Applying Formal Concept Analysis to Semantic File Systems Leveraging Wordnet

Ben Martin

Information Technology and Electrical Engineering
The University of Queensland
St. Lucia QLD 4072, Australia
monkeyiq@users.sourceforge.net

Peter Eklund

School of Economics and Information Systems
The University of Wollongong
Northfields Avenue, Wollongong, NSW 2522, Australia
peklund@uow.edu.au

Abstract *Formal Concept Analysis can be used to obtain both a natural clustering of documents along with a partial ordering over those clusters. The application of Formal Concept Analysis requires input to be in the form of a binary relation between two sets. This paper investigates how a semantic filesystem can be used to generate such binary relations. The manner in which the binary relation is generated impacts how useful the result of Formal Concept Analysis will be for navigating one's filesystem.*

Keywords Document Databases, Document Management

1 Background

Semantic File Systems convey the idea that a document can be found in variety of ways according to its content and the search requirements of the user.

A Semantic File System (SFS) unifies data sources through an extended File System interface. Two of the core features of an SFS are the extraction of metadata from files and the ability to create virtual directories showing filesystem objects which satisfy a query defined on the extracted metadata [10]. For an SFS, metadata describes the content of a filesystem object, for example, the width and height of an image. Metadata for files is stored (or derived) for each file and presented though an Extended Attribute (EA) interface [12, 11].

We focus on a particular opensource SFS implementation: libferris [12, 1]. Motivation for the use of Formal Concept Analysis on filesystems includes the ability for the system to handle over specified queries, the provision of an ordered grouping on query results and the ability to switch between query and navigation [12, 7].

Proceedings of the 10th Australasian Document Computing Symposium, Sydney, Australia, December 12, 2005.

Formal Concept Analysis [9] takes as input a binary relation between two sets and generates as output a set of "formal concepts" and an ordering relation over them. The formal concepts are a binary set of maximal clusters based on objects (files) and file attributes. An order relation is induced over the formal concepts and is referred to as a concept lattice. The input binary relation is referred to as a formal context. A formal concept (or simply concept), can be thought of as the largest connection between two sets which contains a specific element of one of the sets. Typically the two sets which the binary relation is held on are referred to as the Objects G and Attributes M. Thus for each object $g \in G$ one could consider a concept $(A \subseteq G, B \subseteq M)$ to be generated. It is natural for many objects to generate the same formal concept and hence the technique is a form of unsupervised machine learning.

It is natural for one to consider the files and directories of a Semantic File System as forming the object set for Formal Concept Analysis. In this way one might wish to use the metadata for his/her files to form the attribute set for Formal Concept Analysis. If one has some binary metadata about a file, for example is-character-device, then its presence can be taken to directly imply a connection in the input binary relation $I \subseteq G \times M$ for Formal Concept Analysis.

The requirement for input to Formal Concept Analysis to be in the form of a binary relation presents challenges when applying it to an SFS in general. This is due to the fact that the metadata attached to the files in an SFS are rarely simple binary values. Also some metadata which at first appears to be binary may have additional structure which should be taken into consideration. For example, the libferris Semantic File System has the notion of emblems. An emblem allows the user to categorize their files either explicitly or implicitly [11]. This may at first appear to be a simple binary attachment where for a specific emblem a file either has that emblem associated or it does not. However the em-

blems in libferris themselves form a partially ordered set and as such the association of an emblem x also conveys information about a files' association with all the parent emblems of x in this partial order.

Various solutions have been proposed in the Formal Concept Analysis community to allow its application on non binary input. The input to this process is called a many-valued context (G_y, M_y, W_y, I_y) and the process involves taking values, W_y, the many-valued input for each attribute M_y has, into consideration to generate binary attributes as output. These solutions include conceptual scaling [9] and logical scaling [14].

Some standard scaling techniques include: nominal, ordinal and interordinal. A nominal scale for an attribute M_y generates a new attribute in the output for each value of W_y which M_y takes in the input. If an object $g \in G_y$ has value $w \in W_y$ for attribute $m \in M_y$ then it will have attribute M_{my} in the output. An ordinal scale takes an attribute M_y which has a naturally ordered set of values W_y and divides the input value range into many linear intervals to form output attributes. An interordinal scale combines two ordinal scales, one using $\leq$ the other $\geq$ on its ordinal range.

