

# SEARCH AND INFERENCE WITH DIAGRAMS

Michael Wollowski  
Rose-Hulman Institute of Technology  
5500 Wabash Ave., Terre Haute, IN 47803  
USA  
wollowski@rose-hulman.edu

## ABSTRACT

We developed a process for presenting diagrammatic information on the World Wide Web such that the information is searchable and accessible to inference engines. We implemented this process for chart diagrams. Generally, information that is presented diagrammatically is stored in an image file. Typical search engines have access only to image tags and surrounding text. Hence, the information presented in such image files is not accessible to search engines. The process we developed stores information in XML documents conforming to an XML Schema. A server-side rendering engine is called to present the information to the user graphically. The software system we developed enables us to describe, store, and present bar, line, and pie charts. We feel that our process enables us to get the best of both worlds: people like certain information to be presented diagrammatically and simultaneously want to locate such information with search engines and use inference engines to extract information which is implicitly contained in the data.

## KEY WORDS

Search, inference, data mining, diagrams, XML

## 1. Introduction

We developed and implemented a process for presenting bar, line, and pie charts on the World Wide Web while making that information available to search and inference engines.

Many people prefer diagrams over sentences to display certain information [5]. This is the case for information typically displayed in bar, line, and pie charts. An example can be found in figure 1. In this diagram, people find it easy to locate simple information such as the GPA for a particular year. More importantly, information such as trends are very easily read-off. For example, one can tell that the GPA of both men and women have been increasing steadily. Similarly, comparisons between data for a given year can be easily read off. Such information is not explicitly represented in the chart but can be made explicit by inference.

Typically, information that is presented diagrammatically is stored in an image file on the web. However, information presented in an image file is not accessible to search engines. While a search engine may be able to use the image tags and surrounding text to make an educated guess about what an image represents, in the case of diagrams (as compared to photos) it is particularly desirable to be able to access and search information contained within the image.

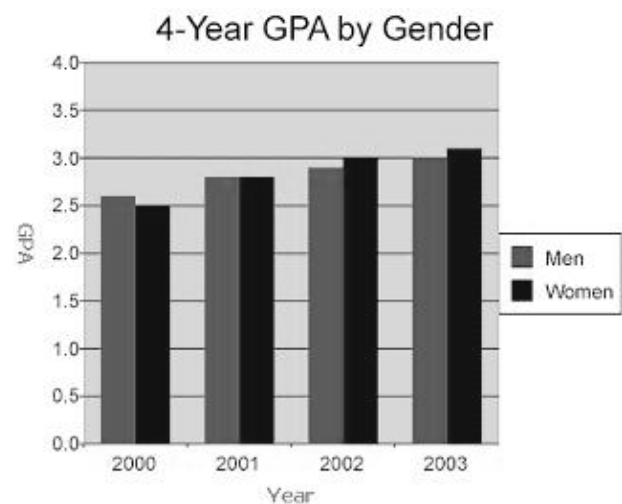


Figure 1. The rendered bar-chart of the XML file of figure 3.

To solve this problem, we store information in an XML file on the web, yet render them as SVG diagram when requested. We developed a custom-fit XML search and inference engine, which enables users to search for particular information contained in the XML documents.

Our system works for any such charts, independent of the information stored in them. The XML Schema of our system can be used to convert information from Microsoft Excel Spreadsheets into XML format. We feel that our system can be adapted to other diagrams, such as wiring diagrams, flowcharts, topological maps.

## 2. Background

Diagrammatic inference systems are special-purpose systems. They differ from first-order logic in that they are custom-fit for the domain for which they are intended. This means that they are limited to that domain, but they are usually complete [2, 6, 8, 9].

Technically speaking, the information in the diagrams shown in this paper are heterogeneous, in the sense that they contain diagrammatic components and sentential components. Without the sentential annotations, the diagrammatic components would be without relevant information about the domain; they would simply be pretty pictures.

