Generating Surrogates to Make the Semantic Web Intelligible to End-Users

Fabien Gandon

Generating Surrogates to Make the Semantic Web Intelligible to End-Users

Fabien Gandon
INRIA ACACIA, 2004 rt des Lucioles, BP93, 06902 Sophia Antipolis, France
Fabien.Gandon@sophia.inria.fr

Abstract

The semantic Web is a vision of a Web augmented with formalized knowledge annotating it. Currently, there is a huge gap between the conceptual structures underlying the semantic Web and the final rendering of a user-interface enabling an end-user to peruse or act on part of it. We describe an approach we experimented to automate part of the process of generating representations for concepts mobilized in the semantic Web. We reuse the notion of surrogate from information retrieval and we show that surrogate patterns tend to be close to the patterns of identity conditions used in ontology engineering. From this observation we propose and discuss a mechanism to derive surrogate templates from structures found in ontologies and rules.

1. Introduction

The semantic Web can be summarized as a Web augmented with formalized knowledge annotating it and that applications can use through inferences for their tasks or to help users navigate, search, modify, etc.

To interact with this semantic Web we need interfaces that make it intelligible to end-users. The problem of intelligibility is different from the one of interoperability. Intelligibility is not ensured by working at the semantic level. Pieces of knowledge are manipulated and combined, and the intelligibility of the results is not ensured through transformations [3].

Currently, there is a huge gap between the conceptual structures underlying the semantic web and the final rendering of a user-interface enabling an end-user to peruse or act on part of it. Most of the time user-interface designers implement in ad hoc ways the transformation from their internal data structures to the interface representations. This is no longer feasible when the data structures, their schemata, transformations, etc. are changing and propagating through networks. In other words this is not possible in the semantic Web. Interfaces will have to be, at least partly, dynamically generated and rendered for every structure coming in contact with the users. This paper describes an experiment we carried out to automate part of the process of generating representations for the concepts mobilized in the semantic web. The idea is to use identity conditions to initialize visualization patterns and then have a human intervention to distinguish between "good" and "bad" identity conditions; we focus here on the initialization phase.

In the next section we introduce the notion of surrogates first through a motivating scenario and then through a survey of its two meanings in information retrieval. In section 3 we first discuss the link that exists between ontologies and interfaces due to the needs for semiotic logics of representations. We then compare the features of a surrogate and the notion of identity conditions in ontologies. Finally we propose a mechanism to derive candidate surrogates from semantic web rules that already exist and are used for information integration. In the last section we summarize the different implementations we tested to automate the generation of surrogates. The first part briefly summarizes and criticizes early attempts. The second part details our implementation and test of the latest approach based on equivalence rules.

2. Surrogate problem in information retrieval

2.1. Query Matching vs. Intelligible Results

Projects designed in our team are based on CORESE [2] to handle semantic web graphs. CORESE interprets RDF [17] in the Conceptual Graphs (CG) [14] and uses their inferences. It is a semantic search engine that loads RDFS [18] schemata and RDF descriptions, applies production rules and answers queries. CORESE has a query language [2] based on the RDF/XML syntax [19] or a triple-based syntax introduced more recently and close to the SPARQL [20] syntax under discussion at W3C. This triple syntax is a cousin of the N3[1] syntax augmented with variables (prefixed with "`?`") and operators such as "`option`" to declare a triple as optional, "`join`" to group results by values, "`not`" for negation as failure, "`~`" for the inclusion of literal values, etc.

As an example, figure 1 shows a query looking for instances of documents (`?d` in the triple of line 1) with at least one author (`?a` in line 2), a person (line 3) whose name (`?n` line 4) contains the literal "`aiman`" (line 5). The triple query is interpreted as a CG query and is processed by a CG projection i.e. a projection of the query-graph on the graphs of the ontologies and annotations. Figure 2 shows an extract of the answer rendered by an XSLT [22] stylesheet transforming the RDF/XML syntax of the result into an XHTML [21] page. The library of generic XSLT templates we developed uses the RDF/XML files of the ontologies and the results from CORESE to display instances of classes and properties using their ontological labels; this also allows us to handle multilingual interfaces as in the CoMMA project [4].

```
01   ?d      rdf:type        ex:Document
02   ?d      ex:author       ?a
03   ?a      rdf:type        ex:Person
04   ?a      ex:name         ?n
05   ?n      ~               "aiman"
```

Figure 1. Query to retrieve documents.