2 Introduction

The libferris Semantic File System includes extensive indexing support for the storage of EA for files and the application of Formal Concept Analysis to this index. This paper explores how the many-valued data that a Semantic File System contains [3] can be transformed into a binary relation suitable for the application of Formal Concept Analysis.

A simplified overview of the process of applying Formal Concept Analysis to the Semantic File System is now described. Firstly, the filesystem is indexed by libferris for fast retrieval. We shall refer to this index as an "findex" to separate it from the other uses of the word index. Various clients, specifically designed for Formal Concept Analysis, are then applied to the findex to generate a concept lattice. The concept lattice is itself stored as part of the findex and to allow for subsequent reexamination. A concept lattice can be represented by a specialized form of Hasse diagram – called a line diagram – though which its partial order can also be exposed as a Semantic File System by libferris.

It has been found that in many cases some pre-analysis for a Semantic File System is needed in order to best expose the Semantic File System without generating a cluttered output.

3 Application

The standard scale types of Formal Concept Analysis: nominal, ordinal and interordinal are supported with extensions through three client applications described in Section 3.3. Using a file's URL as metadata to generate formal attributes in various ways is supported as described in Section 3.5. Together with the applications described below there is a method of restricting which documents from an findex are potentially useful. This allows areas of the findex to be negated from query results *en masse*. For example, one might consider only documents under `/usr/local` to be of interest for a particular analysis and so restrict all results to also satisfy this condition.

To demonstrate, an findex was created on a Fedora Core 4 Linux machine using libferris 1.1.54 of 201,759 files in `/usr/share/`. All libferris clients for creating input for Formal Concept Analysis use either the `gf-create-fca-scale` or `ferris-create-fca-scale` prefix in their command names. The clients are subsequently referred to without prefix.

Section 3.1 discusses application to nominal binary data, section 3.2 applies to geospatial metadata for files, section 3.1 discusses application to numeric domains. The application to NSA SELinux [13] follows in section 3.4 followed by the use of Wordnet [8] to improve the structure of concept lattices created from file URLs in section 3.5.

3.1 Scaling nominal orders

In addition to the standard treatment of nominal scaling [9, 5], two new capabilities for handling ordering over nominal attribute creation have been added.

The first ordering capability is to handle MIME type like strings such as `image/png` by allowing the values of the distribution to be split into distinct parts and have common parent attributes created. Following the MIME example, a common parent for all image files would be the new `mime.image` attribute. Using this form of nominal scaling an ordering can be introduced based on the values of the distribution which will help to generate a taller, narrower concept lattice [5].

The second ordering capability is to take advantage of the ordering over the emblems when performing nominal scaling via an emblem. The ordering on the emblems is a partial order allowing reasonable flexibility in how one designs emblem categories. The ability to handle entire downsets relative to the emblem ordering when generating a formal context allows one to see their lattice including the influence of their emblem ordering. Given an ordered set P and $Q \subseteq P$ then Q is a downset iff $x \in Q, y \in P$ and $y \leq x$ then $y \in Q$.

3.2 Scaling Geospatial information

Geospatial metadata is exposed through two cooperating interfaces in libferris. These are the latitude and longitude EA and the emblem system. Geospatial emblems are those which are a child of the `libferris-geospatial` emblem in the emblem partial ordering. Interaction with the filesystem for tagging and retrieval is usually simpler when emblems with city or place names are used instead of world coordinates.

As the emblem system is employed the scaling methods of Section 3.1 are also applicable for geospatial values. A major advantage of the emblem geotagging system coexisting with the latitude and longitude system is the ability to handle geospatial regions. The emblem partial order can be used to define geospatial regions that expand from point locations to physically containing regions. For example, the Sydney Opera House might be given a specific emblem with Sydney as its parent. The Sydney emblem may have Australia which itself has `libferris-geospatial` as its parent. If less specific emblems in the partial order define containing geospatial regions then the downset handling in Section 3.1 can be used to introduce geospatial refinements into the concept lattice.

Without the ability to represent geospatial regions through the emblem partial ordering in this way one would have to explicitly define the boundaries using bounding box constraints on the latitude and longitude for the region. Consider the difficulty in defining the boundary of the city of Sydney using only equality constraints on latitude and longitude.

3.3 Scaling numeric ranges

Three commands exist for creating formal contexts from numeric data in libferris. These are `numeric-ordinal`, `numeric` and `gf-numeric`.