A good diagram displays information about the domain in a way which makes it easier for people to read off information and to make inference about information implicitly contained in the diagram. For example, increasing bar sizes represent a trend. As can be seen in this paper, a diagram is constructed from nested lists is information which can easily be turned into first order logic statements. As such, one can use the tools of first-order logic to infer explicit or implicitly information contained in the statements. However, a similar approach can be taken for diagrammatic or heterogeneous representations. Just as sentences serve to make partial descriptions of a domain, so can diagrams. “Truth in a model” is a relation. In a similar fashion, we can define a relationship between a diagram and a model or a diagram and some well-formed representation, such as first-order logic. Naturally, one has to give a formal syntax for diagrams in the system. Once the correspondence between the two systems is established, one can develop a notion of logical consequence. That this can be done has been shown in [3].

Our work draws on that of Jon Barwise and John Etchemendy. They looked at diagrammatic reasoning from the standpoint of valid reasoning [4]. Chahab Natar is working on automatically indexing large image and photo databases for the purpose of fast retrieval as well as to assess similarity [7].

## 3. Process Overview

The components of our system are shown in figure 2. At first, one has to develop a language for describing the information stored in a class of diagrams. This language is specified using an extensible mark-up language schema definition (XML Schema). Each diagram is stored on the web-server as an XML file satisfying the XML Schema and, as such, is available to search engines. When requested by a browser, this file is processed by a server-side rendering program, which turns it into an image format that the browser's machine can display.

For our implementation, we chose the diagrammatic sub-domain of charts, specifically bar, line, and pie charts. Our server-side rendering program was written in Java and produces images using the scalable vector graphics (SVG) standard. (Currently, browsers typically use Adobe SVG Viewer to render SVG files.) We adapted the XML Schema so that it can be used by Excel to export spreadsheet data into XML format.

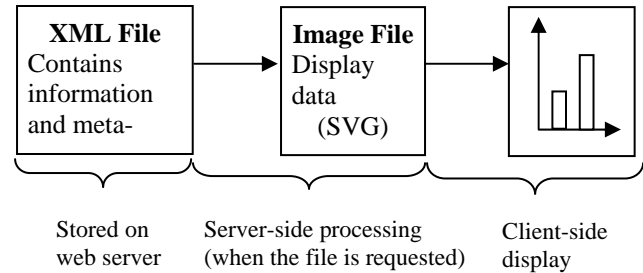


Figure 2: The primary components of our system.

## 4. An Example

In the following example, we show the format of an XML document and how it relates to the rendered image. Figure 3 contains the XML file and figure 1 the rendered image. Data is boldfaced.

The top level element is `chart` and contains a single attribute `type` with which to specify the chart type, since each chart type must be rendered differently. In this example, we show a bar chart. The chart's title is contained in the `title` element and is displayed above the chart proper. The title assists the human viewer in identifying the kind of information presented. The search engine will also rely heavily on the information presented in the title element.

A `legend` element enables the user to specify (by its presence or absence) whether a legend should be generated in the resulting image. The actual label information is pulled from the dataset elements, as described below.

The information enclosed in the `plotarea` element is of a technical as well as semantic nature. It contains `axis` elements that specify relevant information about each axis. Currently, our system only renders two dimensional charts, designated by `variable` attributes with the values "x1" and "x2". The text contained within an `axis` element is used to label it. For a bar chart, the x2-axis specifies a starting and ending value (`min` and `max` attribute) as well as a gradation for the vertical axis (`step` attribute). In contrast, the horizontal x1-axis does not contain these attributes and is instead divided based on the string data present in the x1 attributes of `datapoint` elements. For a line chart, both axes contain numerical data and hence would look like the x2-axis of the

example. In contrast, a pie chart has no axes and has no need of a `plotarea` element.

