MILLENNIUM® +

Breakthrough science. Breakthrough medicine.℠

Matthias Samwald
Medical University of Vienna, Austria
Jonathan Rees
Science Commons
Alan Ruttenberg
Senior Scientist, Computational Biology

Ontology-based URI Resolution

http://tinyurl.com/ {vhb7e, y2v7kj, u6ztt}

ISWC 2006 HCLS Workshop
November 6, 2006

What goes wrong with URLs (courtesy public-semweb-lifesci)

- The server disappears
- The content disappears - 404
- The content might change and you want to know and communicate what it used to be
- Access to the content is too slow
- Access to the content is too public
- The content is very big
- You don't know if a URI is an information resource or not
- You want to record and access metadata - information about some information resource - and you don't know where to get it.
- You don't know what format an information resource is encoded in.

Why talk about this?

- Not really about Health Care/Life Sciences
- But as SW consumers we use URIs ***A LOT***
- This topic has distracted bioinformatics researchers
- A new approach, consonant with SW technology
- *"Take your own medicine"*

Existing proposals: LSIDs

- Authority
- “Location independence”
- Data/Metadata distinction
- Access method independence
- Versioning

- (similar: ARK, purl, DOI…)

Existing proposals: http-range14

- Use http.
- Use result code to recognize potential non-information resources
- *Result code 2xx = information resource*
- *Result code 3xx = any resource, pointer to more information*
- *Result code 4xx unknown type*
- “303” to get more information about the thing

Existing proposals: Content negotiation

- Agent asks for resource
- Server responds with list of content types and where to get each
- Agent chooses which to retrieve

Selected issues with proposed solutions

- http-range14 - “late” don’t know anything until you do the retrieval
- Content negotion - confusion over what the thing is - e.g. foaf human readable document, rdf at same address. Try talking about the ugly font.
- LSID - requires server deployment. Based on web services (slow). Unclear semantics of “versions”, “metadata”, “data”
- No single proposal deals with all issues

An Alternative

- Use the our SW tools to help solve this problem
- Represent the information that you want to know about URIs in OWL.
- Build an ontology to represent consensus/schema.
- Take advantage of consistency checking, inheritence

- *(Don't break the web :)*

Goals

- Transparent/explicit. Contract based.
- Adjustable
- Extendable
- Ontologically sound

Different things

- The temperature of a patient (not an information resource)
- A instrument that measures and reports temperature (not an information resource)
- The record retrieved when you query the instrument (an information resource)
- The record that you retrieved at a certain time and you copied and saved (an information resource)