The `numeric` client can create many binary attributes each exposing a numeric interval of the input data as is standard for Formal Concept Analysis. For example, consider scaling a numeric range of $\{1, 2, ..., 20\}$ into four attributes at an even interval of 5 using $\leq$ as is standard in Formal Concept Analysis. This will produce four attributes with higher successive values having less matches due to the $\leq$ relation.

The standard application of ordinal scaling preserves both linearity *and* density for the input [6]. Due to the intermixing of other attributes in a concept lattice it is hard to take advantage of the preservation of density information. When one places these four attributes alongside another ten attributes and generates a concept lattice the relation between ≤ 10 and ≤ 15 is not so immediately obvious from the concept lattice. One can see the order of the two attributes but the density information is lost due to the introduction of the other attributes.

For some value distributions using a linear interval for the range is ineffective. For example, if one is to scale the values for the file size metadata then the distribution of values may be very ineffectively presented when split into a low number of linear intervals. To overcome this issue the data-driven-scale option was added to allow numeric distributions to be scaled taking the value distribution into account. This option will make an output which will have smaller intervals where many files have similar values and larger intervals where few files match the interval.

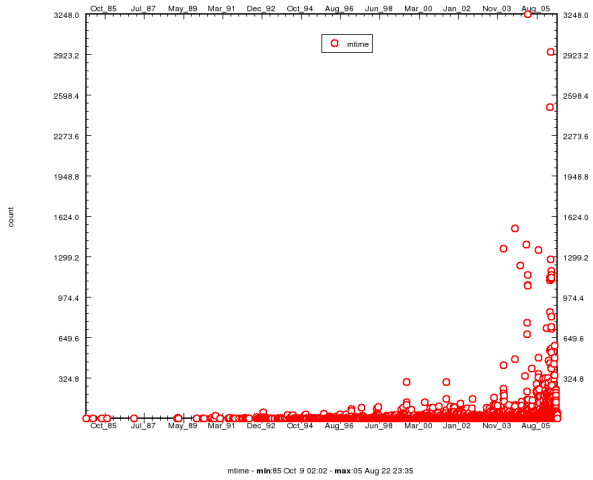

Figure 1: Plot of the modification time of 201,759 files from /usr/share/. Horizontal axis shows time from October 1985 to preent day with almost 2 years between graduations. Vertical axis ranges from 0 to 3248 with around 235 files seperating each graduation.

The std-deviations option has been added to handle simpler value distributions by allowing output to be generated based on the mean and variance of the input distribution.

One can manually select where intervals begin and end using the GTK+ `gf-create` client. Figures 1-4 were generated with `gf-create`. For value distributions which neither fit direct data frequency nor standard deviation models the ability to explicitly choose where intervals begin and end on a value frequency plot can generate a small number of meaningful attributes. For this purpose an interactive graphical client was create allowing intervals to be selected with the mouse.

The plot for the modification time (mtime) of the findex is shown in Figure 1 and the metadata status change time (ctime) in Figure 2. One can see that although modification was more frequent in recent times the ctime plot has explicit natural clusters of values. Such clusters are likely due to large scale system administration activities such as distribution release upgrades. Using the graph a small number of meaningful attributes can be created based on major system update activities.

An EA was added to the libferris system to support the ability for many versions of a file's metadata to exist simultaneously in an findex [2]. This EA returns the current system time when it is read. As expected, the plot for this attribute gives valuable information about when indexing sessions were held as shown in Figure 3. Looking at Figure 3 one would be lead to create three formal attributes, one for each of the major groupings of matching files.

The width EA presents the width in pixels of a file. For many systems this EA must be handled explicitly

