RDF: building block for the Semantic Web

how do we code meaning/knowledge?

RDF data model: a summary so far

- RDF offers an abstract model and framework that tells us how to decompose information/knowledge into small pieces;
- one such small piece of information/knowledge is represented as a statement which has the form (subject, predicate, object). A statement is also called a triple;
- a given RDF model can be expressed either as a graph, or as a collection of statements or triples;
- each statement maps to one edge in the graph. Therefore, the subject and object of a given statement are also called nodes, and its predicate is also called edge;
- subjects and objects denote resources in the real world. Predicates denote the relationship between subjects and objects;

RDF data model: a summary so far

- predicates are also called properties, and objects are also called property values. Therefore, a statement also has the form (**resource, property, propertyValue**);
- URIs are used to name resources and properties. For a given resource or property, if there is an existing URI to name it, you should re-use it instead of inventing your own;
- an RDF statement can only model a binary relationship. To model a n-ary relationship, intermediate resources are introduced, and blank nodes are quite often used;
- an object can take either a simple literal or another resource as its value. If a literal is used as its value, the literal can be typed or un-typed, and can also have an optional language tag.

RDF Serialization: RDF/XML Syntax

- the RDF data model is only an abstract data model, used to express our idea and view
- we need some serialization syntax for creating and reading concrete RDF models, so applications can start to write and share RDF documents
- the W3C specifications define an XML syntax for this purpose. It is called **RDF/XML**, and is used to represent an RDF graph as an XML document
- RDF/XML is not the only serialization syntax that is being used, e.g., n3

RDF Serialization: RDF Vocabulary

- in the world of RDF, we uses URIs (instead of words) to name resources and properties
- in general, RDF refers to *a set of URIs* (often created for a specific purpose) as a **vocabulary**
  - ✓ all the URIs in such a vocabulary normally share a common leading string, which is used as the common prefix in these URIs' QNames
  - ✓ the URIs in this vocabulary will be formed by appending individual local names to the end of this common leading string this prefix (**namespace prefix)**

RDF Serialization: RDF Vocabulary

- to define RDF/XML serialization syntax, a set of URIs are created and are given specific meanings by RDF
- this group of URIs becomes RDF's own vocabulary of terms, and it is called the **RDF Vocabulary**
- the URIs in this RDF Vocabulary all share the following lead strings:

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

remember

by convention:

- this URI prefix string is often associated with namespace prefix `rdf:`
- for this reason, this vocabulary is also referred to as the rdf: vocabulary

RDF Serialization: RDF Vocabulary

often used *terms* in **rdf: vocabulary** are listed here:

Syntax names:
`rdf:RDF`, `rdf:Description`, `rdf:ID`, `rdf:about`,
`rdf:parseType`, `rdf:resource`, `rdf:li`, `rdf:nodeID`,
`rdf:datatype`

Class names:
`rdf:Seq`, `rdf:Bag`, `rdf:Alt`, `rdf:Statement`, `rdf:Property`,
`rdf:XMLLiteral`, `rdf:List`

Property names:
`rdf:subject`, `rdf:predicate`, `rdf:object`, `rdf:type`,
`rdf:value`, `rdf:first`, `rdf:rest_`*`n`*

Resource names:
`rdf:nil`

so, `rdf:`*`name`* will be used to indicate a term from the RDF vocabulary

RDF Serialization: RDF/XML Syntax

subject	predicate	object
myCamera:Nikon_D300	myCamera:is_a	myCamera:DSLR

using the terms from **rdf vocabulary**, the above statement can be expressed in RDF/XML as follows:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
         xmlns:myCamera="http://www.example.com/camera#">

  <rdf:Description rdf:about="http://www.example.com/camera#Nikon_D300">
     <myCamera:is_a rdf:resource="http://www.example.com/camera#DSLR"/>
  </rdf:Description>

</rdf:RDF>
```

RDF Serialization: RDF/XML Syntax

the core is the following statement:

```
<rdf:Description rdf:about="http://www.example.com/camera#Nikon_D300">
   <myCamera:is_a rdf:resource="http://www.example.com/camera#DSLR"/>
</rdf:Description>
```

it reads as this: *This is a description about a resource named* *myCamera:Nikon_D300*, *which is an instance of another resource, namely,* *myCamera:DSLR*.

here is how the statement is structured:

```
<rdf:Description rdf:about="URI of the statement's subject">
  <predicateURI rdf:resource="URI of the statement's object"/>
</rdf:Description>
```

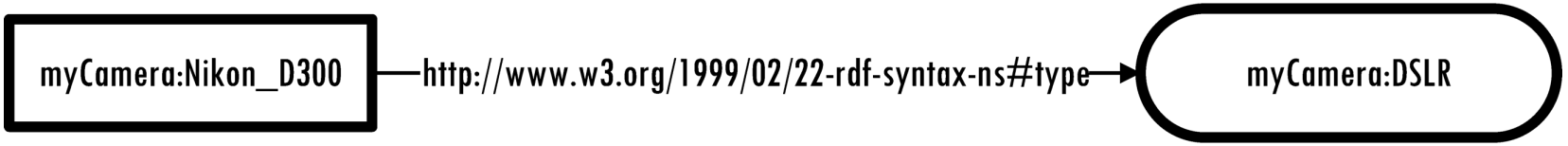
this is also the so-called "long form" RDF statement.

RDF Serialization: RDF/XML Syntax

```
<rdf:Description rdf:about="http://www.example.com/camera#Nikon_D300">
   <myCamera:is_a rdf:resource="http://www.example.com/camera#DSLR"/>
</rdf:Description>
```

- `rdf:Description` and `rdf:about` are all terms from rdf-vocabulary
- `myCamera:is_a` is a term that we invented; it is used to identify the type of a given resource
- rdf vocabulary provides a term, `rdf:type`, just for this purpose:

```
<rdf:Description rdf:about="http://www.example.com/camera#Nikon_D300">
   <rdf:type rdf:resource="http://www.example.com/camera#DSLR"/>
</rdf:Description>
```



- the subject node here is often called a *typed* node in a graph, or typed node element in RDF documents
- assigning a type to a resource has far-reaching implication we will see later

RDF Serialization: RDF/XML Syntax

- you don't have to use **`rdf:type`** much:

```
<myCamera:DSLR rdf:about=http://www.liyangyu.com/camera#Nikon_D300/>
```

- this is the "short-form", and is the same as the previous statement
- "short-form" is more often used, since it is simpler

RDF Serialization: RDF/XML Syntax

similarly, we can add more statements:

```
1: <?xml version="1.0"?>
2: <rdf:RDF
2a:     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3:      xmlns:myCamera="http://www.example.com/camera#">
4:
5:   <rdf:Description
5a:        rdf:about="http://www.example.com/camera#Nikon_D300">
6:     <rdf:type
6a:          rdf:resource="http://www.example.com/camera#DSLR"/>
7:     <myCamera:manufactured_by
7a:        rdf:resource="http://www.dbpedia.org/resource/Nikon"/>
8:     <myCamera:performance rdf:resource=
8a:              "http://www.example.com/camera#PictureQuality"/>
9:   </rdf:Description>
10:
11: <rdf:Description
11a:  rdf:about="http://www.example.com/camera#PictureQuality">
12:   <myCamera:evaluate>5 stars</myCamera:evaluate>
13: </rdf:Description>
14:
15: </rdf:RDF>
```

RDF Serialization: RDF/XML Syntax

- quite long and quite ugly
- you could use **`rdf:ID`** and **`xml:base`** to make RDF/XML simpler

```
<?xml version="1.0"?>
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:myCamera="http://www.liyangyu.com/camera#">

   <rdf:Description rdf:ID="Nikon_D300">
     <rdf:type
         rdf:resource="http://www.liyangyu.com/camera#DSLR"/>
     <myCamera:manufactured_by
       rdf:resource="http://www.dbpedia.org/resource/Nikon"/>
   </rdf:Description>

</rdf:RDF>
```

- **`rdf:ID`** only specifies a fragment identifier; the complete URI of the subject is obtained by concatenating the following 3 pieces together: in-scope base URI + "#" + **`rdf:ID`** value

RDF Serialization: RDF/XML Syntax

- in-scope base URI is not explicitly stated in the RDF document, it is often provided by the RDF parser based on the location of the file
- clearly, the URI changes if the location of the RDF document changes

solution: explicitly state the in-scope base URI by using **`xml:base`** attribute, an RDF parser generates the full URI by using the following mechanism:

`xml:base` + "#" + **`rdf:ID`** value