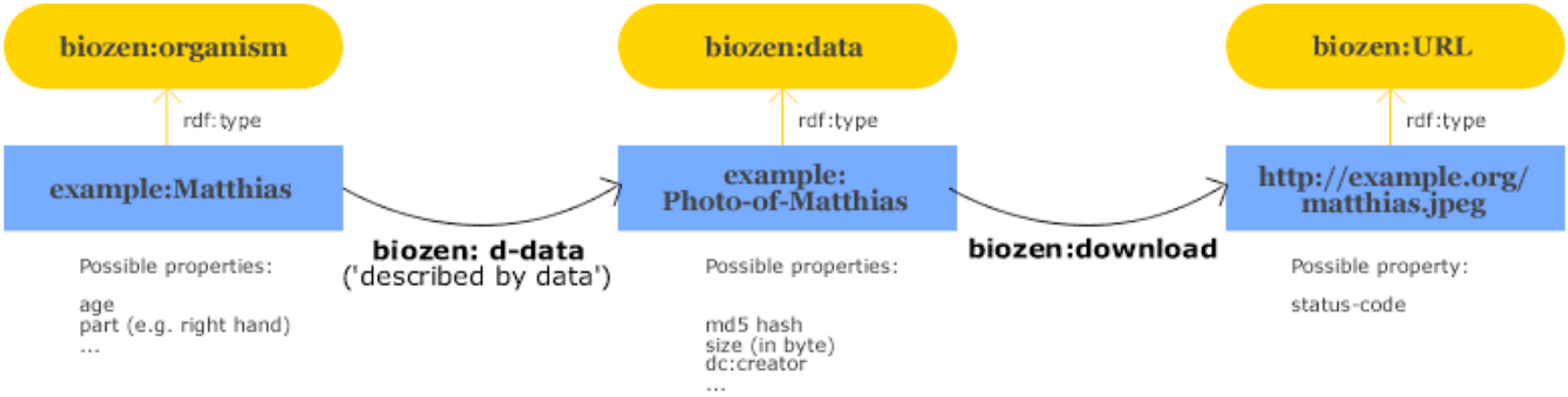
but related

Two iterations of implementation

- Simple, less flexible. “Vitamin Source”, an extension of Samwald’s BioZen ontology.
- **http://tinyurl.com/vhb7e (Web Demo)**
- More elaborate, handles more cases, implemented in OWL, using LSW.
- **http://tinyurl.com/y2v7kj (OWL Ontology)**
- **http://tinyurl.com/u6ztt (Prototype implementation)**

Vitamin Source: Three kinds of resources

- ‚Thing' that exists in physical reality (e.g. a person, a building, a molecule)
- Digital representation of this thing (e.g. an image, a HTML page)
- ‚Location' of the digital representation (e.g. a URL where the digital representation can be downloaded from)



Real-world entity

Digital representation (file description)

HTTP - URL (download location)

Vitamin Source

- A small demonstration of bio-zen digital resource management
- Implemented with **ARC**
  - ‚Appomsphere RDF Classes‘
  - Lightweight SPARQL / RDF library for PHP
  - Compatible with PHP4 and MySQL4 (runs on most webservers, even on cost-free webspace -> low entry barrier)
  - Pushes as much as possible into MySQL
  - Relatively performant

Vitamin Source

- Simulation of 4 different servers:
  - Sparql query interface
  - Sparql endpoint one
  - Sparql endpoint two
  - Fileserver

Find all organisms with the rdfs:label “Matthias“ and show all available data.

```
PREFIX
  biozen: <http://neuroscientific.net/bio-zen.owl#>
SELECT
  ?organism ?data
WHERE {
  ?organism rdfs:label "Matthias" .
  ?organism rdf:type biozen:organism .
  ?organism biozen:d-data ?data .
}
```

Results:

Results from *http://neuroscientific.net/vitamin-source/endpoint-one/endpoint.php*

organism) http://www.example.org/**Matthias**
data) http://www.example.org/**Photo-of-Matthias**

Results from *http://neuroscientific.net/vitamin-source/endpoint-two/endpoint.php*

What information do we have about the photo?

```
SELECT
  ?predicate ?object
WHERE {
  <http://www.example.org/Photo-of-Matthias>
  ?predicate ?object .
}
```

Results:

Results from *http://neuroscientific.net/vitamin-source/endpoint-one/endpoint.php*

predicate) http://neuroscientific.net/bio-zen.owl#**mime-type**
object) "**img/jpeg**"

predicate) http://neuroscientific.net/bio-zen.owl#**download**
object) http://neuroscientific.net/vitamin-source/fileserver/**matthias.jpeg**

predicate) http://neuroscientific.net/bio-zen.owl#**size**
object) "**50000**"

predicate) http://www.w3.org/1999/02/22-rdf-syntax-ns#**type**
object) http://neuroscientific.net/bio-zen.owl#**data**

predicate) http://purl.org/dc/elements/1.1/**rights**
object) "**Public Domain**"

predicate) http://neuroscientific.net/bio-zen.owl#**md5**
object) "**a1c1d3def**"

predicate) http://purl.org/dc/elements/1.1/**date**
object) "**2006-08-22**"


Greetings
from
Austria!

Distributing information across Servers.

- In first example, all of the information resides on one server.

- However, the relations of the *digital resource management model* can be distributed over independent servers.