Each set of data is enclosed in a separate `dataset` element. This element has a `label` attribute identifying the dataset in the legend. In our example, we have data for men and for women. A `dataset` consists of `datapoint` elements.

```
<?xml version="1.0" ?>
<chart type="bar">
  <title>4-Year GPA by Gender</title>
  <legend />
  <plotarea>
    <axis variable="x1">Year</axis>
    <axis variable="x2" min="0" max="4.0"
      step="0.5">GPA</axis>
  </plotarea>
  <dataset label="Men">
    <datapoint x1="2000" x2="2.6" />
    <datapoint x1="2001" x2="2.8" />
    <datapoint x1="2002" x2="2.9" />
    <datapoint x1="2003" x2="3.0" />
  </dataset>
  <dataset label="Women">
    <datapoint x1="2000" x2="2.5" />
    <datapoint x1="2001" x2="2.8" />
    <datapoint x1="2002" x2="3.0" />
    <datapoint x1="2003" x2="3.1" />
  </dataset>
</chart>
```

**Figure 3. A sample XML file.**

Every `datapoint` has two attributes identifying the values across the horizontal axis (`x1`) and the vertical axis (`x2`). Data points are represented by bars in a bar chart and through a connected line in a line chart. In a pie chart, each `dataset` represents a slice of the pie, and instead of containing `datapoint` elements, it has a second attribute, `value`.

## 5. Syntax of Charts

In this section we provide a specification of the syntax for the XML documents in our system. The reader is encouraged to refer to the sample document found in figure 3. The full syntax is defined in an XML Schema (see [1]). By taking advantage of type derivation allowed in XML Schemas, our language accommodates for the wide disparity between bar, line, and pie charts. The element names of our system are: `chart`, `title`, `legend`, `plotarea`, `axis`, `dataset`, and `datapoint`. Some of these elements have attributes: `type`, `variable`, `min`, `max`, `step`, `label`, `value`, `x1`, and `x2`. The occurrence constraints and allowable values for both attributes and elements vary depending on the chart type.

### 5.1 Root Element

The root of the document is the `chart` element. It will always have one attribute called `type`. The value of the `type` attribute is either "bar", "line", or "pie" and is used to determine the type of chart. Its children are the elements: `title`, `legend`, `plotarea`, and `dataset`.

### 5.2 Title and Legend

The `title` element is a child of the root (`chart`) element. A single `title` element may occur within a chart, regardless of chart type. It is a simple string type element with no attributes and is used to specify title information for the chart.

The `legend` element is an empty child of the root (`chart`) element with no attributes. It may occur once or not at all in any chart type. It is used to specify whether the legend should be rendered.

### 5.3 Plot Area

The `plotarea` element is a child of the root (`chart`) element with no attributes. It is required to occur exactly once for bar and line charts and may not be present on pie charts. In principle, it may contain 1 to an unbounded number of axis elements, but our current 2-dimensional implementation requires it to have 2 axis elements for the bar and line charts.

The `axis` element, when present, has a required `variable` attribute. In the present implementation, the value of this attribute may be either "x1" or "x2". Both, bar and line charts require one of each. It must also contain `min`, `max`, and `step` attributes, except for the x1-axis on a bar chart which prohibits them.

### 5.4 Chart Data

The `dataset` element is a child of the root (`chart`) element. It must occur at least once and its maximum is unbounded. It represents a set of data on the chart and has an optional `label` attribute that is used to label that dataset in the legend. On a bar or line chart, it contains one or more `datapoint` children. On a pie chart, it has a `value` attribute but no children.

The `datapoint` element, present on bar and line charts, is an empty child of the `dataset` element. It represents a single point of data in 2-dimensional space and contains `x1` and `x2` attributes for describing that point of data.

## 6. Converting Spreadsheets to XML Data

Microsoft's Office Excel 2003 has the capability of exporting spreadsheet data into an XML data file. In particular, it gives users the ability to link an existing XML Schema file to the data in the spreadsheet. This way, users have control over the XML element names.

We developed an XML Schema which can be used for those purposes. Due to the limitations of the way XML Schemas can be mapped to data in an Excel spreadsheet, we are currently reworking the format of our XML Schema. We expect to have the revised software in place by the time of the conference.

