Detecting Structural Similarities between XML Documents

Sergio Flesca[1], Giuseppe Manco[2], Elio Masciari[2], Luigi Pontieri[2], and Andrea Pugliese[1,2]

[1] DEIS, University of Calabria, Via Bucci 41c, 87036 Rende (CS) - Italy
email: {flesca,apugliese}@si.deis.unical.it
[2] ISI-CNR, Via Bucci 41c, 87036 Rende (CS) - Italy
email: {manco,masciari,pontieri}@isi.cs.cnr.it

Abstract. In this paper we propose a technique for detecting the similarity in the structure of XML documents. The technique is based on the idea of representing the structure of an XML document as a time series in which each occurrence of a tag corresponds to a given impulse. By analyzing the frequencies of the corresponding Fourier transform, we can hence state the degree of similarity between documents. The efficiency and effectiveness of this approach are compelling when compared with traditional ones.

1 Introduction

In this work we address the problem of identifying similarities between XML documents. In recent years, XML has been gaining increasing relevance as a means for exchange information. As a matter of fact, most web applications deal with web data by translating them into XML format, and many commercial database systems (Oracle, IBM DB2) provide tools to deliver information in XML format and to store XML data into relations. An interesting approach to efficiently store and retrieve XML documents is based on grouping together similar XML documents [6]. Algorithms for clustering documents according to their structural similarity could be effectively supported by our technique. Two relevant fields of application of our technique are the integration of semistructured data and the web site structural analysis. Indeed, grouping structurally similar documents can help in both recognizing sources providing the same kind of information and in presenting the information provided by a site. Several methods for detecting the similarity of XML documents [5, 4] have been recently proposed, that are based on the concept of edit distance [9] and use graph-matching algorithms to calculate a (minimum cost) edit script capable of transforming a document into another. Most of these techniques are computationally expensive, i.e. at least $O(N^3)$, where N is the number of element of the two documents. Actually, the sub-optimal technique proposed in [5], works in quasi-linear time. However, all of them are concerned with the detection of changes occurring in XML documents rather than comparing them on the basis of their structural similarity. A different approach is adopted in [3], where a technique for measuring the similarity of a document versus a DTD is introduced. This technique exploits a graph-matching algorithm, which associates elements in the document with element definitions in the DTD. This approach does not seem to be directly applicable to cluster documents without any knowledge about their DTDs, and is not able to point out dissimilarities among documents referring to the same DTD.

Our aim and strategy are completely different. Indeed, we propose to represent the structure of an XML document as a time series, where each tag occurrence corresponds to an impulse. By analyzing the frequencies of the Fourier Transform of such series, we can state the degree of (structural) similarity between documents. As a matter of fact, the exploitation of the Fourier transform to check similarities among time series is not completely new (see, e.g., [1, 8]), and has been proven successful. The main contribution of our approach is the systematic development of an effective encoding scheme for XML documents, in a way that makes the use of the Fourier Transform extremely profitable. Efficiency and effectiveness of our approach are compelling when compared to the above mentioned ones, as we shall show in the rest of the paper.

2 Preliminaries

To our purposes an XML document can be considered as a tree of elements. A pair of tags delimits the area of the document that contains the information associated with any element. In turn, each element may contain further elements, as well as unstructured data (e.g. text). As an example consider the toy XML document shown below, and representing information about books.

```
<xml>
    <book year="1997">
        <title> A First Course in Database Systems </title>
        <author> Ullman </author>
        <author> Widom  </author>
        <publisher> Prentice-Hall </publisher>
    </book>
</xml>
```

Given an XML document d, we denote by $tags(d)$ the *tag set* of the document d, i.e. the set of all the tags occurring within the document; moreover, $tnames(d)$ denotes the set of all the distinct tag names appearing in d. As an example, the *tag set* of the XML document shown above is: {`<xml>`, `<book>`, `<title>`, `</title>`, `<author>`, `</author>`, `<author>`, `</author>`, `<publisher>`, `</publisher>`, `</book>`, `</xml>`}. Moreover, for the same document $tnames$ = {`xml`, `book`, `title`, `author`, `publisher`}. Observe that tags with the same name are not considered to be the same object, so that `<author>` appears twice in the tag set, whereas the set of tag names does not contain duplicates. Furthermore, given a element el of an XML document d, we denote by el_s the starting tag of el and with el_e an ending tag of el. Obviously, the definitions of the sets *tags* and *tnames* can be straightforwardly extended to deal with sets of XML documents.

