
University of
Southampton

University of Southampton

SPARQL Protocol and RDF Query Language

COMP6256 Knowledge Graphs for AI Systems

Dr Nicholas Gibbins – nmg@ecs.soton.ac.uk

University of Southampton

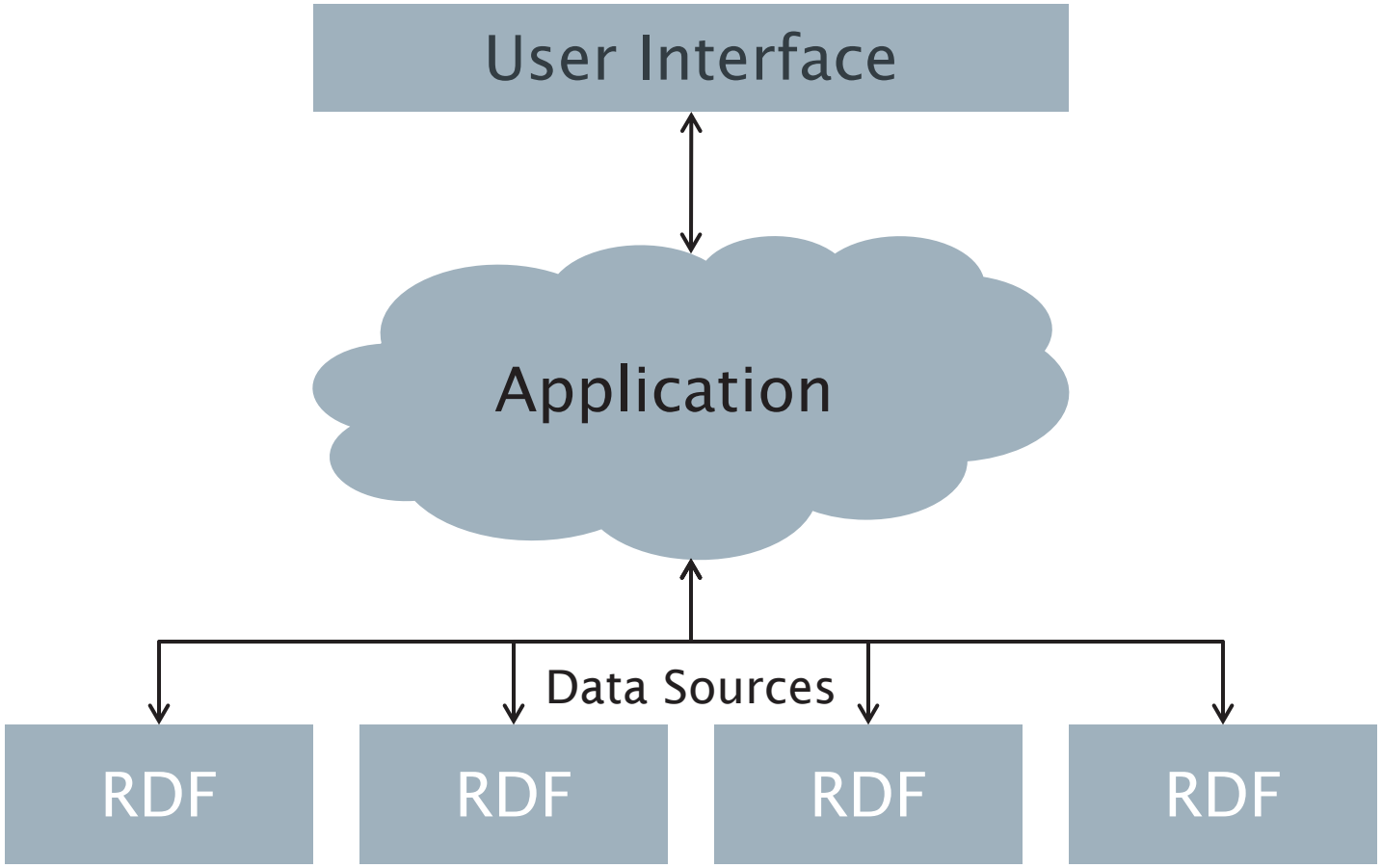
Semantic Web Applications

Technologies considered so far allow us to represent data (RDF)

