Logical Information Systems and the Semantic Web

Sébastien Ferré, Team LIS
Data and Knowledge Management, Irisa

Séminaire L3I, May 5th, 2011

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES

UMR IRISA

Plan

Logical Information Systems (LIS)

The Semantic Web

LIS for the Semantic Web

Conclusion and Future Work

Plan

Classical Paradigms for Information Retrieval

The two *most common paradigms* are:

- Query answering: databases, search engines, ...
  - expressive query language (SQL, keywords)
  - difficult for end-users (SQL) or ambiguous (keywords)
  - no navigation among answers
  - problem of empty results
- Hierarchical navigation: file systems, documents, many websites, ...
  - easy to use (mimick physical organization)
    - e.g., `MyPhotos/2010/ICFCA_Agadir/`
  - paths form a rigid query language
  - restricted navigation

Need for combining querying and navigation!

New Paradigms for Information Retrieval

A *new paradigm* has recently emerged under several names:

- Conceptual Navigation [Godin et al 1993] based on Formal Concept Analysis (FCA)
  - the basis of the LIS approach
  - FCA is also used for data mining and machine learning
- Dynamic Taxonomies [Sacco 2000] and Faceted Search [Hearst 2003]
  - domains: human-computer interaction, information systems
  - becoming the *de facto* standard in e-commerce websites (Yahoo!Shopping) and others (DBLP, Microsoft Sharepoint)

The LIS Approach

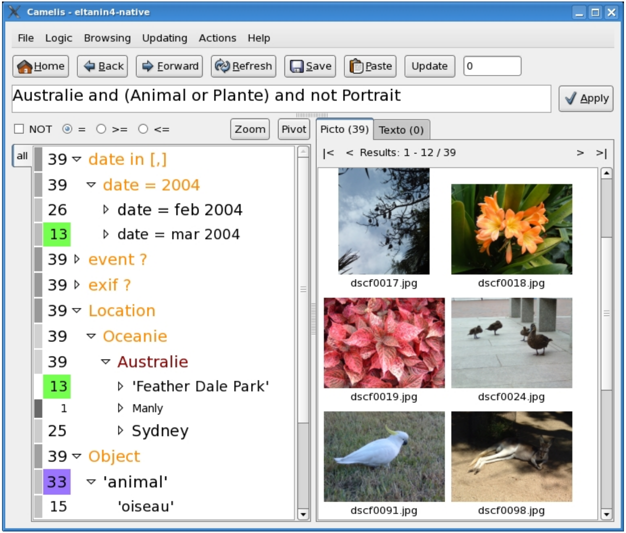
LIS = Logical Information Systems [original idea of Olivier Ridoux]

- tight combination of querying and navigation
  - navigation places are defined by queries
  - navigation links are query refinements
  - navigation never leads to a dead-end (*safeness*)
- logical representation and automated reasoning for object properties and queries (*expressiveness*)
  - `and`, `or`, `not`, date intervals, string patterns, taxonomies, etc.
- based on Formal Concept Analysis (FCA)
  - extended with logics (LCA) [Ferré and Ridoux 2000]
  - the concept lattice is the navigation graph
    - automatically generated and synchronized with data
- a faceted search-like user interface (*ease-of-use*)

Camelis

- implementation of LIS in OCaml
- download:
  `http://www.irisa.fr/LIS/ferre/camelis/`
- 2 interfaces:
  - desktop graphical user interface
  - multi-user web interface: Abilis [Benjamin Sigonneau, Véronique Abily]
    - give it a try at
      `http://ledenez.insa-rennes.fr/abilis/`
- applications:
  - personal data: photos, music, biblio, files, ...
  - collections: biblio, journals and conferences
  - collaborative decision making [Mireille Ducassé]

Camelis: a screenshot



Camelis: a short demo

1. photos of Australia
   - only Sydney, except Sydney
2. photos of all ICFCA conferences
   - only those with people
3. photos of animals and flowers
   - only in Australia, only kangaroos and koalas

Camelis: summary

Camelis enables

- to combine querying and navigation
- to build complex queries by navigating
- to show only relevant navigation links from any query
- to alternate freely navigation and querying in a same search

This tight combination is based on (Logical) Concept Analysis.

