Towards a Generalized Interaction Scheme for Information Access

Yannis Tzitzikas[1], Carlo Meghini[2], and Nicolas Spyratos[3]

[1] *Information Technology, VTT Technical Research Centre of Finland*
ext-yannis.tzitzikas@vtt.fi
[2] *Istituto di Scienza e Tecnologie dell' Informazione [ISTI], CNR, Pisa, Italy*
meghini@isti.cnr.it
[3] *Laboratoire de Recherche en Informatique, Universite de Paris-Sud, France*
spyratos@lri.fr

Abstract.
We introduce the formal framework of a generalized interaction scheme for information access between users and information sources. Within this framework we describe an *interaction manager* which supports more complex interaction schemes than those that are supported by existing systems, including: query by example, answer enlargement/reduction, query relaxation/restriction, index relaxation/contraction, "relevance" feedback, and adaptation facilities. We give the foundations of this interaction manager from a mathematical point of view, in terms of an abstract view of an information source.

1 Introduction

Information sources such as information retrieval systems [2], or databases and knowledge bases [12], aim at organizing and storing information in a way that allows users to retrieve it in a flexible and efficient manner. Commonly, for retrieving the desired information from an information source, the user has to use the query language that is provided by the system.

We propose an interaction scheme whose objective is to make the desired objects easy to find for the user, even if the source has a query language which is *unknown* to the user. This scheme is actually a generalization of the interaction schemes that are currently used by information systems. In particular, we describe an *interaction manager* which supports several kinds of interaction, including: query by example, index relaxation/contraction, query relaxation/restriction, answer enlargement/reduction, "relevance" feedback, and adaptation facilities, in a uniform manner.

We view the interaction of a user with the information source as a sequence of *transitions* between *contexts* where a context is a consistent "interaction state".

The user has at his/her disposal several ways to express the desired transition. Then, it is the interaction manager that has to find (and drive the user to) the new context. Methods allowing the user to specify a transition relatively to the current context are also provided. Furthermore, we describe methods for restricting the set of transitions to those that can indeed lead to a context. As we shall see below, the unified interaction scheme that we introduce allows defining more complex interaction mechanisms than those that are supported by existing systems. We describe this scheme in terms of an abstract view of a source.

The paper is organized as follows. Section 2 introduces contexts and context transitions. Section 3 describes how context transitions can be specified using replacements. Section 4 describes how the interaction manager can find a new context after a context transition specification. Section 5 introduces relative replacements and Section 6 describes context transitions using restricted relative replacements. Section 7 concludes the paper and identifies issues for further research.

2 A Context-based Interaction Scheme

We view a source S as a function $S : Q \rightarrow \mathcal{A}$ where Q is the set of all queries that S can answer, and $\mathcal{A}$ is the set of all answers to those queries, i.e. $\mathcal{A}$=$\{ S(q) \mid q \in Q\}$. As we focus on retrieval queries, we assume that $\mathcal{A}$ is a subset of $\mathcal{P}(Obj)$, the powerset of Obj, where Obj is the set of all objects stored at the source.

Let $\mathcal{S}$ be the set of all sources that can be derived by "updating" the source S (e.g. for adapting it), but for the moment let us suppose that $\mathcal{S}$ is the set of all functions from Q to $\mathcal{P}(Obj)$.

Let $\mathcal{U}$ denote the set of all triples in $\mathcal{S} \times Q \times \mathcal{A}$, i.e. $\mathcal{U} = \mathcal{S} \times Q \times \mathcal{A}$. A triple $c = (S, q, A) \in \mathcal{U}$ is called an interaction context, or *context* for short, if $S(q) = A$. Let $\mathcal{C}$ denote the set of all contexts, i.e. $\mathcal{C}$= $\{ (S, q, A) \in \mathcal{U} \mid S(q) = A\}$. Given a context $c = (S, q, A)$, S is called the *source view* of c, q is called the *query* of c and A is called the *answer* of c. The interaction between the user and the source is carried out by a software module called *Interaction Manager* (IM). We view the interaction of a user with the source as a sequence of *transitions* between contexts. At any given time, the user is in one context, the *focal context*. At the beginning of the interaction with the system the user starts from the initial context $(S, \epsilon, \emptyset)$ where S is the stored information source, ϵ is the empty query and $\emptyset$ is the empty answer[1]. There are several methods the user can use for moving from one context to another, i.e. for changing the focal context. For example, in the traditional query-and-answer interaction scheme, the user actually "replaces" the current query q by formulating a new query q' and the "interaction manager" drives him to the new context (S, q', A') where $A' = S(q')$.

