Very Large Digital Libraries: A Model

Nicolas Spyratos[1] Carlo Meghini[2] Jitao Yang[1]

[1]Université Paris Sud - LRI, Paris
[2]Consiglio Nazionale delle Ricerche - ISTI, Pisa

Ecole thématique BDA
Les Houches, May 17, 2010

EBDA 2010

Outline

Outline

Outline

Outline

Motivations

A Digital Library (DLs) is a curated collection of digital objects that offers services on those objects to communities of users.

- Amazon: books, CDs, DVDs, etc.
- ACM, IEEE, et al.: scientific papers, conferences, journals, etc.
- the web: hypertexts, documents, services, etc.

Europeana:

> *"A unique resource for Europe's distributed cultural heritage, ensuring a common access to Europe's libraries, archives and museums." (Horst Forster, Director, European Commission)*

Europeana is building a Cultural Heritage Portal that mediates amongst:

- single repositories, such as the Biblioteque Nationale de France
- vertical portals, such as Michael
- National aggregators

We see Europeana as a complex DL built on top of simpler DLs.

For its operation, Europeana needs to integrate different types of information:

- metadata about cultural heritage objects
- language resources (dictionaries, stemmers)
- information structures (taxonomies, vocabularies, thesauri)
- authority files (*e.g.*, VIAF)
- search engines
- social networks

The problem, sometimes known as *semantic interoperability*, is currently attacked by *ad hoc* approaches, which are very time-consuming to program and much error prone.

Our goal is to attack the problem at the foundations, building a theory on which to base effective solutions.

We begin by modelling simple DLs.

A formal model, to be used as a basis for implementations, research, extensions, comparisons.

Requirements

We need a level of abstraction over the overwhelming amount of details involved in the management of a DL, *i.e.*, a *data model.*

Operations provided by the model:

- *describe* an object of interest according to the possibly many vocabularies of the DL communities;
- *discover* objects of interest based on their descriptions;
- *view* the content of a discovered object;
- *identify* an object of interest, by assigning an identity to it;
- *re-use* objects in a different context.

We want to define structures for carrying out these operations and give algorithms for their implementation.

Digital Objects

Basic notion: digital object.

A digital object is a piece of information in digital form, such as a PDF document, a JPEG image, a DVD movie, a URI, and so on.

A digital object can be processed by a computer, for instance it can be stored in memory and displayed on a screen.

O : the (non-empty, countable) collection of digital objects.

A digital object has four independent features:

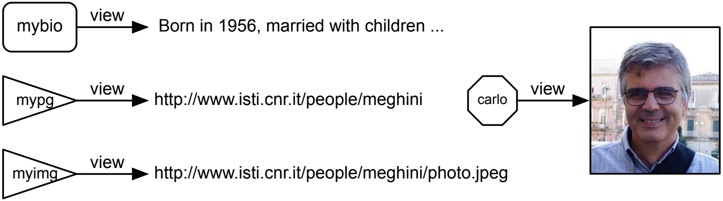
1. it can be viewed
2. it has content
3. it has versions
4. it has descriptions.

View

We assume that each digital object can be *viewed* using an appropriate mechanism.

view(o) : the view of o

view is a total function having the set O as domain. The range of view is outside the scope of our model.



Content

The *content* of an object o is a set of objects which constitute o from a structural point of view.

For example, a book is constituted by the set of its chapters as its content, each chapter being an individual object.

Similarly, an exhibition of paintings has the set of its paintings as its content.

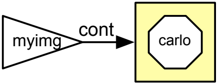
document: a rendering of some content on a specific device.

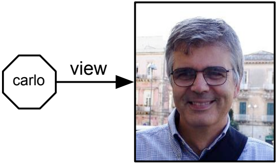
An image identified by a URI

myimg: a digital object (a URI)
view(myimg) = "`http://www.cnr.it/meghini/ph.jpeg`" (a string)

carlo: a digital object (an image)
view(carlo) = *a photograph*

carlo is a content of myimg





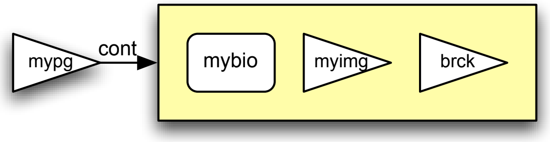
A Web page

mypg: a digital object (a URI)
view(mypg) = "`http://www.cnr.it/meghini/index.html`"

mybio: a digital object (a text)
view(mybio) = "Born 1956, married with children, ..."

brck: a digital object (a URI)
view(brck) = "`http://www.bricksfactory.org`"

mybio, myimg and brck are contents of mypg



Descriptions

Descriptions support the interpretation, the discovery, and the management of objects.