Plan

Logical Information Systems (LIS)

The Semantic Web

LIS for the Semantic Web

Conclusion and Future Work

The Need for Relations

From collections of objects to graphs of objects:

- personal data: people and organizations, genealogy
- geographical data: distance, topological relations (e.g., overlaps, contains)
- bioinformatics data: sequences, gene networks, molecules
- linguistic data: discourse structure, parse trees
- software data: execution traces, models (UML)

Need to search, explore, learn from such complex data, with the same LIS properties: safeness, expressiveness, and ease-of-use

Existing Work on Relations

- Semantic Web
  - standard languages for representing, reasoning, querying, and sharing such relational data
  - e.g., RDF (data), RDFS/OWL (ontology), SPARQL (queries)
- Formal Concept Analysis
  - power-context families [Wille 1997]
    separated navigation spaces for objects and relationships
  - Relational Concept Analysis (RCA) [Rouane et al 2007]
    useful for data-mining
- Faceted Search
  e.g., Ontogator, BrowseRDF, SlashFacet [2006]
  - most make use of Semantic Web data
  - a few extend faceted search queries
    - with limited expressiveness compared to SPARQL
    - mostly traversal of relation paths
    - none of disjunction, negation, and circular pattern

What is the Semantic Web?

- also called the Web of Data
- an evolving extension of the WWW (W3C)
- a set of languages, standards, tools to make the web understandable by machines (“semantics”)
- the convergence of research work in knowledge representation, logics, databases, object-oriented modelling: e.g., conceptual graphs, description logics, Prolog, Datalog, relational databases

Semantic Web Languages

- RDF: resource description
- RDFS, OWL, SWRL: complex reasoning
- SPARQL: querying
- SPARQL-Update: updating

RDF: Resource Description Framework

- A base is a set of triples, i.e., an (hyper-)graph
  - nodes are URIs, literals or blank nodes
  - an edge is a triple (subject, predicate, object)
  - the predicate is a property URI, hence a node itself
- Examples of triples:
  - `lis:ferre foaf:name "Sebastien Ferre"`
  - `lis:ferre foaf:birth _:b1`
  - `_:b1 foaf:date "1976-03-19"^^xsd:date`
  - `lis:ferre :affiliation <http://www.univ-rennes1.fr/>`

RDFS: RDF Schema

- extends the RDF vocabulary
- enables the description of taxonomies and simple ER schemas
- RDF(S) classes: `rdfs:Resource`, `rdfs:Literal`, `rdfs:Class`, `rdf:Property`, `rdfs:Datatype`
- RDF(S) properties: `rdf:type`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain`, `rdfs:range`
- implies limited forms of inference
  e.g., `:father rdfs:range :man` and `?X :father ?Y` implies `?Y rdf:type :man`

SPARQL: a Query Language for RDF

An example

```
SELECT DISTINCT ?p ?n ?a
FROM <http://www.irisa.fr/personnel.rdf>
WHERE { ?p a foaf:Person ;
           foaf:name ?n .
        OPTIONAL { ?p ex:age ?a }
        FILTER (! REGEX(?n, "Bob")) }
ORDER BY ASC(?n) LIMIT 10 OFFSET 20}
```

- SPARQL 1.1 adds negation, aggregations, subqueries
- similar expressivity to SQL

From Concept Analysis to Semantic Web

- Formal Concept Analysis (FCA)
  - object → resource
  - attribute → class
  - incidence → property `rdf:type`
  - attribute set → query
- Logical Concept Analysis (LCA)
  - valued attribute → property and resource
  - pattern → SPARQL filter
  - subsumption → transitive properties
- Relational Concept Analysis (RCA)
  - relation → property

Plan

Extension of the LIS approach to relations

Two perspectives:

- Conceptual Navigation: Relational Conceptual Navigation
- Faceted Search: Semantic Faceted Search

with the following results:

1. semantic compatibility with Semantic Web languages
   - RDF as a description language (data)
   - SPARQL as a query language
   - constraint: only binary relations
2. theoretical properties of conceptual navigation
   - *safeness*: no dead-end
   - *expressiveness*: nearly same expressivity as SPARQL
   - *completeness*: every safe query is reachable
3. positive user evaluation (*ease-of-use*)
   - only a few controls added to the user interface

Extension of the LIS approach to relations

Two perspectives:

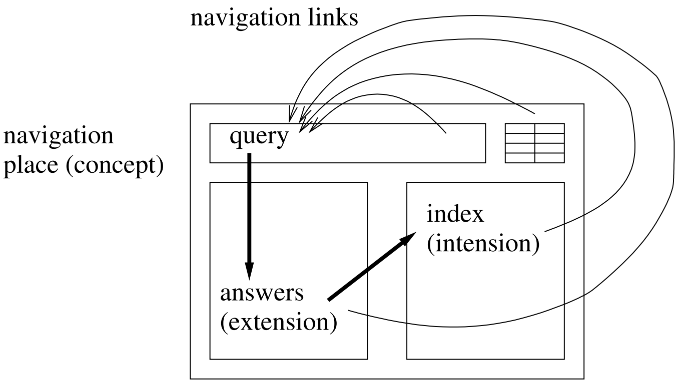
- Conceptual Navigation: Relational Conceptual Navigation
- Faceted Search: Semantic Faceted Search

with the following results:

1. semantic compatibility with Semantic Web languages
   - RDF as a description language (data)
   - SPARQL as a query language
   - constraint: only binary relations
2. theoretical properties of conceptual navigation
   - *safeness*: no dead-end
   - *expressiveness*: nearly same expressivity as SPARQL
   - *completeness*: every safe query is reachable
3. positive user evaluation (*ease-of-use*)
   - only a few controls added to the user interface

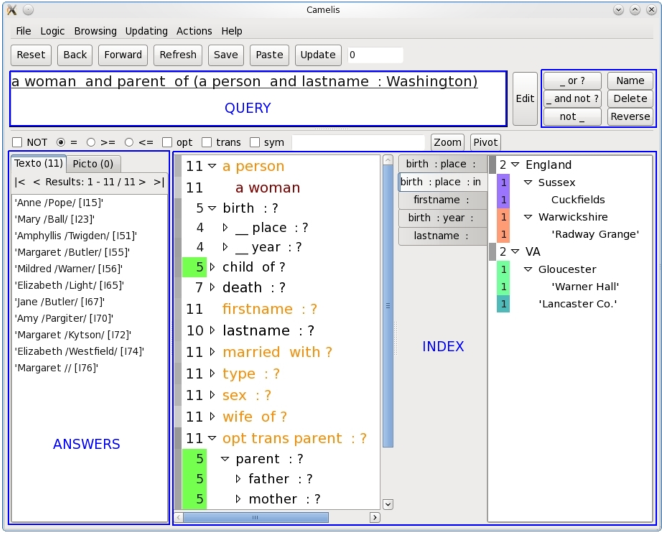
Overview

A schema for the navigation graph and the user interface.



Navigation links are query transformations

User Interface: a Screenshot of Camelis 2



The Query Language LISQL

- has the semantics and expressivity of SPARQL
  - where answers are restricted to sets instead of tables
  - `SELECT <var> WHERE <graph pattern>`
- has a syntax similar to Description Logics
  - it is more concise
  - it avoids most variables
  - query = complex class + focus (underlined subclass)

LISQL: Complex Classes

complex class C	first-order formula $C(x)$	example
individual	$x = individual$	John
$?v$	$x = v$	?X
?	$\top$	?
a *class*	$class(x)$	a person
$P\ C_1$	$\exists y.(P(x,y) \wedge C_1(y))$	father : John
C_1 and C_2	$C_1(x) \wedge C_2(x)$	a man and ?X
C_1 or C_2	$C_1(x) \vee C_2(x)$	John or Paul
not C_1	$\neg C_1(x)$	not father : ?

LISQL: Complex Properties

complex property P	first-order formula $P(x, y)$	example
property :	$pr(x, y)$	father :
property of	$pr(y, x)$	mother of
property with	$pr(x, y) \vee pr(y, x)$	married with
trans P	$P^+(x, y)$	trans parent :
opt P	$P(x, y) \vee x = y$	opt trans parent of

Syntactic sugar:

- `in` = `opt trans part of`
- `contains` = `opt trans part :`

LISQL: an Example

- `a person and birth : (year : (1601 or 1649) and place : (?X and in England)) and father : birth : place : not ?X`
  - *who was born in 1601 or 1649 at some place X in England, and has a father born at a place that is not X*
- same complex class with focus on `?X and in England`
  - *at which place (X) in England, a person was born in 1601 or 1649, and the father of this person was not born*
- equivalent SPARQL query (7 variables)

```
SELECT ?x WHERE { ?p a person. ?p birth
?b. ?b year ?y FILTER (?y=1601 ||
?y=1649). ?b place ?x. ?x in England. ?p
father ?f. ?f birth ?fb. ?fb place ?fl
FILTER ?fl != ?x }
```

