University of Crete

Computer Science Department

Interactive Exploration of Fuzzy RDF Knowledge Bases

Nikos Manolis

Master's Thesis

Heraklion, 25 February 2011

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Διαλογική Εξερεύνηση Βάσεων Γνώσεων σε Fuzzy RDF

Εργασία που υποβλήθηκε από τον

Νικόλαο Ε. Μανώλη

ως μερική εκπλήρωση των απαιτήσεων για την απόκτηση

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Νικόλαος Μανώλης, Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Ιωάννης Τζίτζικας, Επίκουρος Καθηγητής, Επόπτης

Δημήτρης Πλεξουσάκης, Καθηγητής, Μέλος

Αναστασία Αναλυτή, Ερευνήτρια, Ινστιτούτο Πληροφορικής ΙΤΕ, Μέλος

Δεκτή:

Άγγελος Μπίλας, Αναπληρωτής Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών
Ηράκλειο, 25 Φεβρουαρίου 2011

Interactive Exploration of Fuzzy RDF Knowledge Bases

Nikos Manolis

Master's Thesis

Computer Science Department, University of Crete

Abstract

In several domains objects descriptions and associations are accompanied by a numeric degree expressing their strength, importance, or having other application specific semantics (e.g. certainty, trust, etc). These degrees can be provided by humans or be the result of automatic processes (like classification, mining or integration).

Users would like to browse and explore such information sources without having to be aware of the employed terminology or the underlying structuring, nor the query language of the source. Furthermore they would like the ability to reach states whose contents are specified by conditions that involve the degrees of the descriptions/associations.

In this work we consider Fuzzy RDF as the representation framework for such "weighted" descriptions/associations, and we propose a novel model for browsing and exploring such sources, which allows formulating complex queries gradually and through plain clicks. Specifically, in order to exploit the fuzzy degrees, the model proposes interval-based transition markers.

The merits of the proposed model is that it defines formally and precisely the state space of an interaction that (a) allows users to locate the objects of interest, or to get overviews, without having to be aware of the terminology nor the query language of the underlying source, and without reaching states with empty results, (b) exploits fuzzy degrees for enhancing the discrimination power of the interaction (without making it complex for the end user), (c) generalizes the main browsing/exploration approaches for plain RDF/S sources (also clarifying issues regarding schema and instance cyclic property paths), (d) is query language independent, and (e) is visualization independent. Finally we

discuss issues concerning the realization of the model over the available query languages, and we report experimental results regarding efficiency.

Supervisor: Yannis Tzitzikas
Assistant Professor

Διαλογική Εξερεύνηση Βάσεων Γνώσεων σε Fuzzy RDF

Νίκος Μανώλης

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών Κρήτης

Περίληψη

Σε πολλά πεδία οι περιγραφές και οι συσχετίσεις των αντικειμένων συνοδεύονται από έναν αριθμητικό βαθμό ο οποίος μπορεί να εκφράζει την ισχύ, τη σημαντικότητα ή να έχει άλλες σημασίες (βεβαιότητα, εμπιστοσύνη κλπ). Αυτοί οι βαθμοί μπορεί να έχουν δοθεί από ανθρώπους ή να είναι αποτέλεσμα αυτοματοποιημένων διαδικασιών (λ.χ. ταξινόμησης, εξόρυξης ή ολοκλήρωσης).

Οι χρήστες θα ήθελαν να πλοηγούνται και να εξερευνούν τέτοιες πηγές πληροφοριών χωρίς να χρειάζεται να γνωρίζουν την ορολογία που χρησιμοποιείται ή την υποκείμενη δόμηση, αλλά ούτε και την ερωτηματική γλώσσα της πηγής. Συνάμα θα ήθελαν να μπορούν να φτάνουν σε καταστάσεις των οποίων τα περιεχόμενα προσδιορίζονται από συνθήκες που εμπλέκουν τους βαθμούς.

Σε αυτήν την εργασία θεωρούμε τη Fuzzy RDF (Fuzzy Resource Description Framework) ως το πλαίσιο αναπαράστασης τέτοιων 'βεβαρημένων' περιγραφών και συσχετίσεων, και προτείνουμε ένα νέο μοντέλο για την πλοήγηση και εξερεύνηση τέτοιων πηγών, το οποίο επιτρέπει τη διατύπωση σύνθετων επερωτήσεων με σταδιακό τρόπο και μέσω απλών κλικ. Συγκεκριμένα, για την εκμετάλλευση των βαθμών, το μοντέλο προσφέρει σημειωτές μεταβάσεων (transition markers) που αντιστοιχούν σε διαστήματα βαθμών.

Τα θετικά του προτεινόμενου μοντέλου είναι ότι ορίζει επακριβώς και τυπικά το χώρο καταστάσεων μιας αλληλεπίδρασης που (α) επιτρέπει στους χρήστες τον εντοπισμό των αντικειμένων ενδιαφέροντος, ή την εποπτεία αυτών, χωρίς να απαιτείται γνώση της ορολογίας ή της ερωτηματικής γλώσσας της πηγής, και αποτρέποντας καταστάσεις με κενό περιεχόμενο (β) αξιοποιεί τους βαθμούς για να αυξήσει τη διακριτική ικανότητα της αλληλεπίδρασης

(διατηρώντας την όμως απλή στη χρήση), (γ) αποτελεί γενίκευση των βασικών προσεγγίσεων πλοήγησης/εξερεύνησης που έχουν προταθεί για κοινή (όχι fuzzy) RDF/S, αποσαφηνίζοντας συνάμα ζητήματα που αφορούν την πλοήγηση σε κυκλικά μονοπάτια, και (δ) είναι ανεξάρτητη ερωτηματικής γλώσσας και τρόπου οπτικοποίησης. Τέλος σχολιάζονται ζητήματα που αφορούν την υλοποίηση του μοντέλου επί των διαφόρων γλωσσών επερωτήσεων που υπάρχουν, και αναφέρονται πειραματικά αποτελέσματα επίδοσης.

Επόπτης καθηγητής: Γιάννης Τζίτζικας
Επίκουρος καθηγητής

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επόπτη καθηγητή μου κ. Ιωάννη Τζίτζικα για την ουσιαστική καθοδήγηση και συμβολή του στην ολοκλήρωση αυτής της εργασίας, καθώς επίσης και για τη δυνατότητα που μου παρείχε να συνεργαστώ με το Ινστιτούτο Πληροφορικής του Ιδρύματος Έρευνας και Τεχνολογίας.

Επιπλέον, θα ήθελα να ευχαριστήσω τους φίλους και συνεργάτες που συνεισέφεραν στην εκπόνηση αυτής της εργασίας. Ιδιαίτερα ευχαριστώ τους Φώντα Φαφούτη, Γιώργο Πλουμάκη, Νίκο Αρμενατζόγλου και Παναγιώτη Παπαδάκο για την πολύπλευρη βοήθειά τους.

Τέλος ευχαριστώ τους γονείς μου, Ευθύμιο και Δήμητρα, για τη συμπαράστασή τους αυτό το διάστημα.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Several studies [75, 58] show that an important subset (over 60%) of web search queries is *recall-oriented*. The objective in such queries is to locate a set of resources and get information about them (e.g. bibliographic survey writing, medical information seeking, travel planning). Furthermore, the fact that large amounts of structured data have been made publicly available (e.g. billions of RDF triples due to Linked Open Data[1]) makes modern information needs even more complex. However, current WSE (Web Search Engines) do not provide adequate support for such needs. In most of them it is assumed that the user knows exactly what he/she searches for and could express it through a text query or through a structured query language such as SPARQL (in case of structured data).

Therefore, there is a need for general purpose methods for guided exploration which do not presuppose knowledge of the underlying vocabulary or query language. Special focus should be given on session-based interaction, as opposed to the state-less query and response interaction of current WSE, for enabling the gradual formulation of complex conditions, and aiding decision making. Although plain web browsers, support sessional browsing, it is however very primitive (just back and forth). End users need more effective and flexible methods that allow them to progressively reach a state that fulfils their needs.

Moreover, in several domains, objects descriptions and associations are accompanied by a numeric degree expressing their strength, importance, or having other application

[1] http://linkeddata.org

specific semantics (e.g. certainty, trust, etc). These degrees can be provided by humans or be the result of automated tasks like classification [53], clustering [12] and data mining [38]. Furthermore in an open environment like the Web, we may have data of various degrees of credibility, as well data which are copies or modifications of other data. As a result, data of the same entity can be erroneous, out-of-date, or inconsistent/conflicting in different data sources. Therefore, even if the primary data are not fuzzy, the integrated data as produced by an information integration system (that contains tasks like web extraction) could be fuzzy.

Regarding information needs again, users would like to reach states whose contents are specified by conditions that involve the degrees of the descriptions/associations. Although several approaches support guided exploration over various information spaces, there is not yet an approach that exploits fuzzy object descriptions/associations for improving the information thinning process.

In this work we consider *Fuzzy RDF* [51, 70], a fuzzy extension for the RDF/S graph-based data model, as the representation framework for such "weighted" descriptions/associations. We propose a novel model for browsing and exploring such sources, which allows formulating complex queries gradually and through plain clicks. Specifically, in order to exploit the fuzzy degrees, the model proposes interval-based transition markers.

The merits of the proposed model is that it defines formally and precisely the state space of an interaction that (a) allows users to locate the objects of interest, or to get overviews, without having to be aware of the terminology nor the query language of the underlying source, and without reaching states with empty results, (b) exploits fuzzy degrees for enhancing the discrimination power of the interaction (without making it complex for the end user), (c) generalizes the main exploration/browsing approaches for plain RDF sources (also clarifying issues regarding schema and instance cyclic property paths), (d) is query language independent, and (e) is visualization independent.

1.1 Organization of the thesis

This thesis is organized as follows:

Chapter 2 introduces aspects and dimensions of the extensive area of *browsing approaches* and it then presents a survey of the most important ones.

Chapter 3 presents a precise and concise model capturing the essentials of *browsing approaches* over RDF/S. It also studies several issues regarding interaction and the particularities of RDF/S model.

Chapter 4 proposes an interval-based extension of transition markers in order to support browsing and exploration over information spaces with fuzzy object descriptions and associations.

Chapter 5 summarizes the results of this thesis and identifies topics that are worth further work and research.

Chapter 2

Browsing Approaches and Related work

This Chapter introduces aspects and dimensions of the extensive area of *browsing approaches* and it then presents a survey of the most important ones. Section 2.1 introduces aspects and criteria required to characterize various approaches. Then Section 2.2 categorizes related works according to these criteria.

2.1 Aspects of the Landscape

This section introduces important aspects and dimensions of the extensive area of *browsing approaches*. Particularly, we examine the following aspects:

• *User goals and information needs.* Each browsing approach could be characterized by the user goals it succeeds to satisfy. User goals could be further analyzed according to user information needs.

• *Characteristics of the underlying information space.* The structuring of the underlying information space is an important aspect since each case requires tackling different difficulties.

• *Configuration.* Some approaches can be applied without requiring any form of specific configuration or application design with respect to the browsable information space. Contrarily, in other approaches, the contents and structuring of the browsable part have

to be explicitly specified.

• *State Space.* In general we can view the interaction as a state space consisting of *states* and *transitions*, therefore we can characterize, or comparatively evaluate, two browsing approaches by comparing their state spaces, e.g. by identifying properties which are satisfied by their state space. We then analyze in more detail these aspects.

2.1.1 Kinds of Information Needs

In general, we can identify *precision-oriented* needs (e.g. find the telephone of a store), and *recall-oriented* needs (e.g. decide which car/vacation package to buy). According to [75] the majority of information needs have *exploratory nature*, are *recall-oriented* (e.g. bibliographic survey writing, medical information seeking, car buying) and aim at *decision making* (based on one or more criteria). Fig. 2.1 illustrates some of the results reported in [14] regarding WS (Web Search) query analysis. We should underline here that in *informational* queries (corresponding to 50% of queries) the goal is to learn something/everything about a topic. Rose and Levinson [58] have more recently shown that over 60% of web search queries are informational. Furthermore a subcategory of *informational* queries, called *undirected* [58] are exploratory in nature and express recall-oriented needs. We should note at this point that current general-purpose WSE (Web Search Engines) do not provide adequate support for such needs (although some prototype WSE, like [55], support it).

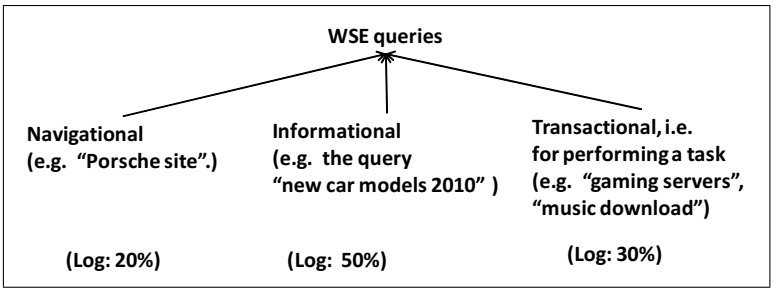

Figure 2.1: Analysis of WSE queries in categories

Fig. 2.2 shows a small taxonomy of user goals. The left side concerns precision-oriented information needs, where the objective is to locate one resource and get information about its attributes or metadata. The right side concerns recall-oriented information needs. Here

the objective is to locate (and get information about) a set of resources. In this category we can distinguish goals that require accessing sets of resources just in *groups*, or in groups accompanied by *count* information for getting an overview of a set of resources. Furthermore we may have goals that require more complex aggregated results like those provided by data warehouses. For instance, [10] proposes aggregations of arithmetic (min, max, average) and Boolean functions over several numeric attributes associated with documents (results of free-text queries). For example, instead of just displaying the number of an author's books in a particular topic, a possible refinement is the average price of each author's books. Moreover, in [19], counts are computed and displayed over combinations (in pairs, triples, etc) of attributes (of grouping criteria in general).

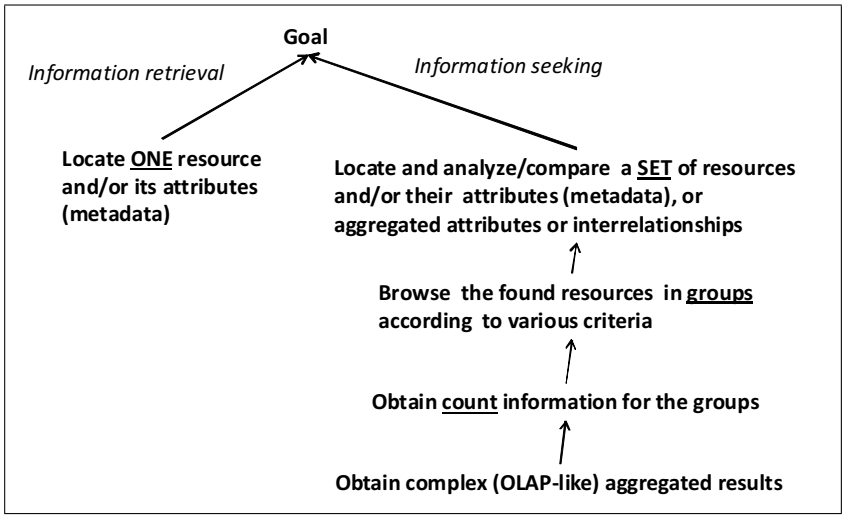

Figure 2.2: Kinds of Information Needs

2.1.2 Information Space

Some browsing approaches are applicable to simple structures (like attribute-value pairs), while others to complex information structures (e.g. OWL-based KBs). Therefore one important aspect is how the underlying information is structured. There are several options and some of them follow. As regards attribute-value pairs, the values could be *flat (e.g.* `name=Yannis`*)* *or* hierarchically organized (e.g. `location=Crete` assuming that `Crete` is a narrower term of `Greece`, and so on). Also, we could have *Set-valued* attributes (e.g. `accessories={ABS, ESP}`). Regarding relational databases, we should note that

they do not have an explicit representation of the conceptual schema.

Furthermore, *multi-entity* or *object-oriented* is a conceptual model for navigation assuming an object-oriented view (e.g. RDF, Linked Open Data). This model describes objects, their attributes and objects associations. We may have attribute-value pairs with flat values and attribute-value pairs with hierarchically-organized values. For instance the RDF/S graph-based data model is such a conceptual model and both alternatives are possible. The values here could be further distinguished in literal values and object values according to the nature of properties (objects attributes or links to other objects).

Additionally, fuzziness could be applied over some of these information spaces in order to extend their expressivity. Several fuzzy extensions for the Entity-Relationship data molel have been studied (from [13], to [25] and [26]). Also, some fuzzy extensions for the RDF/S graph-based data model have recently been proposed in literature [51, 70, 69]. All of these works target to specify the semantics of the *Fuzzy RDF* conceptual model, extending the RDF/S semantics with respect to fuzzy set theory. Compared to plain RDF, a statement in *Fuzzy RDF* can describe simple facts where degrees (in $(0, 1]$) denote the truth value of the statement.

Fig. 2.3 shows the above categories organized hierarchically where an option X is a (direct or indirect) child of an option Y if whatever information can be expressed in Y can also be expressed in X. The value of this diagram is that if a browsing approach is appropriate for an option X then certainly it is appropriate for all options which are parents of X. For instance, a browsing approach appropriate for Fuzzy RDF is also appropriate for plain RDF, as well as for sources formed by attributes with fuzzy and hierarchically organized values.

2.1.3 Configuration

Some approaches can be applied over an information base without requiring any form of configuration or application design with respect to the browsable information space, while others require configuration steps. In the *view-based* approach over a DB or an RDF repository, the contents and structuring of the browsable part should be explicitly specified by the designer. The desired view(s) over an information base could be defined

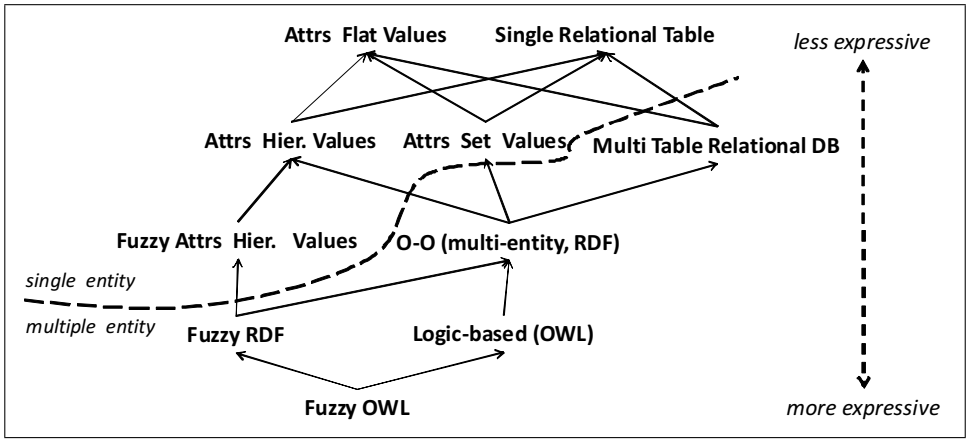

Figure 2.3: Categories of Information Spaces

by using a query language supported by the underlying storage system. Therefore, its structure may be different from that of the underlying information space. For example, we may have an RDF KB, but through a *QL* (query language) we can define its browsable part whose structure may be different from RDF. For example, it can have the form of a relation table, and thus it could be explored using techniques applied over relational tables. Alternatively, the views and the mapping of objects onto views could be defined over a set of ontologies using *logic rules*. For instance, in [37] in order to form a view hierarchy over an RDF KB, classes and the subClassOf relations are projected into a tree. Even though a method like that could be characterized as automatic, it requires a-priori knowledge of the underlying schema(s) in order the required *logic rules* to be defined appropriately. In conclusion, *view-based* approaches can not support navigation over arbitrary datasets.

For the *object-oriented* case (e.g. RDF KBs), Fig. 2.4 shows some categories regarding the applicability (generality), or context dependency, of a *browsing approach*. The left side corresponds to domain specific approaches that follow the *view-based* approach, while the right side to *generic* ones that allow exploration of a dataset without a-priori knowledge of its structure. The latter could be further distinguished to those applicable to triple sets over a single schema and those applicable to heterogeneous datasets without following one fixed schema.

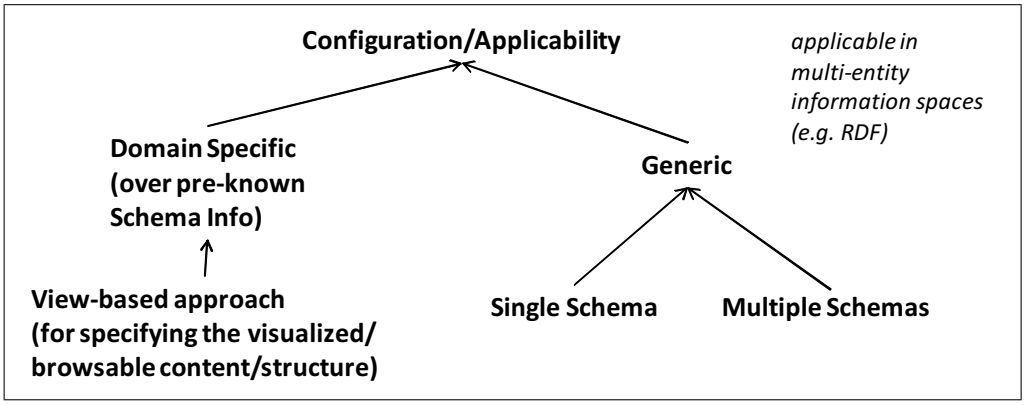

Figure 2.4: Applicability/Genericity of RDF *browsing approaches*

2.1.4 State Space

Independently to the adopted configuration approach, the interaction could be viewed as a state space consisting of *states* and *transitions*. In general we can consider that a *state* has:

- An *extension*: A set of items displayed, e.g. the result set of a web search
- An *intension*: A set of conditions or a query satisfied by the extension
- A *name* or *identity*
- A number of *transitions* each leading to a different state

In addition each state has a *visualization format* for its (a) extension, (b) intention, as well as (c) its transitions (e.g. a tree-control, a list, a table). Note that in some works (e.g. [28]) a state can have more than one visualization formats (and some of these formats may hide some of the transitions of the state). In any case, each *transition* has a clickable *transition marker* signifying the existence of the transition. Subsequently, a *session* is considered to be a sequence of states connected through *transition markers*. Usually these markers are enriched with information regarding the target state. For instance, suppose the user is currently at a state containing all hotels located at `Greece`. From that state there are transitions allowing the user to refine his focus, e.g. there is a transition towards a state that shows only the hotels of `Athens`. The marker of that transition is an indication of the extra condition that will be added to the intention of the

current state (e.g. `location=Athens` will be added), as well as the *size* of the extension of the new state (e.g. the count of hotels located in `Athens` is `80`). This is why the transition markers that correspond to refinements offer a short of *synopsis* or *summarization* of the current extension (a term used in [61] and [22]). Furthermore, and before clicking on `Athens` the user can see that there are more refinements, such as `Historical Center` and `Olympic Stadium`. This means that the user can inspect transitions which are two or more steps away in the state space. The other way around, the transition markers of the transitions that allow the user to move to a broader focus (e.g. from `Olympic Stadium` to `Athens` or to `Europe`), allow the user to realize the *context* of his current focus.

Fig. 2.5 shows the aforementioned structural elements in the form of a UML class diagram (only the important multiplicities of the depicted associations are specified).