A *description* is a statement that gives salient features of some unspecified object.

- tall, blond, likes Mozart, plays tennis
- $\text{Tall}(x) \wedge \text{Blond}(x) \wedge \text{Likes}(x,\text{Mozart}) \wedge \text{Plays}(x, \text{tennis})$
- $\text{Tall} \sqcap \text{Blond} \sqcap (\exists\text{Likes}.\{\text{Mozart}\}) \sqcap (\exists\text{Plays}.\{\text{tennis}\})$

Descriptions are then *assigned* to objects:

- John is tall, blond, likes Mozart, plays tennis
- $\text{Tall}(\text{john}) \wedge \text{Blond}(\text{john}) \wedge \text{Likes}(\text{john},\text{Mozart}) \wedge \text{Plays}(\text{john},\text{tennis})$
- $\text{Tall} \sqcap \text{Blond} \sqcap (\exists\text{Likes}.\{\text{Mozart}\}) \sqcap (\exists\text{Plays}.\{\text{tennis}\})\ (\text{john})$

An object can be seen from different points of view, each leading to a different description of the object. In general, an object may have many descriptions in a DL.

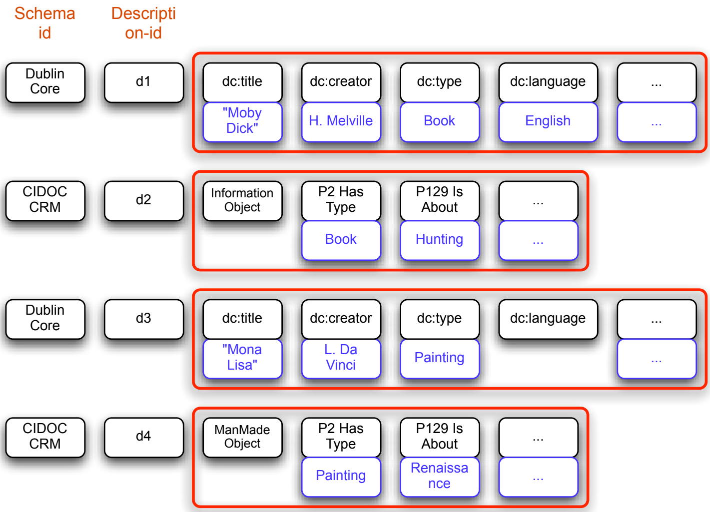
The elements of descriptions are drawn from *schemas*, where they are defined and (possibly) related to each other.

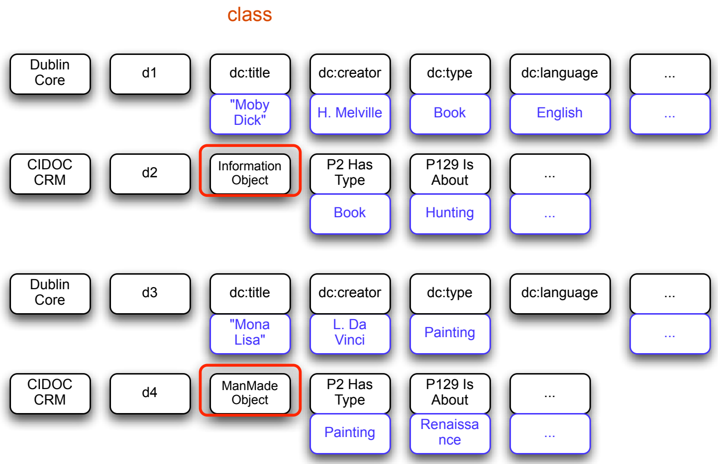
A schema consists of:

- a set of classes
- a set of properties
- sub-class statements forming the class is-a hierarchy
- sub-property statements forming the property is-a hierarchy
- statements about domain and range of properties

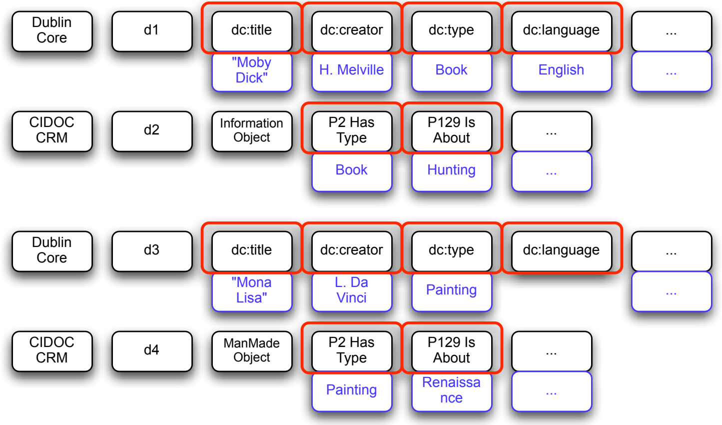
i.e., RDFS.

A common practice in digital libraries is to form descriptions by mixing classes and properties from several schemas.


Schema id
Description-id
Dublin Core
d1
dc:title
"Moby Dick"
dc:creator
H. Melville
dc:type
Book
dc:language
English
...
...
CIDOC CRM
d2
Information Object
P2 Has Type
Book
P129 Is About
Hunting
...
...
Dublin Core
d3
dc:title
"Mona Lisa"
dc:creator
L. Da Vinci
dc:type
Painting
dc:language
...
...
CIDOC CRM
d4
ManMade Object
P2 Has Type
Painting
P129 Is About
Renaissance
...
...


class
Dublin Core
d1
dc:title
"Moby Dick"
dc:creator
H. Melville
dc:type
Book
dc:language
English
...
...
CIDOC CRM
d2
Information Object
P2 Has Type
Book
P129 Is About
Hunting
...
...
Dublin Core
d3
dc:title
"Mona Lisa"
dc:creator
L. Da Vinci
dc:type
Painting
dc:language
...
...
CIDOC CRM
d4
ManMade Object
P2 Has Type
Painting
P129 Is About
Renaissance
...
...

property


Dublin Core
d1
dc:title
dc:creator
dc:type
dc:language
...
"Moby Dick"
H. Melville
Book
English
...
CIDOC CRM
d2
Information Object
P2 Has Type
P129 Is About
...
Book
Hunting
...
Dublin Core
d3
dc:title
dc:creator
dc:type
dc:language
...
"Mona Lisa"
L. Da Vinci
Painting
...
CIDOC CRM
d4
ManMade Object
P2 Has Type
P129 Is About
...
Painting
Renaissance
...

Versions

The user is working on a text, of which he wants to maintain versions:

- folder o
  - file o_1
    - text t_1 (view(t_1) : the initial text)
  - file o_2
    - text t_2 (view(t_2) : the modified text)

We view o as the identifier of our text and o_1 and o_2 as two versions of it.

Which version represents o at any point in time? any of the two, depending on context.

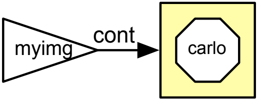
The versions of o are alternatives for o, not necessarily its evolution in time.

Digital Libraries: A Formal Definition

A DL is defined as a certain model of a function-free first-order theory $\mathcal{T}$.