LISQL: an Example

- `a person and birth : (year : (1601 or 1649) and place : (?X and in England)) and father : birth : place : not ?X`
  - *who was born in 1601 or 1649 at some place X in England, and has a father born at a place that is not X*
- same complex class with focus on `?X and in England`
  - *at which place (X) in England, a person was born in 1601 or 1649, and the father of this person was not born*
- equivalent SPARQL query (7 variables)

```
SELECT ?x WHERE { ?p a person. ?p birth
?b. ?b year ?y FILTER (?y=1601 ||
?y=1649). ?b place ?x. ?x in England. ?p
father ?f. ?f birth ?fb. ?fb place ?fl
FILTER ?fl != ?x }
```

LISQL: an Example

- `a person and birth : (year : (1601 or 1649) and place : (?X and in England)) and father : birth : place : not ?X`
  - *who was born in 1601 or 1649 at some place X in England, and has a father born at a place that is not X*
- same complex class with focus on `?X and in England`
  - *at which place (X) in England, a person was born in 1601 or 1649, and the father of this person was not born*
- equivalent SPARQL query (7 variables)

```
SELECT ?x WHERE { ?p a person. ?p birth
?b. ?b year ?y FILTER (?y=1601 ||
?y=1649). ?b place ?x. ?x in England. ?p
father ?f. ?f birth ?fb. ?fb place ?fl
FILTER ?fl != ?x }
```

The Index

- used as a summary of the query answers
- a set of complex classes
- organized as a generalization tree:
  - `'George Washington'`
  - `?X`
  - `a man`
  - `mother : ?`
    - `mother : 'Mary Ball'`
    - `mother : a woman`
    - `mother : birth : ?`
    - `mother : father : ?`
  - ...
- expanded on demand, because of its recursive definition,
- restricted to relevant elements (*safeness*)
  - $ext(q \text{ and } C) \neq \emptyset$

The Navigation Links

Navigation links are query transformations

- they all apply at the current focus
- and: $q' := q \text{ and } C$, for each C taken in the index
  - already present in LIS and faceted search
- or: $q' := q \text{ or } \underline{?}$
- and not: $q' := q \text{ and not } \underline{?}$
- name: $q' := q \text{ and } ?v$, for one fresh variable v
- focus change: for each focus of the query

The Navigation Links

Navigation links are query refinements

1. ?
2. a man
3. a man and name: ?
4. a man and name: ?X
5. a man and name: (?X and 'Georges')
6. a man and name: (?X and ('Georges' or ?))
7. a man and name: (?X and ('Georges' or 'John'))
8. a man and name: (?X and ('Georges' or 'John'))
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

The Navigation Links

Navigation links are query refinements

1. ?
   - **and** a man (index element)
2. a man
3. a man and name: ?
4. a man and name: ?X
5. a man and name: (?X and 'Georges')
6. a man and name: (?X and ('Georges' or ?))
7. a man and name: (?X and ('Georges' or 'John'))
8. a man and name: (?X and ('Georges' or 'John'))
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

The Navigation Links

Navigation links are query refinements

1. ?
2. a man
   - **cross** name: ? (index element)
3. a man and name: ?
4. a man and name: ?X
5. a man and name: (?X and 'Georges')
6. a man and name: (?X and ('Georges' or ?))
7. a man and name: (?X and ('Georges' or 'John'))
8. a man and name: (?X and ('Georges' or 'John'))
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

The Navigation Links

Navigation links are query refinements

1. ?
2. a man
3. a man and name: ?
   - **name** the focus (button)
4. a man and name: ?X
5. a man and name: (?X and 'Georges')
6. a man and name: (?X and ('Georges' or ?))
7. a man and name: (?X and ('Georges' or 'John'))
8. a man and name: (?X and ('Georges' or 'John'))
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

The Navigation Links

Navigation links are query refinements

1. ?
2. a man
3. a man and name: ?
4. a man and name: ?X
   - **and** 'Georges' (extension element)
5. a man and name: (?X and 'Georges')
6. a man and name: (?X and ('Georges' or ?))
7. a man and name: (?X and ('Georges' or 'John'))
8. a man and name: (?X and ('Georges' or 'John'))
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

The Navigation Links

Navigation links are query refinements

1. ?
2. a man
3. a man and name: ?
4. a man and name: ?X
5. a man and name: (?X and 'Georges')
    - **or** (button)
6. a man and name: (?X and ('Georges' or ?))
7. a man and name: (?X and ('Georges' or 'John'))
8. a man and name: (?X and ('Georges' or 'John'))
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

