AxPRE Summaries: Exploring the (Semi-)Structure of XML Web Collections

Mariano P. Consens #1, Flavio Rizzolo #2, Alejandro A. Vaisman *3

# *University of Toronto, Canada*
1 consens@cs.toronto.edu
2 flavio@cs.toronto.edu

* *Universidad de Buenos Aires, Argentina*
3 avaisman@dc.uba.ar

***Abstract*— This paper introduces *AxPRE summaries*, a formalism that allows exploring the (semi-)structure of large XML collections. AxPRE summaries are implemented in a tool, DescribeX, that supports visualizing XML collections via summaries that can be interactively *refined* using a powerful and descriptive *axis path regular expression* language. Experimental results on gigabyte collections have shown that this flexibility does not come at the expense of efficiency.**

I. Introduction

XML continues to be widely used as a common format for web accessible data as well as for data exchanged among web applications. Compared to the earlier wild web days of abundant (not even well-formed) HTML, there is a clear trend toward applications that validate XML documents against schemas. However, despite schema-validity, the actual structure of web documents exhibits significant variations across collections for several reasons. First, schemas can be very lax (e.g., by extensive use of the `<xsd:choice>` construct in XML schema). This is the case for RSS feeds (a format used by content distributors to deliver frequently updated content over the Web). Second, a schema can be very large and only subsets are actually used in a given instance. This is the situation with several industry specific standards that contain hundreds of elements (such as UBL [1] or HR-XML[2]). Finally, a schema can be extended with elements from other namespaces and schemas.

Many web development tasks resort to XPath queries to process XML documents and require an understanding of the actual structure present in the collection. Similar challenges are faced by applications issuing XPath queries to process a collection of feeds that incorporates RSS extensions[3] supporting additional elements for describing pictures, podcasts, and videos.

Our work addresses the need to describe the actual metadata structure of large collections of web documents (by and large encoded and processed as XML). The proposed framework is implemented in DescribeX, a tool (demonstrated in [1]) for describing and visualizing XML collections via summaries that can be tailored using a powerful language: *axis path regular expressions* (*AxPRE*, for short).

XML structural summaries are graphs representing relationships between sets of XML elements with a common structure (paths, subtrees, etc.). Describing metadata in semistructured collections was a major motivation in one of the earliest summary proposals in the literature [2]. Since then, research on summaries has focused mostly on query evaluation, indexing and selectivity estimation. Although these are all interesting problems we can address (see [3] for XPath query evaluation using DescribeX), in this paper we focus on metadata exploration, which has become increasingly relevant over the last few years and has received considerably less attention in the summary literature. *AxPRE summaries* have a unique capability that makes them suitable for describing the (semi-)structure of XML collections: they are the first summaries capable of *describing and refining* partitions created using a powerful language.

II. Motivating Example

We motivate our work describing the tasks of Sue, a developer who has to code a web application that retrieves RSS feeds from several content providers to produce an aggregated meta feed. Figure 1 shows two RSS feed instances represented as axis graphs which make use of the Media RSS extension providing support for media syndication. An axis graph is an abstract representation of the XPath data model extended with edges representing binary relations between elements. Selected axes are shown in the figure: fc, ns, and c (for $firstchild$, $nextsibling$, and $child$, respectively). Axis graphs can include binary relations between elements and/or attributes that are not XPath axes per se, like fc, ns, and even $id\text{-}idrefs$.

Sue has access to a repository containing several months of sample feeds published by the content providers (a collection with thousands of XML files). She is planning to prototype an application using XPath patterns based on the samples. However, to write the required XPath expressions Sue has to understand the structure of the feed collection. She could manually open a few files to get a sense of the structure of the entire collection. At some point Sue will have to check if the patterns she has selected are indeed characteristic of the collection. Fortunately, Sue has access to the DescribeX

[1] `http://oasis-open.org/committees/ubl/`
[2] `http://hr-xml.org`
[3] `http://rss-extensions.org`