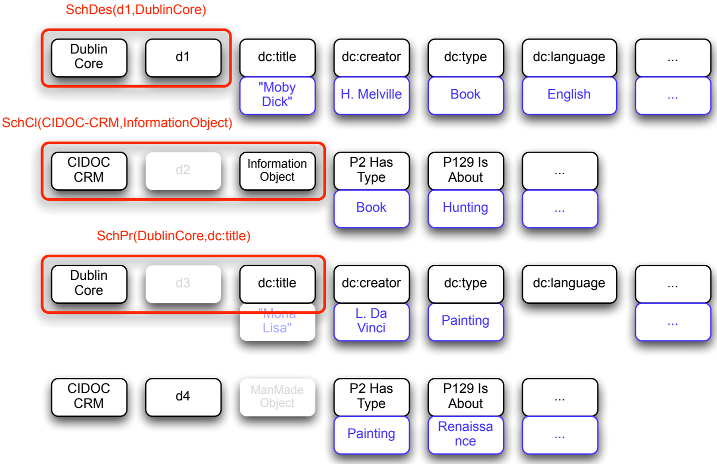
The language $\mathcal{L}$ of the theory consists of a set of predicate symbols for describing content, descriptions and versions of digital objects.

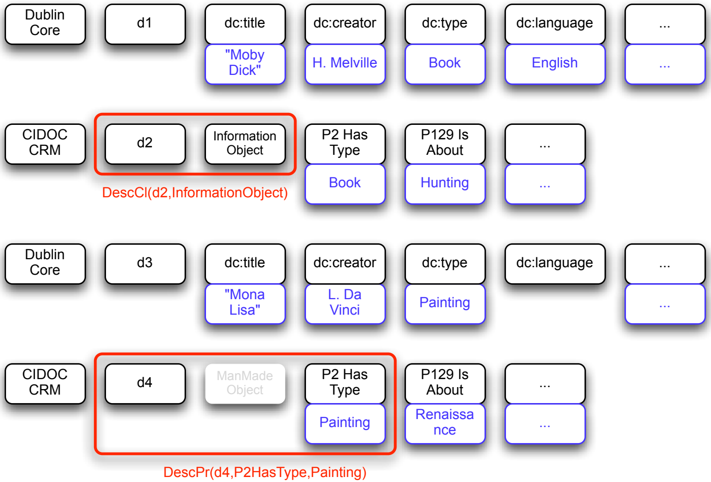
The axioms $\mathcal{A}$ of the theory fix the meaning of the predicate symbols.



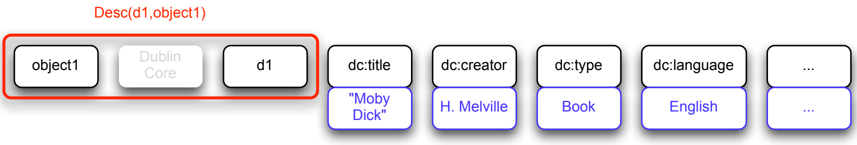
Cont(myimg,carlo)

To represent that carlo is a content of myimg


SchDes(d1,DublinCore)
Dublin Core
d1
dc:title
"Moby Dick"
dc:creator
H. Melville
dc:type
Book
dc:language
English
...
...
SchCl(CIDOC-CRM,InformationObject)
CIDOC CRM
d2
Information Object
P2 Has Type
Book
P129 Is About
Hunting
...
...
SchPr(DublinCore,dc:title)
Dublin Core
d3
dc:title
"Mona Lisa"
dc:creator
L. Da Vinci
dc:type
Painting
dc:language
...
...
CIDOC CRM
d4
ManMade Object
P2 Has Type
Painting
P129 Is About
Renaissance
...
...


Dublin Core
d1
dc:title
"Moby Dick"
dc:creator
H. Melville
dc:type
Book
dc:language
English
...
...
CIDOC CRM
d2
Information Object
P2 Has Type
Book
P129 Is About
Hunting
...
...
DescCl(d2,InformationObject)
Dublin Core
d3
dc:title
"Mona Lisa"
dc:creator
L. Da Vinci
dc:type
Painting
dc:language
...
...
CIDOC CRM
d4
ManMade Object
P2 Has Type
Painting
P129 Is About
Renaissa nce
...
...
DescPr(d4,P2HasType,Painting)

