

Logic and Reasoning in the Semantic Web (part I – RDF/RDFS)



Fulvio Corno, Laura Farinetti

Politecnico di Torino

Dipartimento di Automatica e Informatica

e-Lite Research Group – <http://elite.polito.it>

Outline

- Reasoning in Semantic Web Knowledge Bases
- RDF/RDFS Semantics and Entailments
- OWL Semantics
- OWL reasoning tools

Part I

Outline

- **Reasoning in Semantic Web Knowledge Bases**
- RDF/RDFS Semantics and Entailments
- OWL Semantics
- OWL reasoning tools

Reasoning

- Reasoning is required when a program must determine some information or some action that has not been explicitly told about
- It must figure out what it needs to know from what it already knows

Example

■ Facts:

- ☐ Robins are birds
- ☐ All birds have wings

■ Questions:

- ☐ Are Robins birds? → Yes
- ☐ Do all birds have wings? → Yes
- ☐ Do robins have wings? → ?????

Semantic Web peculiarities

- Traditional KBS used to be developed top down
- The Semantic Web is developed bottom-up
- Problems:
 - **Scalability:** Everybody having a web page is a potential Knowledge Engineer
 - **Distribution:** Pieces of knowledge can be all over the place, finding them can be hard
 - **Heterogeneity:** People use different ways of describing the same information
 - **Quality:** Knowledge will be incomplete and even inconsistent across different sources of mapping approaches

Reasoning in the Semantic Web

- New methods are needed that
 - Scale to large amounts of data and knowledge
 - Are tolerant to errors, incompleteness and inconsistency between and within sources
 - Work on distributed representations and ideally are distributed themselves
 - ...and are compatible with Semantic web standards

What kind of reasoning can we expect?

- Depends on
 - The semantic level of the representation (RDFS, OWL dialect), that implies a given mathematical formalization for the knowledge base
 - The size of the involved knowledge base
 - The presence (or absence) of instances
 - The capabilities of the reasoning tool

Definitions

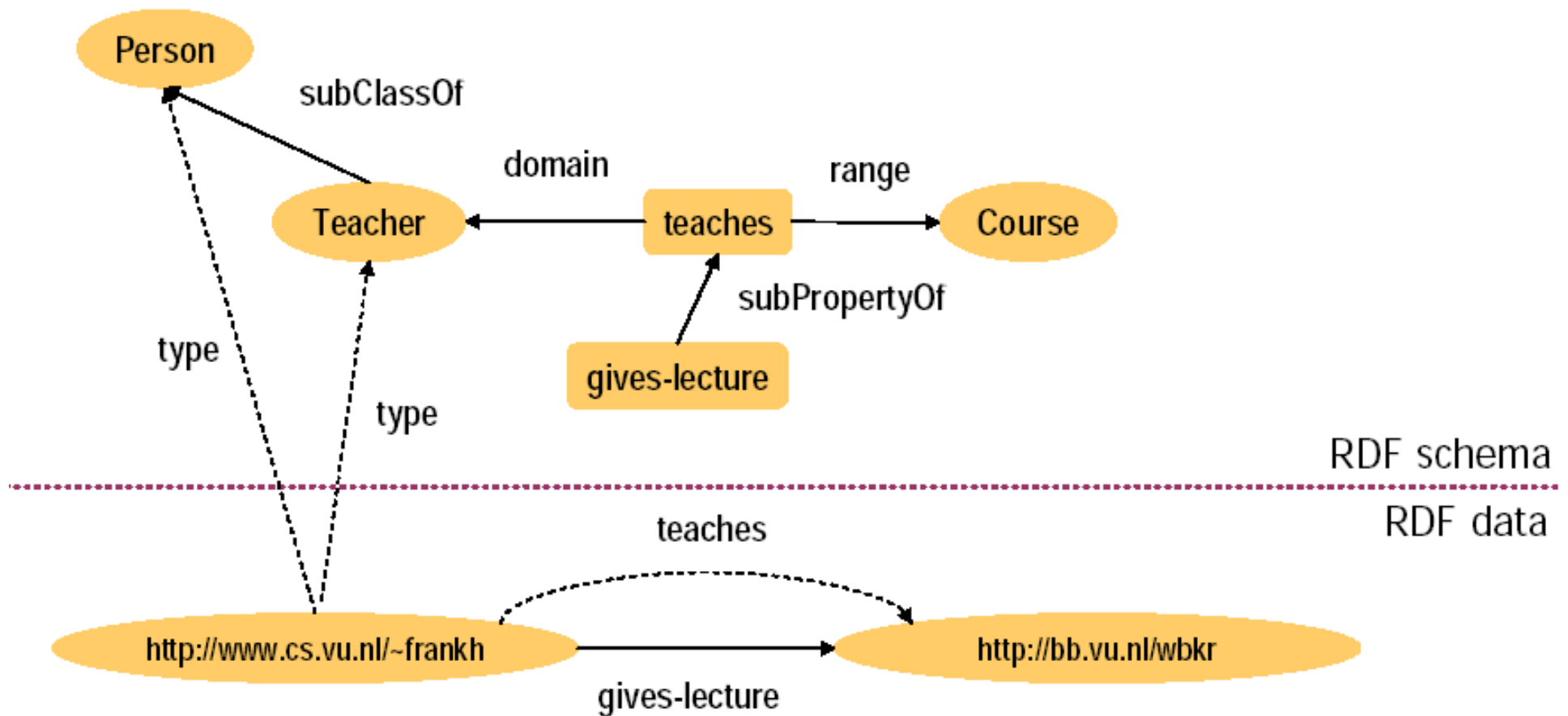
Inference (n.) An act or process of constructing new expressions from existing expressions, or the result of such an act or process. In RDF, inferences corresponding to [entailments](#) are described as *correct* or *valid*.

Inference rule, formal description of a type of inference; **inference system**, organized system of inference rules; also, software which generates inferences or checks inferences for validity.

Outline

- Reasoning in Semantic Web Knowledge Bases
- **RDF/RDFS Semantics and Entailments**
- OWL Semantics
- OWL reasoning tools

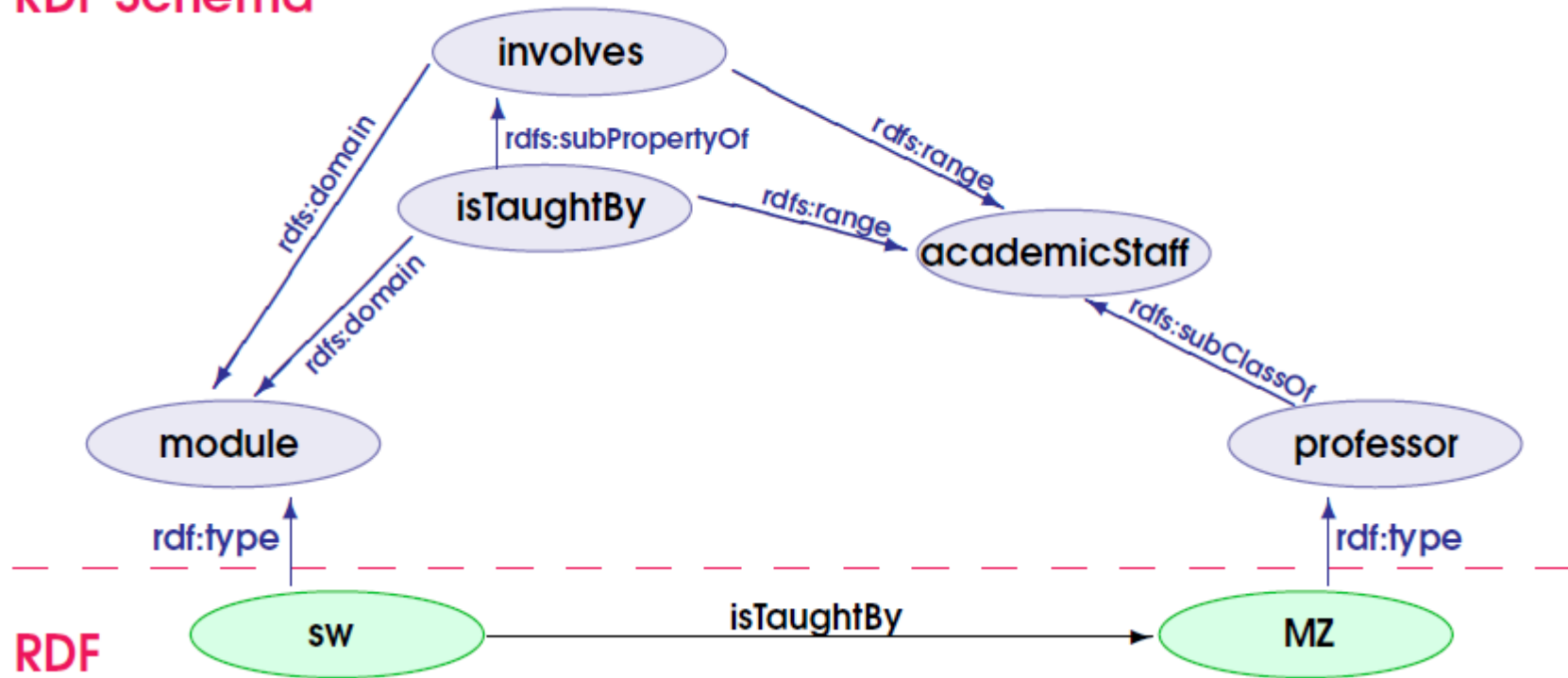
Recall: RDF Schema



Source: Practical Reasoning for the Semantic Web (ESSLLI2005), S. Schlobach, H. Stuckenschmidt, http://www.few.vu.nl/~schlobac/essli_day1.pdf

RDF vs RDFS

RDF Schema



Some RDFS inference rules

- $(X \text{ R } Y), (R \text{ subPropertyOf } Q) \Rightarrow (X \text{ Q } Y)$
- $(X \text{ R } Y), (R \text{ domain } C) \Rightarrow (X \text{ type } C)$
- $(X \text{ type } C), (C \text{ subClassOf } D) \Rightarrow (X \text{ type } D)$
- ...