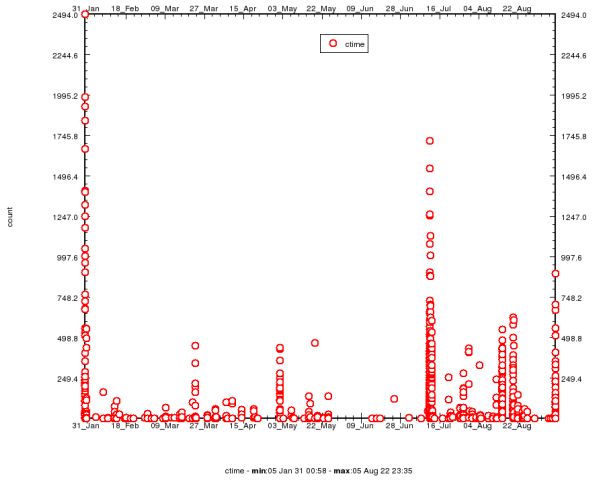

Figure 2: Plot of the ctime of 201,759 files from /usr/share/. The ctime for a file changes whenever any of its metadata (except atime from lstat) changes. Horizontal axis shows time from 31st January to 05 August 2005 with two and a half weeks between graduations. Vertical axis ranges from 0 to 2494 with around 250 files seperating each graduation.

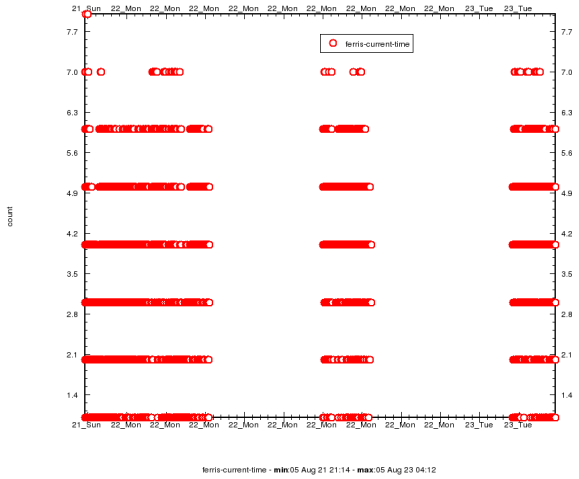

Figure 3: Plot of the ferris-current-time EA of 201,759 files from /usr/share/.

because a small number of extremely large images can easily distort simpler methods of splitting the value distribution. In this case two images stand out, sgvol.png from the kdemultimedia package is 7,140 pixels wide and sunclock_huge_earthmap.jpg from the sunclock_huge_earthmap package is 10,800 pixels wide. All other image files in the findex are below 3,500 pixels wide. The width plot is shown in Figure 4. One can also start from the megapixel count of images as more generalized overview of image size to generate

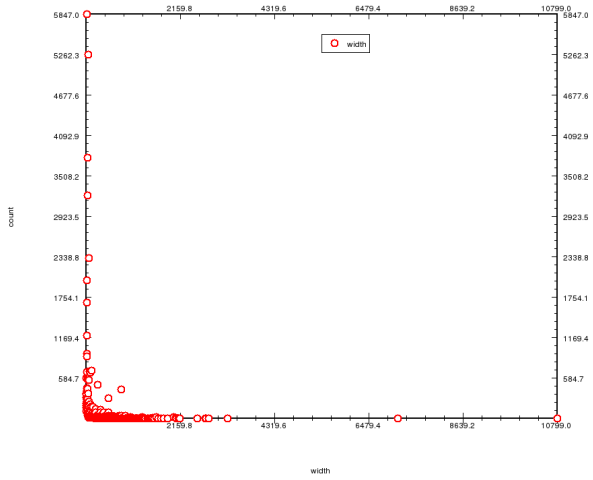

Figure 4: Plot of the the width of image files from /usr/share/.

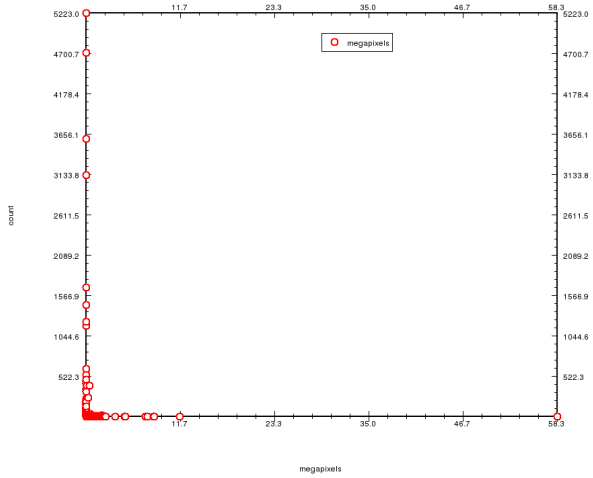

Figure 5: Fewer plot points but a similar overall trend to the width plot. Plot of the the megapixels of image files from /usr/share/.

formal attributes. As can be seen from Figure 5 there is a similar trend as to the width plot.