This simple query-answer interaction is omnipresent in all the projects we participated in so far [4,5,7,11,12]. If we look at the result in figure 2, the answer is perfectly correct from an ontological stand point. However if it is shown to users as it is here, then the only knowledge they can get from this answer is that there exists a novel written by a man called "Gaiman" and an article written by a Woman called "Aiman-Smith"; users would most probably have appreciated to have the title of the novel, the first name of the author, etc. if they were available. One could argue that the users should have included them in the query, but this requirement is so natural to us that these properties should have been automatically included in the query-answer process. In fact we argue that *the usefulness of these properties is ontological knowledge*.

- Novel (http://isbn.nu/0380789035)
  author Man (http://www.neilgaiman.com/)
  name: Gaiman
- Article (http://www.asee.org/jee/papers/content.cfm?name=STEPHEN-209.pdf)
  author Woman (http://www.mgt.ncsu.edu/faculty/busmgt/laiman-smith.html)
  name: Aiman-Smith

Figure 2. Extract of answers to the query in figure 1.

2.2. Duality and Problematics of Surrogates

Indexing resources consists of scanning the set of these resources and building representative surrogates (e.g. a vector of terms) for each one of them. Surrogates for indexing may be as simple as collecting some words of the document or as complex as a natural language analysis of its content resulting in semantic annotations. Surrogates are not limited to representing the content of the resources, but can also include metadata (e.g. editor, author, ISBN, etc.) and more generally external properties of the resource (e.g. number of hyperlinks pointing toward it, etc.). The choice of the surrogate influences to a great extent the whole information systems and motivated a lot of research in the field of information retrieval [8,10]. Thus, in information systems, *the first use of surrogates of information resources is to provide a highly synthetic and representative structure reduced to those features of the resource that are relevant for the intended processing*.

Besides the efficiency of the retrieval algorithm, a major factor in users' acceptance of a system is the user-interface through which they interact with it. Search results in a classical Web search engine usually take the form of a list of pages with for each result some text extracted from the selected page and justifying the selection. In addition, other information may be given (e.g. URL, date of indexing, snapshot) and the results are ranked according to their estimated relevance to the query. To represent this set of selected resources, the system uses a second kind of surrogates [8,10] : *the second use of information resource surrogates is to provide a highly synthetic and representative structure reduced to those features relevant for the users to identify the resources and their position in the set of results*.

An identifier is always present in both types of surrogates, but it is not enough for users since it is usually a system identifier such as a database primary key or a URI (e.g. `http://st5.com/ReportV278.htm#C12`) which barely provides any information about the resource and is usually only used in operations such as joints. Thus the second kind of surrogates requires information such as: title, focused extract, previews, etc. From the choice of the surrogate depends the ability of the system to propose views that organize the results efficiently.

In the case of a semantic search engine (a search engine exploiting formal knowledge representation) where queries and results are only limited by the available ontologies, the problem of finding a generic mechanism to build these surrogates is an open one. One of the difficulties is that *the relevance of the features used to build a surrogate is domain-dependent*.

3. Surrogate patterns from ontologies

3.1. Ontology and Interfaces

Ontologies provide the semantic grounding for communication and as such, are at the frontier between the conceptualization of the system and the one of the users. Thus ontologies need to be understandable both to humans and to machines, otherwise they can no longer play this pivotal role and they are no longer usable, maintainable, etc. Moreover, the whole internal conceptual structures should never be shown to end-users; not only because their logical face is abstruse, but also because, as humans, we do not mobilize our whole conceptualizations each time we communicate, think, act, etc.: we focus. Therefore, a system must not impose to users to handle a whole ontology each time they have an interaction: we focus and user-interfaces must focus with us.

User-interfaces have the unenviable role of bridging the gap between explicit conceptualizations captured in ontologies and day-to-day use of signs to denote concepts with unavoidable ambiguity and fuzziness. The very simple fact of choosing labels in an ontology introduces it in the field of semiotics; user-interface and ontological problems must be tackled in parallel. This brings us back to the problem of choosing surrogates for visualization. This aspect was mostly overlooked in literature while it is vital to support the mechanisms of interpretation associated to our models, our inferences and their results. For example, to represent the instance of a person, it makes sense to build a surrogate including the first name and the surname of the person but the age, height and address may not be useful unless explicitly required by a scenario.