RDF Semantics

- Semantics is not about the “meaning” of assertions
 - The ‘meaning’ of an assertion in RDF or RDFS may depend on many factors, including social conventions, comments in natural language or links to other content-bearing documents, ... (non machine-processable information)
 - Semantics restricts itself to **a formal notion of meaning** which could be characterized as the part that is common to all other accounts of meaning, and can be captured in mechanical inference rules

Model Semantics

- The RDF Semantics W3C Recommendation uses model theory for specifying the semantics of the RDF language.
- Model theory assumes that the language refers to a ‘**world**’, and describes the minimal conditions that a world must satisfy in order to assign an appropriate meaning for every expression in the language
- A particular world is called an **interpretation**
- The idea is to provide an abstract, mathematical account of the properties that any such interpretation must have, making as few assumptions as possible about its actual nature or intrinsic structure, thereby **retaining as much generality as possible**

Goals of the inference process

- To provide a **technical** way to determine when inference processes are valid, i.e., when they preserve truth
- Starting from a set of assertions that are *regarded as* true in an RDF model, derive whether a new RDF model contains *provably* true assertions
- We never know about the “real” truth of any assertion in the “real” world.

Formalization

■ Interpretations (Normative)

- Mapping of RDF assertions into an **abstract model**, based on set-theory
- With an “interpretation operator” **I()**, maps a RDF graph into a highly abstract set of high-cardinality sets
- Highly theoretical model, useful to prove mathematical properties

■ Entailments (Informative)

- **Transformation rules** to derive new assertions from existing ones
- May be **proven complete and consistent** with the formal interpretation

Definitions

Interpretation (of) (n.) A minimal formal description of those aspects of a world which is just sufficient to establish the truth or falsity of any expression of a logic.

World (n.) (with *the*;) (i) The actual world. (with *a*;) (ii) A way that the actual world might be arranged. (iii) An interpretation (iv) A possible world.

Definition

Entail (v.), **entailment** (n.). A semantic relationship between expressions which holds whenever the truth of the first guarantees the truth of the second. Equivalently, whenever it is logically impossible for the first expression to be true and the second one false. Equivalently, when any interpretation which satisfies the first also satisfies the second. (Also used between a set of expressions and an expression.)

Interpretation: minimum notions

- Let V be a vocabulary containing all names (URIs and literals) occurring in RDF triples
- An RDF **interpretation** of V consists of:
 - IR , a non-empty set of **resources** (domain or universe)
 - $IP \subseteq IR$, a set of **properties** (each property is also a resource)
(symbol $v \in V$ is a property if, and only if, $IS(v) \in IP$)
 - $IS: V \rightarrow IR$, an interpretation of resources
(with each symbol from V a resource is associated)
 - $IEXT: IP \rightarrow 2^{IR \times IR}$, an interpretation of properties
(each property is a binary relation, i.e., a subset of $IR \times IR$)

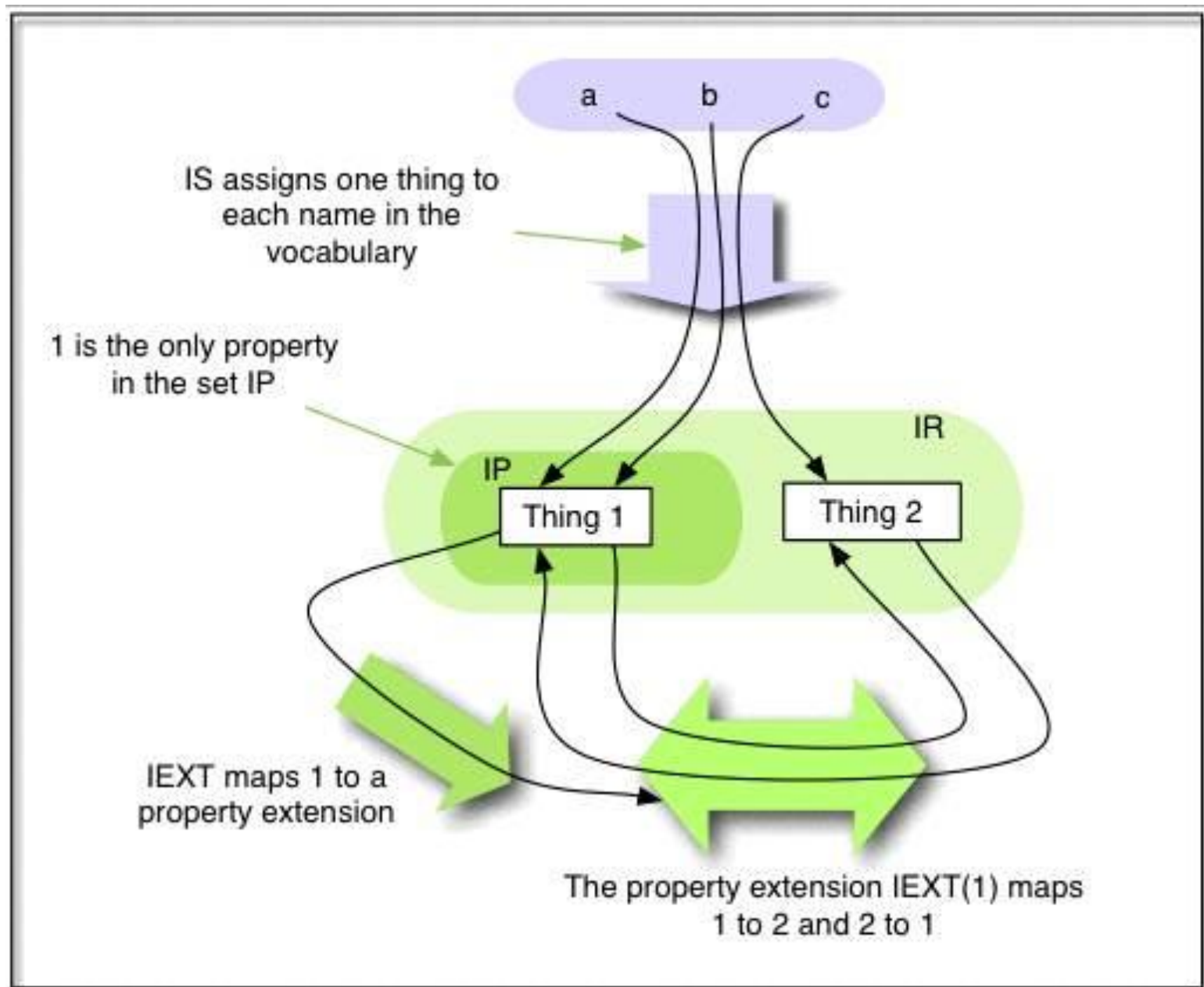
An RDF triple $\langle s, p, o \rangle$ is **true** in I if, and only if,
 $s, p, o \in V$, $IS(p) \in IP$ and $(IS(s), IS(o)) \in IEXT(IS(p))$

An RDF triple is false in I if it is not true in I

An RDF graph is **true** in I if, and only if, every triple of it is true in I

In particular, it is false in I if some triple is not true in I

Conceptual framework



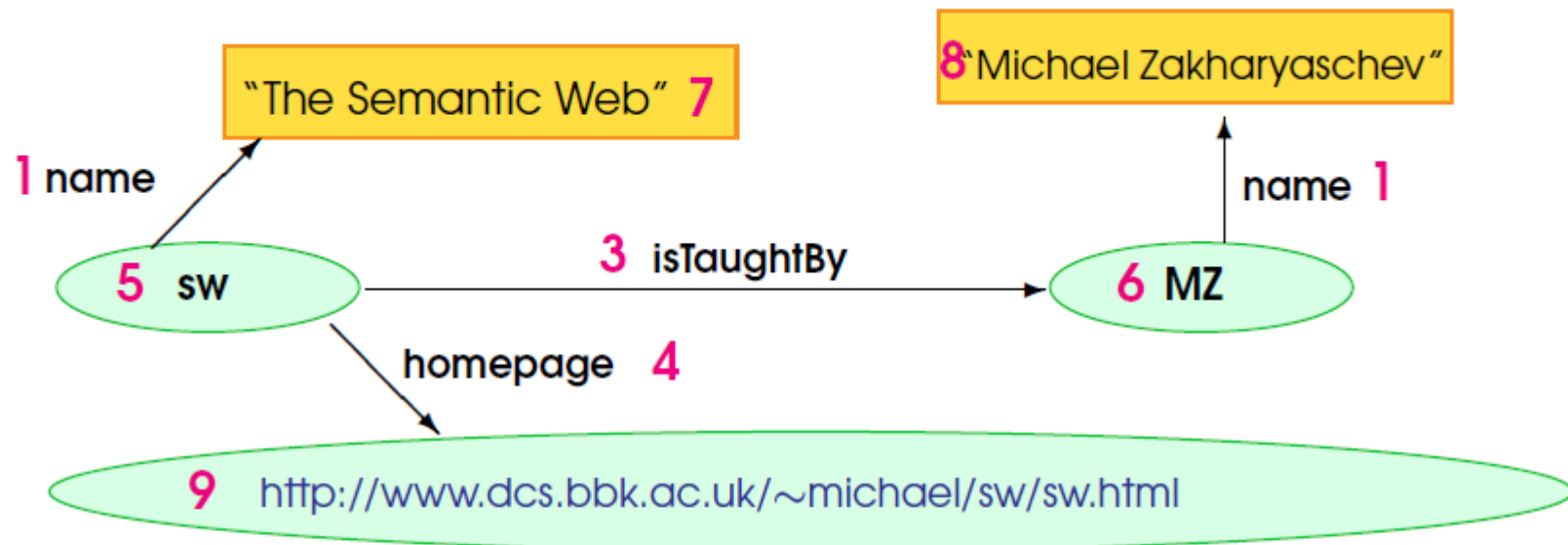
Interpretation: minimum notions



- I : interpretation operator
- E : fragment of RDF syntax
- If E is a graph:
 - $I(E) = \text{false}$ if $I(E') = \text{false}$ for some triple E' in E , otherwise $I(E) = \text{true}$
- If E is a triple $\langle s, p, o \rangle$:
 - $I(E) = \text{true}$ if s, p and o are in V , $I(p)$ is in IP and $\langle I(s), I(o) \rangle$ is in $IEXT(I(p))$
 - otherwise $I(E) = \text{false}$
- IP : set of properties, IR : set of resources
- $IEXT(I(p))$: mapping from IP into the powerset of $IR \times IR$ i.e. the set of sets of pairs $\langle x, y \rangle$ with x and y in IR

Example

Construct an interpretation in which the following RDF graph is true:



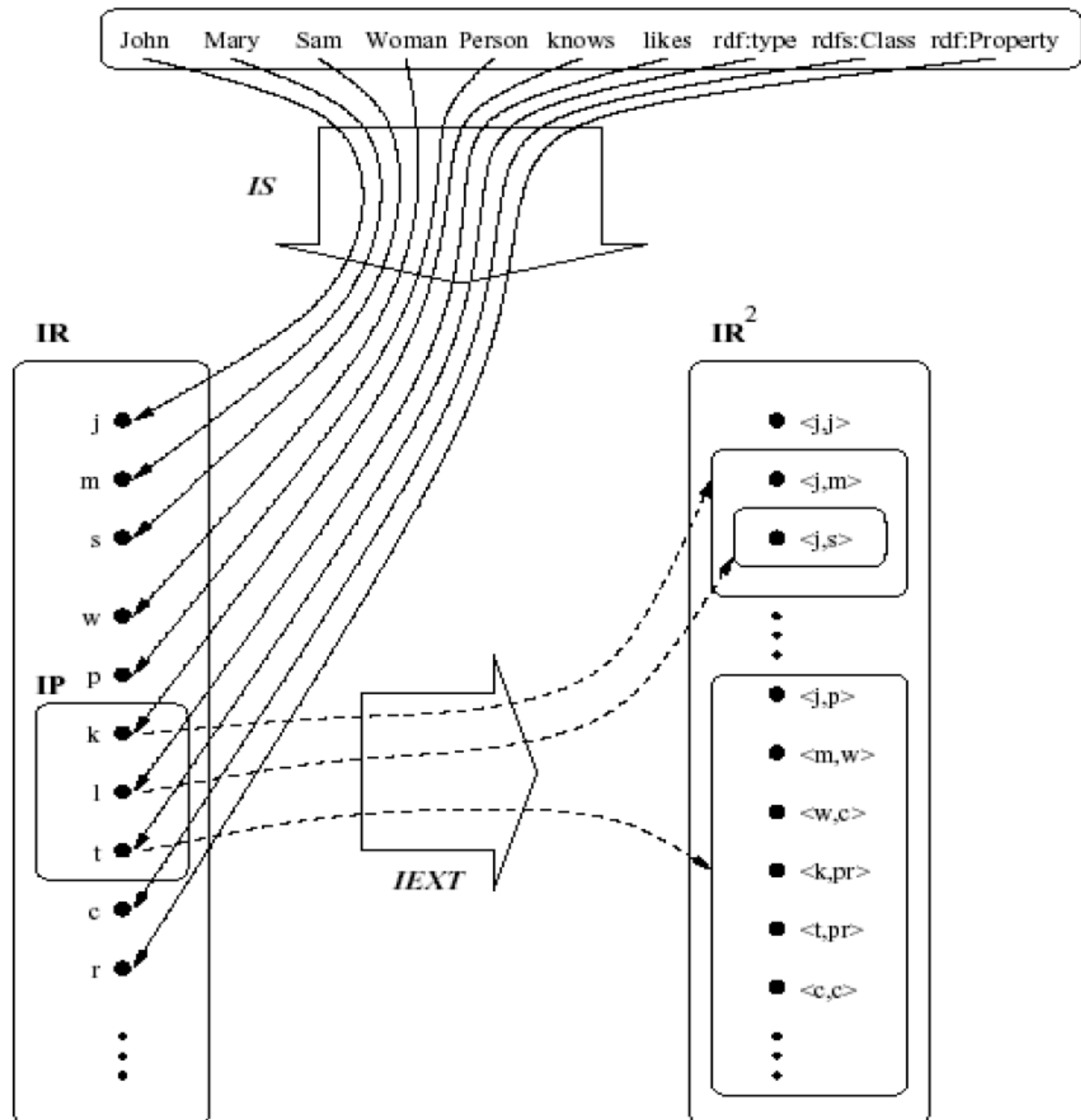
$IR = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and $IP = \{1, 2, 3, 4\}$

$IS(\text{name}) = 1$, $IS(\text{isTaughtBy}) = 3$, $IS(\text{homepage}) = 4$, $IS(\text{sw}) = 5$,
 $IS(\text{MZ}) = 6$, $IS(\text{http://www.dcs.bbk.ac.uk/~michael/sw/sw.html}) = 9$,
 $IS(\text{"The Semantic Web"}) = 7$, $IS(\text{"Michael Zakharyashev"}) = 8$

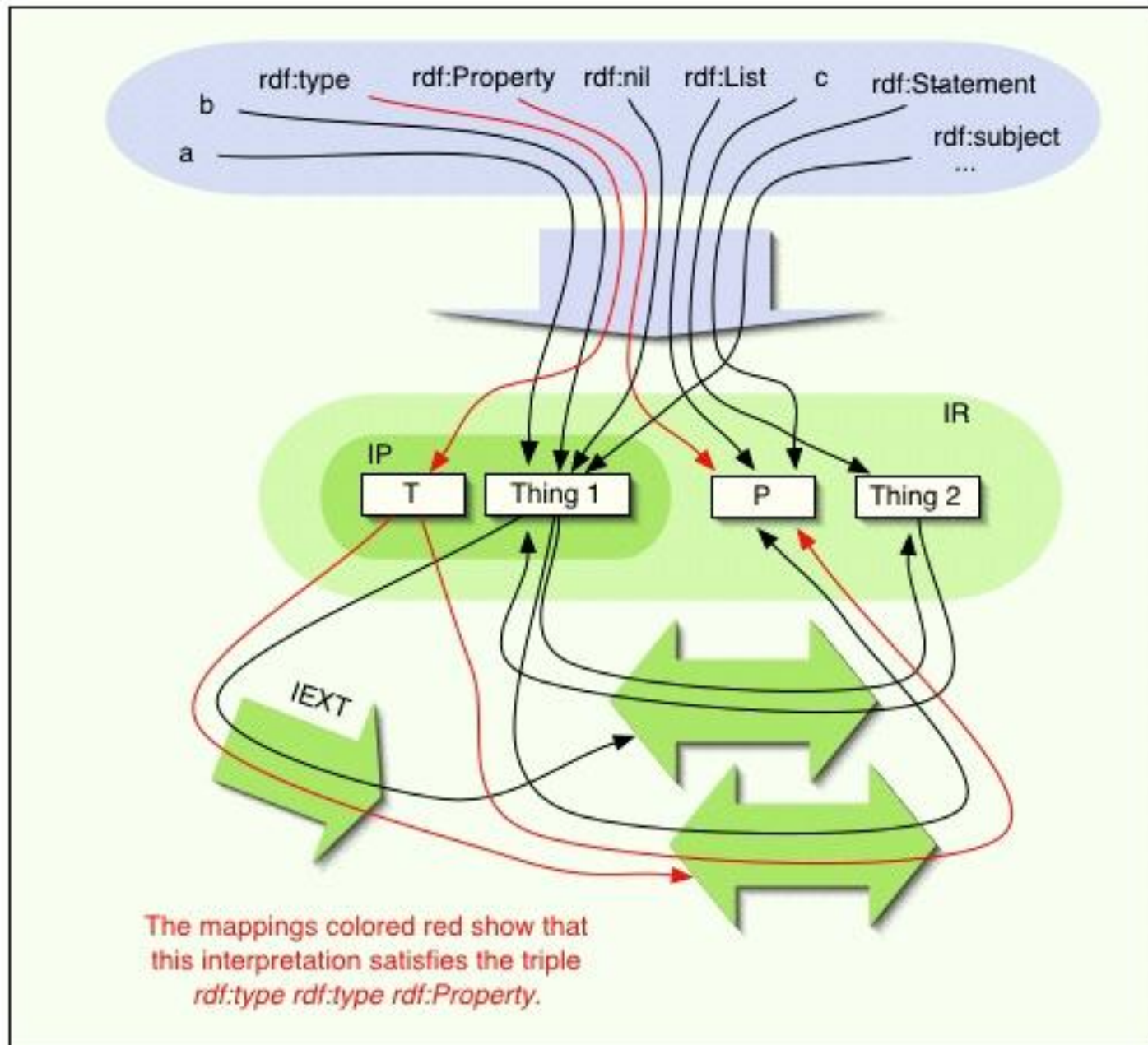
$IEXT(1) = \{(5, 7), (6, 8)\}$, $IEXT(2) = \emptyset$, $IEXT(3) = \{(5, 6)\}$,

$IEXT(4) = \{(5, 9)\}$

Example



Example of interpretation



Entailment

- *I satisfies E* if $I(E)=\text{true}$
- A set *S* of RDF graphs **entails** a graph *E* if every interpretation which satisfies every member of *S* also satisfies *E*
- In human words:
 - assertion = a claim that the world is an interpretation which assigns the value true to the assertion
 - If *A* entails *B*, then
 - any interpretation that makes *A* true also makes *B* true
 - an assertion of *A* already contains the same "meaning" as an assertion of *B*
 - the meaning of *B* is somehow contained in, or subsumed by, that of *A*

Entailment regime

- RDF defines 4 entailment regimes:
 - Simple entailment : does **not** interpret any RDF or RDFS vocabulary (just structural matching, by re-naming blank nodes)
 - RDF entailment : interprets RDF vocabulary
 - RDFS entailment : interprets RDF and RDFS vocabularies
 - D entailment : additional support for datatypes

RDF Entailment rules (legend)

- `aaa`, `bbb`: any URI reference
- `uuu`, `vvv`: any URI reference or blank node identifier
- `xxx`, `yyy`: any URI reference, blank node identifier or literal
- `lll`: any literal
- `_:nnn`: blank node identifiers

Simple Entailment rules

Rule name	If E contains	then add
se1	uuu aaa xxx .	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to xxx by rule se1 or se2.
se2	uuu aaa xxx .	_:nnn aaa xxx . where _:nnn identifies a blank node allocated to uuu by rule se1 or se2.
lg	uuu aaa lll .	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to the literal lll by this rule.
gl	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to the literal lll by rule lg.	uuu aaa lll .

Role of Entailment rules (1/3)

- Simple entailment satisfies the “Interpolation Lemma”:
 - S entails a graph E if and only if a subgraph of S is an instance of E
 - It completely characterizes simple RDF entailment in syntactic terms
- RDF entailment satisfies the “RDF entailment lemma”:
 - S rdf-entails E if and only if there is a graph which can be derived from S plus the RDF axiomatic triples by the application of rule lg and the RDF entailment rules and which simply entails E
 - This means that the entailment rules are “complete”

RDF Entailment rules

Rule name	If E contains	then add
rdf1	uuu aaa yyy .	aaa rdf:type rdf:Property .
rdf2	uuu aaa lll . where lll is a well-typed XML literal .	_:nnn rdf:type rdf:XMLLiteral . where _:nnn identifies a blank node allocated to lll by rule lg.