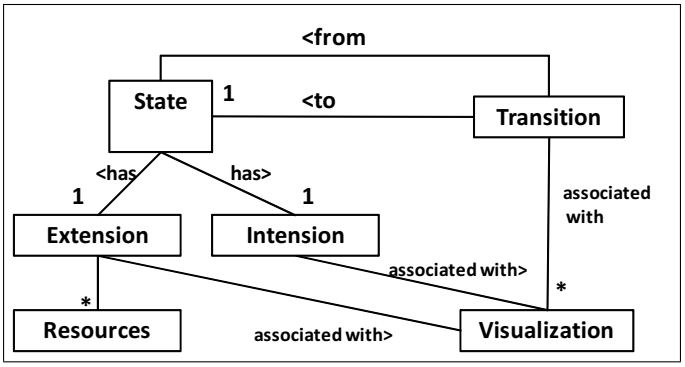

Figure 2.5: State Space from a structural point of view

2.1.4.1 Characterizing Transitions

The key notion in any kind of a browsing approach is that of *transition*. We can distinguish transitions according to various criteria. For instance, we can distinguish transitions on the basis of the user goals shown at Fig. 2.2. Below we will focus on goals that require accessing a *set* of resources. Assuming an object-oriented structuring (since it covers RDF), Fig. 2.6 shows one distinction of such transitions. The left part concerns transitions that do not change the entity type and we can characterize such transitions with respect to the relationship that holds between the extension of current state and that of the target state and with respect to the "handle" that is used for changing the focus (e.g. through attribute values, or through related entities, etc). The right part

concerns transitions that can change the entity type of the current focus. We could also distinguish transitions on the basis of the cardinalities of the extensions of the source and target states. Therefore, there are four types of transitions: one resource-to-one resource, one resource-to-many resources, many-to-many resources, and many-to-one resource.

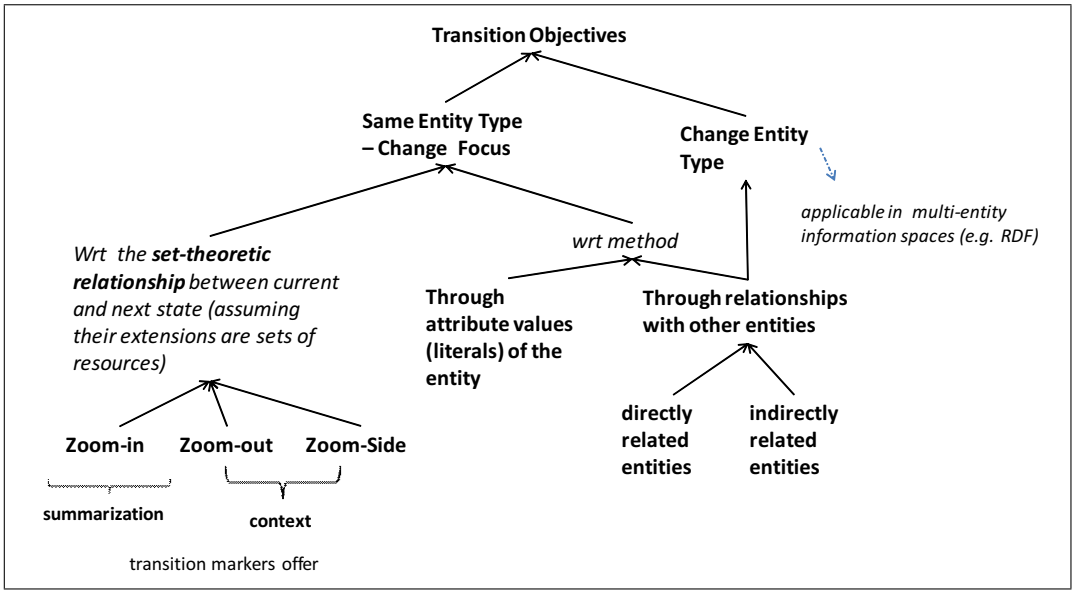

Figure 2.6: Transition Objectives

More precisely, let B denote an information base, and QL denote the query language supported by B. If q is a query in QL, we shall use $q(B)$ to denote its answer over B. A query q is subsumed by q', denoted by $q \leq q'$ if $q(B) \subseteq q'(B)$ in every information base B. Let now $s = (e, q)$ denote a state where e denotes its extension and q its intension. These two elements should satisfy the following constraint: $e = q(B)$. Note that some approaches, like [55], offer exploration over the results of a keyword query kq accomplished with a WSE. In that case B can be considered as the answer of the keyword query, i.e. $B = answer(kq)$ where $answer(kq)$ is a list of hits, each described by metadata values which are hierarchically organized. Now a transition from a state $s = (e, q)$ to an $s' = (e', q')$ can be characterized according to the relationships that hold between their components. More specifically, we can call a transition $s \rightarrow s'$:

- "refinement" (or zoom-in), if $e' \subseteq e$ or $q' \leq q$,
- "relaxation" (or zoom-out), if $e \subseteq e'$ or $q \leq q'$,

- "side-moving", if it is neither refinement nor relaxation but $e \cap e' \neq \emptyset$.

If e and e' contain entities of different types (and therefore it certainly holds $e \cap e' = \emptyset$), we can call the transition $s \rightarrow s'$ "entity type switch". In conclusion, it could be considered that each browsing approach actually defines a set of states, transitions and a visualization method for them.

2.1.4.2 State Space and Ranking

Since the number of states and transitions can be numerous, there is a need for ranking methods. Fig. 2.7 shows a taxonomy for that aspect. The left part concerns what is getting ranked, and the right part concerns the information upon which ranking is performed. For instance, Oren et al. [54] introduce metrics for automatically ranking candidate properties (in the RDF language) based on their *frequency* and the number of values (property values) associated with them. A frequency-based ranking method is also adopted in [28, 27] for ranking properties and property values. In eBay.com the most important attributes are determined in advance through *query and click logs* (as also discussed in [32]). Li et al. [45] propose metrics for ranking attribute hierarchies based on a navigational cost model. In MediaFaces [78] (Yahoo's image search engine) , the candidate properties after a text query are being ranked according to a statistical analysis of image search *query logs*. [20] describes interactive browsing over billions of triples, combining full text search and structured querying (over Virtuoso Cluster Edition), where classes are ordered according to the number of instances that have a property value matching with a particular text query. From another point of view, [34] presents a fuzzy view-based approach in which navigation results are ranked according to resource's fuzzy descriptions (based on manually defined fuzzy concept inclusion axioms).

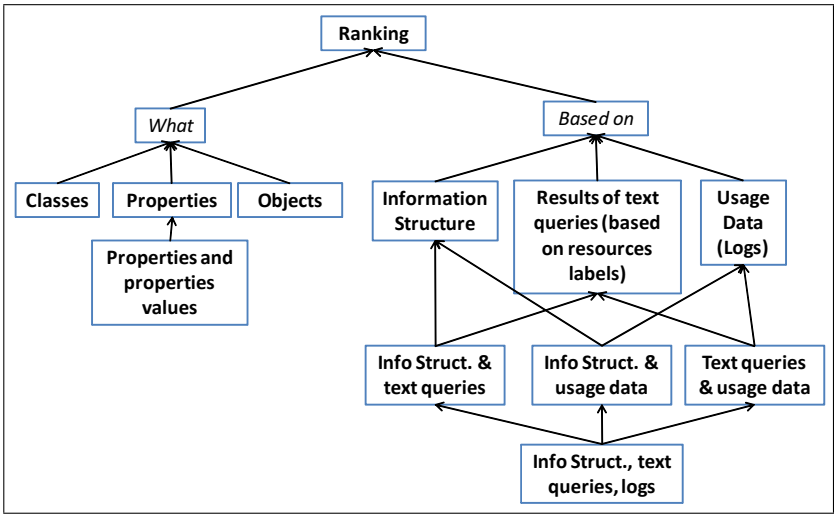

Figure 2.7: *Browsing approaches* and Ranking

2.1.5 Faceted Dynamic Taxonomies

A widely adopted *session-based* interaction scheme for *exploratory search* [74] is *Faceted Dynamic Taxonomies (FDT)*. It bridges the gap between querying and browsing, it provides an overview of the information space, and allows the user to reduce the size of the information space in a simple, flexible and gradual manner [62]. It can be applied over information bases consisting of objects which are described with one or more *terms* (values or concepts) with respect to each *facet* and in addition permits having hierarchically organized attribute domains and multiple classification. At this point we should underline that the notion of *term* corresponds to that of *transition marker* previously described.

The upper part of Table 2.1 defines formally and introduces notations for *terms*, *terminologies*, *taxonomies*, *faceted taxonomies*, *interpretations*, *descriptions* and *materialized faceted taxonomies*. The taxonomies may be predefined or produced by automatic methods, e.g. by on-line results clustering [43] or by methods such as those proposed at [18]. Regarding *object-oriented* conceptual models (i.e. the RDF conceptual model), we should underline that the hierarchical organization of classes through the subclassOf relation could define taxonomic structures. Therefore the provision of faceted dynamic taxonomies is straightforward in case we consider object classification under classes.

MATERIALIZED FACETED TAXONOMIES		
Name	**Notation**	**Definition**
terminology	T	a set of *terms* (can capture categorical/numeric values)
subsumption	$\leq$	a partial order (reflexive, transitive and antisymmetric)
taxonomy	$(T, \leq)$	T is a terminology, $\leq$ a subsumption relation over T
broaders of t	$B^+(t)$	$\{\ t' \mid t < t'\}$
narrowers of t	$N^+(t)$	$\{\ t' \mid t' < t\}$
direct broaders of t	$B(t)$	$minimal_<(B^+(t))$
direct narr. of t	$N(t)$	$maximal_<(N^+(t))$
Top element	$\top_i$	$\top_i = maximal_\leq(T_i)$
faceted taxonomy	$\mathcal{F}$= $\{F_1, ..., F_k\}$	$F_i = (T_i, \leq_i)$, for $i = 1, ..., k$ and all T_i are disjoint
object domain	Obj	any denumerable set of objects
interpretation of T	I	any function $I : T \rightarrow 2^{Obj}$
materialized faceted taxonomy	$(\mathcal{F}, I)$	$\mathcal{F}$ is a faceted taxonomy $\{F_1, ..., F_k\}$ and I is an interpretation of T= $\bigcup_{i=1,k} T_i$
ordering of two interpretations	$I \sqsubseteq I'$	$I(t) \subseteq I'(t)$ for each $t \in T$
model of $(T, \leq)$ induced by I	I	$I(t) = \cup\{I(t') \mid t' \leq t\}$
Descr. of o wrt I	$D_I(o)$	$D_I(o) = \{\ t \in T \mid o \in I(t)\}$
Descr. of o wrt $\bar{I}$	$D_{\bar{I}}(o) \equiv D_I(o)$	$\{\ t \in T \mid o \in \bar{I}(t)\}$ = $\cup_{t \in D_I(o)}(\{t\} \cup B^+(t))$
FDT-INTERACTION: BASIC NOTIONS AND NOTATIONS		
focus	ctx	any subset of T such that ctx = $minimal_<(ctx)$
projection on facet F_i	ctx_i	$ctx \cap T_i$
Kinds of zoom points w.r.t. a facet i while being at ctx		
zoom points	$AZ_i(ctx)$	$\{\ t \in T_i \mid \bar{I}(ctx) \cap \bar{I}(t) \neq \emptyset\}$
zoom-in points	$Z_i^+(ctx)$	$AZ_i(ctx) \cap N^+(ctx_i)$
immediate zoom-in points	$Z_i(ctx)$	$maximal(Z_i^+(ctx))$ = $AZ_i(ctx) \cap N(ctx_i)$
zoom-side points	$ZR_i^+(ctx)$	$AZ_i(ctx) \setminus \{ctx_i \cup N^+(ctx_i) \cup B^+(ctx_i)\}$
maximal zoom-side points	$ZR_i(ctx)$	$maximal(ZR^+(ctx))$
Restriction over an object set $A \subseteq Obj$		
reduced interpretation	I_A	$I_A(t) = I(t) \cap A$
reduced terminology	T_A	$\{\ t \in T \mid \bar{I}_A(t) \neq \emptyset\}$ = $\{\ t \in T \mid \bar{I}(t) \cap A \neq \emptyset\}$ = $\cup_{o \in A} B^+(D_I(o))$

Table 2.1: Faceted information sources and FDT interaction

FDT Interaction Scheme

The main advantages of FDT interaction paradigm is that: (a) it provides an overview of the returned answer (active facets, active values and count information), (b) it releases the user from the effort of formulating queries for locating objects or for restricting his focus (current object set), since he only has to click on the, so called, *zoom points*, usually displayed at the left bar of the GUI (which also make evident how many hits he will get), (d) it does not disappoint the user since clicks always lead to objects (no empty results ever, so there is no need for techniques like those proposed in [16, 4]), (e) it is session-based (in contrast to the state-less query-response) paradigm, thus allowing the user to reach his target *gradually* through the so called *information thinning* process.

The user explores or navigates the information space (a materialized faceted taxonomy according to Table 2.1) by setting and changing his *focus*. The notion of focus can be specified *intensionally*, *extensionally*, or both. Specifically, any conjunction of terms (or any boolean expression of terms in general) is a possible *focus*. For example, the initial focus can be the empty, or the top term of a facet (in this case the entire set Obj is the current object set). However, the user can also start from an arbitrary set of objects, and this is the common case in the context of a Web Search Engine. In that case we can say that the focus is defined *extensionally*. Fig. 2.9 shows the screendump of a WSE that supports FDT-exploration of the answer based on the static and dynamic metadata that are available [55]. Specifically, if A is the result of a free text query q (or if A is a set of tuples returned by an SQL query q), then the interaction is based on the *restriction* of the faceted taxonomy on A. For example, Fig. 2.8(a) shows a taxonomy (consisting of the terms A-G) and 8 indexed objects (1-8). Let's describe over this small example (that consists of one facet), the exploration through Faceted Dynamic Taxonomies (FDT). Fig. 2.8(b) shows the dynamic taxonomy if we restrict our focus to the objects {4,5,6}. The notion of restriction is defined formally at the bottom part of Table 2.1. Fig. 2.8(c) shows the browsing structure that could be provided at the GUI layer and Fig. 2.8(d) sketches user interaction.

At any point during the interaction, the *immediate zoom-in/out/side points* are computed and provided to the user along with count information (as shown in Fig. 2.8(d)).

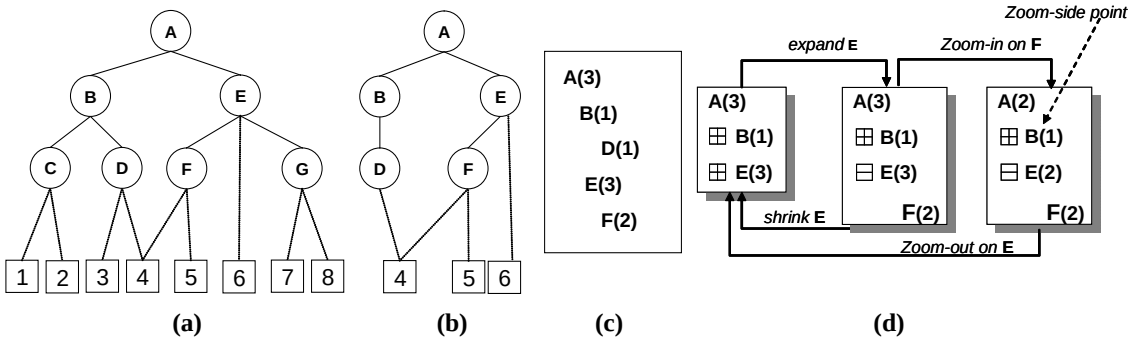

Figure 2.8: An example of Dynamic Taxonomies (assuming one facet)

When the user selects one of these points then the selected term is added to the focus, and so on. The corresponding interaction states are specified extensionally due to $q(B)$ and intentionally due to the conjunction of terms that corresponds to the user clicks. More specifically and in terms of Section 2.1.4, consider a transition from a state $s = (e, q)$ to an $s' = (e', q')$. According to FDT, the *refinement* transitions are in the form of (e', q') such that $e' \neq \emptyset$ and q' is derived by replacing a conjunct of q with a term that is narrower than q and it holds $e' \subseteq e$ or $q' \leq q$.

There are several applications of FDT interaction paradigm in e-commerce (e.g. ebay), library and bibliographic portals (e.g. DBLP) and Web Search Engines (e.g. [55]). Moreover, several works [33, 48, 54, 22] attempt to apply the FDT interaction paradigm in the Semantic Web. Also, in terms of of *information spaces*, it has been applied over flat attribute domains [10], hierarchically organized attribute domains [77, 72], as well as over RDF [33, 54, 22].

2.2 Related Work

Here we list and categorize several *browsing approaches* using the aspects that we have previously described. As regards *information space*, in the simplest case we have attribute-value pairs where the domain of attributes is flat, and such works require the single-entity approach. Relational databases do not have an explicit representation of the conceptual schema and thus they cannot be easily characterized to single or multi entity. Table 2.2 lists some single entity approaches and browsers, and characterizes them according to the

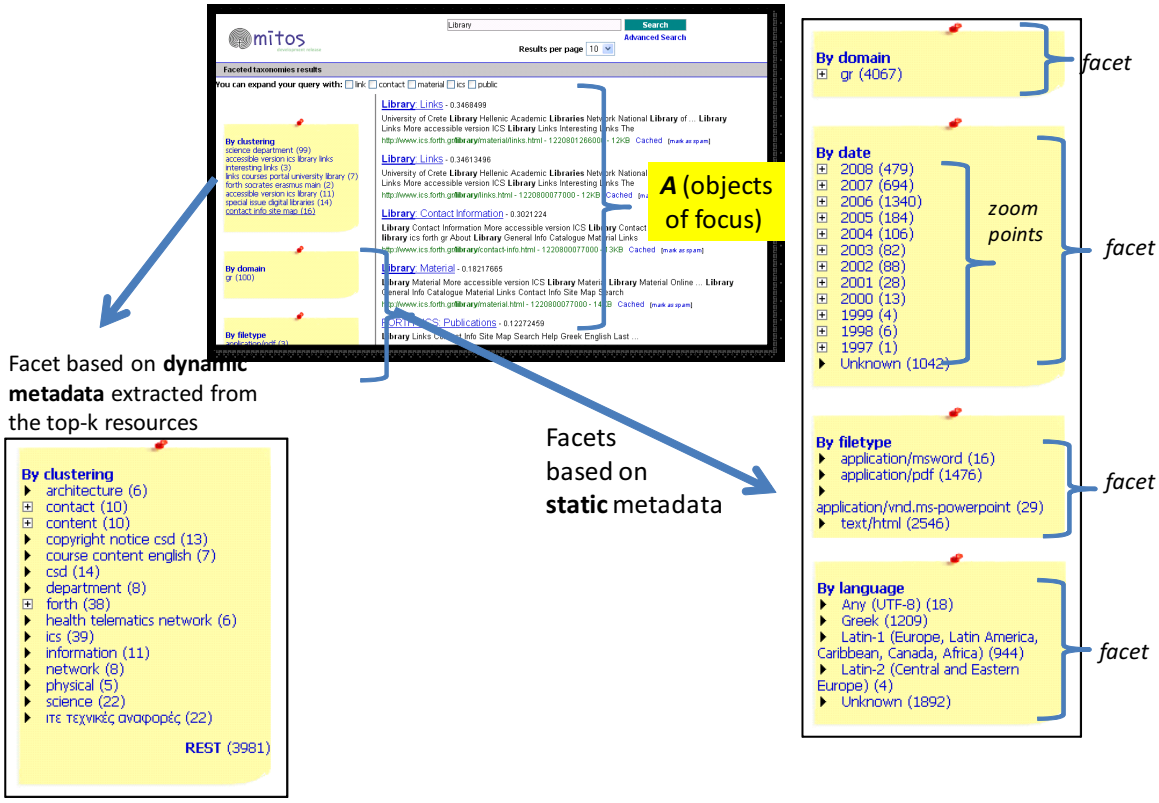

Figure 2.9: `mitos` GUI for FDT-interaction

assumed user goal and the structuring of the information space.

System	Goal	Information space
RB++[83]	Count	AFV
Elastic Lists [68]	Count	AFV
Flexplorer [72]	Count	AHV
Flamenco [77]	Count	AHV
Faceted search impl. [10]	Aggregation	AFV
Dynamic Faceted Search syst.[19]	Aggregation	AHV

Count: Groups accompanied with count info
Aggregation: More complex aggregated results
AFV: Attribute Flat Values
AHV: Attribute Hierarchical Values

Table 2.2: List of Single-Entity browsing approaches and browsers

In an *object oriented* conceptual model we have classes, objects, attributes and links between objects, so it requires a multi-entity approach. The RDF/S model falls into this category, and related works include: BrowseRdf [54], Humboldt [41], VisiNav [28], Parallax [36], Longwell [57], Ontogator [48], /facet [33], Camelis2 [22].

Regarding now *configuration*, we have domain-specific approaches, like mspace [52] and generic approaches like /facet [33] and VisiNav [28]. The latest is applied over Linked Open Data that is probably the most growing collection of linked datasets available in RDF. We should note here, that since web data comes in a plethora of vocabularies and there is no schema describing the interlinking between these datasets [40], generic approaches are required.

Furthermore, as generic methods target to exploration of arbitrary RDF repositories with no fixed schema, they often require automatic methods for facets selection and ranking. For instance, Oren et al. [54] introduce metrics for automatically ranking facets according to their quality for browsing. Hearst describes the selection of important facets based on query logs and click logs [32]. MuseumFinland [37] supports access to heterogeneous content but with a limited variance in schemas from a number of museum collections while mapping rules are used in order to face the peculiarities of the underlying data model. Table 2.3 concerns multi-entity approaches that provide ranking over

the browsable elements. It characterizes them according to what is ranked and on what basis.

System	Ranking of	Based on
BrowseRdf [54]	properties	IS
VisiNav [28]	properties, prop. values	IS
Faceted Wikipedia [27]	properties	IS
MediaFaces [78]	properties	IS+L
Faceted Data Explorer[78]	classes	IS+T
IS: Information Structure **L**: Logs **T** : Text queries		

Table 2.3: Multi-entity browsers and their Ranking Capabilities

Table 2.4 focuses on multi-entity approaches and characterizes them according to the assumed user goal, the structuring of the information space, the configuration requirements, and multitude of underlying data sources. It clearly emerges that although there exist several browsers for navigation over the object-oriented conceptual model, no works for generic exploration of fuzzy information bases have been proposed.

In the following Chapter we define a precise and concise state-based model capturing the essentials of browsing approaches over RDF/S. Then in Chapter 4, we extend the proposed model in order to capture the case of exploration over information spaces with fuzzy objects descriptions and associations.

System	Goal	Information space	Config.	Data Sources
mspace [52]	Set	O-O	V	one
Ontogator [48]	Count	O-O	V	one
MuseumFinland [37]	Count	O-O	V	multiple
Camelis2 [22]	Count	O-O	V	one
Faceted Data Explorer[20]	Count	O-O	V	one
NFB [35]	Set	O-O	V	one
GRQL [7]	Set	O-O	V	one
/facet [33]	Count	O-O	G	one
Longwell [57]	Count	O-O	G	one
Humboldt [41]	Set	O-O	G	one
VisiNav [28]	Count	O-O	G	multiple
Parallax [36]	Count	O-O	G	multiple
Faceted Wikipedia [27]	Count	O-O	G	one
MediaFaces [78]	Count	O-O	G	multiple
BrowseRdf[54]	Set	O-O	G	one
Fuzzy view based search[34]	Set	FRDF	V	one
Odalisque [5]	Count	OWL	V	one

OO: Object Oriented
FRDF: Fuzzy RDF
V:View-based
G:Generic

Table 2.4: List of Multi-Entity browsing approaches and browsers

Chapter 3

A Generic Interaction Model for navigation over RDF/S

This Chapter defines a precise and concise state-based model capturing the essentials of browsing approaches over RDF/S. Also, it studies several issues regarding interaction and the particularities of the RDF/S model. Moreover, we examine here implementation approaches and query language issues for the proposed model. Finally, this Chapter surveys several browsing approaches in terms of the proposed interaction model.

