Active XML and active query answers*

Serge Abiteboul** Omar Benjelloun Tova Milo***

INRIA-Futurs and† LRI

Abstract. XML and Web services are revolutioning the automatic management of distributed information, somewhat in the same way that HTML, Web browser and search engines modified human access to world wide information. We argue in this paper that the combination of XML and Web services allows for a novel query answering paradigm, where the exchanged information mixes materialized and intensional, active, information.
We illustrate the flexibility of this approach by presenting Active XML that is based on embedding Web service calls in XML data. We discuss advantages of an approach of query answering based on intensional and active data.

1 Introduction

Alice: What is the meaning of "intensional"?
Mom: Look it up in the dictionary

There is typically more in answering a query than just listing plain facts. Indeed, as shown by the dialogue between Alice and her mom, a smart answer may teach you more than what you have asked. If Alice looks up "intensional" in the dictionary, she will not only learn the meaning of this particular word, but will also learn how one may find, in general, the meaning of words, and she will be able to reuse this knowledge later.

We refer to a response that gives the means to obtain an answer without answering explicitly, as an *active answer*. Many other examples and motivations for the use of such active answers are presented further. We will argue that Active XML (AXML in short), a new paradigm that combines XML and Web services, provides a simple and unified platform for such active answers, thereby introducing a new dimension to flexible query answering in the distributed context of the Web.

The field of distributed data management has centered for many years around the relational model. More recently, the Web has made the world wide (or intranet) publication of data much simpler, by relying on HTML as a standard

* The research was partly funded by EC Project DBGlobe (IST 2001-32645), the RNTL Project e.dot and the French ACI MDP2P

** Also Xyleme SA

*** On leave from Tel Aviv University

† Gemo Team, PCRI Saclay, a joint lab between CNRS, Ecole Polytechnique, INRIA and Université Paris-Sud

language, universally understood by Web browsers, on plain-text search engines and on query forms. However, because HTML is more of a presentation language for documents than a data model, and because of limitations of the core HTTP protocol, the management of distributed information remained cumbersome. The situation is today dramatically improving with the introduction of XML and Web services. The Extensible Markup Language, XML [18], is a self-describing, semi-structured data model that is becoming the standard format for data exchange over the Web. Web services [21] provide an infrastructure for distributed computing at large, independently of any platform, system or programming language. Together, they provide the appropriate framework for distributed management of information.

Active XML (AXML, for short), is a declarative framework that harnesses these emerging standards for the integration and management of distributed Web data. An AXML document is an XML document where some of the data is given explicitly, while some portions are given only intensionally by means of embedded calls to Web services. By calling the services, one can obtain up-to-date information. In particular, AXML provides control of the activation of service calls both from the client side (pull) or from the server side (push). AXML encourages an approach to query answering based on *active answers*, that is, answers that are AXML documents.

Choosing which parts of a query answer should be given explicitly and which should be given in an active/intensional manner may be motivated and influenced by various parameters. These include logical considerations, such as knowledge enrichment or context awareness, and physical considerations like performance or capabilities. We will give an overview of these various aspects, and describe some possible solutions for guiding this crucial choice.

It should be noted that the idea of mixing data and code is not new. Functions embedded in data were already present in relational systems [14] as stored procedures. Also, method calls form a key component of object-oriented databases [7]. In the Web context, scripting languages such as PHP or JSP have made popular the integration of (query) processing inside HTML or XML documents. Embedding calls to Web services in XML documents is just one step further, but is indeed a very important one. The novelty here is that since both XML and Web services are becoming standards, AXML documents can be universally understood, and therefore can be *exchanged*. We focus here on the new flexibly brought to query answering by this paradigm of exchanging AXML documents.

The rest of this paper is organized as follows. We first briefly recall some key aspects of XML, Web services (Section 2) and Active XML (Section 3). The following two sections informally discuss motivations for the use of active data in query answers, both logical (Section 4) and physical (Section 5). The last section concludes by mentioning some supporting techniques around AXML that are important for query processing.

2 XML and Web services

XML is a new data exchange format promoted by the W3C [20] and widely adopted by industry. An XML document can be viewed as a labeled ordered tree, as seen on the example of [1] Figure 1. XML is becoming a lingua franca, or more precisely an agreed upon syntax, that most pieces of software can understand or will shortly do. Unlike HTML, XML does not provide any information about the document presentation. This is typically provided externally using a CSS or XSL stylesheet.