RDF Axiomatic triples

<code>rdf:type</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:subject</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:predicate</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:object</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:first</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:rest</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:value</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:_1</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>rdf:_2</code>	<code>rdf:type</code>	<code>rdf:Property</code> .
<code>...</code>		
<code>rdf:nil</code>	<code>rdf:type</code>	<code>rdf:List</code> .

Role of Entailment rules (2/3)

- RDF entailment satisfies the “RDF entailment lemma”:
 - S rdf-entails E if and only if there is a graph which can be derived from S plus the RDF axiomatic triples by the application of rule lg and the RDF entailment rules and which simply entails E
 - This means that the entailment rules are “complete”

RDFS Entailment rules (1/2)

Rule name	If E contains	then add
rdfs1	uuu aaa lll. where lll is a plain literal (with or without a language tag)	<code>_:nnn rdf:type rdfs:Literal .</code> where <code>_:nnn</code> identifies a blank node allocated to lll by rule lg.
rdfs2	<code>aaa rdfs:domain xxx .</code> <code>uuu aaa yyy .</code>	<code>uuu rdf:type xxx .</code>
rdfs3	<code>aaa rdfs:range xxx .</code> <code>uuu aaa vvv .</code>	<code>vvv rdf:type xxx .</code>
rdfs4a	<code>uuu aaa xxx .</code>	<code>uuu rdf:type rdfs:Resource .</code>
rdfs4b	<code>uuu aaa vvv .</code>	<code>vvv rdf:type rdfs:Resource .</code>
rdfs5	<code>uuu rdfs:subPropertyOf vvv .</code> <code>vvv rdfs:subPropertyOf xxx .</code>	<code>uuu rdfs:subPropertyOf xxx .</code>

RDFS Entailment rules (2/2)

Rule name	If E contains	then add
rdfs6	<code>uuu rdf:type rdf:Property .</code>	<code>uuu rdfs:subPropertyOf uuu .</code>
rdfs7	<code>aaa rdfs:subPropertyOf bbb</code> <code>uuu aaa yyy .</code>	<code>uuu bbb yyy .</code>
rdfs8	<code>uuu rdf:type rdfs:Class .</code>	<code>uuu rdfs:subClassOf</code> <code>rdfs:Resource .</code>
rdfs9	<code>uuu rdfs:subClassOf xxx .</code> <code>vvv rdf:type uuu .</code>	<code>vvv rdf:type xxx .</code>
rdfs10	<code>uuu rdf:type rdfs:Class .</code>	<code>uuu rdfs:subClassOf uuu .</code>
rdfs11	<code>uuu rdfs:subClassOf vvv .</code> <code>vvv rdfs:subClassOf xxx .</code>	<code>uuu rdfs:subClassOf xxx .</code>
rdfs12	<code>uuu rdf:type</code> <code>rdfs:ContainerMembershipPro</code> <code>perty .</code>	<code>uuu rdfs:subPropertyOf</code> <code>rdfs:member .</code>
rdfs13	<code>uuu rdf:type rdfs:Datatype</code>	<code>uuu rdfs:subClassOf</code> <code>rdfs:Literal .</code>

RDFS Axiomatic triples (1/3)

<code>rdf:type</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:domain</code>	<code>rdfs:domain</code>	<code>rdf:Property</code> .
<code>rdfs:range</code>	<code>rdfs:domain</code>	<code>rdf:Property</code> .
<code>rdfs:subPropertyOf</code>	<code>rdfs:domain</code>	<code>rdf:Property</code> .
<code>rdfs:subClassOf</code>	<code>rdfs:domain</code>	<code>rdfs:Class</code> .
<code>rdf:subject</code>	<code>rdfs:domain</code>	<code>rdf:Statement</code> .
<code>rdf:predicate</code>	<code>rdfs:domain</code>	<code>rdf:Statement</code> .
<code>rdf:object</code>	<code>rdfs:domain</code>	<code>rdf:Statement</code> .
<code>rdfs:member</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdf:first</code>	<code>rdfs:domain</code>	<code>rdf:List</code> .
<code>rdf:rest</code>	<code>rdfs:domain</code>	<code>rdf:List</code> .
<code>rdfs:seeAlso</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:isDefinedBy</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:comment</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdfs:label</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .
<code>rdf:value</code>	<code>rdfs:domain</code>	<code>rdfs:Resource</code> .

RDFS Axiomatic triples (2/3)

<code>rdf:type</code>	<code>rdfs:range</code>	<code>rdfs:Class .</code>
<code>rdfs:domain</code>	<code>rdfs:range</code>	<code>rdfs:Class .</code>
<code>rdfs:range</code>	<code>rdfs:range</code>	<code>rdfs:Class .</code>
<code>rdfs:subPropertyOf</code>	<code>rdfs:range</code>	<code>rdf:Property .</code>
<code>rdfs:subClassOf</code>	<code>rdfs:range</code>	<code>rdfs:Class .</code>
<code>rdf:subject</code>	<code>rdfs:range</code>	<code>rdfs:Resource .</code>
<code>rdf:predicate</code>	<code>rdfs:range</code>	<code>rdfs:Resource .</code>
<code>rdf:object</code>	<code>rdfs:range</code>	<code>rdfs:Resource .</code>
<code>rdfs:member</code>	<code>rdfs:range</code>	<code>rdfs:Resource .</code>
<code>rdf:first</code>	<code>rdfs:range</code>	<code>rdfs:Resource .</code>
<code>rdf:rest</code>	<code>rdfs:range</code>	<code>rdf:List .</code>
<code>rdfs:seeAlso</code>	<code>rdfs:range</code>	<code>rdfs:Resource .</code>
<code>rdfs:isDefinedBy</code>	<code>rdfs:range</code>	<code>rdfs:Resource .</code>
<code>rdfs:comment</code>	<code>rdfs:range</code>	<code>rdfs:Literal .</code>
<code>rdfs:label</code>	<code>rdfs:range</code>	<code>rdfs:Literal .</code>
<code>rdf:value</code>	<code>rdfs:range</code>	<code>rdfs:Resource .</code>

RDFS Axiomatic triples (3/3)

```

rdf:Alt                rdfs:subClassOf      rdfs:Container .
rdf:Bag                rdfs:subClassOf      rdfs:Container .
rdf:Seq               rdfs:subClassOf      rdfs:Container .
rdfs:ContainerMembershipProperty rdfs:subClassOf rdf:Property .

rdfs:isDefinedBy       rdfs:subPropertyOf  rdfs:seeAlso .

rdf:XMLLiteral         rdf:type            rdfs:Datatype .
rdf:XMLLiteral         rdfs:subClassOf     rdfs:Literal .
rdfs:Datatype          rdfs:subClassOf     rdfs:Class .

rdf:_1                rdf:type    rdfs:ContainerMembershipProperty .
rdf:_1                rdfs:domain  rdfs:Resource .
rdf:_1                rdfs:range   rdfs:Resource .
rdf:_2                rdf:type    rdfs:ContainerMembershipProperty .
rdf:_2                rdfs:domain  rdfs:Resource .
rdf:_2                rdfs:range   rdfs:Resource .
...
```

Role of Entailment rules (3/3)

- RDFS entailment satisfies the “RDFS entailment lemma”:
 - S rdfs-entails E if and only if there is a graph which can be derived from S plus the RDF and RDFS axiomatic triples by the application of rule lg, rule gl and the RDF and RDFS entailment rules and which either simply entails E or contains an XML clash

Effect of Entailment rules (1/3)

- The outputs of these rules will often trigger others.

	triggers rule:													
rule:	1	2	3	4a	4b	5a	5b	6	7a	7b	8	9	10	11
1	*	*	*				*	*				*		
2	*	*					*	*	*	*	*	*	*	*
3	*	*					*	*	*	*	*	*	*	*
4a	*	*					*				*			
4b	*	*					*				*			
5a						*	*							
5b	*	*					*							
6	*	*				*	*	*	*	*	*	*	*	*
7a	*	*					*				*	*		
7b	*	*					*							
8							*				*	*		
9		*					*	*	*	*	*	*	*	*
10	*	*			*	*	*							
11	*	*					*				*	*		

Table 1: Dependencies between RDFS entailment rules

Effect of Entailment rules (2/3)

- Rules propagate all `rdf:type` assertions in the graph up the subproperty and subclass hierarchies
- `rdf1` generates type assertions for all the property names used in the graph
 - `uuu aaa yyy → aaa rdf:type rdf:Property`
- `rdfs3` together with the last RDFS axiomatic triple adds all type assertions for all the class names used
 - `→ rdf:_2 rdfs:range rdfs:Resource`
 - `aaa rdfs:range xxx ∧ uuu aaa vvv → vvv rdf:type xxx`

Effect of Entailment rules (3/3)

- Any subproperty or subclass assertion generates type assertions for its subject and object via `rdfs2` and `rdfs3` and the domain and range assertions in the RDFS axiomatic triple set