3.1 RDF Background and Notations

This section introduces notions and notations for the RDF/S model that shall be used in the sequel. Let *URI* be the set of URI references and *LIT* be the set of plain and typed literals. An RDF/S Knowledge Base (KB) is defined by a set of RDF triples of the form (*subject*, *predicate*, *object*), where *subject*, *predicate* $\in$ *URI* and *object* $\in$ *URI* $\cup$ *LIT*. Let K be a KB. The *closure* of KB K, denoted by $\mathcal{C}(K)$, contains all the triples of the form $(s,\ p,\ o)$, where $s,\ o \in URI \cup LIT$ and $p \in URI$, that either are explicitly asserted or can be entailed from K based on RDFS-entailment of the RDF/S semantics [31].

Def. 1 The *schema* of an RDF/S KB K is a 6-tuple $\Gamma_K = \langle C,\ Pr,\ domain, range,\ \leq^*_{cl}$, $\leq^*_{pr} \rangle$, where:

- C is the set of classes of $\mathcal{C}(K)$, i.e. $c \in C$ iff $(c\ type^1\ rdfs{:}Class) \in \mathcal{C}(K)$.
- Pr is the set of properties of $\mathcal{C}(K)$, i.e. $pr \in Pr$ iff $(pr\ type\ rdf{:}Property) \in \mathcal{C}(K)$.
- *domain* is a total function $domain: Pr \rightarrow C$ that maps a property in Pr to its domain, i.e. for $pr \in Pr$ and $c \in C$, $domain(pr) = c$ iff $(pr\ rdfs{:}domain\ c) \in \mathcal{C}(K)$.
- *range* is a total function $range: Pr \rightarrow C$ that maps a property in Pr to its range, i.e. for $pr \in Pr$ and $c \in C$, $range(pr) = c$ iff $(pr\ rdfs{:}range\ c) \in \mathcal{C}(K)$.
- $\leq^*_{cl}$ is the *subClassOf* relation between C, i.e. for $c, c' \in C$, $c \leq^*_{cl} c'$ iff $(c\ rdfs{:}subClassOf\ c') \in \mathcal{C}(K)$.
- $\leq^*_{pr}$ is the *subPropertyOf* relation between Pr, i.e. for $pr, pr' \in Pr$, $pr \leq^*_{pr} pr'$ iff $(pr\ rdfs{:}subPropertyOf\ pr') \in \mathcal{C}(K)$[2]. □

We consider an RDF/S KB K to be *valid* if (i) it has an RDFS-interpretation, according to RDF/S semantics [31], (ii) the relations $\leq^*_{cl}$ and $\leq^*_{pr}$ are acyclic, (iii) the total functions *domain* and *range* are well-defined, i.e. each property in Pr has a single domain and range, and (iv) if $pr \leq^*_{pr} pr'$ then $domain(pr) \leq^*_{cl} domain(pr')$ and $range(pr) \leq^*_{cl} range(pr')$.

We define the *resources* of K as $Res = \{o \mid (o\ type\ rdfs{:}Resource) \in \mathcal{C}(K)\}$ and the *instances* of K as $Inst = Res \setminus (C \cup Pr)$. We also define the *instances of a class* $c \in C$ as $inst(c) = \{o \mid (o\ type\ c) \in \mathcal{C}(K)\}$, while the instances of a property $pr \in Pr$ as $inst(pr) = \{\ (o, pr, o') \mid (o, pr, o') \in \mathcal{C}(K)\}$. Let $\mathcal{V}_{RDF}$ be the set of URI references in the *rdf*: and *rdfs*: namespace [31]. We can distinguish two sets of RDF triples, called *schema triples* and *instance triples*. *Schema triples* are RDF triples of the form provided in Table 3.1, where $c,\ c',\ pr,\ pr' \in URI \setminus V_{RDF}$ and $c'' \in URI$.

Def. 2 (Instance Triples) The *instance triples* of K are the RDF triples of the form $(o\ type\ c)$ and $(o\ pr\ o')$, where $o \in Inst \cap URI$, $o' \in Inst$, $c \in (C \setminus \mathcal{V}_{RDF})$, and $pr \in (Pr \setminus \mathcal{V}_{RDF})$. □

We will refer to instance triples of the form $(o\ type\ c)$, as *class instance triples*, and to instance triples of the form $(o\ pr\ o')$, as *property instance triples*. Also, hereafter with C we will refer to $C \setminus \mathcal{V}_{RDF}$ and with Pr we will refer to $Pr \setminus \mathcal{V}_{RDF}$.

[1]For simplicity, we have eliminated the namespace prefix *rdf*: in front of the term *type*.
[2]Recall that according to the RDF/S semantics [31], $\leq^*_{cl}$ and $\leq^*_{pr}$ are reflexive and transitive relations.

triple	abbreviation
c *rdf*:*type* *rdfs*:*Class*	$Class(c)$
c *rdfs*:*subClassOf* c'	$c \leq_{cl} c'$
pr *rdf*:*type* *rdf*:*Property*	$Property(pr)$
pr *rdfs*:*subPropertyOf* pr'	$pr \leq_{pr} pr'$
pr *rdfs*:*domain* c''	$domain(pr) = c''$
pr *rdfs*:*range* c''	$range(pr) = c''$

Table 3.1: Schema Triples

3.2 A Generic Browsing Model for RDF

The objective is to define a precise and concise model capturing the essentials of RDF browsing approaches for recall-oriented information needs. Also, the browsing approaches we capture with this model support accessing resources in groups and with count information, are applicable to Object-Oriented information spaces like RDF/S, support multi-entity browsing and are generic.

The proposed interaction is modeled by a *state space* like that described in Section 2.1.4. We should remind here that each *state* has an *extension* and a number of *transitions* leading to other states. Each transition is signified by a *transition marker* that is accompanied by a number showing the size of the extension of the targeting state. We will refer to this as *count* information and represent it as $s.count$. Finally, we should underline that our method abstracts from the various visualization approaches. In general each state could have one or more visualization modes for its extension as well as its transition markers.

[Running Example]. We shall use Fig. 3.1 as our running example. An instance of the proposed interaction is sketched at Fig. 3.2. The figure depicts only the part of the UI that shows the transition markers.

[Initial States] Consider that we are in the context of one RDF/S KB with a single namespace with classes C and properties Pr. If s denotes a state we shall use $s.e$ to denote its extension. Let's start from the *initial state*(s). Let s_0 denote an artificial initial state. We can assume that $s_0.e = URI \cup LIT$, i.e. its extension contains every

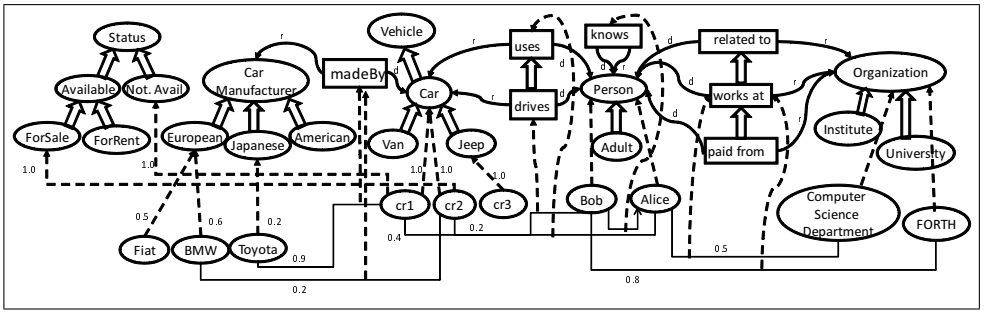

Figure 3.1: An RDF KB

Properties are depicted by rectangles and the letters "d" and "r" are used to denote the domain and the range of a property. Fat arrows denote subClassOf/subPropertyOf relationships, while dashed arrows denote instanceOf relationships.

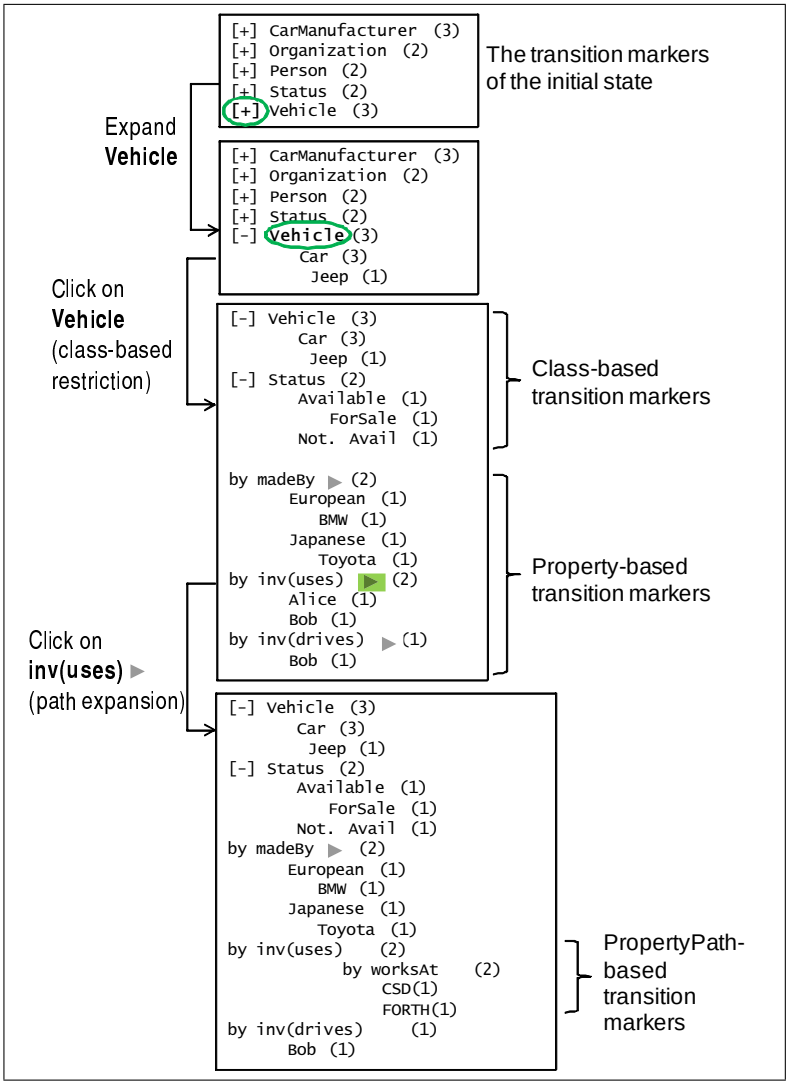

Figure 3.2: Sketch of the GUI part for transition markers

URI and literal of the KB. Alternatively, the extension of the initial state can be the result of a keyword query, or a set of resources provided by an external access method. Given a state we shall show how to compute the transitions that are available to that state. From s_0 the user can move to states corresponding to the maximal classes and properties, i.e. to one state for each $maximal_{\leq_{cl}}(C)$ and each $maximal_{\leq_{pr}}(Pr)$. Specifically, each $c \in maximal_{\leq_{cl}}(C)$ (resp. $p \in maximal_{\leq_{pr}}(Pr)$) yields a state with extension $inst(c)$ (resp. $inst(p)$).

We will define formally the transitions based on the notion of *restriction* and *join*. To this end we introduce some auxiliary definitions. We shall use p^{-1} to denote the inverse direction of a property p, e.g. if $(d, p, r) \in Pr$ then $p^{-1} = (r, inv(p), d)$, and let Pr^{-1} denote the inverse properties of all properties in Pr. If E is a set of resources, p is a property in Pr or Pr^{-1}, v is a resource or literal, *vset* is a set of resources or literals, and c is a class, we define the following notations for *restricting* the set E:

$$\begin{aligned} Restrict(E, p : v) &= \{\ e \in E \mid (e, p, v) \in inst(p)\} \\ Restrict(E, p : vset) &= \{\ e \in E \mid \exists\ v' \in vset \text{ and } (e, p, v') \in inst(p)\} \\ Restrict(E, c) &= \{\ e \in E \mid e \in inst(c)\} \end{aligned}$$

Now we define a notation for *joining* values, for computing either values or objects associated with elements of E:

$$Joins(E, p) = \{\ v \mid \exists\ e \in E \text{ and } (e, p, v) \in inst(p)\}$$

Thereafter, we define precisely transitions and transition markers given a state s with extension $s.e$.

3.2.1 Class-based browsing

The classes that can be used as class-based transition markers, denoted by $TM_{cl}(E)$, are defined by:

$$TM_{cl}(E) = \{c \in C \mid Restrict(E, c) \neq \emptyset\} \tag{3.1}$$

If the user clicks on a $c \in TM_{cl}(s)$, then the extension of the targeting state s' is defined as $s'.e = Restrict(s.e, c)$, and its count information is $s'.count = |s'.e|$. For example,

suppose the user selects the class `Vehicle`. The user can then view its instances and follow one of the following class-based transition markers: `Vehicle, Car, Jeep, Status, Available, ForSale, Not.Available`. Notice that `ForRent` and `Van` are not included because their extension is empty. Consequently, their intersection with the current extension will be empty.

The elements of $TM_{cl}(s)$ can be hierarchically organized based on the subclass relationships among them. Specifically, the layout (e.g. the indentation in a text-based layout) of the transition markers can be based on the relationships of the reflexive and transitive reduction of the restriction of $\leq^*_{cl}$ on $TM_{cl}(s)$ denoted by $R^{refl,trans}(\leq^*_{cl \mid TM_{cl}(s)})$. In our case, we can get what is shown in Fig. 3.3(a).

Furthermore based on the relationship between the extensions $s.e$ and $s'.e$, we could characterize a transition or the corresponding transition marker as zoom-in/out/side (as these are defined in Section 2.1.4.1).

3.2.2 Property-based browsing

Here we focus on another kind of transitions captured by the proposed interaction model, called *property-based transitions*.

We will introduce this kind of transitions by first giving some examples over Fig. 3.1. Suppose the user has focused on the class `Car`, and the extension of this state is {`cr1, cr2, cr3`}. He can further restrict the extension also through properties whose domain or range is the class `Car`, or a superclass of `Car`. In general, any property related with the resources in $s.e$ can be considered as candidate means for restricting it. For example, consider a property `madeBy` whose domain is the class `Car` and suppose its range was the `String Literal` class. In that case the firm names of the current extension can be used as transition markers. Now suppose that the range of the property `madeBy` is not literal, but the class `CarManufacturer`. In this case, the firms (URIs in this case) of the current extension can again be used as transition markers, as shown in Fig. 3.3(b). Notice that `Fiat` is not shown as it is not related to the current focus (i.e. to `cr1`, `cr2` and `cr3`)[3]. Formally, the properties (in their defined or inverse direction) that can be used for

[3]Since `cr3` does not participate to a `madeBy` property, an alternative approach is to add an artificial

deriving transition markers are defined by:

$$Props(s) = \{p \in Pr \cup Pr^{-1} \mid Joins(s.e, p) \neq \emptyset\} \tag{3.2}$$

For each $p \in Props(s)$, the corresponding transition markers are defined by $Joins(s.e, p)$, and if the user clicks on a value v in $Joins(s.e, p)$, then the extension of the new state is $s'.e = Restrict(s.e, p : v)$.

(a)

```
Vehicle(3)
  Car(3)
     Jeep(1)
Status(2)
  Available(1)
    ForSale(1)
  Not. Avail(1)
```

(b)

```
by madeBy(2)
   BMW(1)
   Toyota(1)
```

(c)

```
by madeBy(2)
   European(1)
     BMW(1)
   Japanese(1)
     Toyota(1)
```

(d)

```
by inv(uses)(2)
   Alice(1)
   Bob(1)
by inv(drives)(1)
   Bob(1)
```

(e)

```
by inv(uses)(2)
   by worksAt(2)
      CSD(1)
      FORTH(1)
```

Figure 3.3: Examples of transition markers

Furthermore, the transition markers of a property $p \in Props(s)$, i.e. the set $Joins(s.e, p)$, can be categorized based on their classes. In our example, the firms can be categorized through the subclasses of the class `CarManufacturer`. These classes can be shown as intermediate nodes of the hierarchy that lead to particular car firms, as shown in Fig. 3.3(c). These classes can be computed easily, they are actually given by $TM_{cl}(Joins(s.e, p))$. Furthermore, these values can be used as *complex transition markers*, i.e. as shortcuts allowing the user to select a set of values with disjunctive interpretation (e.g. he clicks on `Japanese` instead of clicking to every Japanese firm). Specifically, suppose the user clicks on such a value vc. The extension of the target state s' will be:

$$s'.e = Restrict(s.e, p : Restrict(Joins(s.e, p), vc)) \tag{3.3}$$

Returning to our example, and while the user has focused on cars, apart from `madeBy`, the user can follow transitions based on the properties `inv(drives)` and `inv(uses)`, as shown in Fig. 3.3(d). In addition, the elements of $Props(s)$ can be hierarchically organized based on the subProperty relationships among them.

value, like `NonApplicable/Uknown`, whose count would be equal to 1, for informing the user that one element of his focus has not value wrt `madeBy`.

3.2.3 Property Path-based browsing

Thereafter, we should be able to extend the *property-based* browsing in order to capture *property paths* of length greater than one. This is required for restricting the extension through the values of complex attributes or through the relationships (direct or indirect) with other resources. Complex attributes are considered to be the attributes represented as blank nodes [4] and identified by properties only. For instance, in the following set of triples, $_{:}nikosAddress$ indicates the presence of a blank node:

$$exstaff : 85740 \quad exterms : address \quad \_ : nikosAddress.$$
$$\_ : nikosAddress \quad exterms : street \quad \text{“}LappaStreet\text{”}.$$
$$\_ : nikosAddress \quad exterms : city \quad \text{”}Heraklion\text{”}.$$

As regards resources relationships, one may want to restrict the set of cars so that only cars which are used by persons working for CSD (Computer Science Department) are shown. In that case we would like transition markers of the form shown in Fig. 3.3(e). It should also be possible the successive “application” of the same property. For example, the user may want to focus to all friends of the friends of `Bob`, or all friends of Bob at distance less than 5. Let's now define precisely, this *property path*-based browsing (expansion and cascading restriction). Let $p_1, \ldots, p_k$ be a sequence of properties. We call this sequence *successive in* s if $Joins(Joins(\ldots(Joins(s.e, p_1), p_2)\ldots p_k) \neq \emptyset$. Obviously such a sequence does not lead to empty results, and can be used to restrict the current focus. Let $M_1, \ldots M_k$ denote the corresponding set of transition markers at each point of the path. Assuming $M_0 = s.e$, the transition markers for all i such that $1 \leq i \leq k$, are defined as:

$$M_i = Joins(M_{i-1}, p_i) \tag{3.4}$$

What is left to show is how selections on such paths restrict the current focus. Suppose the user selects a value v_k from M_k. This will restrict the set of transitions markers in the

[4] http://www.w3.org/TR/rdf-syntax/

following order $M_k, \ldots, M_1$ and finally it will restrict the extension of s. Let $M'_k, \ldots M'_1$ be the restricted set of transitions markers. They are defined as follows: $M'_k = \{v_k\}$, while for each $1 \leq i < k$ we have:

$$M'_i = Restrict(M_i, p_{i+1} : M'_{i+1}) \tag{3.5}$$

for instance, $M'_1 = Restrict(M_1, p_2 : M'_2)$. Finally, the extension of the new state s' is defined as $s'.e = Restrict(s.e, p_1 : M'_1)$. Equivalently, we can consider that M'_0 corresponds to $s'.e$ and in that case Eq. 3.5 holds also for $i = 0$.

For example, consider an ontology containing a path of the form: `Car--hasFirm-->Firm--ofCountry-->Country` and three cars `c1, c2, c3`, the first being `BMW`, the second `VW` and the third `Renault`. The first two firms come from `Germany` the last from `France`. Suppose the user is on `Cars`, and expands the path `hasFirm.ofCountry`. If he selects `Germany`, then the previous list will become `BMW, VW` (so `Renault` will be excluded) and the original focus will be restricted to `c1` and `c2`. It follows that path clicks require disjunctive interpretation of the matched values in the intermediate steps. Figure 3.4 illustrates this process.

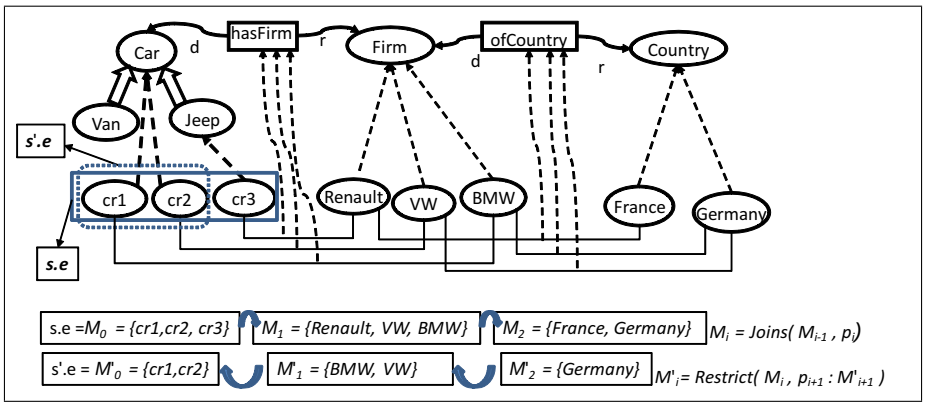

Figure 3.4: Path Expansion Example

The above can be applied also for successive applications of the same property, e.g. `inv(drives).knows`^{2}`.paidFrom` is a property path that can be used to restrict cars to those cars whose drivers know some persons who in turn know some persons who are paid from a particular organization. In conclusion, we would say that the transitions based on *property paths* would be useful on navigation over social data, as in FOAF[5] and SIOC[6]

[5]http://xmlns.com/foaf/spec/
[6]http://rdfs.org/sioc/spec/

spaces where there are many blank nodes and there exist long property "chains" (e.g. `foaf:knows`).

3.2.4 Entity Type Switch

So far we have described methods to restrict the current extension. Apart from the current extension we can move to other objects. At the simplest case, from one specific resource we move to one resource which is directly or indirectly connected to that.

Now suppose that the current focus is a *set* of resources (e.g. cars). Again we can move to one or more resources which are directly or indirectly connected to all, or at least one of the resources of the current focus. For example, while viewing a set of cars we can change our focus to the list of their firms. In this way we interpret disjunctively the elements associated with every object of the focus.

Moreover, we can move to indirectly connected resources. For instance, while viewing a set of cars we can move to all related organizations through the path `inv(uses).worksAt`. An interaction paradigm for the accomplishment of such a transition is sketched at Fig. 3.5. To capture this requirement it is enough to allow users to move to a state whose extension is the current set of transition markers.

To capture the requirement of *entity type switch* it is enough to allow users to move to a state whose extension is the current set of transition markers. As the notion of entity type corresponds to the notion of RDF class, class-based browsing can be considered as entity type switch, even though the source and target type may be subclassOf related. We should note here that the classes of the transition markers are not necessarily different than those of the current focus, e.g. in case we have cycles at schema level.

