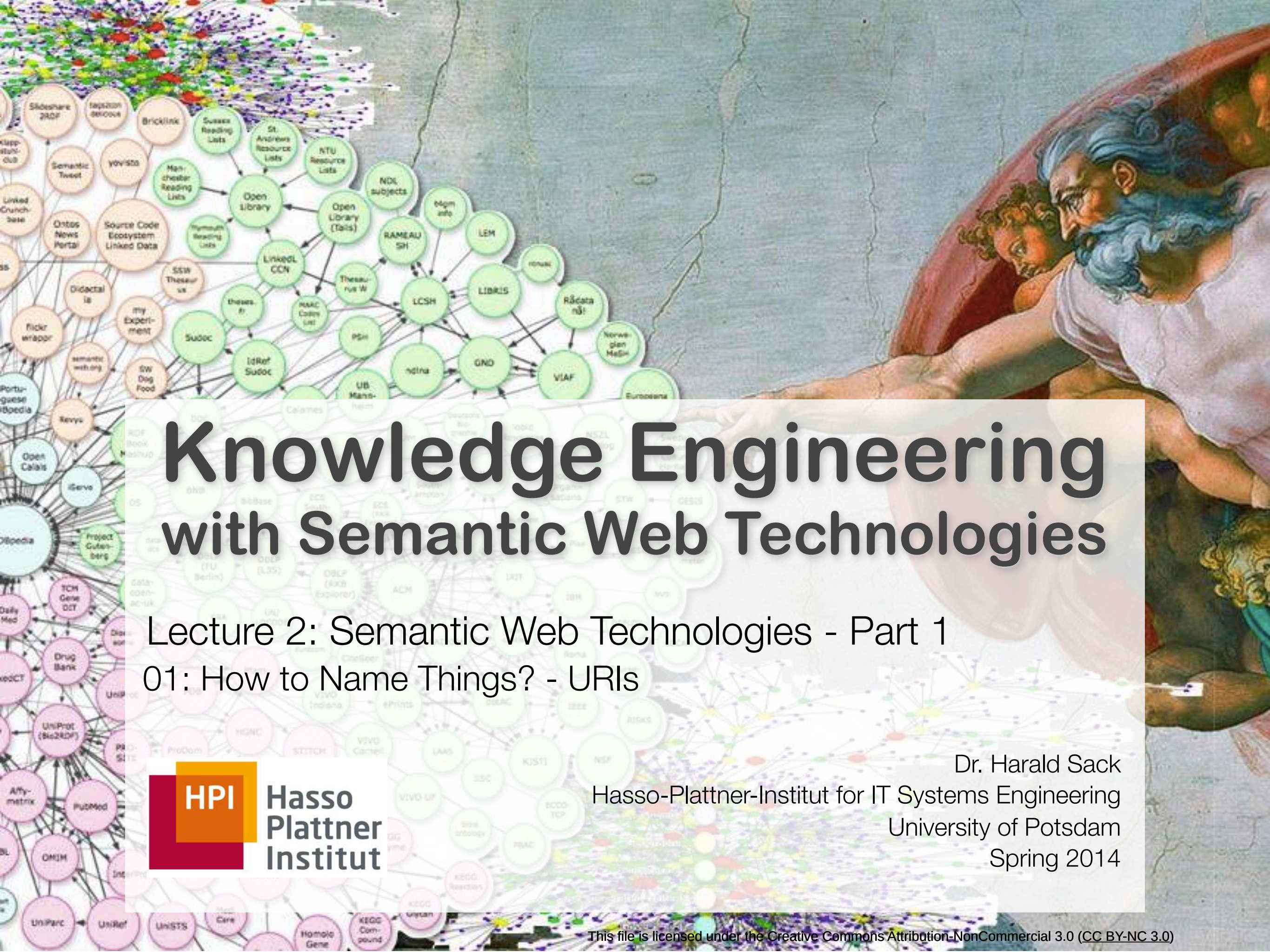




Knowledge Engineering with Semantic Web Technologies

Lecture 2: Semantic Web Technologies - Part 1

01: How to Name Things? - URIs



Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014



Lecture 2: Semantic Web Technologies - Part 1

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies

Semantic Web Technologies

Part 1

The Semantic Web Technology Stack (not a piece of cake...)

Most apps use only a subset of the stack

Querying allows fine-grained data access

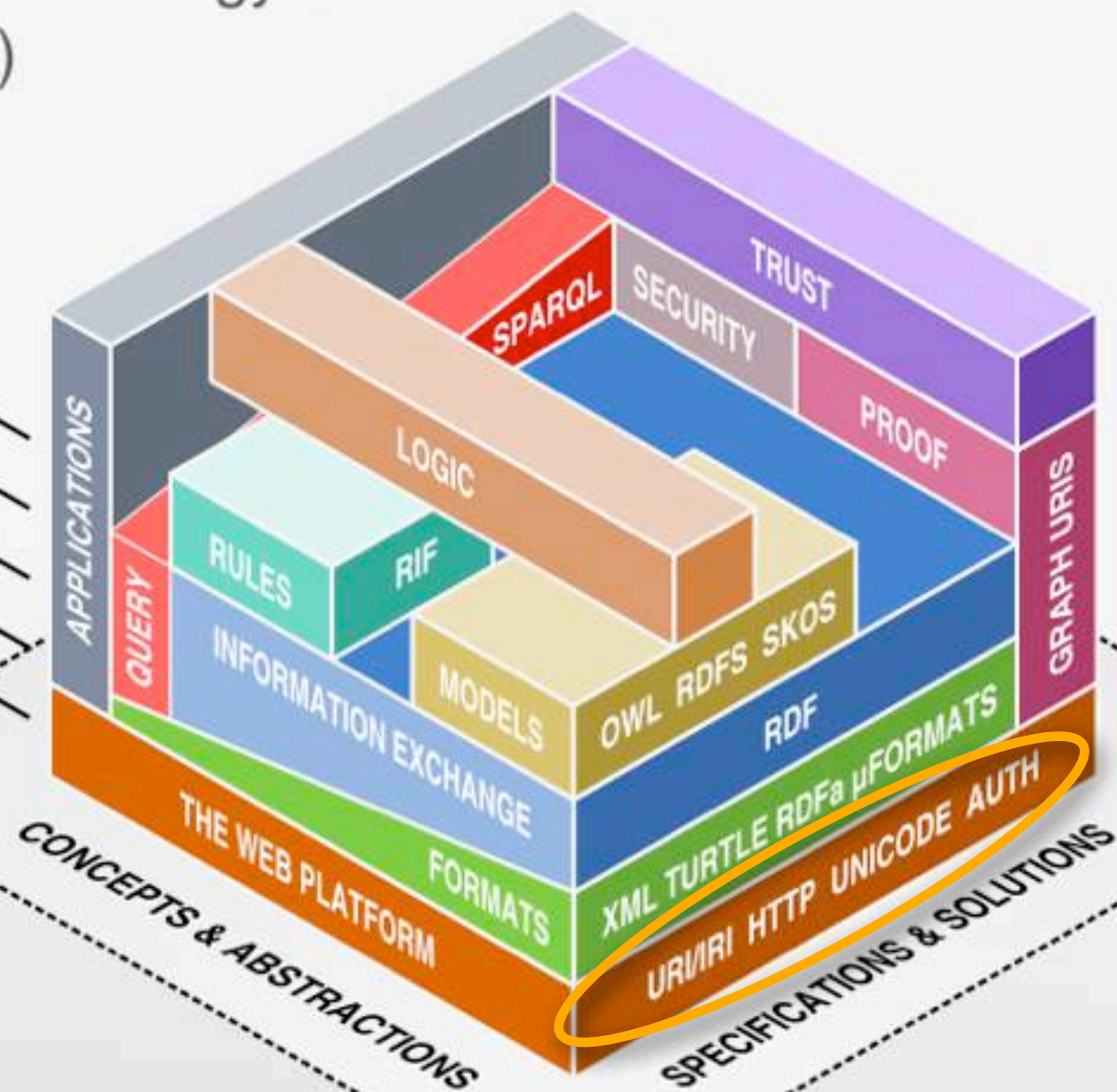
Standardized information exchange is key

Formats are necessary, but not too important

The Semantic Web is based on the Web

Linked Data uses a small
selection of technologies

LINKED DATA

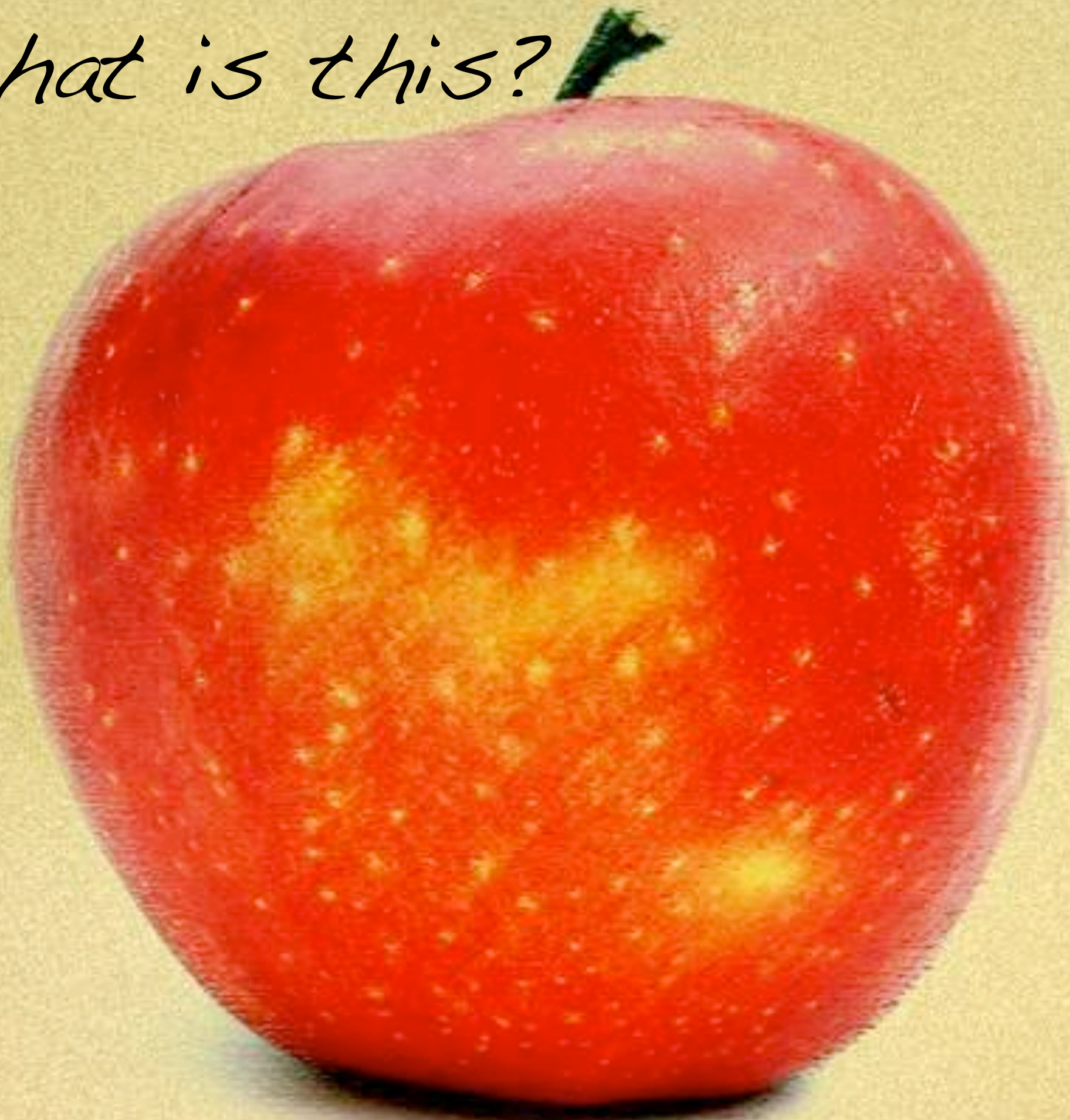




01 How to Name Things? - URIs

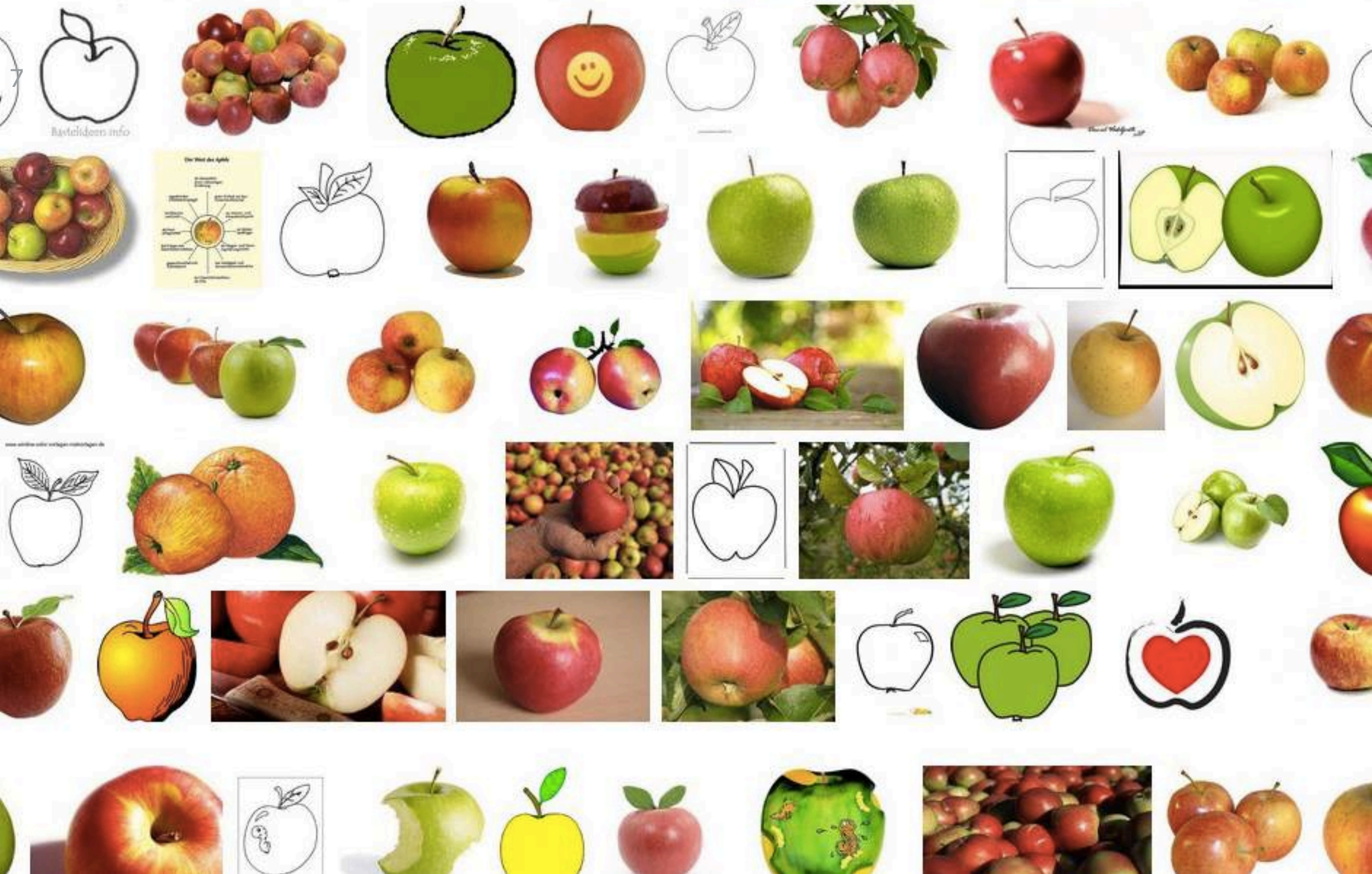
Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I

What is this?

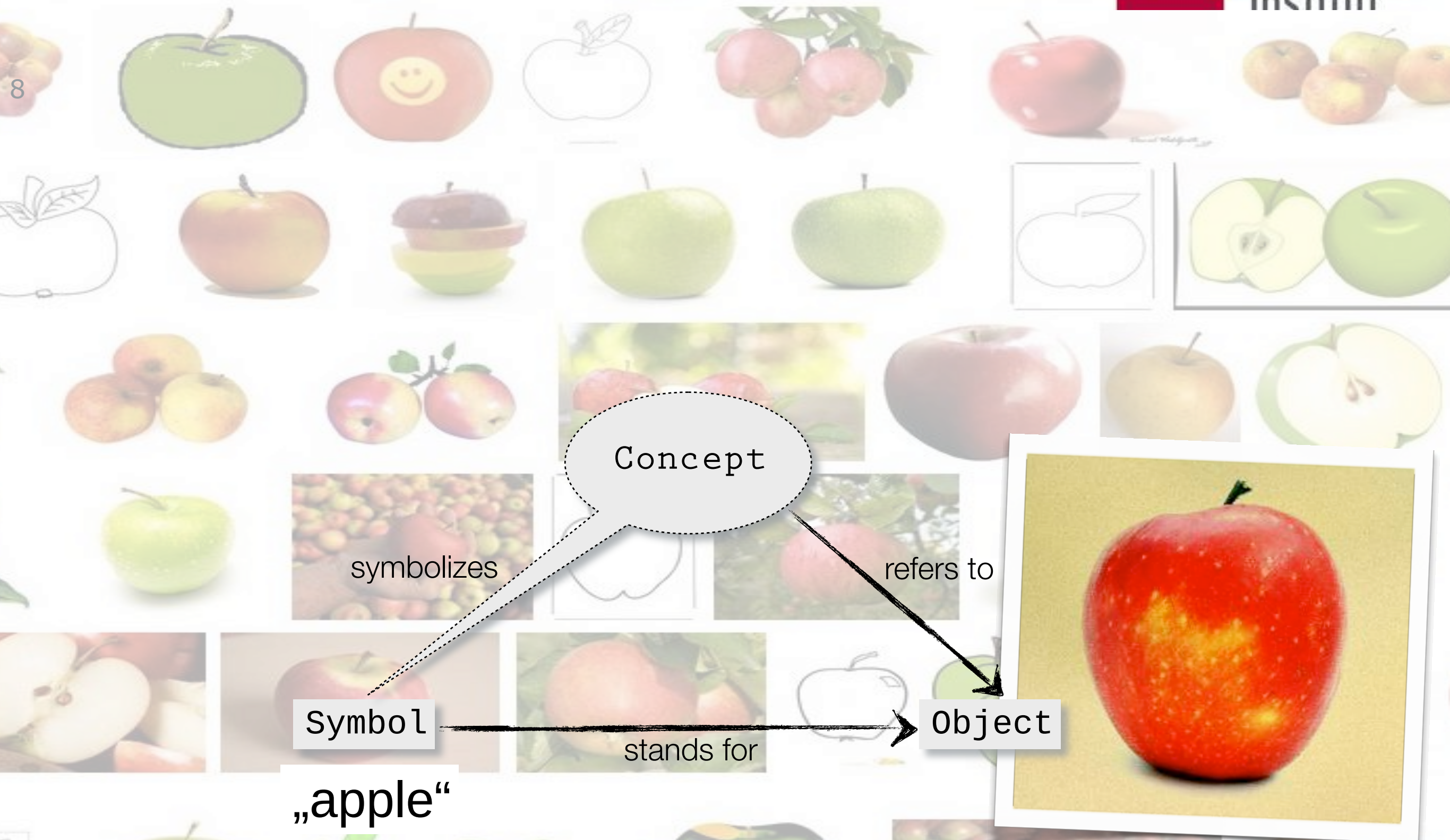




Ceci n'est pas une pomme



Semiotic Triangle



Ogden, Richards: Semiotic Triangle, 1923

Semiotic Triangle



shared concept

Concept

symbolizes

refers to

Symbol

Object

stands for

„apple“

Ogden, Richards: Semiotic Triangle, 1923

Semiotic Triangle



shared concept

Concept

symbolizes

refers to

Symbol

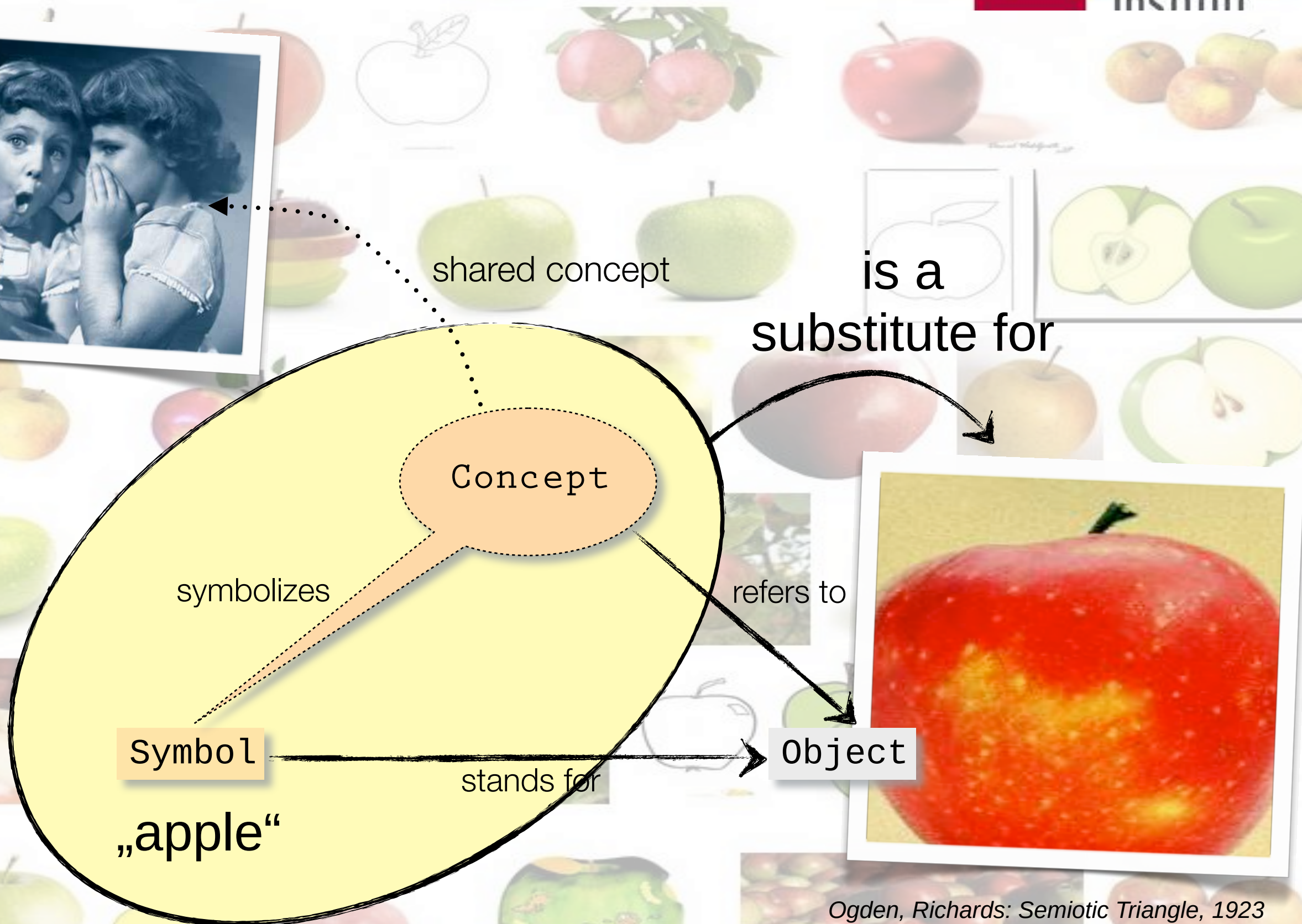
Object

stands for

„apple“

Ogden, Richards: Semiotic Triangle, 1923

Semiotic Triangle



Ogden, Richards: Semiotic Triangle, 1923

10

Uniform Resource Identifier

- different types of resource identifiers all constructed according to a uniform schema
- whatever may be identified via URI
- to distinguish one resource from another

Uniform Resource Identifier

- A **Uniform Resource Identifier** (URI) defines a simple and extensible schema for worldwide unique identification of abstract or physical resources (RFC 3986).

Uniform Resource Identifier

- A **Uniform Resource Identifier** (URI) defines a simple and extensible schema for worldwide unique identification of abstract or physical resources (RFC 3986).
- **Resource** can be every object with a clear identity (according to the context of the application)

Uniform Resource Identifier

- A **Uniform Resource Identifier** (URI) defines a simple and extensible schema for worldwide unique identification of abstract or physical resources (RFC 3986).
- **Resource** can be every object with a clear identity (according to the context of the application)
 - as e.g., web pages, books, locations, persons, relations among objects, abstract concepts, etc.

Uniform Resource Identifier

- A **Uniform Resource Identifier** (URI) defines a simple and extensible schema for worldwide unique identification of abstract or physical resources (RFC 3986).
- **Resource** can be every object with a clear identity (according to the context of the application)
 - as e.g., web pages, books, locations, persons, relations among objects, abstract concepts, etc.
- The concept of URI is already established in various domains, as e.g.,

- A **Uniform Resource Identifier** (URI) defines a simple and extensible schema for worldwide unique identification of abstract or physical resources (RFC 3986).
- **Resource** can be every object with a clear identity (according to the context of the application)
 - as e.g., web pages, books, locations, persons, relations among objects, abstract concepts, etc.
- The concept of URI is already established in various domains, as e.g.,
 - the Web (URL),

- A **Uniform Resource Identifier** (URI) defines a simple and extensible schema for worldwide unique identification of abstract or physical resources (RFC 3986).
- **Resource** can be every object with a clear identity (according to the context of the application)
 - as e.g., web pages, books, locations, persons, relations among objects, abstract concepts, etc.
- The concept of URI is already established in various domains, as e.g.,
 - the Web (URL),
 - Books and publications (ISBN, ISSN),

- A **Uniform Resource Identifier** (URI) defines a simple and extensible schema for worldwide unique identification of abstract or physical resources (RFC 3986).
- **Resource** can be every object with a clear identity (according to the context of the application)
 - as e.g., web pages, books, locations, persons, relations among objects, abstract concepts, etc.
- The concept of URI is already established in various domains, as e.g.,
 - the Web (URL),
 - Books and publications (ISBN, ISSN),
 - Digital Object Identifier (DOI)



Resource

12

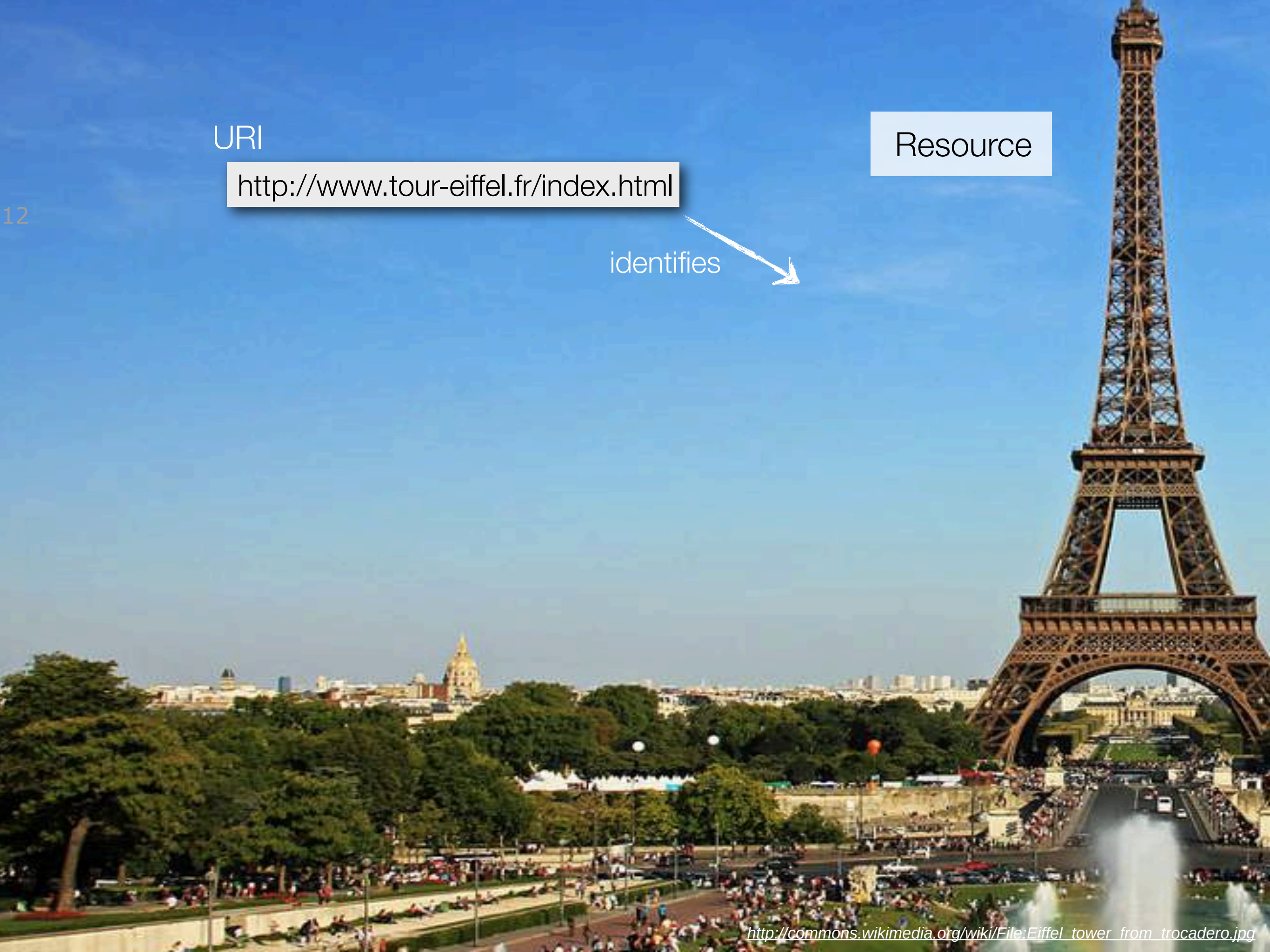


URI

<http://www.tour-eiffel.fr/index.html>

Resource

identifies



URI

`http://www.tour-eiffel.fr/index.html`

Resource

identifies

Representation

represents
(stands for)

Metadata:

Content-type: text/html

Data:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
```

```
<head>
```

```
<title>Le site officiel de la Tour Eiffel</title>
```

```
...
```

```
</html>
```


Representation and Presentation

13

Metadata:

Content-type: text/html

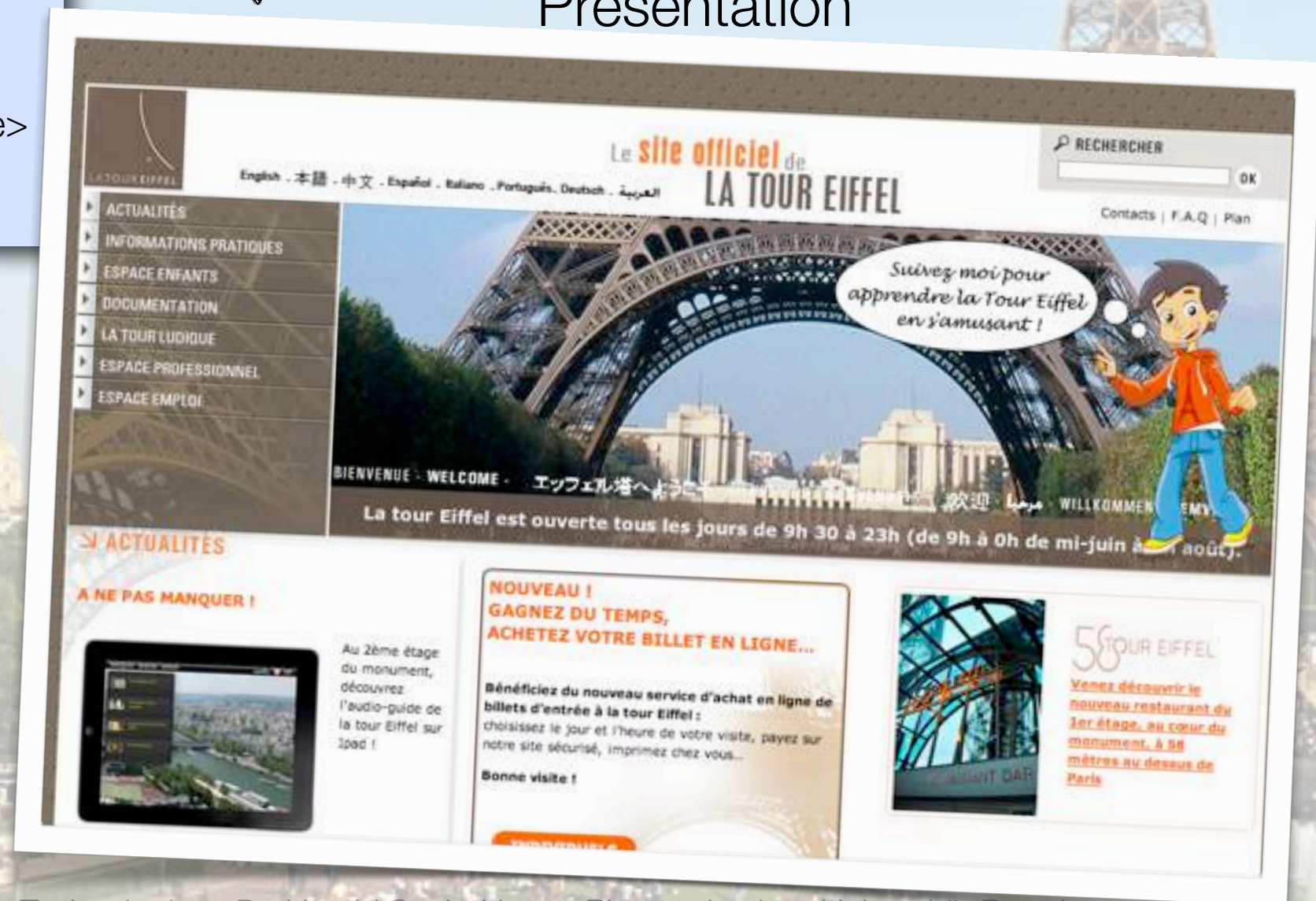
Data:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Le site officiel de la Tour Eiffel</title>
    ...
  </html>
```

defines

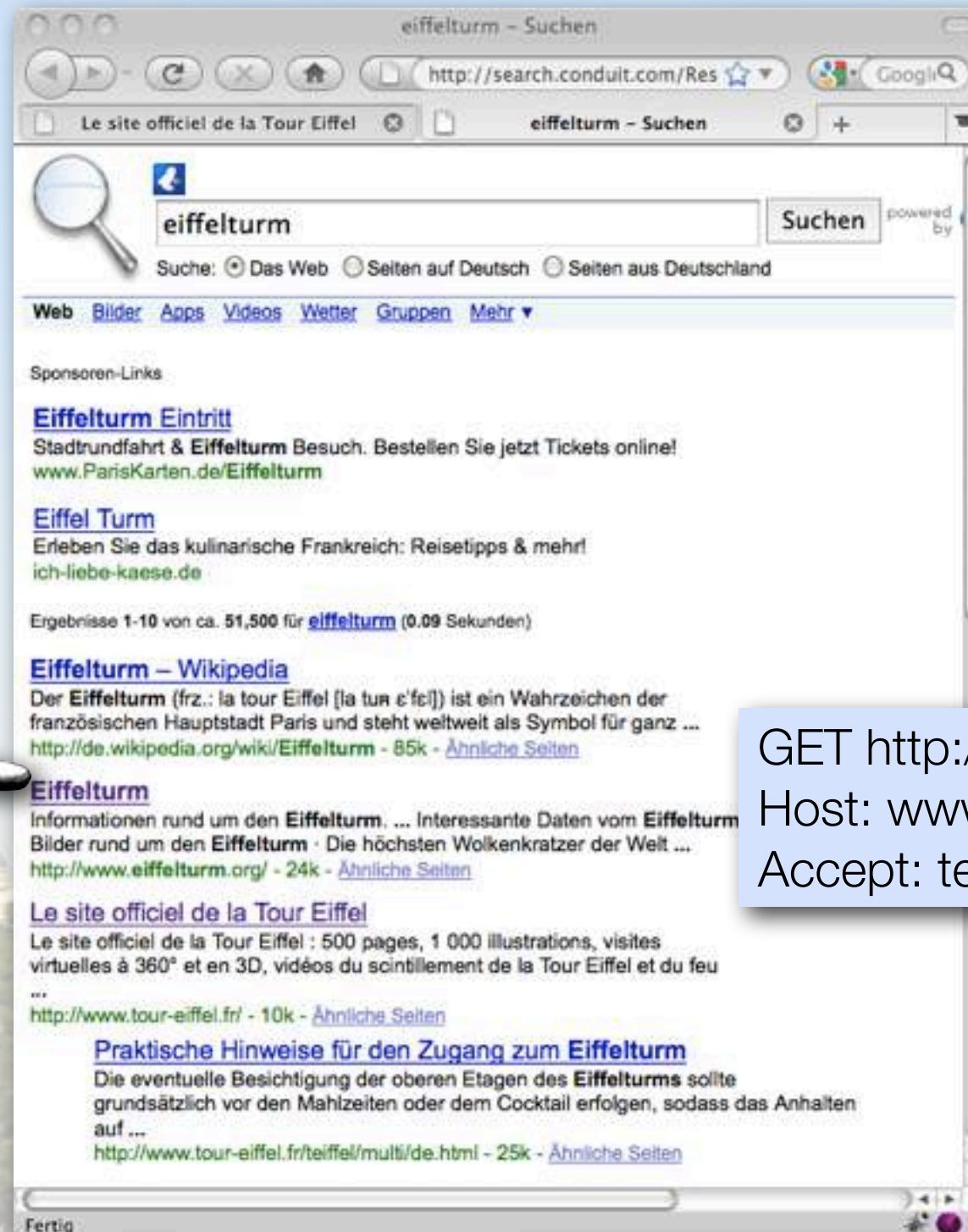
Presentation

Representation



HTTP Request / Response

14



Browser sends
HTTP Request

GET http://www.tour-eiffel.fr/index.html HTTP/1.1
Host: www.tour-eiffel.fr
Accept: text/html

User clicks

HTTP Request / Response

15

Browser sends **HTTP Request**

```
GET http://www.tour-eiffel.fr/index.html HTTP/1.1
Host: www.tour-eiffel.fr
Accept: text/html
```

Web Server
processes request

Web Server sends **HTTP Response**

```
HTTP/1.1 200 OK
Content-Length: 3134
Content-Type: text/html; charset=utf-8

<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Le site officiel de la Tour Eiffel</title>
    ...
  </html>
```

Browser interprets representation
and displays presentation



Designator and Designatum

16

Designator



Designatum

≠

The web page describes
(*designates*) the Eiffel Tower

The Eiffel Tower is described
(*designated*) by the web page





Ceci n'est pas une pipe.