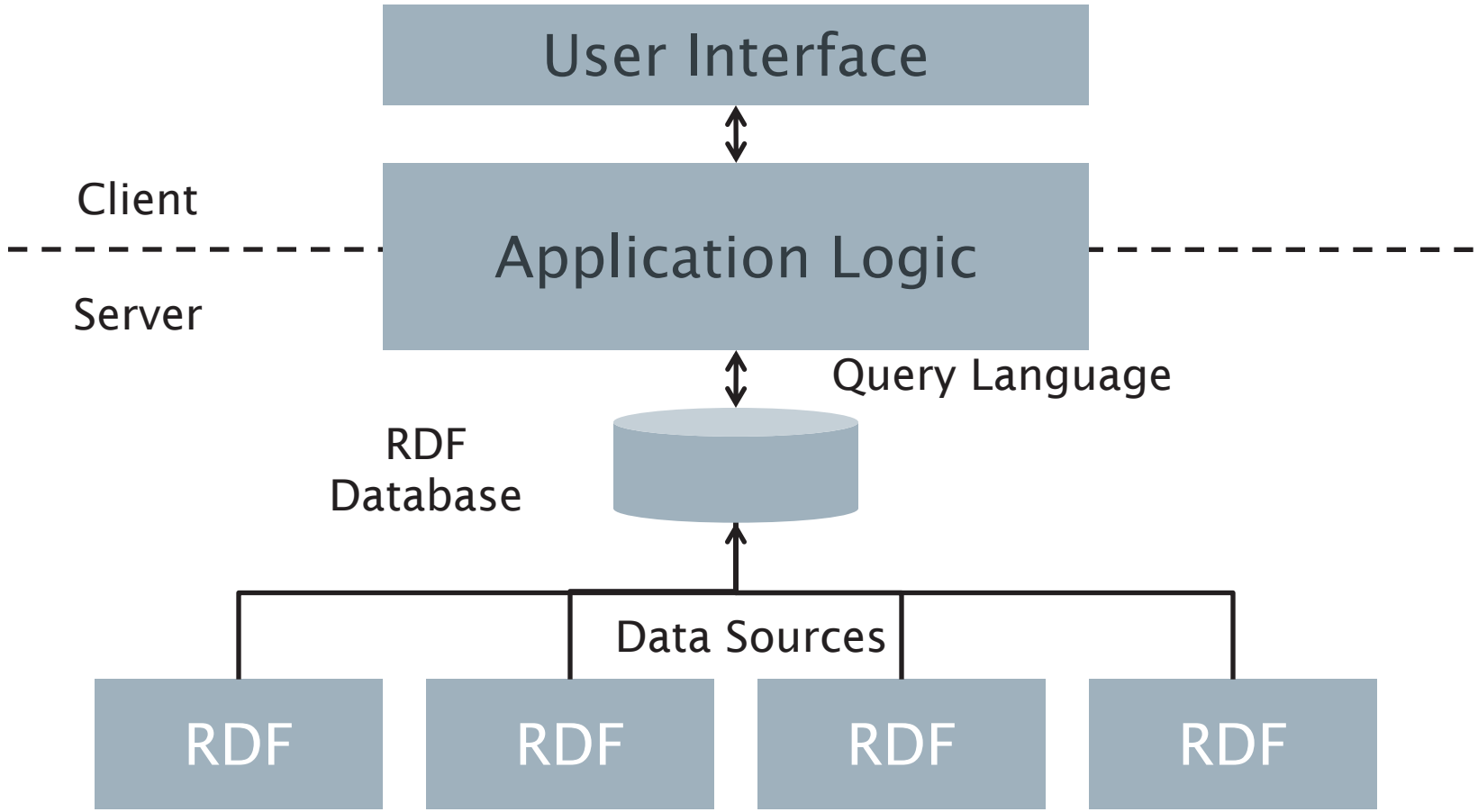
We can put data in – how do we get it back out?

University of Southampton

The Semantic Web Application



The Semantic Web Application



RDF Triplestores

Specialised databases for storing RDF data

Can be viewed as a database containing a single table

- Table contains subject/predicate/object as minimum
- Many triplestores store quads to maintain provenance
- May store other ancillary information

Subject	Predicate	Object	Graph
URI	URI	URI/Literal	URI
…	…	…	…

University of Southampton

SPARQL Query Language

University of Southampton

SPARQL in Brief

SQL-like syntax:

```
SELECT <variables>
FROM <graphs>
WHERE { <triple patterns> }
```

- Variables prefixed with '?':
- Triple patterns expressed in N3-like syntax:

  ```
  ?who foaf:knows <http://example.org/person/fred> .
  ```

- Queries return a result set of bindings for the variables

University of Southampton

Example Query

```
_:a foaf:name   "Johnny Lee Outlaw" .
_:a foaf:mbox   <mailto:jlow@example.com> .
_:b foaf:name   "Peter Goodguy" .
_:b foaf:mbox   <mailto:peter@example.org> .
_:c foaf:mbox   <mailto:carol@example.org> .
```

```
SELECT ?name ?mbox
WHERE {
      ?x foaf:name ?name .
      ?x foaf:mbox ?mbox .
}
```

University of Southampton

Example Query

```
_:a foaf:name   "Johnny Lee Outlaw" .
_:a foaf:mbox   <mailto:jlow@example.com> .

_:b foaf:name   "Peter Goodguy" .
_:b foaf:mbox   <mailto:peter@example.org> .
```

?name	?mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

Namespaces in Queries

The query given in the previous example was over-simplified

- Need to define namespaces for vocabulary terms used:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE {
      ?x foaf:name ?name .
      ?x foaf:mbox ?mbox .
}
```

Matching RDF Literals

```
@prefix ns:  <http://example.org/ns#> .
@prefix :    <http://example.org/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:x  ns:p  "cat"@en .
:y  ns:p  "42"^^xsd:integer .
```

Matching RDF String Literals