[1] We assume that $S(\epsilon) = \emptyset$ therefore $(S, \epsilon, \emptyset)$ is a context.

However this is only one way of changing the focal context. Several other ways will be presented below.

At first we introduce some preliminary material. There are three partially ordered sets (posets) that can be exploited by the interaction manager:

– The poset of answers $(\mathcal{P}(Obj), \subseteq)$

– The poset of queries $(Q, \leq)$. Given two queries q and q' of Q, we write $q \leq q'$ iff $S(q) \subseteq S(q')$ in every possible source S in $\mathcal{S}$. We write $q \sim q'$ is both $q \leq q'$ and $q' \leq q$ hold. Let $Q_{\sim}$ denote the set of equivalence classes induced by $\sim$ over Q.

– The poset of sources $(\mathcal{S}, \sqsubseteq)$. Given two sources S and S' in $\mathcal{S}$, $S \sqsubseteq S'$ iff $S(q) \subseteq S'(q)$ in every query $q \in Q$.

For every element x of the above lattices, we shall use $Br(x)$ to denote the elements that are greater than or equal to x, and $Nr(x)$ to denote the elements that are less than or equal to x in the corresponding poset.

3 Context Transition Specifications (CTSs) through Replacements

At first we study the case where the user *replaces* one component of the focal context (i.e. either S, q or A) by another component by explicitly providing the new component. Later on, we will further study these replacements and we will describe methods that allow the user to specify the desired replacement without having to provide explicitly the new component (i.e. methods for defining the new component relatively to the current).

The replacements that can be applied to a context $c = (S, q, A)$ can be:

(a) query replacement, denoted by $[q \rightarrow q']$,
(b) answer replacement, denoted by $[A \rightarrow A']$, and
(c) source view replacement, denoted by $[S \rightarrow S']$

As mentioned earlier, the user should always be in a context i.e. in a triple (S, q, A) where $S(q) = A$. This means that after any of the above kinds of replacement, the interaction manager should try to reach a context c' by changing one (or both) of the remaining two components of the focal context. Instead of leaving the IM to decide, it is the user that indicates to the IM the component(s) to be changed during a replacement. A replacement plus an indication of the above kind, is what we call a *Context Transition Specification (CTS)*. Below we discuss each possible CTS, assuming a focal context (S, q, A). For each CTS, we also discuss the motivation beneath. We do it in brief because we mainly focus on founding this interaction scheme from a mathematical point of view. However, we think that the set of CTSs that we propose are elemental, in the sense that other more sophisticated interaction schemes can be analyzed using the set of CTSs that we propose.

Query replacement $[q \rightarrow q']$

- $[q \rightarrow q'/A]$
  Here the answer A must be changed. This is the classical query-and-answer interaction scheme: when the user replaces the current query q with a new query q', the user is given a new answer (i.e. an A' such that $A' = S(q')$). Thus we can write: $[q \rightarrow q'/A](S, q, A) = (S, q', S(q'))$.
- $[q \rightarrow q'/S]$
  Here the source view S must be changed. This means that the user replaces the current query q by a query q', because the user wants the current answer A to be the answer of q', not of q. The IM should try to "adapt" to the desire of the user by changing the source view from S to an S' such that $S'(q') = A$.