To associate objects with their descriptions:



d1 is asserted to be a description of object1

The language $\mathcal{L}$

Pred. symbols	*Informal Meaning*
$\mathrm{SchCl}(s, c)$	c is a class in schema s
$\mathrm{SchPr}(s, p)$	p is a property in schema s
$\mathrm{Dom}(s, p, c)$	c is a domain of p in s
$\mathrm{Ran}(s, p, c)$	c is a range of p in s
$\mathrm{IsaCl}(s, c_1, c_2)$	c_1 is a sub-class of c_2 in s
$\mathrm{IsaPr}(s, p_1, p_2)$	p_1 is a sub-property of p_2 in s
$\mathrm{SchDes}(d, s)$	d is a description over s
$\mathrm{DescCl}(d, c)$	any object described by d is an instance of c
$\mathrm{DescPr}(d, p, o)$	any object described by d has o as a value of p
$\mathrm{Cont}(o_1, o_2)$	o_1 is a content of o_2
$\mathrm{Desc}(o_1, o_2)$	o_1 is a description of o_2
$\mathrm{Vers}(o_1, o_2)$	o_1 is a version of o_2

The axioms $\mathcal{A}$

id	*Axiom*
A1	$\mathrm{Dom}(s,p,c) \rightarrow (\mathrm{SchPr}(s,p) \wedge \mathrm{SchCl}(s,c))$
A2	$\mathrm{Ran}(s,p,c) \rightarrow (\mathrm{SchPr}(s,p) \wedge \mathrm{SchCl}(s,c))$
A3	$\mathrm{IsaCl}(c_1,c_2,s) \rightarrow (\mathrm{SchCl}(s,c_1) \wedge \mathrm{SchCl}(s,c_2))$
A4	$\mathrm{IsaPr}(p_1,p_2,s) \rightarrow (\mathrm{SchPr}(s,p_1) \wedge \mathrm{SchPr}(s,p_2))$
A5	$(\mathrm{DescCl}(d,c) \wedge \mathrm{SchDes}(d,s)) \rightarrow \mathrm{SchCl}(s,c)$
A6	$(\mathrm{DescPr}(d,p,o) \wedge \mathrm{SchDes}(d,s)) \rightarrow \mathrm{SchPr}(s,p)$
A7	$(\mathrm{DescCl}(d,c_1) \wedge \mathrm{SchDes}(d,s) \wedge \mathrm{IsaCl}(s,c_1,c_2)) \rightarrow \mathrm{DescCl}(d,c_2)$
A8	$(\mathrm{DescPr}(d,p_1,o) \wedge \mathrm{SchDes}(d,s) \wedge \mathrm{IsaPr}(s,p_1,p_2)) \rightarrow \mathrm{DescPr}(d,p_2,o)$
A9	$(\mathrm{DescPr}(d,p,o) \wedge \mathrm{SchDes}(d,s) \wedge \mathrm{Dom}(s,p,c)) \rightarrow \mathrm{DescCl}(d,c)$

Defining a Digital Library

An interpretation of $\mathcal{L}$: a pair (D, I) where:

- D : the domain of the interpretation
- I : the intepretation function, assigning a relation of the appropriate arity over D to each predicate symbol in $\mathcal{L}$.

DL interpretations always range over digital objects: $D = \mathrm{O}$.

Definition

Given two interpretations I and I' of $\mathcal{L}$, I is *smaller* than I', $I \leq I'$, if $I(p) \subseteq I'(p)$ for each predicate symbols p in $\mathcal{L}$.

Intuitively, I are the facts inserted by users, and the corresponding DL is the minimal model of $\mathcal{T}$ that includes I.

The axioms can be re-written as an equivalent positive datalog program $P_{\mathcal{A}}$, *i.e.* a set of rules r of the form

$$L :- L_1, \ldots, L_n$$

where:

- $n \geq 0$
- the literals $L, L_1, \ldots, L_n$ are all positive
- L is the *head* of the rule r, $head(r)$
- $\{L_1, \ldots, L_n\}$ is the *body* of the rule r, $body(r)$.