Distinguishing between key and non key properties is a scenario-dependent task.

3.2. Identity Conditions and Surrogates

The notion of unique identifier (e.g., URI, ISBN) is artificial for us. We, as human, often require several key attributes to identify an instance, sometimes even running the risk of confusion when unicity is not 100% ensured by the features of the surrogate. Between different systems, in information integration scenarios or for annotation with different points of view, the proof for equivalence of instances and the merging of instances usually uses these identity surrogates. This problem is well know, for instance, in systems trying to detect acquaintance networks e.g.: to detect that the author of a paper is the same person as the author of a given web page, key attributes of the identity are usually chosen, weighted and combined to see if the result exceeds a confidence threshold.

In fact, the problem of the representation of instances is tightly linked to the notion of identity that is: the ontological problem of distinguishing a specific instance of a certain category from other instances by means of a characteristic property, which is unique for it. Guarino and Welty [9] suggested that the identity relies on the existence of a rigid property i.e. a property that necessarily holds for all the instances of a concept. They gave the example of "being a Person" which is rigid since an instance *x* being a Person cannot cease to be a Person unless the instance ceases to be. On the other hand, "being a Student" is anti-rigid since all the instances of Student have the ability to cease to be a Student without ceasing to be. An identity condition for a property ϕ is defined [9] as a relation ρ satisfying (1) and a concept α is semantically rigid if it satisfies (2), the *nec* operator being the modal necessity operator.

$$\phi(x) \wedge \phi(y) \rightarrow (\rho(x,y) \leftrightarrow x = y) \quad \textbf{(1)}$$

$$\forall x \ (x \in \alpha \supset \text{nec} \ (x \in \alpha)) \quad \textbf{(2)}$$

Rather than a single characteristic property, the identity condition ρ can be a group of properties and complex tests that establish the identity, for instance the family name plus the first name plus the date and place of birth. This suggests that there is an interesting overlap between the set of properties used in the identity condition of a concept and the properties used in its surrogate.

In [9], authors also distinguish between supplying an identity condition and simply carrying an identity condition: non-rigid properties can only carry their identity conditions, inheriting those supplied by their subsuming rigid properties. Just like identity conditions are inherited along the hierarchy, key surrogate properties are inherited along the hierarchy e.g.: if the first name is relevant for the surrogate of a person then it is also relevant for the surrogate of a woman.

Let us extend the notion of identity condition with the notion of minimal identity condition: an identity condition is minimal if the identity condition ρ satisfying (1) relies on a set of tests that does not include the set of test defining another identity condition. This is summarized in (3), t_n are unitary tests on properties of x and y.

$$\text{Minimal}(\rho) \leftrightarrow [\rho(x,y) \leftrightarrow \wedge_i t_i(x,y)] \wedge [\neg\exists \rho' ; \text{IC}(\rho') \wedge [\rho'(x,y) \leftrightarrow \wedge_j t_j(x,y)] \wedge \{t_j\} \subset \{t_i\}] \quad \textbf{(3)}$$

This is to avoid considering, for instance, an identity condition using the name, first name, sex, date and place of birth, when another identity condition does not use the sex. Based on this, we made the assumption (4) that the set of properties used by a minimal identity condition can be used to build a candidate surrogate.

$$\text{Minimal}(\rho) \wedge [[\rho(x,y) \leftrightarrow \wedge_i t_i(x,y)] \rightarrow \exists S; \text{Surrogate}(S) \wedge S \equiv \{\text{property } p_j ; p_j \text{ used in test } t_i(x,y)\}] \quad \textbf{(4)}$$

Of course, at this stage, the problem is to find a source for these identity conditions and the properties they are based on. One option is to use the structures generated by a tools supporting such meta-modeling. However we wanted to find these properties in the structures we have at hand in our projects; one of them is the base of rules.