Answer replacement $[A \rightarrow A']$

- $[A \rightarrow A'/q]$
  Here the query must be changed. This interaction scheme may help the user to get acquainted with the query language of the source. It can be construed as an alternative query formulation process. The user selects a number of objects (i.e. the set A') and asks from the IM to formulate the query that "describes" these objects. Subsequently the user can change the query q' in a way that reflects his/her information need. Roughly this resembles the Query By Example (QBE) process in relational databases. It also resembles the relevance feedback mechanisms in Information Retrieval systems. For example, the user selects a subset A' of the current answer A consisting of those elements of A which the user finds relevant to his/her information need. Subsequently, the IM has to change appropriately the query.
- $[A \rightarrow A'/S]$
  Here the source view S has to be changed. This case resembles the CTS $[q \rightarrow q'/S]$ that was described earlier. Here the user wants A' (instead of A) to be the answer of q. The IM should try to adapt to the desire of the user by changing the source view from S to an S' such that $S'(q) = A'$.

Source view replacement $[S \rightarrow S']$

- $[S \rightarrow S'/A]$
  Here the answer A must be changed. This is the classical interaction scheme: whenever the source changes (e.g. after an update) the answers of queries change as well, i.e. here we have $A' = S'(q)$.
- $[S \rightarrow S'/q]$
  Here the query q must be changed. This resembles the way that a relational database management system changes the query q that defines a relational view, after an update of the database, in order to preserve the contents (tuples) of the view.

We have seen six kinds of context transition specifications. Table 1 summarizes the above discussion and it also includes the cases where the IM can change two components of the focal context during one transition. In summary, the user has 9 different ways to specify the desired context transition.

Table 1. Replacements and possible reactions of the interaction manager

Replacements on $c = (S, q, A)$	c'		
Query Replacement $[q \rightarrow q']$	$(S, q', \mathbf{A'})$	$(\mathbf{S'}, q', A)$	$(\mathbf{S'}, q', \mathbf{A'})$
Answer Replacement $[A \rightarrow A']$	$(S, \mathbf{q'}, A')$	$(\mathbf{S'}, q, A')$	$(\mathbf{S'}, \mathbf{q'}, A')$
Source View Replacement $[S \rightarrow S']$	$(S', \mathbf{q'}, A)$	$(S', q, \mathbf{A'})$	$(S', \mathbf{q'}, \mathbf{A'})$

4 Finding the Target Context(s)

Given a focal context c and a context transition specification R, the role of the IM is to find the desired context $R(c)$, if one exists.

Searching for an answer (in the cases $[q \rightarrow q'/A]$ and $[S \rightarrow S'/A]$)
The cases in which the interaction manager has to change the answer in order to reach to a context (i.e. after a query or source replacement), are relatively simple: the desired context always exists and the desired answer A' can be derived using the query evaluation mechanism of the source. In particular, in the case $[q \rightarrow q'/A]$ ("query replacement - change of answer") the new context is $(S, q', \mathbf{S}(\mathbf{q'}))$, while in the case $[S \rightarrow S'/A]$ ("source replacement - change of answer") the new context is $(S', q, \mathbf{S'}(\mathbf{q}))$.

The cases in which the interaction manager has to find a new query or a new source view are less straightforward. The main problem is that the desired context does not always exist.

Searching for a query q (in the cases $[A \rightarrow A'/q]$ and $[S \rightarrow S'/q]$)
In cases $[A \rightarrow A'/q]$ and $[S \rightarrow S'/q]$ we are looking for a $q \in Q$ such that $S(q) = A$ for given S and A. Let us first try to define a "naming service", i.e. a method for computing one or more queries that describe (name) a set of objects $A \subseteq Obj$. For supporting the naming service we would like to have a function $n : \mathcal{P}(Obj) \rightarrow Q$ such that for each $A \subseteq Obj$, $S(n(A)) = A$. Having such a function, we would say that the query $n(A)$ is an exact name for the object set A. Note that if S is an *onto* function then the naming function n coincides with the inverse relation of S, i.e. with the relation $S^{-1} : \mathcal{P}(Obj) \rightarrow Q$. However, this is not always the case, as more often than not, S is not an onto function, i.e. $\mathcal{A} \subset \mathcal{P}(Obj)$. Furthermore, if S is onto and one-to-one, then S^{-1} is indeed a

function, thus there is always a unique q such that $S(q) = A$ for each $A \subseteq Obj$ [2].