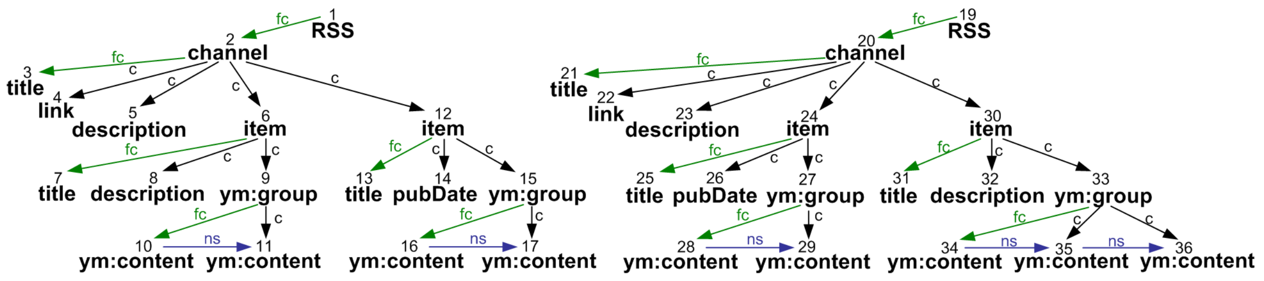

Fig. 1. The axis graph of two sample RSS feeds

tool for exploring the structure of the entire feed collection. The tool can process a collection with thousands of files in a minute to provide a first glimpse of the collection's structure using a label *summary descriptor* (SD, for short), the simplest of the AxPRE summaries created by DescribeX that clusters nodes by their labels.

However, a label SD is not enough for fulfilling Sue's requirements. She needs to characterize feed items that have different structure in order to process them differently. For instance, she will aggregate differently items that contain media in a different number of formats (i.e., items with three content elements will be separated from items with just two). In order to do that, Sue uses DescribeX to interactively *refine* the label SD nodes. The refinement is based on bisimilarity applied to neighbourhoods of nodes described by an AxPRE, a path regular expression on binary relations. Figure 2 depicts an SD where complex refinements, specified using AxPREs, had been applied to nodes of a label SD. The sets of element numbers that appear below the nodes in the figure are called the *extent* of the nodes. An SD edge is labeled by the axis relation it represents (i.e., edge (s_{62}, s_{81}) is labeled by c, which means that there is a c axis relation between some elements in the extent of s_{62} and s_{81}).

Figure 2 shows that AxPRE $c.fc.ns^*$ (SD nodes s_{61}, s_{62} and s_{63}) partitions item elements based on the sequence of their grandchildren labels. Analogously, a refinement with AxPRE ns^* results in SD nodes s_{91}, s_{92} and s_{93}, which separate ym:content elements according to the number of their following siblings. Sue can either choose a refinement from a list suggested by DescribeX containing the most common ones or write her own AxPRE.

Sue can distinguish three kinds of items with different structure beyond their children, a capability not available using DTD's without renaming elements (which sometimes is not an option because the original DTD cannot be modified). In particular, proposals that infer a DTD from an instance (such as [4]) do not help to identify the three kinds of items as done above: a DTD can only give a rule for the children of items, there is no mechanism for giving rules relating items to their grandchildren (or any other elements farther away). In contrast, the AxPRE summary in Figure 2 can represent that the items with three ym:content grandchildren (node s_{63}) are also the items with a description.

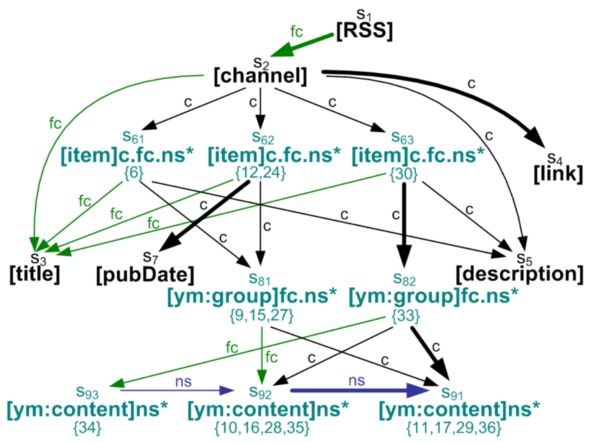

Fig. 2. A refined SD for the RSS samples

III. AxPRE Summaries

This section provides an overview of the DescribeX framework (introduced in [3]). The framework includes a powerful language based on *axis path regular expressions* (AxPREs) for describing the extents in an SD. Extents are defined using a novel notion: bisimilarity applied to neighbourhoods of nodes described by an AxPRE (a path regular expression on binary relations). For representing an XML instance, DescribeX uses a labeled graph model called *axis graph*.