3.3. Extracting Surrogates from Rules

CORESE has an inference engine [2] based on forward-chaining production rules. The rules apply on the base of Conceptual Graphs (CG) and can enrich a graph joining their conclusions to it. For example, the CG rule in figure 3 states that if a person `?m` is the head of team `?t` which has person `?p` has a member (line 1), then person `?m` manages person `?p` (line 2). The rules are applied once the annotations are loaded and before the query processing occurs. Hence, annotations are augmented by rules.

```
01  IF  [Person: ?m]-(head)-[Team: ?t]-
    (hasMember)-[Person: ?p]
02  THEN[Person: ?m]-(manage)-[Person: ?p]
```

Figure 3. Rule propagating the "manage" property.

OWL [16] now includes primitives to state the equivalence of two resources. Therefore in information integration, production rules can be used to automate the detection of equivalences between resources and produce OWL equivalence statements. Figure 4 shows a rule giving an example of using the name (lines 1 and 4), first name (lines 2 and 5) and date of birth (lines 3 and 6) of two instances of person to find equivalences between different instances representing the same person. Such rules encode scripts to detect the equivalence of identity: their consequent is the assertion of an equivalence and their premise expresses tests often derived from an identity condition as defined in the previous section. It is important to stress that the only reason why we are analyzing rules to suggest surrogates, is because they are already available in information integration systems; if we where using other frameworks with more expressive ontology descriptions we could use those instead.

```
01  IF   [Person: ?p1]->(name)->?n
02                     ->(firstname)->?f
03                     ->(birthdate)->?d
04  AND  [Person: ?p2]->(name)->?n
05                     ->(firstname)->?f
06                     ->(birthdate)->?d
07  THEN[Person:?p1]->(equivalent)->[Person: ?p2]
```

Figure 4. Rule to find identical instances of "person".

Therefore our idea was to use the premise of such rules to derive a surrogate for the instances of this type.

If there exists a rule (**5**)
$\wedge_i\{ t_i(p_j(x),p_j(y)), t_i \text{ being a test}\} \wedge type_t(x) \wedge type_t(y) \Rightarrow x \equiv y$
Then a possible surrogate for *x* when $type_t(x)$ holds is
$\cup_j\{\text{property } p_j(x)\}$

For instance from the rule in figure 4 and the formula (5) one can conclude that the properties `name`, `firstname` and `birthdate` can be used to provide a template for surrogates of instances of the type `person` i.e. it makes sense to use these properties and their value to denote an instance of the type person.

A second kind of rules of interest when building these templates for surrogates is the kind of rules encoding sufficient conditions of concept definitions. A special case of this is the external dependence: as defined in [9] a property φ is externally dependent on a property ψ if, for all its instances x, necessarily some instance y of ψ must exist, which is neither a part nor a constituent of x. Authors give the example[9] of `parent` being externally dependent on `child` (one cannot be a parent without having a child), but `person` is not externally dependent on `heart` nor on `body` (because any person has a heart as a part and is constituted of a body). Let us consider the axiom (6) defining the concept of `president`. The implication corresponding to the sufficient condition of this definition can be represented as the rule in figure 5.

$$\text{president}(p) \Leftrightarrow \text{person}(p) \wedge \exists c\ \text{country}(c) \wedge \text{govern}(p,c) \quad (\mathbf{6})$$

```
01  IF   [Person:   ?p]-(govern)-[Country: ?c]
02  THEN [President:?p]
```

Figure 5. Rule defining the concept of President.

From this we can derive that it makes sense to use this property and its value to represent an instance of the type `president`. (in (7)).

If there exists a rule (**7**)
$\wedge_i\{t_i(p_j(x)), t_i \text{ being a test }\} \wedge type_{t1}(x) \Rightarrow type_{t2}(x)$
Then a possible surrogate for *x* when $type_{t2}(x)$ holds is
$\cup_j \{\text{property } p_j(x)\}$

Since different rules generate different templates for surrogates and since templates are inherited along the hierarchy, we need to combine them. As shown by (8) the heuristic we use is to make the union of all the candidate templates that apply to a type and super-types to build a maximal candidate template.

$$\text{Max Surrogate}(type_t) = \cup_j \{\text{Surrogate}(type_{t'}) ; type_t \leq type_{t'}\} \quad (\mathbf{8})$$

Applying (8) to our example on of the type `president` we derive that a maximal template for this type includes: `name`, `firstname`, `birthdate` and `govern`.