- `aaa rdfs:domain xxx ∧ uuu aaa yyy → uuu rdf:type xxx`

- `aaa rdfs:range xxx ∧ uuu aaa vvv → vvv rdf:type xxx`

- For every `uuu` in V , the rules generate all assertions

- `uuu rdf:type rdfs:Resource`

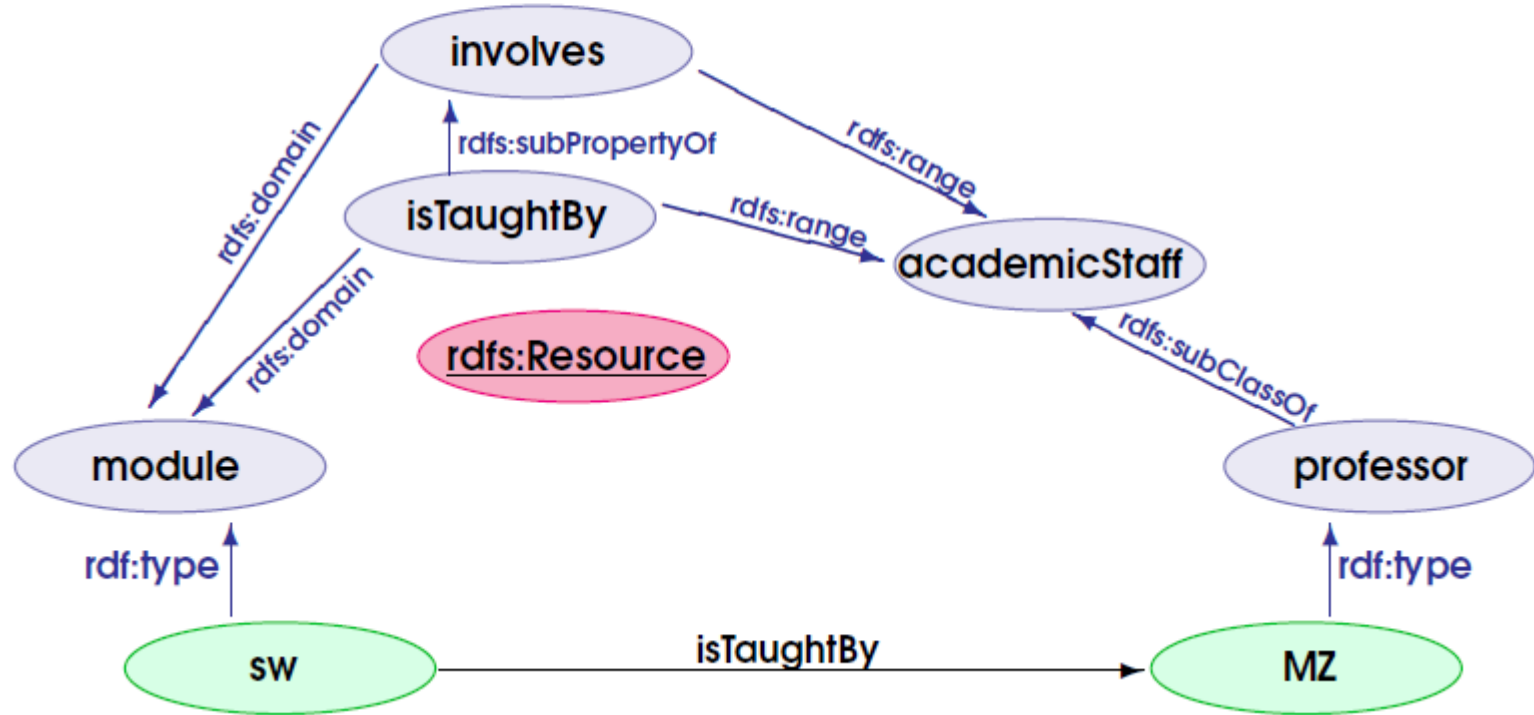
- For every class name `uuu`, the rules generate

- `uuu rdfs:subClassOf rdfs:Resource`

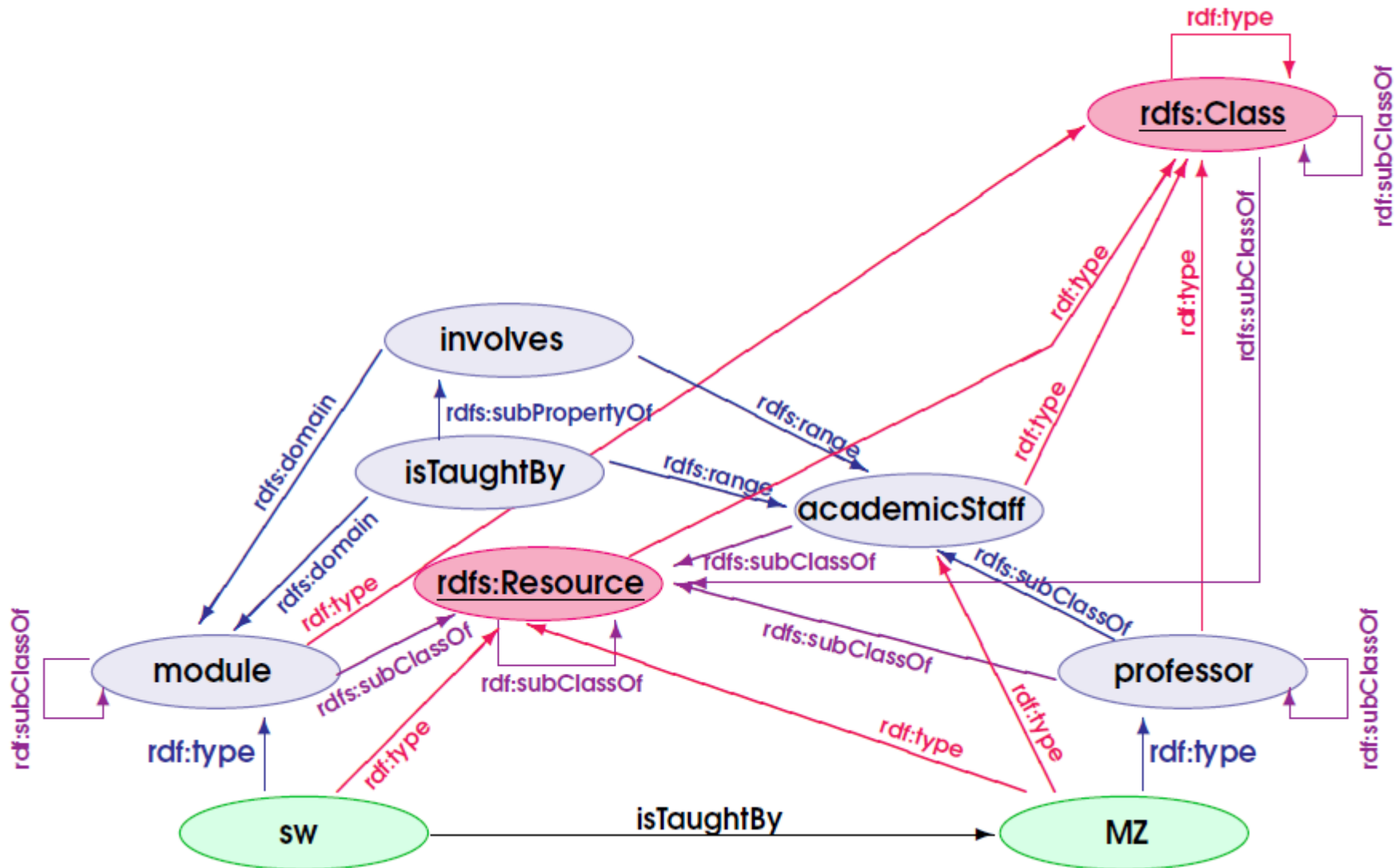
Example (axiomatic nodes)