$$
\begin{aligned}
&\mathrm{SchPr}(s,p) \coloneq \mathrm{Dom}(s,p,c)\\
&\mathrm{SchCl}(s,c) \coloneq \mathrm{Dom}(s,p,c)\\
&\mathrm{SchPr}(s,p) \coloneq \mathrm{Ran}(s,p,c)\\
&\mathrm{SchCl}(s,c) \coloneq \mathrm{Ran}(s,p,c)\\
&\mathrm{SchCl}(s,c_1) \coloneq \mathrm{IsaCl}(c_1,c_2,s)\\
&\mathrm{SchCl}(s,c_2) \coloneq \mathrm{IsaCl}(c_1,c_2,s)\\
&\mathrm{SchPr}(s,p_1) \coloneq \mathrm{IsaPr}(p_1,p_2,s)\\
&\mathrm{SchPr}(s,p_2) \coloneq \mathrm{IsaPr}(p_1,p_2,s)\\
&\mathrm{SchCl}(s,c) \coloneq \mathrm{DescCl}(d,c),\ \mathrm{SchDes}(d,s)\\
&\mathrm{SchPr}(s,p) \coloneq \mathrm{DescPr}(d,p,o),\ \mathrm{SchDes}(d,s)\\
&\mathrm{DescCl}(d,c_2) \coloneq \mathrm{DescCl}(d,c_1),\ \mathrm{SchDes}(d,s),\ \mathrm{IsaCl}(s,c_1,c_2)\\
&\mathrm{DescPr}(d,p_2,o) \coloneq \mathrm{DescPr}(d,p_1,o),\ \mathrm{SchDes}(d,s),\ \mathrm{IsaPr}(s,p_1,p_2)\\
&\mathrm{DescCl}(d,c) \coloneq \mathrm{DescPr}(d,p,o),\ \mathrm{SchDes}(d,s),\ \mathrm{Dom}(s,p,c)
\end{aligned}
$$

The application of $P_{\mathcal{A}}$ to a set of facts J is expressed by the immediate consequence operator $T_{P_{\mathcal{A}}}$.

$inst(P_{\mathcal{A}})$: the set of all rules that can be derived by instanciating the rules in $P_{\mathcal{A}}$ using the constants in J in all possible ways.

$$T_{P_{\mathcal{A}}}(J) = \{head(r) \mid r \in inst(P_{\mathcal{A}}) \text{ and } body(r) \subseteq J\}$$

$T_{P_{\mathcal{A}}}$ is monotone and therefore it admits a minimal fix-point $\mathcal{M}(P_{\mathcal{A}}, I)$ given by:

$$\mathcal{M}(P_{\mathcal{A}}, I) = \min\{X^n \mid X^n = X^{n+1}\}$$

where:

$$\begin{array}{rcl} X^0 & = & I \\ X^k & = & X^{k-1} \cup T_{P_{\mathcal{A}}}(X^{k-1}) \text{ for } k > 0 \end{array}$$

Digital Libraries

Proposition

If I is an interpretation of $\mathcal{L}$, then $\mathcal{M}(P_{\mathcal{A}}, I)$ is the minimal model of $\mathcal{A}$ that contains I.

Definition

Let I be any interpretation of $\mathcal{L}$. The *digital library over I*, DL_I, is the minimal model $\mathcal{M}(P_{\mathcal{A}}, I)$ of $\mathcal{A}$ that contains I.

In practice, one starts with the facts I inserted by the users when they record information about objects, their content, their descriptions and their versions.

The digital library over I can then be generated by applying the consequence operator $T_{P_{\mathcal{A}}}$ to the set I.

Querying a Digital Library

The language that is used to define a digital library is also used to query the digital library.

Problem: Descriptions make queries cumbersome!

The objects that are authored by *Alfred* and about *lattices:*

$$(\exists d_1)(\exists d_2)\mathrm{Desc}(d_1, x) \wedge \mathrm{Desc}(d_2, x) \wedge \mathrm{DescPr}(d_1, author, Alfred) \wedge \mathrm{DescPr}(d_2, abou$$

mentions two descriptions d_1 and d_2 which have nothing to do with the user information need.

A more intuitive and straightforward way of expressing the user information need would be to relate authorship and aboutness directly to the sought objects.

Two new predicate symbols that allow to directly connect description elements with the objects they are associated with.

- $\text{ClExt}(c, o)$ meaning that object o is an instance of class c. An assertion of this kind is called a *class instantiation*.
- $\text{PrExt}(o_1, p, o_2)$ meaning that object o_1 has object o_2 as value of the property p. An assertion of this kind is called a *property instantiation*.