```
<?xml version="1.0"?>
 <rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:myCamera="http://www.liyangyu.com/camera#"
      xml:base="http://www.liyangyu.com/camera">

   <myCamera:DSLR rdf:ID="Nikon_D300">
     <myCamera:manufactured_by
        rdf:resource="http://www.dbpedia.org/resource/Nikon"/>
   </myCamera:DSLR>
  ......
```

Re-thinking RDF: what is missing?

- RDF data model provides a simple and elegant way to present facts – with well-defined structure that machine can understand
- RDF triples are created in a distributed fashion – you can say anything about anything
- RDF data model allows distributed information to be related in a meaningful way – use URI to represent resource/predicate
- if you are talking about Washington as a state (*not George Washington, or Washington DC, or …*), then use the URI that represents Washington as a state – semantic disambiguation

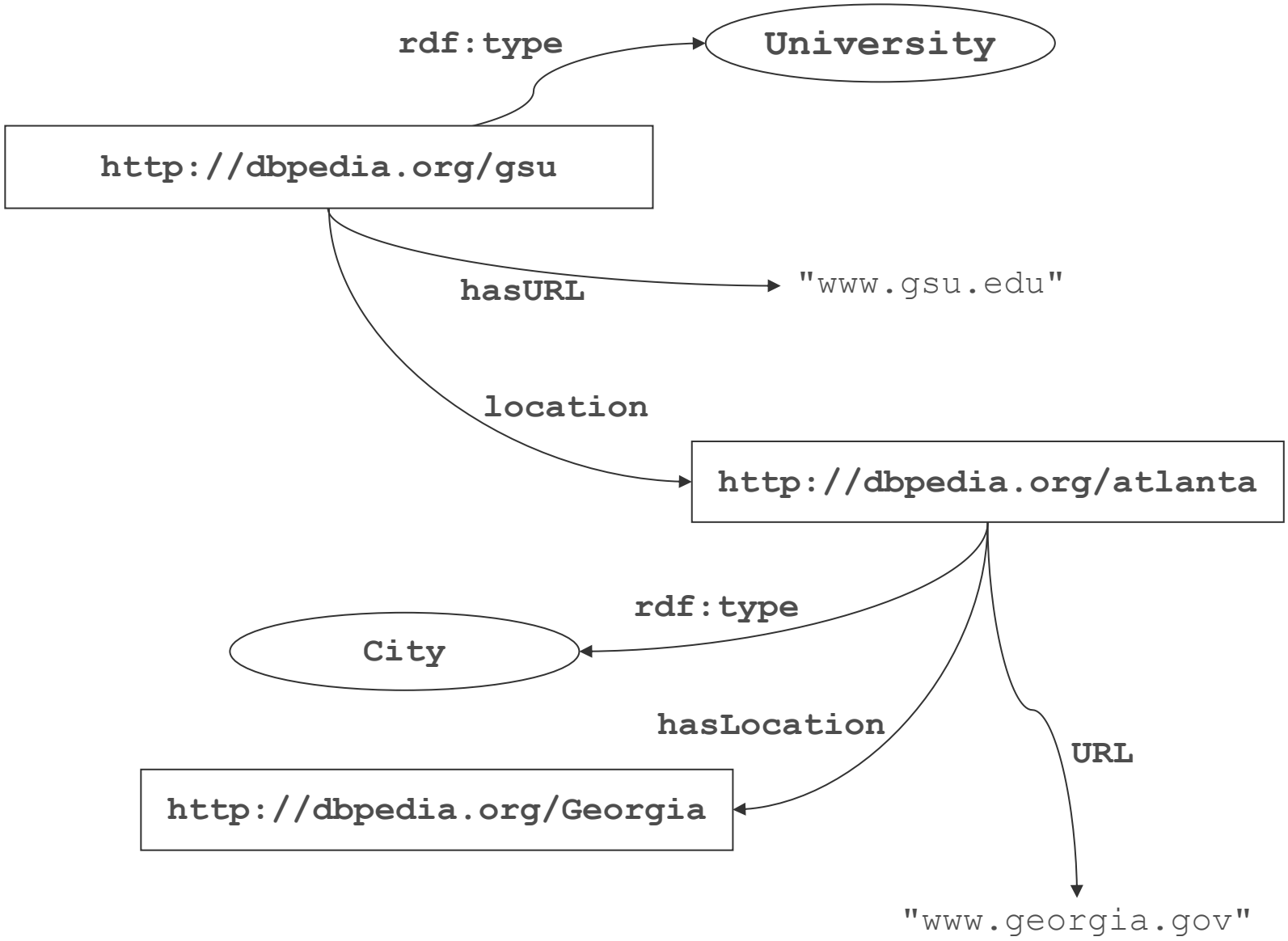
these are the good things about RDF ...
but do you see anything is missing?

Re-thinking RDF: what is missing?

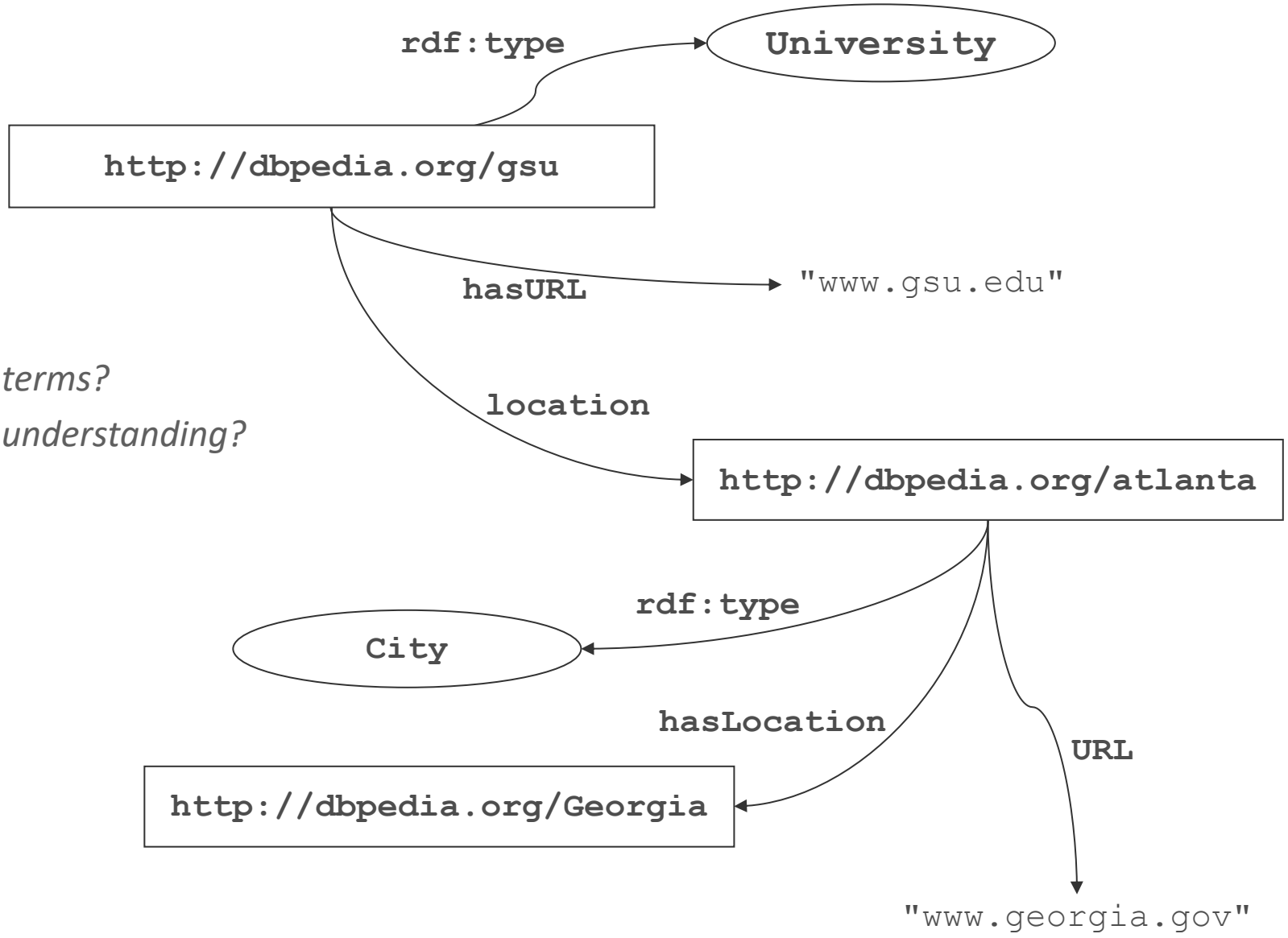
it might be easier to understand this by using one example

- let us use RDF statements to describe GSU, and the city GSU locates in
- use RDF graph only since RDF/XML is too ugly

Re-thinking RDF: what is missing?



