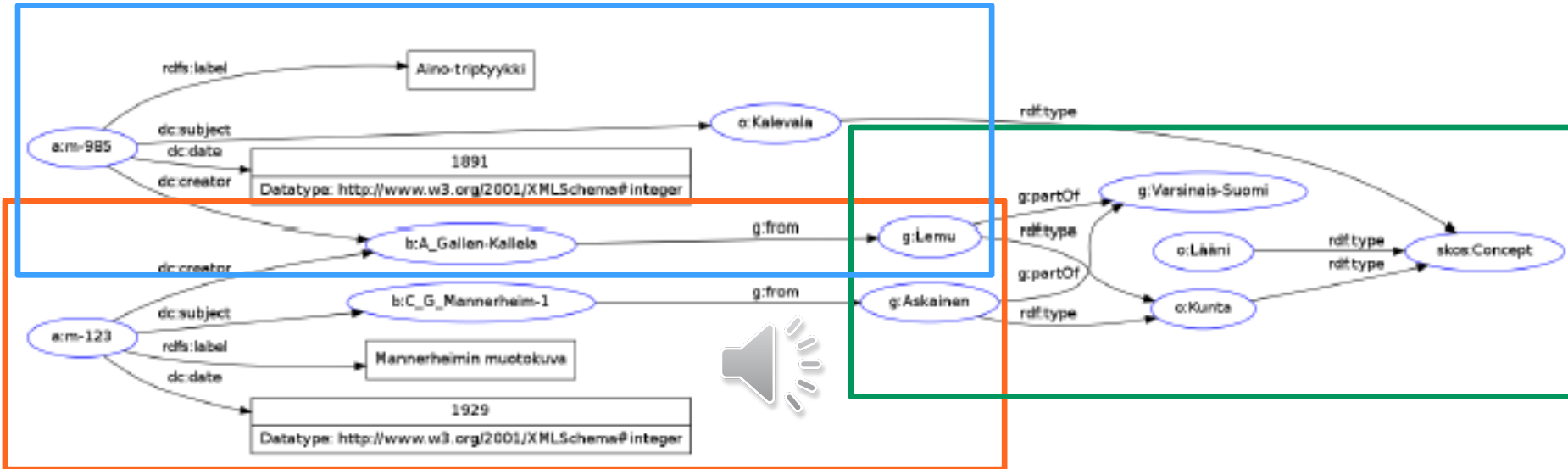
RDF graph

RDF graph = set of triples (statements)

- $\langle \text{start node, arc, end node} \rangle$ i.e.
 $\langle \text{subject, predicate, object} \rangle$

Multiple graphs can be merged with the union operation of set theory

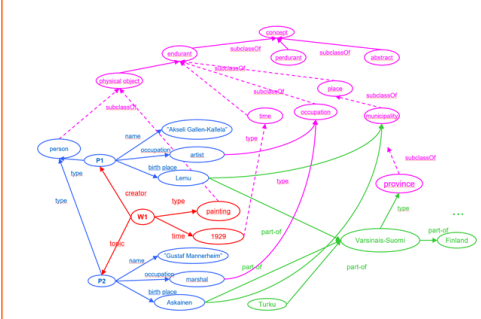




Namespaces:
 a: http://art.org/
 b: http://bio.org/
 g: http://geo.org/
 o: http://onto.org/
 rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
 rdfs: http://www.w3.org/2000/01/rdf-schema#
 skos: http://www.w3.org/2004/02/skos/core#
 dc: http://purl.org/dc/elements/1.1/

An accurate formal representation of our earlier example:

Semantic RDF Graph Combines All Data: Web of Data



Datasets and quads

set of graphs

Datasets, graphs, and quads in RDF 1.1

- **Dataset** consists of a set of **RDF graphs**
 - Multiple *named graphs* and at most one *unnamed (default) graph*
- Graphs are encode sets of quads, where the 4th position is a **graph IRI**
 - `<http://example.org/spiderman>`
`<http://www.perceive.net/schemas/relationship/enemyOf>`
`<http://example.org/green-goblin>`
`<http://example.org/graphs/spiderman>` .
 - If the 4th member is omitted, the triple belongs to the default graph

Quads

A quad is like a triple but with four arguments

Adding the graph information (4. argument) into a triple can be important

- Information modularization
 - *E.g., restricting the search only to a specific named graph*
- Representing provenience information
 - *E.g., the origin of the statement or the date of the addition into the dataset*
 - *Used, e.g., in the Google Knowledge Graph*
 - *Facilitates the management of contents*