Definition 3.1 (Axis Graph): An axis graph $\mathcal{A} = (Inst, Axes, Label, \lambda)$ is a structure where $Inst$ is a set of nodes, $Axes$ is a set of binary relations $\{E_1^{\mathcal{A}}, \ldots, E_n^{\mathcal{A}}\}$ in $Inst \times Inst$ and their inverses, $Label$ is a finite set of node names, and λ is a function that assigns labels in $Label$ to nodes in $Inst$. Edges are labeled by axis names and nodes are labeled by element or attribute names (including namespaces), or by new labels defined using XPath.

An axis graph is an abstract representation of the XPath data model [5] extended with edges that represent XPath relations between elements. The axis graph can also include additional axes, such as *id-idrefs*, *fc*, *ns*, and other binary relations between elements or attributes that can be expressed in XPath.

DescribeX uses AxPREs for characterizing the sets in the partition of elements. An AxPRE gives a precise description of the elements in the extent of an SD node, something not provided by any other proposal in the literature.

Definition 3.2 (Axis Path Regular Expressions): An *axis path regular expression* is an expression generated by the grammar

$$E \longleftarrow axis \mid axis[B(l)] \mid (E \mid E) \mid (E)^* \mid E.E \mid \epsilon$$

where $axis \in Axes$ and ϵ is the symbol representing the empty expression.

Definition 3.2 describes the syntax of path regular expressions on the binary relations (labeled edges) of the axis graph including node label tests ($B(l)$ is a boolean function on a label $l \in Label$ that supports more elaborate tests on labels, beyond just matching it). AxPREs could also be written using a syntax closer to XPath syntax.

AxPREs are used in DescribeX for defining neighbourhoods of nodes computed by intersecting the automaton of the AxPRE and the axis graph starting from a given node.

Definition 3.3 (AxPRE Neighbourhood of v): Let $\mathcal{A}$ be an axis graph, v a node in $\mathcal{A}$, α an AxPRE, and $NFA(\alpha)$ the Thompson construction finite automaton of α accepting all prefixes. The *AxPRE neighbourhood* of v for α, denoted $\mathcal{N}_\alpha(v)$, is the subgraph of $\mathcal{A}$ product of the intersection between $\mathcal{A}$ and $NFA(\alpha)$ such that v intersects with the initial state of $NFA(\alpha)$.

This approach for defining summaries is based on the intuition that nodes that have similar neighbourhoods should be grouped together in the same extent. DescribeX uses the similarity notion of *labeled bisimulation*, which provides a way of computing a double homomorphism between graphs.

Definition 3.4 (Labeled Bisimulation): Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two subgraphs of an axis graph $\mathcal{A}$, such that $Axes_{\mathcal{G}_1} \subseteq Axes$ and $Axes_{\mathcal{G}_2} \subseteq Axes$. A labeled bisimulation between $\mathcal{G}_1$ and $\mathcal{G}_2$ is a symmetric relation $\approx$ such that for all $v \in \mathcal{G}_1$, $w \in \mathcal{G}_2$, $E_i^{\mathcal{G}_1} \in Axes_{\mathcal{G}_1}$, and $E_i^{\mathcal{G}_2} \in Axes_{\mathcal{G}_2}$: if $v \approx w$, then $\lambda(v) = \lambda(w)$; if $v \approx w$, and $\langle v, v' \rangle \in E_i^{\mathcal{G}_1}$, then $\langle w, w' \rangle \in E_i^{\mathcal{G}_2}$ and $v' \approx w'$.

Example 3.1: Consider elements 12 and 24 in the axis graph of Figure 1. They have bisimilar $[item]c.fc.ns^*$ neighbourhoods and therefore belong to the same set in the partition (the one that corresponds to s_{62} in Figure 2).

The widespread use of bisimulation in summaries is motivated by its relatively low computational complexity properties. The bisimulation reduction of a labelled graph can be done in time $O(m \log m)$ (where m is the number of edges in a labelled graph) as shown in [6], or even linearly for acyclic graphs, as shown in [7]. Using bisimulation also allows us to capture all the existing bisimulation-based proposals in the literature (Section IV).