rdf:Property

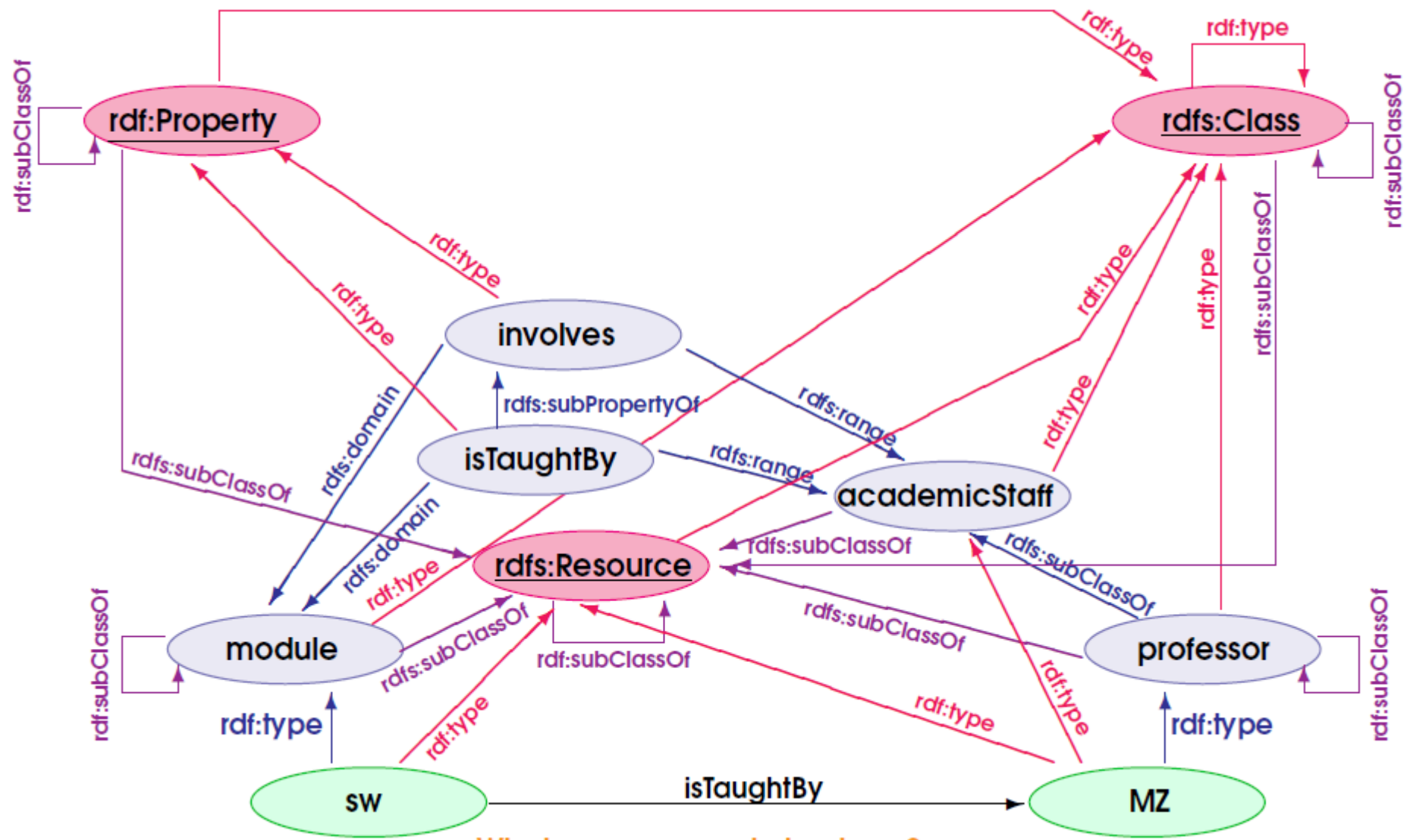
rdfs:Class



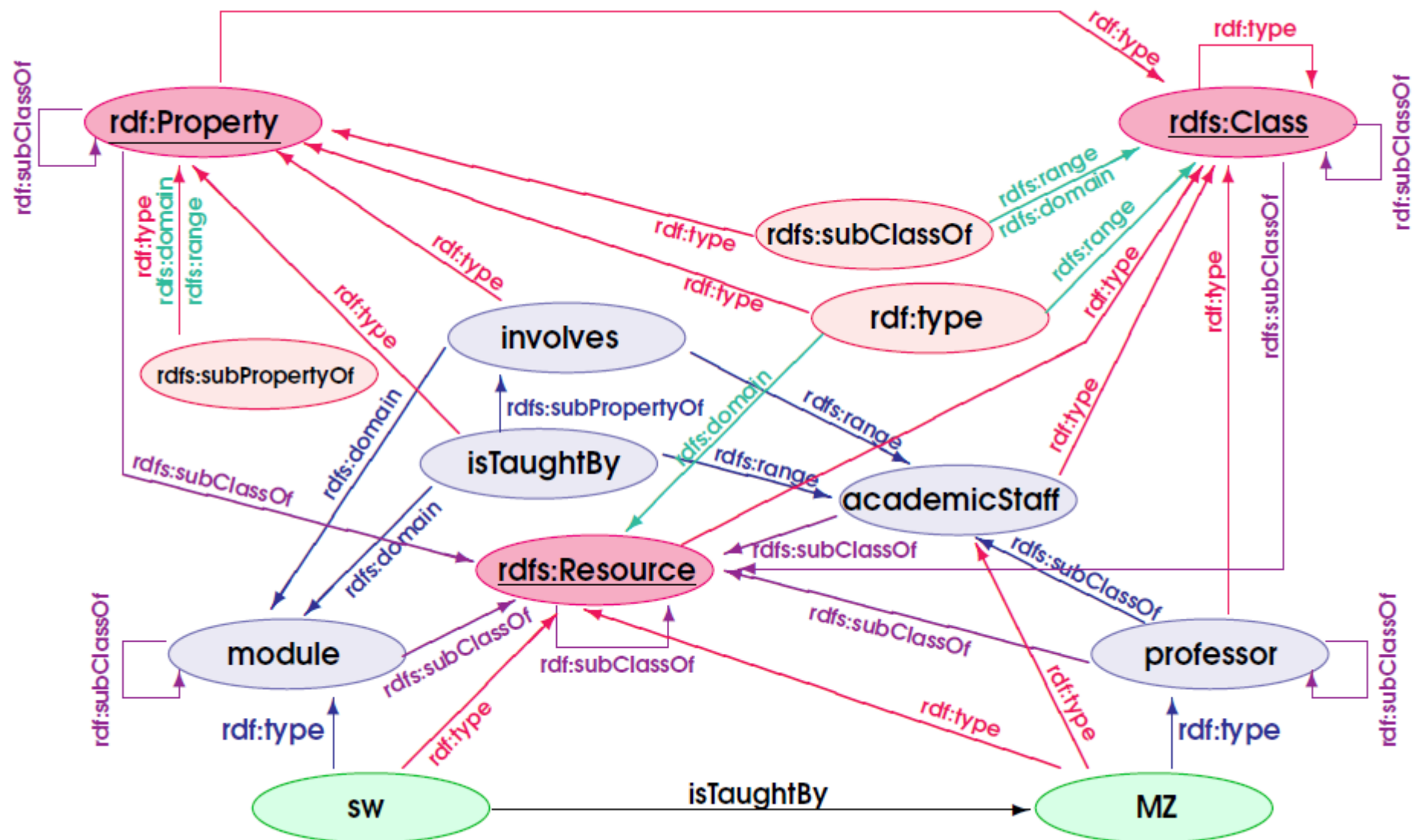
Example (rdf:type and rdfs:subClassOf)



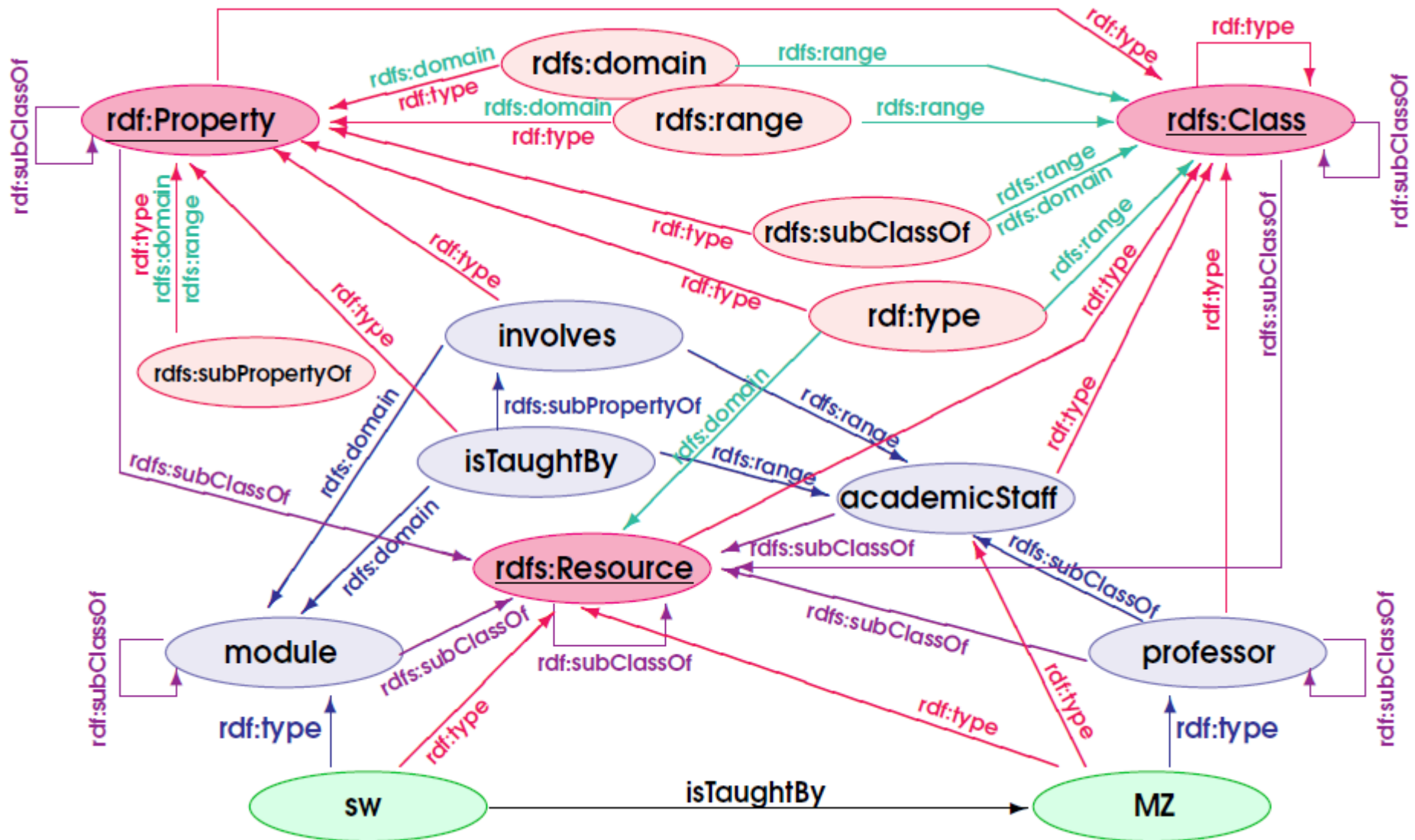
Example (properties, domains, ranges)



Example (reification of properties)



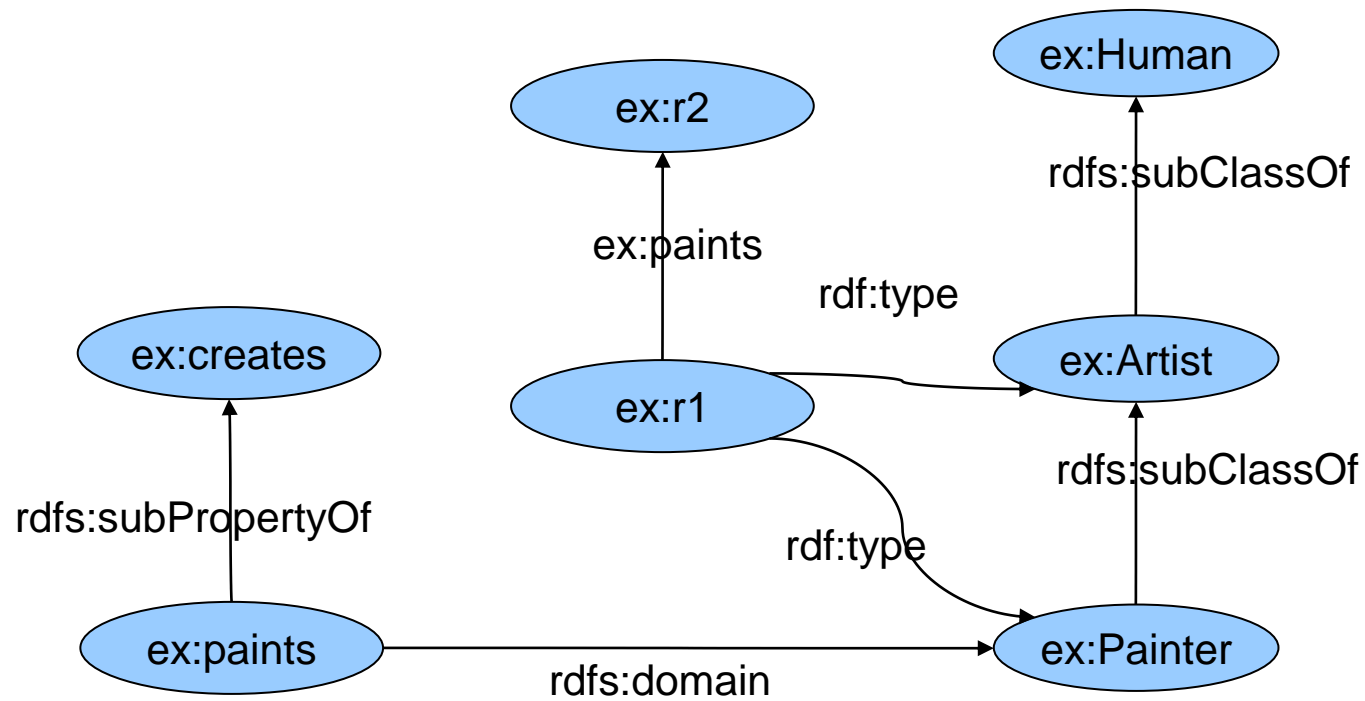
Example (reification of properties+axiomatic properties)