4. Automating the generation of surrogates

4.1. Earlier attempts using RDFS/OWL schema

The very first projects in which we used the CORESE search engine required the users to complete their queries with optional parts they wanted to see in the answer i.e. the query would not fail if these parts could not be retrieved but if they were available then they would be displayed.

The problem is that the burden was on the users and moreover this was possible only for queries triggered by a human not for queries automatically generated by the system. In addition, evaluations showed that users were expecting the system to volunteer additional information when available. Therefore, we started to investigate ways to automate the generation of the list of properties to be added to queries as optional parts.

Our first attempt was to systematically add all the properties that could be found. Figure 6 shows what the initial query of figure 1 looks like in that case. The two additional lines respectively request all the available properties about the document (line 06) and about the author (line 07). The operator "`option`" qualifies these two triples as optional. The property "`cos:Property`" subsumes all the properties and is automatically created by CORESE; the exact property names are given in the result.

```
01  ?d      rdf:type          ex:Document
02  ?d      ex:author         ?a
03  ?a      rdf:type          ex:Person
04  ?a      ex:name           ?n
05  ?n      ~                          "aiman"
06  ?d      option::cos:Property   ?p1
07  ?a      option::cos:Property   ?p2
```

Figure 6. Query for documents and any additional property

The first problem encountered was that this approach may not be sufficient: sometime the query comes back with properties and the URI of the resource they point to, but not its properties and this was not helpful at all. This first point was solved using holophrasting techniques allowing the users to request additional information on a leaf of the answer by clicking on a small icon. The second and most important problem we encountered was that this approach may retrieve too many properties. In fact it generates a lot of noise in the answer e.g. a person was displayed with all the documents it had written making the result awfully long and not user-friendly.

Since the main drawback of the first attempt was that it generated too much noise and was not efficient, we decided to try to identify relevant properties directly in the ontology. To do so we systematically introduced the top-property `surrogate_property` to subsume existing properties useful for surrogates. For instance, the property `designation` was a sub-property of `surrogate_property` and the parent property of `title`, `name`, `firstname`, etc. Figure 7 shows what the initial query of figure 1 looks like to take advantage of this. The two additional lines respectively request all the available sub-properties of `surrogate_property` for the document (line 06) and the author (line 07).

```
01  ?d      rdf:type          ex:Document
02  ?d      ex:author         ?a
03  ?a      rdf:type          ex:Person
04  ?a      ex:name           ?n
05  ?n      ~                 "aiman"
06  ?d      option::ex:surrogate property ?p1
07  ?a      option::ex:surrogate_property ?p2
```

Figure 7. Query for documents with "surrogate_properties"

The first problem encountered was that some properties (e.g. `date`) are relevant for the surrogates of some concept types (e.g. `document`) while they are not relevant for the surrogates of some other concept types (e.g. an `organization`).

The second problem is that this approach requires an additional burden in the engineering and maintenance of the ontology. Yet in some applications we are still using this technique since it is rapid and results are acceptable.

We also envisaged attaching templates of surrogates to every type in the ontology. We faced the problem of generating these templates and we imagined the technique based on rules as described in the next section in order to initialize these templates.

4.2. Latest Attempt Implemented Using Rules

In section 3.3 we introduced equivalence rules as rules asserting the equivalence of two instances. We showed that they were interesting to generate candidate templates for surrogates and we gave an algorithm to do so.

The rule syntax used in CORESE is based on the RDF-Conceptual Graphs mapping [2]. As a simple graph rule is a Horn clause of the form "`if ConceptualGraph1 exists then assert ConceptualGraph2`" and as an RDF graph can be interpreted as a Conceptual Graph, the syntax of our rules is "`if RDFGraph1 then RDFGraph2`" where $RDFGraph_n$ is the RDF markup for $ConceptualGraph_n$. The syntax for these RDF rules, is based on the RDF/XML syntax with the convention that variables are prefixed by '`?`', and are local to the rule.

Figures 8 to 10 are examples of rules asserting the equivalence of two instances; they have been chosen for their relevance to the query example given at the beginning of this article in figure 1. The rule in figure 8 states that if (line 2) two persons (lines 3 and 8) have the same name (lines 4 and 9), the same first name (lines 5 and 10) and the same date of birth (lines 6 and 11), then (line 14) they are the same person (lines 15-16).