XML documents may be typed using a language called XML Schema [19]. A schema mainly enforces structural relationships between labels of elements in the document tree. For instance, it may request a *movie* element to consist of a *title*, zero or more *author*s and *reviews*. The typing proposed by XML Schema is very flexible, in the sense that it can describe, for instance, an HTML webpage, as well as a relational database instance, thus marrying the document world with the structured or semistructured world of databases. The presence of structure in XML documents enables the use of queries beyond keyword search, using query languages such as XPath or XQuery.

Web Services are a major step in the evolution of the Web. Web servers, that originally provide HTML pages for human consumption, become gateways to available services. Although most of the hype around Web services comes from e-commerce, one of their main current uses is for the management of distributed information. If XML provides the data model, Web services provide the adequate abstraction level to describe the various actors in data management such as databases, wrappers or mediators and to manage the communications between them.

Web services in fact consist of an array of emerging standards. To find the desired service, one can query the yellow-pages: a UDDI [17] directory (Universal Discovery Description and Integration). Then, to understand how to interact with it, one relies on WSDL [22] (Web Service Definition Language), something like Corba's IDL. One can then access the service using SOAP [16], an XML-based lightweight protocol for the exchange of information. Of course, life is more complicated, so one often has to sequence operations (see Web Services Choreography [23]) and consider issues such as confidentiality, transactions, etc.

XML and Web services are nothing really new from a technical viewpoint. However, their use as a large scale infrastructure for data sharing and communication provides a new, trendy environment to utilize some old ideas in a new context. For instance, tree automata techniques regained interest as they best model essential manipulations on XML, like typing and querying.

This new distributed, setting for data and computation also raises many new challenges for computer science research in general, and query answering

[1] We will see in the next section that this XML document is also an Active XML document.

```
<directory>
  <movies>
    <director>Hitchcock</director>
    <sc service="movies@allocine.com" >Hitchcock</sc>
    <movie> <title>Vertigo</title>
      <actor>J. Stewart</actor> <actor>K. Novak</actor>
      <reviews> <sc service="reviews@cine.com" >Vertigo</sc></reviews>
    </movie>
    <movie> <title>Psycho</title>
      <actor>N. Bates</actor>
      <reviews> <sc service="reviews@cine.com" >Psycho</sc></reviews>
    </movie>
  </movies>
</directory>
```

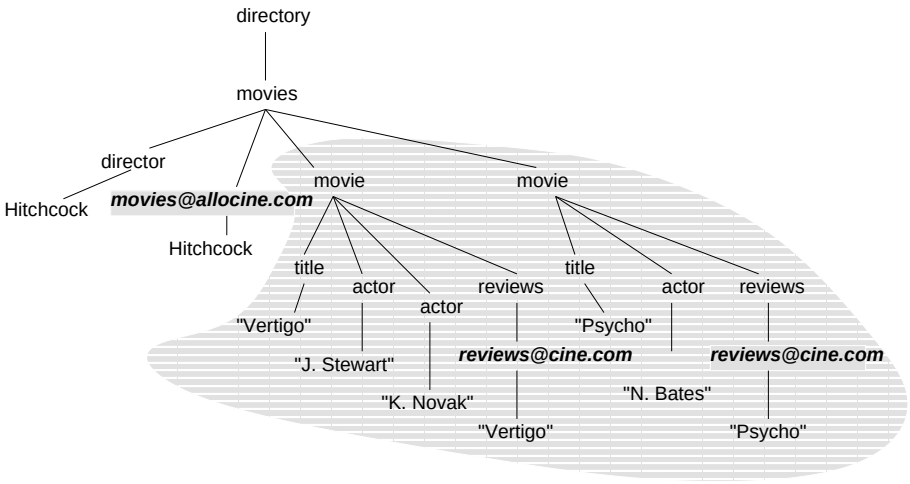

Fig. 1. An Active XML document and its tree representation

in particular. For instance, data sources ought to be discovered and their informations integrated dynamically, and this may involve data restructuring and semantic mediation. Classical notions such as data consistency and query complexity must be redefined to accommodate the huge size of the Web and its fast speed of change.

3 Active XML

To illustrate the power of combining XML and Web services, we briefly describe Active XML, a framework based on the idea of embedding calls to Web services in XML documents. This section is based on works done in the context of the Active XML project [6].

In Active XML (AXML for short), parts of the data are given explicitly, while other parts consist of calls to Web services that generate more data. AXML is based on a P2P architecture, where each AXML peer acts as a client, by activating Web service calls embedded in its documents, and also acts as a server, by providing Web services that correspond to queries or updates over its repository of documents. The activation of calls can be finely controlled to happen periodically, or in reaction to some particular (in the style of database triggers), or in a "lazy" way, whenever it may contribute data to the answer of a query.