Given an XML document d, we define the *skeleton* of d as the sequence of tags appearing within d, i.e. $sk(d)$ is the sequence $[t_0, t_1, \cdots, t_n]$ such that $t_i \in tags(d) \Leftrightarrow t_i \in sk(d)$ and t_i precedes t_j within d if and only if $i > j$. In our approach, the skeleton of an XML document represents a description of the document structure. Moreover, it can be looked at as an XML document (with empty element content). As an instance, the skeleton of the previous document is: `<xml>`, `<book>`, `<title>`, `</title>`, `<author>`, `</author>`, `<author>`, `</author>`, `<publisher>`, `</publisher>`, `</book>`, `</xml>`.

Finally, given a document d and a tag t in it, we define $nest_d(t)$ as the set of the start tags el_s in d occurring before t and for which there is no end tag el_e matching el_s and appearing before t. We also denote by l_t the nesting level of the tag t, i.e. $l_t = |nest_d(t)|$. For a given set D of documents, $maxdepth(D)$ denotes the maximum nesting level of tags appearing in a document $d \in D$.

3 Detecting Document Similarities

In this section we introduce our technique for detecting XML structural similarity. Intuitively, two documents are said to have a similar structure if they correspond in the type of elements they contain and in the way these elements are combined in the two documents. In the following we provide a way to estimate this similarity by decomposing this high-level statement.

The main idea of the approach is that of representing the skeleton of a document as a time series. More precisely, we can assume that we are visiting the tree-structure of an XML document in a depth-first, left-to-right way. As soon as we visit a node of the tree, we emit an impulse containing the information relevant to the tag. The resulting signal shall represent the original document as a time series. As a consequence, the comparison of two documents is done by looking at their corresponding signals. In the following, we first describe a technique for encoding the structure of a document into a time series, and next we define how to measure the similarity of such signals.

3.1 Document Structure Coding

A simple association of each tag name with a given number usually does not suffice to specify a suitable translation of a document. Indeed, the resulting time series is required to summarize the main features of the document. Moreover, we need to represent these features in a suitable way, so

that we can effectively distinguish two different documents by simply looking at their encodings. In this respect, we have considered several ways of encoding an XML document, obtained by specifying an encoding method for both the tags and the structure of the document. In a sense, a tag encoding corresponds to the analysis of the *locality* of a tag. On the other side, the nesting of different tags within the whole document provides an *overall* perspective: we look at the document as a globally uniform entity.

Tag Encoding. Given a set D of XML documents, a function γ from $tags(D)$ to $\mathbb{N}$ is a *tag encoding function* for D. We can assign a number n to each tag in several ways: for instance, by generating it randomly, or using a hash function. However, a good tag encoding function should at least ensure *injectivity*, i.e., tags having different name are associated with two different numbers: for obvious reasons, collisions correspond to losing relevant information. A further desirable property is the capability to *contextualize* a given tag, i.e., to capture information about its neighbors.

We studied several tag encoding functions and in this work we will explain the one that guaranteed a good compromise between efficiency and accuracy in the encoding. This tag encoding function, called *direct tag encoding* and denoted by γ_d, is built up as below specified.

Given a set D of XML documents, we build a sequence of distinct tag names $[tn_1, tn_2, \cdots, tn_k]$ by considering a (randomly chosen) linear order on $tnames(D)$. Given an element el, the direct encoding simply associates each tag el_s with the position n of the tag name tn of el in the sequence ($\gamma_d(el_s) = n$). For the end tags, we consider two possible versions: either symmetric or null. A tag encoding function γ is said to be *symmetric* iff for each document d and for each element $el \in d$, $\gamma(el_e) = -\gamma(el_s)$; γ is *null* if $\gamma(el_e) = 0$ (for all d and el). For instance, the direct symmetric encoding of the document shown above is: $\gamma_d(\texttt{<xml>}) = 1$, $\gamma_d(\texttt{<book>}) = 2$, $\gamma_d(\texttt{<title>}) = 3$, $\gamma_d(\texttt{</title>}) = -3$, $\gamma_d(\texttt{<author>}) = 4$, $\gamma_d(\texttt{</author>}) = -4$, $\gamma_d(\texttt{<publisher>}) = 5$, $\gamma_d(\texttt{</publisher>}) = -5$, $\gamma_d(\texttt{</book>}) = -2$, $\gamma_d(\texttt{</xml>}) = -1$.