The rule in figure 9 states that if two persons (lines 3 and 6) have the same e-mail (lines 4 and 7) then they are the same person (line 11-12).

The rule in figure 10 states that if two documents (lines 3 and 8) have the same title (lines 4 and 9), the same author (lines 5 and 10) and the same date (lines 6 and 11) then they are the same document (lines 15-16).

Using an XSLT stylesheet, we automatically transform rules asserting an equivalence into surrogate templates. As shown in figure 11 the previous rules were transformed in three surrogate templates: two templates for the class Person (lines 1-5 and 7-9) and one template for the class document (lines 11-15). W added to CORESE the ability to load queries in a knowledge base just like any RDF annotation. Then one can query this base to extract predefined queries including a given pattern.

```
01  <cos:rule>
02    <cos:if>
03      <ex:Person rdf:about="?person1">
04        <ex:name>?name</ex:name>
05        <ex:firstname>?firstname</ex:firstname>
06        <ex:birthdate>?birthdate</ex:birthdate>
07      </ex:Person>
08      <ex:Person rdf:about="?person2">
09        <ex:name>?name</ex:name>
10        <ex:firstname>?firstname</ex:firstname>
11        <ex:birthdate>?birthdate</ex:birthdate>
12      </ex:Person>
13    </cos:if>
14    <cos:then>
15      <ex:Person rdf:about="?person1">
16        <owl:sameAs rdf:resource="?person2" />
17      </ex:Person>
18    </cos:then>
19  </cos:rule>
```

Figure 8. Rule to detect equivalent persons (name & birth).

```
01  <cos:rule>
02    <cos:if>
03      <ex:Person rdf:about="?person1">
04        <ex:email>?email</ex:email>
05      </ex:Person>
06      <ex:Person rdf:about="?person2">
07        <ex:email>?email</ex:email>
08      </ex:Person>
09    </cos:if>
10    <cos:then>
11      <ex:Person rdf:about="?person1">
12        <owl:sameAs rdf:resource="?person2" />
13      </ex:Person>
14    </cos:then>
15  </cos:rule>
```

Figure 9. Rule to detect equivalent persons using emails.

```
01  <cos:rule>
02    <cos:if>
03      <ex:Document rdf:about="?doc1">
04        <ex:title>?title</ex:title>
05        <ex:author rdf:resource="?author" />
06        <ex:date>?date</ex:date>
07      </ex:Document>
08      <ex:Document rdf:about="?doc2">
09        <ex:title>?title</ex:title>
10        <ex:author rdf:resource="?author" />
11        <ex:date>?date</ex:date>
12      </ex:Document>
13    </cos:if>
14    <cos:then>
15      <ex:Document rdf:about="?doc1">
16        <owl:sameAs rdf:resource="?doc2" />
17      </ex:Document>
18    </cos:then>
19  </cos:rule>
```

Figure 10. Rule to detect equivalent documents

```
01  <ex:Person>
02   <ex:name>?name</ex:name>
03   <ex:firstname>?firstname</ex:firstname>
04   <ex:birthdate>?birthdate</ex:birthdate>
05  </ex:Person>
06
07  <ex:Person>
08   <ex:email>?email</ex:email>
09  </ex:Person>
10
11  <ex:Document>
12   <ex:title>?title</ex:title>
13   <ex:author rdf:resource="?author"/>
14   <ex:date>?date</ex:date>
15  </ex:Document>
```

Figure 11. Surrogate templates.

In our case one can load all the surrogate templates and then retrieve for a given concept type the list of templates relevant to this type and its super types then merge them as discussed in section 3.3. Coming back to our initial example of figure 1, one can now extend this query with optional surrogate properties as shown in figure 12, for the class Document (lines 6-8) and for the class Person (lines 9-11).

```
01   ?d      rdf:type                ex:Document
02   ?d      ex:author               ?a
03   ?a      rdf:type                ex:Person
04   ?a      ex:name                 ?n
05   ?n      ~                       "aiman"
06   ?d      option::ex:title        ?p1
07   ?d      option::ex:date         ?p2
08   ?d      option::ex:author       ?p3
09   ?a      option::ex:firstname    ?p4
10   ?a      option::ex:birthdate    ?p5
11   ?a      option::ex:email        ?p6
```

Figure 12. Query of fig. 1 augmented by surrogate template.