René Magritte:
La trahison des images (1928)

- A **resource** can be described (*designated*) via **Metadata**
- Even if the resource itself cannot be delivered from the web server, probably its **representation** might be available that describes the resource sufficiently.



URI, Resource and Metadata

URI

<http://www.tour-eiffel.fr/index.html>

Resource:
Eiffel Tower

identifies

Representation

Metadata:

Content-type: text/html

Data:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD  
HTML 4.01 Transitional//EN"
```

```
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
```

```
<head>
```

```
<title>Le site officiel de la Tour Eiffel</title>
```

```
...
```

```
</html>
```

represents

(stands for)



describes
(designates)

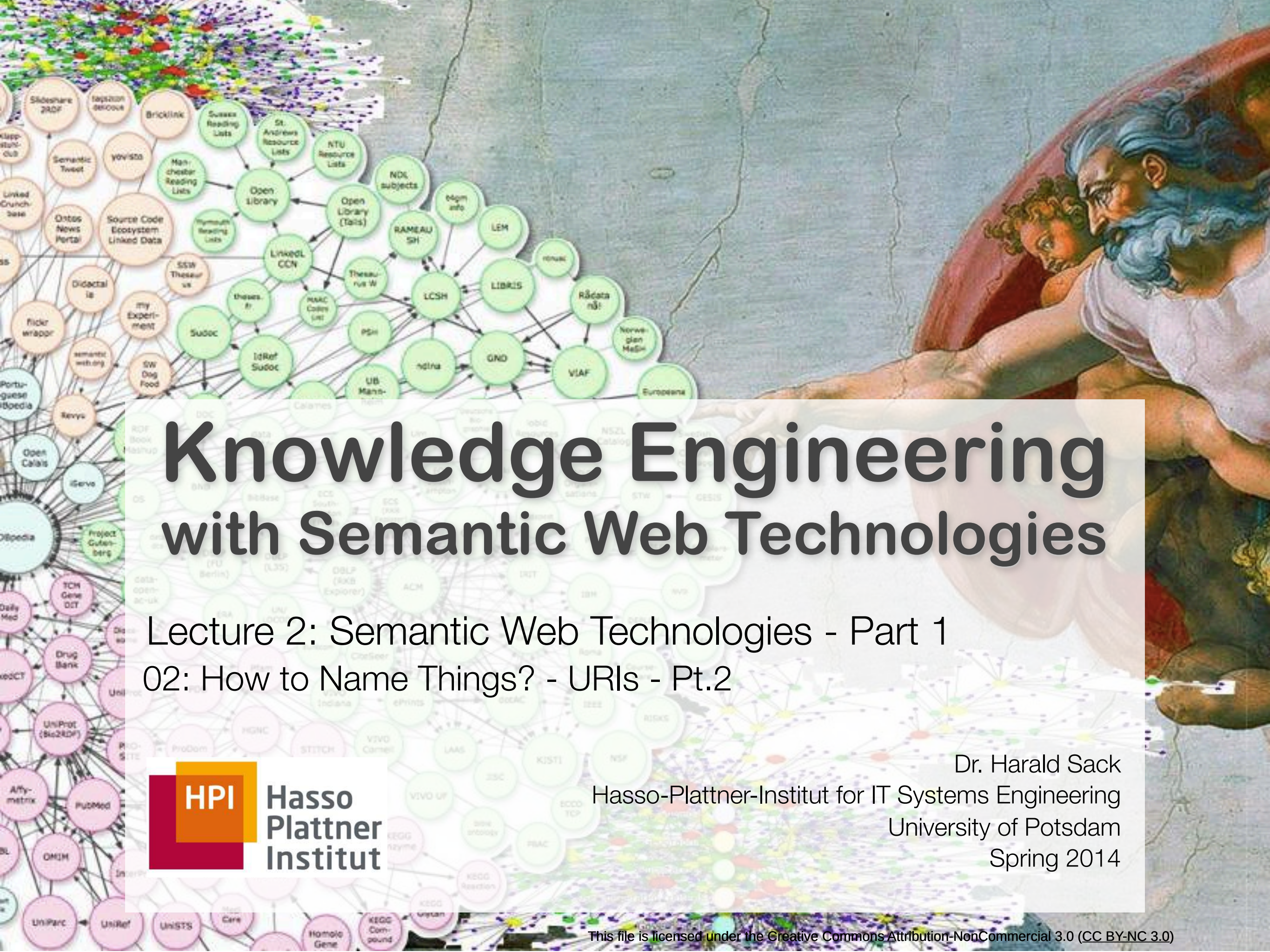
Resource:
Eiffel Tower
Metadata

(stands for)



02 How to Name Things? - URIs Pt.2

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I



Knowledge Engineering with Semantic Web Technologies

Lecture 2: Semantic Web Technologies - Part 1
02: How to Name Things? - URIs - Pt.2



Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014



Lecture 2: Semantic Web Technologies - Part 1

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies



02 How to Name Things? - URIs Pt.2

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I

Use URIs to name things on the Web

Resource: Eiffel Tower

URI

<http://www.tour-eiffel.fr/index.html>

identifies

(stands for)



5 ■ URI combines

□ **Address (Locator)**

- **Uniform Resource Locator** (URL, RFC 1738)
- denotes, **where** a resource can be found in the Web by stating its primary access mechanism
- might change during life cycle

5 ■ URI combines

□ **Address (Locator)**

- **Uniform Resource Locator** (URL, RFC 1738)
- denotes, **where** a resource can be found in the Web by stating its primary access mechanism
- might change during life cycle

□ **Identity (Name)**

- **Uniform Resource Name** (URN, RFC 2141)
- persistent identifier for a Web resource
- remains unchanged during life cycle

URI - Generic Syntax

- US-ASCII Encoding
- Percent Encoding for reserved characters, or characters that do not exist in US-ASCII encoding existieren

pct-encoded = "%" HEXDIG HEXDIG

- Reserved characters with special function

reserved = gen-delims / sub-delims

gen-delims = ":" / "/" / "?" / "#" / "[" / "]" / "@"

sub-delims = "!" / "\$" / "&" / "'" / "(" / ")" / "*" / "+" / "," / ";" / "="

- Permitted characters

unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"

- Extension to Universal Character Code (Unicode/ISO 10646)

- Internationalized Resource Identifier (IRI, RFC 3987)

e.g.:

http://كق#فغ=عظ;طض=صش?سز/رذ/دخ/حج.ثت.با

URI - Generic Syntax

URI = schema"://" [userinfo"@"] host[:port] [path] ["?" query] ["#" fragment]

URI - Generic Syntax

URI = schema"://"[userinfo"@"]host[:port][path]["?"query]["#"fragment]

- **schema:** e.g. http, ftp, mailto,...

URI - Generic Syntax

URI = schema"://"[userinfo"@"]host[:port][path]["?"query]["#"fragment]

- **schema**: e.g. http, ftp, mailto,...
- **userinfo**: e.g. username:password

URI - Generic Syntax

URI = schema"://[userinfo"@"]host[:port][path]["?"query]["#"fragment]

- **schema**: e.g. http, ftp, mailto,...
- **userinfo**: e.g. username:password
- **host**: Domain-Name, IPv4/IPv6 Address

URI - Generic Syntax

URI = schema"://"[userinfo"@"]host[:port][path]["?"query]["#"fragment]

- **schema**: e.g. http, ftp, mailto,...
- **userinfo**: e.g. username:password
- **host**: Domain-Name, IPv4/IPv6 Address
- **port**: e.g. :80 for standard http port

URI - Generic Syntax

URI = schema"://[userinfo"@"]host[:port][path]["?"query]["#"fragment]

- **schema**: e.g. http, ftp, mailto,...
- **userinfo**: e.g. username:password
- **host**: Domain-Name, IPv4/IPv6 Address
- **port**: e.g. :80 for standard http port
- **path**: path in file system of WWW server

URI - Generic Syntax

URI = schema"://"[userinfo"@"]host[:port][path]["?"query]["#"fragment]

- **schema**: e.g. http, ftp, mailto,...
- **userinfo**: e.g. username:password
- **host**: Domain-Name, IPv4/IPv6 Address
- **port**: e.g. :80 for standard http port
- **path**: path in file system of WWW server
- **query**: parameters to be passed over to applications

URI - Generic Syntax

URI = schema"://[userinfo"@"]host[:port][path]["?"query]["#"fragment]

- **schema**: e.g. http, ftp, mailto,...
- **userinfo**: e.g. username:password
- **host**: Domain-Name, IPv4/IPv6 Address
- **port**: e.g. :80 for standard http port
- **path**: path in file system of WWW server
- **query**: parameters to be passed over to applications
- **fragment**: determines a specific fragment of a document

What if a URI for a resource does not exist?

- Define a URI by yourself:
 - **avoid overlaps** → use your own website
 - enable **documentation at the same place** → Content Negotiation
- Use **separate URIs** for the resource (*Designatum*) and its documentation (*Designator*) via
 - Content Negotiation or
 - URI references (i.e. via "#" fragment identifier)

Content Negotiation

9

Let's try an example:

- I want to have **information** about the Eiffel Tower from DBpedia

http://dbpedia.org/resource/Eiffel_Tower

URI represents the *Designatum*

9

Let's try an example:

- I want to have **information** about the Eiffel Tower from DBpedia

http://dbpedia.org/resource/Eiffel_Tower

1

HTTP GET request

Accept Header: text/html

URI represents the *Designatum*

9

Let's try an example:

- I want to have **information** about the Eiffel Tower from DBpedia

http://dbpedia.org/resource/Eiffel_Tower

1

HTTP GET request

Accept Header: text/html

URI represents the *Designatum*

HTTP/1.1 303 See Other

2

http://dbpedia.org/page/Eiffel_Tower

Content Negotiation

9

Let's try an example:

- I want to have **information** about the Eiffel Tower from DBpedia

1

http://dbpedia.org/resource/Eiffel_Tower

HTTP GET request

Accept Header: text/html

URI represents the *Designatum*

HTTP/1.1 303 See Other

2

http://dbpedia.org/page/Eiffel_Tower

URI represents the *Designator*

Content Negotiation

9

Let's try an example:

- I want to have **information** about the Eiffel Tower from DBpedia

1

http://dbpedia.org/resource/Eiffel_Tower

HTTP GET request

Accept Header: text/html

URI represents the *Designatum*

HTTP/1.1 303 See Other

2

http://dbpedia.org/page/Eiffel_Tower

URI represents the *Designator*

3

HTTP GET request

Accept Header: text/html

Content Negotiation

9

Let's try an example:

- I want to have **information** about the Eiffel Tower from DBpedia

1

http://dbpedia.org/resource/Eiffel_Tower

HTTP GET request

Accept Header: text/html

URI represents the *Designatum*

HTTP/1.1 303 See Other

2

http://dbpedia.org/page/Eiffel_Tower

URI represents the *Designator*

3

HTTP GET request

Accept Header: text/html

HTML Document

4



Content Negotiation

Let's try an example:

- I want to have **machine readable information** about the Eiffel Tower from DBpedia

http://dbpedia.org/resource/Eiffel_Tower

URI represents the *Designatum*

Content Negotiation

Let's try an example:

- I want to have **machine readable information** about the Eiffel Tower from DBpedia

http://dbpedia.org/resource/Eiffel_Tower

1

HTTP GET request

Accept Header: `application/rdf+xml`

URI represents the *Designatum*

Content Negotiation

Let's try an example:

- I want to have **machine readable information** about the Eiffel Tower from DBpedia

http://dbpedia.org/resource/Eiffel_Tower

1

HTTP GET request

Accept Header: `application/rdf+xml`

URI represents the *Designatum*

HTTP/1.1 303 See Other 2

http://dbpedia.org/page/Eiffel_Tower

Content Negotiation

Let's try an example:

- I want to have **machine readable information** about the Eiffel Tower from DBpedia

http://dbpedia.org/resource/Eiffel_Tower

1

HTTP GET request

Accept Header: `application/rdf+xml`

URI represents the *Designatum*

HTTP/1.1 303 See Other 2

http://dbpedia.org/page/Eiffel_Tower

URI represents the *Designator*

Content Negotiation

Let's try an example:

- I want to have **machine readable information** about the Eiffel Tower from DBpedia

http://dbpedia.org/resource/Eiffel_Tower

1

HTTP GET request

Accept Header: `application/rdf+xml`

URI represents the *Designatum*

HTTP/1.1 303 See Other 2

http://dbpedia.org/page/Eiffel_Tower

3

HTTP GET request

Accept Header: `application/rdf+xml`

URI represents the *Designator*

Content Negotiation

Let's try an example:

- I want to have **machine readable information** about the Eiffel Tower from DBpedia

http://dbpedia.org/resource/Eiffel_Tower

1

HTTP GET request

Accept Header: `application/rdf+xml`

URI represents the *Designatum*

HTTP/1.1 303 See Other 2

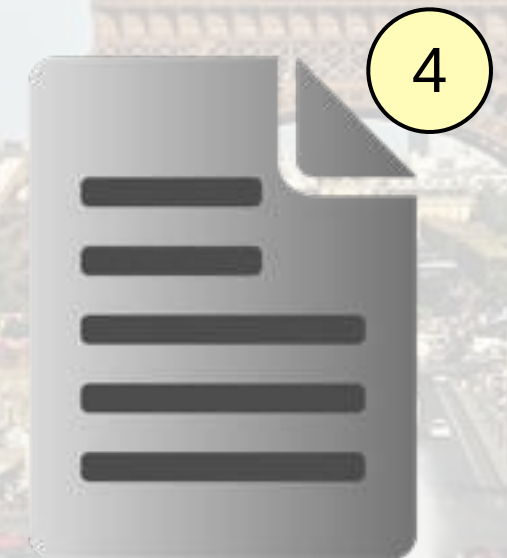
http://dbpedia.org/page/Eiffel_Tower

URI represents the *Designator*

3

HTTP GET request

Accept Header: `application/rdf+xml`



RDF Document

Fragment Identifier

11 Let's try another example:

- I want to have **machine readable information** myself

e.g.: http://hpi.uni-potsdam.de/harald_foaf.rdf

Designator



FOAF Document
with information
about me

http://hpi.uni-potsdam.de/harald_foaf.rdf#me

identifies me

Designatum

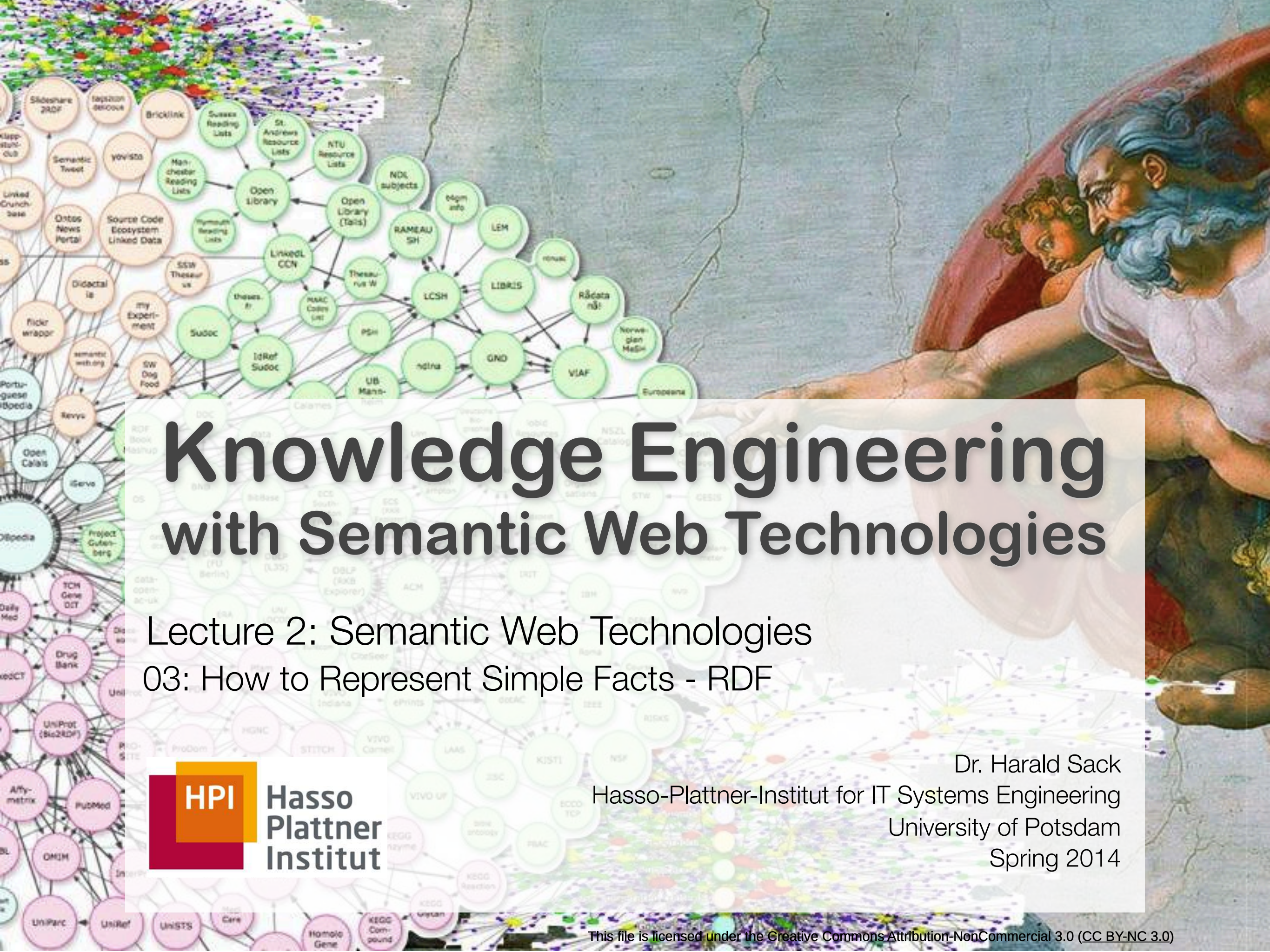


Next section

12

03 How to Represent Simple Facts - RDF

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I



Knowledge Engineering with Semantic Web Technologies

Lecture 2: Semantic Web Technologies
03: How to Represent Simple Facts - RDF



Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014



Lecture 2: Semantic Web Technologies - Part 1

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies

Introduction

03 How to Represent Simple Facts - RDF

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I

2. Semantic Web Technologies

Part 1

The Semantic Web Technology Stack (not a piece of cake...)

Most apps use only a subset of the stack

Querying allows fine-grained data access

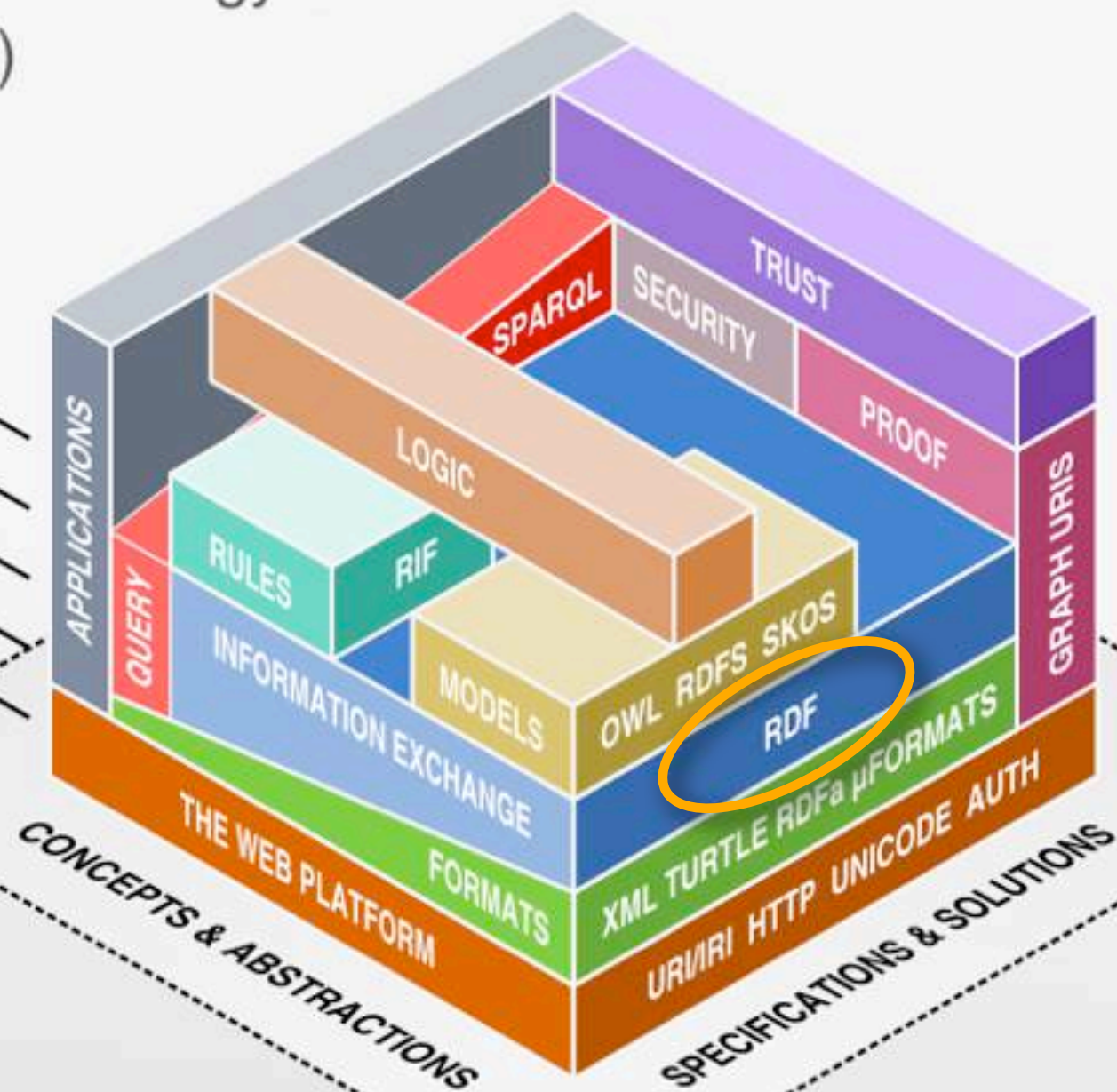
Standardized information exchange is key

Formats are necessary, but not too important

The Semantic Web is based on the Web

Linked Data uses a small
selection of technologies

LINKED DATA



Knowledge Representation

... a simple example



■ How do I represent

„Harald Sack has the phone number ++49 (331) 5509-927“ in XML?

Knowledge Representation

... a simple example

■ How do I represent

„Harald Sack has the phone number ++49 (331) 5509-927“ in XML?

```
<phonenr>  
  <owner>Harald Sack</owner>  
  <number>++49 (331) 5509-927</number>  
</phonenr>
```


Knowledge Representation

... a simple example

■ How do I represent

„Harald Sack has the phone number ++49 (331) 5509-927“ in XML?

```
<phonenr>  
  <owner>Harald Sack</owner>  
  <number>++49 (331) 5509-927</number>  
</phonenr>
```

```
<person name="Harald Sack">  
  <phonenr>49 (331) 5509-927</phonenr>  
</person>
```


Knowledge Representation

... a simple example

■ How do I represent

„Harald Sack has the phone number ++49 (331) 5509-927“ in XML?

```
<phonenr>  
  <owner>Harald Sack</owner>  
  <number>++49 (331) 5509-927</number>  
</phonenr>
```

```
<person name="Harald Sack">  
  <phonenr>49 (331) 5509-927</phonenr>  
</person>
```

```
<person name="Harald Sack" phone="49 (331) 5509-927" />
```


Knowledge Representation

... a simple example

■ How do I represent

„Harald Sack has the phone number ++49 (331) 5509-927“ in XML?

```
<phonenr>  
  <owner>Harald Sack</owner>  
  <number>++49 (331) 5509-927</number>  
</phonenr>
```

```
<person name="Harald Sack">  
  <phonenr>49 (331) 5509-927</phonenr>  
</person>
```

```
<person name="Harald Sack" phone="49 (331) 5509-927" />
```

is there a unique (intuitive) way to model knowledge (in XML)?

Knowledge Representation

... with XML?

- Successful communication requires translation from one XML serialization (Schema A) to another XML serialization (Schema B)

```
<phonenr>  
  <owner>Harald Sack</owner>  
  <number>++49 (331) 5509-927</number>  
</phonenr>
```

XML Schema A

XML Schema B

```
<person name="Harald Sack">  
  <phonenr>49 (331) 5509-927</phonenr>  
</person>
```

XSLT Transformation

```
<xsl:stylesheet version="1.0"  
  xmlns:xsl="http://...Transform"  
  <xsl:template match="/">  
    ....  
  </xsl:template>  
</xsl:stylesheet>
```


Knowledge Representation

... a simple example

■ How do I represent:

„Harald Sack has the phone number ++49 (331) 5509-927“

Harald Sack

Knowledge Representation

... a simple example

■ How do I represent:

„Harald Sack has the phone number ++49 (331) 5509-927“

Harald Sack

has the phone number

Knowledge Representation

... a simple example

■ How do I represent:

„Harald Sack has the phone number ++49 (331) 5509-927“

Harald Sack

has the phone number

++49 (331) 5509-927

Knowledge Representation

... a simple example

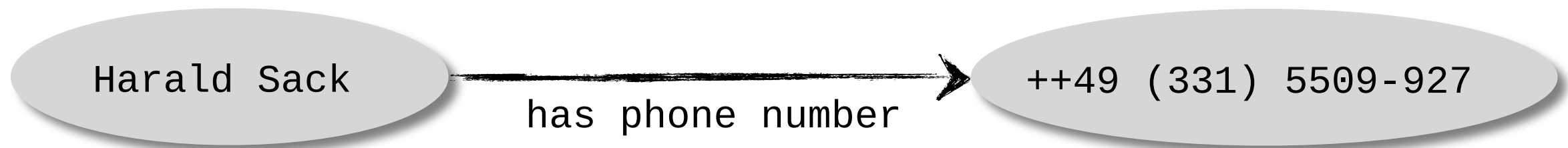


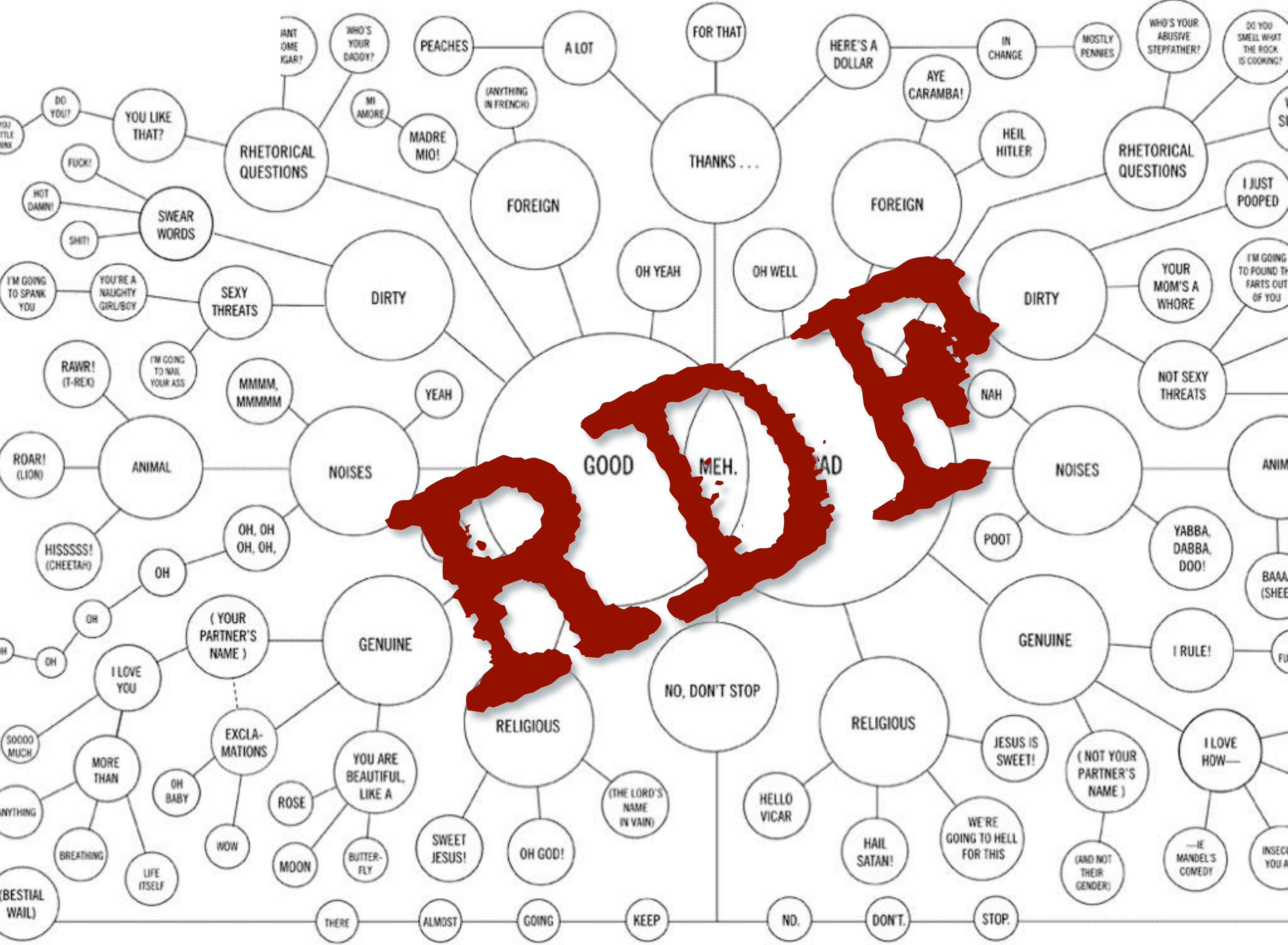
- How do I represent:
„Harald Sack has the phone number ++49 (331) 5509-927“
- Intuitive knowledge representation with **directed graphs**

Knowledge Representation

... a simple example

- How do I represent:
„Harald Sack has the phone number ++49 (331) 5509-927“
- Intuitive knowledge representation with **directed graphs**







Resource Description Framework

Resource

- can be everything (in principle)...
- must be uniquely identified and be referencable
- ...simply via URI



Resource Description Framework

Description

- = Descriptions of resources
- ...via representing properties and relationships among resources
- ...relationships can be represented as graphs



Resource Description Framework

Framework

- = Combination of web based protocols (URI, HTTP, XML,...)
- based on formal model (semantics)
- defines all allowed relationships among resources



Resource Description Framework

Harald Sack

has phone number

++49 (331) 5509-927

- Knowledge in RDF is expressed as a **list of statements**
- all RDF statements follow the same simple schema



Resource Description Framework

Harald Sack **Subject**

has phone number **Property**

++49 (331) 5509-927 **Object**



Resource Description Framework

Harald Sack **Subject**

→ **URI**

has phone number **Property** → **URI**

++49 (331) 5509-927 **Object** → **URI/Literal**



Resource Description Framework

- **Statements (RDF-Triple):**

Subject + Property + Object / Value

URI

URI

URI / Literal



Resource Description Framework

Constituents of the RDF-Graph

- **URI:**
 - to reference **resources** uniquely

`http://harald.sack.de/foaf.rdf#harald`

Subject

`http://harald.blogspot.com/`

Object

`http://xmlns.com/foaf/0.1/weblog`
Property



Resource Description Framework

Constituents of the RDF-Graph

- **Literals:**

- describe **data values** that do not have a separate existence
- strings, interpretation via datatype

`http://harald.sack.de/foaf.rdf#harald`

S

`++49-331-5509-927`

O

`http://xmlns.com/foaf/0.1/phone`

P



Resource Description Framework

Constituents of the RDF-Graph

- **Literals:**

- typed literals can be expressed via **XML Schema datatypes**
- Namespace for typed literals: `http://www.w3.org/2001/XMLSchema#`
 - Example: `"Semantics"^^<http://www.w3.org/2001/XMLSchema#string>`
- Language Tags denote the (natural) language of the text:
 - Example: `"Semantik"@de` , `"Semantics"@en`

Core types	<u>xsd:string</u>	Character strings
	<u>xsd:boolean</u>	true, false
	<u>xsd:decimal</u>	Arbitrary-precision decimal numbers
	<u>xsd:integer</u>	Arbitrary-size integer numbers
IEEE floating-point numbers	<u>xsd:double</u>	64-bit floating point numbers incl. $\pm\text{Inf}$, ± 0 , NaN
	<u>xsd:float</u>	32-bit floating point numbers incl. $\pm\text{Inf}$, ± 0 , NaN
Time and date	<u>xsd:date</u>	Dates (yyyy-mm-dd) with or without timezone
	<u>xsd:time</u>	Times (hh:mm:ss.sss...) with or without timezone
	<u>xsd:dateTime</u>	Date and time with or without timezone
	<u>xsd:dateTimeStamp</u>	Date and time with required timezone
Recurring and partial dates	<u>xsd:gYear</u>	Gregorian calendar year
	<u>xsd:gMonth</u>	Gregorian calendar month
	<u>xsd:gDay</u>	Gregorian calendar day of the month
	<u>xsd:gYearMonth</u>	Gregorian calendar year and month
	<u>xsd:gMonthDay</u>	Gregorian calendar month and day
	<u>xsd:duration</u>	Duration of time
	<u>xsd:yearMonthDuration</u>	Duration of time (months and years only)
	<u>xsd:dayTimeDuration</u>	Duration of time (days, hours, minutes, seconds only)
Limited-range	<u>xsd:byte</u>	-128...+127 (8 bit)
	<u>xsd:short</u>	-32768...+32767 (16 bit)
	<u>xsd:int</u>	-2147483648...+2147483647 (32 bit)
	<u>xsd:long</u>	-9223372036854775808...+9223372036854775807 (64 bit)
	<u>xsd:unsignedByte</u>	0...255 (8 bit)
	<u>xsd:unsignedShort</u>	0...65535 (16 bit)

Partial dates	<u>xsd:gMonthDay</u>	Gregorian calendar month and day
	<u>xsd:duration</u>	Duration of time
	<u>xsd:yearMonthDuration</u>	Duration of time (months and years only)
	<u>xsd:dayTimeDuration</u>	Duration of time (days, hours, minutes, seconds only)
Limited-range integer numbers	<u>xsd:byte</u>	-128...+127 (8 bit)
	<u>xsd:short</u>	-32768...+32767 (16 bit)
	<u>xsd:int</u>	-2147483648...+2147483647 (32 bit)
	<u>xsd:long</u>	-9223372036854775808...+9223372036854775807 (64 bit)
	<u>xsd:unsignedByte</u>	0...255 (8 bit)
	<u>xsd:unsignedShort</u>	0...65535 (16 bit)
	<u>xsd:unsignedInt</u>	0...4294967295 (32 bit)
	<u>xsd:unsignedLong</u>	0...18446744073709551615 (64 bit)
	<u>xsd:positiveInteger</u>	Integer numbers >0
	<u>xsd:nonNegativeInteger</u>	Integer numbers ≥0
Encoded binary data	<u>xsd:negativeInteger</u>	Integer numbers <0
	<u>xsd:nonPositiveInteger</u>	Integer numbers ≤0
	<u>xsd:hexBinary</u>	Hex-encoded binary data
	<u>xsd:base64Binary</u>	Base64-encoded binary data
Miscellaneous XSD types	<u>xsd:anyURI</u>	Absolute or relative URIs and IRIs
	<u>xsd:language</u>	Language tags per [BCP47]
	<u>xsd:normalizedString</u>	Whitespace-normalized strings
	<u>xsd:token</u>	Tokenized strings
	<u>xsd:NMTOKEN</u>	XML NMTOKENs
	<u>xsd:Name</u>	XML Names
	<u>xsd:NCName</u>	XML NCNames

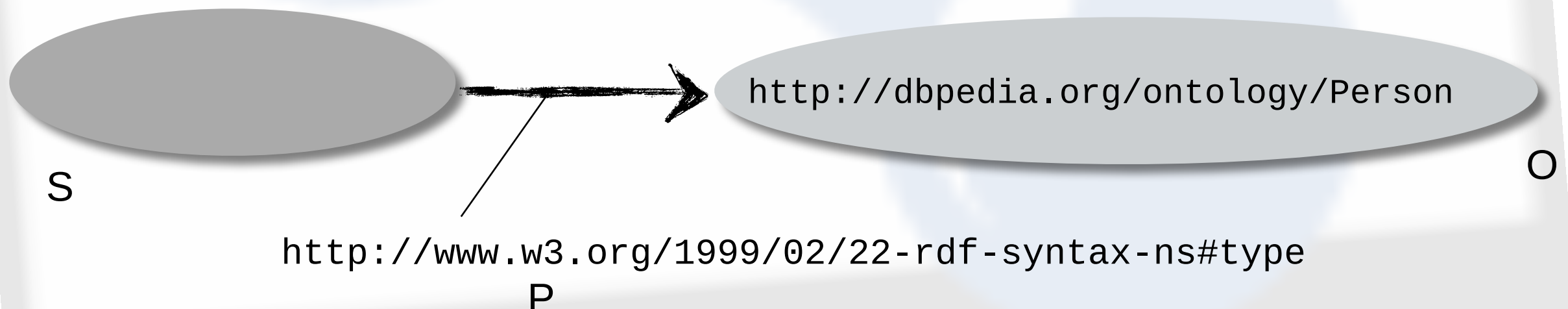


Resource Description Framework

Constituents of the RDF-Graph

- **Blank Nodes:**

- denote **existence of an individual** with specific attributes, but **without providing an identification or reference**





Resource Description Framework

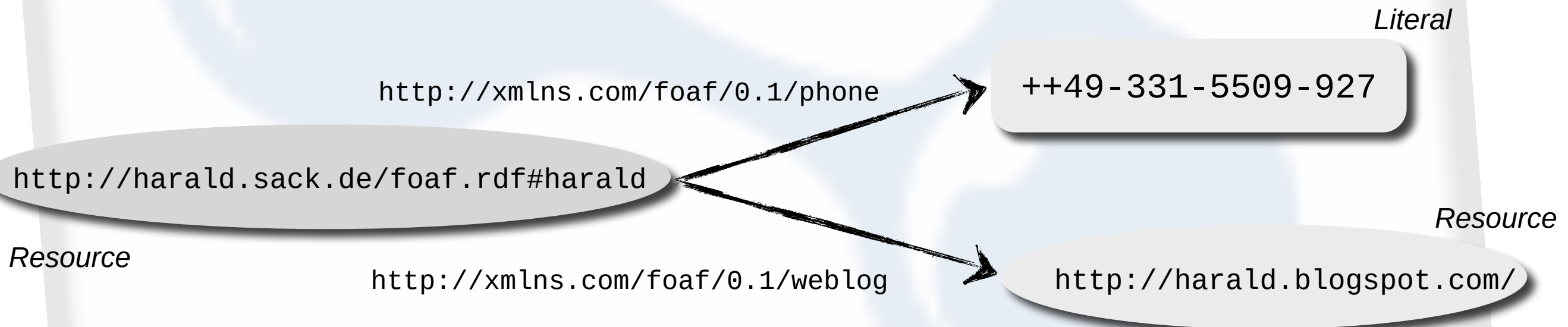
RDF Representations



Resource Description Framework

RDF Representations

- **Node-Edge-Node Triple**





Resource Description Framework

23

`http://xmlns.com/foaf/0.1/phone`

`++49-331-5509-927`

Literal

`http://harald.sack.de/foaf.rdf#harald`

Resource

`http://xmlns.com/foaf/0.1/weblog`

`http://harald.blogspot.com/`

Resource

- **N-Triples Notation**

- **URIs/IRIs** in angle brackets
- **Literals** in quotation marks
- **Triple** ends with a **period**

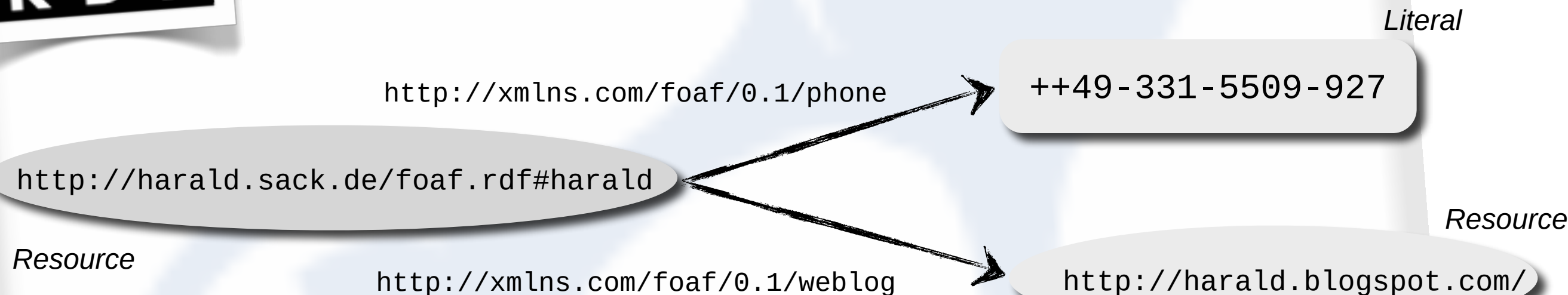
```
<http://harald.sack.de/foaf.rdf#harald> <http://xmlns.com/foaf/0.1/phone> "+49-331-5509-927" .
```