As S is not always an onto function, we shall introduce two "approximate" naming functions, a *lower* naming function n^- and an *upper* naming function n^+. We can define the function n^- and n^+ as follows:

$$n^-(A) = lub\{\ q \in Q \mid S(q) \subseteq A\}$$
$$n^+(A) = glb\{\ q \in Q \mid S(q) \supseteq A\}$$

where A is any subset of Obj. Now let A be a subset of Obj for which both $n^-(A)$ and $n^+(A)$ are defined (i.e. the above lub and glb exist). It is clear that in this case it holds:

$$S(n^-(A)) \subseteq A \subseteq S(n^+(A))$$

and that $n^-(A)$ and $n^+(A)$ are the best "approximations" of the exact name of A. Note that if $S(n^-(A)) = S(n^+(A))$ then both $n^-(A)$ and $n^+(A)$ are exact names of R. If Q is a query language that (a) supports disjunction ($\vee$) and conjunction ($\wedge$) and it is closed with respect to both these operations, and (b) has a top ($\top$) and a bottom ($\bot$) element such that $S(\top) = Obj$ and $S(\bot) = \emptyset$, then the functions n^- and n^+ are defined for every subset A of Obj. Specifically, in this case $(Q_\sim, \leq)$ is a complete lattice, thus these functions are defined as:

$$n^-(A) = \bigvee\{\ q \in Q \mid S(q) \subseteq A\}$$
$$n^+(A) = \bigwedge\{\ q \in Q \mid S(q) \supseteq A\}$$

As Q is usually an infinite language, $n^-(A)$ and $n^+(A)$ are queries of infinite length. This means that in practice we also need a method for computing a query of finite length that is equivalent to $n^-(A)$ and another that is equivalent to $n^+(A)$. If however Q does not satisfy the above conditions (a) and (b), then $n^-(A)$ and $n^+(A)$ may not exist. For such cases, we can define n^- and n^+ as follows: $n^-(A) = max\{\ q \in Q \mid S(q) \subseteq A\}$ and $n^+(A) = min\{\ q \in Q \mid S(q) \supseteq A\}$, where *max* returns the maximal element(s), and *min* the minimal element(s). Clearly, in this case we may have several lower and upper names for a given A. A specialization of these naming functions for the case of taxonomy-based sources is described in [10].

Let us now return to our problem. If a naming function n_S is available for source S, then the context we are looking for exists and can be found as follows:

- $[A \rightarrow A'/q](S, q, A) = (S, n_S(A'), A')$
- $[S \rightarrow S'/q](S, q, A) = (S', n_{S'}(A), A)$

[2] Usually, sources are not one-to-one functions. For instance, the supported query language may allow formulating an infinite number of different queries, while the set of all different answers $\mathcal{A}$ is usually finite (e.g. $\mathcal{P}(Obj)$).

If only approximate naming functions are available, then two "approximate" next contexts exist:

– $[A \rightarrow A'/q](S,q,A) = \begin{cases} (S, n_S^-(A'), S(n_S^-(A'))) \\ (S, n_S^+(A'), S(n_S^+(A'))) \end{cases}$
– $[S \rightarrow S'/q](S,q,A) = \begin{cases} (S', n_{S'}^-(A), S'(n_{S'}^-(A))) \\ (S', n_{S'}^+(A), S'(n_{S'}^-(A))) \end{cases}$

Notice that in the above cases, IM does not change only q, but also the answer. In the case $[A \rightarrow A'/q]$, the new context has an answer, say A", which is the closest possible to the requested (by the user) answer A'. In the case $[S \rightarrow S'/q]$, the new context has an answer A' which is the closest possible to the current A.