A stylesheet transforms the RDF/XML syntax of the result now including the surrogate into an XHTML page; the display is improved as shown in Figure 13.

- Novel (http://isbn.nu/0380789035)
  title: American Gods
  date: April 30, 2002
  author Man (http://www.neilgaiman.com/)
    name: Gaiman
    first name: Neil
- Article(http://www.asee.org/jee/papers/content.cfm?name=STEPHEN-209.pdf)
  title: Algorithm for High Technology Engineering and Management Education
  author Woman(http://www.mgt.ncsu.edu/faculty/busmgt/laiman-smith.html)
    name: Aiman-Smith
    first name: Lynda
    e-mail: lynda_aiman-smith@ncsu.edu

Figure 13. An answer to the query in figure 3.

One can notice that in that case, the date of birth was neither available nor useful and for "Neil Gaiman" the e-mail was not available. But since the surrogate properties are optional, they did not hamper the query solving or the display of the result. As far as the users or services are concerned, they submit the very same query but get results with additional properties that might prove useful when whole or part of the conceptual structure of this result is to be rendered for display, speech generation, etc.

5. Discussion and future work

In this article we focused on a problem we faced in our projects [4,5,7,11,12]: the generation of semiotic representations for conceptual structures such as the annotations, and query results on the semantic Web. We do not assume that automatic generation is the only alternative but in our experience, we found that it is not reasonable to expect that users will enter each and every attributes they prefer to see in their templates for the thousands of classes present in the ontologies of the application they are using. The warming-up time would turn out to be discouraging for initial users and this is why we looked at ways to automate the production of surrogates.

Drawing on the parallel between the patterns of such surrogates and the notion of identity conditions, we proposed and explained a mechanism exploiting the information integration rules to automate the generation of candidate templates for these surrogates. We showed how these candidate templates already improve representation, for instance when viewing the results of a query. The approach focused on generating templates providing the properties to include in a surrogate, regardless of the way it is rendered (text, graphics, speech, etc.). We followed an opportunistic approach where a candidate template is obtained as a union of candidate properties. Further differentiation requires human intervention. Still, like the surrogates, these integration rules are domain and scenario dependent: in a system where `name`, `firstname`, `department` are enough to conclude an equivalence, then these properties may also form a good surrogate.

The quality of the templates can deteriorate when complex equivalence rules are introduced. We believe that the technique is better used in a semi-automatic way, where the system proposes a number of candidate templates that are then reviewed and tuned by the ontologists or refined through selection and learning techniques using the feedbacks from the users. First, we will have to run tests with users to validate the improvement using identity conditions to initialize surrogates templates. Then we could study what follow-up queries are formulated and what relations are requested by users to validate, adapt and learn the surrogates.

One might also notice that subtypes frequently require fewer properties for their surrogates than their supertypes (e.g. Economist John Adam Smith) and that this is in conflict with our description. First of all, the additional properties introduced by a subtype may be excellent identity conditions but very poor properties for visual surrogates e.g. the ISBN property introduced for a book vs. a document is excellent for defining an identity condition but it does not change the fact that the title and the authors remains the best way to describe the book, just like a document in general. Secondly, we do not say that all the selected properties are needed; such a choice is dependent on the scenario, the context, the user's profile and history of use. Our goal was to detect a maximum of these properties that were potentially interesting; then fine tuning as to take place. In addition, our approach and implementation relied on rules because the CORESE platform is based on conceptual graphs and graph rules. In other platforms offering other formalization means or insights in the ontology engineering process, other sources than rules could be exploited to derive surrogate properties from identity conditions. In addition we intend to study the scalability of the approach and the ability to intelligently store surrogates for instance in hierarchies.

When a surrogate includes relations to resources, it is tempting to apply the same technique to these resources. However this is a recursive process possibly leading to retrieving the whole knowledge base. In one of our earlier attempts we used the holophrasting technique to solve this problem introducing widgets that allow the users to request additional information on a leaf of a result. Improving visualization does not mean that we have to ensure 100% unicity in surrogates; holophrasting-like widgets can allow the users to request additional information on a resource and start a disambiguation dialog with the system just like we do when talking to each others. It is much better than having answers aggregating too many details.