For a more generic example, consider a user who starts from the class `Persons`, and then restricts his focus to those persons who `workAt FORTH`. Subsequently, he restricts his focus through the property path `drives.madeBy` and by selecting the complex transition marker `European`. At that point he asks to switch the entity type to `Cars`. This means that the entity type of the extension of the new state should be `Cars`, and the extension of the new state will contain *European cars which are driven by persons working at FORTH*. The property `drives` (actually its inverse direction), could be used to restrict the current

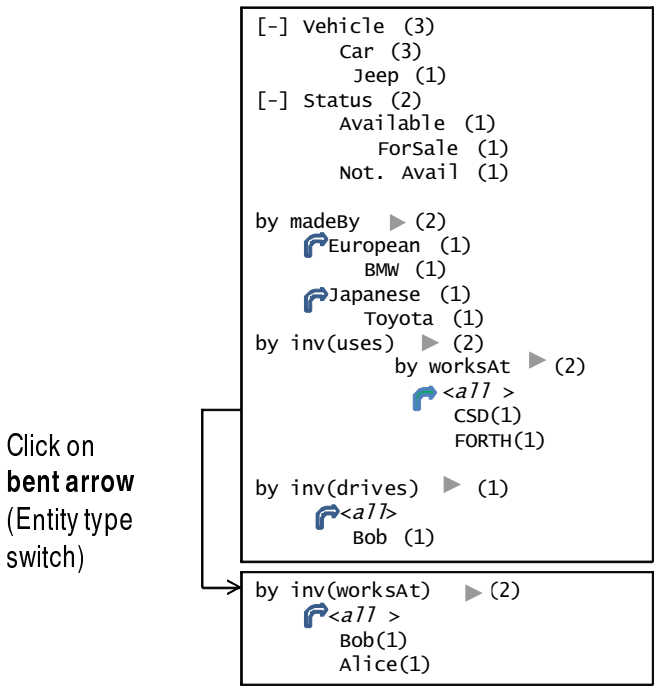

Figure 3.5: Entity type switch on all the tms of the property path `inv(uses).worksAt`

focus. Furthermore, the user can proceed and restrict his focus, that is *European cars which are driven by persons working at FORTH*, to those which are `ForSale`, and so on.

3.3 Characterizing Sessions

A key characteristic of interactive search is that it is *session-based*. For this reason it is worth identifying problematic cases, as redundant steps, for offering better guidance during the interaction, e.g. notify or disallow the user from making redundant steps. For example, the user may use path expansion using the same property in a direct and indirect direction in an alternate fashion. Obviously this is redundant. A non trivial case of *extensionally equivalent* sequences is that of cycles at schema and instance level, for which we propose a special treatment in Section 3.4. Some related definitions follow.

- A *session* is a sequence of states (connected through transition markers).
- A sequence of steps is a *refinement session* if for any successive pair of states, s and s', it holds $s'.e \subseteq s.e$. If $s'.e \subset s.e$, the session $s.s'$ is *strictly restrictive*, while if $s'.e = s.e$. then it is *extensionally equivalent*.

- A session is *single entity* if it does not contain any entity switch transition. In case a session is not single entity, then it contains two successive states s and s' such that the intersection of their extension is empty.

3.4 Path Expansion and Cycles: MaxExpansionSteps

We may have cycles at schema and instance level. At schema level, a property sequence may have the same starting and ending class forming a cycle. Cyclic properties can be considered as a special case where the length of the sequence is 1 (or the domain and the range of a property is the same class).

Now consider a user who wants to restrict the initial focus set, e.g. a set of persons, through other persons connected with them through the property `knows` at depth m. For this reason the user expands the property `knows` m times and then selects a person. However, at some point we may want to stop suggesting path expansions, in order to avoid prompting to the user the same set of transition markers (which may restrict the initial focus in the same way). Fig. 3.7 shows how the transition markers change while expanding the path `inv(owns).knows`^{i} over the example shown in Fig. 3.6. When the property `knows` is expanded over 3 times, the transition markers (and their count info) are being repeated while at the same time they lead to *extensionally equivalent* states. At Fig. 3.7 we use only the first letter of a name, and paths over `knows` are depicted as sequences of such letters.

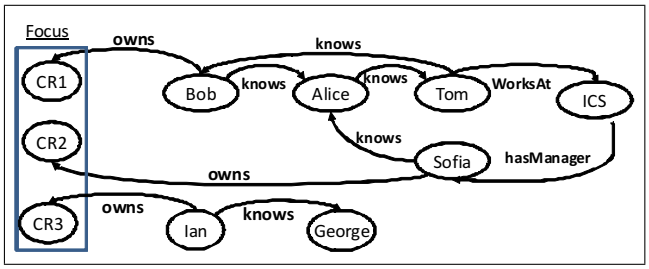

Figure 3.6: Instance cycles example

We propose adopting the following policy:

> *Stop path expansion when each object of the s.e has been made accessible (i.e. restrictable) through all transition markers that are possible.*

TMs for `inv(owns).knows`^{i}		ipath from cr1	ipath from cr2	ipath from cr3
i=1	A(2), G(1)	B.**A**	S.**A**	I.**G**
i=2	T(2)	B.A.**T**	S.A.**T**	-
i=3	B(2)	B.A.T.**B**	S.A.T.**B**	-
i=4	A(2)	B.A.T.B.**A**	S.A.T.B.**A**,	-
i=5	T(2)	B.A.T.B.A.**T**	S.A.T.B.A.**T**	-

Figure 3.7: Path expansions of the example of Fig. 3.6

Below we explain how we can compute the maximum number of expansion steps at interaction time.

Let $\Gamma = (N, R)$ be a directed graph. We define a *path* from a node n_1 to a node n_{k+1}, as any sequence of edges of the form $(n_1, n_2) \dots (n_k, n_{k+1})$ where $(n_i, n_{i+1}) \in R$, and $i \neq j$ implies that $n_i \neq n_j$. The length of such a path is k, and we shall write $n_1 \overset{k}{\rightsquigarrow} n_{k+1}$ to denote that there exists a path of length k from n_1 to n_{k+1}. We shall also write $n \overset{*}{\rightsquigarrow} n'$ to denote that exists one or more paths from n to n'. Now we define the distance from n to n' as the length of the *shortest* path from n to n', i.e. $Dist(n \to n') = \min\{\ k \mid n \overset{k}{\rightsquigarrow} n'\}$. Given two subsets A and B of N (i.e. $A, B \subseteq N$), we define the distance from A to B as the *maximum* distance between any pair of nodes form these sets, i.e.

$$Dist(A, B) = \max\{\ Dist(a \to b) \mid a \in A, b \in B\} \tag{3.6}$$

Returning to our problem, N is the set of all nodes of the RDF graph. For a property $p \in Pr$ we can define the edges $R_p = \{(a, b) \mid (a, p, b) \in \mathcal{C}(K)\}$. We can now define the reachable nodes from a node n (through p property instances) as $Reachable_p(n) = \{\ n' \mid n \overset{*}{\rightsquigarrow}_p n'\}$, where the meaning of the subscript p is that paths are formed from elements of R_p. Being at a state s, the maximum number of path expansion steps (for property p) that is required are:

$$MaxExpansionSteps(s, p) = Dist(s.e, \bigcup_{n \in s.e} Reachable_p(n)) \tag{3.7}$$

With this number of steps it is guaranteed that each object of $s.e$ has been made accessible (restrictable) through all tms which are possible. The proof is trivial: the path

starting from an object o with length bigger than $MaxExpansionSteps(s, p)$ will not encounter a tm that has not already been reached.

Now consider path expansions over different properties, e.g. consider the property path `inv(owns).knows.knows`. In such cases we would like to identify the maximum expansion steps for each p that is used in the expansion, or the maximum expansion steps in general. Let $pset$ be a set of properties (i.e. $pset \subseteq Pr$). We can define the edges by considering all properties in $pset$, i.e. $R_{pset} = \{(a, b) \mid p \in pset,\ (a, p, b) \in \mathcal{C}(K)\}$. Now we can define $Reachable_{pset}(n) = \{\ n' \mid n \rightsquigarrow^*_{pset} n'\}$, where the subscript $pset$ means that paths consist of edges in R_{pset}. The set $Reachable_{pset}(n)$ is the set of all tms that provide access to n through paths consisted of instances of properties in $pset$. Therefore the tms that correspond to all objects in s are given by $\bigcup_{n \in s.e} Reachable_{pset}(n)$. Being at a state s, the maximum number of path expansion steps (using properties from $pset$) that is required is:

$$MaxExpansionSteps(s, pset) = Dist(s.e, \bigcup_{n \in s.e} Reachable_{pset}(n)) \tag{3.8}$$

By assuming in Fig. 3.6 that $pset$ ={`inv(owns), knows, worksAt, hasManager`} then the overall maximum expansion steps number is:
$MaxExpansionSteps(s, pset) = Dist(s.e, \{Bob, Ian, George, Sofia, Alice, Tom, ICS\}) =$ 5, due to the path:
`CR1.inv(owns).Bob.knows.Alice.knows.Tom.WorksAt.ICS.hasManager.Sofia` Note that this does not specify the maximum expansion steps for each property in $pset$ (which is obviously lower), but the overall. For example, if $pset$ consists only of the property `inv(owns)` then just one expansion is enough.

To find and propose to the user the appropriate properties for further expanding a property path which uses properties in $pset$, for each candidate property p (such that $Joins(M_i, p) \neq \emptyset$), we compute $MaxExpansionSteps(s, tmp)$, where $tmp = pset \cup \{p\}$. Then, we examine if the length of the path already formulated is equal to $MaxExpansionSteps(s, tmp)$ or less than it. If less then the expansion of p would be rejected. For example, consider the property path `inv(owns).knows.knows` at Fig. 3.6 which has length equal to 3 and $pset$=$\{inv(owns), knows\}$. Here we have $M_3 = \{Tom\}$

and the properties that satisfy the condition $Joins(M_3, p) \neq \emptyset$ are: `knows`, `inv(knows)` and `WorksAt`. Then for the property `knows`, we have that: $tmp = \{inv(owns), knows\}$ and $MaxExpansionSteps(s, tmp) = 3$. As the path length is equal to $MaxExpansionSteps(s, tmp)$, the expansion of the property *knows* is rejected. As for the property `inv(knows)`, we have that: $tmp = \{inv(owns), knows\}$ and $MaxExpansionSteps(s, tmp) = 3$. However, the path length is equal to $MaxExpansionSteps(s, tmp)$ and the expansion of the property *inv(knows)* is getting rejected. Finally as regards the property `WorksAt` we have that: $tmp = \{inv(owns), knows, WorksAt\}$ and $MaxExpansionSteps(s, tmp) = 4$. Since the length of the already formulated path is less than 4 that property would be proposed to user and we would have that $pset = \{inv(owns), knows, WorksAt\}$.

3.5 Tracking History

We can distinguish *user clicks* to (a) *state-changing* and (b) *state-preserving*. Clicks on transitions markers correspond to (a) since they lead to a different state. On the other hand, clicks which change only the visibility of the transition markers of the current state, fall into category (b).

3.5.1 State-changing Clicks

Here we have clicks upon transitions markers which trigger a transition and thus a state change.

Such clicks can be used for a history mechanism; the history of a state can be the sequence of all state-changing clicks that leaded to the state (starting from the initial state). To keep such clicks we need a simple format. Specifically, a click on a:

(i) class $c \in TM_{cl}(s)$ can be represented by the string: "`CL c`",

(ii) value $v \in Joins(s.e, p)$ can be represented by the string: "`PR p:v`",

(iii) class c of the values in $Joins(s.e, p)$ (recall complex transition markers), can be represented by the string: "`PR p:CL c`".

We can replace p by a property path $p1.p2.\ldots.pk$, to capture transition markers

that correspond to property paths.

(iv) a request for *entity switch* over the values in $Joins(s.e, p)$ can be represented by the string: "`SWITCH p`",
over a class c of the values in $Joins(s.e, p)$ can be represented by the string: "`SWITCH p CL c`",
over the values reachable through a property path pp can be represented by the string: "`SWITCH` *pp*",
or over the values reachable through a property path pp restricted by a class c can be represented by the string: "`SWITCH` *pp* `CL c`".

If we want for each state to keep its history, then each state s can have a list of strings denoted by $s.hist$, and whenever the user clicks on a transition marker v then the history of targeting state s' becomes $s'.hist = s.hist + Click(v)$, where "+" appends a new string at the end of the list and $Click(v)$ denotes the string that represents the click according to the format we described above.

3.5.2 State-preserving Clicks.

These clicks expand (or shrink) one class or a property path, thus changing the transition markers which are visible in the current state. We shall assume the following set of symbols/icons $\{\boxplus, \boxminus, \blacktriangleright, \blacktriangleleft\}$ since they are quite common (e.g. see Fig. 3.2). Specifically in this category we can have clicks on the:
$\boxplus$ of a class c whose outcome is to show the transition markers which are direct subclasses of c,
$\boxminus$ of a class c whose outcome is to hide the tms which are sublasses of c,
$\blacktriangleright$ of a property for expanding the property path with one more property,
$\blacktriangleleft$ for shrinking a property path.

Although such clicks do not change the current state, we have to keep them in order to restore a past state and its visible transition markers. Let $s.vis$ denote the sequence of such clicks, where each click is represented by a pair in $SPC = \{\boxplus, \boxminus, \blacktriangleright, \blacktriangleleft\} \times X$ where an element of X can be the string that we denoted by $Click(v)$ before.

If we want to retain the visibility of transition markers when the user changes states (i.e. those which are applicable), then we can set $s'.vis = s.vis$. This can be exploited for not letting the user to accomplish redundant steps. For example consider the case of Fig. 3.8 and assume that $s.e = \{CR1, CR2, CR3\}$.

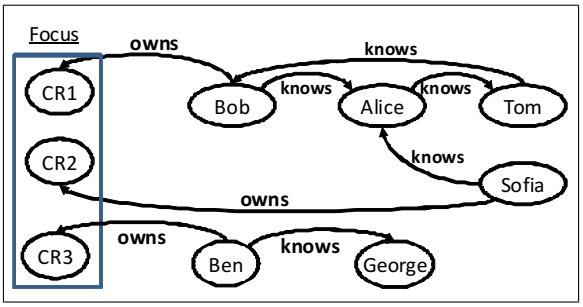

Figure 3.8: Instance cycles example

Suppose the user expands the property path $inv(owns).knows$ and clicks on the transition marker *Alice*, leading to a new state s' s.t. $s'.e = \{CR1, CR2\}$. If the new state does not have the property path $inv(owns).knows$ expanded, then the user could expand again the same property path and select *Alice*, leading to an extensionally equivalent state s''.

3.6 Caching and State Identity

Caching can be used to save computational cost and speed-up the interaction (for a single user or several users). Note that the existing caching techniques have been focused on query-and-answer process (e.g. *result caching* [71]), or *inverted list caching* [8]). Caching for speeding up the exploratory process has not been studied. However session-based caching techniques are crucial for obtaining real-time exploration especially if the process contains time consuming tasks. We should also mention that some RDF triple-stores employ query caching mechanisms for improving the performance of triple-store based applications, and there is currently an interest on how the results of a cached query can be reused for answering subsequent queries [50].

A cache for exploratory searching can store information about states which have been visited. Specifically for each such state it can keep stored its extension as well as its

transition markers along with their counts. However this raises the question on what the *identity* of a state is. This is important since the identity of the state can be used as the *key* for storing and looking up/retrieving cached states. Some choices are discussed:

- *Extension (i.e. s.e).* This is not practical since $s.e$ can be too big to store or to perform comparison operators. Such policy could make sense only if the extension is small and the computation of the transition markers is expensive.

- *History (i.e. s.hist).* The history could be used as a key, however one shortcoming is that there are more than one transition paths that lead to the same state, that is a state having the same extension. This means that if the history is used as a key for caching then we will get less cache hits than those that would be correct (false negatives).

- *Intention.* Since each state essentially corresponds to a number of conditions, an internal format can be used to keep these conditions for enabling the identification of more or ideally all equivalent states. This is the more promising approach for caching. One approach to describe the intention of a state is to process the history of the state, and this is further analyzed in Section 3.8.

3.7 RDF/S Query Languages

We have defined the interaction model using only extensions, since the expression of the intention depends on the Query Language (QL), or the abstraction of the QL that one adopts. However, in many cases the underlying information source would be accessible through a particular QL, e.g. the triples can be stored in a triple-store, like SESAME[7] or Virtuoso[8], that offers a query language (e.g. RQL[39], or SPARQL [3]).

Table 3.2 shows the notations we have used for defining RDF browsing, and their expression in SPARQL. In this description we consider that the extension of the current state is stored in a temporary class with name `temp`. Note that if the underlying triple-store supports SPARQL VIEWS (i.e. named SPARQL queries), then `temp` can be a

[7] http://www.openrdf.org/
[8] http://docs.openlinksw.com/virtuoso/

view. However to the best of our knowledge the existing RDF triple-stores do no support views. We should mention here that [64] proposes extending SPARQL by namely binding variables to results of filter expressions and supporting views on RDF graphs as datasets for queries (by including CONSTRUCT queries in FROM clauses). In addition, RVL [47] is a language which have been proposed (not implemented though) for specifying views over RDF/S KBs. Alternatively, a query language that supports nested queries could be used. For instance, nSPARQL [56] that has recently been proposed, supports nested regural expressions to navigate RDF data. Also, it is expressive enough to capture the deductive rules of RDF/S by directly traversing the input graph.

Furthermore in Table 3.2 we assume that all inferred triples are stored in a DBMS. However, we should note that *Virtuoso* [21], which is a general purpose RDBMS, supports an extended SPARQL version with `subClassOf` and `subPropertyOf` inference at query level. This means that triples entailed by subclass or subproperty statements in an inference context (built from one or more graphs containing RDF schema triples) are not physically stored, but they are added to the result set during query answering. This is similar in spirit with [29] which proposes to query the Web of Linked Data by traversing RDF links during run-time. This means that the SPARQL expressions of Table 3.2 would not require another change except from defining the inference context.

[Query Construction Method]

Since views are not supported, here we describe a query construction method. The method takes as input the history *s.hist* of the current state (in the format introduced at Section 3.5.1) and returns a SPARQL query which is the intention of the current state. At the beginning the query starts with `select ?x`. The process consumes each entry of the history and extends the query as described at Table 3.3. Whenever we use a variable with name starting from `xN` we denote a new variable which has not already been used during the query construction process. The case of entity type switch is described below.

[Query Construction Method: Entity Type Switch]

Consider the query that returns the set of "red cars owned by persons working at CSD". This query could be yielded by a process that starts from cars, restricts the focus by a property restriction over `hasColour`, and then performs a property path restriction

Notation	Expression in SPARQL

$Restrict(E, p : vset)$, $vset = \{v_1, ..., v_k\}$	`select ?x`
	`where { ?x rdf:type ns:temp; ns:p ?V.`
	`Filter (?V = ns:v_1 ||...|| ?V = ns:v_k)}`

$Restrict(E, c)$	`select ?x` ` where { ?x rdf:type ns:temp; rdf:type ns:c.}`

$Joins(E, p)$, where $E = \{e_1, ..., e_k\}$	`select Distinct ?v`
	`where { ?x ns:p ?v. Filter (?x = ns:e_1 ||...||`
	`?x = ns:e_k)}`

$TM_{cl}(s)$ and counts	`select Distinct ?c count(*)`
	`where{?x rdf:type ?c; rdf:type ns:temp.}`
	`group by ?c`

$Props(s)$	`select Distinct ?p`
	`where{ {?x rdf:type ns:temp; ?p ?v.}`
	`UNION {?m rdf:type ns:temp. ?n ?p ?m. }}`

$Joins(s.e, p)$ and counts	`select Distinct ?v count(*)`
	`where{ ?x rdf:type ns:temp; ns:p ?v.}`
	`groupby ?v`

Table 3.2: SPARQL-expression of Notations for RDF Browsing

History Entry	SPARQL pattern
`CL c`	`(?x type c)`
`PR p:v`	`(?x p v)`
`PR p:CL c`	`(?x p ?xN) (?xN type c)`
`PR p1.p2...pk:v`	`(?x, p1, ?xN1)(?xN1, p2, ?xN2)....(?xNk-1,pk,v)`

Table 3.3: Query Generator

through the path `ownedby.worksat`. In SPARQL it can be expressed as:

```
SELECT ?x
WHERE {?x type cars.
       ?x hascolour red.
       ?x ownedby ?p.
       ?p worksat CSD. }
```

An entity type switch from *cars* to *persons* can be accomplished by the query:

```
SELECT ?p
WHERE{ ?x type cars.
       ?x hascolour red.
       ?x ownedby ?p.
       ?p worksat CSD. }
```

As stated earlier entity type switch can be achieved by turning the transition markers into focus. In our case the persons who belong to the answer of the second query, were the restricted tms (i.e. M_1' according to the notations of Section 3.2.2) during the property restriction `ownedby.worksat` and the selection of the value `CSD`. In terms of the query language this can be achieved by changing the variable in the SELECT clause, not the whole graph pattern, of the previously formulated query (i.e. by selecting the variable that correspond to the desired transition markers).

More specifically, and assuming the history format, in our example we would have an entry "`SWITCH` ***`ownedby`***". To derive the desired query we locate the corresponding pattern of the already formulated SPARQL query (i.e. here the pattern `?x ownedby ?p`) and we put the right variable of that pattern at the `SELECT` clause. As regards the case of "`SWITCH` ***`pp`*** `CL c`", it is treated as if it were "`SWITCH` ***`pp`***" since the predicate for the class restriction is already in the formulated query. Analogously, after having focused on "CSD employees who are owners of red cars", we can switch our focus to "the hometowns of CSD employees who are owners of red cars" with the following query:

```
SELECT ?town
WHERE{ ?x type cars.
       ?x hascolour red.
       ?x ownedby ?p.
       ?p worksat CSD.
       ?p bornIn ?town}
```

The hometowns were actually the tms of a property restriction over `bornIn`.

Regarding related work, [7] proposes the creation of dynamic views during user's navigation, that relies on the generation of an appropriate RQL query. An algorithm translates the user navigation into a set of schema triples and at each browsing step it attempts to compute the minimal RQL query.

In Humboldt [41], a command queue tracks all user interactions in a session and represents the implicit construction of a query without having to store intermediate result sets. When the user makes state-changing clicks the extension is computed by parsing the whole command queue and executing SPARQL queries on the data source.

The examples of entity type switch that we gave earlier are supported by [22]. Although that work explains how a query result set (the so-called *extent*) is represented in user interface terms, the implementation approach adopted for handling (keeping) a query result set is not discussed.

3.8 On History Reduction

In this Section we aim to define the intention, and thus a kind of identity, of the current state by exploiting its history. The objective is to achieve an intentional description, which does not contain any redundant condition with respect to history information as that described in Section 3.5.

History Reduction Rules

Given a state s with history $s.hist$, we can eliminate redundant elements of $s.hist$ by applying rules like those shown at Table 3.4. In particular these rules can be used to

reduce the subsessions of the history in case these are *single entity* subsessions. In such subsessions no entity switch is accomplished. Methodologically, we can first identify such subsessions, and then apply the rules to each one of them. Each rule consists of four parts. The first and the second part (the two columns under title "Sequence" in Table 3.4) are considered to be entries of the subsession. Regarding the rules $R1 - R4$ if the second part occurs after (not necessarily immediately after) the first, and the condition of the 4th column is satisfied, then we can delete from the sequence the first entry. In brief,
(R1) concerns class-based restriction and exploits the semantics of the `subClassOf` relationships,
(R2) concerns property-based restriction and exploits the semantics of the `subPropertyOf` relationships,
(R3) concerns property-based restriction and exploits the semantics of the `subClassOf` relationships, and
(R4) concerns property-based restriction and exploits the semantics of the classification relationships.

We can generalize and define a rule (R5) for reducing property path-based restrictions. Recall that a property path-based restriction is logged by a sequence of entries of the from "`PR: p:CL c`" or "`PR: p:v`". Let denote such an entry that has length k by $a = a_1 \cdot \ldots \cdot a_k$. The entry a can eliminate another (property path-based restriction) entry $b = b_1 \cdot \ldots \cdot b_k$, if each a_i is equal to b_i or a_i can eliminate b_i using the rules (R2), (R3) or (R4).