Two concept lattices were generated using the width EA and the modification time for the examples 201,759 files. Both scale the width and modification time metadata using 7 formal attributes for each. The first one shown in Figure 6 uses the standard linear ranges to generate the formal attributes. Shown in Figure 7 is the concept lattice that results when dividing the input ranges based on value density.

Because the formal attributes in Figure 7 are data driven there is much more interaction between concepts in the resulting concept lattice.

For some numeric EA such as: group-owner-number, user-owner-number the user may wish to

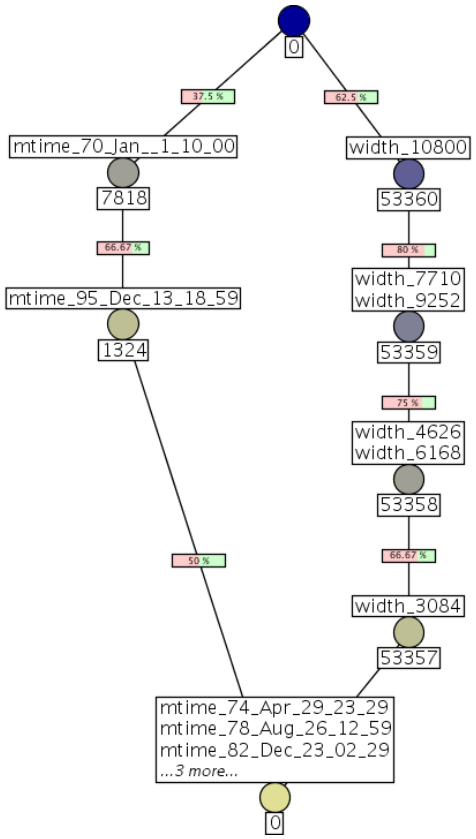

Figure 6: 7 formal attributes for each of mtime (modification time) and width using a standard linear range division. Concepts are represented as circles. Labels above a concept show the formal attribute which is introduced by that concept and labels below a concept show the number of filesystem objects which match that concept or one of its refinements. An introduced formal attribute is a formal attribute for which this concept is the highest one in the lattice with that attribute. Thus, where a concept has an introduced formal attribute all concepts reachable transitively downwards will also have this formal attribute.

explicitly specify the range for each formal attribute based on knowledge of the computer system. For example, on many Linux installations the numeric user and group identifiers above 500 are used for normal user accounts.

3.4 SELinux

Security Enhanced Linux (SELinux) [13] allows modern Linux installations to offer Mandatory Access Control (MAC) as well as the more familiar Discretionary Access Control (DAC). Under DAC, file access is granted or denied based on the user running an application. Assume that my user account as read and write access to my thesis and read access to my music collection. Under DAC a music player has the ability to overwrite my thesis just as xemacs can read my music files. With MAC programs can be allowed access only to the files that are required for them to operate. For example, using MAC I can disallow my media player access to any files relating to my thesis. It should be noted that my user account will still be the owner of my music files and thesis though the media player run by me will be disallowed access to some files owned by my user account.

SELinux information which is attached to files is comprised of three datum: the identity, role and type. The identity is a SELinux user account, the role is ignored for files and the type is the primary security attribute for making authorization decisions.

In a minimal installation one has an SELinux user_u account which is shared by all users in a similar category and a system_u for daemon usage. The example 201,759 files have three identities: root, user_u and system_u. Also there are nine types: etc_t, fonts_t, httpd_sys_content_t, lib_t, locale_t, man_t, shlib_t, snmpd_var_lib_t, and usr_t.

A very high level view of how access is granted or denied follows, for details see [13]. Each process also has an associated SELinux context. Access is granted or denied based on the SELinux context of the process and the file together with the operation requested to be performed. As such viewing only the SELinux context for files provides an incomplete picture of overall security policy.