Using these two symbols, the previous query can be expressed as follows:

$$\text{PrExt}(x, \mathit{author}, \mathit{Alfred}) \wedge \text{PrExt}(x, \mathit{about}, \mathit{lattices})$$

which is a direct translation of the user information need.

$\mathcal{Q}$: the first-order language having as predicate symbols those in $\mathcal{L}$ and the two above symbols.

Semantics?

Semantics of ClExt and PrExt

Intuitively, o is an instance of c if there exists some description d to that effect. This may happen in one of two different ways:

- c is a class in d and d is a description of o.
- c is the range of a property p in the schema of d, and d assigns o to property p. In this case, d may or (more likely) may not be a description of o.

Object o_2 is a p-value of object o_1 just in case o_1 has a description d that assigns o_2 to p.

Formally:

1. $\mathrm{Desc}(d, o) \wedge \mathrm{DescCl}(d, c) \rightarrow \mathrm{ClExt}(c, o)$
2. $\mathrm{Desc}(d, o_1) \wedge \mathrm{SchDes}(d, s) \wedge \mathrm{Ran}(s, p, c) \wedge \mathrm{DescPr}(d, p.o_2) \rightarrow \mathrm{ClExt}(c, o_2)$
3. $\mathrm{Desc}(d, o_1) \wedge \mathrm{DescPr}(d, p, o_2) \rightarrow \mathrm{PrExt}(o_1, p, o_2)$

These axioms are translated into the equivalent positive datalog program $P_{\mathcal{Q}}$ given by:

$$
\begin{aligned}
&\mathrm{ClExt}(c, o) \text{ :– } \mathrm{Desc}(d, o),\ \mathrm{DescCl}(d, c)\\
&\mathrm{ClExt}(c, o_2) \text{ :– } \mathrm{Desc}(d, o_1),\ \mathrm{SchDes}(d, s),\ \mathrm{Ran}(d, p, o_2),\ \mathrm{DescPr}(d, p, o_2)\\
&\mathrm{PrExt}(o_1, p, o_2) \text{ :– } \mathrm{Desc}(d, o_1),\ \mathrm{DescPr}(d, p, o_2)
\end{aligned}
$$

Definition

(Query over a digital library) A *query over* a digital library is any open well-formed formula $\alpha(x_1, \ldots, x_n)$ of $\mathcal{Q}$ with $n \geq 1$ free variables $x_1, \ldots, x_n$.

The answer of a query with n free variables is the set of n-tuples of objects $\langle o_1, \ldots, o_n \rangle$ such that, when every variable x_i is bound to the corresponding object o_i, the resulting ground formula of $\mathcal{L}$ is true in DL_I.

Definition

(Answer of a query) The *answer* of a query $\alpha(x_1, \ldots, x_n)$ over a digital library DL_I is given by:

$$ans(\alpha, I) = \{\langle o_1, \ldots, o_n \rangle \mid \alpha(o_1, \ldots, o_n) \in \mathcal{M}(P_{\mathcal{A}} \cup P_{\mathcal{Q}}, I)\}$$

Query evaluation

A simple strategy:

1. Store the initial set of facts I in a relational database $RDB(I)$.
2. Expand $RDB(I)$ until $RDB(\mathcal{M}(P_{\mathcal{A}} \cup P_{\mathcal{Q}}, I))$ is obtained; this requires adding tuples to the tables in $RDB(I)$ using the inference mechanism that we have described earlier.
3. Map each query q against the digital library to an equivalent SQL query $SQL(q)$.
4. Evaluate $SQL(q)$ against $RDB(\mathcal{M}(P_{\mathcal{A}} \cup P_{\mathcal{Q}}, I))$.

One of the problems here is to design optimal algorithms for maintaining $RDB(\mathrm{DL}_I)$ in presence of user updates.

Alternatively: compute query answers directly on $RDB(I)$ *without* expanding it to $RDB(\mathrm{DL}_I)$.

In this case, the problem is defining an inference mechanism for query answering directly over $RDB(I)$.

Conclusions and future work

We have the main elements of a DL model.

We have started investigating federation of digital libraries.

Future work:

- RDF implementation
- query evaluation

Thank you!

Any question?