Notice that choosing a nondeterministic order on $tnames(D)$ avoids any possibility of exploiting context information. A possibility for improving the proposed scheme is that of imposing a significant order (e.g., by relating the position of each tag with its maximum nesting level within the documents). However, different approaches for representing tag context information may be defined, but we do not expose them here for the sake of conciseness.

Document Encoding. A document encoding is essentially a function that associates an XML document with a time series, representing the structure of the document. Let D be the set of all the possible XML documents. A *document encoding* is a function that associates each $d \in D$ with a sequence of positive integers, i.e. $enc(d) = h_0, h_1, \cdots, h_n$.

A document encoding *enc* is said to be *without structural loss* (*WSL*) iff for each pair of documents d_1, d_2, $enc(d_1) = enc(d_2)$ implies that $sk(d_1) = sk(d_2)$. Of course, the *WSL* property is desirable because it implies that we do not lose information about the document structure when considering its encoding, and we can reconstruct the structure of a document from its encoding. However, even if the *WSL* property holds, we are not guaranteed that we can effectively distinguish two documents by simply comparing their encodings. We have examined several document encoding functions for representing the document structure, that exploit a tag encoding function to identify suitable sequences.

In this work we will show only one of them, namely the *multilevel encoding*, which captures, in a satisfactory way, the structure of XML documents. This encoding strategy weights each tag t by using its level of nesting l_t. In particular, consider a set D of XML documents, a document $d \in D$ with $sk(d) = [t_0, \cdots, t_n]$ and γ a tag encoding function for D. Then, a *multilevel encoding* of d is a sequence $[S_0, S_1, \cdots, S_n]$, where:

$$S_i = \gamma(t_i) \times B^{maxdepth(D) - l_{t_i}} + \sum_{t_j \in nest_d(t_i)} \gamma(t_j) \times B^{maxdepth(D) - l_{t_j}}$$

Usually, we set $B = |tnames(D)+1|$. This choice, together with the use of the symmetric direct tag encoding function, avoids to "mix" together the contributions of different nesting levels. Indeed, by simply considering only a value in the time series, associated with a tag t, it is possible to completely reconstruct the path name of t.

3.2 Similarity Measures

Faced with the above definitions, we can now detail the similarity measure for XML documents. As already mentioned, we suppose that we are visiting the tree-structure of an XML document d (in a depth-first, left-to-right way) starting from an initial time t_0. We also assume each node (tag) within the tree is found after a fixed time interval Δ. The total time spent to visit the document shall be $t_0+N\Delta$ (where N is the size of $tags(d)$). During the visit, as we find a start-tag, we produce an impulse, that is assumed to stand until we reach the corresponding end-tag, and depends from a given tag encoding e and the overall structure of the document (i.e., the document encoding enc).

As a result of the above physical simulation, the visit of the document produces a signal $h_d(t)$, that usually changes its intensity, in the time interval $[t_0, t_0 + N\Delta)$. The intensity variations are directly related to the presence/absence of tags:

$$h_d(t) = \begin{cases} [enc(d)](k) & \text{if } t_0 + k\Delta \leq t < t_0 + (k+1)\Delta \\ 0 & \text{if } t < t_0 \text{ or } t > t_0 + N\Delta \end{cases}$$

Comparing two such signals, however, can be as difficult as comparing the original documents. First of all, comparing documents having different lengths requires to combine resizing and alignment, in a way that can be particularly difficult to define. Stretching (or narrowing) signals is not a solution, since even with two signals having equal length the problem of defining the correct correspondences among the impulses can be extremely difficult. Finally, the intensity of a signal strongly depends on the encoding scheme adopted, that, in turn, may depend on the context (as in the case, e.g., of the multilevel document encoding scheme). To this purpose it is particularly convenient to examine the DFT of the signal, since it reveals much about the distribution and meaning of signal frequencies. In a sense, given an encoding $h_d(t) = enc(d)$ of a document d, we can define a function $\tilde{h}_d(t)$ as the periodic extension of the $h_d(t)$ function. Hence, we are *windowing* the time series $\tilde{h}_d(t)$ of the document. In our particular case, we can compare the structure of two XML documents by exploiting the Fourier transforms. Given a document d, we denote as $\texttt{DFT}(enc(d))$ the Discrete Fourier Transform of the time series resulting from the encoding. In particular, such a transform represents the frequencies ranging within the interval $[-.5, .5]$ (obtained by choosing the value $\Delta = 1$).