Search for Parkin mRNA

```
PREFIX
  biozen: <http://neuroscientific.net/bio-zen.owl#>
SELECT
  ?rna ?label
WHERE {
  ?rna rdfs:label ?label .
  ?rna rdf:type biozen:rna-population .
  FILTER regex(?label, "parkin")
}
```

Results:

Results from *http://neuroscientific.net/vitamin-source/endpoint-one/endpoint.php*

rna) http://www.example.org/**parkin-mRNA-molecule-population**
label) "**Population of parkin mRNA molecules**"

Search for data about (this specific population of) Parkin mRNA

```
PREFIX
  biozen: <http://neuroscientific.net/bio-zen.owl#>
SELECT
  ?data ?predicate ?object
WHERE {
 <http://www.example.org/parkin-mRNA-molecule-
population>
 biozen:d-data ?data .
  ?data ?predicate ?object
}
```

Results from *http://neuroscientific.net/vitamin-source/endpoint-one/endpoint.php*

data) http://www.example.org/NM_013987_XML
predicate) http://neuroscientific.net/bio-zen.owl#**mime-type**
object) "**rdf/xml**"

data) http://www.example.org/NM_013987_XML
predicate) http://neuroscientific.net/bio-zen.owl#**download-schema**
object) http://www.ncbi.nlm.nih.gov/dtd/NCBI_Seqset.dtd

data) http://www.example.org/NM_013987_XML
predicate) http://www.w3.org/2000/01/rdf-schema#**label**
object) "**Entrez Nucleotide entry NM_013987 (XML version)**"

data) http://www.example.org/NM_013987_XML
predicate) http://www.w3.org/1999/02/22-rdf-syntax-ns#**type**
object) http://neuroscientific.net/bio-zen.owl#**data**

data) http://www.example.org/NM_013987_FASTA
predicate) http://neuroscientific.net/bio-zen.owl#**mime-type**
object) "**text/plain**"

data) http://www.example.org/NM_013987_FASTA
predicate) http://www.w3.org/2000/01/rdf-schema#**label**
object) "**Entrez Nucleotide entry NM_013987 (XML version)**"

How can we get the XML data?

```
PREFIX
  biozen: <http://neuroscientific.net/bio-zen.owl#>
SELECT
  ?url
WHERE {
 <http://www.example.org/NM_013987_XML>
 biozen:download ?url .
}
```

This time, results is from different endpoint

Results:

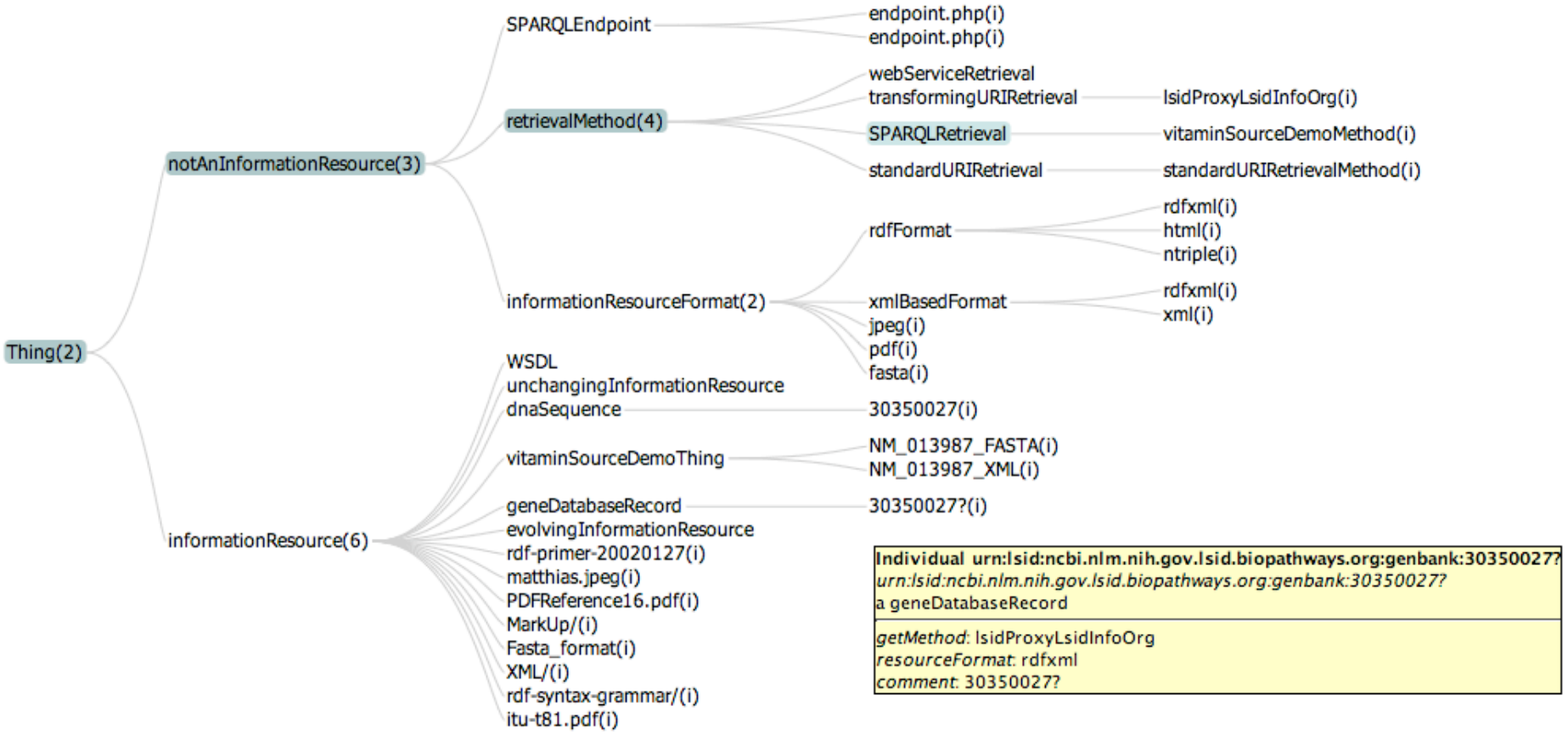
Results from *http://neuroscientific.net/vitamin-source/endpoint-one/endpoint.php*