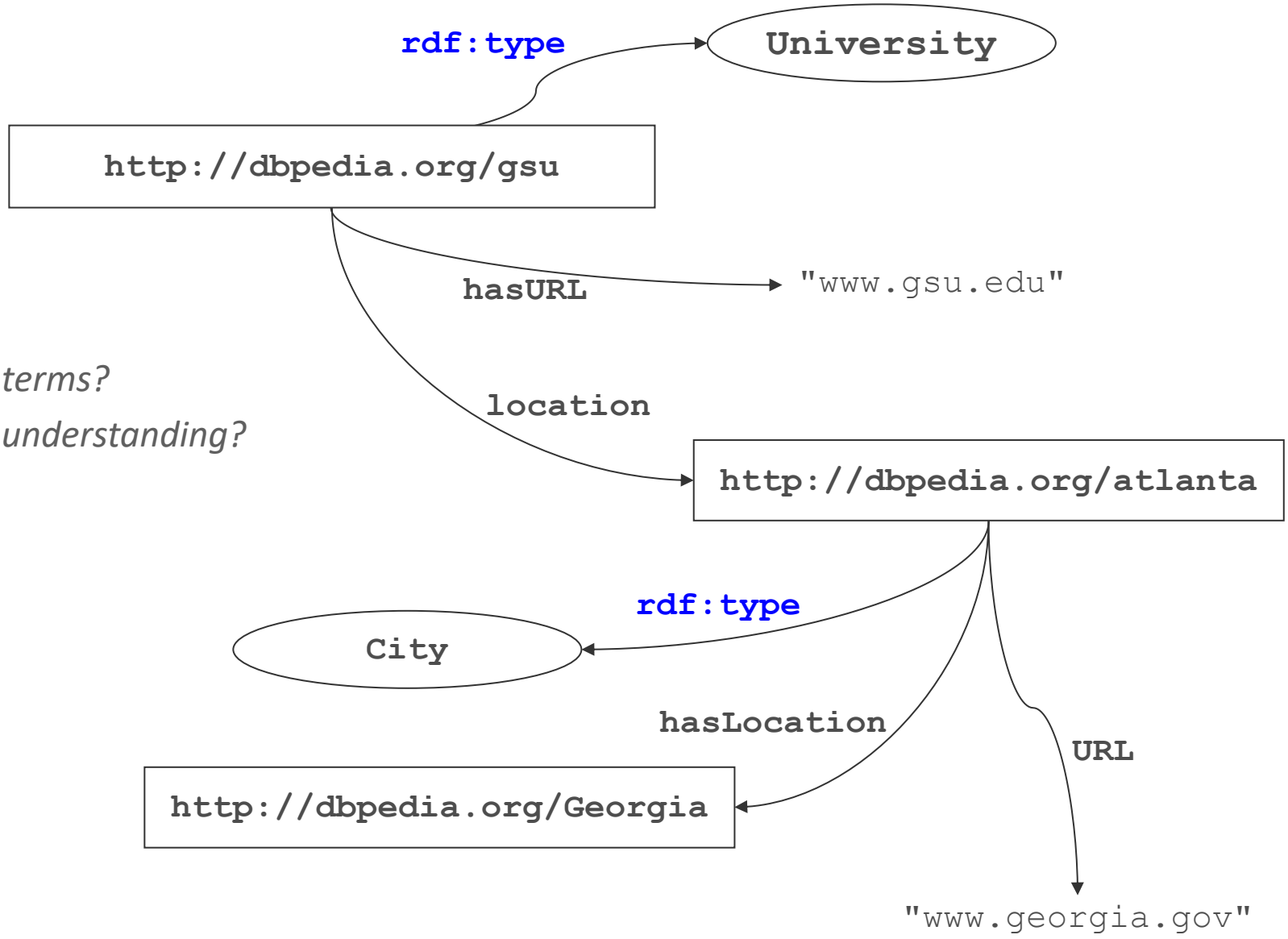
Re-thinking RDF: what is missing?



different terms?

different understanding?

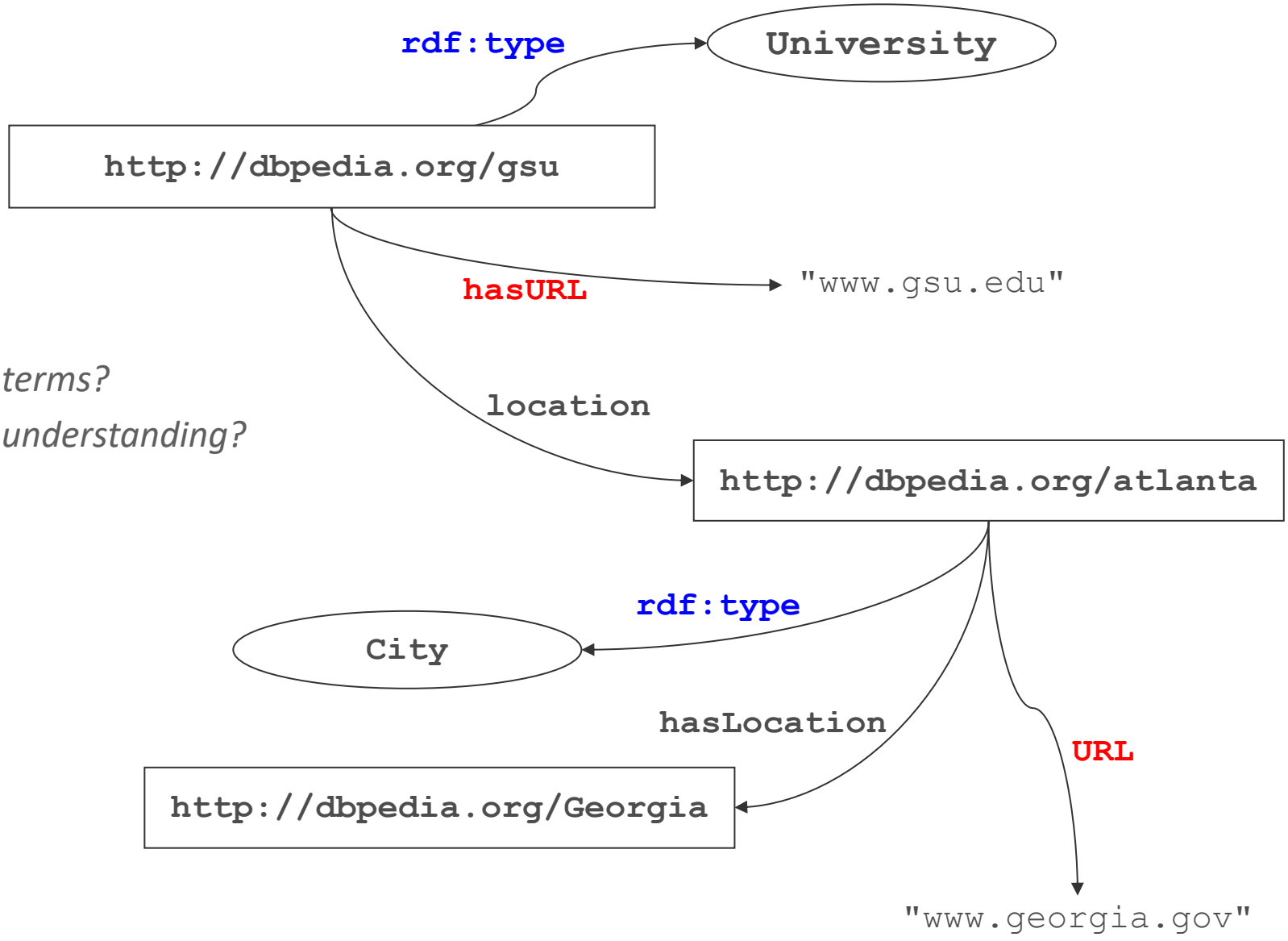
Re-thinking RDF: what is missing?



different terms?

different understanding?

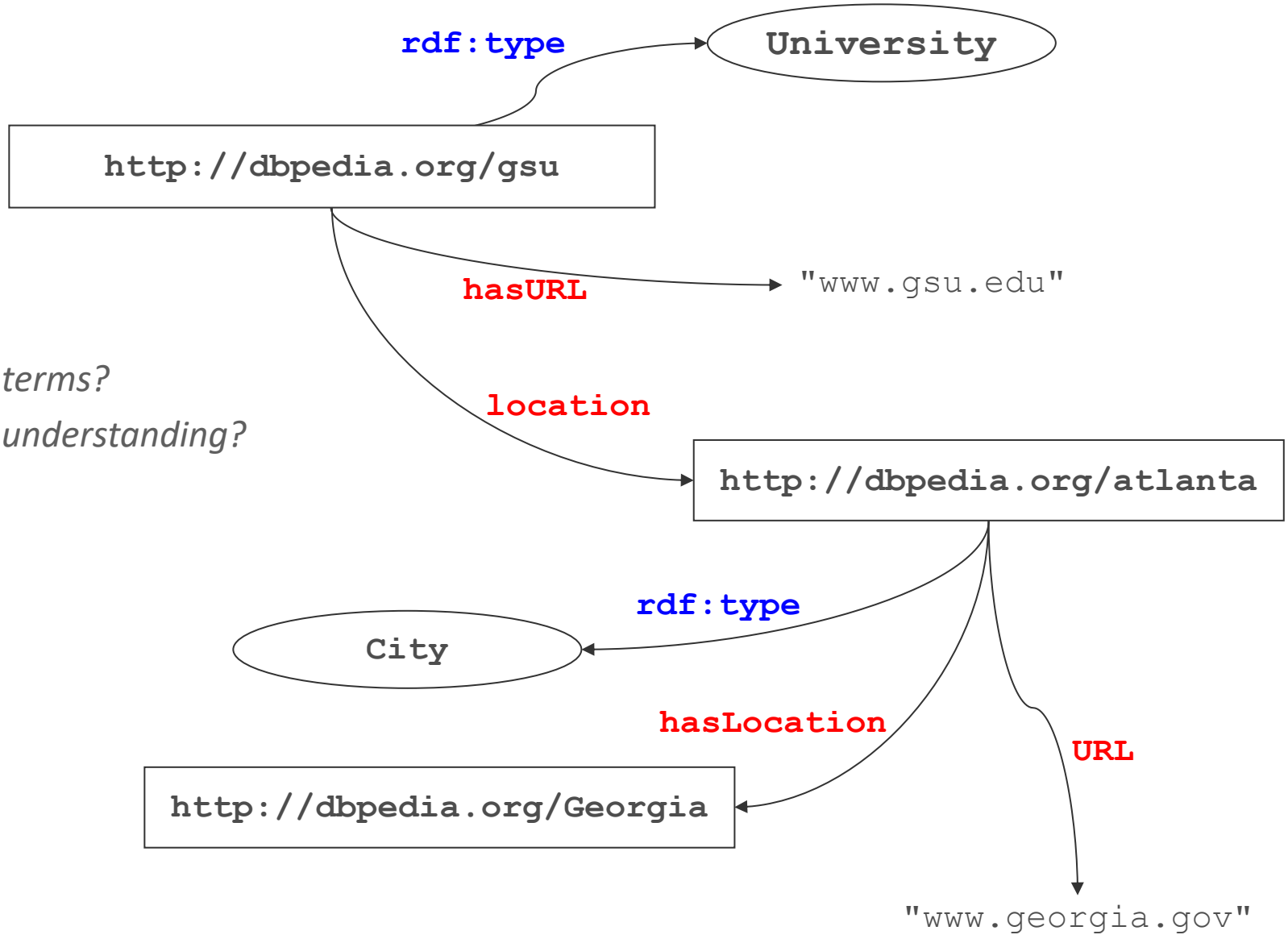
Re-thinking RDF: what is missing?



different terms?

different understanding?

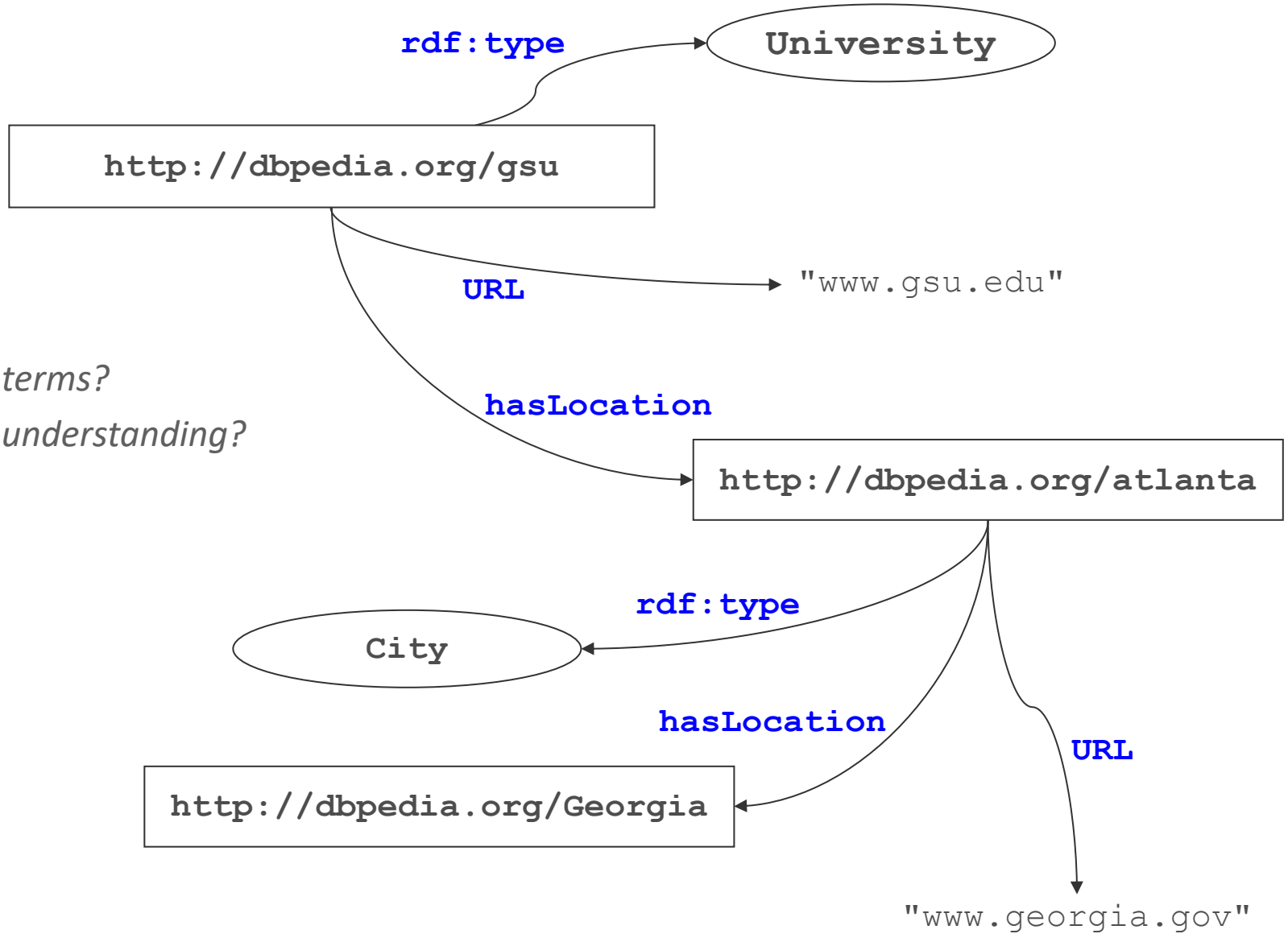
Re-thinking RDF: what is missing?



different terms?

different understanding?

Re-thinking RDF: what is missing?



different terms?

different understanding?

Re-thinking RDF: what is missing?

- we need a way to specify what terms we can use when describing resources, because "common terms = shared understanding"
- in addition, who define the terms such as `University`, `City`? do we define them before we can use them?
- remember `rdf:type` which actually "means" `is_a` relationship? terms like that would be great
- but rdf vocabulary only exists to help machines operate on RDF statements, it is not there to provide the common terms we need

Re-thinking RDF: what is missing?

we need a ***dictionary***, so everyone can share the same understanding when we say things

Re-thinking RDF: what is missing?

dictionary ≅ **ontology**

(*we used to call them ontologies …*
until we found it scared people away)