In order to compare two documents d_i and d_j, hence, we can work on their DFT transforms. In particular, a possibility [1, 8] is to exploit that, from Parseval's theorem, energy is an invariant in the transformation. However, a more effective discrimination can simply exploit the difference in the single frequencies: in a sense, we are interested (i) in abstracting from the length of the document, and (ii) in knowing whether a given subsequence (representing a subtree in the original XML document) exhibits a certain regularity, no matter where the subsequence is located within the signal. Following the above interpretation, we can measure the distance between two documents by computing the integral, over the given frequency range, of the magnitude difference of their transforms. In the discrete interpretation of the Fourier transform, each point in the transform corresponds to a given frequency. Now, if d_i and d_j, have different size, the corresponding (discrete) transform shows values corresponding to different frequencies. In order to approximate the above integral, we have to interpolate the corresponding values. In particular, the interpolation shall produce a new transform $\tilde{\texttt{DFT}}$ for each documents, having $M = N_{d_i} + N_{d_j} - 1$ points (where $N_{d_i} = |tags(d_i)|$ and $N_{d_j} = |tags(d_i)|$).

More formally, the overall computation of the dissimilarity between documents can be done as follows. Let d_1, d_2 be two XML documents, and enc be a document encoding, such that $h_1 = enc(d_1)$ and $h_2 = enc(d_2)$. Let DFT be the Discrete Fourier Transform of the (normalized) signal.

We define the *Discrete Fourier Transform* distance of the documents as the approximation of the difference of the magnitudes of the two signals:

$$dist(d_1, d_2) = \left(\sum_{k=1}^{M/2} \left(|[\tilde{\mathtt{DFT}}(h_1)](k)| - |[\tilde{\mathtt{DFT}}(h_2)](k)| \right)^2 \right)^{\frac{1}{2}}$$

where $\tilde{\mathtt{DFT}}$ is an interpolation of DFT to the frequencies appearing in both h_1 and h_2, and M is the total number of points appearing in the interpolation. Alternative ways of comparing the documents can be defined. For example, we can choose to compare only a given number of values in the transforms, in the style of [1, 8]. However, it is important to stress here that we aim at comparing w.r.t. graph-based approaches. To this purpose, it is worth noticing that the complexity of the computation of the above distance is mainly influenced by the computation of the DFT .
As a matter of fact, when comparing two documents with length N, our method requires $O(N \log N)$, since computing their transforms is $O(N \log N)$.