Results from *http://neuroscientific.net/vitamin-source/endpoint-two/endpoint.php*

url) http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nucleotide&id=7669537&rettype=xml

Second Implementation

http://tinyurl.com/y2v7kj



InformationResource
NotAnInformationResource

- Information resources are conceptually “Gettable”
- They might not be able to be retrieved at a particular time
- They might change
- Ask yourself: “Would it be possible to get the thing itself over a network”
- Disjoint

UnchangingInformationResource
EvolveableInformationResource

- UnchangingInformationResource is like LSID "data". A promise is made that the content will never change.
- EvolveableInformationResource are resources that might change (even if we don't want them to, e.g. NCBI gene records)
- Disjoint

RetrievalMethod

- A way to get an information resource.
- Some examples
  - StandardURIRetrieval
  - TransformingUriRetrieval
  - SPARQLEndpointRetrieval
  - WebServiceRetrieval

RetrievalMethod (notes)

- There may be more than one.
- When more than one try them all in random order, or explicitly represent preference (exercise to the reader).
- E.g. For company specific retrieval, add another RetrievalMerthod to an appropriate upper class (one more triple)

InformationResourceFormat

- Explicitly give enough information to know what you will have to parse should you retrieve the resource (so you can choose whether or not to retrieve)
- Like mime/type - BUT only the format, not the type (that's for defining by class). Each instance has property(*formatSpecification*) who's value is a specification of the format.
  - RDFXML
  - RDFTurtle
  - JPEG
  - PDF
  - HTML

SPARQLEndpoint Definitions

```
(datatype-property !queryPattern)
(datatype-property !URIVariableString)
(object-property !useSPARQLEndpoint)
(class !SPARQLEndpoint
   "A sparql endpoint can accept sparql queries. The sparql query
    constructed is expected to have a single variable ?url.
    It is constructed by substituting the uri of the individual with the
    URIVariableString in the !queryPattern. This query is then run against
    the endpoint in the property !useSPARQLEndpoint"
    :partial !notAnInformationResource)
 (class !SPARQLRetrieval :partial
  (intersection-of
     !retrievalMethod
     (restriction !queryPattern (cardinality 1))
     (restriction !URIVariableString (cardinality 1))
     (restriction !useSPARQLEndpoint (some-values-from !SPARQLEndpoint))
     (restriction !useSPARQLEndpoint (all-values-from !SPARQLEndpoint))))
```