AXML is an XML dialect, as illustrated by the document in Figure 1. (Note that the syntax is simplified in the example for presentation purposes.) The `sc` elements are used to denote embedded service calls. Here, reviews are obtained from `cine.com`, and information about more Hitchcock movies may be obtained from `allocine.com`. The data obtained from the call to `allocine.com` corresponds to the shaded part of the tree. In case the relashionship between data and the service call is maintained, we say that data is *guarded* by the call.

The data obtained by a call to a Web service may be viewed as intensional (it is originally not present). It may also be viewed as dynamic, since the same service call possibly returns different, data when called at different times. When a service call is activated, the data it returns is inserted in the document that contains it. Therefore, documents evolve in time as a consequence of call activations. Of particular importance is thus the decision to activate a particular service call. In some cases, this activation is decided by the peer hosting the document. For instance, a peer may decide to call a service only when the data it provides is requested by a user; the same peer may choose to refresh the data returned by another call on a periodic basis, say weekly. In other cases, the service provider may decide to send updates to the client, for instance because the latter registered to a subscription-based, continuous service.

A key aspect of this approach is that AXML peers exchange AXML documents, i.e., documents with embedded service calls. Let us highlight an essential difference between the exchange of regular XML data and that of AXML data. In frameworks such as Sun's JSP or PHP, dynamic data is supported by programming constructs embedded inside documents. Upon request, all the code

is evaluated and replaced by its result to obtain a regular, fully materialized HTML or XML document. But since Active XML documents embed calls to Web services, and the latter provide a standardized interface, one does not need to materialize all the service calls before sending some data. Instead, a more flexible data exchange paradigm is possible, where the sender sends an XML document with embedded service calls (namely, an AXML document) and gives the receiver the freedom to materialize the data if and when needed. In the following sections, we will investigate both logical and physical motivations for exchanging AXML documents, in particular as answers to queries, and discuss the benefits obtained from doing so.

4 Logical motivations for active answers

In this section, we present the main *logical* motivations for returning active answers to queries, namely reasons motivated by the desire to provide more functionality. *Physical* motivations will be considered in the next section.

We consider three classes of logical motivations.

Reusability of answers

A first family of motivations is about enabling the client to reuse autonomously some of the information in the query answer without having to ask the query again, by giving her more knowledge about its origin or means to get it.

Dynamicity Perhaps the main motivation for returning active answers is to capture the fact that parts of the answer may change over time. For instance, suppose someone asks for the description of a ski resort, say Megève. A portion of the answer may describe the snow condition, an information that is typically volatile. If this answer is inspected tomorrow or in a month, we would like the information to still be up-to-date, without having to ask the initial query again. So we may want to make the snow condition part active, that is, to guard it with a Web service call to provide real-time snow conditions.

Knowledge In the dialogue between Alice and her mom, the mother answers intensionally. By looking in a dictionary, Alice will learn something beyond the meaning of the word "intensional": She may learn how to obtain the meaning *of any word*. In general, by obtaining the information via a service call, the client becomes more independent knowledge-wise and may be able to *generalize* the use of the service, by reusing it over time, or calling it with different parameters, e.g. other ski resorts in the snow conditions example, or other words in the case of Alice.

Partial answers

A second family of motivations has to do with providing the client with a partial representation of query answers, instead of the full computed result of the query.

Summarization If the client of the query is a human user, then the full answer may simply be too large and unreadable for her. The natural way to summarize it is to "intensionalize" some of it. We are here at the border between man machine interfaces and query answer. Active data may be viewed as an environment to provide several views of a query answer, as well as means to navigate between them, by zooming in and out. This can easily be accomplished using AXML, with the service calls acting as navigational links.

Incompleteness Another important reason is that we may want the answer to be incomplete. For instance, the answer may be very large and we may be satisfied with just some of it, e.g., we ask the query "XML" to Google and there are 22 300 000 answers. The server decides not to send them all but provide, say the first 10, and a link to obtain more. This may be viewed as a very simplistic form of active data but the situation does not have to be so simple. Consider a search engine that is aggregating three different data sources. It receives a query and one of the sources is down. Then the search engine may want to answer in AXML with the concrete data it knows of, and with a call to the last source with the meaning that more data may be obtained from there when the source will come back up.