```
@prefix ns:  <http://example.org/ns#> .
@prefix :    <http://example.org/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:x  ns:p  "cat"@en .
:y  ns:p  "42"^^xsd:integer .
```

The following query returns no bindings:

```
SELECT ?v
WHERE {
      ?v ?p "cat" .
}
```

Matching RDF String Literals

```
@prefix ns:  <http://example.org/ns#> .
@prefix :    <http://example.org/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:x  ns:p  "cat"@en .
:y  ns:p  "42"^^xsd:integer .
```

“cat” must have language tag, as in graph: “cat”@en

The following query returns a single binding:

```
SELECT ?v
WHERE {
      ?v ?p "cat"@en .
}
```

Matching RDF Numeric Literals

```
@prefix ns:  <http://example.org/ns#> .
@prefix :    <http://example.org/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:x  ns:p  "cat"@en .
:y  ns:p  "42"^^xsd:integer .
```

The following query returns a single binding:

```
SELECT ?v
WHERE {
      ?v ?p 42 .
}
```

Integers without quotes in SPARQL queries are implicitly typed as xsd:integer

Blank Nodes

Blank nodes in a result set are scoped to that result set

- No guarantee that blank node labels will be unchanged over repeated queries

```
_:a  foaf:name   "Alice" .
_:b  foaf:name   "Bob" .
```

```
SELECT ?x ?name
WHERE  {
      ?x foaf:name ?name .
}
```

University of Southampton

Blank Nodes

?x	?name
_:a	“Alice”
_:b	“Bob”

?x	?name
_:x	“Alice”
_:y	“Bob”

SPARQL Constraints

Constraints can be applied to variables:

```
SELECT ?title
WHERE {
  ?x dc:title ?title .
  FILTER regex(?title, "^SPARQL")
}
```

```
SELECT ?title ?price
WHERE {
  ?x ns:price ?price .
  FILTER (?price < 30.5)
  ?x dc:title ?title .
}
```

University of Southampton

Group Graph Patterns

Set of patterns enclosed in { }

- Determines scoping for FILTER operators (and other operators)

Optional Graph Patterns

OPTIONAL allows us to state that a group graph pattern need not be matched in order for a binding to be produced

```
SELECT ?name ?mbox
WHERE {
  ?x foaf:name  ?name .
  OPTIONAL {
    ?x  foaf:mbox  ?mbox .
  }
}
```

Optional Graph Patterns

```
_:a  rdf:type  foaf:Person .
_:a  foaf:name  "Alice" .
_:a  foaf:mbox  <mailto:alice@example.com> .
_:a  foaf:mbox  <mailto:alice@work.example.com> .
_:b  rdf:type  foaf:Person .
_:b  foaf:name  "Bob" .
```

```
SELECT ?name ?mbox
WHERE {
  ?x foaf:name  ?name .
  OPTIONAL {
    ?x  foaf:mbox  ?mbox .
  }
}
```

Optional Graph Patterns

?name	?mbox
“Alice”	<mailto:alice@example.com>
“Alice”	<mailto:alice@work.example.com>
“Bob”	

Union Graph Patterns

UNION effectively gives us a disjunctive query: either match this or match that

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE {
  { ?book dc10:title  ?title . }
  UNION
  { ?book dc11:title  ?title . }
}
```

University of Southampton

Specifying Datasets

RDF data may be published as multiple graphs (serialised in multiple documents)

Sometimes we wish to be able to restrict a query to certain sources (graphs or datasets)

Default Graph

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT  ?name
FROM <http://example.org/foaf/aliceFoaf>
WHERE {
  ?x foaf:name ?name .
}
```

`http://example.org/foaf/aliceFoaf` is the default graph

Unless otherwise specified, all triples are taken from that graph

Named Graphs

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?who ?g ?mbox
FROM <http://example.org/dft.ttl>
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
WHERE {
  ?g dc:publisher ?who .
  GRAPH ?g { ?who foaf:mbox ?mbox . }
}
```

Ordering

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ont: <http://example.org/ontology/>
SELECT ?name
WHERE {
  ?x foaf:name ?name ;
     ont:empId ?emp .
}
ORDER BY ?name DESC(?emp)
```

Order results by ?name in ascending order, and then by ?emp in descending order

ASC() sorts in ascending order (this is the default)

LIMIT and OFFSET

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
  ?x foaf:name ?name .
}
ORDER BY ?name
LIMIT   5
OFFSET  10
```

Returns five results, after skipping the first ten results

DISTINCT

DISTINCT removes duplicate results

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name
WHERE {
  ?x foaf:name ?name .
}
```

Other SPARQL Verbs

SELECT is not the only verb

- `CONSTRUCT`
- `ASK`
- `DESCRIBE`

University of Southampton

CONSTRUCT

A CONSTRUCT query returns an RDF graph, specified by a graph template

- For each row in the result set, substitute the variables in the template with the values in that row
- Combine the resulting graphs into a single graph (set union of triples)

CONSTRUCT

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT   {
  ?x vcard:FN ?name .
  ?x vcard:EMAIL ?mail .
}
WHERE {
  ?x foaf:name ?name .
  ?x foaf:mbox ?mail .
}
```