Vitamin Source endpoint instances

```
(individual !<http://neuroscientific.net/vitamin-source/endpoint-one/endpoint.php>
  (type !SPARQLEndpoint))

(individual !<http://neuroscientific.net/vitamin-source/endpoint-two/endpoint.php>
  (type !SPARQLEndpoint))

(individual !vitaminSourceDemoMethod
    "Example of Matthias SPARQL endpoint for resolving URLs"
    (type !SPARQLRetrieval)
    (value !useSPARQLEndpoint
       !<http://neuroscientific.net/vitamin-source/endpoint-two/endpoint.php>)
    (value !useSPARQLEndpoint
       !<http://neuroscientific.net/vitamin-source/endpoint-one/endpoint.php>)
    (value !URIVariableString "%%URI%%")
    (value !queryPattern
      "PREFIX biozen: <http://neuroscientific.net/bio-zen.owl#>
       SELECT ?url
       WHERE {%%URI%% biozen:download ?url . }"))
```

Vitamin Source individuals

```
(class !vitaminSourceDemoThing :partial
    (intersection-of
     !informationResource
     (restriction !getMethod (has-value !vitaminSourceDemoMethod))))

 (individual !<http://www.example.org/NM_013987_XML>
   (type !vitaminSourceDemoThing)
   (value !resourceFormat !xml))

 (individual !<http://www.example.org/NM_013987_FASTA>
   (type !vitaminSourceDemoThing)
   (value !resourceFormat !fasta))
```

Vitamin Source resolution, in action

(get-information-resource-location !<http://www.example.org/NM_013987_XML>)

http://tinyurl.com/u6ztt

1. Get the **getMethod** of NM_013987_XML

=> !vitaminSourceDemoMethod (inherited)

2. Get the direct type of the method. It is **!SPARQLRetrieval**
3. Dispatch to code based on type
4. Code retrieves **queryPattern** => “PREFIX biozen:….%%URI%%…”
5. Code retrieves **URIVariableString** => “%%URI%%”
6. Code replaces %%URI%% with NM_013987_XML uri
7. Code retrieves **useSPARQLEndpoint** (2 of them)
8. Constructs http gets, e.g.

   http://neuroscientific.net/vitamin-source/endpoint-one/endpoint.php?query=PREFIX%20biozen:%20%3Chttp://neuroscientific.net/bio-zen.owl%23%3E%20SELECT%20?url%20WHERE%20%7B%3Chttp://www.example.org/NM_013987_XML%3E%20biozen:download%20?url%20.%20%7D"

9. Query returns =>

http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nucleotide&id=7669537&rettype=xml

LSID via HTTP resolution

```
(datatype-property !matchingPattern)
(datatype-property !replacementPattern)
(class !transformingURIRetrieval
   "In a transformingURIRetrieval, the URI is matched against a regular
   expression with group captures. Then the groups can be substituted into a
   replacement pattern to form a url suitable for get-url. E.g. lsid http proxy"
   :partial
    (intersection-of
     !retrievalMethod
     (restriction !matchingPattern (cardinality 1))
     (restriction !replacementPattern (cardinality 1))))

(individual !lsidProxyLsidInfoOrg
   "An example of an LSID http resolver from, implemented as a
   transformingURIRetrieval"
   (type !transformingURIRetrieval)
   (value !matchingPattern "(.*)")
   (value !replacementPattern "http://lsid-info.org/$1"))
```

LSID via HTTP resolution

```
(class !dnaSequence "Representations of DNA sequences"
    :partial !informationResource)
(class !geneDatabaseRecord "Records describing genes"
    :partial !informationResource)

(individual !<urn:lsid:ncbi.nlm.nih.gov.lsid.biopathways.org:genbank:30350027>
  (type !dnaSequence)
  (value !getMethod !lsidProxyLsidInfoOrg)
  (value !resourceFormat !fasta))

(individual !<urn:lsid:ncbi.nlm.nih.gov.lsid.biopathways.org:genbank:30350027?>
  (type !geneDatabaseRecord)
  (value !getMethod !lsidProxyLsidInfoOrg)
  (value !resourceFormat !rdfxml)))
```

About sharing bare URIs

- Don’t, unless you have to. Generally messages should be a set of triples giving adequate information about type, resolution
- If you do, use existing best practices to make them last, e.g. PURL

Conclusions

- It's feasible - Prototype implementation provided
- It's extendable - OWL ontology tells you enough to write the supporting code
- It's transparent - Everything you need to know is in the ontology

- Let's use this for HCLS and see how it plays out!

Breakthrough science. Breakthrough medicine.SM