Other Approaches to RDF syntax (in addition to N-Triples and Turtle)



N-Quads

Extends N-Triples notation for representing triples with graph information (line by line)

```
<http://example.org/spiderman>  
<http://www.perceive.net/schemas/relationship/enemy  
Of> <http://example.org/green-goblin>  
<http://example.org/graphs/spiderman> .
```

TriG

Extends basic Turtle notation for representing datasets (set of graphs)


```
# This document contains a default graph and two named graphs.

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

# default graph
{
  <http://example.org/bob> dc:publisher "Bob" .
  <http://example.org/alice> dc:publisher "Alice" .
}

<http://example.org/bob>
{
  _:a foaf:name "Bob" .
  _:a foaf:mbox <mailto:bob@oldcorp.example.org> .
  _:a foaf:knows _:b .
}

<http://example.org/alice>
{
  _:b foaf:name "Alice" .
  _:b foaf:mbox <mailto:alice@work.example.org> .
}
```



RDF/XML

- RDF/XML is the original RDF syntax in the RDF 1.0 recommendation
 - RDF serialized in XML notation
 - Existing XML tools available
- Complicated syntax for humans to use
- Turtle is now in common use instead



Example of RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:ex="http://example.org/stuff/1.0/">
  <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar"
    dc:title="RDF/XML Syntax Specification (Revised)">
    <ex:editor rdf:nodeID="abc"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="abc"
    ex:fullName="Dave Beckett">
    <ex:homePage rdf:resource="http://purl.org/net/dajobe/" />
  </rdf:Description>
</rdf:RDF>
```

More information: <https://www.w3.org/TR/rdf-syntax-grammar/>

For web programming

JSON-LD

JSON-LD (JSON Linked Data)

- Human-readable notation with built-in support in programming languages/environments, such as JavaScript and Python
- See the interactive JSON-LD ”playground” online:
 - <http://json-ld.org/>
- For a general JSON tutorial online, see:
 - https://www.w3schools.com/js/js_json_intro.asp



Example of JSON-LD

<http://json-ld.org/playground/index.html>

JSON-LD Playground

Play around with JSON-LD markup by typing out some JSON below and seeing what gets generated from it at the bottom of the page. Pick any of the examples below to get started. The playground uses the [jsonld.js JSON-LD processor](#) which fully conforms to the JSON-LD [Syntax](#) and [API](#) specifications.

Examples: Person Event Place Product Recipe Library Permalink Shortcuts

Document URL

JSON-LD Input

```
{
  "@context": "http://schema.org/",
  "@type": "Person",
  "name": "Jane Doe",
  "jobTitle": "Professor",
  "telephone": "(425) 123-4567",
  "url": "http://www.janedoe.com"
}
```

Transform JSON-LD into triples

Expanded Compacted Flattened Framed Quads Normalized

```
_:c14n0 <http://schema.org/jobTitle> "Professor" .
_:c14n0 <http://schema.org/name> "Jane Doe" .
_:c14n0 <http://schema.org/telephone> "(425) 123-4567" .
_:c14n0 <http://schema.org/url> <http://www.janedoe.com> .
_:c14n0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Person> .
```


RDF data validation

- The validity of different RDF syntaxes can be checked with validators
- As part of the validation the serialized representation can be visualized as an RDF graph
- <http://www.ldf.fi/service/rdf-grapher/> at Linked Data Finland
- [W3C RDF/XML validator](#)
- [More validators](#)



Summary: Serialization of RDF

Representing RDF graphs as linear text (string)

- E.g., in a file: reading and writing

Alternative serializations for different needs

1. Intuitive for humans to read/write
 - ***N-triples***, *Notation 3*
 - ***Turtle***
 - *TriG*, *N-Quads*
2. XML-interpretability for machines
 - ***RDF/XML***
 - *Existing XML tools available*
3. For **web** programming
 - ***JSON-LD***
4. Embedding in web pages
 - ***RDFa***
 - *Publishing information for, e.g., search engines*
 - *To be explained later in the course*



Summary (RDF)

- RDF provides a foundation for representing and processing metadata
- RDF has a graph-based data model
- RDF has different syntaxes
- RDF has a decentralized philosophy for data linking
 - *Incremental building of knowledge*
 - *Sharing and reusing metadata*
- RDF is domain-independent