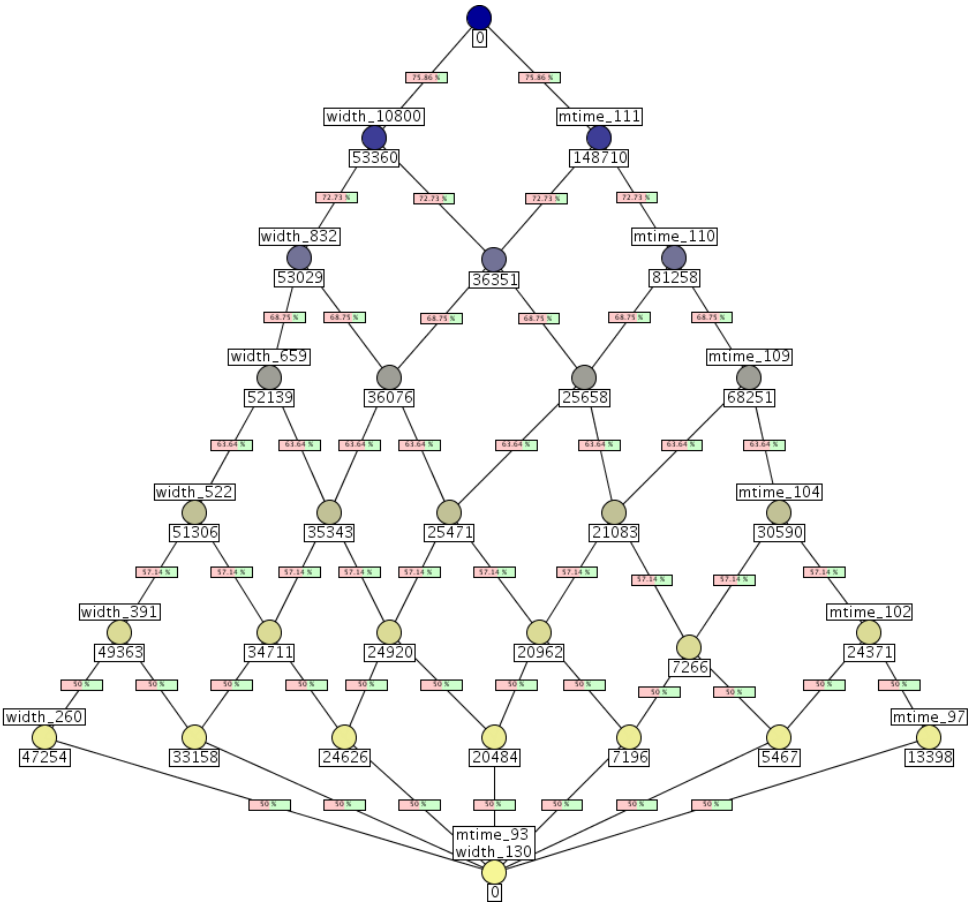

Figure 7: 7 formal attributes for each of mtime (modification time) and width. Formal attributes are generated based on the density of the input metadata.

Using the SELinux type and identity of the example 201,759 files the concept lattice shown in Figure 8 is generated. The concept 11 in the middle of the bottom row shows that user_u identity is only active for 3 fonts_t typed files. Many of the links to the lower concepts are caused by the root and system itentities being mutually exclusive while the system identitiy combines with every attribute that the root identity does.

3.5 Structuring with URLs

Often the URL for a file is comprised of metadata forming an ad-hoc hierarchy [4]. To put such metadata into the URL itself requires arbitrary decisions about the ordering of such metadata. For example, one must decide if they are to first classify a file by its conference name or conference year in the URL `.../adcs/2005/martin-eklund/...`.

The `scale-urls` client creates a formal context from the directory components in URLs. Additional processing can be performed to present a more attractive and useful concept lattice. For example, heuristics can be used to strip version information from directory names such as `java-1.5.0`.

Wordnet [8] is also employed to explicitly allow the generation of formal concepts for hypernyms of common directory names. Explicit hypernym concepts are generated as follows: each URL is divided into its directory name components with a number of the rightmost path components being dropped (normally just one, the filename), each directory name is then stripped of version information and added to the set D. Many such preprocessed directory names $d \in D$ are then candidates for use with Wordnet. If d can be found in Wordnet then its synonym set X is found and all the hypernyms for X are collected. When two or more d have the same synonym set X then the hypernyms for X are emitted into the formal context with a prefix "wn_" to denote that they have been mechanically added.