University of Southampton

CONSTRUCT and bNodes

CONSTRUCT can create graphs with bNodes

bNode labels are scoped to the template for each solution

- If the same label occurs twice in a template, then there will be one blank node created for each query solution
- There will be different blank nodes for triples generated by different query solutions

CONSTRUCT and bNodes

```
CONSTRUCT {
  ?x  vcard:N _:v .
  _:v vcard:givenName ?gname .
  _:v vcard:familyName ?fname .
}
WHERE {
  { ?x foaf:firstname ?gname } UNION { ?x foaf:givenname   ?gname } .

  { ?x foaf:surname   ?fname } UNION { ?x foaf:family_name ?fname } .
}
```

CONSTRUCT and bNodes

```
_:a foaf:givenname "Alice" .
_:a foaf:family_name "Hacker" .
_:b foaf:firstname "Bob" .
_:b foaf:surname  "Hacker” .


_:v1 vcard:N _:x .
_:x vcard:givenName "Alice" .
_:x vcard:familyName "Hacker" .
_:v2 vcard:N _:z .
_:z vcard:givenName "Bob" .
_:z vcard:familyName "Hacker" .
```

CONSTRUCT is not a rule language

CONSTRUCT matches graph patterns and returns a new graph

We could implement a simple rule-based system using CONSTRUCT queries to represent rules

University of Southampton

CONSTRUCT is not a rule language

From the Data Access Working Group Charter:

> "While it is hoped that the product of the RDF Data Access Working Group will be useful in later development of a rules language, development of such a rules language is out of scope for this working group. However, any serializations of a query language must not preclude extension to, or inclusion in, a rules language. The group should expend minimal effort assuring that such an extension be intuitive and and consistent with any query language produced by the group."

University of Southampton

Accessing graphs with CONSTRUCT

```
CONSTRUCT {
  ?s ?p ?o .
}
WHERE {
  GRAPH <http://example.org/aGraph> { ?s ?p ?o } .
}
```

Accessing graphs with CONSTRUCT

```
CONSTRUCT {
      ?s ?p ?o .
}
WHERE {
      GRAPH ?g { ?s ?p ?o } .
      ?g dc:publisher <http://www.w3.org/> .
      ?g dc:date ?date .
      FILTER ( ?date > "2005-02-28T00:00:00Z"^^xsd:dateTime ) .
}
```

ASK

An ASK query is used to check whether a query pattern has a solution.

- No information is returned about query solutions (no variable bindings), just a boolean value

```
_:a  foaf:name  "Alice" .
_:a  foaf:homepage  <http://work.example.org/alice/> .
_:b  foaf:name  "Bob" .
_:b  foaf:mbox  <mailto:bob@work.example> .
```

```
ASK { ?x foaf:name  "Alice" }
```

(returns true)

University of Southampton

DESCRIBE

Returns a single graph containing information about a resource or resources

Nature of description left unspecified

- Blank node closure
- Concise Bounded Description

Structure of returned data is determined by the SPARQL query processor

DESCRIBE Examples

```
DESCRIBE <http://example.org/>


DESCRIBE ?x
WHERE {
      ?x foaf:name "Alice" .
}


DESCRIBE ?x ?y <http://example.org/>
WHERE {
      ?x foaf:knows ?y .
}
```

University of Southampton

Concise Bounded Description

Algorithm for extracting a subgraph from an RDF graph, given a resource of interest

- Produces a graph where the object nodes are either URI references, literals, or blank nodes not serving as the subject of any statement in the graph
- Asymmetric view of a resource – follows only outgoing links

1. Include in the subgraph all statements in the source graph where the subject of the statement is the starting node;
2. Recursively, for all statements identified in the subgraph thus far having a blank node object, include in the subgraph all statements in the source graph where the subject of the statement is the blank node in question and which are not already included in the subgraph.

Concise Bounded Description Example

```
_:a  foaf:name        "Alice" .
_:a  foaf:homepage    <http://work.example.org/alice/> .
_:b  foaf:name        "Bob" .
_:b  foaf:mbox        <mailto:bob@example.org> .
<http://example.org/> dc:creator _:a .
<http://example.org/> dc:title “Example Inc. Website” .
<http://example.org/> dc:date “2005-05-23” .
```

Concise Bounded Description Example

```
DESCRIBE <http://example.org/>


_:a  foaf:name       "Alice" .
_:a  foaf:homepage   <http://work.example.org/alice/> .
_:b  foaf:name       "Bob" .
_:b  foaf:mbox       <mailto:bob@example.org> .
<http://example.org/> dc:creator _:a .
<http://example.org/> dc:title “Example Inc. Website” .
<http://example.org/> dc:date “2005-05-23” .
```

University of Southampton

SPARQL Protocol

University of Southampton

SPARQL Protocol

SPARQL Query Language is protocol-independent

- SPARQL Protocol defines the method of interaction with a SPARQL processor

Two standard bindings:

- HTTP
- SOAP

Standard XML results format

University of Southampton

SPARQL over HTTP

Two methods of submitting a query to a processor:

HTTP GET

- Query encoded as HTTP request URI
- Limitation on query length

HTTP POST

- Query encoded in HTTP request body
- No limitation on query length

HTTP GET

```
GET /sparql/?query=EncodedQuery HTTP/1.1
Host: www.example.org
User-agent: my-sparql-client/0.1
```

SPARQL Results Format

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="book"/>
    <variable name="who"/>
  </head>
  <results distinct="false" ordered="false”>
    <result>
      <binding name="book">
        <uri>http://www.example/book/book5</uri>
      </binding>
      <binding name="who">
        <bnode>r29392923r2922</bnode>
      </binding>
    </result>
    …
  </results>
</sparql>
```

SPARQL Results Format

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="book"/>
    <variable name="who"/>
  </head>
  <results distinct="false" ordered="false”>
    <result>
      <binding name="book">
        <uri>http://www.example/book/book5</uri>
      </binding>
      <binding name="who">
        <bnode>r29392923r2922</bnode>
      </binding>
    </result>
    …
  </results>
</sparql>
```

variables

SPARQL Results Format

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="book"/>
    <variable name="who"/>
  </head>
  <results distinct="false" ordered="false”>
    <result>
      <binding name="book">
        <uri>http://www.example/book/book5</uri>
      </binding>
      <binding name="who">
        <bnode>r29392923r2922</bnode>
      </binding>
    </result>
    …
  </results>
</sparql>
```

results section

SPARQL Results Format

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="book"/>
    <variable name="who"/>
  </head>
  <results distinct="false" ordered="false”>
    <result>
      <binding name="book">
        <uri>http://www.example/book/book5</uri>
      </binding>
      <binding name="who">
        <bnode>r29392923r2922</bnode>
      </binding>
    </result>
    …
  </results>
</sparql>
```

a single result (row)

SPARQL Results Format

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="book"/>
    <variable name="who"/>
  </head>
  <results distinct="false" ordered="false”>
    <result>
      <binding name="book">
        <uri>http://www.example/book/book5</uri>
      </binding>
      <binding name="who">
        <bnode>r29392923r2922</bnode>
      </binding>
    </result>
    …
  </results>
</sparql>
```

binding of the variable “book”

SPARQL Results Format

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="book"/>
    <variable name="who"/>
  </head>
  <results distinct="false" ordered="false”>
    <result>
      <binding name="book">
        <uri>http://www.example/book/book5</uri>
      </binding>
      <binding name="who">
        <bnode>r29392923r2922</bnode>
      </binding>
    </result>
    …
  </results>
</sparql>
```

binding of the variable “who”

SPARQL Results Format

CONSTRUCT and DESCRIBE queries return RDF data

- application/rdf+xml, text/turtle or similar

ASK queries return a boolean value:

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head></head>
  <boolean>true</boolean>
</sparql>
```

University of Southampton

Further Reading

SPARQL Query Language for RDF
http://www.w3.org/TR/rdf-sparql-query/

SPARQL Protocol for RDF
http://www.w3.org/TR/rdf-sparql-protocol/

SPARQL Query XML Results Format
http://www.w3.org/TR/rdf-sparql-XMLres/

University of Southampton

SPARQL 1.1

University of Southampton

Extending SPARQL

Initial version of SPARQL became a W3C Recommendation in 2008

Other features requested by community:

- Update
- Aggregates
- Subqueries
- SELECT expressions
- Property paths
- Federation
- Entailment

University of Southampton

SPARQL 1.1

W3C SPARQL Working Group started in 2009

SPARQL 1.1 became a W3C Recommendation in March 2013

University of Southampton

Update

SPARQL only provides a means for querying a database

Other capabilities required by many applications:

- Create
- (Read)
- Update
- Delete

SPARQL 1.1 Update derived from earlier SPARUL proposal

Update: INSERT DATA

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
INSERT DATA
{
  <http://example/book1> dc:title "A new book" ;
                         dc:creator "A.N.Other" .
}
```

Update: DELETE DATA

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
DELETE DATA
{
  <http://example/book2> dc:title "David Copperfield" ;
                         dc:creator "Edmund Wells" .
}
```

Update: DELETE/INSERT

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
WITH <http://example/addresses>
DELETE
{
  ?person foaf:givenName 'Bill'
}
INSERT
{
  ?person foaf:givenName 'William'
}
WHERE
{
  ?person a foaf:Person .
  ?person foaf:givenName 'Bill'
}
```

University of Southampton

Update: Graph Operations

- LOAD: load triples from URI into graph
- CLEAR: clear all triples from graph
- CREATE: create new graph in store
- DROP: remove graph from store
- COPY: copy all triples from one graph to another
- MOVE: move all triples from one graph to another (remove from source)
- ADD: add all triples in one graph to another

SELECT Expressions

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title (?p*(1-?discount) AS ?price)
WHERE {
  ?x ns:price ?p .
  ?x dc:title ?title .
  ?x ns:discount ?discount
}
```

Aggregates