Partial computations Active data may also be used to represent computations that still need to be carried out to obtain the full answer. An example of this comes from the management of security and encryption. A secret phone number may be sent encrypted. So, the answer has to be decrypted to be understood, so the answer is active. As another example, some news systems provide as results lists of news headlines, each of them with a summary and the means to buy the article (say using a Web service). With an active data approach, this may be viewed as part of the answer, to be obtained by the data receiver via some additional computation (calling the service and paying the required fee).

Enriching answers

A last family of motivations consists in enriching the query answer with active parts that provide the client with extra functionality that better fits her needs.

Metadata Consider, as an example of a "raw" query answer, the result of a scientific experiment. This is very static information. By activating it, we can add some meta-information to it. For instance, the document may be in AXML with some service calls to obtain details on the data provenance and others to access existing annotations that were made on this experiment or to attach some new annotation. The metadata could also allow obtaining different versions of the same document and access related information. The active part may be in charge of maintaining (possibly implicitly) the metadata and, in particular, the provenance information.

Context awareness and personalization A piece of information, be it extensional or intensional, may depend on some context. For instance, if a user obtained a list of restaurants, the content of the list may depend on predefined preferences, like the user's taste for certain kinds or food, or on religious restrictions. Also, the explicit presence of some details in this list may be guided by the device used by the user, possibly to accommodate limitations of her screen, or by the way the information will be used. For instance, all data should be extensional if the information is to be used off-line.

Enriching data This is the process of adding value to an answer by applying some external tools. The enrichment may occur at the level of the entire query result or at a different granularity, e.g., an XML element. Typical enrichment may include links with the user context, extracting related concepts from an ontology, transforming from one type (XML schema) to another, or translating the answer to a preferred language.

In all these various scenarios, an active answer may be viewed as an agent in charge of managing the answer and providing it to the user.

5 Physical motivation for active answers

In the previous section, we considered motivations for using active answers to provide more functionality. We next briefly illustrate how the notion of active answer may be useful for guiding the evaluation of queries, when information and query processing are distributed.

The presence of intensional information in data that is transfered between systems on the Web is rather standard. For instance, in PHP or JSP pages, the code that is inserted inside HTML or XML documents is a form of intensional information. But typically, all this intensional data is materialized *before* being sent, and only "plain" data is exchanged. This does not have to be the case in a Web service setting. The decision whether to invoke service calls before or after the data transfer may be influenced by physical considerations such as the current system load or the cost of communication. For instance, if the server is overloaded or if communication is expensive, the server may prefer to send smaller files and delegate as much materialization of the data as possible to the client. Otherwise, it may decide to materialize as much data as possible before transmission, in order to reduce the processing on the client's side.

Note however that although Web services may in principle be called remotely from anywhere on the Internet, it may be the case that the particular client of the intensional document cannot perform them, e.g., a newspaper reader's browser may not be able to handle the intensional parts of a document, or the user may not have access to a particular service, e.g., by lack of access rights. In such cases, it is compulsory to materialize the corresponding information before sending the document.

More generally, it is interesting to note that there are analogs to most of the aspects that played a role at the logical level. We next illustrate some of them.

To see one aspect that typically plays an important role on the performance and quality of Web servers, consider *Dynamicity*. Suppose some Web server S publishes a book catalog and that a comparative shopping site S' downloads the pages of the catalog regularly. The cache of S' will contain these pages, but information such as book prices that change rapidly will often be stale. The Web server S may decide to make catalog pages active. Then a reload of a page that is now controlled by a Web service may just result in sending the "delta" between the old and the new page (e.g., just a change of prices). This results in savings in communication. Furthermore, in an Active XML framework, S' is able to subscribe to price changes and to be notified of such changes without having to reload pages regularly. As a consequence, the cache of S' will have a more accurate copy of the catalog.

As another example, consider *Knowledge*. Knowledge may be exchanged in the form of intensional information, for instance to facilitate tasks such as routing. Suppose some peer A requests some data from peer B and B in fact finds the data in some peer C. Then the data from C to A would transit via B. However, if B answers A intensionally, then A may obtain the data directly from C, which possibly results in saving in communication.

6 Supporting techniques

We have briefly discussed XML and Web services and the advantages of answering queries with active data. We presented AXML, that enables such answering style. To conclude this paper, we mention three important issues in this setting, and recent works performed in these directions.

To call or not to call Suppose someone asks a query about the "Vertigo" movie. We may choose to call `cine.com` to obtain the reviews or not before sending the data. As we saw, this decision may be guided by considerations such as performance, cost, access rights, security, etc. Now, if we choose to activate the service call, it may return a document with embedded service calls and we have to decide whether to activate those or not, and so on, recursively. We introduce in [11] a technique to decide whether some calls should be activated or not based on typing. First, a negotiation between the peers determines the schema of data to exchange. Then, some complex automata manipulation are used to cast the query answer to the type that has been agreed upon. The general problem has deep connections with alternating automata, i.e., automata alternating universal and existential states [13].