Definition 3.5 (AxPRE Partition): Let $\mathcal{A}$ be an axis graph, $N \subseteq Inst$, and α an AxPRE. An *AxPRE partition* of N for α, denoted $\mathcal{P}_\alpha(N)$, is a partition of the nodes in N defined as follows: two nodes $v, w \in N$ belong to the same class $P_i \in \mathcal{P}_\alpha(N)$ iff there exists a *labeled bisimulation* $\approx$ between $\mathcal{N}_\alpha(v)$ and $\mathcal{N}_\alpha(w)$ such that $v \approx w$.

Definition 3.6 (Summary Descriptor): A *summary descriptor* (SD for short) of an axis graph $\mathcal{A}$ consists of a partition $\{N_i\}_i$ of $\mathcal{A}$ and a set of AxPRE partitions $\{\mathcal{P}_{\alpha_i}(N_i)\}_i$, $1 \leq i \leq m$, together with a labeled graph G, called *SD graph*, representing axis relationships between elements in the equivalence classes of the AxPRE partitions. Each node s in the SD graph has associated one set in some $\mathcal{P}_{\alpha_i}(N_i)$ called the *extent* of s and is labeled by α_i. Edges are labeled by the $axis$ relation they represent.

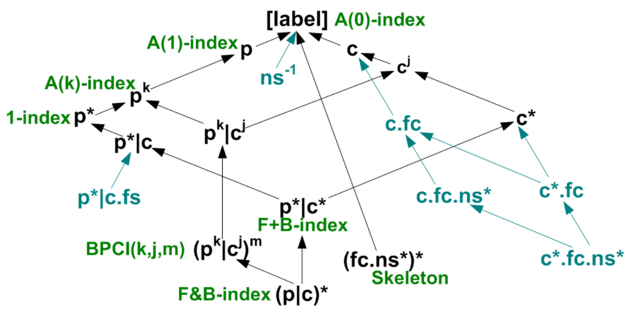

Fig. 3. DescribeX lattice capturing earlier homogeneous proposals

When the extents of all nodes in a SD $\mathcal{D}$ are defined with the same AxPRE α we have an *homogeneous* SD. In this case we say that $\mathcal{D}$ is an α SD. In contrast, if at least two different nodes are defined with different AxPREs we have an *heterogeneous* SD.

IV. DescribeX Lattice

DescribeX summaries can be classified in a lattice that describes a *refinement* relationship between homogeneous summaries. Figure 3 shows a fragment of such a lattice capturing earlier proposals based on the notion of bisimilarity. Each node in the lattice corresponds to a homogeneous SD defined by an AxPRE. The node labels indicated in green are the names of the proposal that each node captures [8]. Nodes and edges in blue are a sample of the richer SDs that were never considered in the literature, like the one that appears in Figure 2 ($c.fc.ns^*$).

V. Acknowledgments

Financial support was provided by the Natural Sciences and Engineering Research Council (NSERC), an IBM Faculty Award, and an IBM Center for Advanced Studies Fellowship.

References

[1] M. S. Ali, M. P. Consens, S. Khatchadourian, and F. Rizzolo, "DescribeX: interacting with AxPRE summaries," in *ICDE*, 2008.

[2] S. Nestorov, J. D. Ullman, J. L. Wiener, and S. S. Chawathe, "Representative objects: Concise representations of semistructured, hierarchial data," in *ICDE*, 1997.

[3] M. P. Consens and F. Rizzolo, "Fast answering of XPath query workloads on web collections," in *XSym*, 2007.

[4] G. J. Bex, F. Neven, T. Schwentick, and K. Tuyls, "Inference of concise DTDs from XML data," in *VLDB*, 2006.

[5] W3C, "XPath 2.0," 2002, http://www.w3.org/TR/xpath20.

[6] R. Paige and R. E. Tarjan, "Three partition refinement algorithms," *SIAM Journal on Computing*, vol. 16, no. 6, pp. 973–989, 1987.

[7] A. Dovier, C. Piazza, and A. Policriti, "An efficient algorithm for computing bisimulation equivalence," *Theoretical Computer Science*, vol. 311, no. 1-3, pp. 221–256, 2004.

[8] M. P. Consens, F. Rizzolo, and A. A. Vaisman, "Exploring the (semi-)structure of XML web collections," UofT - DCS, Tech. Rep., 2007, http://www.cs.toronto.edu/∼ consens/describex/.