Another promising heuristic is to apply the full templates of the surrogates to each node of the query and then only apply the "data type properties" of the templates to the added resources, ignoring the "object type properties". This heuristic coupled with a holophrasting technique seems a good compromise.

Finally our experiments underlined the need for a richer model of the links between the conceptual structures of the semantic web and the semiotic level of the classic web for humans. In the statement of interest [3], the author calls for the involvement of the semioticians in the modeling and inference mechanisms underlying the semantic web, pointing to works on semiotic algebra [6]. This is becoming vital in a pervasive World Wide Web using multi-modal, multi-media devices and growing more and more mobile every day. We need to be able to identify and differentiate between alternative signs and media channel, and link these semiotic alternatives and their logics to the underlying semiotic structures and logics. The question of the link between the ontologies and the semiotic systems [15] they interact with is to be explored thoroughly. Looking even further there is a need to model the combination of semantic web resources and pragmatic web resources to produce semiotic web resources [13].

The generation of the interfaces for the semantic web will be dynamic and will use: the users' profile, the context and history of interactions, semiotic modeling primitives added to our meta-model, signs linked to the primitives of our ontologies, logics of semiotics and surrogate generation, in addition to the conceptual structures to be communicated to the users.

References

[1] T. Berners-Lee, "Getting into RDF & Semantic Web using N3", 2003

[2] O. Corby, R. Dieng-Kuntz, C. Faron-Zucker, "Querying the Semantic Web with the CORESE search engine". In Proc. of the 16th European Conference on Artificial Intelligence, sub-conference PAIS, Valencia, 2004, IOS Press, pp. 705-709.

[3] J. Euzenat, "Towards formal knowledge intelligibility at the semiotic level", In Proc. of Workshop Applied Semiotics: Control Problems, ECAI, 2000, pp. 59–61,

[4] F. Gandon, "Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web", Ph.D. Thesis, 2002

[5] F. Gandon, N.Sadeh, "Semantic Web Technologies to Reconcile Privacy and Context Awareness", Web Semantics Journal, 1(3), 2004

[6] J. Gogen, "An introduction to algebraic semiotics with applications to user interface design", LNCS 1562, 1999

[7] J. Golebiowska, R. Dieng-Kuntz, O. Corby, D. Mousseau, "Building and Exploiting Ontologies for an Automobile Project Memory", Proc. K-CAP, 2001

[8] E. Greengrass, "Information Retrieval: A Survey", 2000 http://www.csee.umbc.edu/cadip/readings/IR.report.120600.book.pdf

[9] N. Guarino, C. Welty, "Towards a methodology for ontology-based model engineering", In Proc. Workshop on Model Engineering, ECOOP, 2000

[10] R. Korfhage, "Information Storage and Retrieval", Wiley & Sons, 1997

[11] N. Lazaric, C. Thomas, "The coordination and codification of knowledge inside a network, or the building of an 'epistemic community': The 'Telecom Valley' case study", Proc. EAEPE: The Information Society - Understanding Its Institutions Interdisciplinarily, 2003

[12] C. Medina-Ramirez, "Semantic Information Retrieval: representing and querying a heterogeneous biological documentary memory", Proc. Avances en bases de datos y recuperación de información ENC, 2003

[13] A. Pietarinen, "The semantic + pragmatic web = the semiotic web", Proc. IADIS International Conference WWW/Internet, IADIS Press, pp. 981–984.

[14] J. Sowa, "Conceptual Structures: Information Processing in Mind and Machine", Addison-Wesley, 1984

[15] J. Sowa, "Ontology, Metadata, and Semiotics, Conceptual Structures: Logical, Linguistic, and Computational Issues", LNAI1867, Springer-Verlag, (2000) 55–81.

[16] W3C Rec.: OWL Web Ontology Language Overview, 2004

[17] W3C Rec.: Resource Description Framework (RDF), 2004

[18] W3C Rec.: RDF Schema, 2004

[19] W3C Rec.: RDF/XML Syntax Specification (Revised), 2004

[20] W3C Draft: SPARQL Query Language for RDF, 2004

[21] W3C Rec.: XHTML 1.0, 2002

[22] W3C Rec.: XSL Transformations (XSLT) V1.0, 1999