About XML integration and distribution

Maarten Fokkinga

Version of April 24, 2003, 15:39

Abstract. We attempt to characterize the goals that we have in mind when formalizing the relation between distributed XML query processing and integration.

1 Introduction. Elsewhere we have sketched an approach to a formalization of the XML notions of document, schema, validation, and query, where the abstraction level is suitable for human understanding and proof (rather than for algorithmic realization). We got stuck in the treatment of XML schema integration. So, obeying Joeri's command, here we step back and set out our goals in a self-supporting document. Comments will be appreciated.

2 Notation. Whatever a query and a document formally is, we denote the answer of a query q on a document D by $q\,.\,D$ (the symbol $.$ resembles a symbol for function application, and indeed, answering a query on a document resembles "applying a query to a document"). Being a bit sloppy, we use the same notation for a *series* of queries, and for a *series* of documents.

We use Ds as a short-hand notation for $D_1 \ldots D_k$.

The term *distrigration* is mnemonic for distribution and integration.

3 Distrigration: intuition. We speak of distribution and integration if there is a collection of "distributed" documents Ds and another (materialized or virtual) "integrated" document D such that querying D may alternatively be answered by querying the distributed documents Ds and then combining the answers. We do not stipulate whether D is constructed out of Ds or the other way around; that may vary between different applications. The relation between Ds and D is denoted f; we even assume that f is functional from Ds to D, that is, $D = f\,Ds$.

4 Distrigration: definition. A triple (f, g, h) is called a *distrigration* if:

- f is a function that, given documents Ds, yields another document D $(= f\,Ds)$;
- g is a function that, given a query q, yields a series of queries $g\,q$;
- h is a function that, given a series of query answers, yields another answer; and
- for all documents Ds and queries q, it is true that answering q on the integrated document D $(= f\,Ds)$ yields the same result as first splitting the query into queries $g\,q$,

distributing and answering these on documents Ds, and then combining the answers via h:

$$q \,.\, f\, Ds \;=\; h(g\, q \,.\, Ds) \qquad \text{i.e.,} \quad \begin{array}{ccc} Ds & \xrightarrow{\;f\;} & D \\ {\scriptstyle g\,q\,.}\downarrow & & \downarrow{\scriptstyle q\,.} \\ . & \xrightarrow{\;h\;} & . \end{array} \qquad \text{“distrigration”}$$

5 Goals. What we hope to achieve with this approach (if time, brains and other resources permit!) is the following:

- An *algorithmic* verifiable predicate on triples f, g, h that is equivalent to, or implies the truth of the distrigration equation in paragraph 4.
- A way to *compute* g and h out of a given f (in such a way that distrigration is true).
- Maybe(?), a way to *compute* f out of given g and h (such that distrigration is true).
- In the end, efficiency is what distinguishes mathematics from computing science; so, what about the *efficiency* of the two alternative paths to answering queries?

Presumably, in order to achieve these goals we have to restrict the kind of functions f, g, h we are willing to deal with. Even more, it might be necessary to stipulate a language (“syntax”) for expressing such functions, and restrict f, g, h to functions expressible in that language.

Note that XQuery is such a language: it allows to express functions f (but certainly not all conceivable functions f); a *Query* in XQuery constructs an XML document out of a collection of other XML documents. However, XQuery is tailored, in syntax and expressive power, to efficient automatic processing — which is not our primary concern.

6 The role of schemas. We conceive the role of XML schemas as a means to predict properties, in the same way as typing in (programming) languages predict properties of the well-typed expressions. In particular, if document D matches schema S, and q doesn't “match” schema S, then it might be nonsense to answer q in D, or alternatively the answer $q \,.\, D$ is empty. These kind of predictable properties might be exploited by an algorithmic realization of query answering.

The presence of schemas leads, of course, to additional goals (again, we hope that these are achievable in principle; the amount of work might be too much):

- A construction of g and h out of f in such a way that not only the distrigration equation holds true, but also $g\, q$ is a series of queries that match the schemas Ss; under the assumption that $f\, Ds$ matches the integration schema S if documents Ds match schemas Ss.
- Similarly for constructing f out of g and h.
- Based on some of the previous results, a *proof* that Marko's design method for schema integration is sound.

* * *