The semantic commonalities between directory names are made more explicit in the output concept lattice using the Wordnet hypernym associations. Another advantage is that because the "wn_" attributes effectively form the join of many existing attributes they are closer to the top of the concept lattice. If the concept lattice is being read in the usual way from top to bottom or is itself being navigated as a filesystem the placement towards the top of the lattice advantageous to have these wordnet attributes to assist in navigation to the desired concept.

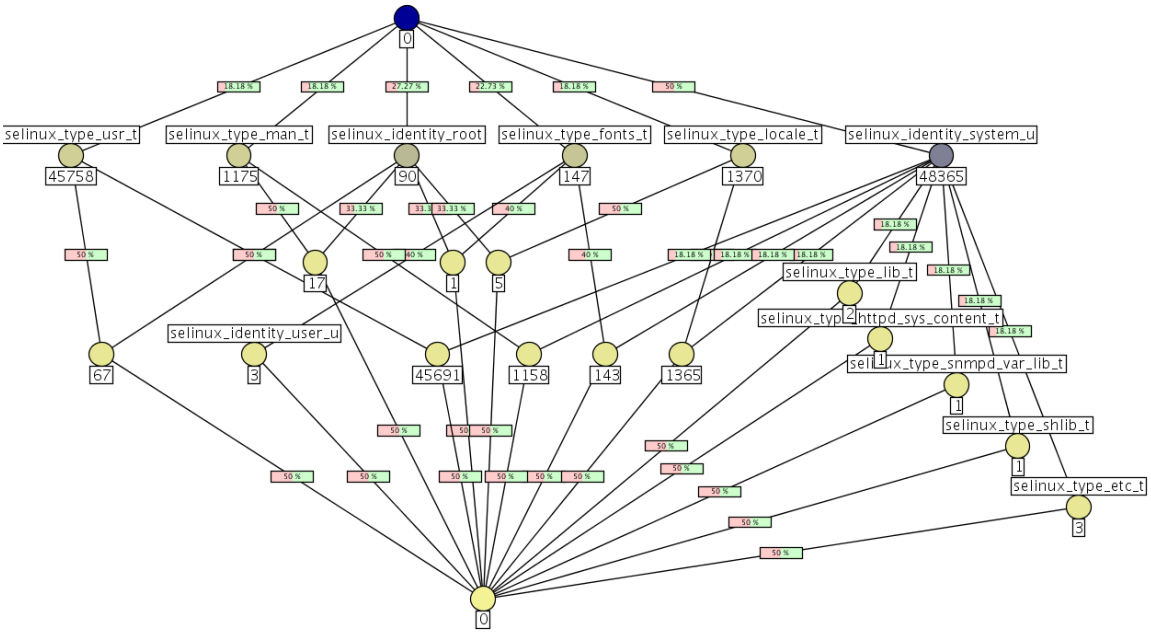

Figure 8: Concept lattice for SELinux type and identity of files in /usr/share/ on a Fedora Core 4 Linux installation. The Hasse diagram is arranged with three major sections; direct parents of the root are in a row across the top, refinements of selinux_identity_system_u are down the right side with combinations of the top row in the middle and left of the diagram.

For the example findex, dropping only the final directory component (i.e., filename) from each URL the wordnet scheme above generated 403 new "wn_" attributes. An example of one such attribute is wn_article which is equal to `f`eature $\vee$ paper.

Shown in Figure 9 is small example concept lattice where the two files of interest are both masked from the top concept by attributes which one does not immediately consider related to the child concept. Consider that both the concepts labeled with $a1$ and $a2$ may have many more child concepts than those shown. The introduction of a semantically more general attribute form wordnet in Figure 10 may allow the user greater ease in locating their desired concept.

4 Conclusion and Future Directions

The application of Formal Concept Analysis to a Semantic File System is not straightforward. One must choose what metadata to use to construct the input for Formal Concept Analysis: the formal attributes. This stage is critical to the generation of an interesting concept lattice.

This paper has shown some methods of applying Formal Concept Analysis to an Semantic File System. The generation of formal attributes using Wordnet can have a positive effect on the usability of a concept lattice when using URLs. This is very important as the file paths are normally the only metadata explicitly supplied by the computer user.

When considering ordinal metadata the density of files with particular values of metadata can be taken advantage of to generate fewer formal attributes for the same metadata. When metadata which is ordinal is to be transformed into formal attributes one can have fewer attributes if they use non linear ranges to break up the metadata. The density information that is retained by using linear ranges to break up ordinal metadata is not readily apparent in a concept lattice with many formal attributes from different metadata sources.