## 7. Rendering of Charts

In this section, we provide an explanation of how chart images are rendered from chart XML documents.

The specifics of how `dataset` and `plotarea` elements are rendered varies depending on the `type` attribute of the `chart` element. However, all types of charts handle title and legend information in the same manner. First, if a `title` element exists, the text it contains is placed at the top of the chart, centered horizontally. Next, a color is assigned to each dataset. Then, if a `legend` element exists, a legend box is constructed on the right side of the chart. The composition of this box is as follows. Datasets are placed on separate rows, from top to bottom, in the order they are found in the file. Each row consists of a box of the same color as the dataset and the value of the dataset's `label` attribute, if any. Finally, the remaining area is used to construct the chart proper.

### 7.1 Bar Charts

The information contained in the `plotarea` element is used to construct the horizontal (`x1`) axis and vertical (`x2`) axis. On the vertical axis, the `min` attribute is used to denote the first value at the origin and grid lines appear at intervals specified by the `step` attribute until the value of the `max` attribute is reached at the top of the chart. The horizontal axis does not have `min`, `max`, or `step` attributes and its demarcation is determined by the datasets. For both axes, the text within the respective axis element is used as a label.

Each dataset element represents a different series of bars. The horizontal axis is divided into sections for each distinct `x1` attribute value found in a `datapoint` element of any dataset. This `x1` value appears centered below each section to label it. They are ordered as found in the document, starting with the first dataset and proceeding through each successive dataset.

The `x2` attribute value of each `datapoint` is used to construct a box for that dataset in the section

corresponding to its `x1` value. The box begins at the horizontal axis and goes up until it has reached the level on the vertical axis corresponding to its `x2` value. These boxes (one for each dataset) occur in the same left to right order in each segment. If a dataset does not have a `datapoint` with a given `x1` value, or if the `x2` value of that `datapoint` is outside the bounds of the vertical axis, it is considered invalid and not drawn.

### 7.2 Line Charts

The information contained in the `plotarea` element is used to construct the horizontal (`x1`) axis and vertical (`x2`) axis. On each axis, the `min` attribute is used to denote the first value at the origin and grid lines appear at intervals specified by the `step` attribute until the value of the `max` attribute is reached at the end of the axis. Additionally, the text within the respective axis element is used as a label.

Each dataset element represents a line on the chart and each `datapoint` element in that dataset represents a point on the line. The points are drawn at the position designated by their `x1` and `x2` attribute values and connected in the order they appear within the dataset. There is no restriction that datasets must use the same values for either the `x1` or `x2` attributes. Additionally, the values of either attribute may occur in any order and may be repeated within a dataset.

### 7.3 Pie Charts

Pie charts contain neither axes nor a `plotarea` element. Hence, there is no need to render a grid. Each dataset element represents a slice of the pie. Instead of containing `datapoint` elements, each dataset has a `value` attribute which is the value to assign to that slice. The total value of the pie is calculated by a summation of the `value` attributes of each dataset. From this, the percentage of the pie corresponding to each dataset is calculated and the slices are drawn with percentage labels.

## 8. Diagrammatic Search Engine

In order for a search engine to return meaningful results, we designed a special-purpose search engine which is tailor made for our XML format. See [9] for more information on special-purpose XML search engines.

To get a sense of how to interpret the data of charts in our system, we will rewrite the file from figure 3. First, we substitute in the contents of the `title` element and the value of the `label` attribute from its surrounding dataset element. We then substitute the `x1` and `x2` attributes with the contents of their corresponding axis elements. The resulting data is presented in figure 4.