```
PREFIX : <http://books.example/>
SELECT (SUM(?lprice) AS ?totalPrice)
WHERE {
      ?org :affiliates ?auth .
      ?auth :writesBook ?book .
      ?book :price ?lprice .
}
GROUP BY ?org
HAVING (SUM(?lprice) > 10)
```

Other aggregate functions:

- COUNT, MIN, MAX, GROUP_CONCAT, SAMPLE

Subqueries

```
PREFIX : <http://people.example/>
PREFIX : <http://people.example/>
SELECT ?y ?minName
WHERE {
   :alice :knows ?y .
   {
     SELECT ?y (MIN(?name) AS ?minName)
     WHERE {
       ?y :name ?name .
     }
     GROUP BY ?y
   }
}
```

Property Paths: Alternatives

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?displayString
WHERE {
  :book1 dc:title|rdfs:label ?displayString
}
```

“|” as sugared syntax for UNION

Property Paths: Sequences

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
  ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows/foaf:name ?name .
}
```

Sugared syntax for `?x foaf:knows [ foaf:name ?name ]`

Property Paths: Arbitrary Sequences

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
  ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows+/foaf:name ?name .
}
```

Repeat operators

- * (zero or more occurrences)
- + (one or more occurrences)
- {n} (exactly n occurrences)
- {n,m} (between n and m occurrences)