	Sequence		Condition	Reduced Sequence
(R1)	CL c	CL c'	$c' \leq c$	CL c'
(R2)	PR p:v	PR p':v	$p' \leq p$	PR p':v
(R3)	PR p:CL c	PR p:CL c'	$c' \leq c$	PR p:CL c'
(R4)	PR p:CL c	PR p:v	$v \in inst(c)$	PR p:v
(R5)	$a_1 \cdot \ldots \cdot a_k$	$b = b_1 \cdot \ldots \cdot b_k$	$\forall\ i \in 1..k\ a_i = b_i$ or a_i can eliminate b_i using the rules (R2), (R3), (R4).	$a_1 \cdot \ldots \cdot a_k$

Table 3.4: History Reduction Rules

We can iteratively apply these rules until no further entry can be deleted.

For instance suppose the following history entry:

CL Person
PR drives:CL Car
PR uses.madeby:CL Japanese
PR drives:CL Jeep
PR uses.madeby:Toyota.

By applying the reduction rules shown at table 3.4, we will get:

CL Person
PR drives:CL Jeep
PR drives.madeby:Toyota.

We have just seen how to reduce each single-entity subsession of the history.

The reduction of sessions that include entity type switches is more complex and depends on the implementation method adopted. If entity type switch has been implemented by changing the SELECT variable as described at Section 3.7, then the last `SWITCH` in the history should be taken into account.

If entity type switch is implemented using a *view*-based approach, then the problem of intention minimization (and thus intention-based state identification) can be reduced to the general problem of *query minimization* and *equivalence*. Query minimization can reduce the history, while in case the cache stores the original (and unreduced) history, then query equivalence should be checked during cache lookups. Clearly such methods depend on the QL adopted. Regarding related work, [66] proposes query minimization techniques with respect to RQL while [65] studies query optimization aspects for SPARQL. We should also stress here that SPARQL does not support nested queries, a mechanism that would be useful for implementing entity type switch. Only recently, some extensions of SPARQL supporting nested queries have been proposed in literature [56, 6]. In general, we could say that the associated problems are subject of research that there are no mature and widely adopted/tested techniques.

3.8.1 On Cycles

The mechanism for detecting redundant transitions (in case of cyclic property paths) cannot be reduced to evaluation of a single query as it relies on property traversals of variable length. Such traversals can be done in main memory (over a graph based model of the involved part of the KB), or by adopting a rule-based language since with rules we can express the required recursion.

3.8.2 Client-Server Issues

Users would like to browse RDF KBs through their Web browsers. It follows that the RDF KB should be accessible through a Web application. An important point is that the application can be designed in a way where that client (web browser) at certain occasions receives more data than those requested by a user click. In this way some of the required processing (i.e. for computing transition markers and their counts, as well as for performing transitions) can be done at client side (i.e. using JavaScript). The benefit of this strategy is that the server receives less requests, and thus it has less load (consequently lower infrastructure costs are needed).

For instance, suppose the user is on a state s and clicks on a transition marker. The server instead of sending to the client the new state s' (and its transition markers), it can send all information required for all possible subsequent class-based restrictions of s'. In this way the client can undertake all clicks on these transition markers[9]. In general, chunks can contain property-based or property path-based subspaces.

3.9 Related Works on RDF/S

Here we review existing works in the light of the model described earlier.

[9]For example see the *labor intensive strategy* described at [61] for the case of faceted dynamic taxonomies over AHV information spaces.

3.9.1 Transitions Supported By Related Works

In this section we describe which transitions of the proposed interaction model are supported by existing systems. However, as we have already mentioned, the related literature uses a quite heterogeneous terminology. For this reason below at Table 3.5 we list some of the terms that we have used in this paper, and alternative terms which have been used in the literature.

Our terminology	Literature Terminology
transitions	*navigation modes* [22], *navigation links* [5]
transition markers	*zoom points* [55], *index term* [22], *restriction value* [54]
property-based browsing	*basic selection* [54]
property path-based browsing	*indirect facets* [17], *join selection* [54]
entity switch	*pivoting* [41], *refocusing* [35], *reversal* [5, 22], *path traversal* [28]
complex transition markers	*complex class* [22]

Table 3.5: Terminology

For each kind of transition, below we list the systems that support it (in a way). Table 3.6 provides an overview, where ✓ means yes, × no, and ? not specified.

Class-based restriction: /facet [33], Fuzzy view-based Search[34], Camelis2 [22], Ontogator [48], MuseumFinland [37], Faceted Data Explorer [20], GRQL [7]

Property-based restriction: /facet [33], Longwell[2], Camelis2 [22], Faceted Wikipedia [27], MediaFaces [78], GRQL [7]

Complex transition markers: To the best of our knowledge only the prototypes Odalisque [5] and Camelis2 [22] support the functionality of *complex transition markers*.

Property path-based restriction: BrowseRdf [54], NFB [35]. The *directional* behavior [76] of mspace [52], that automatically proposes the next property (and candidate values) after the selection of a particular property value, could act indirectly as such a mechanism. However, in such an interaction model, the candidate properties for property path formulation are pre-determined. However, none of the above works supports a mechanism

System	Supported Transitions					
	Class-based	Property-based	Complex Transition Markers	Property Path-based	ESwitch	Cycles
/facet	✓	✓	×	×	✓	×
Fuzzy view-based Search	✓	×	×	×	×	×
Ontogator	✓	×	×	×	×	×
MuseumFinland	✓	×	×	×	×	×
mspace	×	✓	×	✓	×	?
Faceted Data Explorer	✓	×	×	×	×	×
Longwell	×	✓	×	×	×	×
Camelis2	✓	✓	✓	×	✓	×
Faceted Wikipedia	×	✓	×	×	×	×
MediaFaces	×	✓	×	×	×	×
Odalisque	✓	✓	✓	×	✓	×
BrowseRDF	✓	✓	×	✓	×	×
NFB	×	✓	×	✓	✓	×
GRQL	✓	✓	×	×	✓	×
Humboldt	×	✓	×	×	✓	×
VisiNav	×	✓	×	×	✓	×
Parallax	×	✓	×	×	✓	×

Table 3.6: Supported Transitions by Existing Systems

to recognize cycles at schema or instance level and prevent user from making redundant steps.

Entity Type Switch: /facet [33], GRQL [7], Humboldt [41], VisiNav [28], Parallax [36], Odalisque [5], Camelis2 [22].

We should note that systems like Humboldt and Parallax provide users an indirect means to accomplish *property path-based* browsing: by combining the functionality of *property-based* transitions and *entity type switch*. However, this may prove cumbersome for end-users in case the focus entity type is identical with the targeting one (e.g. consider that the focus entity is Cars and the user wants to find Cars owned by persons over 30 years old, who also work at ICS). In such a case the user has to turn back to the initial focus entity type (Cars) after accomplishing *entity type switch* and *distant* restrictions. Although a history mechanism could help the user in accomplishing that, Clarkson, Navathe et al. [17] stress the need for flexible methods that allow expressing zoom-in conditions (refinements) based on properties which are distant from the current entity type, without having to change the entity type.

On Disjunction

Although in the proposed model the user provides plain single clicks, the interaction includes actions which have *disjunctive* nature. In particular, property paths, as well as complex transition markers (Eq. 3.3) are the means to express disjunction. Also, note that the semantics for `subclassOf/subpropertyOf` relations are disjunctive in nature (i.e. the set of instances of a class/property is the union of the instances of its subclasses/subproperties).

Also, it would not be difficult to provide the user with a means to select all possible transition markers for a given property. In that case a *property-based* transition would return all resources of the initial focus that have this property independently of its values. In literature this kind of transition is mentioned as *existential selection* [54]. Finally, as already explained, when an *entity type switch* takes place, we interpret disjunctively the elements associated with every object in the focus. For example, while viewing a set of cars we can move to and focus on the list of their firms. In this way we interpret *disjunctively* the elements associated with every object of the focus.

3.9.2 Linked Data Browsers

Linked Data browsers allow users to navigate between data sources by following links expressed at RDF triples. Browsers that support such a functionality are Tabulator [11], Disco[10] , Marbles[11](provenance of resources is also indicated) and the OpenLink Data Explorer [12], a browser extension for viewing Data Sources associated with Web Pages.

A common point of these browsers is that they support *one resource-to-one resource* transitions (navigation along in/outgoing links) while at each step of browsing only the directly connected resources are accessible. Thus these approaches do not exploit the structured organization of data for providing more sophisticated query capabilities. However, we should note that Tabulator allows the user to formulate queries in a query-by-example style. Specifically, the user can highlight a formulated pattern of interest, query for any similar patterns and further analyze the results through maps, timeline or other conventional data presentation methods.

3.9.3 Graphical Query Formulators

There are some tools, usually called *visual query systems*, that offer interactive and graphical methods to formulate SPARQL or SQL queries, e.g. NITELIGHT [59], SEWASIE [15]. Such systems aim at providing a graphical query formulation process per se, while the surveyed systems in this paper aim at supporting the *information seeking* process and satisfying *recall-oriented* information needs as an ongoing sequence of browsing actions. Essentially, these tools do not show an overview of the results of the partially formulated query. None of the systems we surveyed is geared towards graphical query formulation only.

3.9.4 Inference Materialization

In this section we investigate the approach that some related works follow for handling inferable information.

[10]http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/disco/
[11]http://beckr.org/marbles
[12]http://ode.openlinksw.com

In /facet [33], which is *a generic* browser, in order to bypass the prohibiting (for interaction purposes) slow-down caused by RDFS and OWL-based reasoning at run time, they compute and add closures of transitive and inverse properties to a triple store (supporting prolog-based querying) when new data is added.

Longwell [2] can access data in several triple storage systems (e.g. Jena, Joseki, 3Store, Sesame) but needs several configurations steps. Although it utilizes its built-in inference mechanism to adapt the views to the changing RDF content, it does not have the option of integrating an external inference engine and it does not use a RDF query language.

It emerges that there is not a clear correspondence in the level of required configuration of a browser (view-based vs generic) and the kind of inference implementation.

3.9.5 History Tracking and Caching Mechanisms

Some of the latest related works support history tracking mechanisms. For instance, Humboldt [41] displays the browsing history as a linear sequence of nodes, each representing an *entity-type* that has been traversed at an *entity-type switch*. When the user steps back to one of the earlier nodes and makes a refinement, then all other nodes are being refined appropriately (similar functionality is supported by Parallax [36]). Regarding history mechanism implementation, the core is a command queue which tracks all user interactions in a session. When state-changing clicks take place, the extension of the result state is computed by parsing the whole command queue and by executing SPARQL queries over the data source. In conclusion, all required information is stored in the command queue and there is no need for storing intermediate state extensions.

MultiBeeBrowse [44], a *generic* browser, allows users to keep track of the history of the supported transitions (including *refinements* and *entity switches*). A browsing session begins with a keyword query. The browser provides users with a structured means to access previous states and continue browsing after selecting one of them. Also, a caching mechanism allows users to review (in a time-based organization) all previous sessions, in which a particular query (user independent) has been invoked. The continuation of browsing after the selection of a particular session is supported. Regarding the implementation of mechanisms previously described, it is based on Service-Oriented Architecture

(SOA) paradigm.

Chapter 4

Exploration over Fuzzy Object Descriptions and Associations

In this Chapter we propose an interval-based extension of transition markers in order to support browsing and exploration over information spaces with fuzzy object descriptions and associations. Specifically, after giving some background information about fuzzy set theory then Section 4.2 proposes an extension of the *Faceted Dynamic Taxonomies (FDT)* interaction scheme, by refining the notion of transition markers (or *zoom points*). The proposed extension supports browsing over fuzzy and hierarchically organized attribute domains. Regarding *object-oriented* conceptual models, in Section 4.3 we consider Fuzzy RDF as the representation framework, and we propose a novel model for browsing and exploration over such sources. Finally, Section 4.4 reports some experimental results regarding efficiency.

4.1 Background

4.1.1 Fuzzy Set Theory

A fuzzy set A is defined by a membership function, which indicates the degree to which the element u of the universe of discourse U is included in the concept represented by A. Then a fuzzy set A over U is defined as a set of pairs given by:

$$A = (u_i, \mu(u_i)),$$

where $u_i \in U$, $\mu_A : U \rightarrow [0, 1]$ is the membership function of A and $\mu_A(u_i)$ is the membership degree of the element u_i to the fuzzy set A.

Standard fuzzy set operations are usually defined, after Zadeh [79], as:

- Standard complement: $\sim A(x) = 1 - A(x)$
- Standard intersection: $(A \cap B)(x) = min[A(x), B(x)]$
- Standard union: $(A \cup B)(x) = max[A(x), B(x)]$

where A(x) expresses the membership degree of x to A.

We can exploit fuzzy set theory in many ways. Some of them follow:
i) The definition of linguistic variables [81, 82] through membership functions. For example the linguistic variable "age" could have as linguistic values the following: *young, middle-aged, old*. Each linguistic value is represented by a fuzzy set with membership functions $\mu_{young}(u)$, $\mu_{middle-aged}(u)$, $\mu_{old}(u)$, where u in real positive numbers expresses the years of a person.
ii) As a means to define a similarity function to measure the similarity between two elements of a domain D of defined labels.
iii) In order to construct possibility distributions on the labels of a domain D, extending the possibilities for expressing imprecise values, in such a way that each value $d_i \in D$ has a degree of truth associated with it.

The fuzzy set theory has been successfully applied in various domains such as the fuzzy control, fuzzy diagnosis, fuzzy data analysis and fuzzy classification. Also it has been applied to several systems such as pattern recognition, knowledge systems (databases, expert systems etc).

4.2 Fuzzy Descriptions and Taxonomy-based Information Bases

This Section presents an extension of the *Faceted Dynamic Taxonomies (FDT)* interaction scheme for *exploratory search*, that permits browsing hierarchically organized attribute domains with fuzzy descriptions. It investigates how the *information thinning process* can leverage the fuzzy descriptions. Moreover, implementation requirements and approaches are being discussed.

4.2.1 Quantified Terms and their Semantics

In the context of Faceted Dynamic Taxonomies we call *quantified term*, for short *qterm*, any pair of the form (t, d) where t is a term and $d \in (0, 1]$. At this point we should remind that the notion of *term* corresponds straightforwardly to that of *transition marker*. An object can be described by associating it with one or more qterms. If (t, d) is associated with o we shall write $directDegree(o, t) = d$. Now we will discuss the semantics of such degrees, the semantics of the taxonomic relationships, and the consequences of these two.

Consider an object o described by two qterms (t_1, d_1) and (t_2, d_2). Now suppose that both t_1 and t_2 are the only narrower terms of a term t', i.e. $t_1 < t'$ and $t_2 < t'$ hold. The first question is what is the degree d' of o with respect to t'. In other words, what is the degree of o with respect to $t_1 \vee t_2$ (since $\bar{I}(t') = I(t_1) \cup I(t_2)$). The second question is what is the degree d_{12} of o with respect to $t_1 \wedge t_2$. To answer the above questions we need to know the semantics of the degrees. We can distinguish two basic interpretations:

Fuzzy Interpretation. Such degrees express the strength of an association, and can capture various application-specific semantics, such as *relevance*, *precision*, *certainty*, *trust*, etc. According to the well known Fuzzy Set Theory by Zadeh [80], disjunction corresponds to max and conjunction to min. It follows that in our case we have $d' = max(d_1, d_2)$ and $d_{12} = min(d_1, d_2)$.

Probabilistic Interpretation. If t_1 and t_2 correspond to disjoint elements of the sample space, and d_1 and d_2 are their probabilities, then $d' = d_1 + d_2$. If they are not disjoint, then we need to know the probability of their conjunction for computing d' (i.e.

$d' = d_1 + d_2 - P(t_1 \wedge t_2)$). Regarding conjunctions, if t_1 and t_2 are independent elements of the sample space then $d_{12} = d_1 * d_2$. If they are not independent then we need to know the conditional probability i.e. $d_{12} = d_1 * P(t_2|t_1) = d_2 * P(t_1|t_2)$.

It follows that if the taxonomies have been formed in a way that the children of a term denote disjoint elements of the sample space then we can compute d'. Analogously, if the facet terminologies correspond to probabilistically independent elements of the sample space, then we can compute the degrees that correspond to conjunctions (e.g. d_{12}). Now consider the case where d_1 and d_2 are *frequencies* (normalized or not). If t_1 and t_2 correspond to disjoint elements of the sample space then $d' = d_1 + d_2$. If not, then we need to know the frequency of their conjunction. It is not hard to see that frequencies have the same requirements (regarding disjointness and independence) with probabilities.

Hereafter we shall focus on the fuzzy interpretation since it captures several application scenarios and does not require extra sources of information for computing the degrees of term disjunctions and conjunctions.

4.2.2 Fuzzy Taxonomy-based Information Bases

Here we describe information sources that consist of *fuzzy object descriptions*. We call *fuzzy term*, for short *fterm*, any pair of the form (t, d) where t is a term and $d \in (0, 1]$ that has a fuzzy interpretation. There is a function $\mathbb{D}$ that associates to each object o one or more fuzzy terms. If (t, d) is associated with o (i.e. if $(t, d) \in \mathbb{D}(o)$), we shall write $directDegree(o, t) = d$. The *interpretation of a fuzzy term* (t, d), denoted by $J(t, d)$, consists all those objects which are described by a pair (t, d') and $d \leq d'$. For example if we have two objects the first described by $(red, 0.6)$ and the second with $(red, 0.8)$ then only the second belongs to the interpretation of the fuzzy term $(red, 0.7)$.

The *model interpretation of a fuzzy term* (t, d), denoted by $\bar{J}(t, d)$, is defined according to the standard semantics of fuzzy set theory. Specifically, the degree of membership of an object o to a term t, denoted by $degree(o, t)$, is the maximum degree with which o belongs to t or to one of the narrower terms of t, i.e. we can write $degree(o, t) =$

$max\{directDegree(o,t') \mid t' \leq t\}$. We can therefore define $\bar{J}(t,d)$ as follows:

$$\begin{aligned}\bar{J}(t,d) &= \{\ o \in Obj \mid degree(o,t) \geq d\} \\ &= \{\ o \in Obj \mid \exists (t',g) \in \mathbb{D}(o)\ \ and\ d \leq g\ and\ t' \leq t\}\end{aligned}$$

Recall that $\bar{I}(t)$ is the union of all $I(t')$ where $t' \leq t$. It is not hard to see that it holds $\bar{J}(t,d) = \bigcup_{t' \leq t} J(t',d)$. Now the model interpretation of a conjunction of terms is again defined according to the semantics of fuzzy set theory. Specifically, the degree of membership of an object o to a conjunction of terms t is the minimum degree with which o belongs to each term t of the conjunction. Recall that $\bar{I}(t_1 \wedge \ldots \wedge t_k) = \cap_{i=1}^{k} \bar{I}(t_i)$. The degree of membership of an object o to $\bar{I}(t_1 \wedge \ldots \wedge t_k)$ is the minimum degree of membership of o to $\bar{I}(t_i)$, $i = 1..k$. Table 4.1 summarizes the above notions and notations.

Name	**Notation**	**Definition**
terminology	T	a set of names, called *terms* (they may capture both categorical and numeric values)
subsumption	$\leq$	a partial order (reflexive, transitive and antisymmetric)
taxonomy	$(T, \leq)$	T is a terminology, $\leq$ a subsumption relation over T
faceted taxonomy	$\mathcal{F}$= $\{F_1, ..., F_k\}$	$F_i = (T_i, \leq_i)$, for $i = 1, ..., k$ and all T_i are disjoint
object domain	Obj	any denumerable set of objects
fuzzy terms	B	$T \times [0,1]$
description of o	$\mathbb{D}$	any function $\mathbb{D}$: $Obj \rightarrow 2^B$
interpretation of T	I	$I(t) = \{\ o \mid \exists (t,g) \in \mathbb{D}(o)\}$
model induced by I	$\bar{I}$	$\bar{I}(t) = \cup\{I(t') \mid t' \leq t\}$
fuzzy interpretation	$J(t,d)$	$J(t,d) = \{\ o \mid \exists (t,g) \in \mathbb{D}(o)\ \ and\ d \leq g\}$
fuzzy model induced by J	$\bar{J}(t,d)$	$\bar{J}(t,d) = \{\ o \mid \exists (t',g) \in \mathbb{D}(o)\ \ and\ d \leq g\ and\ t' \leq t\}$

Table 4.1: Basic notions and notations for Fuzzy taxonomy-based sources

4.2.3 Exploration over Fuzzy Descriptions

This section describes how the *information thinning process* (described at Section 2.1.5) can leverage the fuzzy descriptions. Specifically we propose a new method for *extending and refining* the notion of zoom point. Finally we discuss how such degrees have been exploited for ordering a) *zoom points*, and b) *objects*.

4.2.3.1 On Refining Zoom Points with Fuzzy Counts

Here we refine and extend the notion of zoom point based on fuzzy degrees. The idea is to analyze the count information of each zoom point to more than one counts each corresponding to the objects whose membership degrees fall to a specified interval. For example, consider a zoom point "z(20)". We can present it as "z [low(10), medium(4), high(6)]" where "low" may correspond to degrees (0,0.3], "medium" to (0.3,0.6] and "high" to (0.6,1]. So the user has now three clicks (i.e. three focus refinement options) instead of one. Therefore the set of all possible foci is more, so the discrimination power of interaction increases.

Formally, let P be a partition of $(0, 1]$ to a number of intervals, i.e. $P = \{p_1, \ldots p_m\}$ where $p_i = [a_i, b_i]$ where $0 \leq a_i < b_i \leq 1$, $a_1 = 0$, $b_m = 1$ and $b_i = a_{i+1}$. We do not (for the moment) describe formally the boundaries (open or closed) of the intervals. A common assumption is that P forms a *partition*, i.e. its elements *cover* the interval (0,1], since this guarantees that all objects are accessible, and the elements of P are *pairwise disjoint* (so each object contributes to the count of one interval). For example, $P = \{(0, 0.5], (0.5, 0.8], (0.8, 1]\}$ is a partition. However, overlapping intervals could also be used and might be useful for some tasks [73]. Optionally each interval in P can also be associated with a name (linguistic variable), e.g. "low", "medium", "large".

Suppose a zoom point t. Instead of associating it with one count information we now have $|P|$ counts (where $|P|$ denotes the number of intervals) and each count is clickable and leads to those objects whose degree of membership d falls into the corresponding interval. Suppose the user is in a focus specified by a set of objects A and suppose that t is a zoom-point. If p_i is one interval of P then the objects that their $degree(o, t)$ falls in this specific interval are given by:

$$Objects(t, p_i) \quad = \quad \{o \in \bar{I}(t) \cap A \mid degree(o, t) \in p_i\}$$

The cardinality of objects with $degree(o, t)$ in p_i is given by $count(t, pi) = |Objects(t, p_i)|$ and this is what it is displayed. We shall call such counts, *fcounts*.

4.2.3.2 Application Scenarios

Here we sketch three application scenarios from different domains for demonstrating the interaction with fuzzy counts.

Recipes Info-Base Fig. 4.1 shows a source with three facets for describing recipes. The first facet (`Location Of Origin`) is hierarchically organized using part-of relationships. Every object (recipe) is indexed under only one term of this taxonomy and the descriptions of objects w.r.t. that facet are crisp. The second facet (`Taste`) is non-hierarchical, and multiple classification is allowed for that facet as well as fuzzy degrees. Finally, the third facet (`Nutrition Value`) has the same characteristics with the last one, except that it is hierarchical. The last facet has the more complex navigation requirements since it is fuzzy, multi-valued and hierarchically organized. Regarding the interaction scheme that we propose, each zoom-point is accompanied by three counts each corresponding to one degree interval (as shown in Fig. 4.2(a)).