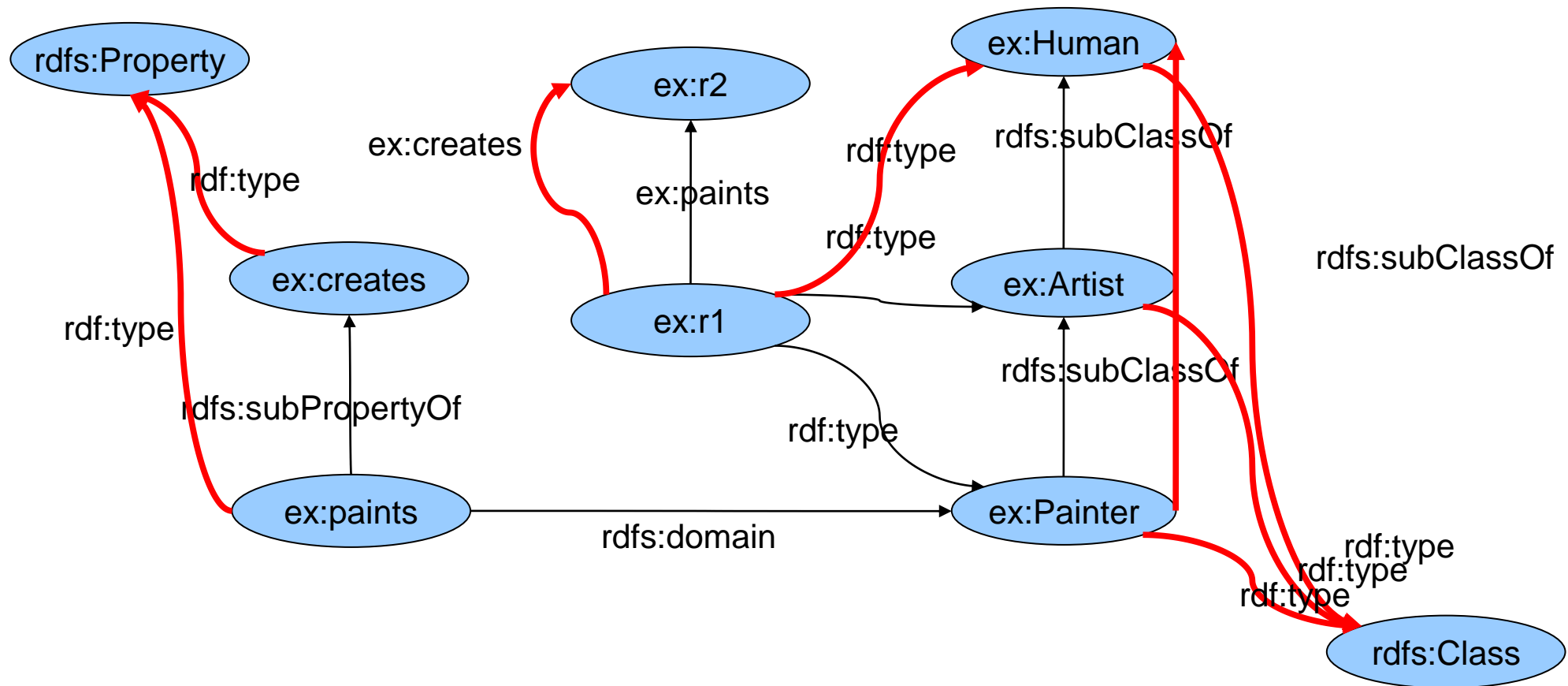
Closure and Reduction

- The **closure** of a graph is the graph defined by all triples that are **inferred** by the deduction system
 - Using the closure of a graph for storing minimizes query processing time
- The **reduction** of a graph is the **minimal** subset of triples needed to compute its closure
 - Using the reduction of a graph for storing minimizes the storage space needed.