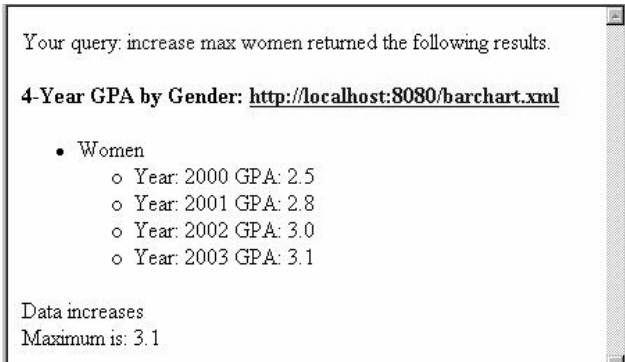
```

<4-Year GPA by Gender Men "Year = 2000" "GPA = 2.6"/>
<4-Year GPA by Gender Men "Year = 2001" "GPA = 2.8"/>
<4-Year GPA by Gender Men "Year = 2002" "GPA = 2.9"/>
<4-Year GPA by Gender Men "Year = 2003" "GPA = 3.0"/>
<4-Year GPA by Gender Women "Year = 2000" "GPA = 2.5"/>
<4-Year GPA by Gender Women "Year = 2001" "GPA = 2.8"/>
<4-Year GPA by Gender Women "Year = 2002" "GPA = 3.0"/>
<4-Year GPA by Gender Women "Year = 2003" "GPA = 3.1"/>

```

**Figure 4. A modified way of presenting the data of the example from figure 2.**

Each modified datapoint element now contains all information pertaining to it. At this point, we can search all datapoints and display each that satisfies a given query. Suppose that the user enters the keyword “women”. The results of this query are displayed in figure 5. In it, we return all datapoints, as modified above, which contain this keyword. The results are presented in a nested fashion, so as to avoid repetition of information. Furthermore, a link to the source XML document is presented. When clicked, this link will bring up the rendered image.



**Figure 5. The results of a query containing the keyword “women”.**

Consider table 1, which contains several sample queries and their expected results. Suppose that the user enters the keywords “men“ and “2002“ as presented in query (i). An intuitive interpretation is that the user is interested in the GPA that the men attained in 2002. Hence, we believe that the user considers the keywords of the query as a conjunct. While this violates the interpretation given to such a query by regular search engines, the document search can be viewed as an iterative filtering process on the set of all modified datapoints. Each disjunction represents a filter where only datapoints matching at least one term are kept. This means that the user has to type in the keyword “or” explicitly in order to obtain the GPA of the men over the years as well as the GPA of the men and women during the year 2002. The results to be returned

then should be the information that the GPA was 2.9, together with the contextual information as displayed in the modified datapoint.

If the user enters just “men“ as query, then the information displayed in query (iii) should be returned. In general, our search engine returns each datapoint, as modified in figure 4, which contains all of the keywords contained in the processed query. Our search engine searches the structure of the XML document and never modifies the XML document in the way outlined above.

Query	Search Terms	Search Results
i	men 2002	GPA = 2.9
ii	women 3.1	Year = 2003
iii	Men	Year = 2000, GPA = 2.6 Year = 2001, GPA = 2.8 Year = 2002, GPA = 2.9 Year = 2003, GPA = 3.0
iv	2000	Men, GPA = 2.6 Women, GPA = 2.5

**Table 1: Search queries and their results.**

In the section on the “Rendering of Charts,“ we explained what we consider invalid data. Just as our software does not display data that is considered invalid, the search engine does not return data that is considered invalid.

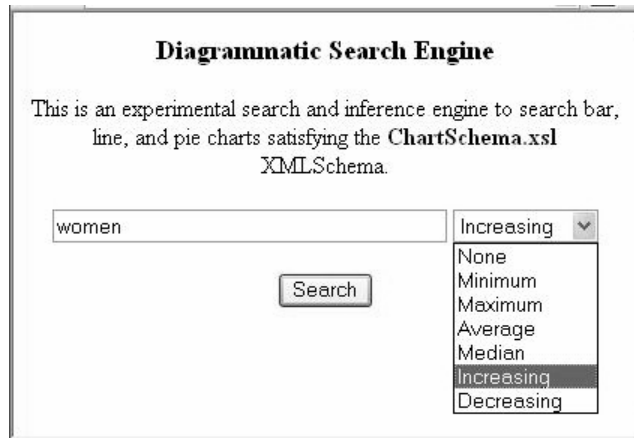
In the section on “Chart XML Syntax“, some elements and attributes were presented as optional. An example is the title element which may be non-existent, contain blank text, or, even when not empty, can be inconsistent or incomplete. For example, the information given in the title element of figure 2 (“4-Year GPA by Gender,“) does not specify the institution or department at which the GPAs were achieved. There are many other possibilities to store inconsistent data, involving the axis labels, the values of the x1 and x2 gradations and the data itself. In general, the editors of the data determine its quality. As with all information accessible on the web, the user still has to ascertain its veracity.