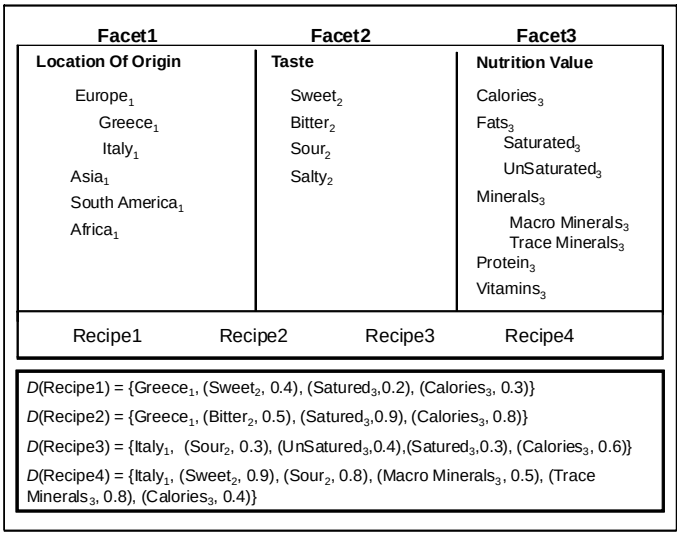

Figure 4.1: Recipes with fuzzy descriptions

Each count can be visualized using a different hue saturation for aiding the discrimination of the degree intervals (increased saturation denotes higher membership degree). Fig. 4.2(b) shows how the faceted taxonomy changes if the user clicks (zooms-in) on *fcount* "low" of term `Saturated`.

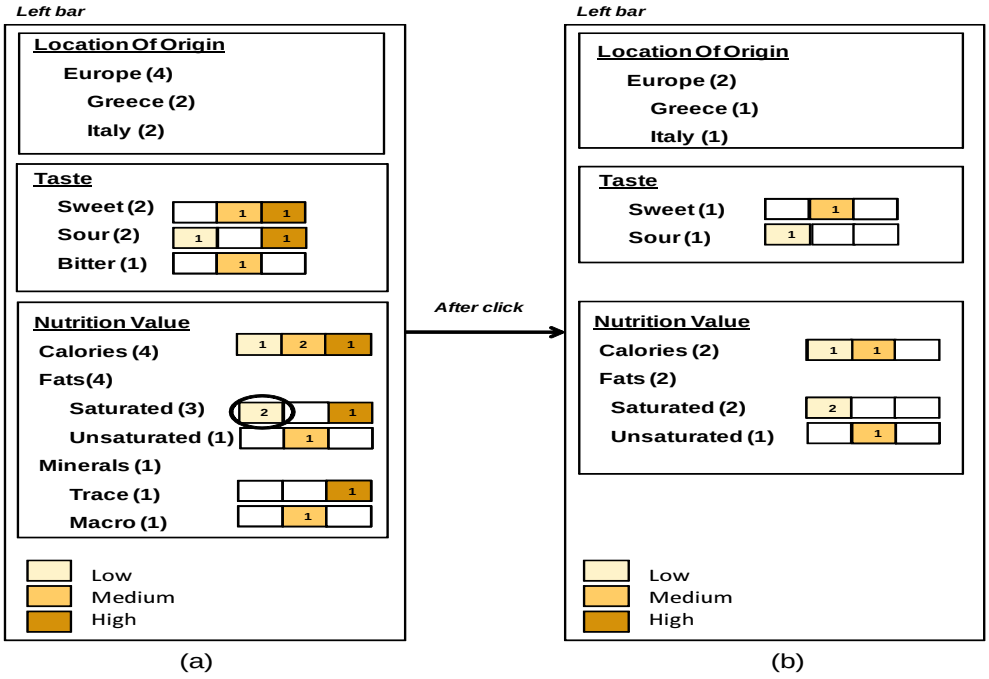

Figure 4.2: Interaction based on zoom points with fcounts (Example 1)

Actors Images The following scenario is inspired from W3C Incubator Group's *Report for uncertainty reasoning for the World Wide Web* and based on a real world industrial application scenario about casting for TV commercials and spots. Suppose a production company with an image database, like the one of CINEGRAM S.A.[1], containing a huge amount of vague data and a method to produce and query a fuzzy knowledge base (like that described in [67]). The company wants to publish on the Web this knowledge base [2] so that producers could easily find the most appropriate actors to employ by interacting through the FDT exploration scheme.

As an example, Fig. 4.3 shows a faceted taxonomy with 3 facets related to actors. The descriptions of images under concepts of Facet1 are crisp (it could be considered that $degree(o,t) = 1$) while fuzzy in the other two facets.

Regarding the interaction scheme that we propose, each zoom-point is accompanied by three counts each corresponding to one degree interval (as shown in Fig. 4.4(a)). Fig. 4.4(b) shows how the faceted taxonomies change if the user clicks on fcount "high" of

[1]http://www.cinegram.gr/
[2]In [62] it has been shown how a knowledge base can fit into the framework of dynamic taxonomies.

term Slim. In this way an expressive query like "*Find actors who are very slim*" can be formulated easily. If then the user selects the fcount "low" of term Aggresive, the formulated query becomes "*Find actors who are very slim and their faces are slightly aggressive*". We should stress here that from a set with 10^3 images the user has restricted his focus to 10 objects with 2 clicks.

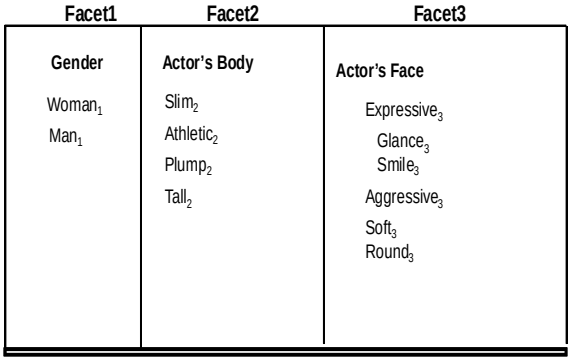

Figure 4.3: Actors related Faceted Taxonomy

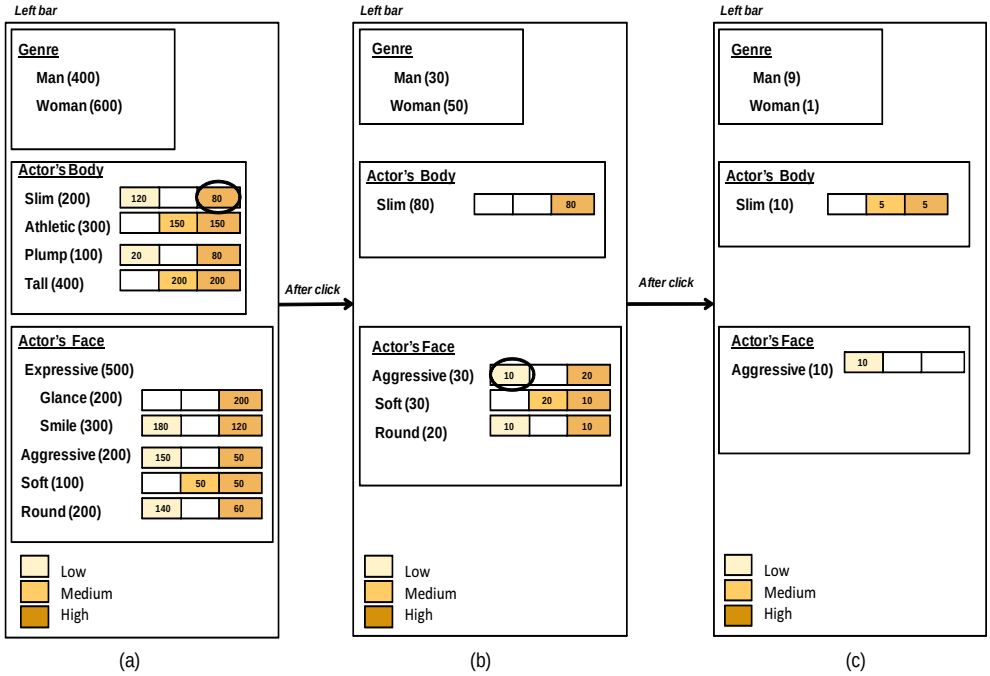

Figure 4.4: Interaction based on zoom points with fcounts (Example 2)

4.2.3.3 Plain Counts vs Fuzzy Counts

At first we should note, as discussed also in [74], that in order to evaluate exploratory search systems several metrics have to be examined simultaneously because such systems affect multiple aspects of the information seeking behavior. However, there are no standard collections and metrics for comparatively evaluating interaction information retrieval schemes[3].

Returning to the problem at hand, in our case we have to compare the interaction with plain counts versus the interaction with fuzzy counts. An aspect that has been used in the literature for comparing interaction schemes is the *discrimination power*, i.e. the set of queries that can be expressed by the navigation mechanism (e.g. [22]). Compared to plain counts, an advantage of fuzzy counts is that they correspond to *range queries over the membership degrees*, whose evaluation respects the fuzzy inference along the taxonomy hierarchy as explained in Section 4.2.2). Obviously this is not possible with plain counts. Furthermore, fuzzy counts allow formulating conjunctions over such queries (i.e. conjunctions of range queries over the membership degrees) through the session-based interaction scheme.

The discrimination power can also be quantified by counting the number of states. If we assume only conjunctions that consist of only one term from each T_i then the upper bound of the number of states in the plain FDT interaction is $B = |T_1| * \ldots * |T_k|$. The upper bound is reached when all conjunctions have non empty extension in $\bar{I}$ (we also ignore A-restrictions). According to our approach the upper bound is $B' = |T_1| * (|P| + 1) * \ldots * |T_k| * (|P| + 1) = B(|P| + 1)^k$, i.e. $(|P| + 1)^k$ times more states than the plain FDT interaction (e.g. if $|P| = 3$ and $k = 5$ then we get $45 = 1024$ times more states, i.e. three orders of magnitude more). If we assume that the intentional part of a state can be any conjunction of terms in T, where $T = T_1 \cup \ldots \cup T_K$, then the upper bound of states in plain FDT iteration is $2^{|T|}$. With fcounts the upper bound of states is $(|P| + 2)^{|T|}$ since each term can be absent, or it can be present, or it can participate with one of the $|P|$ intervals. This means that with fcounts we have $\left(\frac{|P|+2}{2}\right)^{|T|} = \left(1 + \frac{|P|}{2}\right)^{|T|}$ more states.

[3]In the IR domain, the most close track of TREC was the Interactive Track which was organized in 2002 (http://trec.nist.gov/data/t2002_interactive.html).

To summarize, the benefits of the proposed interaction scheme is that it (a) provides *richer and more informative overviews* of the information space regarding membership degrees, and (b) it greatly increases the *discrimination power of interaction*.

4.2.3.4 Ordering Zoom Points

We have just showed how to increase the discrimination power of the information thinning process, through fcounts. Apart from this, fuzzy degrees can be used for *ordering the zoom points* (which are usually shown at the left bar of a GUI).

In plain sources zoom points are usually ordered based on their count information. In fuzzy sources, they can be also ordered using metrics that exploit fuzzy degrees, e.g. by summing or averaging the degrees of the objects which are directly or indirectly associated with them. For instance, we can sum the degrees of memberships of the objects in the extension of a term t in order to get an "accumulated fuzzy value" for each zoom point. Specifically, if t is a zoom-in point and A is the current focus, then we define the *fuzzy value* of t by summing or averaging, i.e:

$$\begin{aligned} fvalue_{sum}(t) &= \sum_{o \in A \cap \bar{I}(t)} degree(o,t) \\ fvalue_{avg}(t) &= avg_{o \in A \cap \bar{I}(t)} degree(o,t) \end{aligned}$$

The application/user can order the zoom points of a facet in descending order with respect to these values. The decision between sum, avg or other operators (e.g. min, max) depends on the application, and therefore all of these options should be provided.

Several variations are also possible, and below we discuss those that have been proposed in the literature. For instance, in [42] the classification degrees of pages to topics (topics are what in this paper we call terms) are used for ordering the topics based on the number of pages that belong to them with a degree over a threshold θ:

$$fvalue_{avg,\theta}(t) = \frac{|\{\ o \in A \cap \bar{I}(t)\ |\ degree(o,t) \geq \theta\}|}{|A \cap \bar{I}(t)|}$$

We should stress at this point that the fcounts that we propose, allow even to a non-expert user to easily discriminate the objects having low from those having high degrees, without being asked to specify thresholds values.

Finally, in e-RARE [63], a shade of red indicates the average frequency of signs in the current set of diseases. Since signs are hierarchically organized, each entry has a color indication of the average frequency of all its descendant signs. However, as clarified in Section 4.2.2, in a probabilistic/frequency interpretation we may not have the information required for computing $degree(o,t)$. Probably, for this reason the author of [63] adopts a formula of the form:

$$\begin{aligned} fvalueDirect_{avg}(t) &= avg_{t' \leq t}\ tvalueDirect_{avg}(t') \text{ where} \\ tvalueDirect_{avg}(t) &= avg_{o \in \bar{I}(t) \cap A}(directDegree(o,t)) \end{aligned}$$

4.2.3.5 Ordering Objects

The ordering of the objects of the current focus is usually specified by an external access method (e.g. in the case of a WSE). However, while the user restricts his focus by clicking on various zoom points, the degrees of membership to the terms of the current focus could also be exploited for re-ordering objects. Regarding related work, in [63] the elements of the focus, which consists of diseases, are ordered by decreasing average frequency of symptoms in the focus. In this case fuzzy interpretation is not applicable, so the author of that work uses:

$$score_{ctx,avg}(o) = avg_{t \in ctx} directDegree(o,t)$$

Now in [46] the score of the objects depend on their fuzzy descriptions plus on weights provided by the users.

If we would like to adopt a general method, which is consistent with the semantics of fuzzy set theory, we could define:

$$Score_{ctx}(o) = min_{t \in ctx} degree(o,t)$$

That score could be used for ordering objects independently of the externally provided object ranking.

4.2.4 Implementation Requirements and Approaches

We will discuss implementation requirements and approaches comparatively to those for exploring non fuzzy taxonomy-based sources, for short plain sources, aiming at revealing the main performance trade-off. The key capability required for implementing the proposed model is that of traversals (due to taxonomies). We can dichotomize implementation approaches, on the basis of the assumed application scenario. In the first scenario updates are not frequent and emphasis is given on maximizing the speed of exploration services. At the other extreme, we have a scenario where the updates are frequent and emphasis is given on reducing the storage space and the effort/cost for maintaining the integrity of the data after updates. A general implementation approach for the first scenario is to materialize inferred information for avoiding traversals at run time at the cost of extra memory space and more costly updates. A general implementation approach for the second scenario is to avoid storing any inferable information so that updates are supported efficiently at the cost of less efficient exploration services (due to the required traversals).

For scenarios with rare-updates dedicated indexes have been proposed (e.g. [60, 10]). For scenarios with frequent updates DBMS have also been used (e.g. [77]). However, we should mention that the latter approach requires knowing the depth of the hierarchies, or adopting query languages that support recursion for computing the zoom points with one query (alternatively we can use *SQL with while*).

Returning to the sources at hand, i.e. fuzzy information bases, the aforementioned tradeoffs apply here as well. An implementation approach for a rare-updates scenario requires extending the physical index with fuzzy degrees and for providing fast exploration services it is beneficial to keep stored every $degree$ (instead of $directDegree$). On the other hand, a DBMS implementation approach for the frequent-update scenarios apart from extending the stored data with fuzzy counts, it requires tackling the fuzzy semantics (max for unions, min for conjunctions) using the supported query language. An issue here is whether the supported query language offers a straightforward way to express the required queries. Fuzzy extensions for relational databases have been studied (from [13], to [25] and [26]). Although there exist implementations of such languages (SQLf [13], FSQL[25])

[4] over PostgreSQL [49] and over Oracle [24], it is not referred to support recursive queries in combination with fulfilment thresholds at the query conditions concerning attribute values. An alternative approach to tackle the need for recursion and fuzziness is to exploit the reasoner of a deductive system that supports fuzziness. We have accomplished some experiments over such a system the results of which are presented in the Section 4.4.

4.3 Exploration over Fuzzy RDF

In Section 4.2 we have proposed an interval-based refinement of transitions markers for improving the information thinning process in case we have fuzzy (taxonomy-based) object descriptions. However, in case of *multi-entity* information spaces we could exploit degrees that are also applied over object associations. In this Section, we propose a generic interaction model for exploration over the representation framework of Fuzzy RDF.

4.3.1 Fuzzy RDF Background and Notations

A statement in *Fuzzy RDF* can describe simple facts where degrees (in $(0, 1]$) denote the truth value of the statement. In order to add such meta-statements about RDF triples, reification [1] is the only standardized mechanism. For instance, the statement "A topic of interest of Nikos is the Semantic Web with a degree of truth 0.9", would be represented via reification by the following triples (the subject in each case is "$\_ : stmt3$"):

```
_:stmt3
rdf:type  rdf:Statement ;
rdf:subject  ns:Nikos ;
rdf:predicate  foaf:topicOfinterest;
rdf:object  dbpedia:Semantic Web;
fuzzy:value  "0.9" xsd:decimal.
```

[4] More details for these languages and some query examples are given in the Appendix

The main advantage of the reification method is that no change to RDF data is required. However, it remains without a semantic specification meaning, e.g. reified statements are not affected by RDF/S inferences. Thereafter, such statements have to be appropriately managed by a deductive system according to the underlying theory. Finally, although there are recent proposals [69] on methods to extend fuzzy annotations in order to interplay with standards like RDF/S and SPARQL, there is not yet a standard general annotation framework.

Here we introduce some notations for Fuzzy RDF that will be used in the sequel. Each instance triple tr, either class or property instance triple, is accompanied by a degree expressing the truth value of the statement. Such a degree will be denoted by $directdegree(tr)$. We can now define the degree of a triple tr, denoted by $degree(tr)$, based on the semantics of RDF/S, and the axioms of Fuzzy Set Theory. We will adopt Zadeh's theory and consequently we shall use max for union and min for conjunctions [5]. Specifically,

$$\begin{aligned} degree(o, type, c) &= \max\{\ directdegree(o, type, c') \mid c' \leq_{cl} c\} \\ degree(o, p, o') &= \max\{\ directdegree(o, p', o') \mid p' \leq_{pr} p\} \end{aligned}$$

Let $\Phi = \{\varphi_1, \ldots \varphi_m\}$ be a set of intervals in [0,1]. We define:

$$\begin{aligned} inst(c, \varphi) &= \{\ o \in inst(c) \mid degree(o, type, c) \in \varphi\} \\ inst(p, \varphi) &= \{\ (o, p, o') \in inst(p) \mid degree(o, p, o') \in \varphi\} \end{aligned}$$

4.3.2 An Interaction Model for Fuzzy RDF

The general idea is that each transition of the interaction model (introduced in Chapter 3) is now analyzed into $|\Phi|$ transitions, one for each $\varphi \in \Phi$. Each one is signified by the count of the resultant state's extension. Fig. 4.5 shows how fuzzy degrees are exploited. For example, consider a transition marker "z(20)". We can present it as "z(20)[low(10), medium(4), high(6)]" where "low" may correspond to degrees (0,0.3], "medium" to (0.3,0.6] and "high" to (0.6,1]. So the user has now three more clicks, each corresponding to a transition.

[5] However one could also adopt alternative definitions for the operators $\bigotimes$, $\bigoplus$ (e.g. as proposed in [70]).

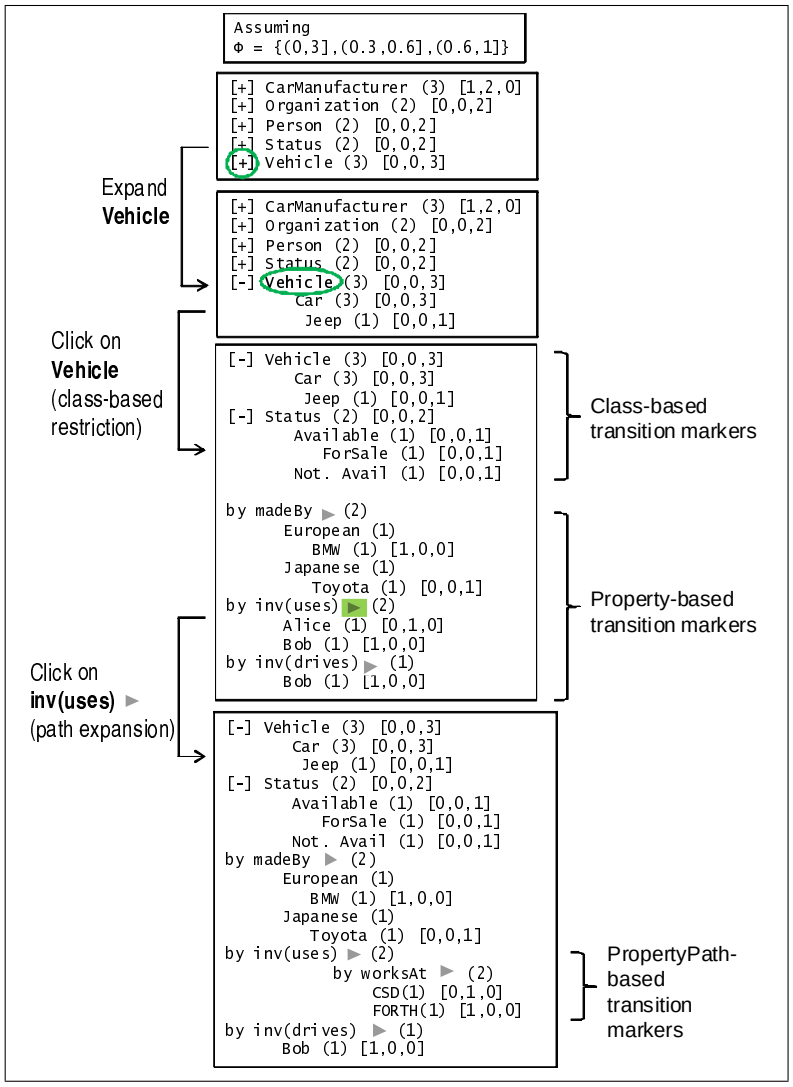

Figure 4.5: Sketch of the GUI part for transition markers

To define these transitions we extend the definitions given in Section 3.2 so that each of them takes an interval as additional parameter. Specifically, each *Restrict* takes as input an additional parameter φ, and the same for *Joins*, i.e.:

$$\begin{aligned} Restrict(E, c, \varphi) &= \{\ e \in E \mid e \in inst(c, \varphi)\} \\ Restrict(E, p : v, \varphi) &= \{\ e \in E \mid (e, p, v) \in inst(p, \varphi)\} \\ Restrict(E, p : vset, \varphi) &= \{\ e \in E \mid \exists\ v' \in vset \text{ and } (e, p, v') \in inst(p, \varphi)\} \\ Joins(E, p, \varphi) &= \{\ v \mid \exists\ e \in E \text{ and } (e, p, v) \in inst(p, \varphi)\} \end{aligned}$$

Regarding *class-based transitions*, it follows that for each tm in $TM_{cl}(s)$ we now have one $TM_{cl}(s, \varphi)$ for each $\varphi \in \Phi$, where:
$TM_{cl}(s, \varphi) = \{c \in C \mid Restrict(s.e, c, \varphi) \neq \emptyset\}$, and if the user clicks on a $c \in TM_{cl}(s, \varphi)$, then $s'.e = Restrict(s.e, c, \varphi)$.