The choice of a schema for data exchange between peers in general, or for query answers in particular is itself an interesting reasoning problem. We need some formal model to describe preferences of the peers as well as their knowledge of the context that will guide this selection.

Lazy service calls and query composition As mentioned earlier, it is possible in AXML to specify that a call is activated only when the data it returns may be needed, e.g., to answer a query. Suppose that a user has received some active document and wants to extract some information from it, by evaluating a query. A difficulty is then to decide whether the activation of a particular call is needed or not to answer that query. For instance, if someone asks for information about the actors of "The 39 steps" of Hitchcock, we need to call `allocine.com` to get more movies by this director. Furthermore, if this service is sophisticated enough, we may be able to ask only for information about that particular movie (i.e., to "push" the selection to the source). Algorithms for guiding the invocation of relevant calls, and pushing queries to them are presented in [1]. Some surprising connections between this problem and optimization techniques for deductive database and logic programming are exhibited in [5].

Cost model We describe in [3] a framework for the management of distribution and replication in the context of AXML. We introduce a cost model for query evaluation and show how it applies to user queries and service calls. In particular, we describe an algorithm that, for a given peer, chooses data and services that the peer should replicate in order to improve the efficiency of maintaining and querying its dynamic data. This is a first step towards controlling the use of intensional answers in a distributed setting.

Efficient query processing of structured and centralized data was made feasible by relational databases and its sound logical foundation [14, 4]. Deep connections with descriptive complexity have been exhibited [9, 4]. For the management of answers with active components, we are now at a stage where a field is building and a formal foundation is still in its infancy. The development of this foundation is a main challenge for the researchers in the field.

References

1. S. Abiteboul, O. Benjelloun, B. Cautis, I. Manolescu, T. Milo, N. Preda: Lazy Query Evaluation for Active XML, Sigmod 2004.
2. S. Abiteboul, P. Buneman, and D. Suciu, Data on the Web, Morgan Kaufmann Publishers, 2000.
3. S. Abiteboul, A. Bonifati, G. Cobena, I. Manolescu, T. Milo, Active XML Documents with Distribution and Replication, ACM SIGMOD, 2003.
4. S. Abiteboul, R. Hull and V. Vianu, Foundations of databases, Addison-Wesley, 1995.
5. S. Abiteboul and T. Milo, Web Services meet Datalog, 2003, submitted.
6. The AXML project, INRIA, http://www-rocq.inria.fr/verso/Gemo/Projects/axml.
7. The Object Database Standard: ODMG-93, editor R. G. G. Cattell, Morgan Kaufmann, San Mateo, California, 1994.
8. Tata, Tree Automata Techniques and Applications, H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison and M. Tommasi, www.grappa.univ-lille3.fr/tata/

9. N. Immerman, Descriptive Complexity, Springer 1998.
10. M. Lenzerini, Data Integration, A Theoretical Perspective, ACM PODS 2002, Madison, Winsconsin, USA, 2002.
11. T. Milo, S. Abiteboul, B. Amann, O. Benjelloun, F. Dang Ngoc, Exchanging Intensional XML Data, ACM SIGMOD, 2003.
12. M.T. Ozsu, and P. Valduriez, Principles of Distributed Database Systems, Prentice-Hall, 1999.
13. A. Muscholl, T. Schwentick, and L. Segoufin, Active Context-Free Games, Symposium on Theoretical Aspects of Computer Science, 2004.
14. J.D. Ullman, Principles of Database and Knowledge Base Systems, Volume I, II, Computer Science Press, 1988.
15. F. M. Cuenca-Acuna, C. Peery, R. P. Martin, T. D. Nguyen, Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities, Department of Computer Science, Rutgers University, 2002.
16. The SOAP Specification, version 1.2, http://www.w3.org/TR/soap12/
17. Universal Description, Discovery and Integration of Web Services (UDDI), http://www.uddi.org/
18. The Extensible Markup Language (XML), http://www.w3.org/XML/
19. XML Typing Language (XML Schema), http://www.w3.org/XML/Schema
20. The World Wide Web Consortium (W3C), http://www.w3.org/
21. The W3C Web Services Activity, http://www.w3.org/2002/ws/
22. The Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl/
23. W3C, Web Services Choreography, http://www.w3.org/2002/ws/chor/