```
<http://harald.sack.de/foaf.rdf#harald> <http://xmlns.com/foaf/0.1/weblog>  
<http://harald.blogspot.com/> .
```




Resource Description Framework

24



- **Turtle (Terse RDF Tripel Language) Notation**

- extension of N-Triples

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@base <http://harald.sack.de/foaf.rdf>

<#harald> foaf:phone "++49-331-5509-927" .
<#harald> foaf:weblog <http://harald.blogspot.com/> .
```



Resource Description Framework

RDF Representations

- **RDF/XML Notation**

```
<xml version="1.0" encoding="utf-8">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/" >

  <rdf:Description rdf:about="http://harald.sack.de/foaf.rdf#harald">
    <foaf:phone>++49-331-5509-927</foaf:phone>
  </rdf:Description>

  <rdf:Description rdf:about="http://harald.sack.de/foaf.rdf#harald">
    <foaf:weblog>
      <rdf:Description rdf:about="http://haraldblogspot.com/"></rdf:Description>
    </foaf:weblog>
  </rdf:Description>
</rdf:RDF>
```




Resource Description Framework

RDF Representations

S

- **RDF/XML Notation**

```
<xml version="1.0" encoding="utf-8">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/" >

  <rdf:Description rdf:about="http://harald.sack.de/foaf.rdf#harald">
    <foaf:phone>++49-331-5509-927</foaf:phone>
  </rdf:Description>

  <rdf:Description rdf:about="http://harald.sack.de/foaf.rdf#harald">
    <foaf:weblog>
      <rdf:Description rdf:about="http://haraldblogspot.com/"></rdf:Description>
    </foaf:weblog>
  </rdf:Description>
</rdf:RDF>
```



Resource Description Framework

RDF Representations

S

P

- **RDF/XML Notation**

```
<xml version="1.0" encoding="utf-8">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/" >

  <rdf:Description rdf:about="http://harald.sack.de/foaf.rdf#harald">
    <foaf:phone>++49-331-5509-927</foaf:phone>
  </rdf:Description>

  <rdf:Description rdf:about="http://harald.sack.de/foaf.rdf#harald">
    <foaf:weblog>
      <rdf:Description rdf:about="http://haraldblogspot.com/"></rdf:Description>
    </foaf:weblog>
  </rdf:Description>
</rdf:RDF>
```




Resource Description Framework

RDF Representations

S

P

O

- **RDF/XML Notation**

```
<xml version="1.0" encoding="utf-8">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/" >

  <rdf:Description rdf:about="http://harald.sack.de/foaf.rdf#harald">
    <foaf:phone>++49-331-5509-927</foaf:phone>
  </rdf:Description>

  <rdf:Description rdf:about="http://harald.sack.de/foaf.rdf#harald">
    <foaf:weblog>
      <rdf:Description rdf:about="http://haraldblogspot.com/"></rdf:Description>
    </foaf:weblog>
  </rdf:Description>
</rdf:RDF>
```



Resource Description Framework

RDF Representations

- **JSON-LD Notation** (*RDF 1.1*)

```
{
  "http://harald.sack.de/foaf.rdf#harald" : {
    "http://xmlns.com/foaf/0.1/phone" : [ { "value" : "++49-331-5509-927",
                                             "type" : "literal" } ]
  }
  {
    "http://xmlns.com/foaf/0.1/weblog" : [ { "value" : "http://harald.blogspot.com/",
                                             "type" : "uri" } ]
  }
}
```




Resource Description Framework

RDF Representations

S

P

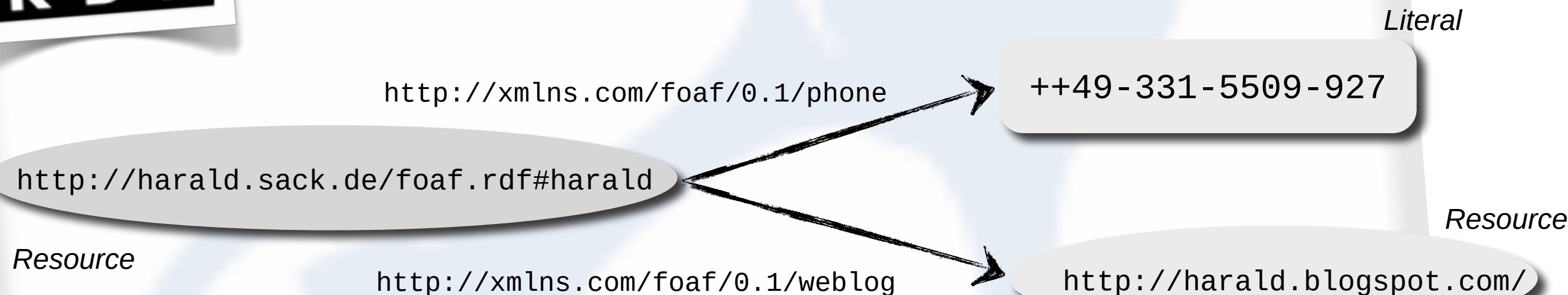
O

- **JSON-LD Notation** (*RDF 1.1*)

```
{  
  "http://harald.sack.de/foaf.rdf#harald": {  
    "http://xmlns.com/foaf/0.1/phone": [ { "value" : "++49-331-5509-927",  
                                           "type" : "literal" } ]  
  }  
  {  
    "http://xmlns.com/foaf/0.1/weblog": [ { "value" : "http://harald.blogspot.com/",  
                                           "type" : "uri" } ]  
  }  
}
```



Resource Description Framework



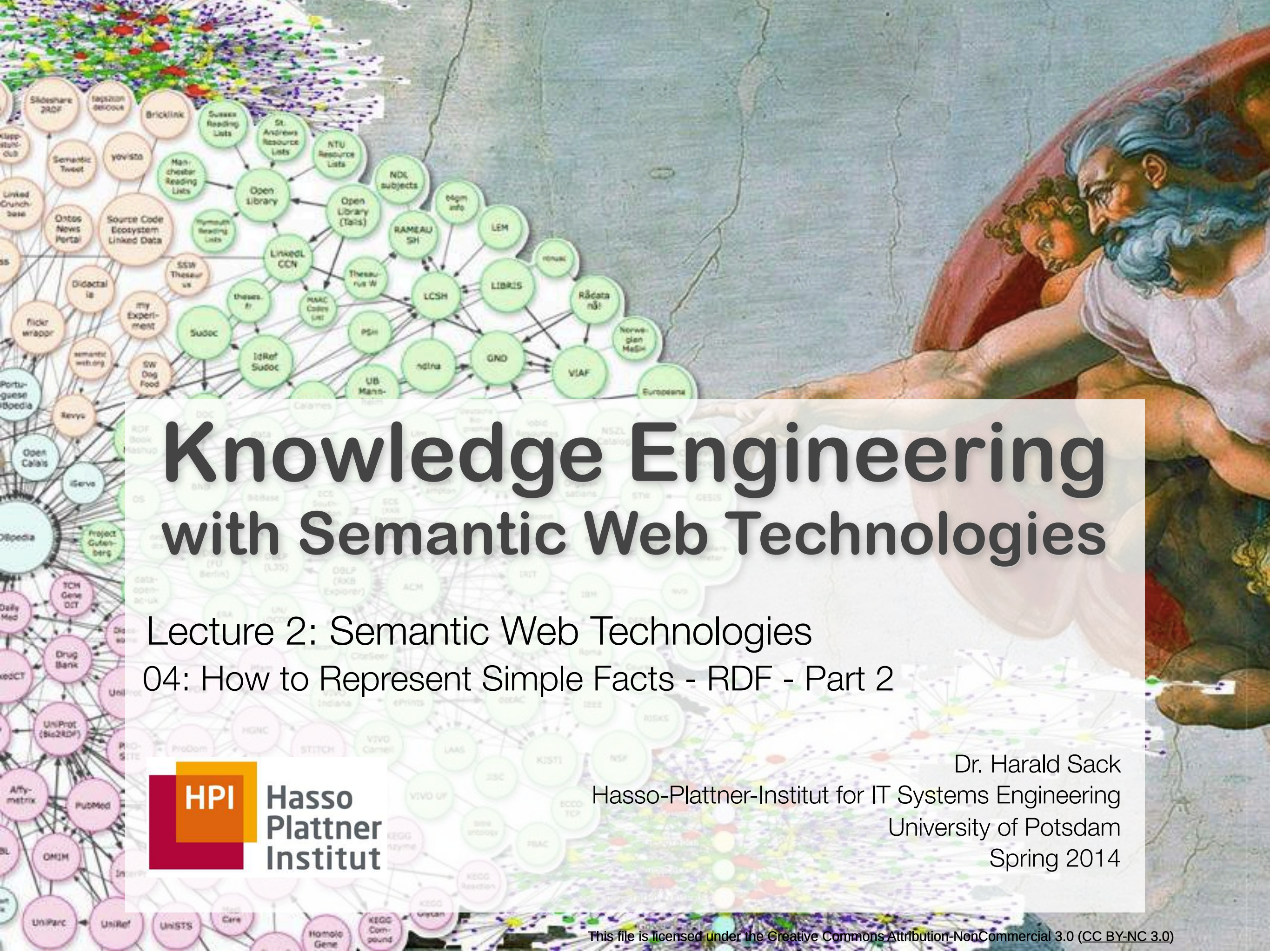
- **Turtle (Terse RDF Tripel Language) Notation**

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
@base <http://harald.sack.de/foaf.rdf> .  
  
<#harald> foaf:phone "++49-331-5509-927" .  
<#harald> foaf:weblog <http://harald.blogspot.com/> .
```


Simple Facts in RDF

04 How to Represent Simple Facts - RDF Pt.2

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I



Knowledge Engineering with Semantic Web Technologies

Lecture 2: Semantic Web Technologies

04: How to Represent Simple Facts - RDF - Part 2



Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014



Lecture 2: Semantic Web Technologies - Part 1

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies

3

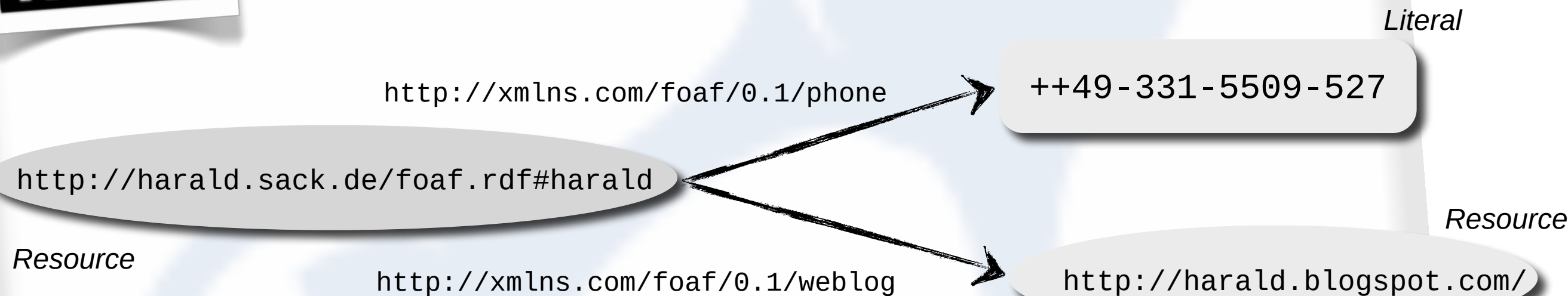
**Simple Facts in RDF
with Turtle**

04 How to Represent Simple Facts - RDF Pt.2

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I



Resource Description Framework



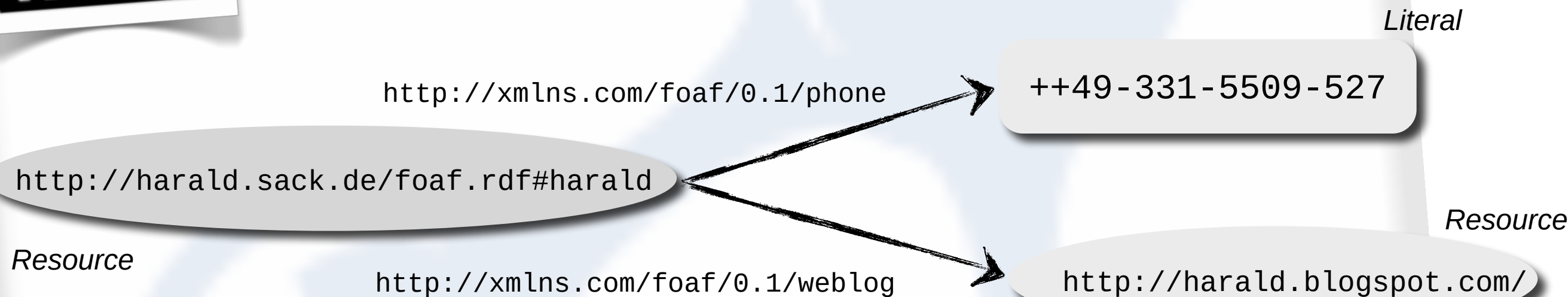
- **Turtle (Terse RDF Tripel Language) Notation**
- ...starting from N-Triples

```
<http://harald.sack.de/foaf.rdf#harald> <http://xmlns.com/foaf/0.1/phone> "+49-331-5509-527" .
```

```
<http://harald.sack.de/foaf.rdf#harald> <http://xmlns.com/foaf/0.1/weblog> <http://semweb2014.blogspot.com/> .
```



Resource Description Framework



```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

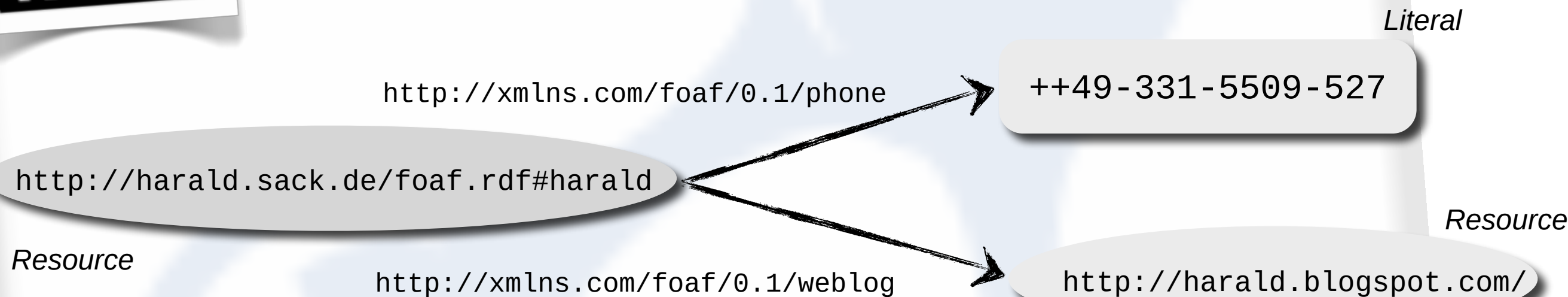
```
@base <http://harald.sack.de/foaf.rdf> .
```

```
<#harald> foaf:phone "++49-331-5509-527" .
```

```
<#harald> foaf:weblog <http://semweb2014.blogspot.com/> .
```




Resource Description Framework

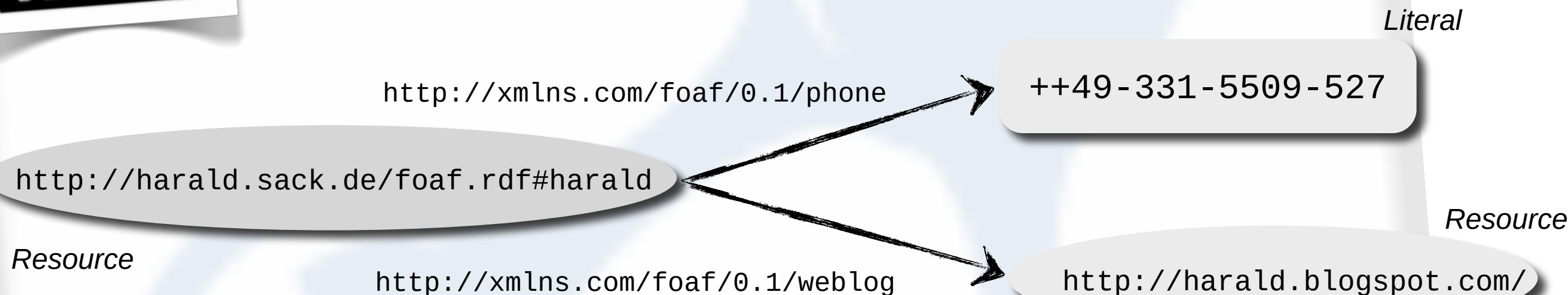


@prefix directive associates prefix-label with IRI

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
@base <http://harald.sack.de/foaf.rdf> .  
  
<#harald> foaf:phone "++49-331-5509-527" .  
<#harald> foaf:weblog <http://semweb2014.blogspot.com/> .
```



Resource Description Framework



@prefix directive associates prefix-label with IRI

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

@base <http://harald.sack.de/foaf.rdf> .

<#harald> **foaf:phone** "++49-331-5509-527" .

<#harald> **foaf:weblog** <http://semweb2014.blogspot.com/> .

usage of prefix label



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

`http://xmlns.com/foaf/0.1/phone`

`++49-331-5509-527`

Literal

`http://xmlns.com/foaf/0.1/weblog`

`http://harald.blogspot.com/`

Resource

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
@base <http://harald.sack.de/foaf.rdf> .
```

```
<#harald> foaf:phone "++49-331-5509-527" .
```

```
<#harald> foaf:weblog <http://semweb2014.blogspot.com/> .
```



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

`http://xmlns.com/foaf/0.1/phone`

`++49-331-5509-527`

Literal

`http://xmlns.com/foaf/0.1/weblog`

`http://harald.blogspot.com/`

Resource

@base directive provides IRI to complement all relative IRIs

`@prefix foaf: <http://xmlns.com/foaf/0.1/> .`

`@base <http://harald.sack.de/foaf.rdf> .`

`<#harald> foaf:phone "++49-331-5509-527" .`

`<#harald> foaf:weblog <http://semweb2014.blogspot.com/> .`



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

`http://xmlns.com/foaf/0.1/phone`

`++49-331-5509-527`

Literal

`http://xmlns.com/foaf/0.1/weblog`

`http://harald.blogspot.com/`

Resource

@base directive provides IRI to complement all relative IRIs

`@prefix foaf: <http://xmlns.com/foaf/0.1/> .`

`@base <http://harald.sack.de/foaf.rdf> .`

`<#harald> foaf:phone "++49-331-5509-527" .`

`<#harald> foaf:weblog <http://semweb2014.blogspot.com/> .`

example for relative IRI in RDF



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

`http://xmlns.com/foaf/0.1/phone`

`++49-331-5509-527`

Literal

`http://xmlns.com/foaf/0.1/weblog`

`http://harald.blogspot.com/`

Resource

file `http://harald.sack.de/foaf.rdf`

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
<#harald> foaf:phone "++49-331-5509-527" .
```

```
<#harald> foaf:weblog <http://semweb2014.blogspot.com/> .
```




Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

`http://xmlns.com/foaf/0.1/phone`

`++49-331-5509-527`

Literal

`http://xmlns.com/foaf/0.1/weblog`

`http://harald.blogspot.com/`

Resource

file `http://harald.sack.de/foaf.rdf`

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
<#harald> foaf:phone "++49-331-5509-527" .
```

```
<#harald> foaf:weblog <http://semweb2014.blogspot.com/> .
```

if there is no @base directive, relative URIs/IRIs will be complemented by the document's own URI/IRI



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

`http://xmlns.com/foaf/0.1/phone`

`++49-331-5509-527`

Literal

`http://xmlns.com/foaf/0.1/weblog`

`http://harald.blogspot.com/`

Resource

`file http://harald.sack.de/foaf.rdf`

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
<#harald> foaf:phone "++49-331-5509-527" .
```

```
<#harald> foaf:weblog <http://semweb2014.blogspot.com/> .
```

if there is no @base directive, relative URIs/IRIs will be complemented by the document's own URI/IRI



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

`http://xmlns.com/foaf/0.1/phone`

`++49-331-5509-527`

Literal

`http://xmlns.com/foaf/0.1/weblog`

`http://harald.blogspot.com/`

Resource

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
@base <http://harald.sack.de/foaf.rdf> .
```

```
<#harald> foaf:phone "++49-331-5509-527" ;  
          foaf:weblog <http://semweb2014.blogspot.com/> .
```



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

`http://xmlns.com/foaf/0.1/phone`

`++49-331-5509-527`

Literal

`http://xmlns.com/foaf/0.1/weblog`

`http://harald.blogspot.com/`

Resource

semicolon indicates that subsequent
triples have the same subject
(**predicate list**)

`@prefix foaf: <http://xmlns.com/foaf/0.1/> .`

`@base <http://harald.sack.de/foaf.rdf> .`

`<#harald> foaf:phone "++49-331-5509-527";
foaf:weblog <http://semweb2014.blogspot.com/> .`



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

`http://xmlns.com/foaf/0.1/phone`

`++49-331-5509-527`

Literal

`http://xmlns.com/foaf/0.1/weblog`

`http://harald.blogspot.com/`

Resource

semicolon indicates that subsequent
triples have the same subject
(**predicate list**)

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

@base <http://harald.sack.de/foaf.rdf> .

<#harald> foaf:phone "++49-331-5509-527";
foaf:weblog <http://semweb2014.blogspot.com/> .

subject of subsequent triples can be left out



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

`foaf:weblog`

`http://semweb2014.blogspot.com/`

`foaf:weblog`

`http://semweb2013.blogspot.com/`

`foaf:weblog`

`http://semweb2012.blogspot.com/`

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
<#harald> foaf:weblog <http://semweb2014.blogspot.com/> ,  
                  <http://semweb2013.blogspot.com/> ,  
                  <http://semweb2012.blogspot.com/> .
```

file `http://harald.sack.de/foaf.rdf`



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

foaf:weblog

`http://semweb2014.blogspot.com/`

foaf:weblog

`http://semweb2013.blogspot.com/`

foaf:weblog

`http://semweb2012.blogspot.com/`

comma indicates that subsequent triples have same subject and property (**object list**)

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
<#harald> foaf:weblog <http://semweb2014.blogspot.com/> ,
```

```
<http://semweb2013.blogspot.com/> ,
```

```
<http://semweb2012.blogspot.com/> .
```

file `http://harald.sack.de/foaf.rdf`



Resource Description Framework

`http://harald.sack.de/foaf.rdf#harald`

Resource

foaf:weblog

`http://semweb2014.blogspot.com/`

foaf:weblog

`http://semweb2013.blogspot.com/`

foaf:weblog

`http://semweb2012.blogspot.com/`

comma indicates that subsequent triples have same subject and property (**object list**)

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
<#harald> foaf:weblog <http://semweb2014.blogspot.com/> ,  
               <http://semweb2013.blogspot.com/> ,  
               <http://semweb2012.blogspot.com/> .
```

file `http://harald.sack.de/foaf.rdf`

subject and property of subsequent triples are the same



Resource Description Framework

`http://hpi-web.de/Lecture#name`

Knowledge Engineering

`http://hpi-web.de/Spring14#KE`

`http://hpi-web.de/Lecture#hours`

4

- **Typed Literals**

```
@prefix lec: <http://hpi-web.de/Lecture#> .
```

```
<http://hpi.web.de/Spring14#KE>
```

```
lec:name "Knowledge Engineering"^^<http://www.w3c.org/2001/XMLSchema#string> ;
```

```
lec:hours "4"^^<http://www.w3c.org/2001/XMLSchema#integer> .
```



Resource Description Framework

`http://hpi-web.de/Lecture#name`

Knowledge
Engineering

`http://hpi-web.de/Spring14#KE`

`http://hpi-web.de/Lecture#hours`

4

- **Typed Literals**

```
@prefix lec: <http://hpi-web.de/Lecture#> .
```

```
<http://hpi.v  
lec:na  
lec:h
```

```
@prefix lv: <http://hpi-web.de/Lecture#> .
```

```
<http://hpi.web.de/Spring14#KE> lec:name "Semantic Web Technologies" ;  
lec:hours 4 .
```




Resource Description Framework

Multi Valued Relations and Blank Nodes

- Example:
 - A lecture takes place twice a week in two different rooms
 - How to model this in RDF?

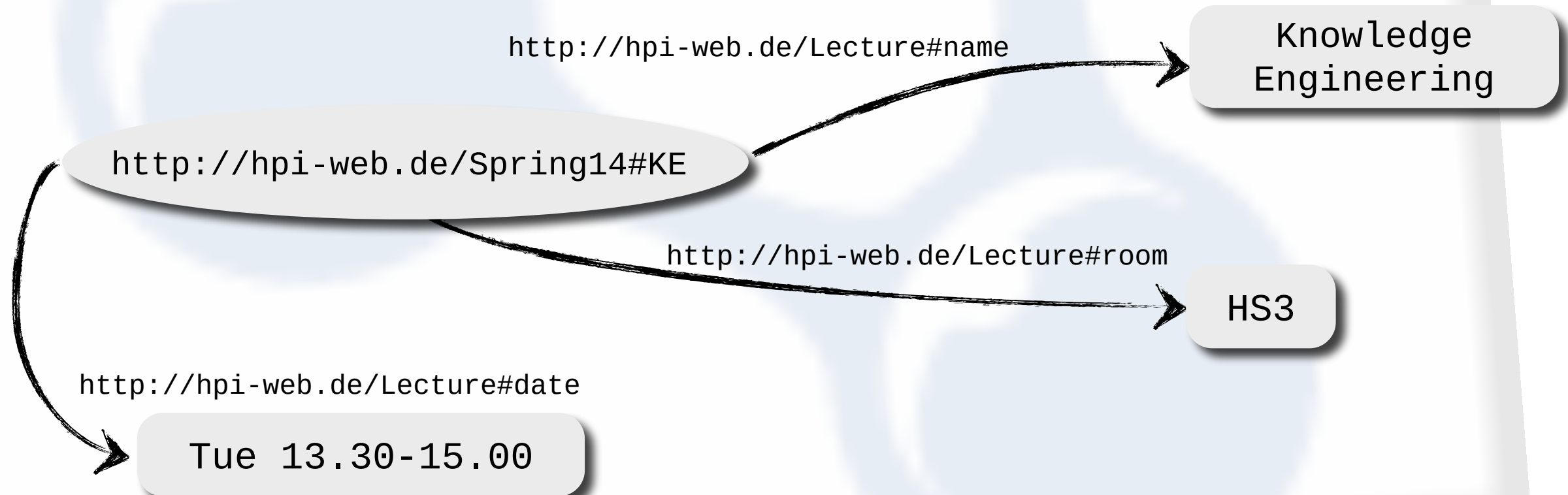




Resource Description Framework

Multi Valued Relations and Blank Nodes

- Example:
 - A lecture takes place twice a week in two different rooms
 - How to model this in RDF?

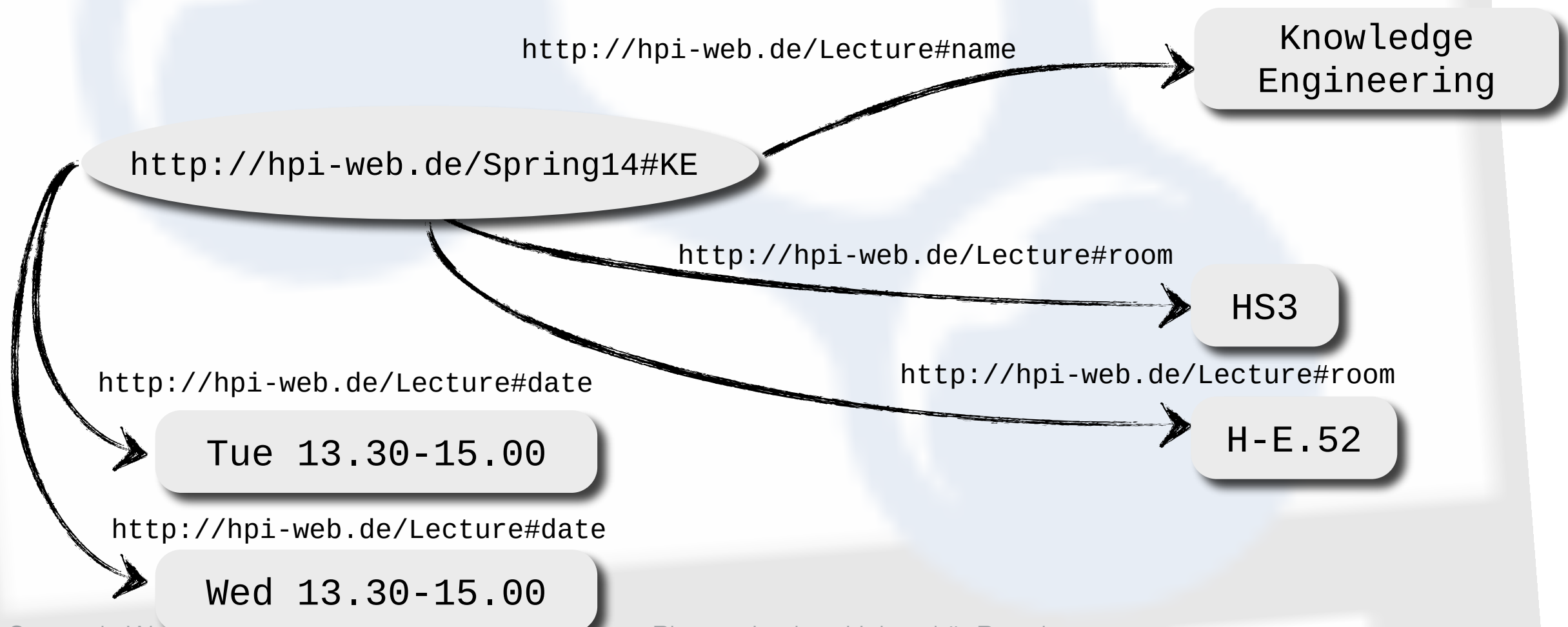




Resource Description Framework

Multi Valued Relations and Blank Nodes

- Example:
 - A lecture takes place twice a week in two different rooms
 - How to model this in RDF?

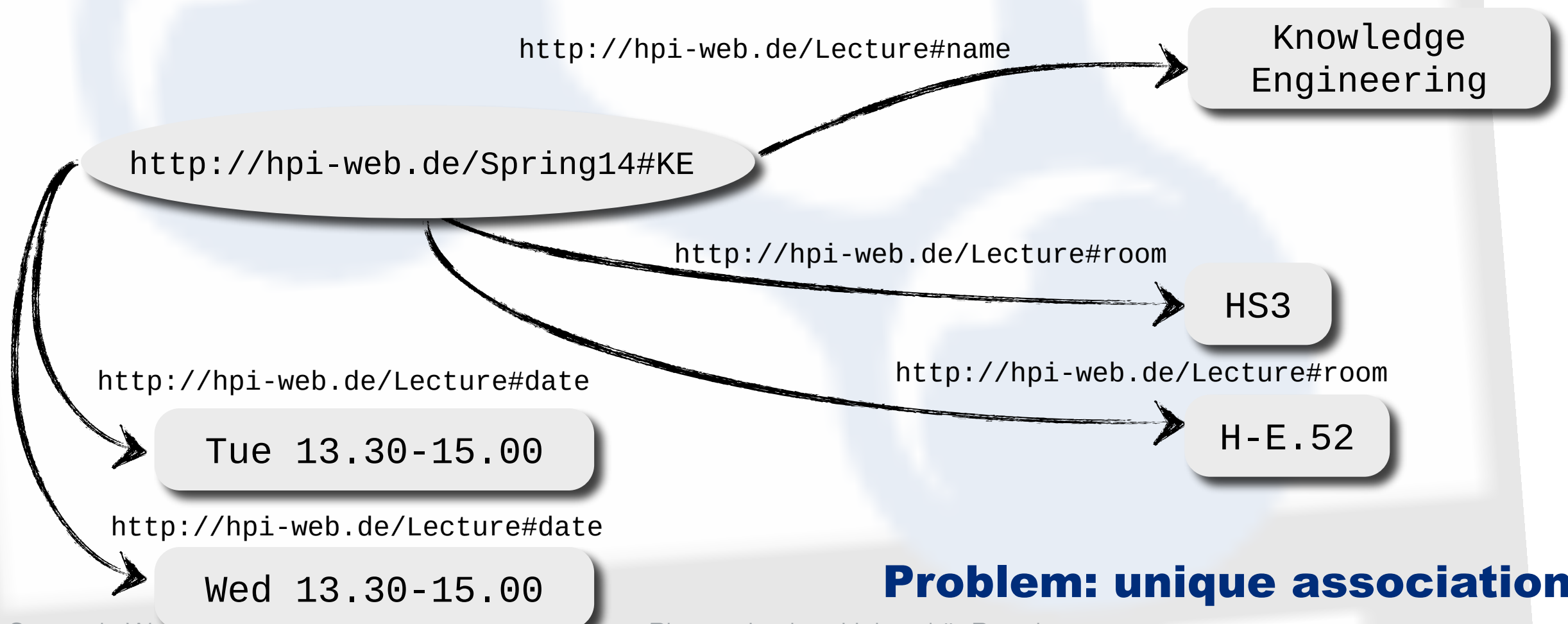




Resource Description Framework

Multi Valued Relations and Blank Nodes

- Example:
 - A lecture takes place twice a week in two different rooms
 - How to model this in RDF?

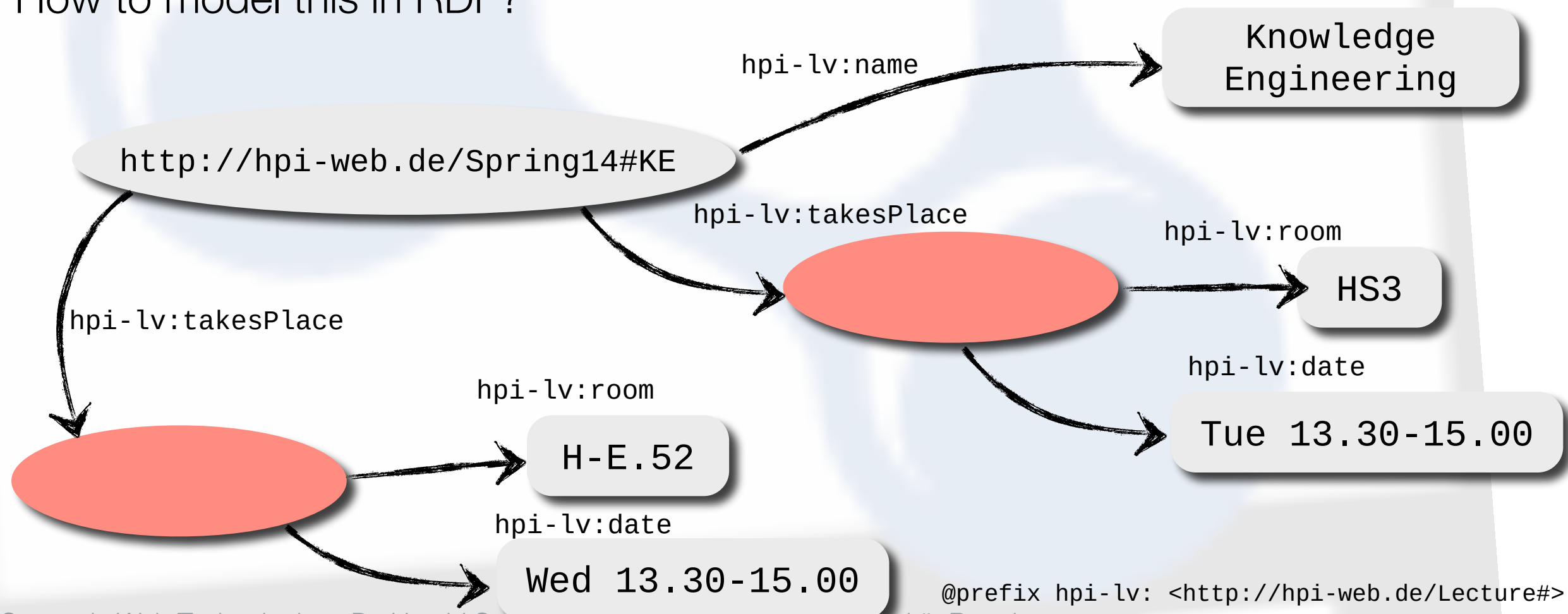




Resource Description Framework

Multi Valued Relations and Blank Nodes

- Example:
 - A lecture takes place twice a week in two different rooms
 - How to model this in RDF?

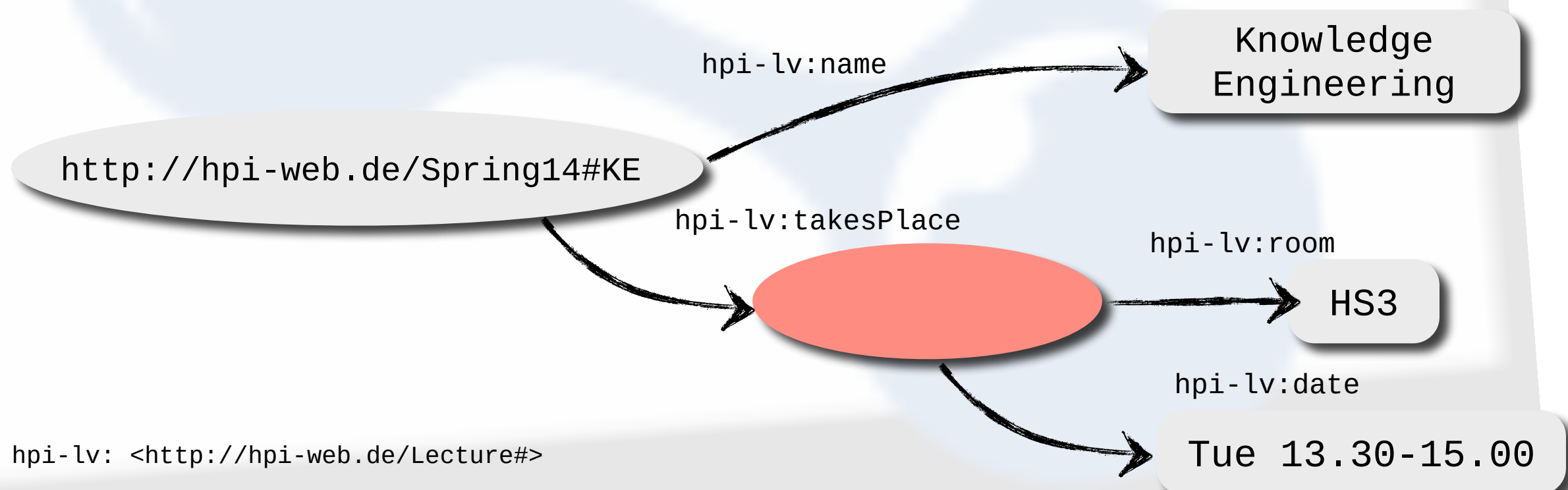




Resource Description Framework

Blank Nodes

- Blank Nodes (Bnodes) can be introduced to represent multi valued relationships
- Blank Nodes can be introduced for resources that don't need a name (auxiliary nodes)

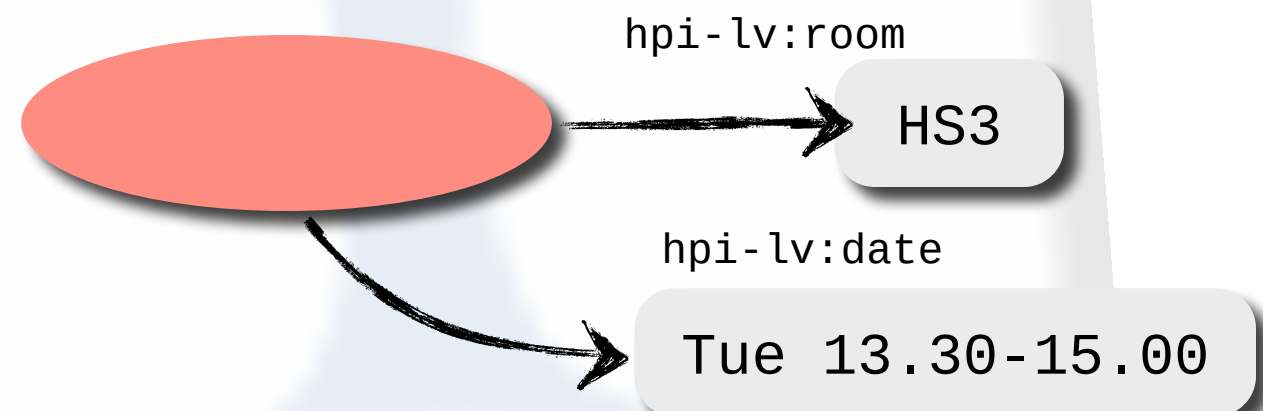


@prefix hpi-lv: <http://hpi-web.de/Lecture#>



Resource Description Framework

Anonymous Blank Nodes



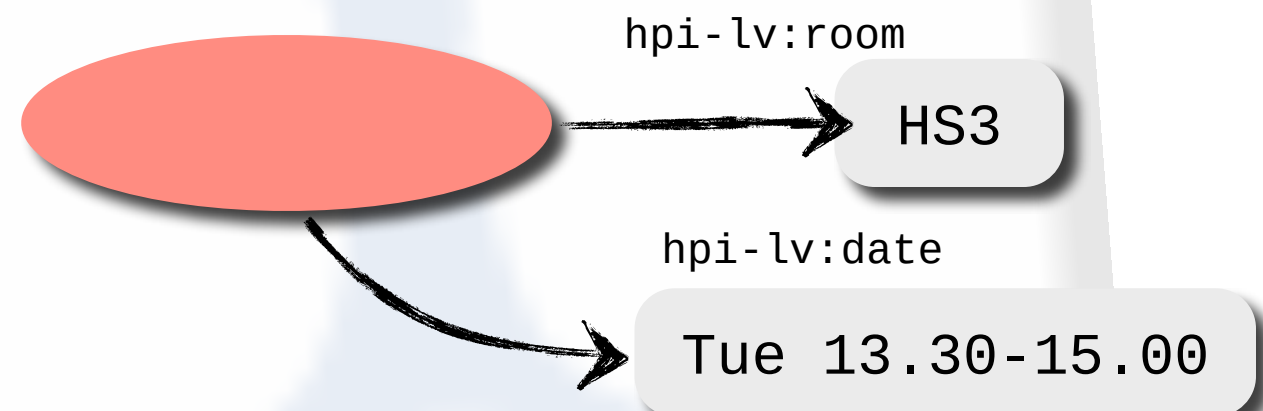
```
@prefix hpi-lv: <http://hpi-web.de/Lecture#> .
```