Regarding *property-based transitions*, for each $p \in Props(s)$, the corresponding transition markers in plain RDF were defined by $Joins(s.e, p)$. Now, each element in $Joins(s.e, p)$ is analyzed to one $Joins(s.e, p, \varphi)$ for each $\varphi \in \Phi$. If the user clicks on a value v in $Joins(s.e, p, \varphi)$, then $s'.e = Restrict(s.e, p : v, \varphi)$. Regarding *presentation*, we do not show intervals, instead we show the corresponding count information. For example, for each $e \in E = \bigcup_{\varphi \in \Phi} TM_{cl}(s, \varphi)$ we show e once and its counts for each $\varphi \in \Phi$. Analogously for properties.

Let's now focus on *property paths*. For example consider two property instances pi_1 and pi_2 that form a path (e.g. `(cr2,inv(uses),Bob,0.2)` and `(Bob,worksAt,CSD,0.8)`), each associated with fuzzy degree d_1 and d_2, respectively. The degree of path $pi_1 \cdot pi_2$ is $\min(d_1, d_2)$, in our case 0.2, since each path actually corresponds to a conjunction. This means that if the user's focus is cars and he wants to restrict it through the organization of the users of the cars, then the path $pi_1 \cdot pi_2$ will be taken into account for computing the count of the transition marker `CSD` whose interval encloses the degree $\min(d_1, d_2)$. To define this precisely, we first introduce some notations. Let $pp = p_1, \ldots, p_k$ be a property path. An *instance path of pp* is a sequence of the form $ip = (v_0, p_1, v_1) \cdot \ldots \cdot (v_{k-1}, p_k, v_k)$ where for all $1 \leq i \leq k$: $(v_{i-1}, p_i, v_i) \in \mathcal{C}(K)$. The *degree* of an instance path ip is defined as the *minimum* degree of its edges (property instance triples). The *degree of a path from o to o'* over pp, denoted as $degree(o, pp, o')$, is the *maximum* degree of all instance paths

of *pp* between these two objects. We can now define *joins* and *restrictions* based on *fuzzy paths*:

$$Joins(E, pp, \varphi) = \{ v_k \mid \exists\, e \in E \text{ such that } degree(e, pp, v_k) \in \varphi \} \quad (4.1)$$

$$Restrict(E, pp : v_k, \varphi) = \{ e \in E \mid degree(e, pp, v_k) \in \varphi \} \quad (4.2)$$

Now we will analyze the algorithmic aspect of the above (since the previous two definitions were declarative). Consider a property path $pp = p_1 \cdot ... \cdot p_k$. The transition markers at each stage are defined as before, i.e. $M_i = Joins(M_{i-1}, p_i)$. For each individual element $e \in s.e$ we define the set of transition markers of level i (where $1 \leq i \leq k$) which are associated with it, as:

$$ETM_i(e) = \{m_i \in M_i \mid \exists\, s \in ETM_{i-1}(e) \text{ and } (s, p_i, m_i) \in inst(p_i)\} \quad (4.3)$$

assuming that $ETM_0(e) = \{e\}$.

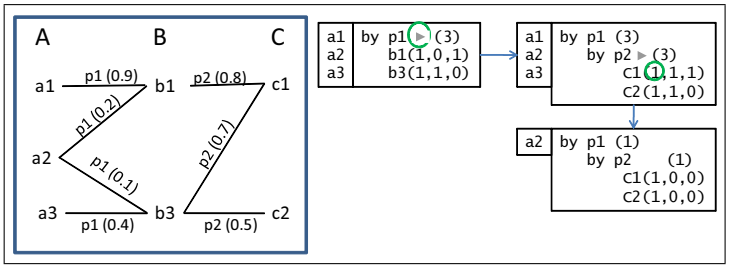

Figure 4.6: Interaction over fuzzy paths

For example consider the case shown at Fig. 4.6. Let A be the extension of the current state ($M_0 = A = s.e$). For a path consisting only of one property p_1 we have that $M_1 = Joins(M_0, p_1) = \{b1, b3\}$, while for $p_1 \cdot p_2$ we have $M_2 = Joins(M_1, p_2) = \{c1, c2\}$. Now, for each element $e \in s.e$ we have the following sets of transition markers of level i:

level 0:	$ETM_0(a1) = \{a1\}$	$ETM_0(a2) = \{a2\}$	$ETM_0(a3) = \{a3\}$
level 1:	$ETM_1(a1) = \{b1\}$	$ETM_1(a2) = \{b1, b3\}$	$ETM_1(a3) = \{b3\}$
level 2:	$ETM_2(a1) = \{c1\}$	$ETM_2(a2) = \{c1, c2\}$	$ETM_2(a3) = \{c1, c2\}$

In addition, for each element $e \in s.e$ and transition marker $m_i \in ETM_i(e)$, we introduce a value denoted by $Deg(e, m_i)$, which is actually the degree of a path from e to

m_i over pp (note that if pp is empty then we assume $Deg(e, e) = 1$). This value can be computed gradually (i.e. as the path gets expanded) as follows:

$$Deg(e, m_i) = \max_{m_{i-1} \in ETM_{i-1}(e)} \{\min(degree(m_{i-1}, p_i, m_i), Deg(e, m_{i-1}))\} \qquad (4.4)$$

In our example we have: $Deg(a2, b1) = \max_{a \in ETM_0(a2)}\{\min(degree(a, p_1, b1), Deg(a2, a))\} = \min(degree(a2, p_1, b1), Deg(a2, a2)) = 0.2$. Analogously, $Deg(a2, b3) = 0.1$. Now, the degree of $a2$ to the transition marker $c1$ is computed as:
$Deg(a2, c1) = \max_{b \in ETM_1(a2)}\{ \min(degree(b, p_2, c1), Deg(a2, b))\} = \max\{\min(degree(b1, p_2, c1), Deg(a2, b1)), \min(degree(b3, p_2, c1), Deg(a2, b3))\} = \max\{\min(0.8, 0.2), \min(0.7, 0.1)\} = \max\{0.2, 0.1\} = 0.2$. Analogously, $Deg(a1, c1) = 0.8$ and $Deg(a3, c1) = 0.4$.

Finally, the *count* for each m_i of M_i that corresponds to φ is given by:

$$count(m_i, \varphi) = |\{e \in s.e \mid Deg(e, m_i) \in \varphi\}| \qquad (4.5)$$

e.g. at Fig. 4.6 and for $\varphi = (0, 0.3]$, we have $count(c1, \varphi) = 1$. By clicking on the count $count(m_i, \varphi)$ the extension of the current state is restricted as follows:
$s'.e = \{e \in s.e \mid Deg(e, m_i) \in \varphi\}$.

4.3.3 Discrimination power

For each of the above transitions, the main advantage of the interval-based transition markers is that they increase the discrimination power of the interaction. We should remind here that each transition marker is an indication of the extra condition that could be added to the intentional part of the current state. For each such condition we have now $|\Phi|$ refinements. This means that for states corresponding to queries with k conditions we now have $|\Phi|^k$ more states.

For example consider the query *" Japanese cars for sale which are driven by persons who work at FORTH and know a person who knows Bob",* where each individual underlined part represents a condition. In this example we have $k = 6$ conditions. By considering that for each of them the possible refinements are $|\Phi| = 3$, then we would have $|\Phi|^k = 3^6$ more states. In conclusion, the interval-based transition markers enhance significantly the discrimination power of the interaction.

4.3.4 Implementation Requirements and Approaches

This section examines implementation requirements and approaches. The key aspect for implementing the model is that of traversals due to the `subClassOf` and `subPropertyOf` hierarchies. We can dichotomize implementation approaches, on the basis of the assumed application scenario. In Section 4.2.4, we have already described these scenarios (rare-updates vs frequent-updates) and discussed the related tradeoffs.

The aforementioned tradeoffs apply here as well. A general implementation approach for the rare-updates scenario is to materialize information inferred by RDF/S rules and fuzzy logic, for avoiding traversals at run time but at the cost of extra memory space and more costly updates. A general implementation approach for the frequent-updates scenario is to avoid storing any inferable information so that updates are supported efficiently at the cost of less efficient exploration services (due to the required traversals). In general, we can say that implementations are dichotomized to forward and backward chaining approaches.

4.3.4.1 Query Languages for Fuzzy RDF

As regards the frequent-updates scenario and in order to avoid storing inferred information, a query language for RDF/S is required to effectively accomplish inference at query level. Regarding QL for fuzzy RDF, there is not a standardized language widely used. However, authors of [9] have recently proposed a fuzzy extension of the RDF/S model and a query language named FSAQL to query fuzzy annotations defined under fuzzy RDF/S semantics. As an example, $Restrict(E, c, \varphi)$ can be expressed as:

```
SELECT ?X
FROM <Fuzzy RDFS repository>
WHERE n:<?X rdf:type ns:temp; rdf:type ns:c>
HAVE VALUES (n >= 0.8)
```

where we consider that the extension of the current state is stored under a class with name `ns:temp` and $degree(x, type, ns : temp) = 1$. To evaluate FSAQL queries it is not required

to compute the transitive closure of the RDFS graph but several rules are invoked based on the statements defined in the query.

Another proposed language is tSPARQL [30], an extension of SPARQL which presupposes a weighted RDF graph and permits accessing trust values in queries through *trust-aware* basic graph pattern matching. The trustworthiness of RDF triples is represented by a trust value in the interval [-1,1] where -1 corresponds to absolute "negative trust", 0 ignorance and 1 to absolute trust. In our context, we could use this machinery as follows: we use only positive values in the interval (0,1], and the instances of the temporary class have trust equal to 1 (i.e. $degree(x, type, ns : temp) = 1$). However, this QL does not support fuzzy inference wrt subclassOf/supropertyOf hierarchies at query level, meaning that $degree(o, type, c)$ has to be computed by an external method. As an example, $Restrict(E, c, \varphi)$ can be expressed as:

```
SELECT ?x
WHERE ?x rdf:type ns:temp; rdf:type ns:c.
ENSURE TRUST (phi.low, phi.up).
```

4.3.4.2 A Deductive Approach

As regards the rare-updates scenario, several works in literature treat the problem of querying fuzzy ontologies as a problem of storing fuzzy knowledge. One approach is to store fuzzy knowledge in existing storing systems and by exploiting a fuzzy reasoning engine. For instance, Simou et al. [67] propose a method based on the use of blank nodes in order to store membership degrees. Also they extend SPARQL in order to evaluate threshold queries defined in a fuzzy reasoning engine over the Sesame RDF store. In our context, the required SPARQL query for $Restrict(s.e, c, \phi)$ would be:

```
SELECT ?x
WHERE  ?x rdf:type C
?x ns:degreeC ?dom1
?x rdf:type Temp
?x ns:degreeTemp ?dom2
```

```
Filter (?dom1 >= "1.0" xsd:float)
Filter (?dom2 >= "0.7" xsd:float)
```

Moreover, some recently proposed deductive systems, e.g. [70, 69], support fuzzy answering over unions of conjunctive queries, by computing the closure of a fuzzy RDF graph, storing it into a relational database and then using internally SQL queries. That is, these systems exploit their reasoner for materializing all inferred triples in the underlying DBMS. For example, $degree(o, type, c)$ is computed as we have defined it, however $degree(o, p, o')$ is not directly supported.

More specifically, in *fuzzy RDF* system [70], the membership degrees are represented syntactically through RDF reification. Then the closure of the considered fuzzy RDF graph is computed and stored is MonetDB[6]. The underlying relational schema consists of the following tables:

- `type(subject, object,degree)`,
- `subclassOf(subject, object, degree)`,
- `subpropertyOf(subject, object, degree)`,
- `domain(subject, object,degree)`,
- `range(subject, object,degree)` and
- $prop_i$`(subject, object, degree)` ,where $prop_i$ is a table for every distinct property.

Table 4.2 shows directly the SQL queries that are needed by our interaction model for Fuzzy RDF.

Regarding *property paths* (as described in 4.3.2), at each step we can compute $M_i = Joins(M_{i-1}, p_i)$ with a single SQL query (as shown in Table 4.2). In case we consider that M_{i-1} is stored under a class $c1$ then the query would be:

```
SELECT p_i.object
FROM p_i,type
WHERE p_i.subject=type.subject AND type.object='C1'
```

[6]http://monetdb.cwi.nl/

Notation	Expression in SQL

$Restrict(E, c, \varphi)$	`select subject from type`
	`where object='c' and subject in E`
	`and degree>phi.low and degree<=phi.up`

$Restrict(E, p : vset, \varphi)$	`select subject from p_i`
	` where object in VSET and subject in E and`
	`degree>phi.low and degree<=phi.up`

$Joins(E, p)$	`select object from p_i` `where subject in E`

$TM_{cl}(s, \varphi)$ and counts	`select object,`
	`sum(case when degree>phi.lower`
	`and degree <= phi.upper then 1 else 0 end),`
	`from type`
	`where subject in s.e`
	`group by object`

$ETM(e, Prev)$	`select object from p_i` `where subject in Prev`

$Deg_{MIN}(e, subj, m_i, d)$, where $subj \in ETM_{i-1}(e)$ and $d = Deg(e, subj)$	`select case when degree > d`
	`then d else degree end as DEG_MIN`
	`from p_i`
	`where subject='subj' and object='m_i'`

Table 4.2: SQL-expression of Notations for Fuzzy RDF Browsing

The difference of fuzzy paths vs non fuzzy paths, is that for moving from a stage i of the path to a stage $i+1$, we have for each $e \in s.e$ to keep (a) $ETM_i(e)$, and (b) $Deg(e, m_i)$ for each $m_i \in ETM_i(e) \subseteq M_i$. To compute $ETM_i(e)$ we can use a query of the form $ETM(e, Prev)$ (shown in Table 4.2) where $Prev = ETM_{i-1}(e)$. For instance, the following simple query would return $ETM_1(e)$: `select object from` p_i `where subject='e'`. To compute $Deg(e, m_i)$ we can use a query of the form $Deg_{MIN}(e, subj, m_i, d)$ for every $subj \in ETM_{i-1}(e)$ (and accordingly $d = Deg(e, subj)$) and then get the max.

4.4 Experimental results

This Section describes some experiments accomplished over *fuzzy RDF* system [70]. Particularly, we compare the required computational cost for computing the count information of the "refined" transition markers [7] versus that of plain regarding the case of class-based transitions and property-based transitions as well. We show that the required computation cost is not prohibitive for real-time interaction even in case of large datasets.

4.4.1 Class-based browsing

Here we report experimental results that compare the required times to compute the plain counts versus fcounts corresponding to transition markers in case we only have class-based browsing.

In many practical cases, fuzzy descriptions are automatically extracted from datasets with plain attributes. Therefore, we have not localized a large dataset with fuzzy objects descriptions. For this reason we accomplished the following experiments using datasets synthetically generated. Specifically, we created a schema consisting of 209 classes forming a tree with depth 4. The top level of the hierarchy contains a single class, the second level 8 classes, the 3rd and 4th levels have 40 and 160 classes, respectively. The average number of direct subClassOf relationships is 8 for a class of level 1, 5 for a class of level 2 and 4 for a class of level 3. We have defined all subClassOf relationships as crisp (not fuzzy). Subsequently, we created 3 datasets (D1-D3) with 10, 10^2, 10^3

[7] We will use for short the term "fcounts".

resources each. Although we aimed at creating bigger datasets, when we created a dataset with 10^4 resources the JVM (over which the *fuzzy RDF* system runs) crashed during the computation of the closure (due to the high main memory requirements). We have to note that for each resource instance statement with a fuzzy degree, 5 triples are being added (by the Jena[8] model which is included in the implementation of *fuzzy RDF*) for reification purposes. So, each dataset (before the computation of closure) is represented by 50, $5*10^2$, $5 * 10^3$ triples while the schema constantly contains 208 triples. All resources have been categorized in a fuzzy manner and randomly under the class hierarchy just described. A resource can be instance of multiple leaf nodes.

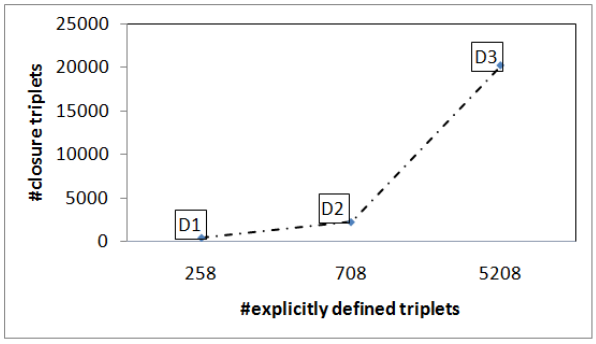

Figure 4.7: Sizes of datasets and their closures

The X-axis of Fig. 4.7 corresponds to the number of explicitly defined schema and instance triples of each dataset, while the Y-axis corresponds to the number of triples after the computation of closure. The measurements verify the theoretically derived result that the size of the closure of an RDF graph has a quadratic upper bound.

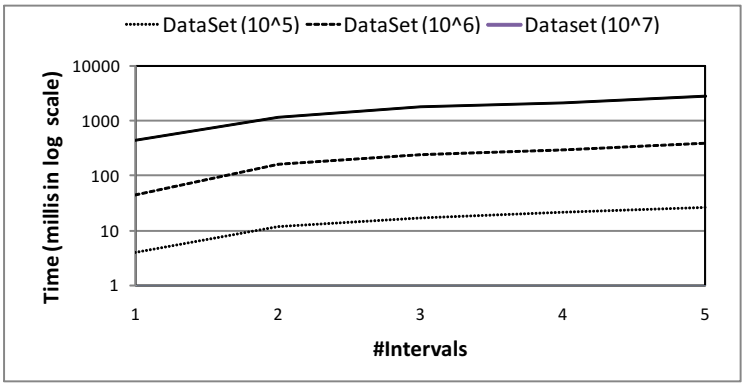

Figure 4.8: Query evaluation times for various $|\Phi|$ in bigger datasets (Y-axis in logscale)

[8]http://jena.sourceforge.net/

Subsequently we measured the times required to compute counts and fcounts. For plain counts we measured the time required to compute the counts of all classes. This means that we actually request all resources which are direct or indirect instances of a class (of the schema) with a degree in (0,1]. As regards fuzzy counts, we distinguish different cases each corresponding to a different number of intervals that partition the interval (0,1], i.e. to a different $|\Phi|$. It follows that a class gets $|\Phi|$ different fcounts. In our experiments we used the values from 2 to 5 for $|\Phi|$ and again we measured the time required to compute the fcounts for all classes of the schema. Although every query according to *fuzzy RDF* system's language, is interpreted to one SQL query that is executed over MonetDB (where the closure of the RDF graph is stored), we used directly MonetDB queries to get these results. We used one query to get either the plain or the fuzzy counts of all classes. Specifically for $|\Phi| = 1$:

```
SELECT object, count(*)
FROM type
GROUP BY object
```

while for $|\Phi| = 2$:

```
SELECT object,
     SUM(CASE WHEN degree < 0.5 OR degree = 0.5 THEN 1 ELSE 0 END),
     SUM(CASE WHEN degree > 0.5 THEN 1 ELSE 0 END)
FROM type
GROUP BY object
```

and so on. For each dataset, we posed the corresponding query (where $|\Phi|$ was ranging from 1 to 5) several times and measured the average query evaluation times. In all cases the times were less than 4 ms.

To bypass Jena's inability to compute the closure for big datasets (using limited main memory), we created bigger datasets (10^5, 10^6 and 10^7 instances) and stored them directly to MonetDB without first computing their closure. This allows us to test the efficiency of MonetDB in the above kinds of queries on larger datasets. The measurements (shown

at Fig. 4.8 in log scale) are representative for scenarios where the closure is already materialized, or for scenarios where the terms of the facets are flat (not hierarchically organized). We observe that the computation takes less than 0.4 secs for the 10^6 dataset and $|\Phi| = 5$. Also, for the 10^7 dataset it takes less than 3 secs for the case of $|\Phi| = 5$. Comparatively to the cost for computing plain counts ($|\Phi| = 1$), the cost for computing fcounts is around $|\Phi| * 1.75$ times greater in the case of one million objects and $|\Phi|$*1.3 in that of 10^7 dataset.

We also measured the time required to *order* all classes (that have instances) according to the average degree of their instances (i.e. $fvalue_{avg}(t)$, however recall that here we do not have hierarchically organized values), and at the same time to compute their *fcounts*. For $|\Phi|$=2 we use the following query:

```
SELECT AVG(degree) AS score, object,
     SUM(CASE WHEN degree < 0.5 OR degree = 0.5 THEN 1 ELSE 0 END),
     SUM(CASE WHEN degree > 0.5
     THEN 1 ELSE 0 END)
FROM type
GROUP BY object
ORDER BY score
```

We tested this type of queries over the 10^6 and 10^7 datasets. The required times (for $|\Phi|$ from 1 to 5) are almost equal to those measured in case of just computing fcounts (shown in Fig. 4.8). To synopsize, if degrees are materialized, then the required cost to compute fcounts is affordable for real-time interaction even over datasets with 10^7 resources.

4.4.2 Property-based browsing

We have also conducted an experiment over the *fuzzy RDF* system regarding property-based transition markers. Specifically, in this Section we present experimental results that compare the required times to compute the plain count information versus that of "refined" transition markers corresponding to a particular property.

In order to create a synthetic dataset, we consider a schema with 2 classes $(c1, c2)$ related through the property pr. As instances of the class $c1$ are considered to be all the resources in the current state extension, while instances of $c2$ are all the resources related with instances of $c1$ through property pr (all the candidate transition markers for the property pr). We consider that all possible inferred information (i.e. by RDF/S inference rules for subclassOf and subproperty hierarchies) has been pre-computed and stored in the database. Also, we should note that the degrees applied over property instances have been randomly selected.

Subsequently, we have created 2 datasets (D1,D2) with 10^5, 10^6 resources under class $c1$ in each dataset, respectively. All resources have been categorized in a fuzzy manner and randomly under the class hierarchy previously described. In both datasets, under $c2$ there have been added 3 instances, $tm1$, $tm2$ and $tm3$. We consider that each one is possible to restrict 1%, 10% and about 100% of the initial focus set (resources under $c1$) respectively. We should note that a particular resource under $c1$ is possible to be related with more than one transition markers.

For each resource instance statement (considering classes or properties) with a fuzzy degree, 5 triples are being added by the Jena model which is included in the implementation of the *fuzzy RDF* for reification purposes. Also degrees of truth are selected and applied randomly over property instances. At this point, we should also remind that the underlying relational schema in *fuzzy RDF* system keeps a table $prop_i$`(subject, object, degree)` for every distinct property. In our context, resources under $c1$ are possible *subjects* while $tm1 - tm3$ are the possible *objects*.

Then the required time to compute counts and fcounts could be measured. Regarding plain counts, we measured the required time to compute the number of objects in the focus set that are related with a particular transition marker through property $Prop$. As regards fuzzy counts, we distinguish different cases each corresponding to a different number of intervals that partition the interval $(0, 1]$. Each tm gets $|\Phi|$ different fcounts. It is possible to use one query to get either the plain or the fuzzy counts of all transition markers related with the focus set through property $Prop$ [9]. Specifically, for $|\Phi| = 1$:

[9]We consider that resources under c1 are stored in the *type* table.

```
SELECT prop.object, count(*)
FROM prop,type
WHERE prop.subject=type.subject AND type.object='c1'
GROUP BY prop.object
```

while for $|\Phi| = 2$:

```
SELECT prop.object
SUM (CASE WHEN prop.degree < 0.5 OR prop.degree=0.5 THEN 1 ELSE 0
END),
SUM (CASE WHEN prop.degree > 0.5 THEN 1 ELSE 0 END)
FROM prop,type
WHERE prop.subject=type.subject AND type.object='c1'
GROUP BY prop.object
```

and so on.