Property Paths: Inverses

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?y
WHERE {
      <mailto:alice@example> ^foaf:mbox ?x .
      ?x foaf:knows/^foaf:knows ?y .
      FILTER(?x != ?y)
}
```

University of Southampton

Entailment Regimes

So far, we've only looked at queries on graphs without inference

- Inference changes matters – our patterns can match inferred triples
- Which triples are inferred depends on the *entailment regime*

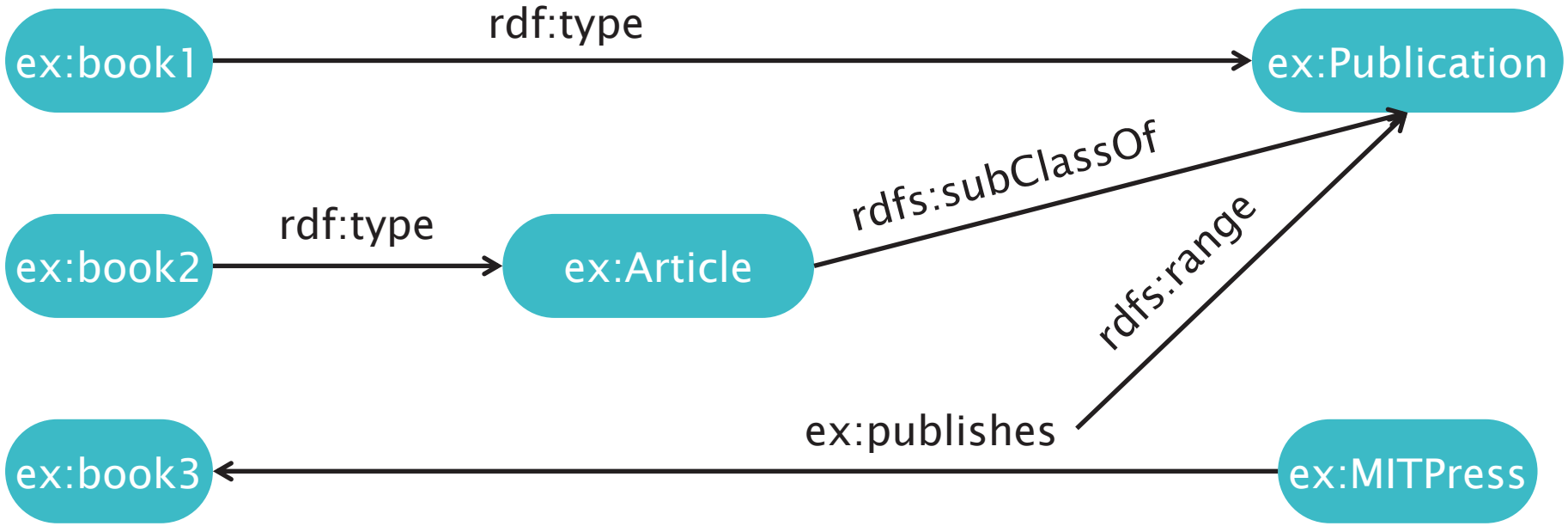
Several entailment regimes, corresponding to:

- The ontology languages used: RDF, RDFS, OWL
- The use of datatypes
- The use of rule languages

https://www.w3.org/TR/sparql11-entailment/

RDF Entailment

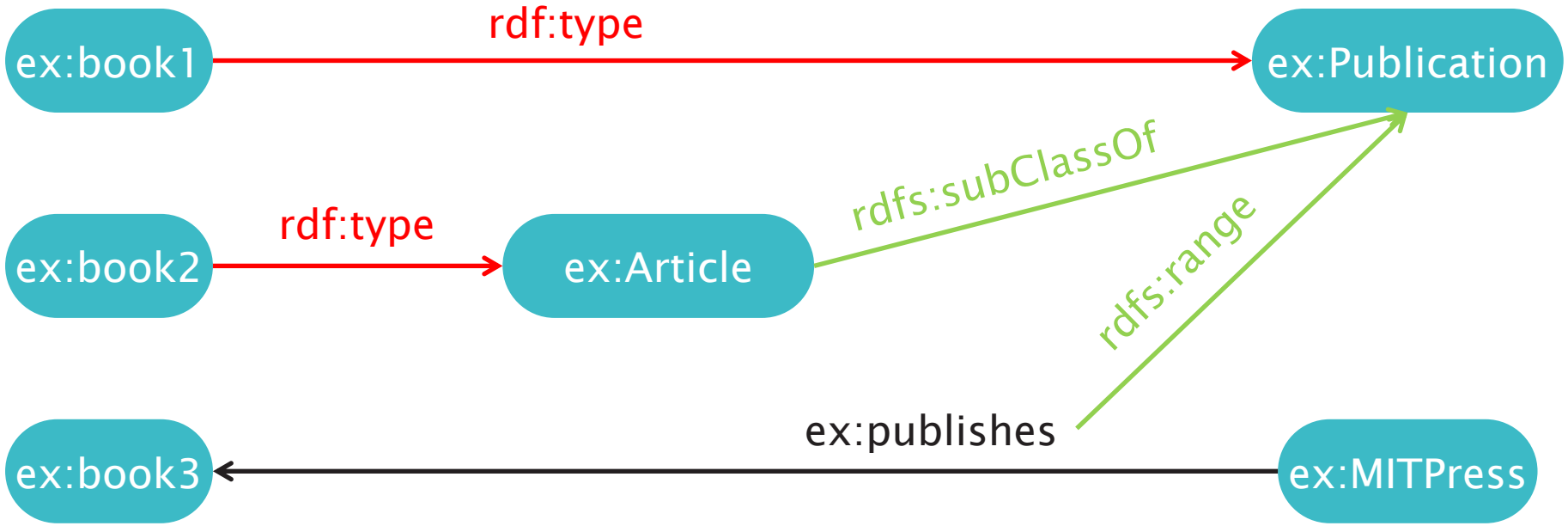
Consider the following graph:



RDF Entailment

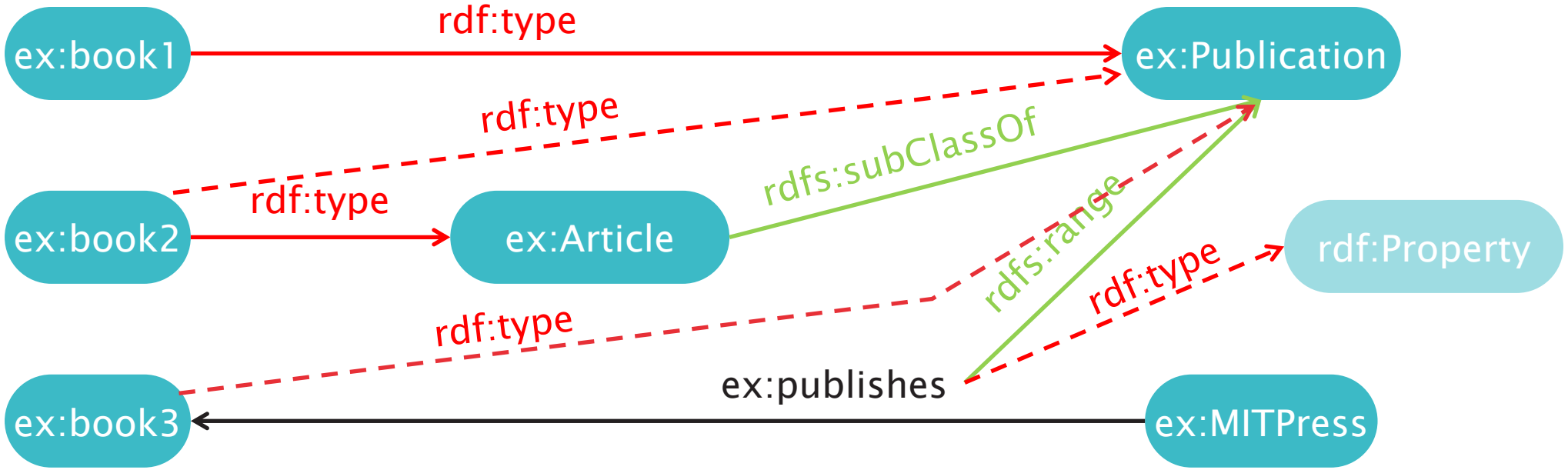
RDF special terms

RDFS special terms



RDF Entailment

The use of special terms means that certain other triples are entailed.

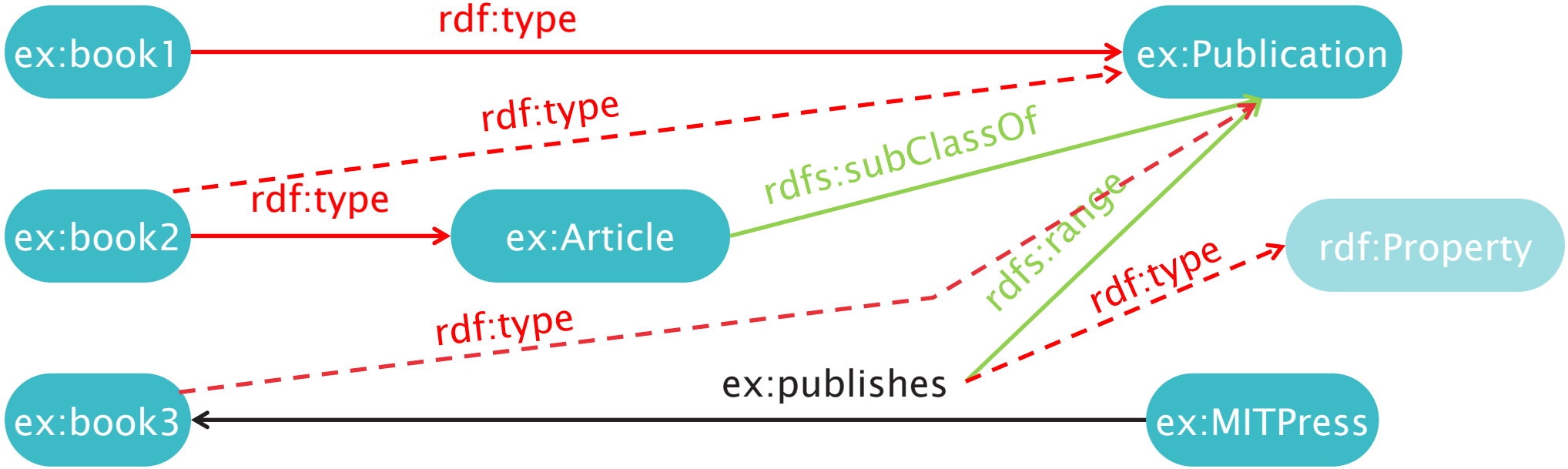


RDF Entailment