The Navigation Links

Navigation links are query refinements

1. ?
2. a man
3. a man and name: ?
4. a man and name: ?X
5. a man and name: (?X and 'Georges')
6. a man and name: (?X and ('Georges' or ?))
    - **and** 'John' (extension element)
7. a man and name: (?X and ('Georges' or 'John'))
8. a man and name: (?X and ('Georges' or 'John'))
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

The Navigation Links

Navigation links are query refinements

1. ?
2. a man
3. a man and name: ?
4. a man and name: ?X
5. a man and name: (?X and 'Georges')
6. a man and name: (?X and ('Georges' or ?))
7. a man and name: (?X and ('Georges' or 'John'))
    - **focus change** (query element)
8. a man and name: (?X and ('Georges' or 'John'))
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

The Navigation Links

Navigation links are query refinements

1. ?
2. a man
3. a man and name: ?
4. a man and name: ?X
5. a man and name: (?X and 'Georges')
6. a man and name: (?X and ('Georges' or ?))
7. a man and name: (?X and ('Georges' or 'John'))
8. a man and name: (?X and ('Georges' or 'John'))
   - **cross** father: name: ? (index element)
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

The Navigation Links

Navigation links are query refinements

1. ?
2. a man
3. a man and name: ?
4. a man and name: ?X
5. a man and name: (?X and 'Georges')
6. a man and name: (?X and ('Georges' or ?))
7. a man and name: (?X and ('Georges' or 'John'))
8. a man and name: (?X and ('Georges' or 'John'))
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
   - **and** ?X (index element)
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

The Navigation Links

Navigation links are query refinements

1. ?
2. a man
3. a man and name: ?
4. a man and name: ?X
5. a man and name: (?X and 'Georges')
6. a man and name: (?X and ('Georges' or ?))
7. a man and name: (?X and ('Georges' or 'John'))
8. a man and name: (?X and ('Georges' or 'John'))
9. a man and name: (?X and ('Georges' or 'John')) and father: name: ?
10. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X
    - **focus change** (query element)
11. a man and name: (?X and ('Georges' or 'John')) and father: name: ?X

Theoretical Properties of the Navigation Graph

Theorem (safeness)

For every navigation path going from $?$ *to q, ext*$(q) \neq \emptyset$.

- there is no dead-end
- frustation of empty results is avoided

Theorem (completeness)

For every query q that has answers at each focus, there is a finite navigation path going from $?$ *to q.*

- there is no unreachable query
  - users never need to edit the query
  - even stronger result than in non-relational LIS

User Evaluation

- 20 students (from IFSIC and INSA Rennes)
- dataset: genealogy of George Washington (70 persons)
- 18 questions of increasing difficulty
  - property chains, negation, disjunction, variables
  - the number of navigation steps ranges from 0 to 12
- results
  - all answered correctly to $\geq$11/18 questions
  - 8/20 answered correcty to $\geq$16/18 questions
  - the average time spent on the test is 40min ([21,58]min)
  - for each category of question, $\geq$18/20 answered correctly to at least one question of the category
  - for most categories, success rate and response time improve on 2nd and 3rd queries

Some Questions of the Study

1. How many women are named Mary ?
   `a woman and name : Mary`
2. Which man was born in 1659 ?
   `a man and birth : year : 1659`
3. Which man is married with a woman born in 1708 ?
   `a man and married with (a woman and birth : year : 1708)`
4. Which women have for mother Jane Butler or Mary Ball ?
   `a woman and mother : ('Jane' or 'Mary')`
5. How many women have a mother whose death's place is not Warner Hall ? `a woman and mother : death : place : not 'Warner Hall'`
6. Who died in the same area where they were born ?
   `a person and death : place : in ?X and birth : place : in ?X`

Plan

Conclusion

- We have shown that the LIS approach, and therefore both Conceptual Navigation and Faceted Search,
  - can be used on RDF graphs
  - with an expressive SPARQL-like query language
  - where users can entirely rely on navigation
  - without ever falling in dead-ends
  - after a short training stage
- We have implemented this approach:
  - first but partially in GEOLIS [Olivier Bedel]:
    plus spatial relations (distance, topology)
  - then completely in CAMELIS 2 (demo on demand)

Future Work

A common framework for future works in the LIS team:

- expressiveness: *n*-ary relations, implicit relations (e.g., spatial relations), grouping and aggregation [PhD P. Allard]
- query syntax: closer to natural language, multilingual [A. Foret]
- usability: evaluation and improvement for lambda users [visiting PhD L. Spagnolo]
- knowledge edition: combining an expressive description/update language (e.g., SPARUL) and an interactive construction process [PhD A. Hermann]