Searching for a source view S (in the cases $[A \rightarrow A'/S]$ and $[q \rightarrow q'/S]$)
In cases $[A \rightarrow A'/S]$ and $[q \rightarrow q'/S]$ we are looking for a $S \in \mathcal{S}$ such that $S(q) = A$ for given q and A. Note that the desired source S always exists if and only if $\mathcal{S}$ is the set of all functions from Q to $\mathcal{A}$. We cannot describe any mechanism for finding the desired S' within such an abstract framework. However, later on we shall see how restricted relative replacements and a source relaxation/contraction mechanism can be exploited in order to support this kind of interaction.

The following list summarizes how the IM can find the target context after each kind of CTS:

(1) $[q \rightarrow q'/A](c) = (S, q', S(q'))$, i.e. it relies on query evaluation
(2) $[S \rightarrow S'/A](c) = (S', q, S'(q))$, i.e. it relies on query evaluation
(3) $[A \rightarrow A'/q](c) = (S, n_S(A'), A')$, i.e. it relies on naming functions
(4) $[S \rightarrow S'/q](c) = (S', n_{S'}(A), A)$, i.e. it relies on naming functions
(5) $[A \rightarrow A'/S](c) =?$, i.e. this is an open issue
(6) $[q \rightarrow q'/S](c) =?$, i.e. this is an open issue

5 Relative Replacements and Relative CTSs

Until now we have described how the IM could react to a context transition specification. Now we shall focus on the replacements. Certainly, the user can manually specify a replacement, e.g. submit a new query, select a set of objects $A' \subseteq Obj$, or update the source S by adding, deleting or modifying some of the contents of S.

Here we describe methods that allow the user to specify the desired replacement without having to provide explicitly the new component, i.e. methods for defining the new component relatively to the current. These methods can be helpful for the user during the interaction with the system.

The three partial orders mentioned earlier can be exploited for moving to a component (answer, query, or source) that covers, or is covered by, the current component. Given a component x, let x^+ denote a component that covers x, and let x^- denote a component that is covered by x [3]. Note that there may not always exist a unique x^+ or a unique x^-. Specifically, we may have zero, one, or more x^+'s and x^-'s for a given x.

Let $Up(x)$ denote a component among those greater than x and that the IM can compute (ideally $Up(x) = x^+$), and let $Down(x)$ denote a component among those less than x and that the IM can compute (ideally $Down(x) = x^-$). Notice that these Up and $Down$ functions correspond to

- a *query relaxation/contraction* mechanism, if applied to Q,
  (i.e. $[q \rightarrow Up(q)]$ and $[q \rightarrow Down(q)]$)
  Such mechanisms have been proposed for several kinds of sources, including relational [6], semi-structured [4], taxonomy-based [11], Description-Logics-based [9, 3] and Web sources [7].
- an *answer enlargement/reduction* mechanism, if applied to $\mathcal{A}$,
  (i.e. $[A \rightarrow Up(A)]$ and $[A \rightarrow Down(A)]$)
- a *source relaxation/contraction* mechanism, if applied to $\mathcal{S}$
  (i.e. $[S \rightarrow Up(S)]$ and $[S \rightarrow Down(S)]$).
  An example of such a mechanism for the case of taxonomy-based sources, founded on abduction, is described in [8].

These relative replacements can be used within context transition specifications. At the user interface level, this means that the user can have at his/her disposal two buttons, "Up" and "Down" for every component of the focal context. By pressing a button, the corresponding replacement is specified. Figure 1 sketches such a user interface. The option control labelled "Change" allows the user to indicate to the IM the component (S, q, or A) that should be changed in order to reach the new context. Then, it is the task of the IM to try to reach a context, using the approach indicated by the user, and subsequently to deliver this context to the user. A CTS defined by a relative replacement will be called *relative CTS*.