```
SELECT ?pub WHERE { ?pub rdf:type ex:Publication }
```

Without entailment: ex:book1

With entailment: ex:book1, ex:book2, ex:book3

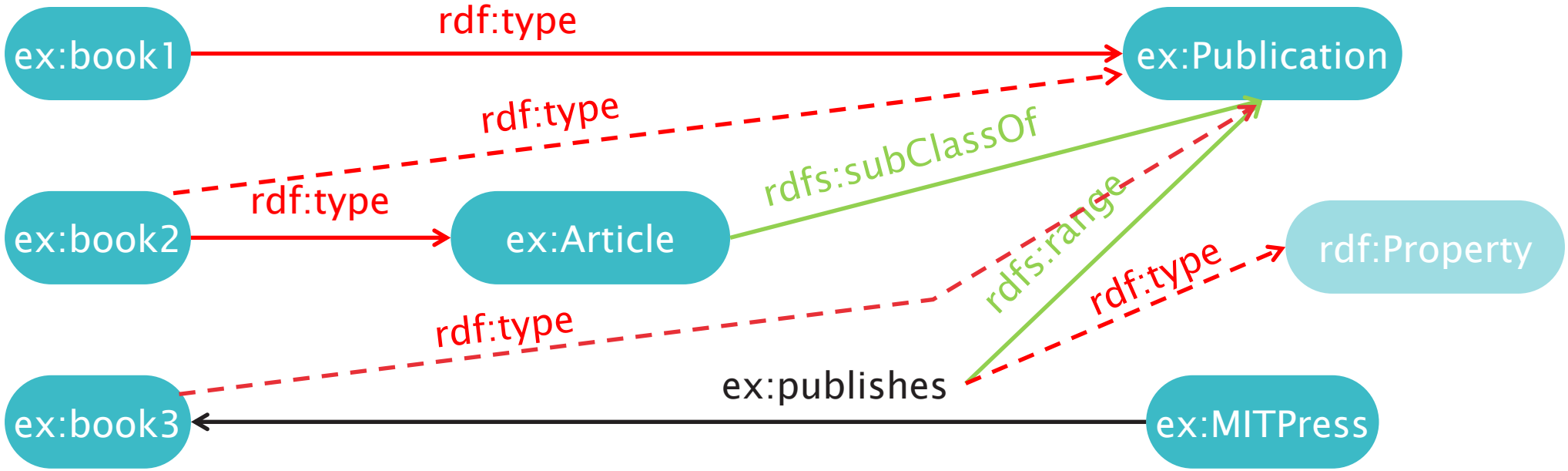


RDF Entailment

```
SELECT ?prop WHERE { ?prop rdf:type rdf:Property }
```

Without entailment: empty answer

With entailment: ex:publishes, what else?



University of Southampton

Further Reading

SPARQL 1.1 Query Language
http://www.w3.org/TR/sparql11-query/

SPARQL 1.1 Update
http://www.w3.org/TR/sparql11-update/

SPARQL Query XML Results Format
http://www.w3.org/TR/rdf-sparql-XMLres/

SPARQL 1.1 Entailment Regimes
https://www.w3.org/TR/sparql11-entailment/

University of Southampton

Next Lecture: Ontologies