References

[1] libferris, http://witme.sourceforge.net/libferris.web/. Visited Oct 03.

[2] libferris in the linux journal, http://www.linuxjournal.com/article/7771. Visited Jan 05.

[3] A virtual filesystem on steroids: Mount anything, index and search it. To appear: Linux Kongress 2005 proceedings.

[4] Peter Becker and Richard Cole. Querying and analysing document collections with formal concept analysis. In *Australian Document Computing Symposium (ADCS03)*. University of Queensland, 2003.

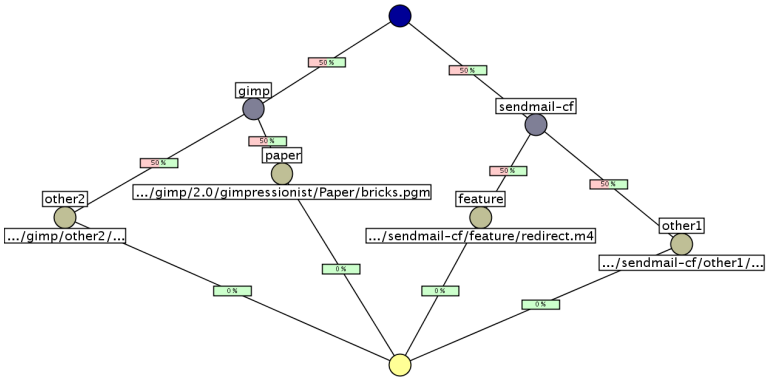

Figure 9: Example lattice with no wordnet augmentation.

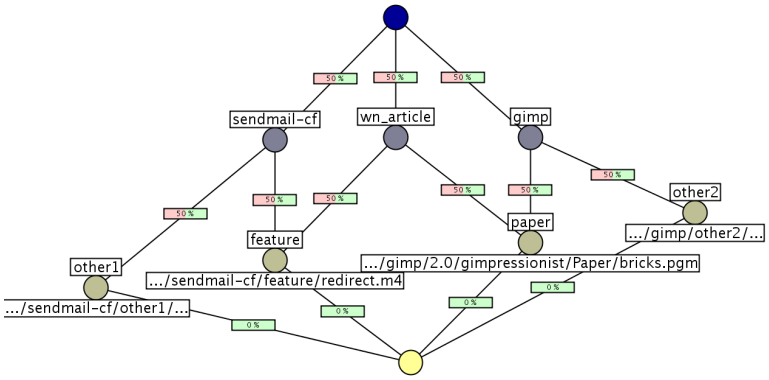

Figure 10: Example lattice using wordnet augmentation, notice how the wn_article concept is the common parent of both feature and paper and is also closer to the top of the lattice than either hyponym.

[5] Claudio Carpineto and Giovanni Romano. *Concept Data Analysis*. Wiley, England, 2004.

[6] Robert Colomb. *Information spaces : the architecture of cyberspace*. Springer, London, 2002.

[7] Sebastien Ferré and Olivier Ridoux. A file system based on concept analysis. In *Computational Logic*, pages 1033–1047, 2000.

[8] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. J. Miller. Introduction to wordnet: An on-line lexical database. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 112–119, 1990.

[9] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis — Mathematical Foundations*. Springer–Verlag, Berlin Heidelberg, 1999.

[10] David K. Gifford, Pierre Jouvelot, Mark A. Sheldon and James W. Jr O'Toole. Semantic file systems. In *Proceedings of 13th ACM Symposium on Operating Systems Principles*, ACM SIGOPS, pages 16–25, 1991.

[11] Ben Martin. File system wide file classification with agents. In *Australian Document Computing Symposium (ADCS03)*. University of Queensland, 2003.

[12] Ben Martin. Formal concept analysis and semantic file systems. In Peter W. Eklund (editor), *Concept Lattices, Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, Proceedings*, Volume 2961 of *Lecture Notes in Computer Science*, pages 88–95. Springer, 2004.

[13] Bill McCarty. *SELinux: NSA's Open Source Security Enhanced Linux*. O'Reilly & Associates, Sebastopol, California, 2004.

[14] Susanne Prediger. Logical scaling in formal concept analysis. In *International Conference on Conceptual Structures*, pages 332–341, 1997.