```
[ ] hpi-lv:date "Tue 13.30-15.00";  
    hpi-lv:room "HS3" .
```



Resource Description Framework

Anonymous Blank Nodes



```
@prefix hpi-lv: <http://hpi-web.de/Lecture#> .
```

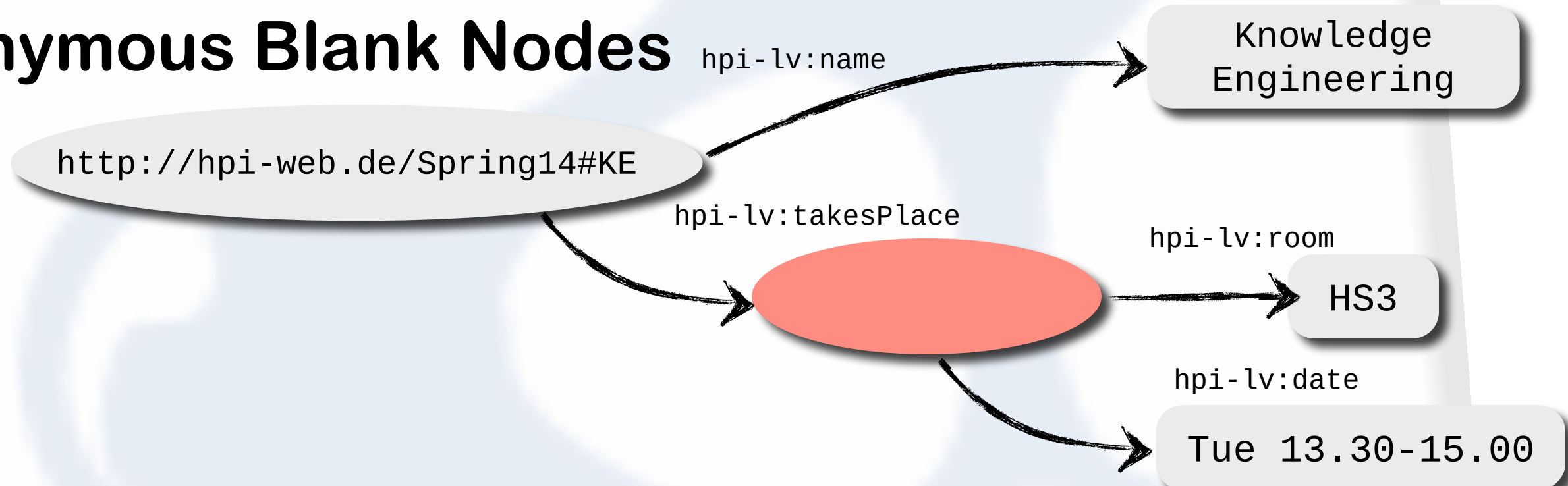
```
[ ] hpi-lv:date "Tue 13.30-15.00";  
    hpi-lv:room "HS3" .
```

anonymous blank node



Resource Description Framework

Anonymous Blank Nodes



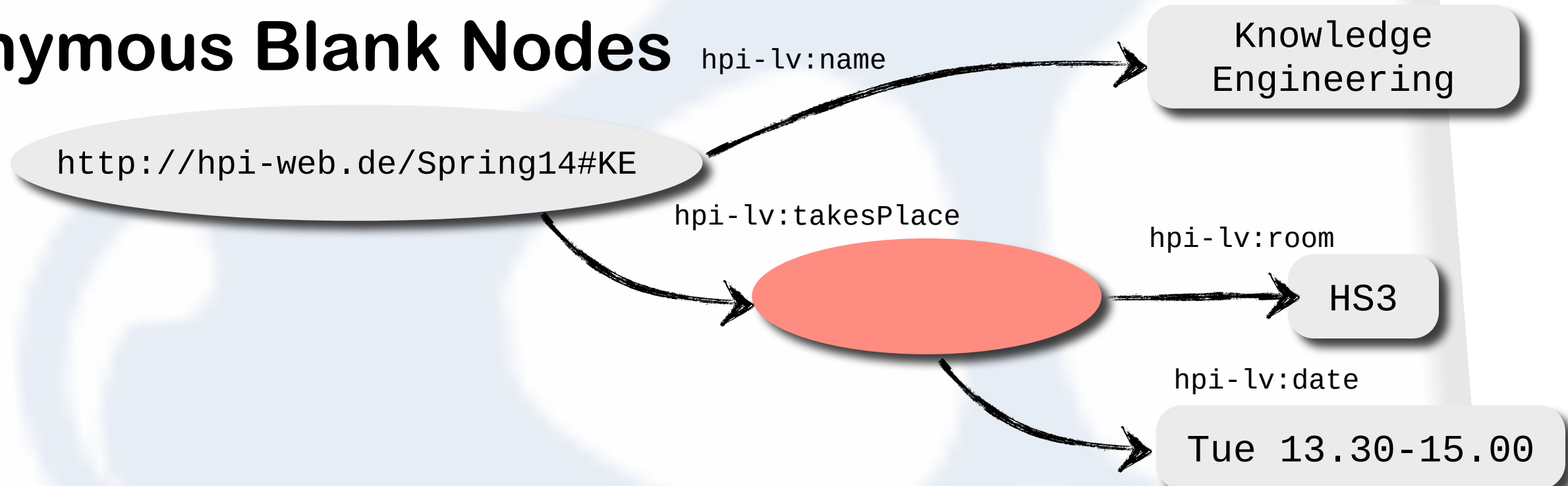
@prefix hpi-lv: <http://hpi-web.de/Lecture#>.

```
<http://hpi-web.de/Spring14#KE> hpi-lv:name "Knowledge Engineering";  
                                hpi-lv:takesPlace [  
                                    hpi-lv:date "Tue 13.30-15.00";  
                                    hpi-lv:room "HS3" ] .
```



Resource Description Framework

Anonymous Blank Nodes



@prefix hpi-lv: <http://hpi-web.de/Lecture#>.

```
<http://hpi-web.de/Spring14#KE> hpi-lv:name "Knowledge Engineering";  
hpi-lv:takesPlace [  
  hpi-lv:date "Tue 13.30-15.00";  
  hpi-lv:room "HS3" ] .
```

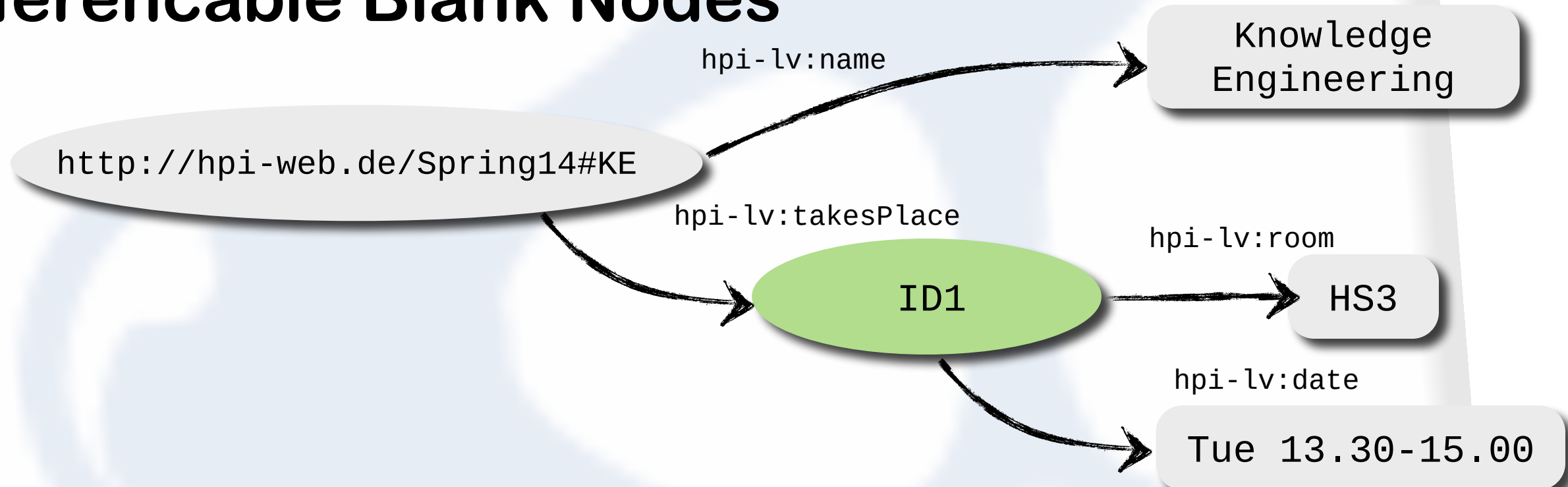
nested anonymous blank node



Resource Description Framework

16

Dereferencable Blank Nodes



@prefix hpi-lv: <http://hpi-web.de/Lecture#>.

<http://hpi-web.de/Spring14#KE> hpi-lv:name "Knowledge Engineering";
hpi-lv:takesPlace **_:ID1** .

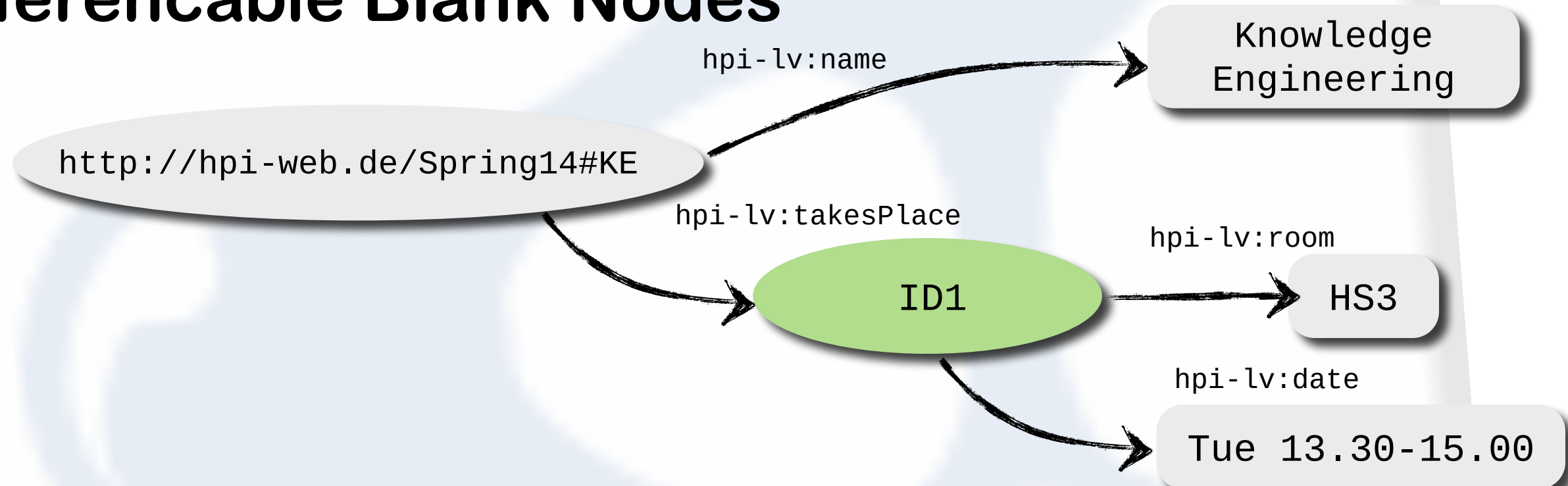
_:ID1 hpi-lv:date "Tue 13.30-15.00";
hpi-lv:room "HS3" .



Resource Description Framework

16

Dereferencable Blank Nodes



@prefix hpi-lv: <http://hpi-web.de/Lecture#>.

<http://hpi-web.de/Spring14#KE> hpi-lv:name "Knowledge Engineering";
hpi-lv:takesPlace **_:ID1** .

_:ID1 hpi-lv:date "Tue 13.30-15.00";
hpi-lv:room "HS3" .

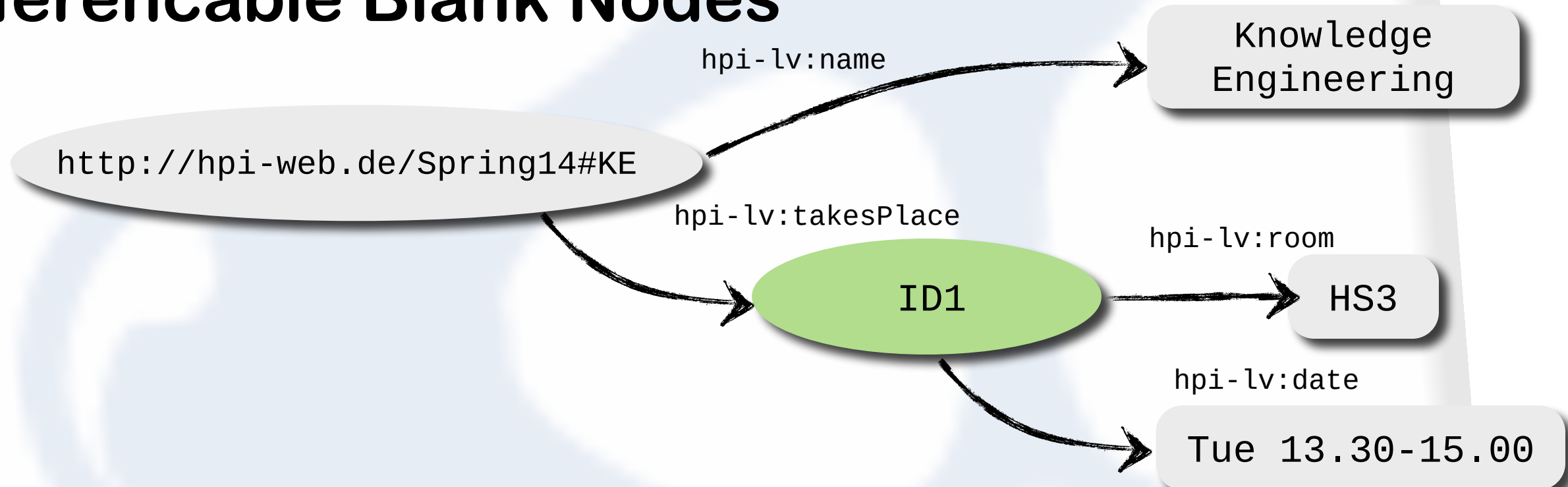
dereferencable blank node



Resource Description Framework

16

Dereferencable Blank Nodes



@prefix hpi-lv: <http://hpi-web.de/Lecture#>.

<http://hpi-web.de/Spring14#KE> hpi-lv:name "Knowledge Engineering";
hpi-lv:takesPlace **_:ID1** .

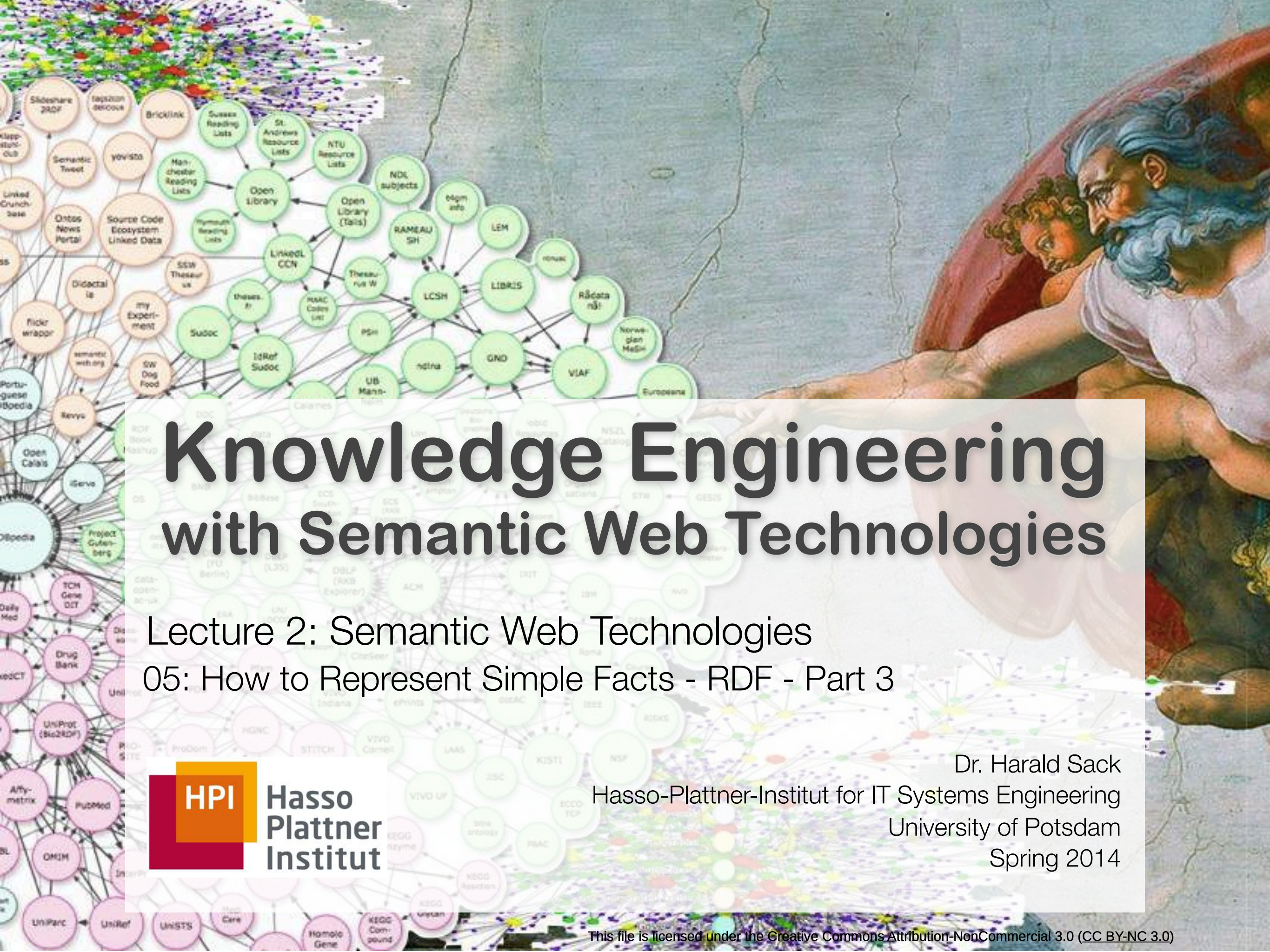
_:ID1 hpi-lv:date "Tue 13.30-15.00";
hpi-lv:room "HS3" .

dereferencable blank node

Lists in RDF

05 How to Represent Facts? - RDF (3)

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I



Knowledge Engineering with Semantic Web Technologies

Lecture 2: Semantic Web Technologies

05: How to Represent Simple Facts - RDF - Part 3



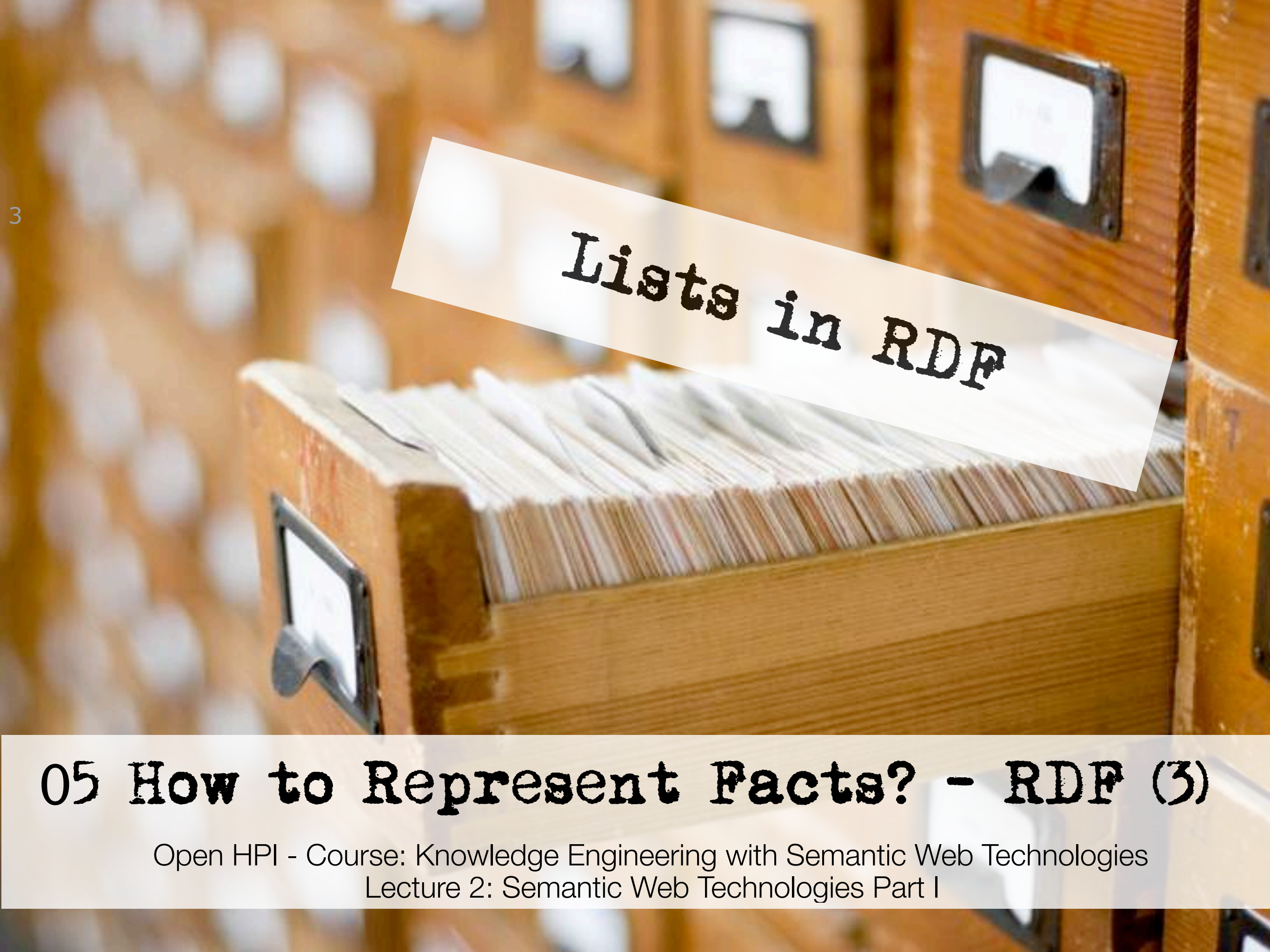
Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014



Lecture 2: Semantic Web Technologies - Part 1

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies

3



Lists in RDF

05 How to Represent Facts? - RDF (3)

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I



Resource Description Framework



*More Syntactic
Sugar....*



Resource Description Framework

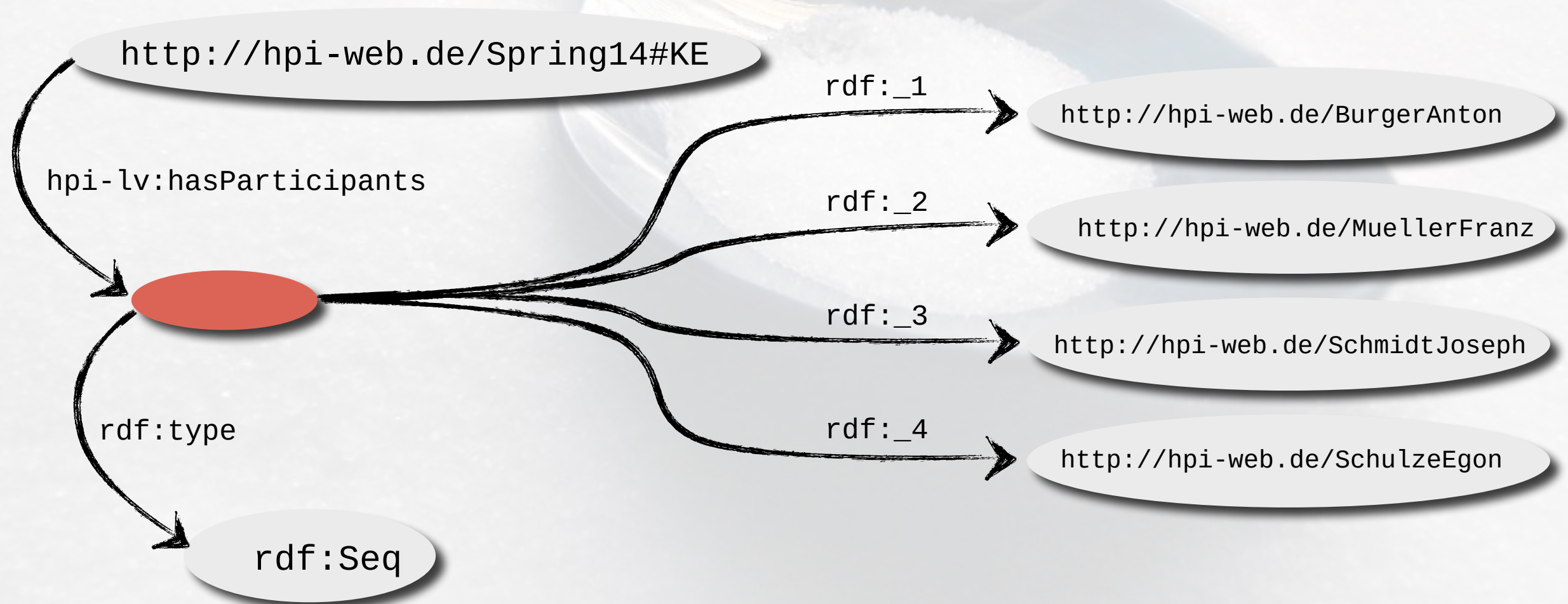
Lists

- General Data structure to enumerate any resources or literals
- Only shortcuts, no additional semantic expressivity
- Distinguish between
 - **Container**
open list, i.e. extension (new entries) possible
 - **Collections**
closed list, i.e. no extension possible



Resource Description Framework

RDF Container



@prefix hpi-lv: <http://hpi-web.de/Lecture#> .

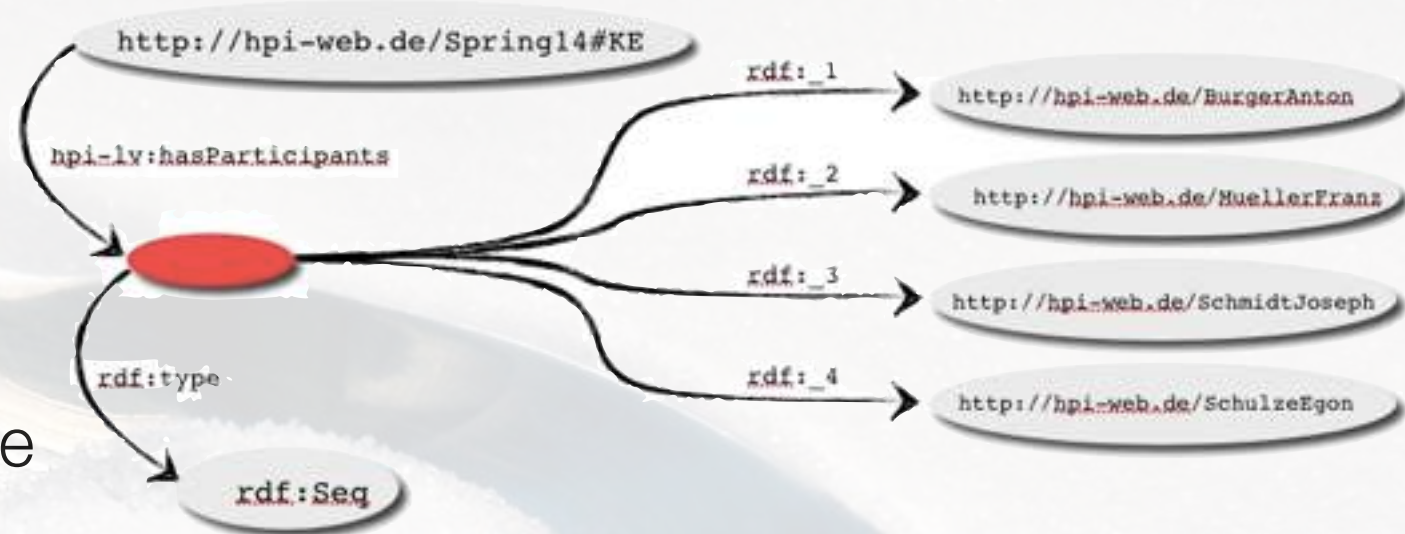
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#> .



Resource Description Framework

RDF-Container

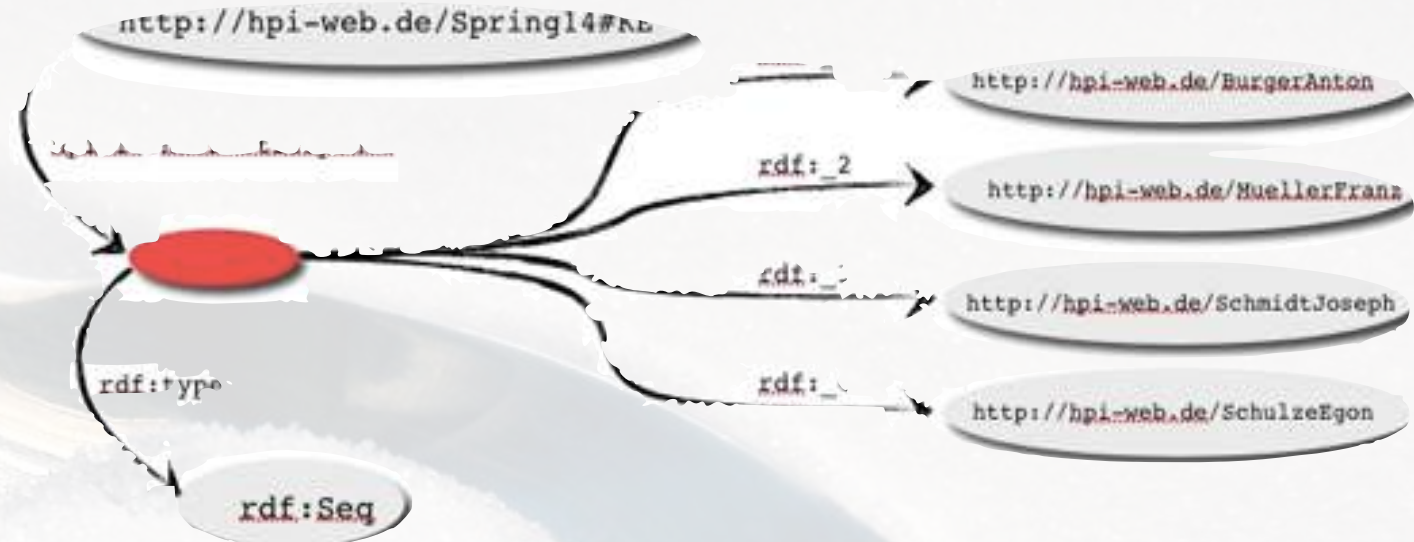
- the root node of the container is assigned a container-type via `rdf:type`
- **`rdf:Bag`**
unordered set of elements,
there is no given order of elements
- **`rdf:Seq`**
ordered set of elements,
- **`rdf:Alt`**
defines alternatives of elements
only one element of the given alternatives is relevant for the application





Resource Description Framework

RDF Container



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix lv: <http://hpi-web.de/Lecture#> .  
@base <http://hpi-web.de/>.
```

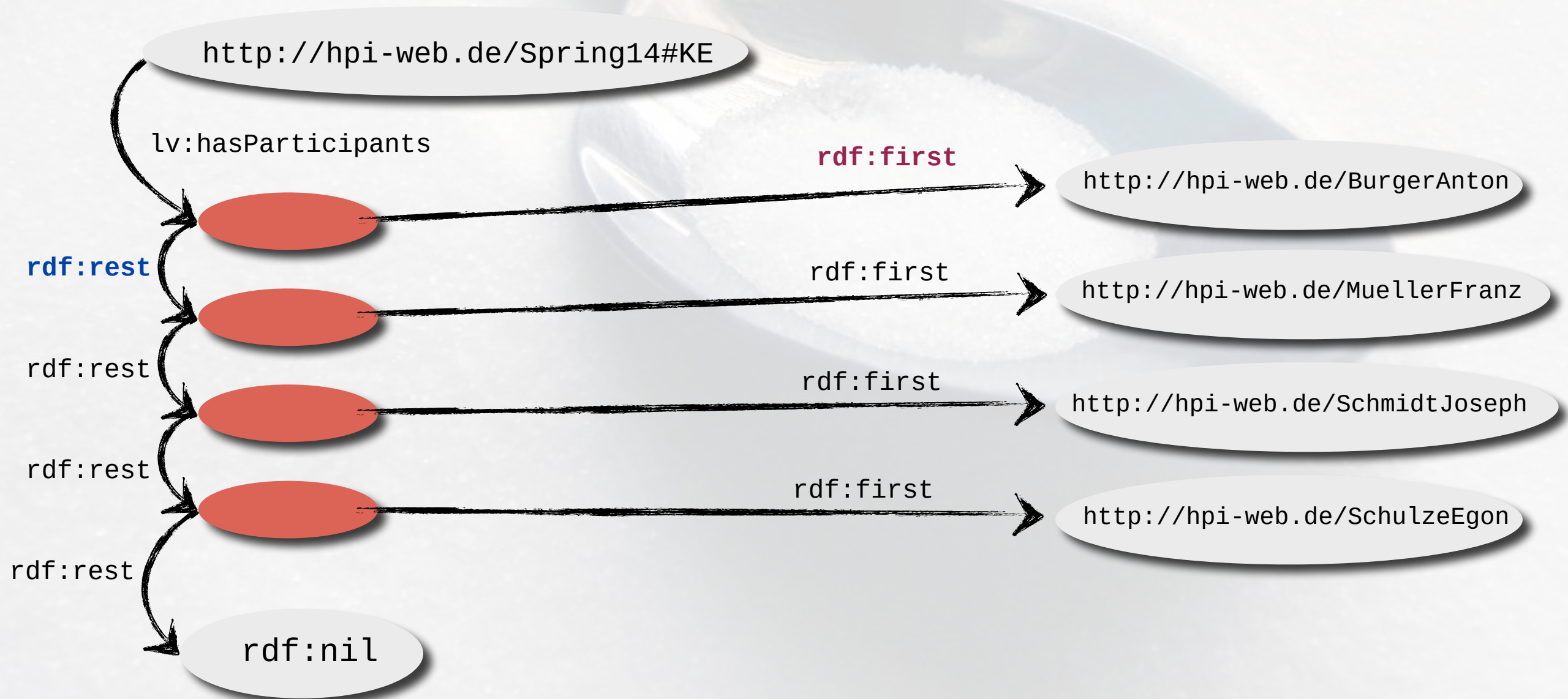
```
<Spring14#KE> lv:hasParticipants [  
  a rdf:Seq;  
  rdf:_1 <BurgerAnton>;  
  rdf:_2 <MuellerFranz>;  
  rdf:_3 <SchmidtJoseph>;  
  rdf:_4 <SchulzeEgon>  
].
```




Resource Description Framework

RDF-Collection

@prefix hpi-lv: <http://hpi-web.de/Lecture#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .



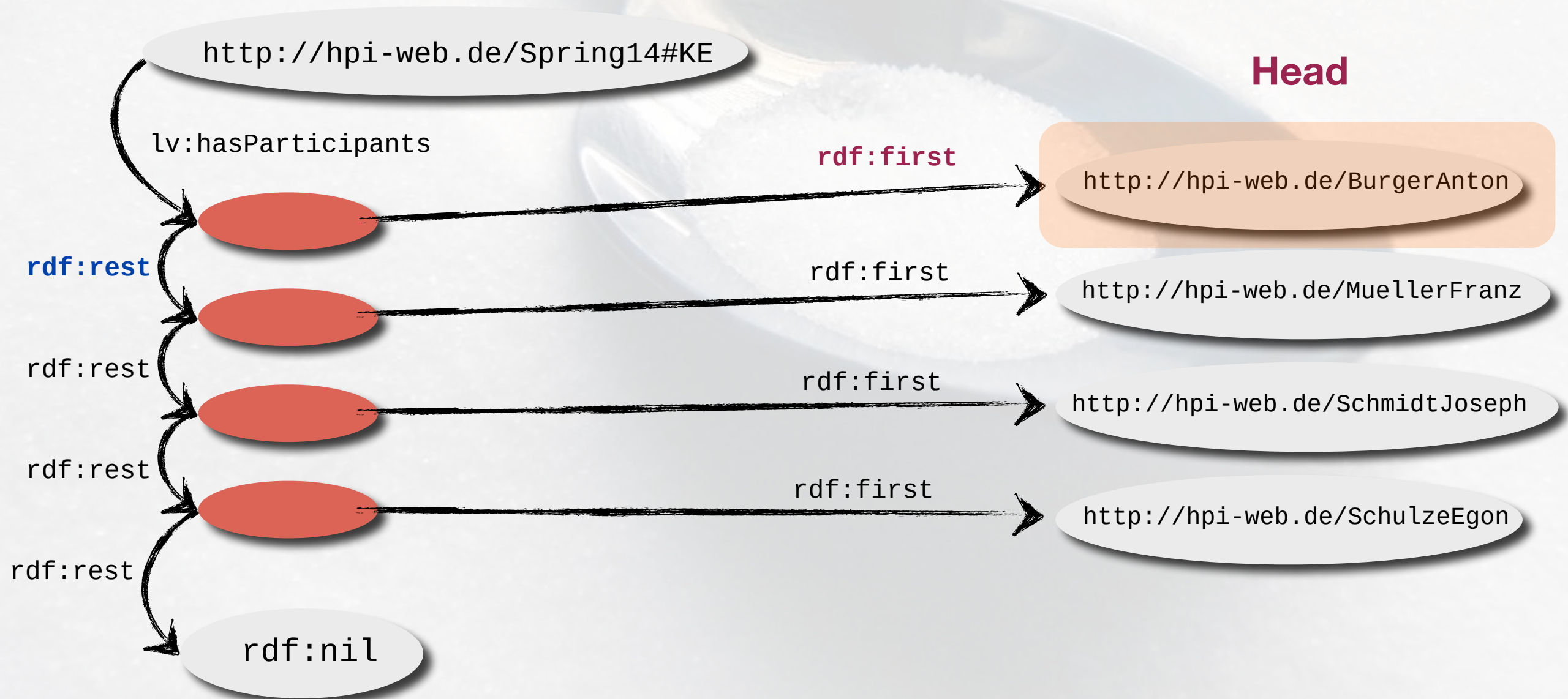
List is splitted recursively in **Head** (first) and **Tail** (rest).



Resource Description Framework

RDF-Collection

@prefix hpi-lv: <http://hpi-web.de/Lecture#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .



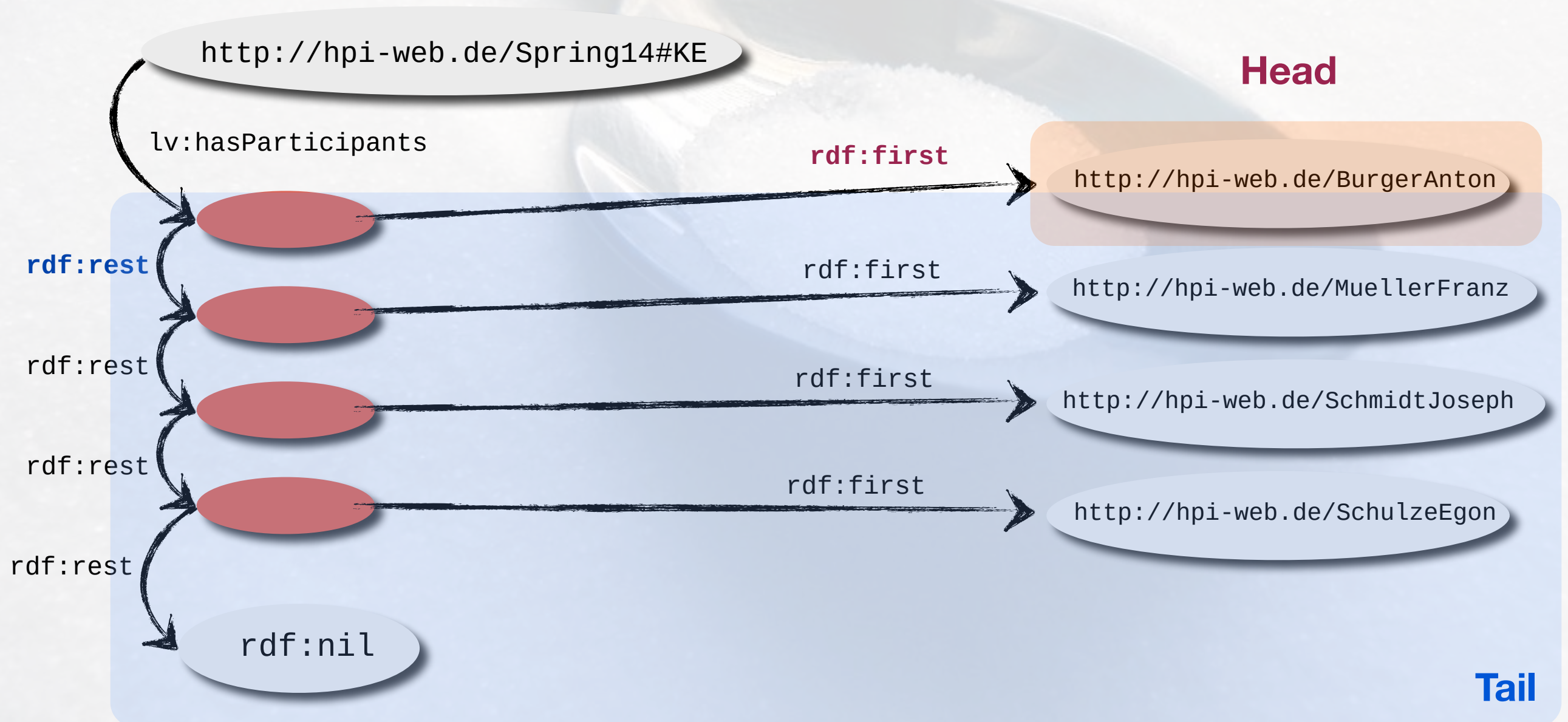
List is splitted recursively in **Head** (first) and **Tail** (rest).



Resource Description Framework

RDF-Collection

@prefix hpi-lv: <http://hpi-web.de/Lecture#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .



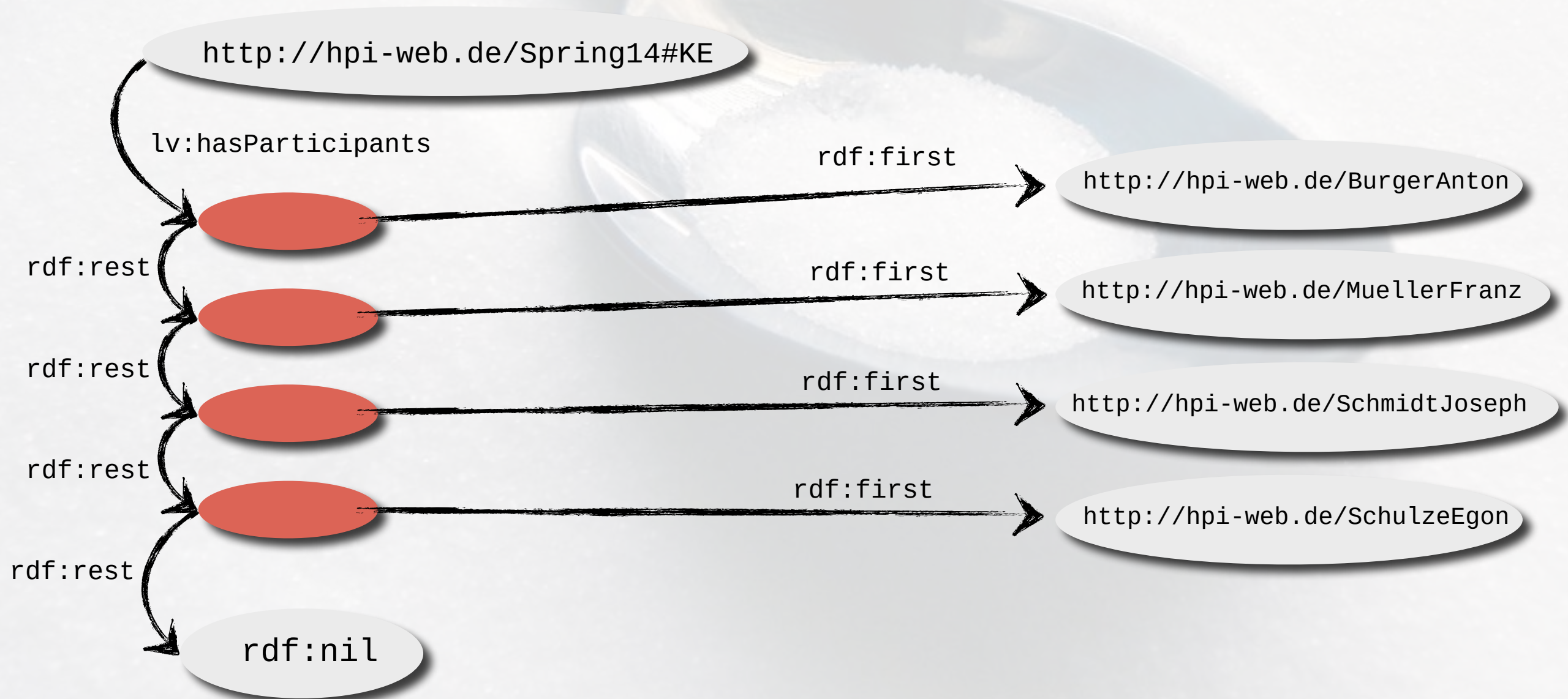
List is splitted recursively in **Head** (first) and **Tail** (rest).



Resource Description Framework

RDF-Collection

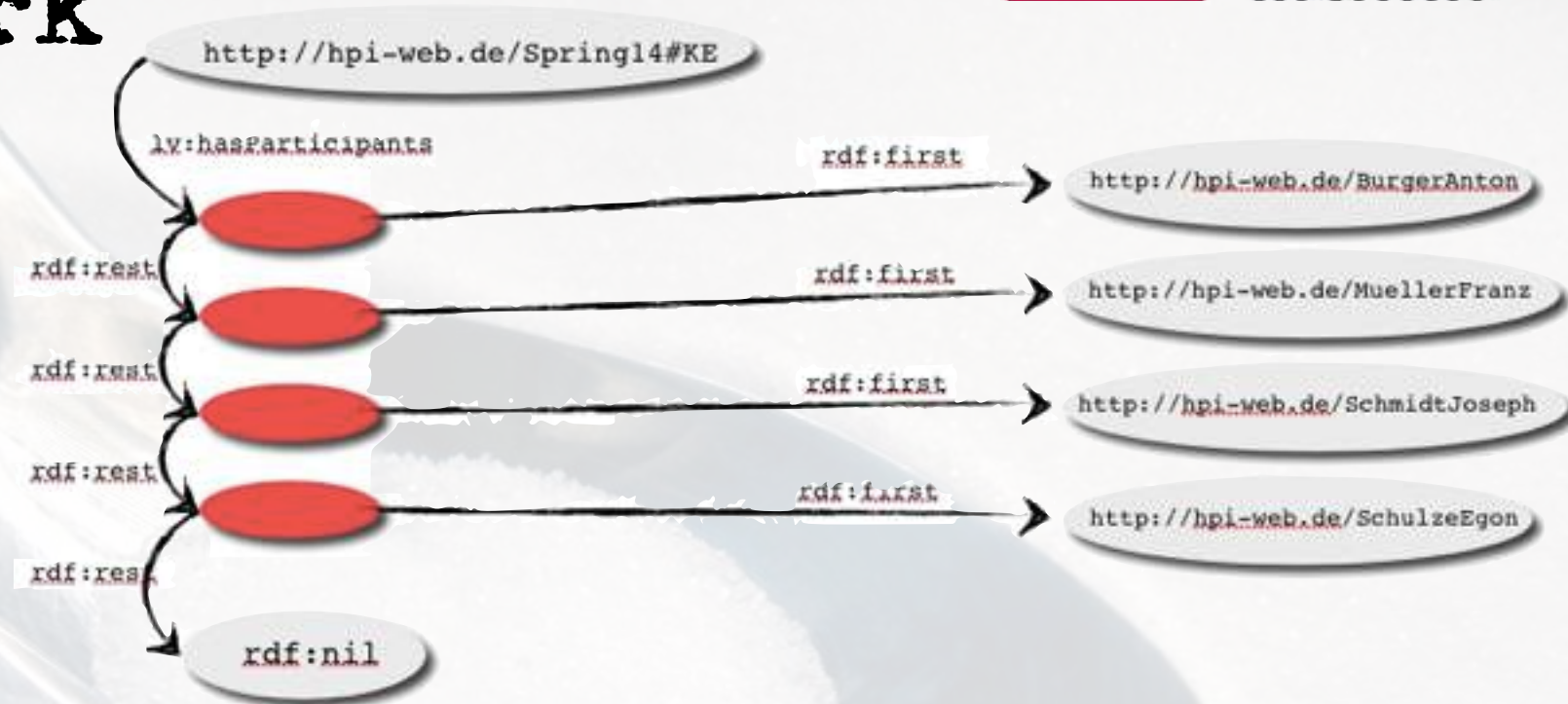
@prefix hpi-lv: <http://hpi-web.de/Lecture#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .



List is splitted recursively in **Head** (first) and **Tail** (rest).

Resource Description Framework

RDF-Collection

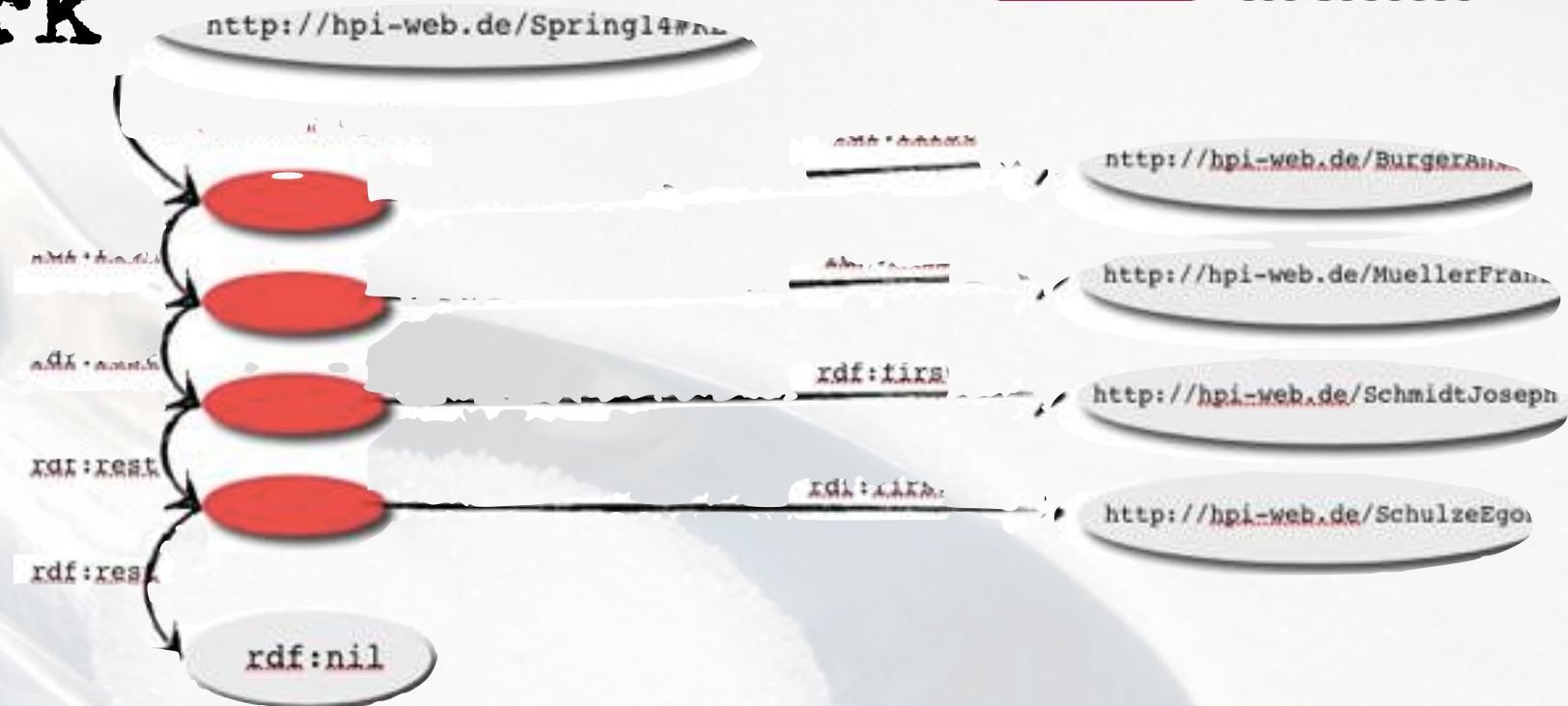


```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix lv: <http://hpi-web.de/Lecture#> .
@base <http://hpi-web.de/>.
```

```
<Spring14#KE> lv:hasParticipant [
  rdf:first <BurgerAnton>; rdf:rest [
    rdf:first <MuellerFranz>; rdf:rest [
      rdf:first <SchmidtJoseph>; rdf:rest [
        rdf:first <SchulzeEgon>;
        rdf:rest rdf:nil
      ]
    ]
  ]
].
```

Resource Description Framework

RDF-Collection



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

```
@prefix lv: <http://hpi-web.de/Lecture#> .
```

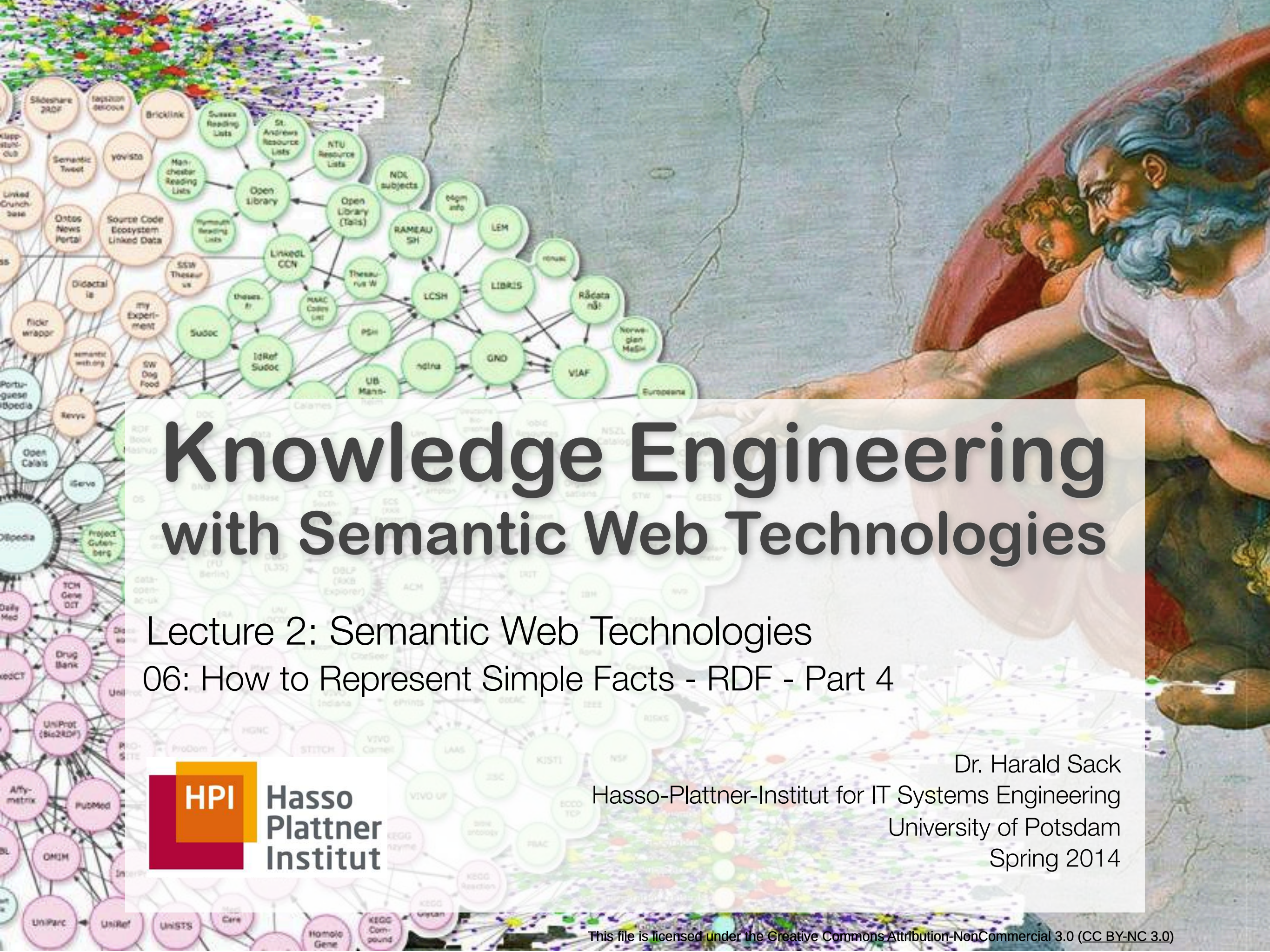
```
@base <http://hpi-web.de/>.
```

```
<Spring14#KE> lv:hasParticipant (<BurgerAnton> <MuellerFranz>  
<SchmidtJoseph> <SchulzeEgon>).
```


Reification

06 How to Represent Facts? - RDF (4)

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I



Knowledge Engineering with Semantic Web Technologies

Lecture 2: Semantic Web Technologies

06: How to Represent Simple Facts - RDF - Part 4



Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014



Lecture 2: Semantic Web Technologies - Part 1

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies

Reification & RDF Data Integration

06 How to Represent Facts? - RDF (4)

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I





„My dear Watson,
...I suppose that it was the
Gardener who has killed
the Butler!“



Resource Description Framework

RDF-Reification

- RDF permits interleaving of statements, i.e. to make statements about statements
- Example:
 - Sherlock Holmes supposes that the Gardener has killed the Butler



Resource Description Framework

RDF-Reification

- RDF permits interleaving of statements, i.e. to make statements about statements
- Example:
 - Sherlock Holmes supposes that the Gardener has killed the Butler
 - **Part 1:** The Gardener has killed the Butler
`exv:Gardener exv:hasKilled exv:Butler .`



Resource Description Framework

RDF-Reification

- RDF permits interleaving of statements, i.e. to make statements about statements
- Example:
 - Sherlock Holmes supposes that the Gardener has killed the Butler
 - **Part 1:** The Gardener has killed the Butler
`exv:Gardener exv:hasKilled exv:Butler .`
 - **Part 2:** Sherlock Holmes supposes....
`exv:SherlockHolmes exv:supposes ????`



Resource Description Framework

RDF-Reification

- **rdf:Statement**
defines an RDF Statement, consisting of Subject, Predicate and Object
- **rdf:subject**
 - the described resource
- **rdf:predicate**
 - the original property
- **rdf:object**
 - the value of the property

The Gardener has killed the Butler



Resource Description Framework

RDF-Reification

- **rdf:Statement**

defines an RDF Statement, consisting of Subject, Predicate and Object

- **rdf:subject** - the described resource
- **rdf:predicate** - the original property
- **rdf:object** - the value of the property

The Gardener has killed the Butler



Resource Description Framework

RDF-Reification

- **rdf:Statement**

defines an RDF Statement, consisting of Subject, Predicate and Object

- **rdf:subject** - the described resource
- **rdf:predicate** - the original property
- **rdf:object** - the value of the property

The Gardener has killed the Butler

rdf:Statement

rdf:subject

rdf:predicate

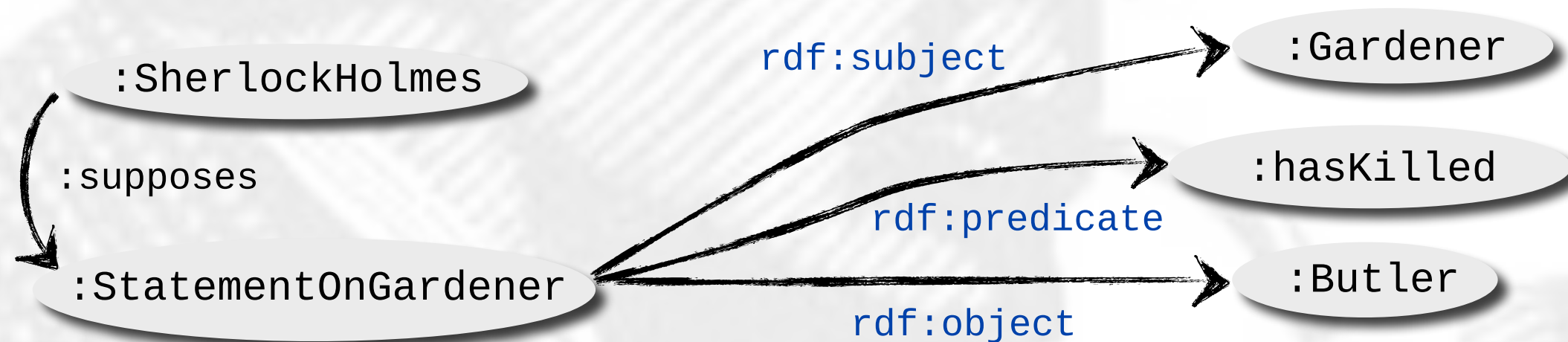
rdf:object



Resource Description Framework

RDF-Reification

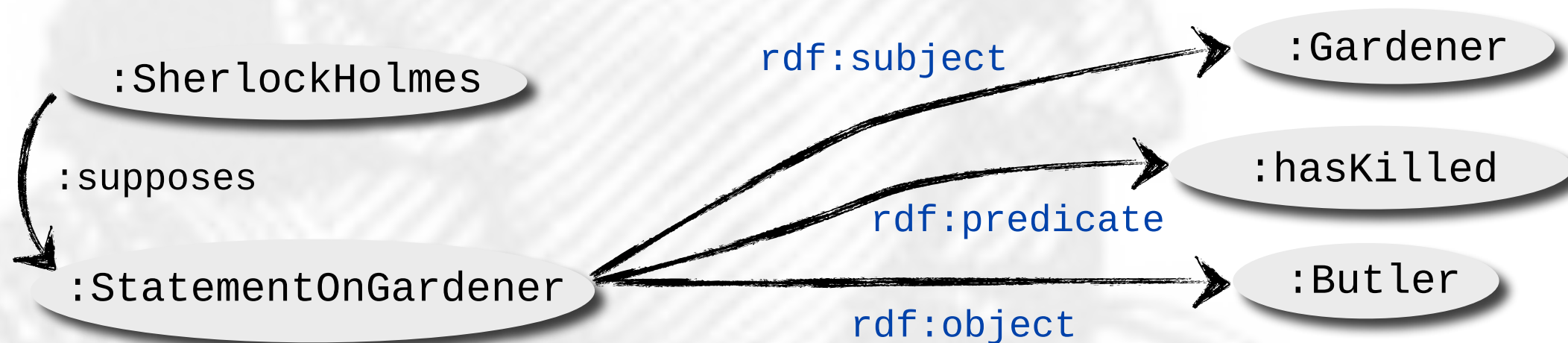
Sherlock Holmes supposes that the Gardener has killed the Butler



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix : <http://example.org/Crimestories#> .
```




Resource Description Framework



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix : <http://example.org/Crimestories#> .
```

```
:SherlockHolmes :supposes :StatementOnGardener .  
:StatementOnGardener a rdf:Statement ;  
    rdf:subject :Gardener ;  
    rdf:predicate :hasKilled ;  
    rdf:object :Butler .
```



Resource Description Framework

RDF-Reification

- Is Reification really necessary?
 - Modeling **data provenance**
 - Formalizing statements about **Reliability** (Trust)
 - Definition of **Metadata about Statements** (Assertions, Statements)
- for (general) knowledge representation (Ontologie Construction):
 - Transformation of Relations into Classes
- **But be careful:** Reification causes type conflicts
(classes, individuals, statements)



Resource Description Framework

Example

RDF Vocabulary

RDF Container `rdf:List` `rdf:nil` `rdf:first` `rdf:rest`

RDF Collection `rdf:Seq` `rdf:Bag` `rdf:Alt` `rdf:_1` `rdf:_2` ...