4 Experimental Results

In this section we describe some experiments we performed to evaluate the effectiveness of the proposed method in measuring the structural similarity among XML documents. We assessed the validity of the proposed approach by comparing its results to some prior knowledge about the document similarities. Indeed, any dataset we considered consists of XML documents conforming to one of some previously chosen DTDs, so that it may be looked at as a set of structurally homogenous groups. For the sake of presentation we shall refer to any of such groups as *document class* or, more shortly, as *class*. We carried out several experiments on both real and synthesized datasets. In the following we describe some of the results on synthesized data, generated from the 5 DTDs shown in fig. 1. Some of the experiments we conducted on real data are presented in the Appendix.

```
<!DOCTYPE DTD1 [
   <!ELEMENT XML (a*) >
   <!ELEMENT a (b,c,d,e*) >
   <!ELEMENT b (f?) >
   <!ELEMENT c (g|h) >
   <!ELEMENT d EMPTY >
   <!ELEMENT e EMPTY >
   <!ELEMENT f EMPTY >
   <!ELEMENT g EMPTY >
   <!ELEMENT h EMPTY >
]>
```

```
<!DOCTYPE DTD2 [
   <!ELEMENT XML (a1*) >
   <!ELEMENT a1 (b1,c1,d1*,e1) >
   <!ELEMENT b1 (f1?) >
   <!ELEMENT c1 (g1|h1) >
   <!ELEMENT d1 EMPTY >
   <!ELEMENT e1 EMPTY >
   <!ELEMENT f1 EMPTY >
   <!ELEMENT g1 EMPTY >
   <!ELEMENT h1 EMPTY >
]>
```

```
<!DOCTYPE DTD3 [
   <!ELEMENT XML (h*) >
   <!ELEMENT h (f,g) >
   <!ELEMENT f (d*) >
   <!ELEMENT g (b|c) >
   <!ELEMENT d (a?) >
   <!ELEMENT e EMPTY >
   <!ELEMENT b EMPTY >
   <!ELEMENT c EMPTY >
   <!ELEMENT a EMPTY >
]>
```

```
<!DOCTYPE DTD4 [
   <!ELEMENT XML ((x,y)*) >
   <!ELEMENT x ((a,w)|z*) >
   <!ELEMENT x ((a,w)|z*) >
   <!ELEMENT x ((a,w)|z*) >
   <!ELEMENT a EMPTY >
   <!ELEMENT w (c?) >
   <!ELEMENT c EMPTY >
   <!ELEMENT z (v,c) >
   <!ELEMENT v EMPTY >
]>
```

```
<!DOCTYPE DTD5 [
   <!ELEMENT XML (m*,n) >
   <!ELEMENT m (q*) >
   <!ELEMENT q (x,y) >
   <!ELEMENT x ((a,c)|z*) >
   <!ELEMENT a EMPTY >
   <!ELEMENT c EMPTY >
   <!ELEMENT z EMPTY >
   <!ELEMENT n EMPTY >
]>
```

Fig. 1. Example DTDs for Synthesized Data

In order to build synthetic data sets, we implemented an XML document generator, which can produce a set of documents from a given DTD, according to various statistical models. Within each expression defining a DTD element, this system associates any occurrences of the operators $*$ and $+$ with a log-normal stochastic variable representing the length of the sequence that may be produced. Analogously, | and ? operators are modelled by Bernoulli tests. The result of the experiments is a matrix representing the structural similarity degree for each pair of XML documents in the data set. In order to give an immediate and overall perception of the similarity relationships in the data set, we draw, in Figure 2, the similarity matrix as an image, where the grey level of each pixel is proportional to the value stored in the corresponding cell of the matrix.

Moreover, we introduce some summary measures to support simple quantitative analysis. Since we are interested in evaluating how much the similarity measure recognizes the a priori known class affinities, we compute all average intra-class similarities and all inter-class similarities. To this purpose, we report below a matrix CS, where the element $CS(i, j)$ contains the average of the similarity values corresponding to every pair of distinct documents such that the first belongs to the class C_i and the second belongs to the class C_j.

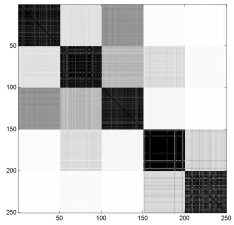

Fig. 2. Multilevel Encoding

	$\mathbf{C_1}$	$\mathbf{C_2}$	$\mathbf{C_3}$	$\mathbf{C_4}$	$\mathbf{C_5}$
$\mathbf{C_1}$	0.9655	0.6418	0.8153	0.4822	0.3935
$\mathbf{C_2}$	0.6418	0.9684	0.7485	0.6586	0.5037
$\mathbf{C_3}$	0.8153	0.7485	0.9619	0.5402	0.4313
$\mathbf{C_4}$	0.4822	0.6586	0.5402	0.9782	0.6817
$\mathbf{C_5}$	0.3935	0.5037	0.4313	0.6817	0.9452

The results obtained by the encoding scheme here analyzed are very interesting. Indeed, they show that the method produces a neat distinction between elements belonging to a class and element outside that class. In addition, it is capable to capture structural affinities relating XML documents belonging to different classes. For instance, the results evidence a relatively high degree of resemblance between the classes 1 and 3, whose DTDs exhibit quite similar structures.

5 Conclusions and Future Works

In this paper we showed an approach for measuring the structural similarity between XML documents. We proposed to represent the XML documents as time series and compute the structural similarity between two documents by exploiting the Discrete Fourier Transform of the corresponding signals. Experimental results showed the effectiveness of our approach, with particular reference to the proposed encoding schemes. Our technique could be refined by exploiting information retrieval techniques. In particular, the combination of the distance measure we propose with traditional techniques, such as Jaccard or Cosine similarity, can be extremely profitable. Furthermore an FFT-based distance measures different from the one introduced here could be used.