Below we enumerate all possible relative CTSs:

(1) $[q \rightarrow Up(q)/A]$: query relaxation resulting in answer enlargement
$[q \rightarrow Down(q)/A]$: query contraction resulting in answer reduction
(2) $[S \rightarrow Up(S)/A]$: source relaxation resulting in answer enlargement
$[S \rightarrow Down(S)/A]$: source contraction resulting in answer reduction
(3) $[A \rightarrow Up(A)/q]$: answer enlargement resulting in query relaxation
$[A \rightarrow Down(A)/q]$: answer reduction resulting in query contraction
(4) $[S \rightarrow Up(S)/q]$: source relaxation resulting in query contraction
$[S \rightarrow Down(S)/q]$: source contraction resulting in query relaxation
(5) $[A \rightarrow Up(A)/S]$: answer enlargement resulting in source relaxation
$[A \rightarrow Down(A)/S]$: answer reduction resulting in source contraction
(6) $[q \rightarrow Up(q)/S]$: query relaxation resulting in source contraction
$[q \rightarrow Down(q)/S]$: query contraction resulting in source relaxation

[3] An element x^+ covers an element x if $x^+ > x$ and $x^+ \leq x'$ for every $x' > x$.
An element x is covered by an element x^- if $x^- < x$ and $x^- \geq x'$ for every $x' < x$.

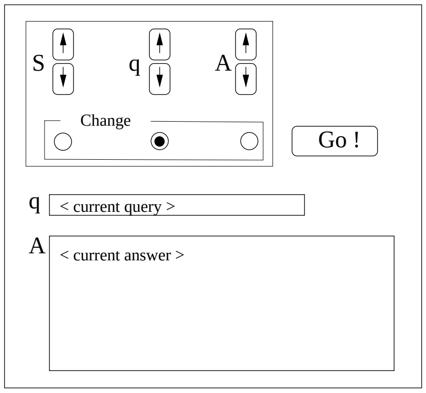

Fig. 1. A user interface for specifying context transitions through relative replacements

6 Restricting the Relative CTSs

The three partial orders mentioned in Section 2 (and their interrelationships) can be exploited in order to restrict the set of relative CTSs to those for which the IM can indeed compute the target context. Below we describe how we can restrict relative replacements according to the CTS in which they appear.

Up/Down on Sources

As we saw earlier, source replacements appear in cases (2) and (4) of CTSs. In these cases, there is no need to restrict $Up(S)$ or $Down(S)$ because the IM can always find the target context. In particular, CTS (2) relies on query evaluation and CTS (4) on naming functions:

(2) $[S \to S'/A](c) = (S', q, S'(q))$, i.e. it relies on query evaluation
(4) $[S \to S'/q](c) = (S', n_{S'}(A), A)$, i.e. it relies on naming functions

Up/Down on Answers

Answer replacements appear in cases (3) and (5) of CTSs. CTS (2) relies on naming functions, while CTS (5) is still open:

(3) $[A \to A'/q](c) = (S, n_S(A'), A')$, i.e. it relies on naming functions
(5) $[A \to A'/S](c) = ?$, i.e. this is an open issue

Note that we can restrict $Up/Down(A)$ so that to support CTS (3) even if naming functions are not available (or if they are available, but there is no exact name for A'). This can be achieved by defining:

$$Up(A) = S(Up(q))$$
$$Down(A) = S(Down(q))$$

In this way, the IM can find the target context without using any naming function. i.e.

(3R) $[A \rightarrow S(Up(q))/q](c) = (S, Up(q), S(Up(q)))$
$[A \rightarrow S(Down(q))/q](c) = (S, Down(q), S(Down(q)))$

Another interesting remark is that by restricting $Up(A)$ or $Down(A)$ the IM can support CTS (5). This can be achieved by defining $Up(A)$ and $Down(A)$ as follows:

$$Up(A) = Up(S)(q)$$
$$Down(A) = Down(S)(q)$$

In this way, the IM can find the target context:

(5R) $[A \rightarrow Up(S)(q))/S](c) = (Up(S), q, Up(S)(q))$
$[A \rightarrow Down(S)(q))/S](c) = (Down(S), q, Down(S)(q))$

Up/Down on Queries

Query replacements appear in cases (1) and (6) of CTSs:

(1) $[q \rightarrow q'/A](c) = (S, q', S(q'))$, i.e. relies on query evaluation
(6) $[q \rightarrow q'/S](c) =$?, i.e. this is an open issue

CTS (1) relies on query evaluation, hence no restriction is needed. CTS (6) is still open but by restricting $Up/Down(q)$ the IM can support it, if naming functions are available. This can be achieved by defining $Up(q)$ and $Down(q)$ as follows:

$$Up(q) = n_{Down(S)}(A)$$
$$Down(q) = n_{Up(S)}(A)$$

In this way, the IM can find the target context:

(6R) $[q \rightarrow n_{Up(S)}(A)/S](c) = (Up(S), n_{Up(S)}(A), A)$
$[q \rightarrow n_{Down(S)}(A)/S](c) = (Down(S), n_{Down(S)}(A), A)$

In this section we showed how the IM can support all kinds of relative CTSs through restricted relative replacements.

7 Concluding Remarks

We introduced a unified formal framework which captures several interaction schemes between users and information systems. This unified view allowed us to describe more complex interaction mechanisms that those supported by the existing systems. The value of this unification is not only theoretical. It can be directly reflected in the interaction between the user and the information source, i.e in the user interface.

Further research includes specializing the introduced framework for the case where the source is a taxonomy-based source (such as a Web Catalog), a relational database, a semistructured database [1], or a Description-Logics database [5].

Acknowledgements

The first author wants to thank his wife for being an endless source of happiness and inspiration.

References

1. Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Series in Data Management Systems, 1999.
2. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *"Modern Information Retrieval"*. ACM Press, Addison-Wesley, 1999.
3. Alain Bidault, Christine Froidevaux, and Brigitte Safar. "Repairing Queries in a Mediator Approach". In *Proceedings of the ECAI'02*, Lyon, France, 2002.
4. Lee Dongwon. *"Query Relaxation for XML Model"*. PhD thesis, University of California, 2002.
5. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. "Reasoning in Description Logics". In Gerhard Brewka, editor, *Principles of Knowledge Representation*, chapter 1, pages 191–236. CSLI Publications, 1996.
6. Terry Gaasterland. "Cooperative Answering through Controlled Query Relaxation". *IEEE Expert: Intelligent Systems and Their Applications*, 12(5), 1997.
7. Wen-Syan Li, K. Seluk Candan, Quoc Vu, and Divyakant Agrawal. "Query Relaxation by Structure and Semantics for Retrieval of Logical Web Documents". *IEEE Transactions on Knowledge and Data Engineering*, 14(4), 2002.
8. Carlo Meghini and Yannis Tzitzikas. "An Abduction-based Method for Index Relaxation in Taxonomy-based Sources". In *Proceedings of MFCS 2003, 28th International Symposium on Mathematical Foundations of Computer Science*, number 2747 in Lecture notes in computer science, pages 592–601, Bratislava, Slovak Republic, August 2003. Springer Verlag.
9. E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. "Estimating Information Loss for Multi-ontology Based Query Processing". In *Proceedings of Second International and Interdisciplinary Workshop on Intelligent Information Integration*, Brighton Centre, Brighton, UK, August 1998.
10. Yannis Tzitzikas and Carlo Meghini. "Ostensive Automatic Schema Mapping for Taxonomy-based Peer-to-Peer Systems". In *Seventh International Workshop on Cooperative Information Agents, CIA-2003*, number 2782 in Lecture Notes on Artificial Intelligence, pages 78–92, Helsinki, Finland, August 2003. (Best Paper Award).
11. Yannis Tzitzikas, Nicolas Spyratos, and Panos Constantopoulos. "Query Translation for Mediators over Ontology-based Information Sources". In *Second Hellenic Conference on Artificial Intelligence, SETN-2002*, Thessaloniki, Greece, April 2002.
12. Jeffrey D. Ullman. *"Principles of Database and Knowledge-Base Systems, Vol. I"*. Computer Science Press, 1988.