RDF Reification `rdf:Statement` `rdf:subject` `rdf:predicate` `rdf:object`
`rdf:type` `rdf:Property` `rdf:XMLLiteral`



RDF and Data Integration



RDF and Data Integration

- Simple Example: Bibliography Database

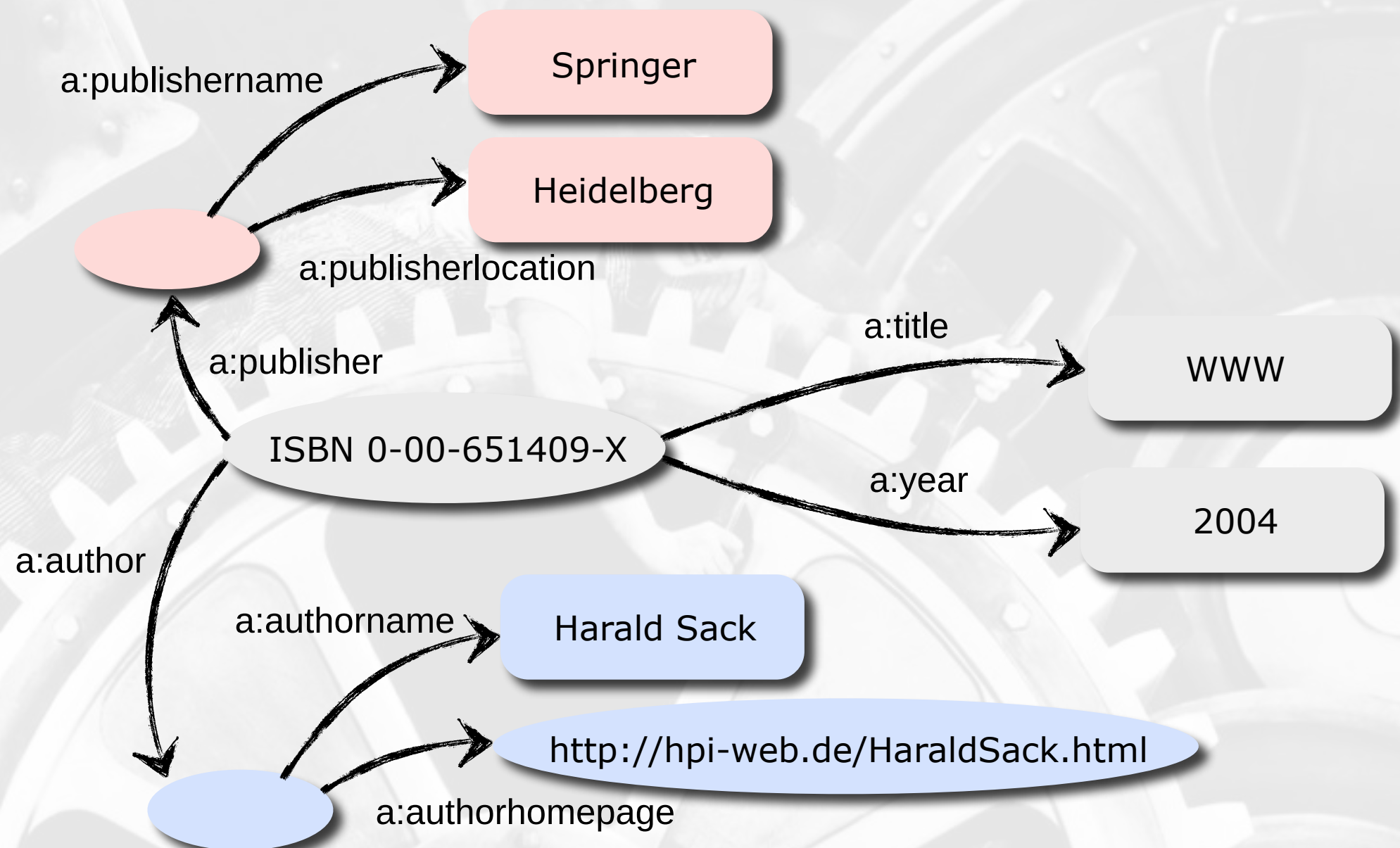
ID	Author	Title	Publisher	Year
ISBN 0-00-651409-X	HS-123	WWW	S-001	2004

ID	Name	Homepage
HS-123	Harald Sack	http://hpi-web.de/HaraldSack.html

ID	Publisher	Location
S-001	Springer	Heidelberg

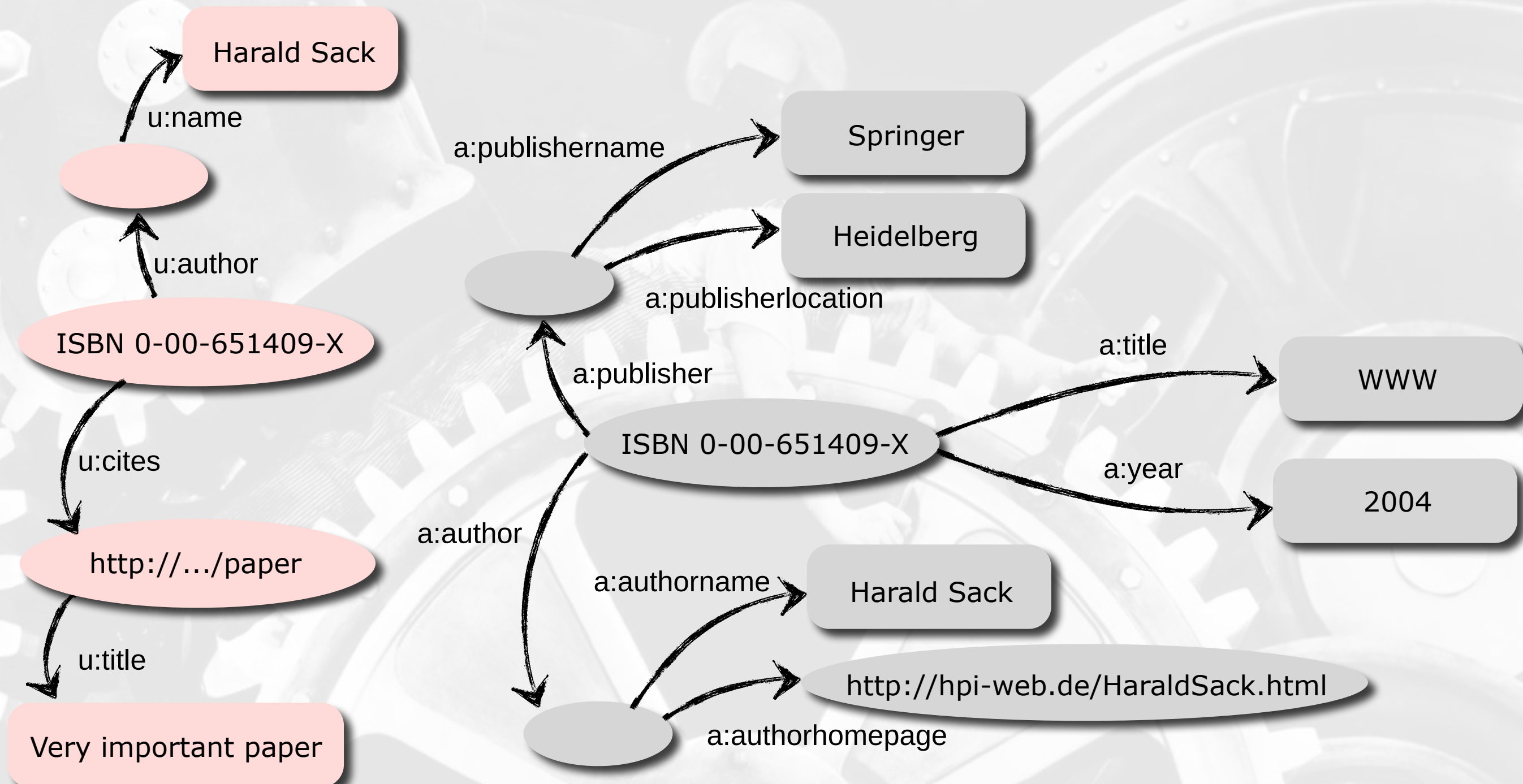
RDF and Data Integration

- Database export into a set of relations



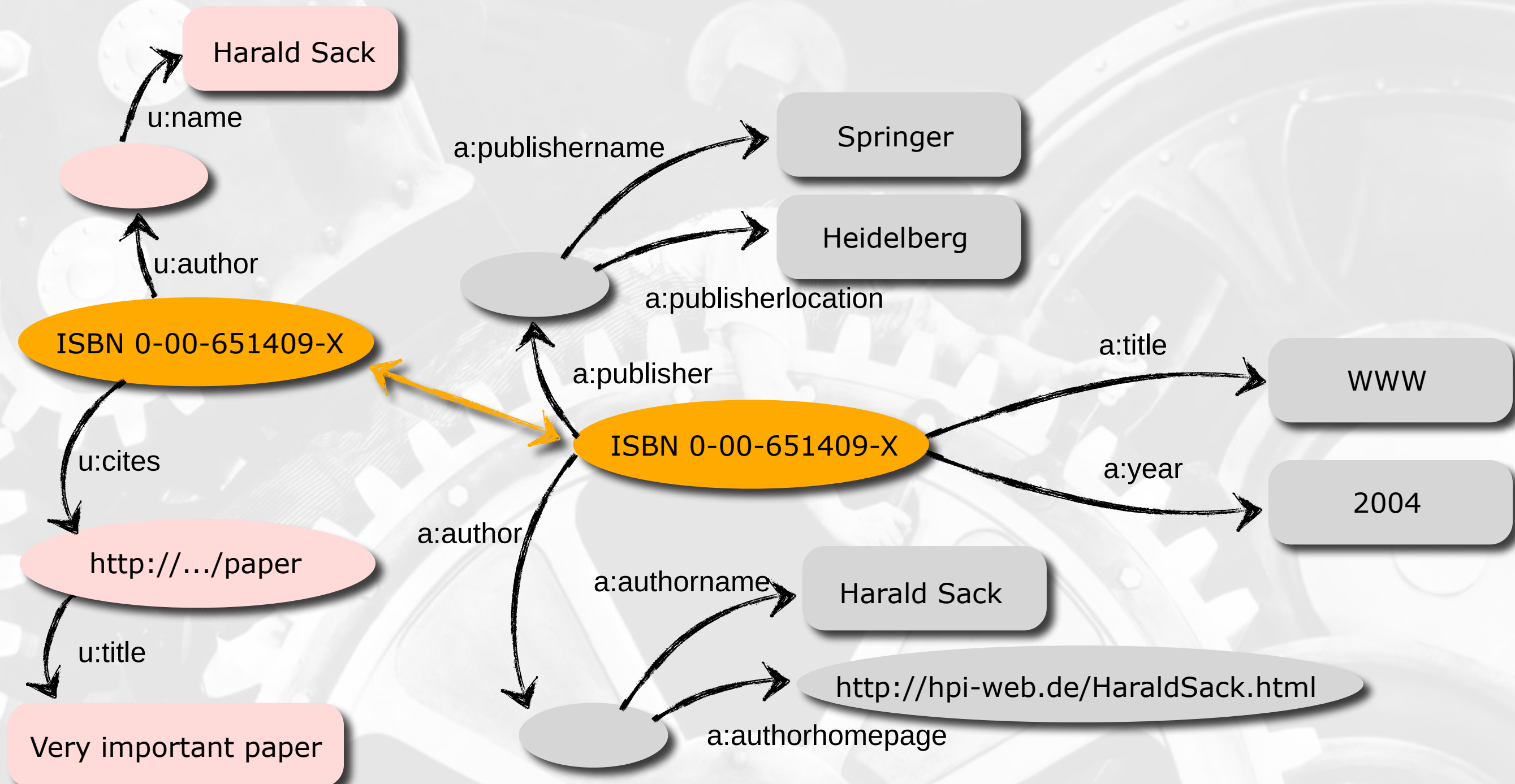
RDF and Data Integration

14



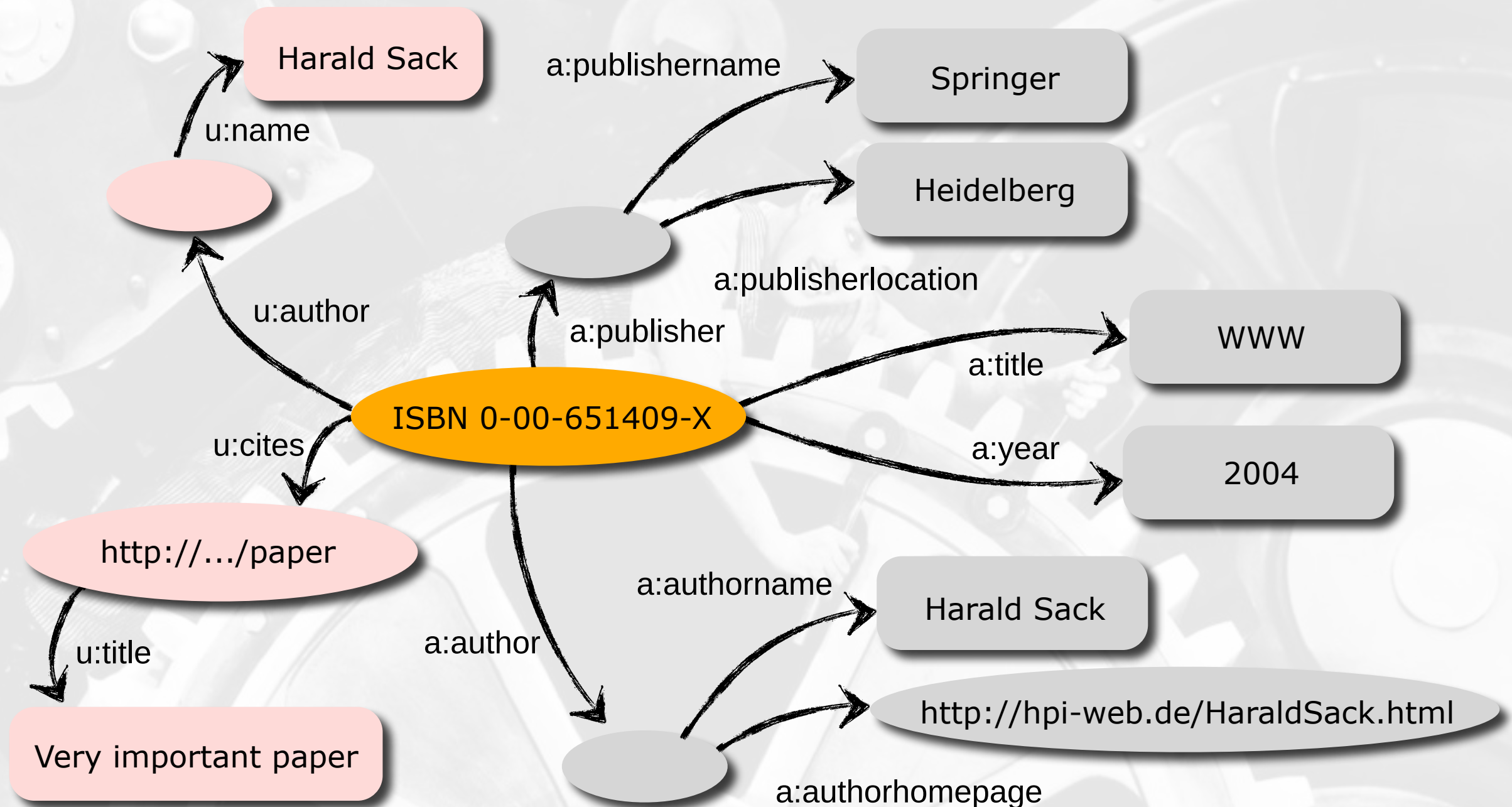
RDF and Data Integration

15



RDF and Data Integration

16

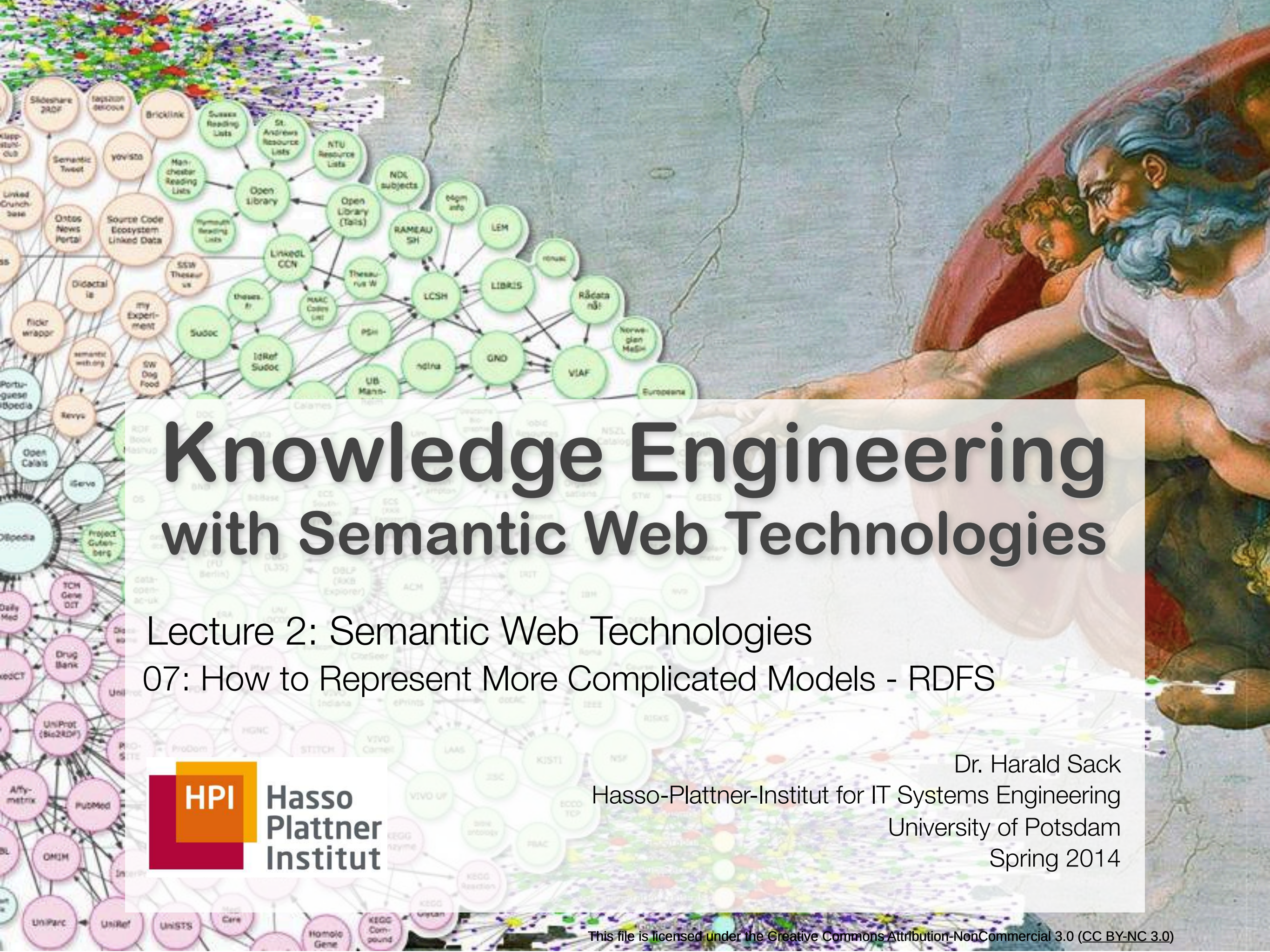


07 How to represent more complicated models? - RDFS

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I



SO ... WE NEED A WAY TO
DEFINE RDF SCHEMATA FOR
CLASSES AND RELATIONS ?!



Knowledge Engineering with Semantic Web Technologies

Lecture 2: Semantic Web Technologies

07: How to Represent More Complicated Models - RDFS



Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014



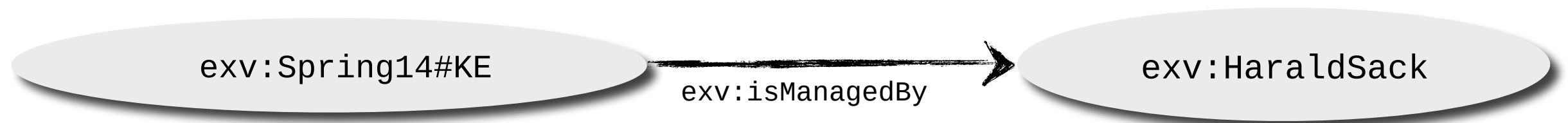
Lecture 2: Semantic Web Technologies - Part 1

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies

What about the Meaning?

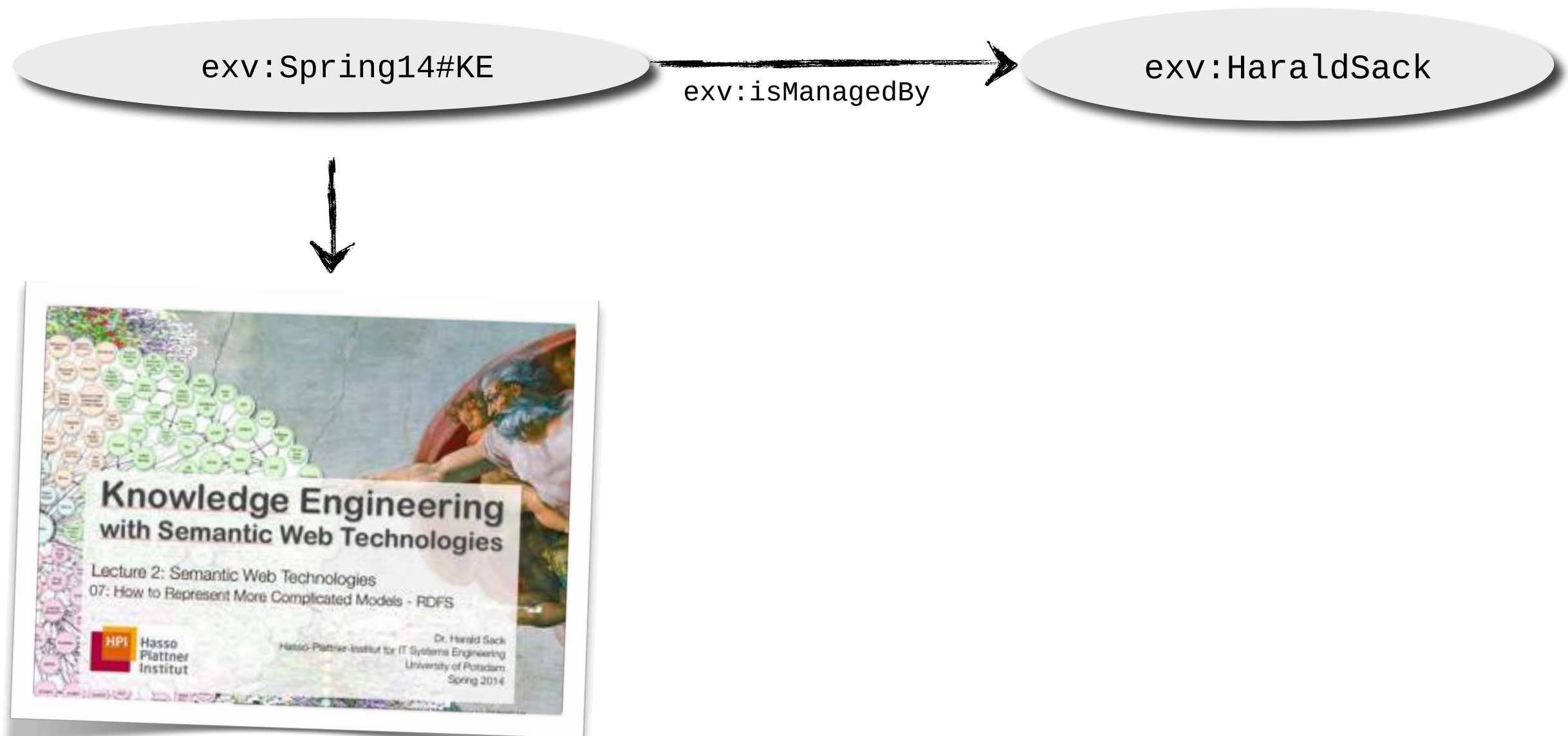
3

RDF



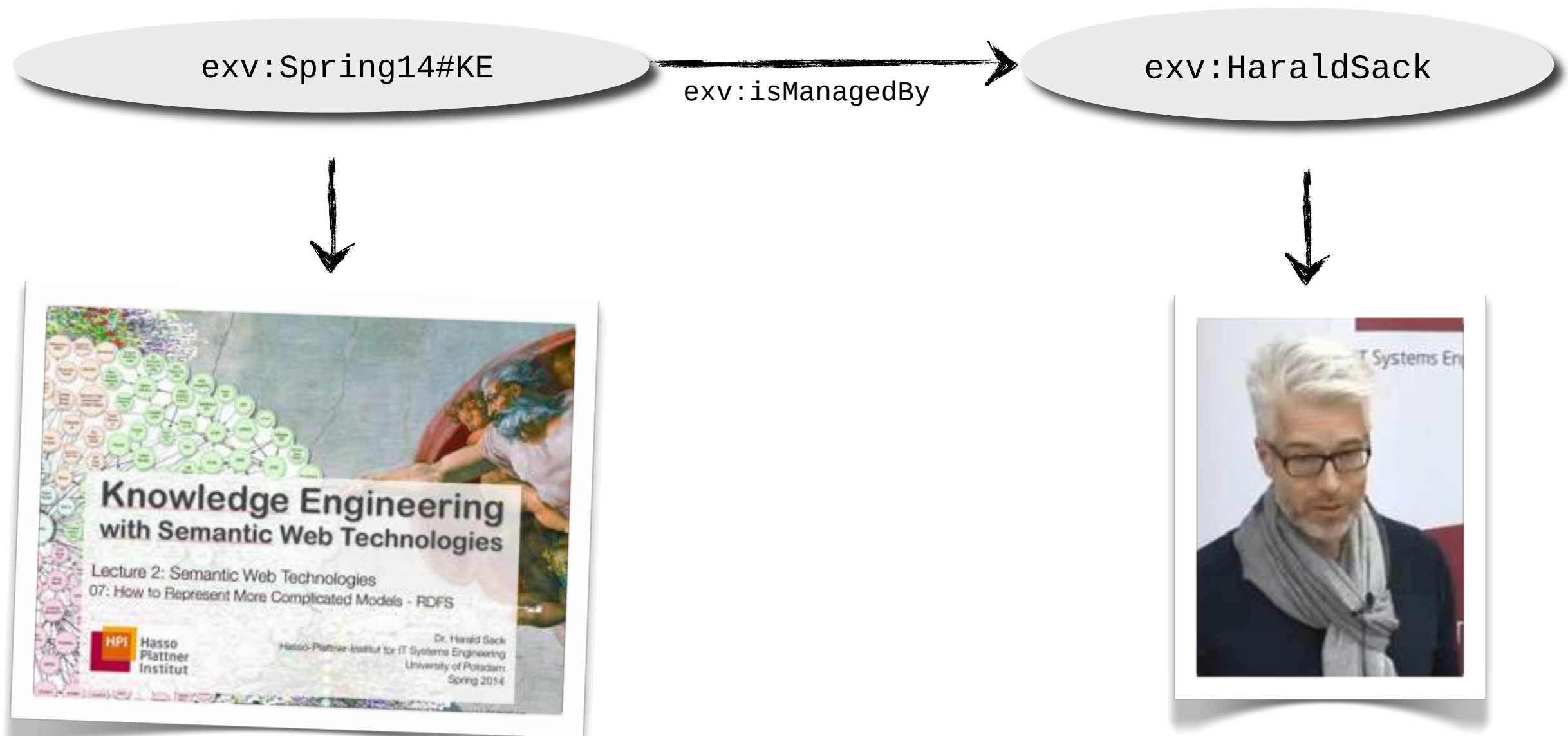
What about the Meaning?

RDF



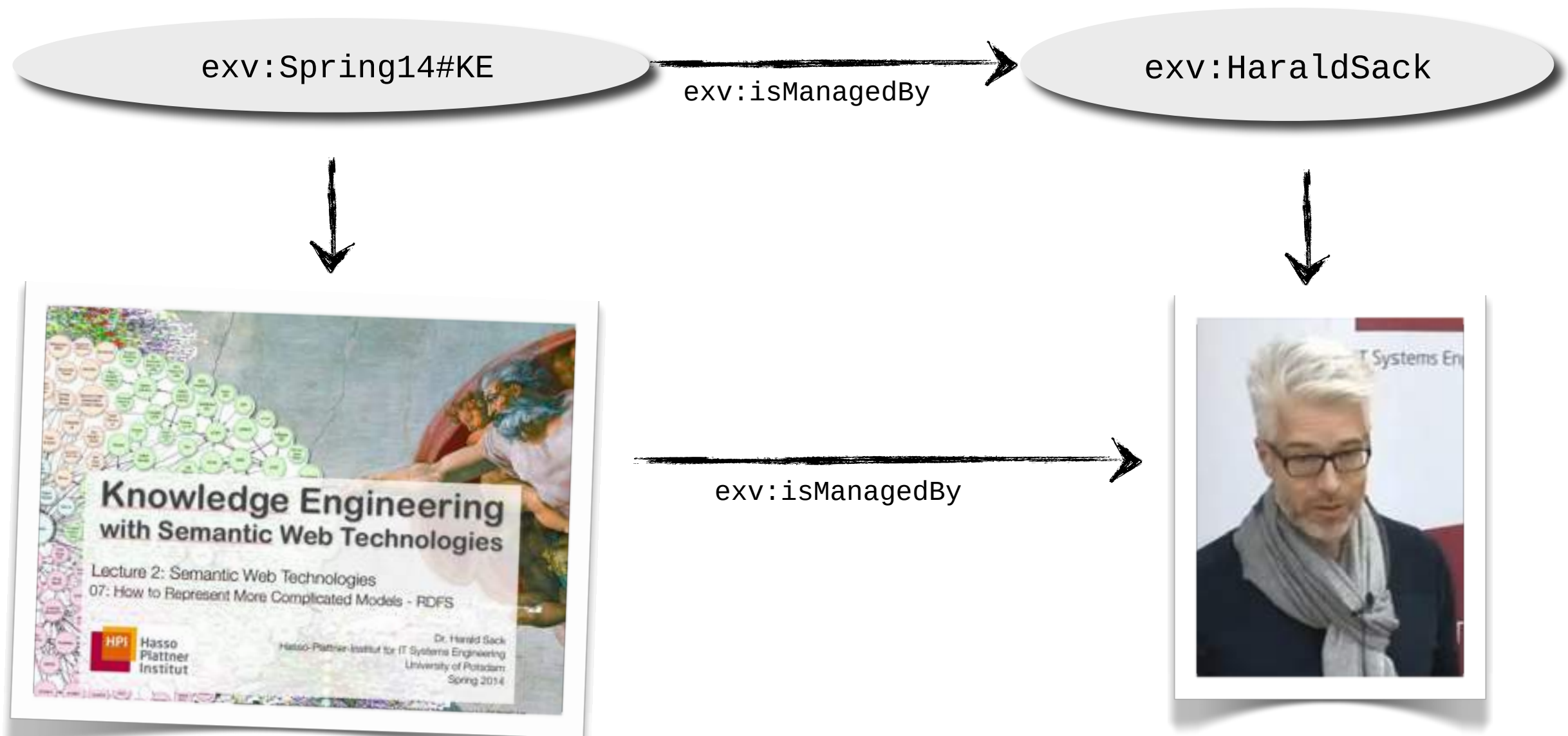
What about the Meaning?

RDF



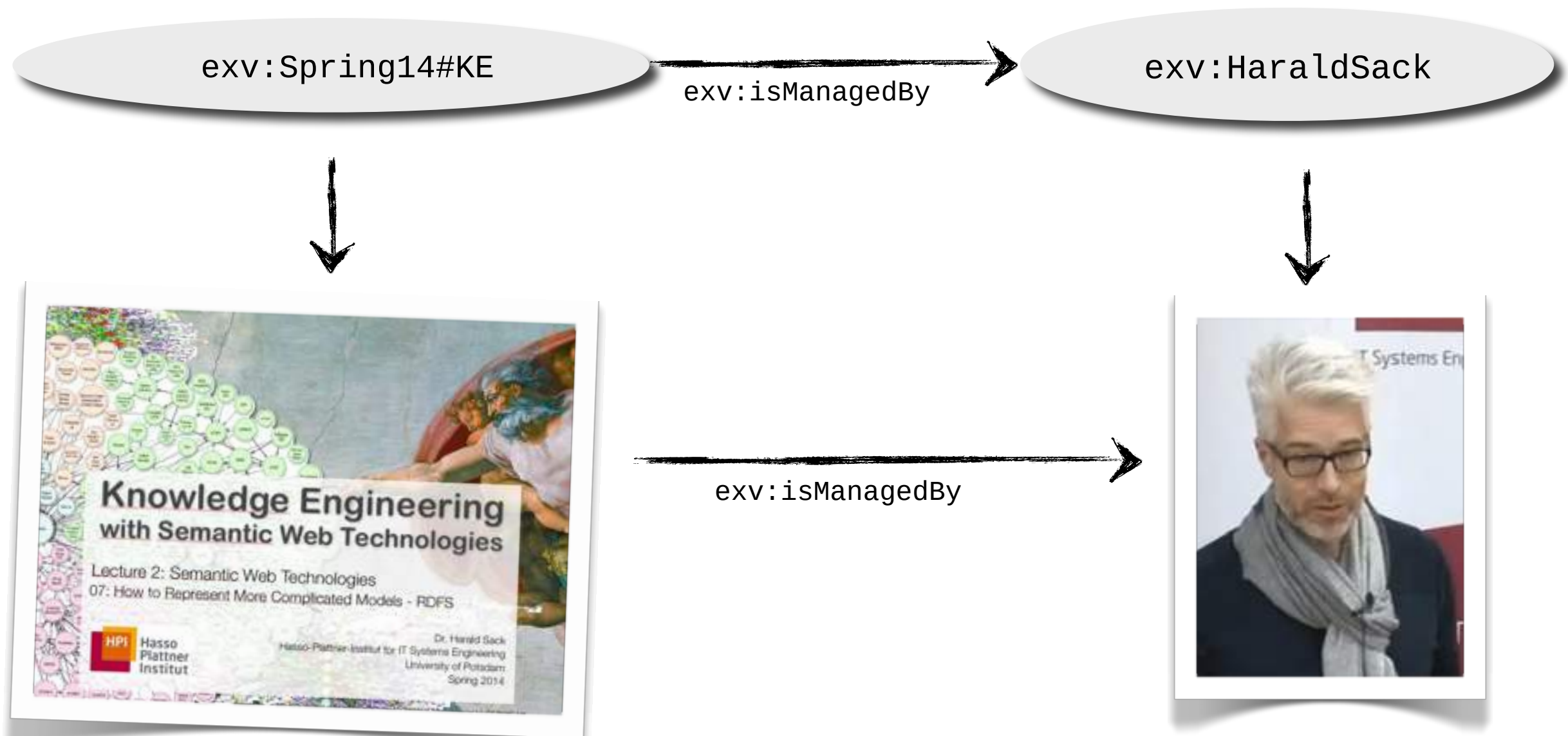
What about the Meaning?

RDF



What about the Meaning?

RDF

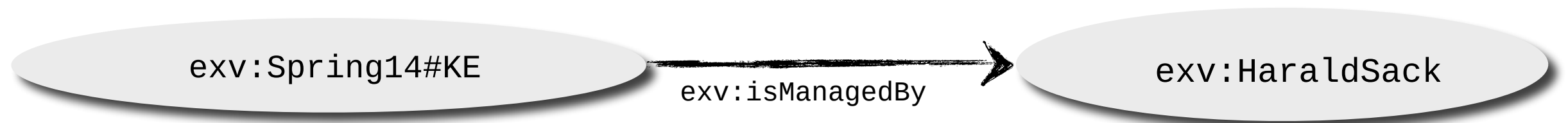


Where does the intended Meaning come from?

What about the Meaning?

4

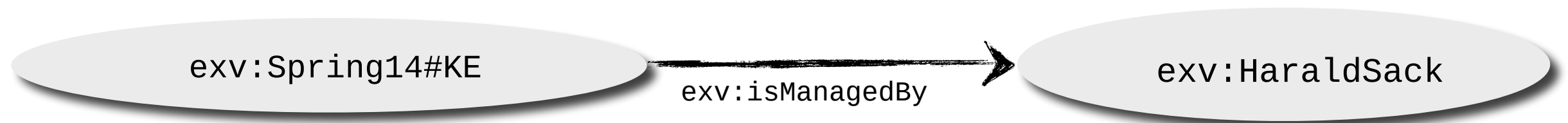
RDF



What about the Meaning?

4

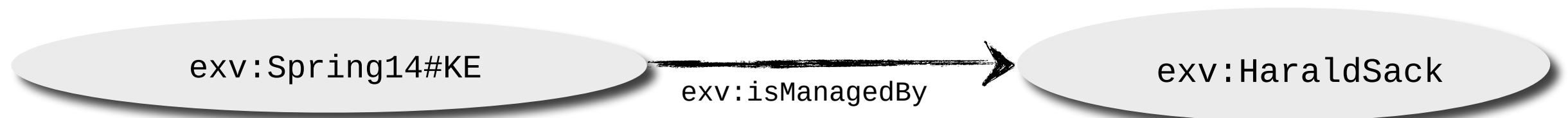
RDF



What about the Meaning?

4

RDF

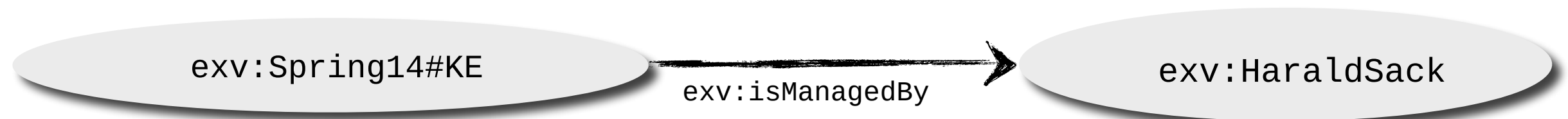


*something from the **namespace** **exv:**
with the **URI** **Spring14#KE***

What about the Meaning?

4

RDF



*something from the **namespace exv:**
with the **URI Spring14#KE***

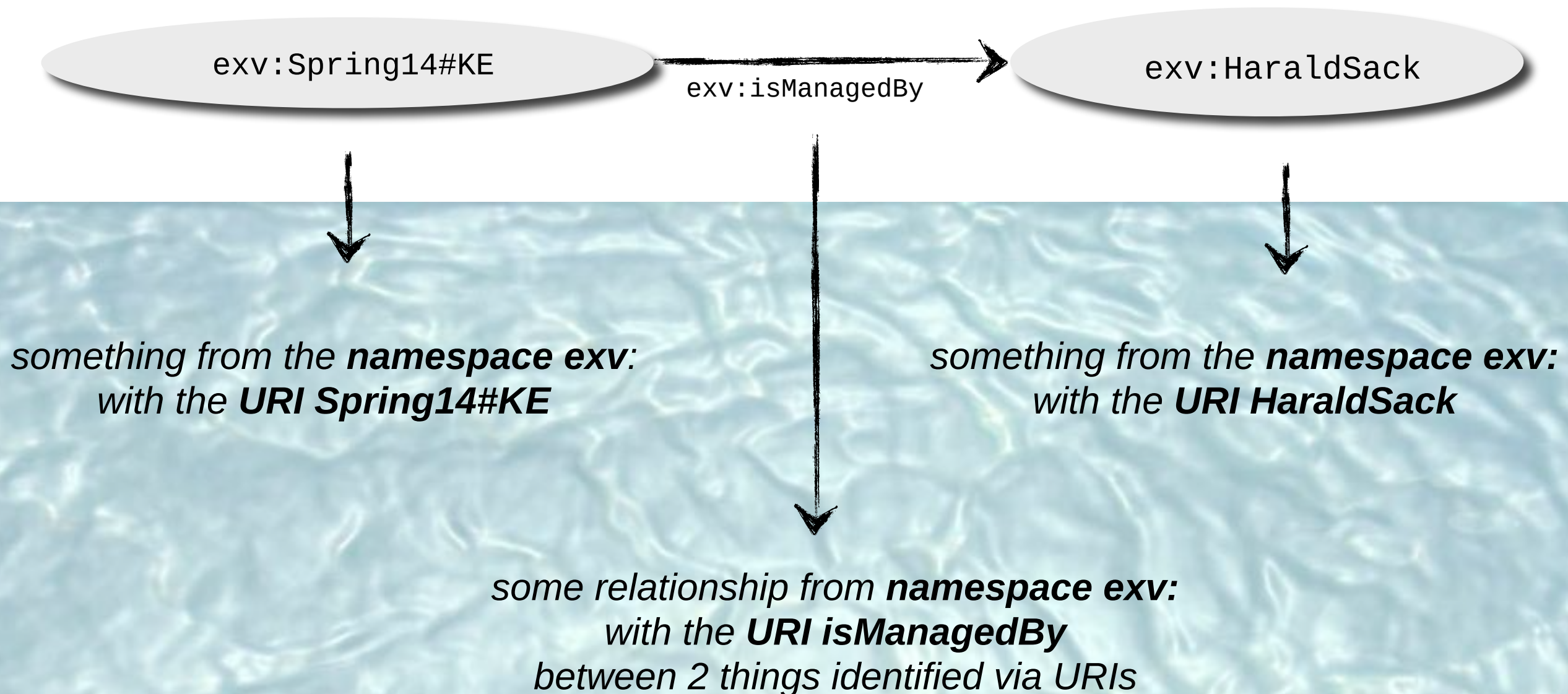


*something from the **namespace exv:**
with the **URI HaraldSack***

What about the Meaning?

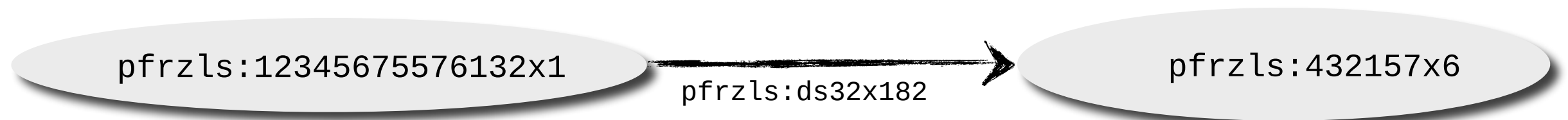
4

RDF



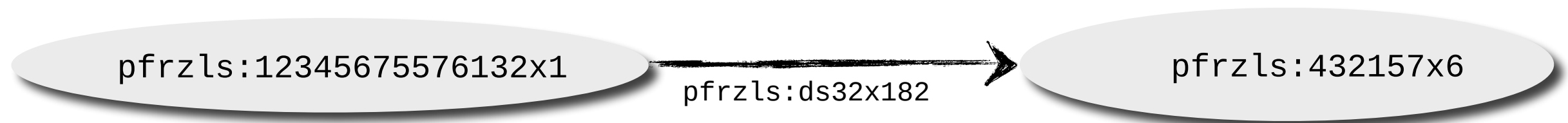
What about the Meaning?

RDF



What about the Meaning?

RDF



Probably we need a little
bit more to make something
out of it....

07 How to represent more complicated models? - RDFS

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I



SO ... WE NEED A WAY TO
DEFINE RDF SCHEMATA FOR
CLASSES AND RELATIONS ?!

2. Semantic Web Technologies

Part 1

The Semantic Web Technology Stack (not a piece of cake...)