References

1. R. Agrawal, C. Faloutsos, and A. Swami. Efficient Similarity Search in Sequence Databases. In *Procs. 4th Int'l Conf. of Foundations of Data Organization (FODO'93)*, 1993.
2. R. Baumgartner, S. Flesca, and G. Gottlob. Visual Web Information Extraction with Lixto. In *Procs. 27th VLDB Conf. (VLDB'01)*, pages 119–128, 2001.
3. E. Bertino et al. Measuring the stuctural similarity among XML documents and DTDs. Technical Report DISI-TR-02-02, Department of Computer Science, University of Genova, 2002. Available at `http://www.disi.unige.it/person/MesitiM`.
4. S. S. Chawathe et al. Change Detection in Hierarchically Structured Information. In *Procs of the Int'l Conf. on Management of Data (SIGMOD'96)*, pages 493–504, 1996.
5. G. Cobena, S. Abiteboul, and A. Marian. Detecting Changes in XML Document. In *18th Int.l Conf. on Data Engineering (ICDE 2002)*, 2002. To appear.
6. J. McHugh and J. Widom. Query Optimization for XML. In *Procs. 25th VLDB Conf. (VLDB'99)*, pages 315–326, 1999.
7. G. Mecca, P. Merialdo, and P. Atzeni. Araneus in the Era of XML. *IEEE Data Engineering Bulletin*, 22(3):19–26, 1999.
8. D. Rafiei and A. Mendelzon. Efficient Retrieval of Similar Time Series. In *Procs. 5th Int'l Conf. of Foundations of Data Organization (FODO'98)*, 1998.
9. K. Zhang and D. Shasha. Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.

In this appendix we show further experiments we performed but this section is not required for the understanding of the paper.

A Experiments on Real Data

This paragraph describes the results of the experiments we performed on real XML documents extracted from collections available on the Internet. The documents used can be collected in three main classes:

- `astronomy`, a sample of 217 documents extracted from the XML-based metadata repository. Such a repository describes the archive of catalogs and journal publications maintained by the *Astronomical Data Center*[1] at NASA/GSFC. In particular, each document contains the metadata for a dataset and all of the associated tables, descriptions, and history.
- `sigmod`, a sample of 51 XML documents containing issues of SIGMOD Record. Such documents were obtained from the XML version of the ACM SIGMOD web site [2] produced within the *ARANEUS* project [7].
- `wrapper`, 53 XML documents representing wrapper programs for web sites, obtained by means of the *LIXTO* system [2].

Each of the three classes has an associated DTD. Thereby, we expect to obtain a first, loose-grain separation of the documents according to their DTD. However, it is worth noticing that the distributions of the tags within the documents is quite heterogeneous, due to the complexity of the chosen DTDs and the semantic variety of the documents. In particular, wrapper programs (belonging to the third class) may have substantially different forms, as a natural consequence of the structural differences existing among the various web sites they are built on, and of the dynamic nature of these sites. As a consequence, we expect a finer-grain analysis to be able of detecting such differences.

In fig. 3 we show the similarity matrix produced by the Multilevel encoding scheme, and report the corresponding average values in the following table.

	$\mathbf{C_1}$	$\mathbf{C_2}$	$\mathbf{C_3}$
$\mathbf{C_1}$	0.8036	0.6195	0.0186
$\mathbf{C_2}$	0.6195	0.9772	0.0210
$\mathbf{C_3}$	0.0186	0.0210	0.2745

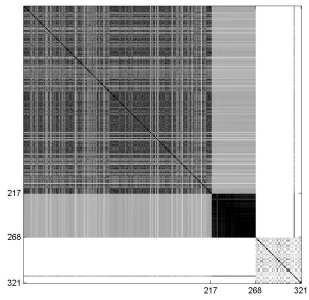

Fig. 3. Similarity Matrix for Multilevel Encoding on `astronomy`, `sigmod` and `wrapper`

[1] http://adc.gsfc.nasa.gov/
[2] http://www.dia.uniroma3.it/Araneus/Sigmod/