In order to get either the plain or the fuzzy counts of a particular transition marker the appropriate queries would for $|\Phi| = 1$:

```
SELECT prop.object, count(*)
FROM prop,type
WHERE prop.object='tm' AND prop.subject=type.subject
GROUP BY prop.object
```

while for $|\Phi| = 2$:

```
SELECT prop.object
SUM (CASE WHEN prop.degree < 0.5 OR Prop.degree=0.5 THEN 1 ELSE 0
END),
SUM (CASE WHEN prop.degree > 0.5 THEN 1 ELSE 0 END)
FROM prop,type
WHERE object='TMi' AND prop.subject=type.subject
GROUP BY prop.object
```

For each dataset, we posed the corresponding query (where $|\Phi|$ was ranging from 1 to 5) several times and measured the average query evaluation times. The measurements for the 10^5 and 10^6 datasets are shown at Fig. 4.9 and Fig. 4.10 respectively. Note that for each $|\Phi|$ there exist 3 bars, each representing the required computation time for each transition marker's counts and fcounts. We observe that for the 10^5 dataset the computation takes less than 0.3 secs in the worst case that a transition marker is related with almost 10^5 objects in the focus set and $|\Phi| = 5$. Moreover, for the same case it takes about 2 secs in the 10^6 dataset and about 3 secs if almost the whole initial focus set is restricted by a transition marker ($tm3$ case). We observe that as $|\Phi|$ changes, the computation times considering a transition marker's fcounts in a particular dataset, do not change remarkably.

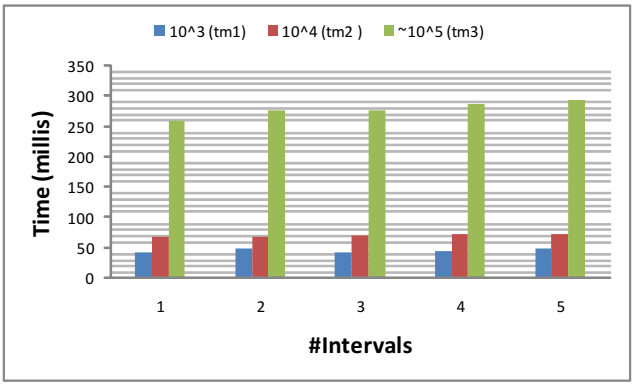

Figure 4.9: Query evaluation times for various $|\Phi|$ over the 10^5 dataset

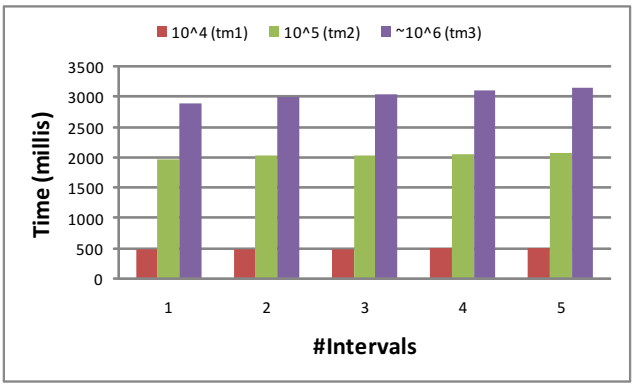

Figure 4.10: Query evaluation times for various $|\Phi|$ over the 10^6 dataset

In all previous queries we have compared the required times to compute counts versus fcounts for transition markers corresponding to a particular property considering that the

availability of this property is a priori known. However in practice, given a focus set, it would be required to find the appropriate properties (properties in $Props(s)$) that can be used for deriving transition markers. We remind that in the *fuzzy RDF* system, the underlying relational schema contains a table `domain(subject,object,degree)`, where subjects are properties and objects are their domain classes. Therefore, an appropriate query to get candidate properties would be the following:

```
SELECT domain.subject
FROM domain, type
WHERE type.object=domain.object
```

As this query does not contain any aggregation function, the required time to get a response would not be larger than that previously counted in the $|\Phi| = 1$ case.

Regarding now *property paths*, an important aspect is how we would keep M_i and $ETM_i(e)$ required for moving from a stage i of the path to a stage $i+1$ for the plain and fuzzy case, respectively. In order to testify if the *fuzzy RDF* system could be used to store the above in real-time interaction, we have measured the times required for inserting objects sets of various orders of magnitude in MonetDB (the underlying database of the *fuzzy RDF* system). The inserted resources are considered to be stored as instances of a temporary class. In order to accomplish the insertions, we have used a SQL query of the form: `INSERT INTO type VALUES ('subj','temp','Degree')`, where *subj* is a resource's identifier, *temp* the temporary class and $Degree = 1$. For the 10^2 object set, the total insertion time is 2 secs while for the 10^3 and 10^4 object sets the required times are about 22 secs and 242 secs, respectively, that are not affordable for real time interaction.

In conclusion, the required times for computing (over a DBMS) a particular transition marker's counts and fuzzy counts are not prohibitive for real-time interaction. However, storing in a DBMS the required sets of transition markers is not affordable for real-time interaction in case of large datasets. A practical solution is a *query construction method* (like the one proposed in Section 3.8) targetting to property paths, that could efficiently support the interaction model we have proposed as it overrides the requirements for storing intermediate sets of transition markers. The input of such a method would be the expanded property path while the output would be a SPARQL query that returns

the appropriate transition markers at each step. Alternatively, an extension of SPARQL that supports nested queries and regular expressions (i.e. nSPARQL [56]) could be used. However, there is not yet a consensus on how to define such a QL in the Semantic Web community.

Chapter 5

Conclusion

This thesis proposed a novel session-based model for browsing and exploring Fuzzy RDF knowledge bases. The proposed model allows formulating complex queries gradually and through plain clicks over interval-based transition markers. The contribution of this thesis lies in:

- proposing a model that it defines formally and precisely the state space of an interaction that is query language and visualization independent and never leads to empty result sets.
- clarifying issues regarding schema and instance cyclic property paths. Particularly, in Section 3.4 we have proposed an algorithmic way to treat instance cyclic property paths in order to eliminate possible redundant steps in sessions.
- analyzing the requirements that concern the implementation of the proposed model over the available query languages. Specifically, we have identified some important limitations of the current query languages (nested queries, transitive traversal of properties) and triple-stores (named queries, query minimization and equivalence). The advancement of existing QLs and triple-stores with such capabilities would allow a straightforward implementation of the proposed model and would greatly reduce the required application code.
- accomplishing and reporting experimental results regarding the efficiency and scalability of the proposed interaction method and in case all the inferred information

is stored in a DBMS. However, as regards the case of property path-based browsing, implementation issues have to be studied more thoroughly while more experiments (on efficiency) should be accomplished.

- surveying the main exploration/browsing approaches for plain RDF sources in terms of a state-based analysis.

Directions for further research regard:

- ranking methods for the transition markers. For instance, fuzzy degrees can be exploited for clustering transition markers, or for ranking them through more refined methods than those which have been proposed for plain RDF sources, e.g. [54, 23].
- the exploitation of fuzzy degrees applied at schema level (fuzzy subsumption relations) for even more efficient guided exploration.
- the adjustment of the model in order to be applicable over objects descriptions and associations accompanied by intervals of degrees instead of degrees.
- the refinements of transition markers by exploiting other domains of Annotated RDF/S [69], like the temporal one where instead of degree intervals we would have time intervals.

Bibliography

[1] Rdf semantics. `http://www.w3.org/TR/rdf-mt/`.

[2] SIMILE: Longwell RDF Browser (2003-2005). `http://simile.mit.edu/wiki/Longwell/`.

[3] SPARQL Query Language for RDF, W3C Candidate Recommendation, 15 January 2008 (http://www.w3.org/TR/rdf-sparql-query/).

[4] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *Procs. of CDIR*, 2003.

[5] P. Allard and S. Ferré. Dynamic taxonomies for the semantic web. In *Procs of the 19th International Conference on Database and Expert Systems Application (DEXA)*, 2008.

[6] R. Angles and C. Gutierrez. SQL Nested Queries in SPARQL. In *Procs of Alberto Mendelzon Workshop on Foundations of Databases, AMW 2010*, 2010.

[7] N. Athanasis, V. Christophides, and D. Kotzinos. Generating on the fly queries for the semantic web: The ics-forth graphical rql interface (grql). In *Intern. Semantic Web Conf. (ISWC)*, 2004.

[8] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri. Predictive caching and prefetching of query results in search engines. In *Procs of the 30th annual Intern. ACM SIGIR Conf. on Research and development in information retrieval, SIGIR 07*, Amsterdam, The Netherlands, July 2007.

[9] A. Bahri, R. Bouaziz, and F. Gargouri. Querying fuzzy rdfs semantic annotations. In *IEEE International Conference on Fuzzy Systems (FUZZ)*, 2010.

[10] O. Ben-Yitzhak, N. Golbandi, N. Har'El, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. Shekita, B. Sznajder, and S. Yogev. Beyond basic faceted search. *In WSDM '08: Proceedings of the International Conference on Web Search and Web Data Mining*, pages 33–44, 2008.

[11] Tim Berners-lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *In Procs of the 3rd International Semantic Web User Interaction Workshop (SWUI06)*, 2006.

[12] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 1981.

[13] P. Bosc and O. Pivert. Fuzzy queries and relational databases. In *SAC '94: Procs of the 1994 ACM symposium on Applied computing*, pages 170–174, NY, USA, 1994. ACM.

[14] Andrei Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[15] Tiziana Catarci, Tania Di Mascio, Enrico Franconi, Giuseppe Santucci, and Sergio Tessaris. An ontology based visual tool for query formulation support. In *OTM 2003Workshops*, pages 32–33. Springer Berlin / Heidelberg, 2003.

[16] S. Chaudhuri and L. Gravano. Evaluating top-k selection queries. In *Procs. of VLDB*, pages 399–410, 1999.

[17] Edward C. Clarkson, Shamkant B. Navathe, and James D. Foley. Generalized formal models for faceted user interfaces. In *JCDL '09: Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 125–134, New York, NY, USA, 2009. ACM.

[18] W. Dakka and P.G. Ipeirotis. Automatic extraction of useful facet hierarchies from text databases. In *Procs of the 24th Intern. Conf. on Data Engineering, (ICDE'08)*, pages 466–475, Cancún, México, April 2008.

[19] Debabrata Dash, Jun Rao, Nimrod Megiddo, Anastasia Ailamaki, and Guy Lohman. Dynamic faceted search for discovery-driven analysis. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 3–12, 2008.

[20] O. Erling and I. Mikhailov. Faceted views over large-scale linked data. In *Procs of the WWW2009 Workshop on Linked Data on the Web*, 2009.

[21] Orri Erling and Ivan Mikhailov. Rdf support in the virtuoso dbms. In *Procs of 1st Conf. on Social Semantic Web*, 2007.

[22] S. Ferré. Conceptual Navigation in RDF Graphs with SPARQL-Like Queries. *Formal Concept Analysis*, pages 193–208, 2010.

[23] N. Fuhr. A probability ranking principle for interactive information retrieval. *Information Retrieval*, 11(3):251–265, 2008.

[24] Galindo. FSQL (fuzzy SQL): A fuzzy query language. `http://www.lcc.uma.es/~ppgg/FSQL/#Ref`, April 2010.

[25] J. Galindo. *Fuzzy Databases: Modeling, Design, and Implementation*. IGI Publishing, 2006.

[26] Jose Galindo. *Handbook of Research on Fuzzy Information Processing in Databases*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2008.

[27] Rasmus Hahn, Christian Bizer, Christopher Sahnwaldt, Christian Herta, Scott Robinson, Michaela Bürgle, Holger Düwiger, and Ulrich Scheel. Faceted wikipedia search. In *Business Information Systems*, volume 47 of *Lecture Notes in Business Information Processing*, pages 1–11. 2010.

[28] A. Harth. Visinav: Visual web data search and navigation. In *Procs of the 20th Intern. Conf. on Database and Expert Systems Applications (DEXA '09)*, 2009.

[29] O. Hartig, C. Bizer, and J.-C. Freytag. Executing sparql queries over the web of linked data. In *Procs of the 8th Intern. Semantic Web Conference (ISWC '09)*. Springer, 2009.

[30] Olaf Hartig. Querying trust in rdf data with tsparql. In *6th Annual European Semantic Web Conference (ESWC09)*, pages 5–20, 2009.

[31] P. Hayes. RDF Semantics, W3C Recommendation, 2004.

[32] Marti A. Hearst. Uis for faceted navigation: Recent advances and remaining open problems. In *Workshop on Computer Interaction and Information Retrieval, HCIR 2008*, 2008.

[33] M. Hildebrand, J. Ossenbruggen, and L. Hardman. /facet: A browser for heterogeneous semantic web repositories. In *Procs of ISWC '06*, 2006.

[34] M. Holi and E. Hyvönen. Fuzzy view-based semantic search. In *ASWC*, 2006.

[35] D. Huynh. Nested faceted browsing. `http://people.csail.mit.edu/dfhuynh/projects/nfb/`, 2010.

[36] D. Huynh and D. Karger. Parallax and companion: Set-based browsing for the data web. *(submitted to WWW '09)*, 2009.

[37] E. Hyvönen, E. Mäkelä, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, and S. Kettula. MUSEUMFINLAND - Finnish Museums on the Semantic Web. *Journal of Web Semantics*, 3(2-3):224–241, 2005.

[38] Cengiz Kahraman. *Fuzzy Applications in Industrial Engineering (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[39] G. Karvounarakis, V. Christophides, and D. Plexousakis. RQL: A Declarative Query Language for RDF. In *Eleventh International World Wide Web Conference (WWW)*, Hawaii, USA, May 2002.

[40] S. Khatchadourian and M. Consens. Explod: Summary-based exploration of interlinking and rdf usage in the linked open data cloud. *The Semantic Web: Research and Applications*, pages 272–287, 2010.

[41] G. Kobilarov and I. Dickinson. Humboldt:Exploring linked data. In *Linked Data on the Web Workshop at WWW2008*, Beijing, China, 2008.

[42] Christian Kohlschütter. Using link analysis to identify aspects in faceted web search. *SIGIR 2006 Workshop on Faceted Search*, pages 55–59, 2006.

[43] S. Kopidaki, P. Papadakos, and Y. Tzitzikas. STC+ and NM-STC: Two novel online results clustering methods for web searching. In *WISE '09: Procs of the 10th Intern. Conf. on Web Information Systems Engineering*, October 2009.

[44] Sebastian Ryszard Kruk, Adam Gzella, Filip Czaja, Wladyslaw Bultrowicz, and Ewelina Kruk. Multibeebrowse: accessible browsing on unstructured metadata. In *Proceedings of the OTM - Volume Part I*, OTM'07, 2007.

[45] Chengkai Li, Ning Yan, Senjuti Basu Roy, Lekhendro Lisham, and Gautam Das. Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In *WWW*, pages 651–660, 2010.

[46] J. Lu, Y. Zhu, X. Zeng, L. Koehl, J. Ma, and G. Zhang. A fuzzy decision support system for garment new product development. In *Australasian Conf. on Artificial Intelligence*, pages 532–543, 2008.

[47] Aimilia Magkanaraki, Val Tannen, Vassilis Christophides, and Dimitris Plexousakis. Viewing the semantic web through rvl lenses. *Web Semant.*, 1, October 2004.

[48] E. Mäkelä, E. Hyvönen, and S. Saarela. Ontogator - A Semantic View-Based Search Engine Service for Web Applications. In *Procs of ISWC '06*, pages 847–860, 2006.

[49] J. Maraboli R., Abarzua. *FSQL-f representacion y consulta por medio del leguaje PL/PGSQL de informacion imperfecta.* PhD thesis, Universidad Catolica del Maule,Chile, 2006.

[50] Michael Martin, Jörg Unbehauen, and Sören Auer. Improving the performance of semantic web applications with sparql query caching. In *ESWC (2)*, pages 304–318, 2010.

[51] M. Mazzieri. A fuzzy rdf semantics to represent trust metadata. In *1st Workshop on Semantic Web Applications and Perspectives (SWAP2004)*, 2004.

[52] schraefel m.c, D.A. Smith, A. Owens, A. Rusell, C. Harris, and M.L. Wilson. The evolving mSpace platform: leveraging the Semantic Web on the Trial of the Memex. *In Proceedings of Hypertext 2005*, pages 174–183, 2005.

[53] Andreas Meier and Nicolas Werro. A fuzzy classification model for online customers. *Informatica (Slovenia)*, 31(2):175–182, 2007.

[54] E. Oren, R. Delbru, and S. Decker. Extending Faceted Navigation for RDF Data. In *Procs of ISWC '06*, 2006.

[55] P. Papadakos, S. Kopidaki, N. Armenatzoglou, and Y. Tzitzikas. Exploratory web searching with dynamic taxonomies and results clustering. In *ECDL '09: Proceedings of the 13th European Conference on Digital Libraries*, September 2009.

[56] J. Pérez, M. Arenas, and C. Gutierrez. nSPARQL: A navigational language for RDF. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2010.

[57] E. Pietriga, C. Bizer, D. Karger, and R. Lee. Fresnel - a browser-independent presentation vocabulary for rdf. In *Procs of the Second InterN. Workshop on Interaction Design and the Semantic Web*, pages 158–171. Springer, 2006.

[58] Daniel E. Rose and Danny Levinson. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, 2004.

[59] Alistair Russell, Paul R. Smart, Dave Braines, and Nigel R. Shadbolt. Nitelight: A graphical tool for semantic query construction. In *Semantic Web User Interaction Workshop (SWUI 2008)*, April 2008.

[60] G. M. Sacco. Efficient implementation of dynamic taxonomies. Technical report, Univ. di Torino, 2004.

[61] G. M. Sacco and Y. Tzitzikas (Editors). *Dynamic Taxonomies and Faceted Search: Theory, Practise and Experience*. Springer, 2009. ISBN = 978-3-642-02358-3.

[62] G. M. Sacco and Y. Tzitzikas. *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*. Springer, 2009.

[63] Giovanni Maria Sacco. e-rare: Interactive diagnostic assistance for rare diseases through dynamic taxonomies. In *DEXA Workshops*, pages 407–411. IEEE Computer Society, 2008.

[64] Simon Schenk. A sparql semantics based on datalog. In *KI 2007: Advances in Artificial Intelligence*, volume 4667 of *Lecture Notes in Computer Science*, pages 160–174. Springer Berlin / Heidelberg, 2007.

[65] M. Schmidt, M. Meier, and G. Lausen. Foundations of SPARQL query optimization. In *Proceedings of the 13th International Conference on Database Theory*, pages 4–33. ACM, 2010.

[66] G. Serfiotis, I. Koffina, V. Christophides, and V. Tannen. Containment and minimization of RDF/S query patterns. *The Semantic Web–ISWC 2005*, pages 607–623, 2005.

[67] N. Simou, G. Stoilos, V. Tzouvaras, G. Stamou, and S. Kollias. Storing and querying fuzzy knowledge in the semantic web. In *7th Int. Workshop on Uncertainty Reasoning For the Semantic Web*, 2008.

[68] M. Stefaner, T. Urban, and M. Seefelder. Elastic lists for facet browsing and resource analysis in the enterprise. In *Procs of the 19th Intern. Conf. on Database and Expert Systems Application (DEXA'08)*, pages 397–401, 2008.

[69] U. Straccia, N. Lopes, G. Lukacsy, and A. Polleres. A general framework for representing and reasoning with annotated semantic web data. In *Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2010)*, 2010.

[70] Umberto Straccia. A minimal deductive system for general fuzzy rdf. In *Procs of the 3rd Intern. Conf. on Web Reasoning and Rule Systems (RR '09)*, 2009.

[71] F. Silvestri T. Fagni, R. Perego and S. Orlando. Boosting the performance of web search engines Caching and prefetching query results by exploiting historical usage data. In *ACM Transactions on Information Systems (TOIS)*, pages 51–78, 2006.

[72] Y. Tzitzikas, N. Armenatzoglou, and P. Papadakos. FleXplorer: A Framework for Providing Faceted and Dynamic Taxonomy-Based Information Exploration. In *Database and Expert Systems Application, 2008. DEXA'08. 19th International Conference on*, pages 392–396, 2008.

[73] W. A. Voglozin, G. Raschia, L. Ughetto, and N. Mouaddib. Querying a summary of database. *J. Intell. Inf. Syst.*, 26(1):59–73, 2006.

[74] R. W. White and R. A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan & Claypool Publishers, 2009.

[75] Ryen W. White, Bill Kules, Steven M. Drucker, and m.c. schraefel. Supporting exploratory search, introduction, special issue, communications of the acm. *Communications of the ACM*, 49(4):36–39, April 2006.

[76] Max L. Wilson, Paul André, and mc schraefel. Backward highlighting: enhancing faceted search. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, 2008.

[77] K.-P Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. *In Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 401–408, 2003.

[78] Roelof Z., B. Sigurbjornsson, R. Adapala, L. Garcia Pueyo, A. Katiyar, K. Kurapati, M. Muralidharan, S. Muthu, V. Murdock, A. Ng, P.and Ramani, A. Sahai, S. T. Sathish, H. Vasudev, and U. Vuyyuru. Faceted exploration of image search results. In *WWW'10: Proceedings of the 19th international conference on World wide web*, 2010.

[79] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets Syst.*, pages 3–28, 1978.

[80] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[81] Lotfi A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning - i. *Inf. Sci.*, 8(3):199–249, 1975.

[82] Lotfi A. Zadeh. From computing with numbers to computing with words - from manipulation of measurements to manipulation of perceptions. In *Intelligent Systems and Soft Computing*, pages 3–40, 2000.

[83] J. Zhang and G. Marchionini. Evaluation and evolution of a browse and search interface: Relation browser++. In *Procs of the national conference on Digital government research (DG. '05)*. Digital Government Society of North America, 2005.

Chapter 6

Appendix

6.1 Fuzzy Queries over Relational DBs

Several fuzzy extensions for querying relational databases have been proposed in literature. A flexible querying process operating on relational databases searches the tuples for adequacy to a query using an extension of a standard language, usually SQL [73].

6.1.1 SQLf

The querying language SQLf [13], proposed by Bosc and Pivert, is a fuzzification of SQL allowing fuzzy queries over precise data producing discriminated answers. In SQLf, linguistic terms may appear as fuzzy values, modifiers (i.e. very, really, more), and quantifiers (i.e. most, a dozen) in the WHERE clause and other clauses. It includes extensions of the SQL standards until the SQL3 comprehending recursive queries too. However, SQLf allows users to specify desired thresholds for general satisfaction degree but not over a particular attribute. For instance the threshold degree 0.75 in the following query will determine totally which data will participate in the answer.

```
SELECT *
FROM PEOPLE
WHERE Weight = Weighted AND Stature = Tall WITH CALIBRATION 0.75 ;
```

6.1.2 FSQL

FSQL (Fuzzy SQL) [25], another fuzzy extension of the SQL language, supports queries over fuzzy data elements and fulfilment thresholds that can be applied at different levels of querying condition. SQL is extended to allow flexible conditions using linguistic labels, fuzzy comparison operators, fuzzy constants and many other fuzzy constructs. Each condition in a query can be given a threshold that sets the minimum satisfaction degree for the condition. The following query follows the syntax of FSQL:

```
SELECT *
FROM PEOPLE
WHERE Weight FEQ Weighted 0.75 AND Stature FEQ Tall 0.75 ;
```

Although there exist implementations of such languages (SQLf, FSQL) over PostgreSQL [49] and over Oracle [24], it is not referred to support recursive queries in combination with fulfilment thresholds at the query conditions concerning attributes values.