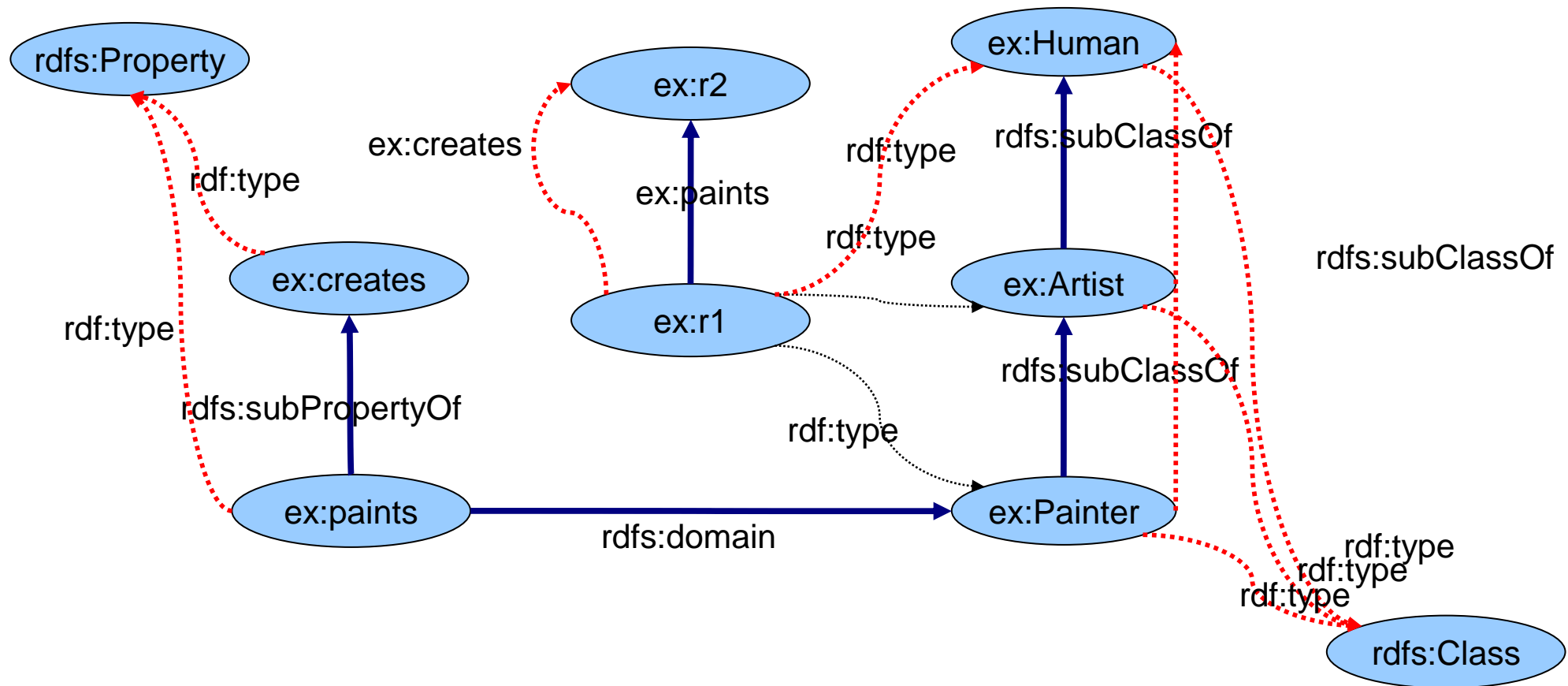
Example



Example: closure



Example: reduction

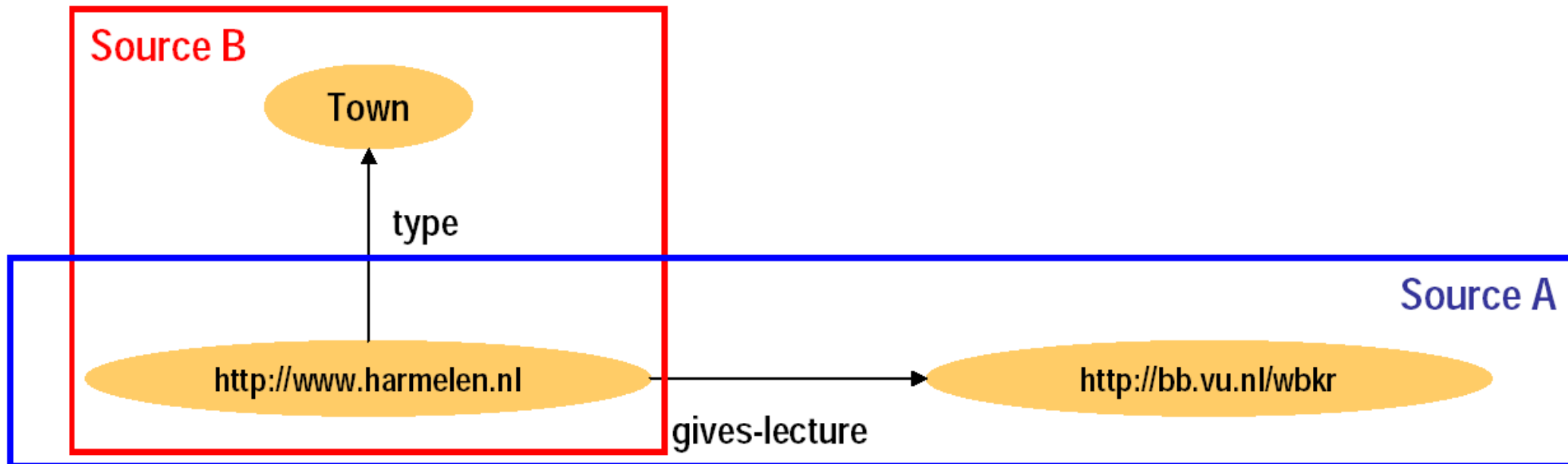


Discussion on RDF

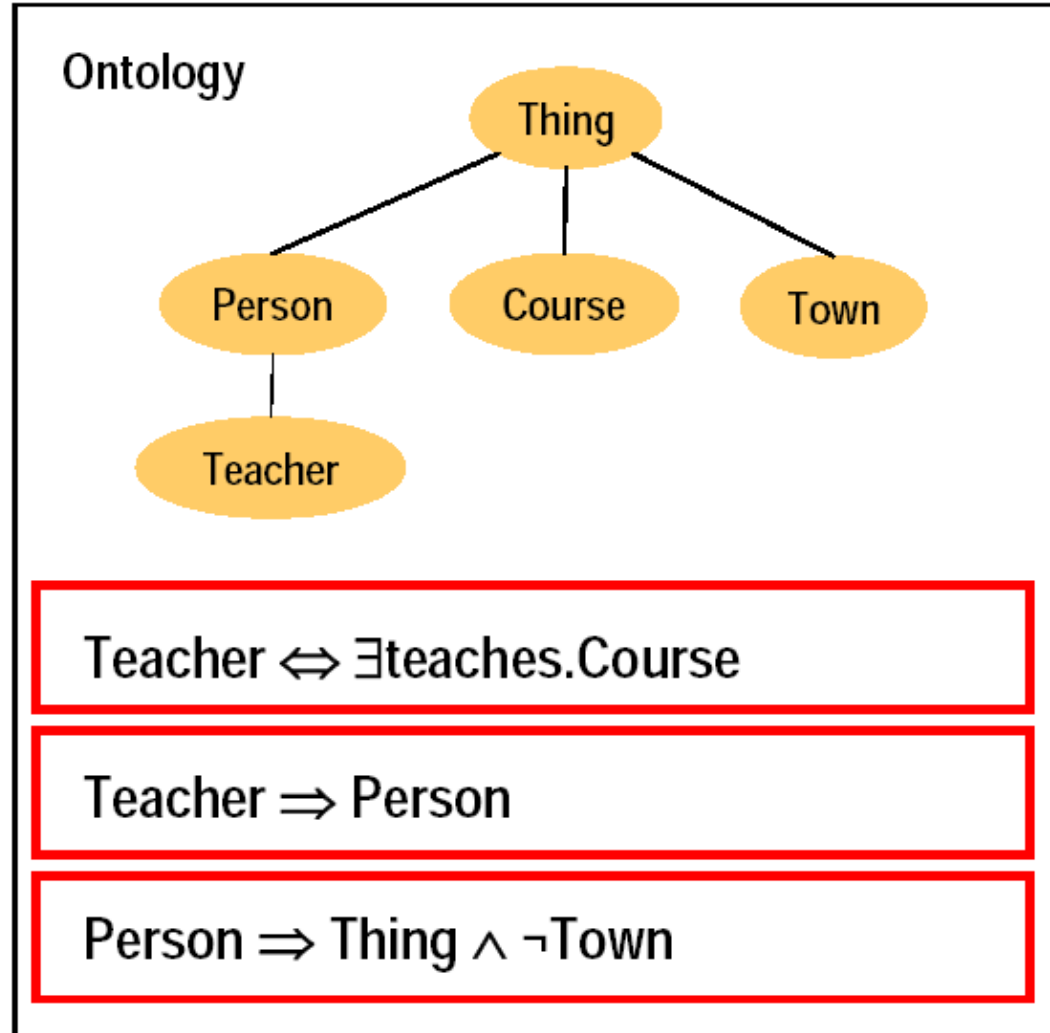
- RDF is a simple, but useful language to make relations between resources explicit
- It allows to define a simple form of schema which mostly consists of class and property hierarchies as well as relations with domain and range restriction
- There is a simple rule based reasoning mechanism for RDF schema that in most cases can be used offline because the increase is only factor 1.5 – this can further be reduced by only completing the schema offline
- Unlike most traditional KR languages RDF schema allows to freely combine modeling primitives which affects reasoning

Problems with RDF

- Not always meaningful representation

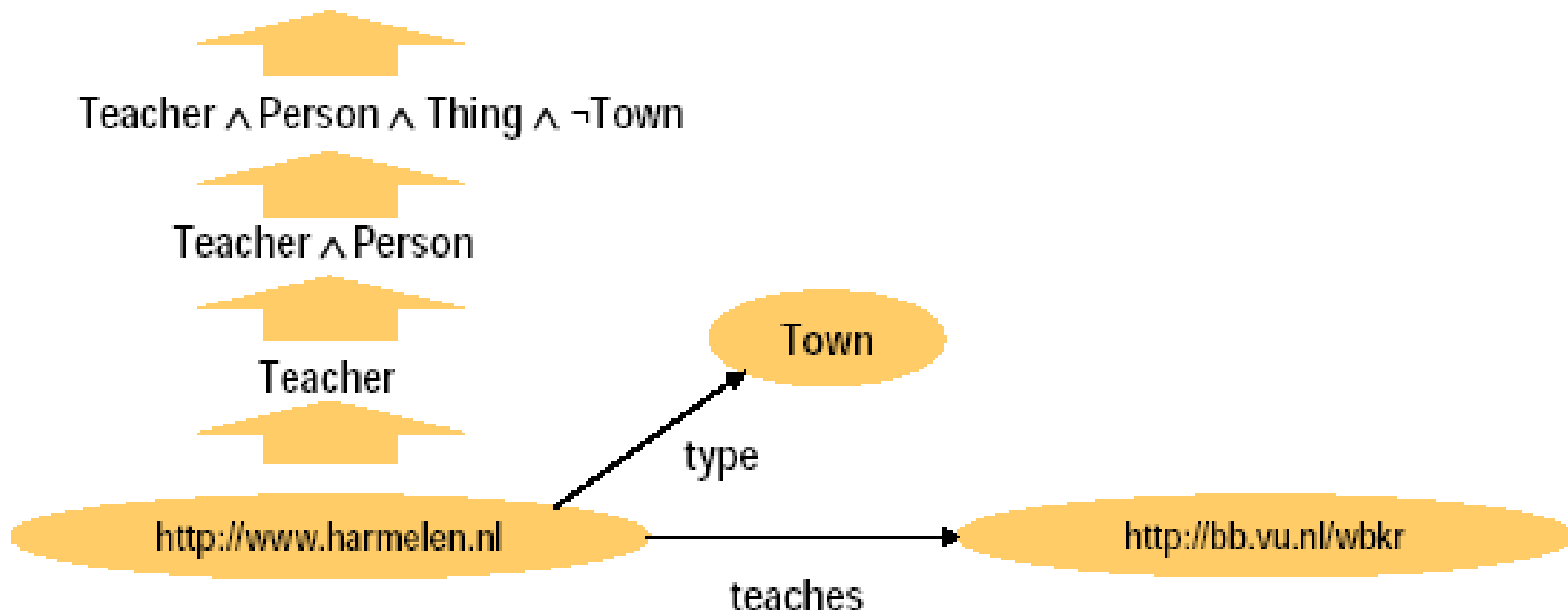


Need logical axioms!



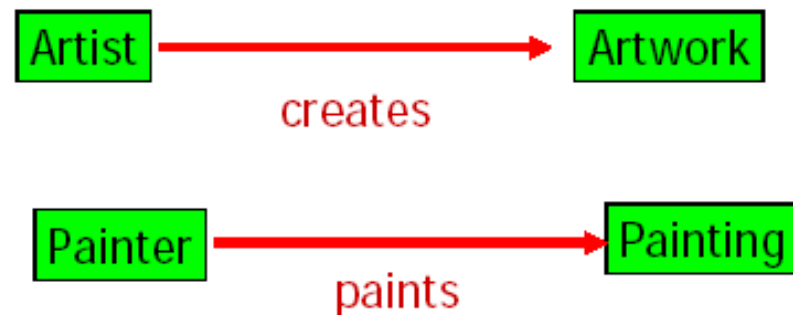
We can detect inconsistencies

$\text{Teacher} \wedge \text{Person} \wedge \text{Thing} \wedge \neg \text{Town} \wedge \text{Town}$



What properties do we need?

- Subclass relations
- Sub-properties



- Set operations

- Intersection (logical conjunction)
 - $\text{Painter} \cap \text{Writer} = \{x \mid x \subset \text{Painter} \text{ AND } x \subset \text{Writer}\}$
- Union (logical disjunction)
 - $\text{Painter} \cup \text{Writer} = \{x \mid x \subset \text{Painter} \text{ OR } x \subset \text{Writer}\}$
- Complement
 - `complementOf(III, Healthy)`
- Disjoint
 - `disjointWith(Lung, Liver, Kidney)`

What properties do we need? /2

- Cardinality restriction



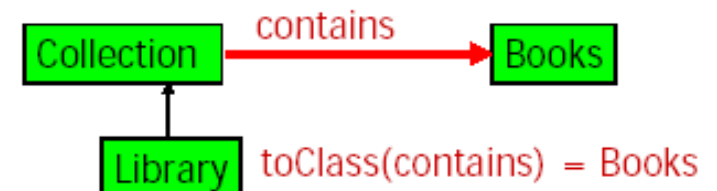
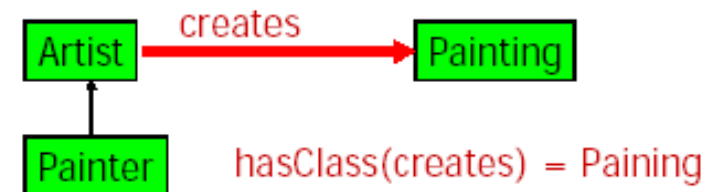
- Cardinality(has_location) = 1 :



- Min_Cardinality(creates) = 1 :

- Existential quantifier

- Universal quantifier

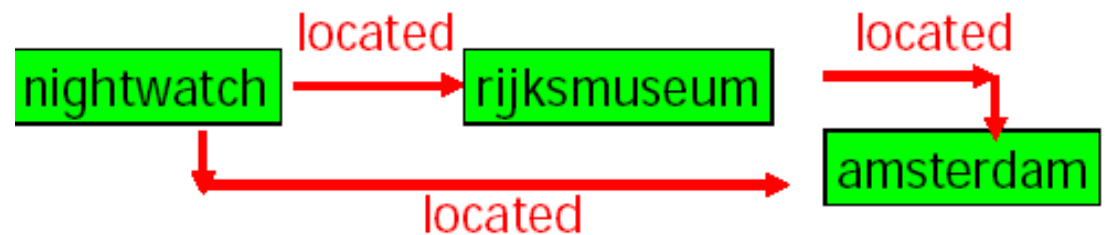


What properties do we need? /3

- Inverse properties
- Symmetric properties
- Transitivity



borders(Liver,Gall) -> borders(Gall,Liver)



References

- RDF Semantics - W3C Recommendation 10 February 2004
 - <http://www.w3.org/TR/rdf-mt/>
- Course material for “Practical Reasoning for the Semantic Web” course at the 17th European Summer School in Logic, Language and Information (ESSLLI)
 - <http://www.few.vu.nl/~schlobac/>

References

- <http://www.aaai.org/AITopics/html/reason.html>
- <http://media.cwi.nl/survey/>
- <http://www.mkbergman.com>
- <http://www.dcs.bbk.ac.uk/~michael/sw/sw.html>
- <http://www.inf.unibz.it/~debruijn/teaching/swt/>

License



- This work is licensed under the Creative Commons **Attribution-Noncommercial-Share Alike 3.0** Unported License.
- To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.