Most apps use only a subset of the stack

Querying allows fine-grained data access

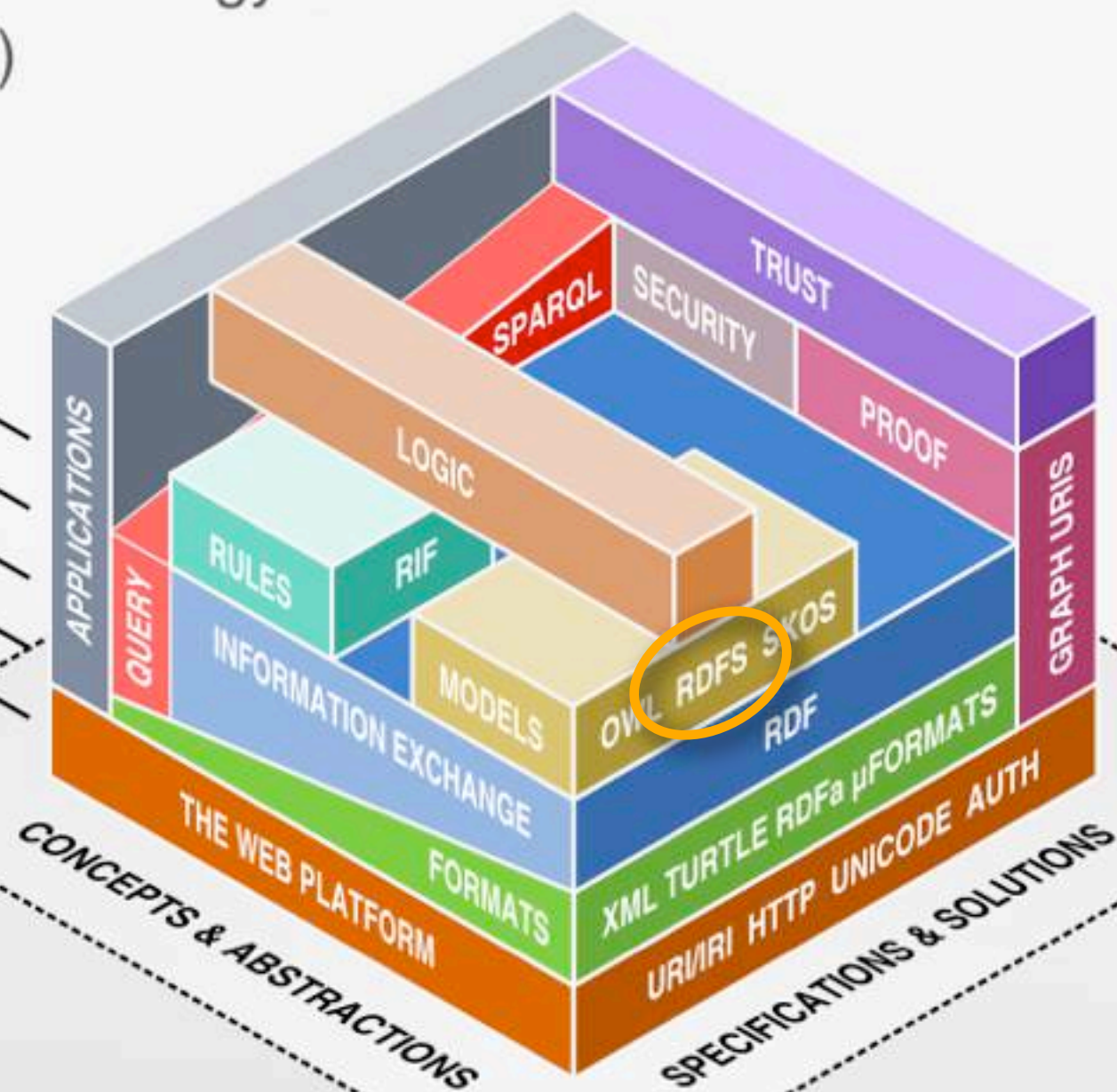
Standardized information exchange is key

Formats are necessary, but not too important

The Semantic Web is based on the Web

Linked Data uses a small
selection of technologies

LINKED DATA



Why RDFSchema...?

Why RDFSchema...?

- An **application program** can define and use **RDF data**

Why RDFSchema...?

- An **application program** can define and use **RDF data**
-if the application program knows, **which terms and classes to use**, as e.g.

Why RDFSchema...?

- An **application program** can define and use **RDF data**
-if the application program knows, **which terms and classes to use**, as e.g.
 - name, title, year, ...

Why RDFSchema...?

- An **application program** can define and use **RDF data**
-if the application program knows, **which terms and classes to use**, as e.g.
 - name, title, year, ...
 - name, blog, phone number...

Why RDFSchema...?

- An **application program** can define and use **RDF data**
-if the application program knows, **which terms and classes to use**, as e.g.
 - name, title, year, ...
 - name, blog, phone number...
 - author, cites, ...

Why RDFSchema...?

- An **application program** can define and use **RDF data**
-if the application program knows, **which terms and classes to use**, as e.g.
 - name, title, year, ...
 - name, blog, phone number...
 - author, cites, ...
- But.....

Why RDFSchema...?

- An **application program** can define and use **RDF data**
-if the application program knows, **which terms and classes to use**, as e.g.
 - name, title, year, ...
 - name, blog, phone number...
 - author, cites, ...
- But.....
 - Are all terms known?

Why RDFSchema...?

- An **application program** can define and use **RDF data**
-if the application program knows, **which terms and classes to use**, as e.g.
 - name, title, year, ...
 - name, blog, phone number...
 - author, cites, ...
- But.....
 - Are all terms known?
 - Are all terms correct?

Why RDFSchema...?

- An **application program** can define and use **RDF data**
-if the application program knows, **which terms and classes to use**, as e.g.
 - name, title, year, ...
 - name, blog, phone number...
 - author, cites, ...
- But.....
 - Are all terms known?
 - Are all terms correct?
 - Are there (logical) relations among the terms?

Why RDFSchema...?

- An **application program** can define and use **RDF data**
-if the application program knows, **which terms and classes to use**, as e.g.
 - name, title, year, ...
 - name, blog, phone number...
 - author, cites, ...
- But.....
 - Are all terms known?
 - Are all terms correct?
 - Are there (logical) relations among the terms?
- We need a language for data definition: **RDF Schema**

Why RDFSchema...?

- An **application program** can define and use **RDF data**
-if the application program knows, **which terms and classes to use**, as e.g.
 - name, title, year, ...
 - name, blog, phone number...
 - author, cites, ...
- But.....
 - Are all terms known?
 - Are all terms correct?
 - Are there (logical) relations among the terms?
- We need a language for data definition: **RDF Schema**
 - officially called “**RDF Vocabulary Description Language**”

RDFSchema



- RDFSchema allows:
 - Definition of **classes**
 - **Class instantiation** in RDF via `<rdf:type>`
 - Definition of **properties** and **restrictions**
 - Definition of **hierarchies**
 - Subclasses and superclasses
 - Subproperties and superproperties

• Classes

- **`rdfs:Class`**

Concept of a class, defines an abstract object and is applied (with `rdf:type`) to create instances

- **`rdf:Property`**

Base class for properties

- **`rdfs:Literal`**

Class for literals

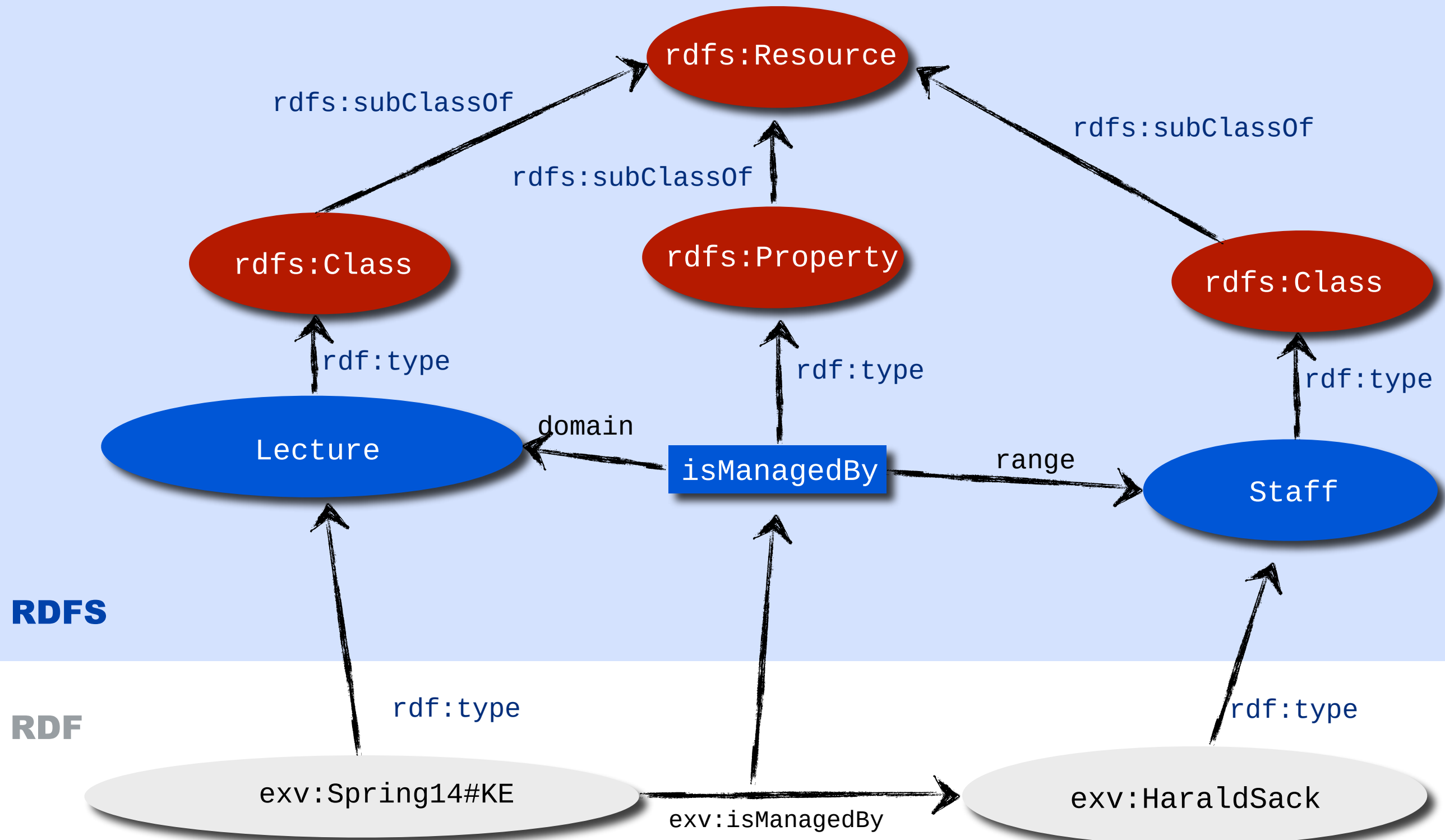
- **`rdfs:Resource`**

every entity of an RDF model is instance of this class

- and additionally

**`rdfs:Datatype`, `rdf:XMLLiteral`, `rdfs:Container`,
`rdfs:ContainerMembershipProperty`**

11



- Properties

- `rdfs:subClassOf`

- transitive property to define inheritance hierarchies for classes

- `rdfs:subPropertyOf`

- transitive property to define inheritance hierarchies for properties

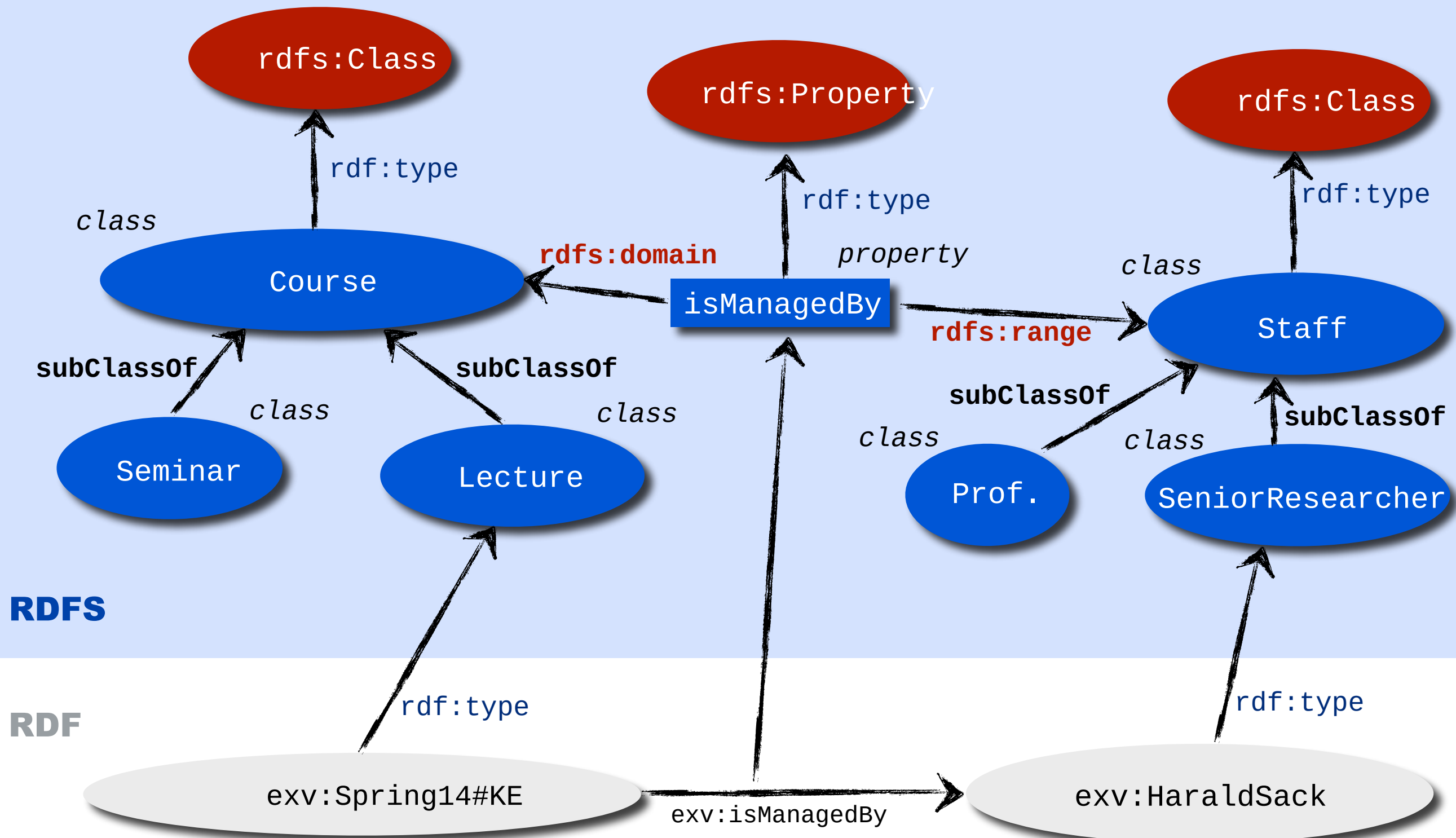
- `rdfs:domain`

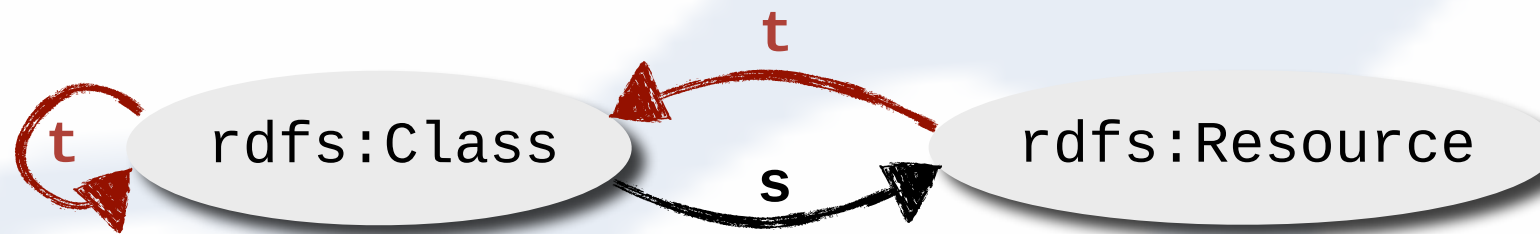
- defines the domain of a property concerning a class

- `rdfs:range`

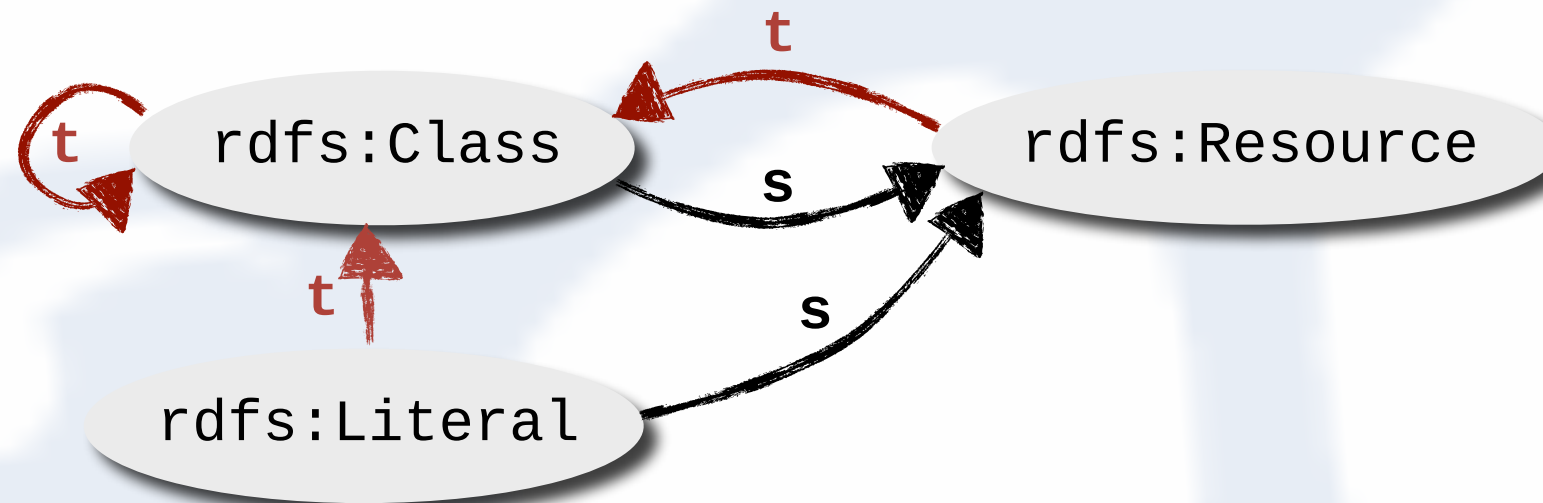
- defines range of a property concerning a class

13

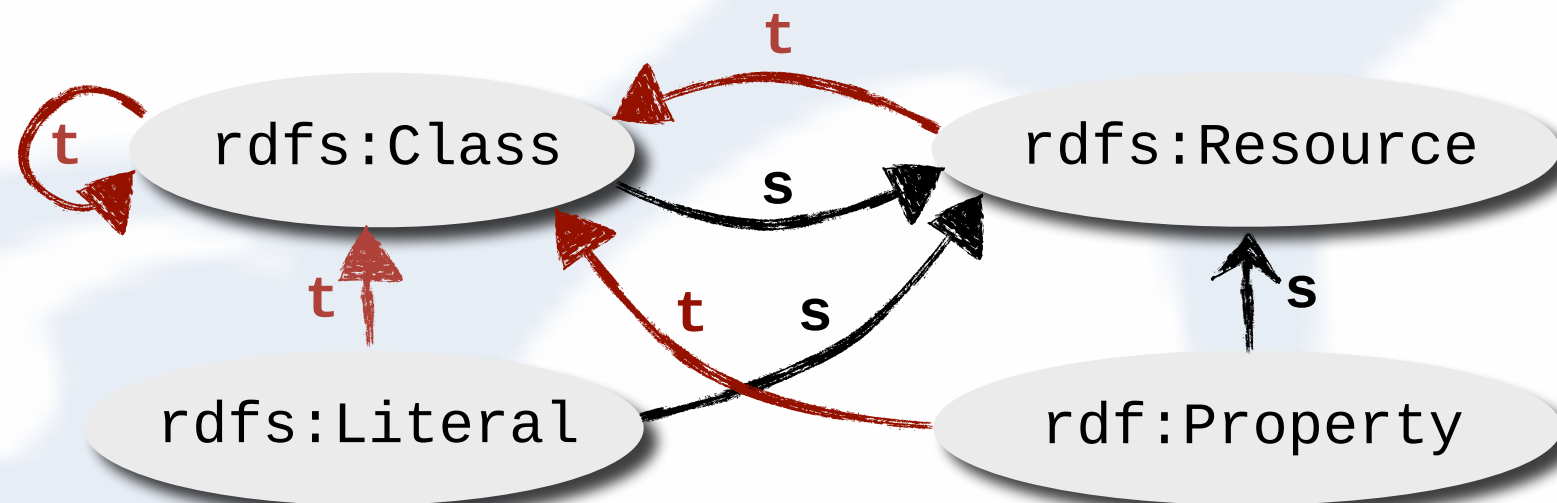




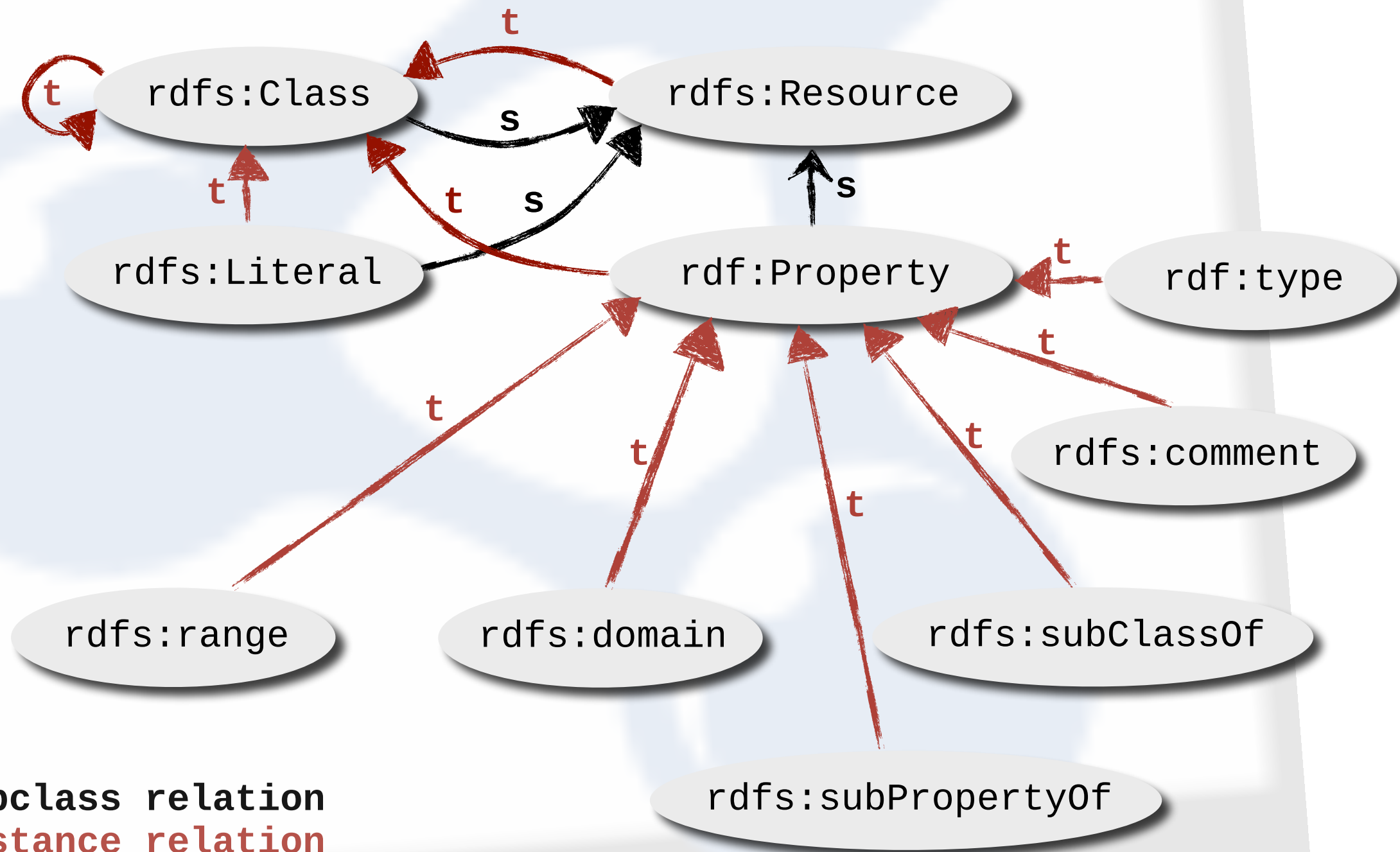
s - subclass relation
t - instance relation



s - subclass relation
t - instance relation



s - subclass relation
t - instance relation

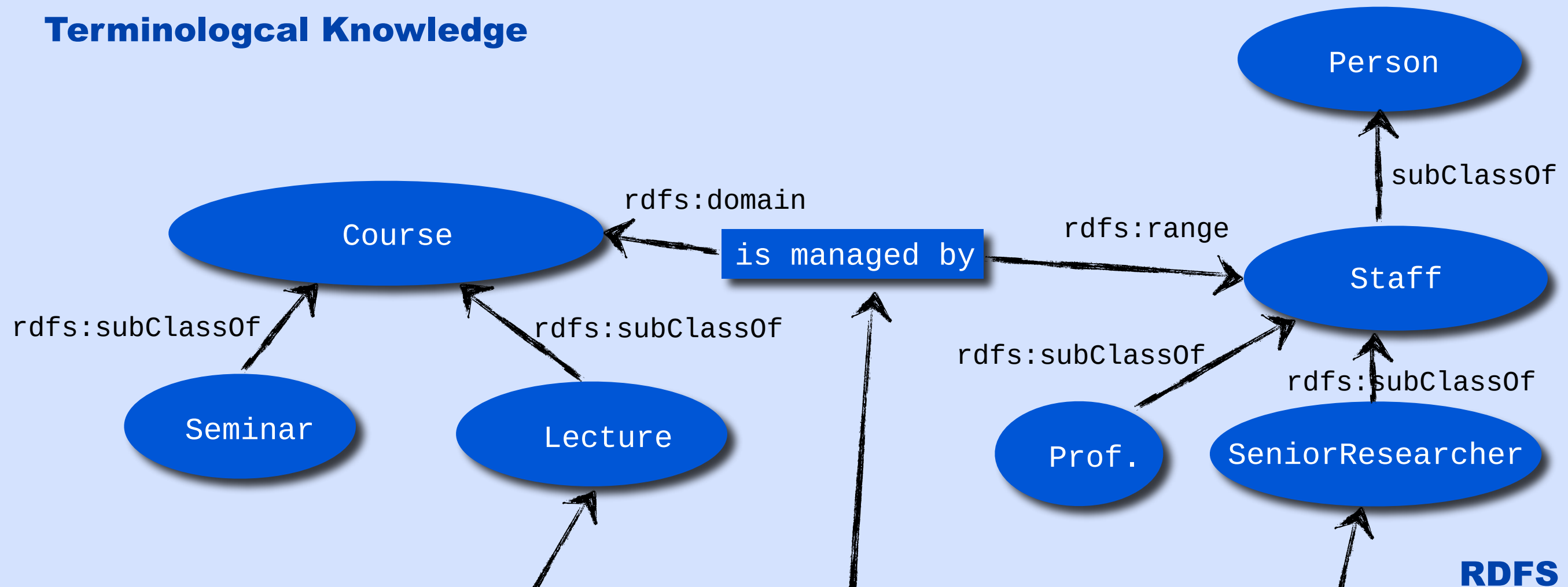


RDF(S) Knowledge Base

15

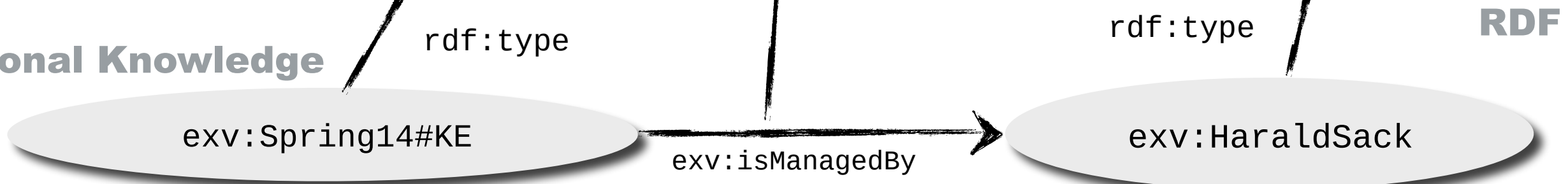
T-Box

Terminological Knowledge



A-Box

Assertional Knowledge




```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> . Turtle
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://example.org/> .
```

```
:Lecture a rdfs:Class;
    rdfs:subClassOf :Course.
:Seminar a rdfs:Class ;
    rdfs:subClassOf :Course.
```

```
:Person a rdfs:Class .
:Staff a rdfs:Class ;
    rdfs:subClassOf :Person .
:SeniorResearcher a rdfs:Class ;
    rdfs:subClassOf :Staff .
:Professor a rdfs:Class ;
    rdfs:subClassOf :Staff.
:isManagedBy a rdf:Property ;
    rdfs:domain :Course ;
    rdfs:range :Staff .
```

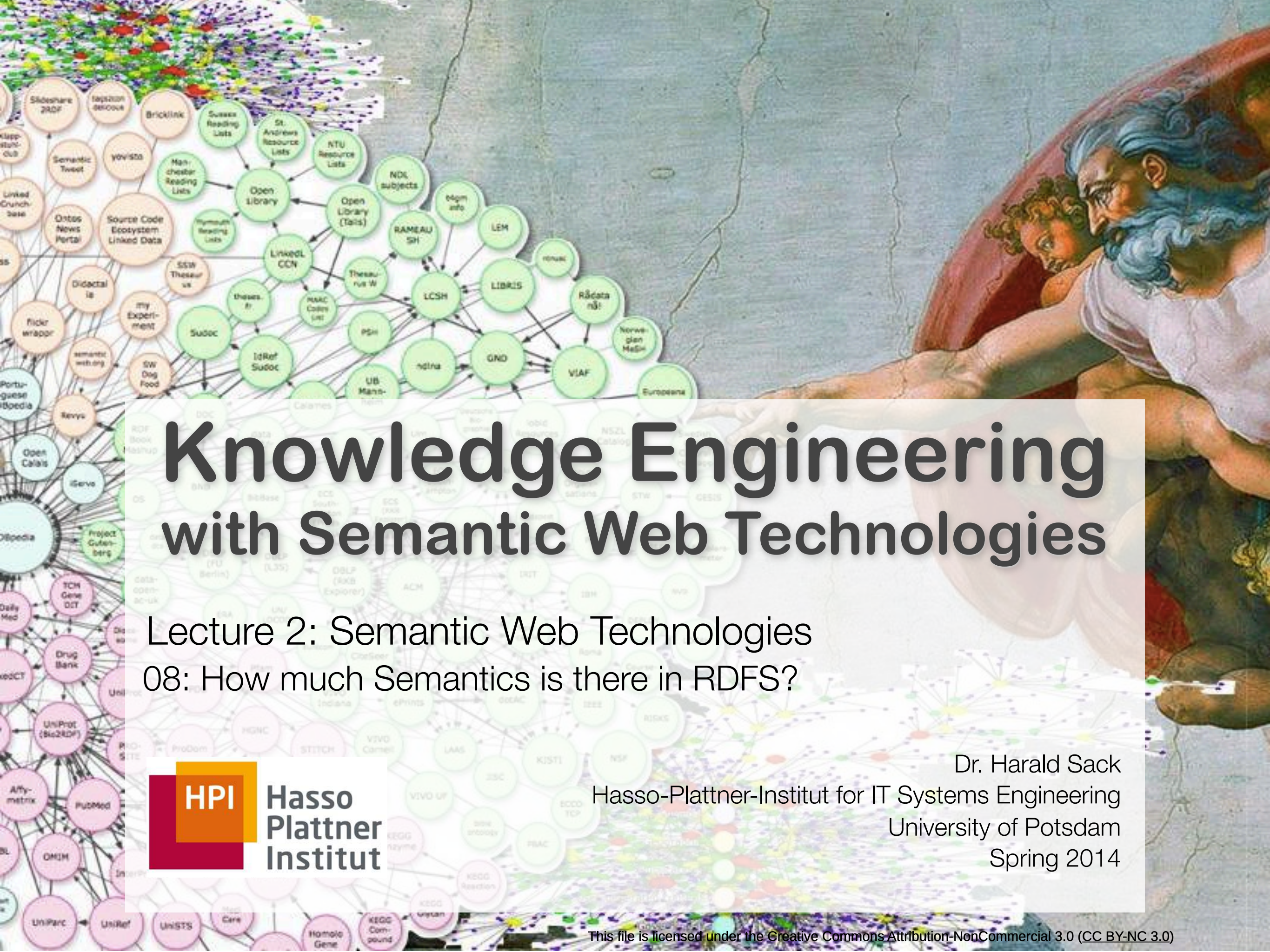
```
<Spring14#KE> a :Lecture .
:HaraldSack a :SeniorResearcher .
<Spring14#KE> :isManagedBy :HaraldSack .
```

- RDFSchema defines a data model for the definition of simple ontologies (knowledge representations).
- Via RDFSchema ontologies RDF statements (facts) can be expressed
- More than XML:
 - (small) ontological agreement about modelling primitives
 - Possibility to define own vocabularies



08 How much semantics is there in RDF(S)

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I



Knowledge Engineering with Semantic Web Technologies

Lecture 2: Semantic Web Technologies
08: How much Semantics is there in RDFS?