## 9. Diagrammatic Inference Engine

A search engine enables a user to find information that is explicitly contained in a document. An inference engine enables a user to also search for information that is implicitly contained in the document. For example, the highest GPA achieved by women is 3.1. Furthermore, the GPA went up each year. Those two results are indicated in the search displayed in figure 5. They are two pieces of information that people looking at bar-charts are very

interested in. We implemented those two predicates, as well as predicates for minimum, average, median, and decreasing. The user selects the predicates through a drop-down list, as shown in figure 6.

The inference done in our system is quite limited. We did not develop a proof system with which to give proofs that one would find in a theorem prover. Instead, predicates are implemented through methods. Nevertheless, we feel that the kind of inference is quite appropriate for the chosen domain as it implements the kind of inference people would do with our kinds of charts.



**Figure 6. Interface to inference engine.**

We are currently implementing predicates which enable us to compare data values across datasets. For example, we can read off the diagram that in 2003, women had a better GPA than men.

## 10. Conclusions

Our system of storing chart information in XML files on the web demonstrates a general process by which diagrammatic information can be stored on a web server in sentential format without losing the preferred visual presentation of the information to the user. The primary benefit of this process is that what the user perceives as an image is in fact a text file that can be indexed by a search engine spider. The choice of an XML format reaps the additional benefit of having structured metadata, as demonstrated in the design of our diagrammatic search and inference engine.

We feel that the system presented here can be adapted to other diagrammatic systems, such as wiring diagrams and flowcharts. In general, any diagrammatic system for which a syntax can be defined has sufficient structure such the information contained in it can be represented in XML format, which in turn makes it a good candidate for our process.

Future work would investigate diagrammatic systems as suggested above. It would also attempt to integrate this kind of system into the machinery of the semantic web.

## References:

- [1] <http://www.cs.rose-hulman.edu/~wollowski/xml/dept/chart.xsd>
- [2] G. Allwein & J. Barwise (eds.), *Logical reasoning with diagrams* (Oxford, England: Oxford University Press, 1996).
- [3] J. Barwise & J. Etchemendy, *Hyperproof* (Stanford, CA: Center for the Study of Language and Information, 1994).
- [4] J. Barwise & J. Etchemendy, Visual information and valid reasoning, in: Zimmerman & Cunningham (eds.), *Visualization in mathematics* (Washington, D.C.: Mathematical Association of America, 1990).
- [5] J. Glasgow, N.H. Narayanan & B. Chandrasekaran, B. (eds.), *Diagrammatic reasoning* (Menlo Park, CA: AAAI/MIT Press, 1995).
- [6] E. Hammer, *Logic and visual information* (Stanford, CA: CSLI Press, 1995).
- [7] C. Nastar, Finding the right picture in the corbis collection, Presentation at the 8th Search Engine Meeting, Boston, MA, 2003.
- [8] M. Wollowski & E. Hammer, Heterogeneous systems for modeling dynamic worlds, in: Pareschi & Fronhoefer (eds.), *Dynamic worlds – from the frame problem to knowledge management* (Dordrecht, The Netherlands: Kluwer Academic Publishers, 1999).
- [9] M. Wollowski & R. Signorelli, Towards a general-purpose XML search engine, Presentation at the 8th Search Engine Meeting, Boston, MA, 2003