Dr. Harald Sack
Hasso-Plattner-Institut for IT Systems Engineering
University of Potsdam
Spring 2014



Lecture 2: Semantic Web Technologies - Part 1

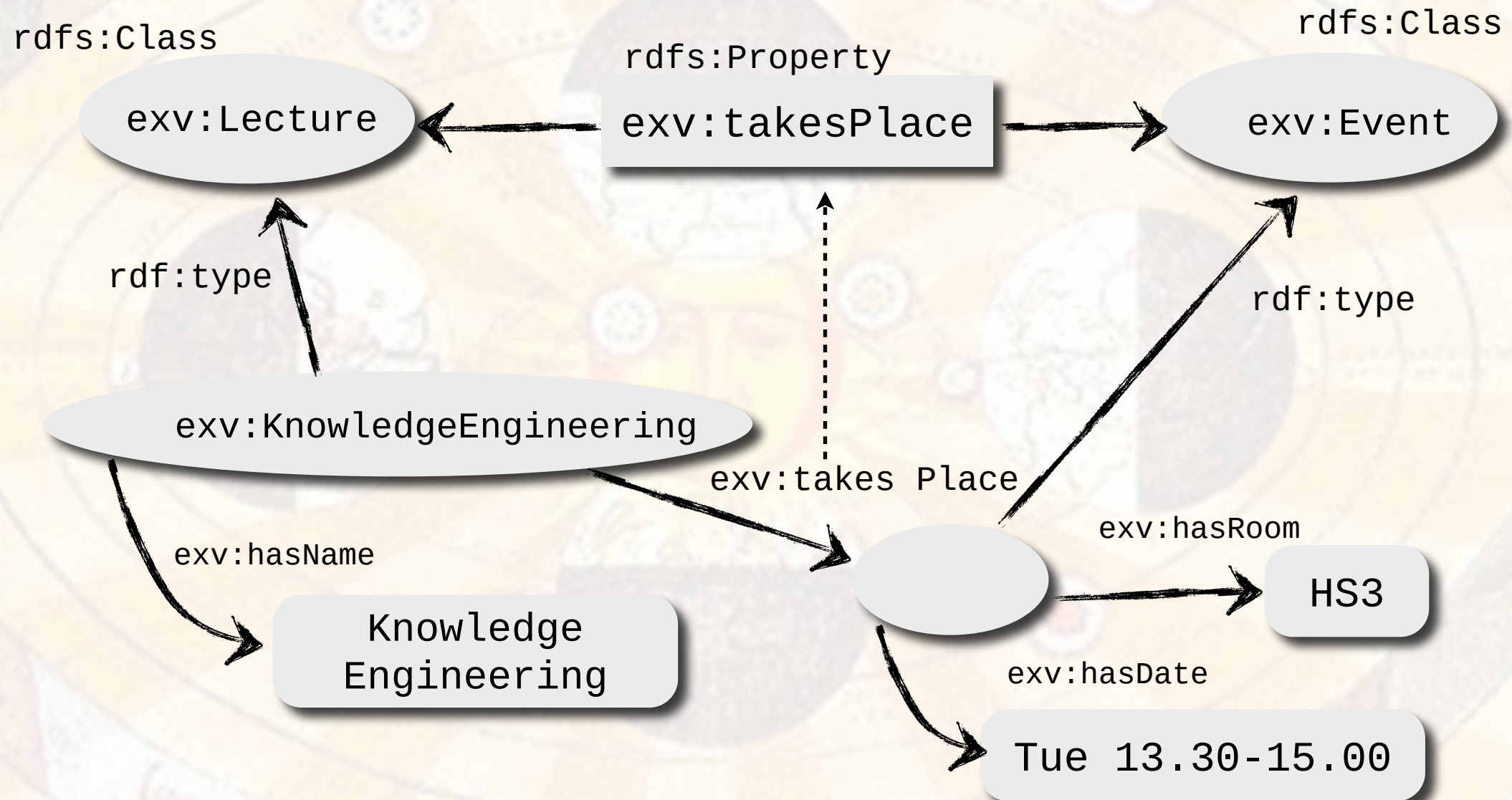
Open HPI - Course: Knowledge Engineering with Semantic Web Technologies



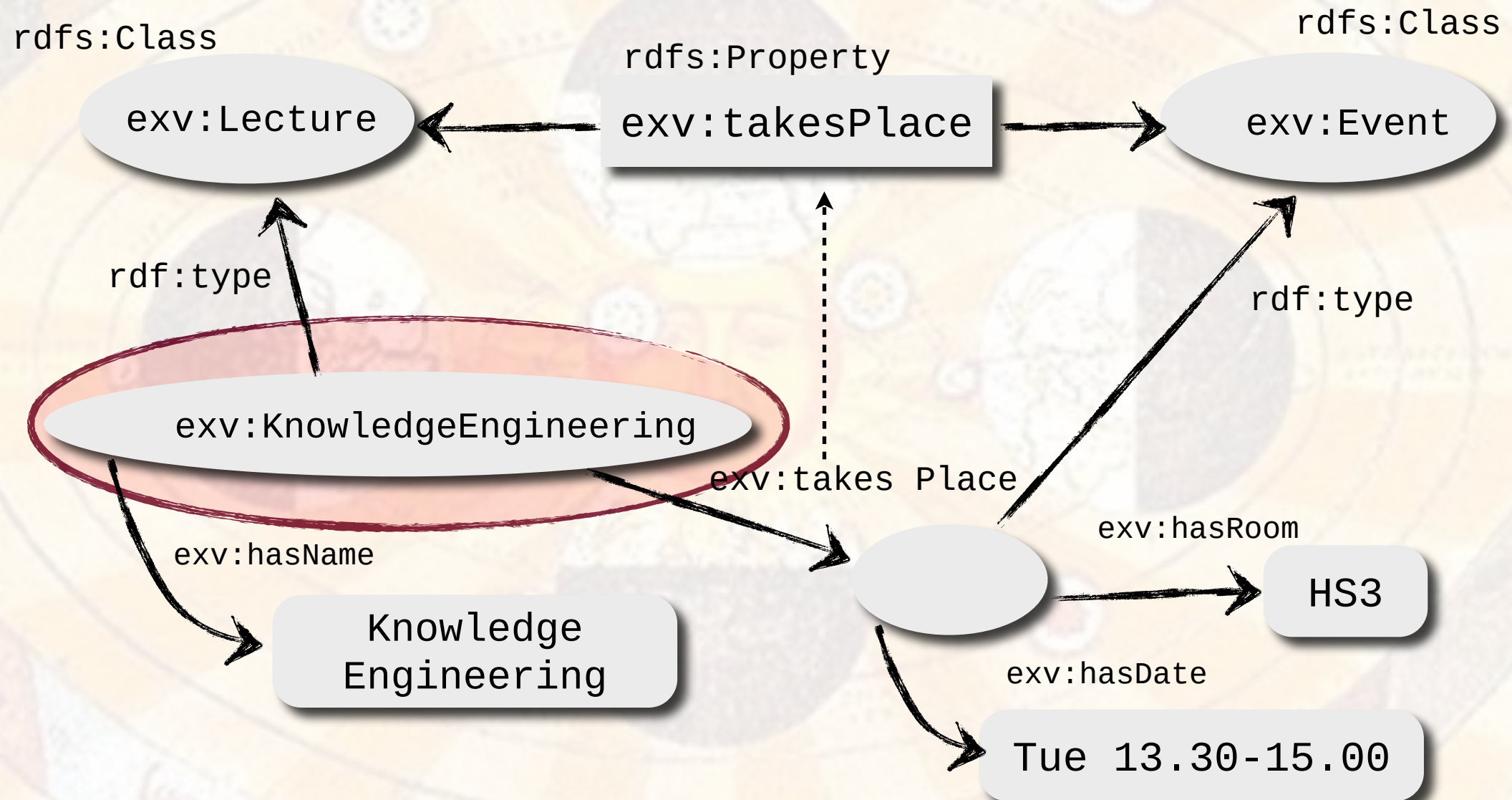
08 How much Semantics is there in RDF(S)

Open HPI - Course: Knowledge Engineering with Semantic Web Technologies
Lecture 2: Semantic Web Technologies Part I

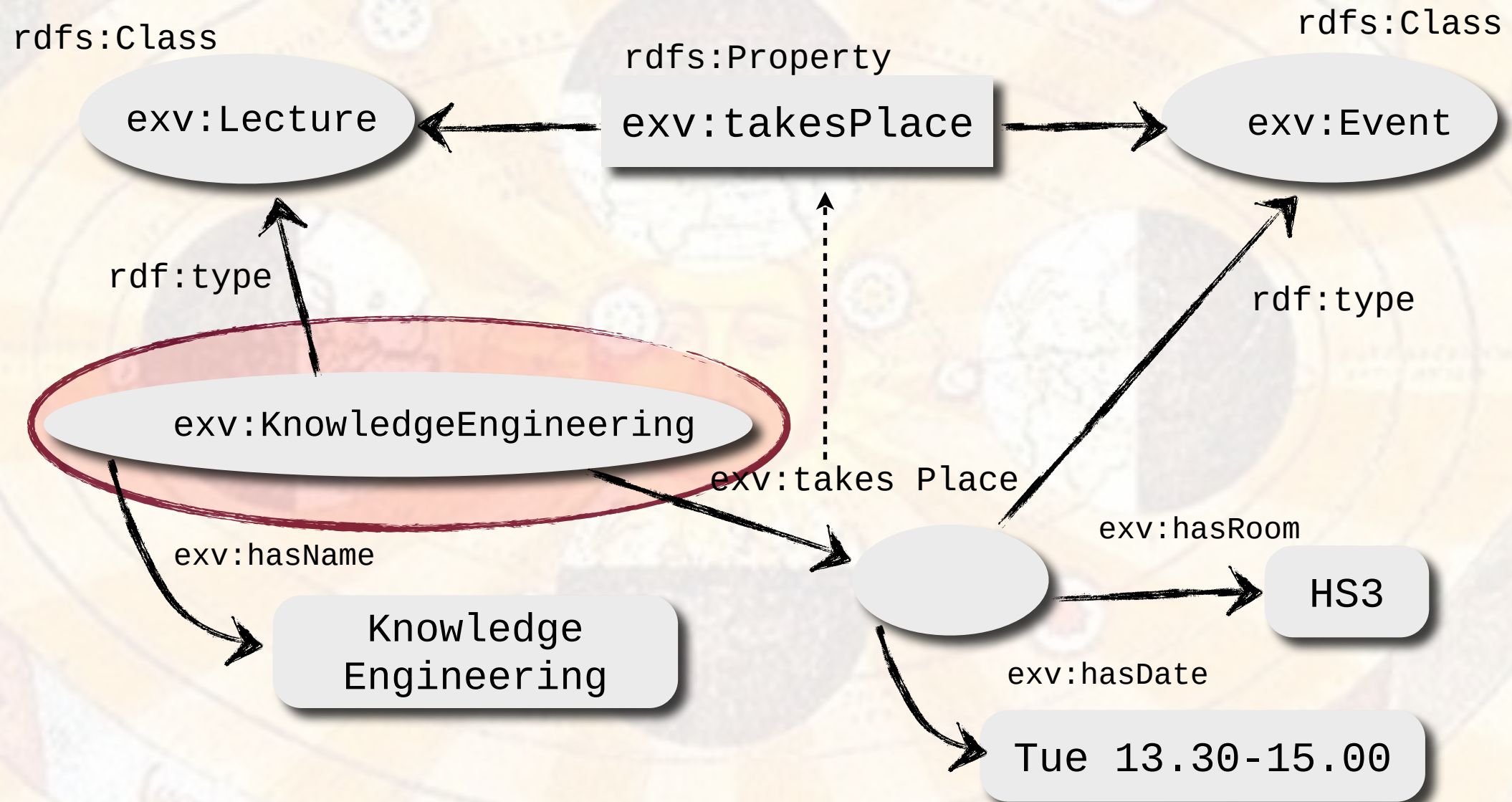
How much knowledge (semantics) is there in RDF(S)



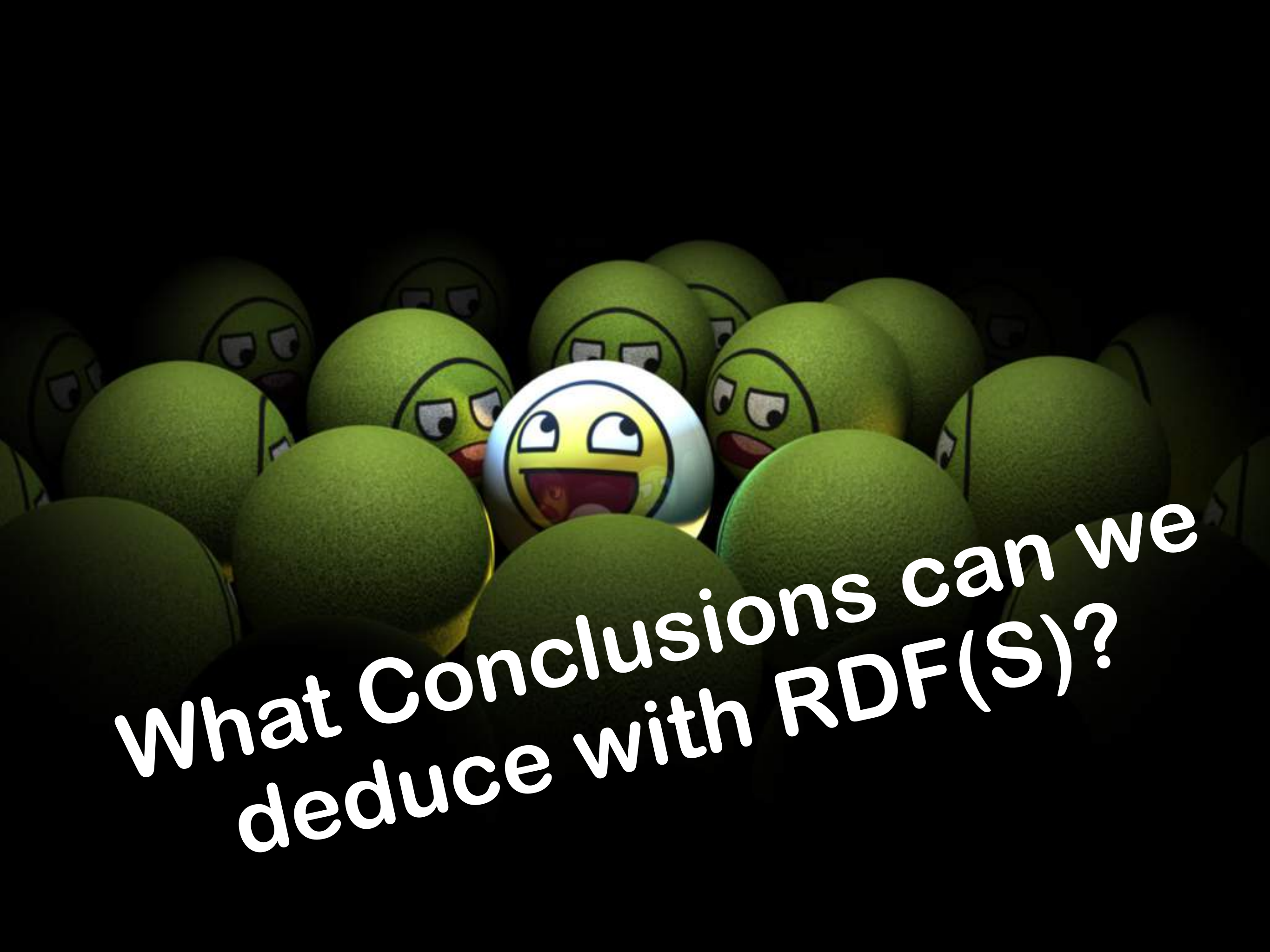
How much knowledge (semantics) is there in RDF(S)



How much knowledge (semantics) is there in RDF(S)

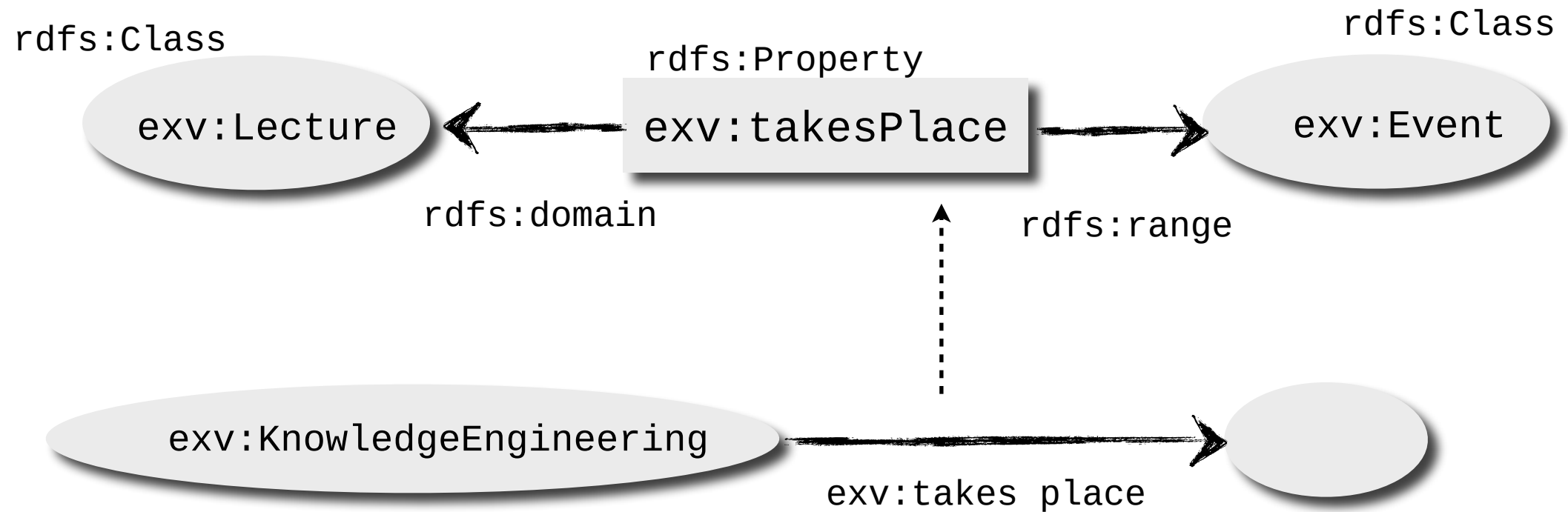


- The semantics of a term from an RDF(S) ontology is given in terms of its properties and its values (instances)

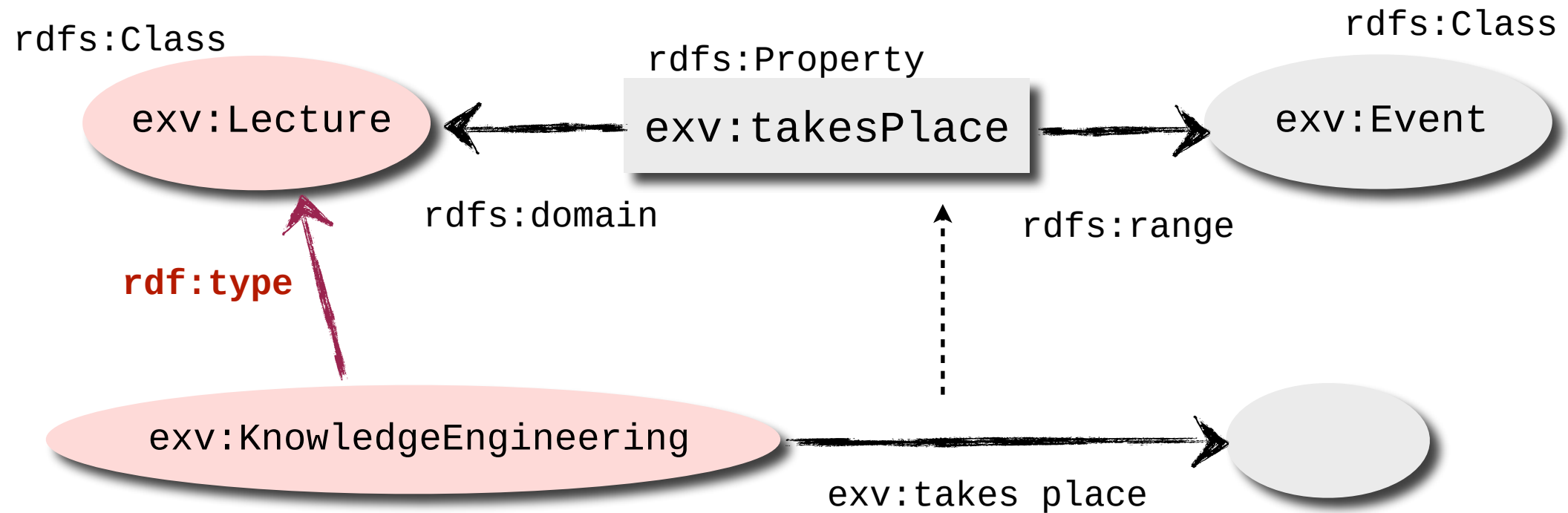


What Conclusions can we
deduce with RDF(S)?

What conclusions can we deduce with RDF(S)?

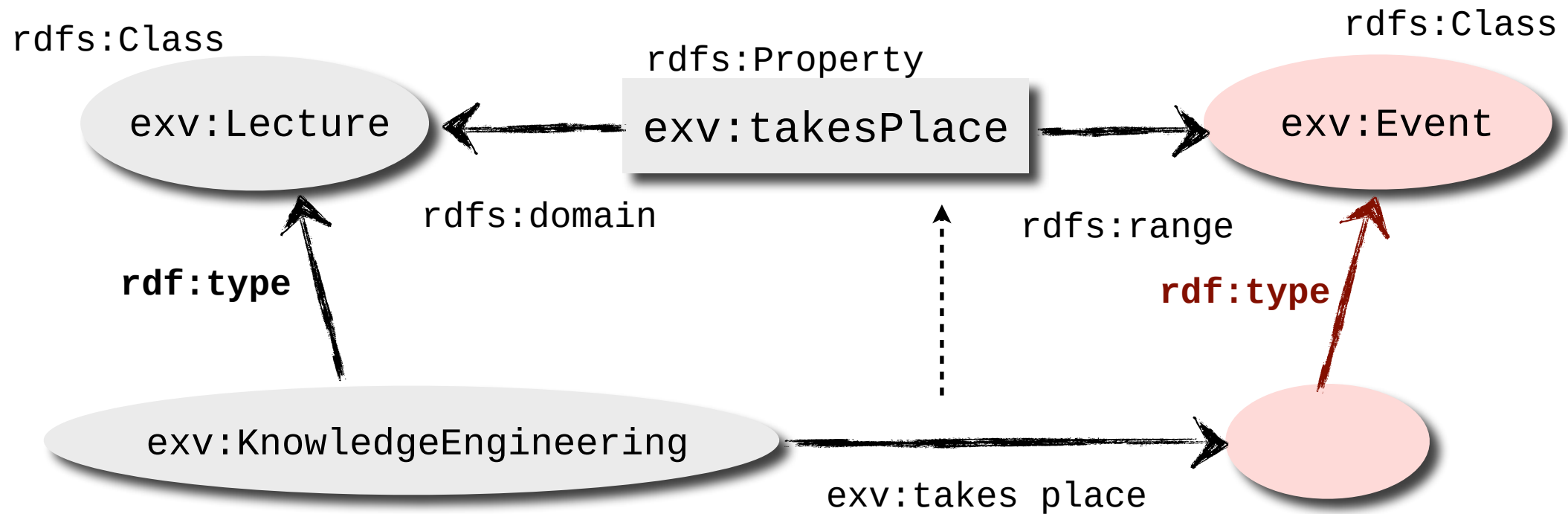


What conclusions can we deduce with RDF(S)?



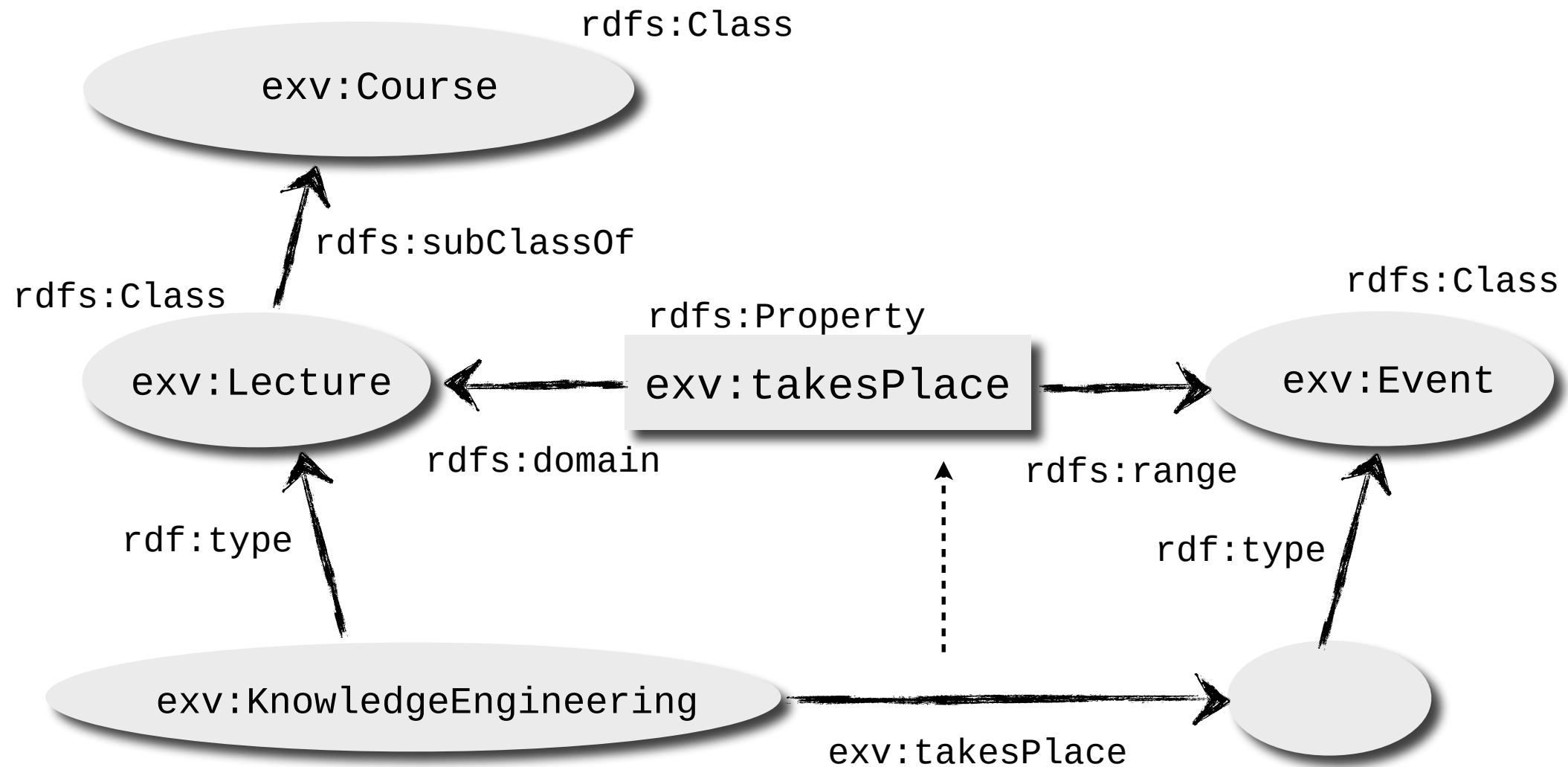
- Deduction of entity **class membership** from the domain of one of its properties

What conclusions can we deduce with RDF(S)?

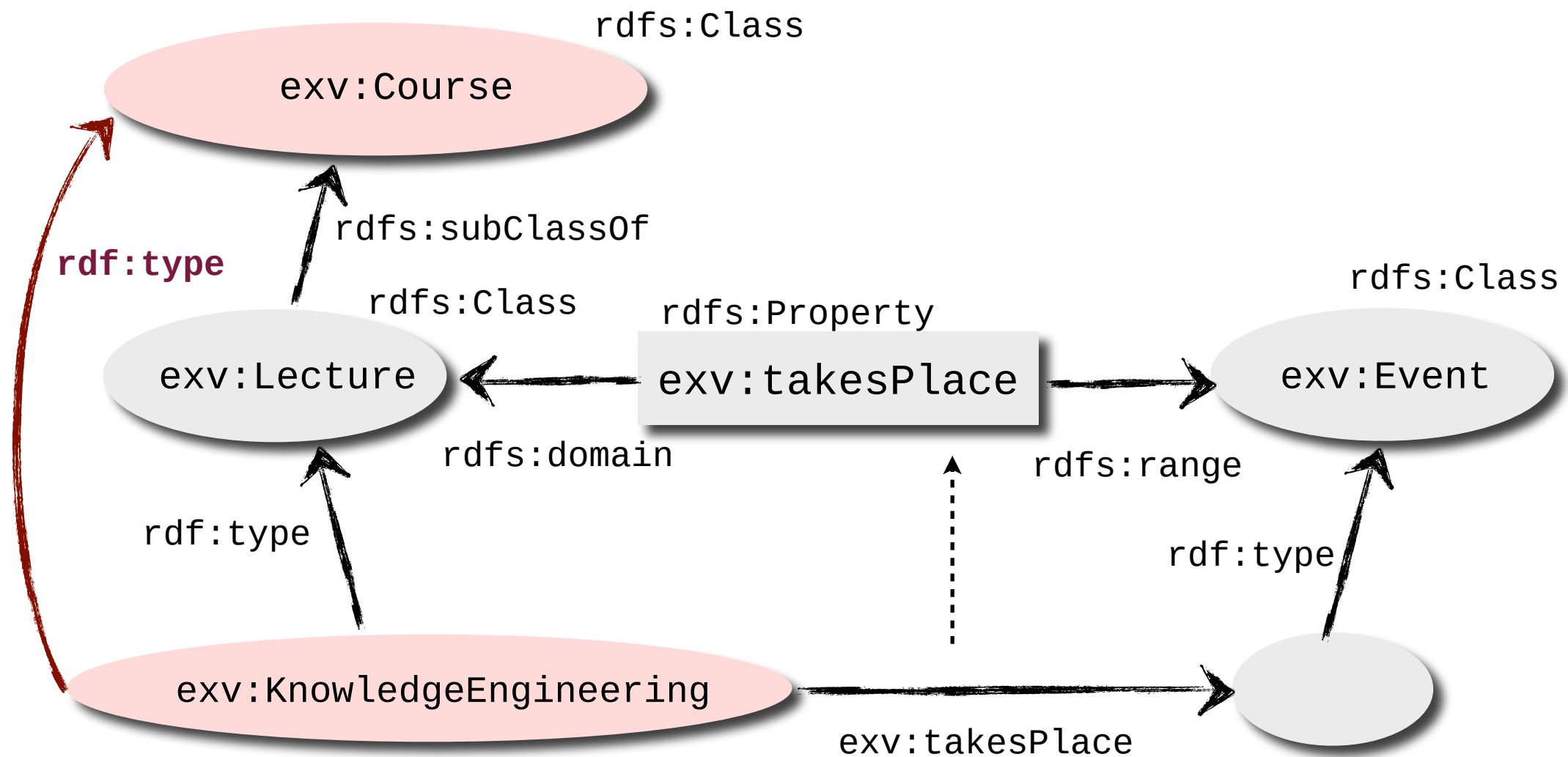


- Deduction of entity **class membership** from the range of one of its properties

What conclusions can we deduce with RDF(S)?

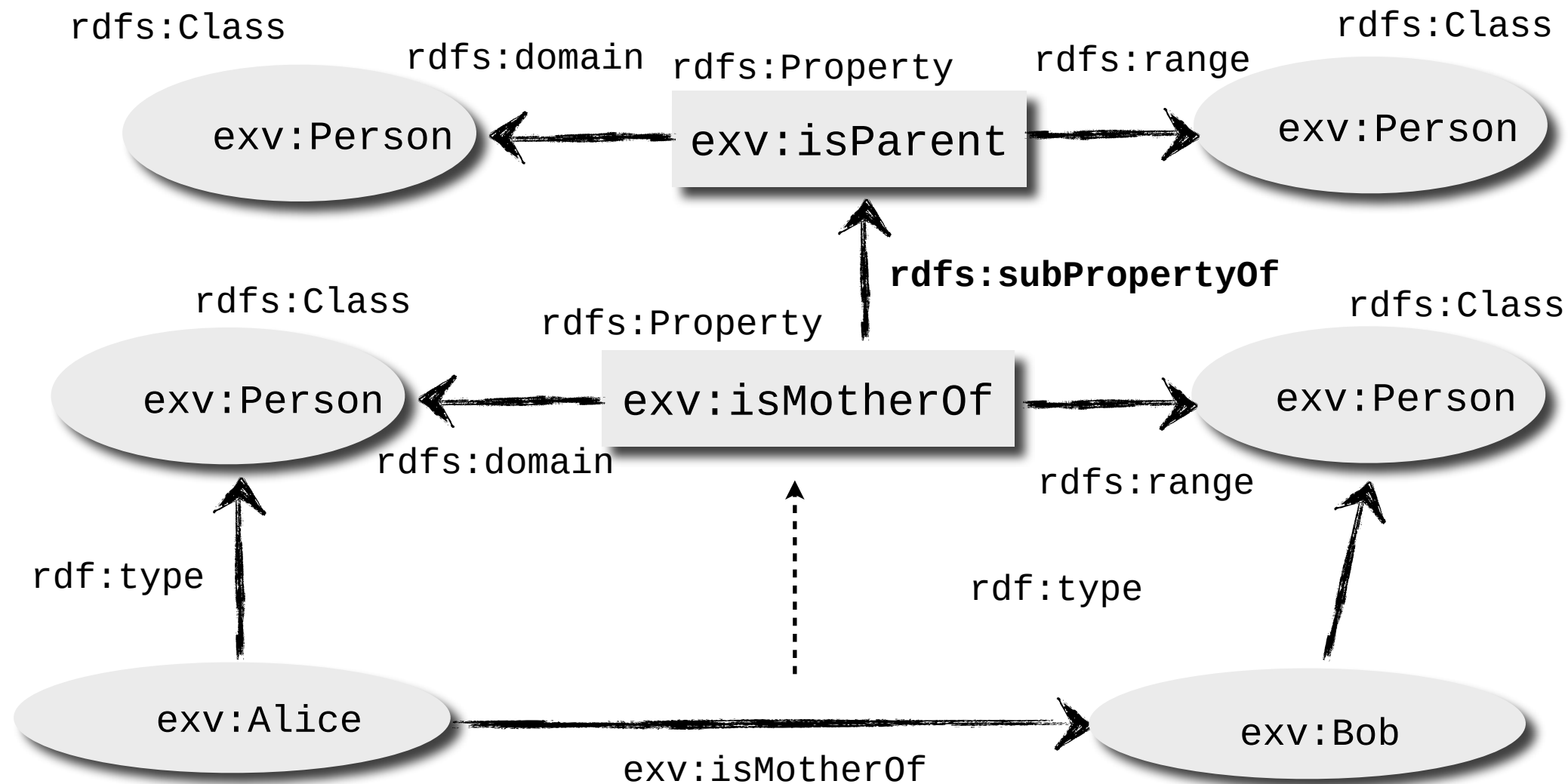


What conclusions can we deduce with RDF(S)?

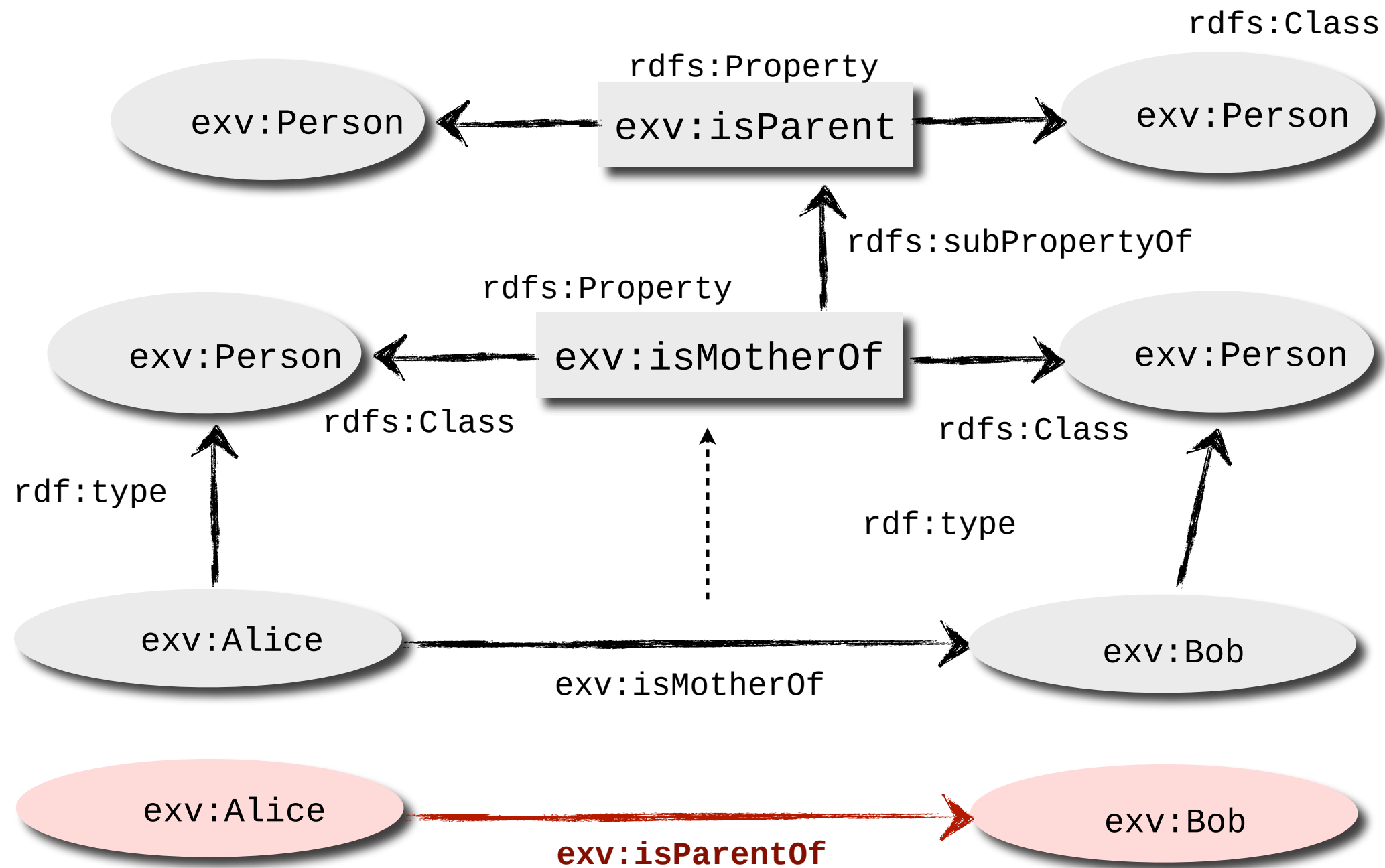


- Deduction of entity super**class membership** from a class hierarchy.

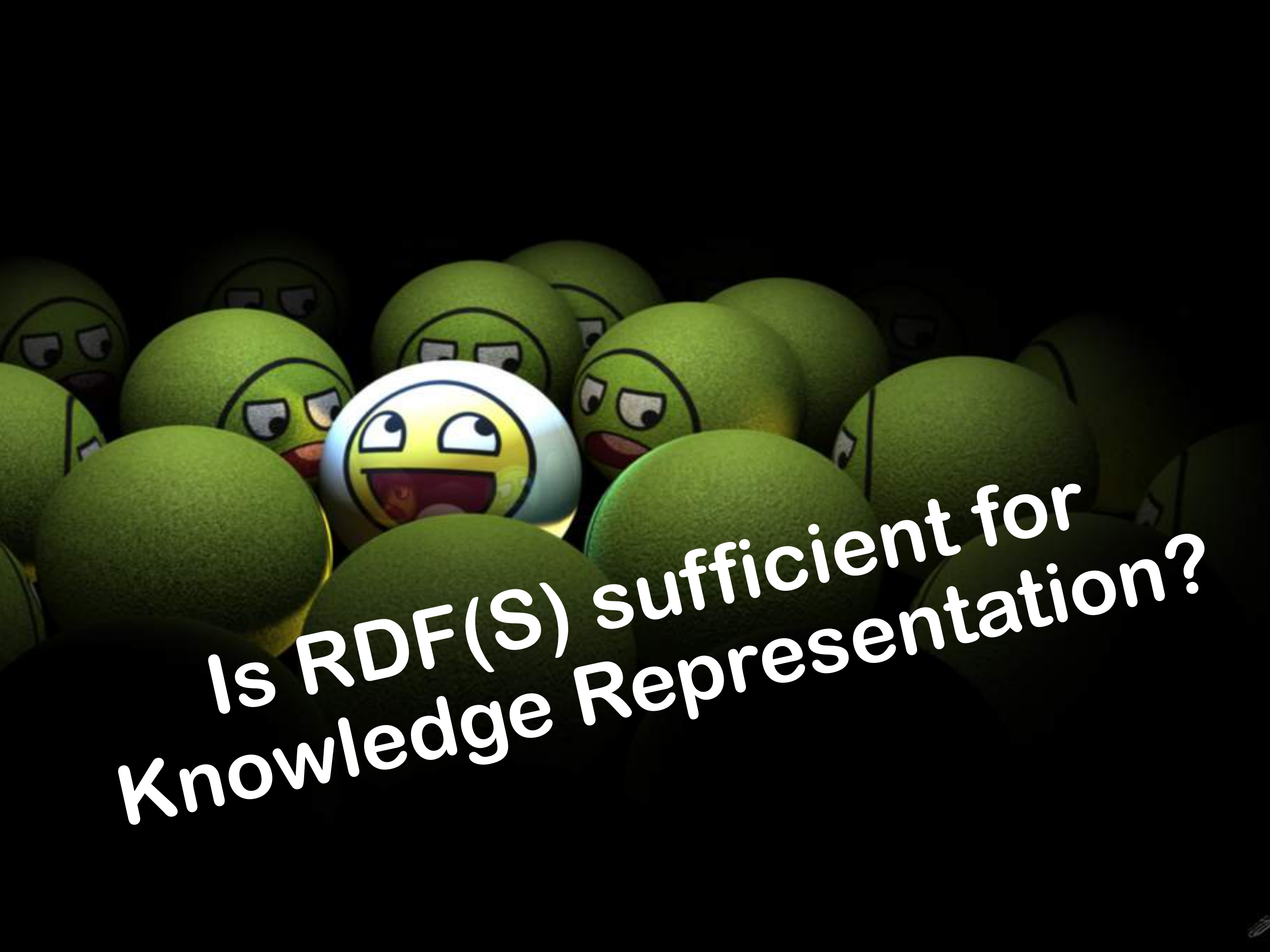
What conclusions can we deduce with RDF(S)?



What conclusions can we deduce with RDF(S)?



- Deduction of **new facts** from subproperty relationships



Is RDF(S) sufficient for
Knowledge Representation?



Lecture 3.1 How to query RDF(S) ? - SPARQL

Knowledge Engineering with Semantic Web Technologies - Semantic Web Technologies Pt.2