ConceptOntoFs: A Semantic File System for Inferno

Abhey Shah, *University Of York*
Leo Caves, University Of York
{as234,lsdc1}@york.ac.uk

Abstract

We present a novel filesystem that allows the automatic arrangement of the data contained within it into directories of similar entities. The example source dataset used is of micro array gene expression data and with this data set the Gene Ontology can be used to name the newly created directories.

The exempler dataset used here is Micro-Array Gene Expression experiments; almost every micro-array experiment generates lists of genes that are up-regulated or down-regulated under some environmental condition or at some time point. These lists of interesting genes are used to construct a hierarchical arrangement of folders that contain experiments that involved similar patterns of genetic behaviour.

Data mining across multiple experiments and integrating that with information from other sources such as the Gene Ontology to form new hypotheses is the key goal of this system.

1 Motivation

The ability to rapidly navigate, share and understand large bodies of structured and unstructured data is a challenge. In the post genomic era in biology, the volume of available data and complexity of data have mushroomed and have motivated the search for better methods to represent, structure, present and infer biological meaning from data.

File systems offer more then just storage and management of data, they can also offer an aid to increasing findability by self categorizing their contents. However conventional file hierarchies have files logically contained within only a single parent directory. Thus when navigating through a file system the user is constrained to follow only one path to a given file, (this ignores the hard and soft links of UNIX and the bind operator in Plan 9 and Inferno). For the vast majority of data a straightforward hierarchical file system is enough, however data in the real world doesn't always fit with this Platonic approach to classification[Mor05]. This scheme is overly restrictive to be applicable to all files and all collections of data. There are situations where support for richer semantic relationships would be beneficial.

Within Plan 9 this problem of resolving the representation of a set of files with multiple meta data arises within the jukebox audio player. The single parent structure of a file system is insufficient to capture all the ways of categorising music files. The solution offered is to store this meta data within a separate set of map files. The collection of map files represents multiple different faceted classifications of the underlying music files. The weakness of this system is that the map files need to be manually generated.

There is a well developed mathematical model, Formal Concept Analysis[GW99] (FCA) that can be applied to the structuring of data sets based on their attributes. Here we review some basic FCA theory and some novel approaches to semantic file systems based on it. Furthermore we introduce a simple tool, ConceptOntoFs, developed for Inferno, to infer semantic relationships between a set of files and represent these relationships as directories within a synthetic user level file system. Furthermore, as an aid to navigation for a specific domain, the tool can name each directory of this file system hierarchy by mapping the sets of attributes to an appropriate entry from an ontology. The development of ConceptOntoFs was partly motivated by the development of semantic file systems based on FCA on other computing platforms.[Mar04, FR00]

1.1 The Biological problem

The human genome contains about 30,000 genes that make the proteins, that run all our biological processes. Many of these processes require a whole host of proteins to act in concert. A useful analogy is to consider the genes as a set of components in a device like a radio[Laz02]. Recently with the human genome project we have identified different "components and their location" however we don't know the "circuit diagram" for this radio or even the principles governing it's behaviour. Without such an understanding, it is difficult to control or engineer biological systems e.g. to develop targeted drugs without side effects.

To discover the function of gene expression, micro-array experiments can be used[DPB$^+$96]. Almost every micro-array experiment generates lists of genes that are up-regulated or down-regulated under some environmental condition or at some time point.

An example data set used to demonstrate these ideas is that from multiple gene expression experiments taken from the L2L[NW05] Micro-array Analysis database. L2L is a collection of published micro-array data, in the form of a set of files, with each file describing a single experiment. Each file contains a list of genes of interest. From this the methodology of Formal Concept Analysis is used to construct a hierarchical arrangement of directories that contain experiments that involved similar genes. The Gene Ontology is used as a knowledge structure to make the system more usable and accessible.

2 The theory behind FCA and Ontologies

2.1 Basics of Formal Concept Analysis ordered structures

The following theoretical exposition is taken from [GW99].

Formal Context: A *formal context* is a triple $\mathbb{k} := (G, M, I)$ where G and M are two sets and I is a binary relation between G and M. The elements of G are objects and each has some attributes from the set M (the choice of the letters G and M is retained here to keep consistency with the original German exposition). The attributes that each object has is defined by the binary relation I. If $g \in G$ and $m \in M$ and $(g, m) \in I$ then g has attribute m. In practise a formal context takes the form of a binary table.

Formal Concept: For $A \subseteq G$, we define $A' := \{m \in M | \forall g \in A : (g,m) \in I\}$. Correspondingly we define $B' := \{g \in G | \forall m \in B : (g,m) \in I\}$. We can consider each prime operator as a function from power sets. So

- $f : \mathcal{P}(G) \to \mathcal{P}(M), f(A) = A' = \{m \in M | \forall g \in A : (g,m) \in I\}$
- $g : \mathcal{P}(M) \to \mathcal{P}(G), g(B) = B' = \{g \in G | \forall m \in B : (g,m) \in I\}$

The compound operators $g \circ f(A)$ and $f \circ g(B)$ are now closure operators over $\mathcal{P}(G)$and $\mathcal{P}(M)$. If we denote the compound operators by " then this means that Z⊆Z" and (Z")"=Z" for any Z in $\mathcal{P}(G)$ or Z in $\mathcal{P}(M)$. Now a *formal concept* is a pair (A, B) such that $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. The set A is then called the extent (it is a subset of the objects) and the set B the intent (it is a subset of the descriptions) of the formal concept (A, B). A concept can be considered to be a set of objects which all posses a set of attributes

The set of all formal concepts is thus a family of closed subsets under the " operator and a key result within FCA is that this is a lattice. That is for any two concepts there exists a unique greatest lower bound ($\vee$) and a lowest upper bound ($\wedge$). There is a canonical set of algorithms within FCA that can be used to create a lattice from a given context.

Concept Lattices: Each formal context forms a conceptual hierarchy, called a *concept lattice,* the hierarchical relation between concepts is formalised by

$$(A, B) \leq (C, D) :\Leftrightarrow A \subseteq C (\Leftrightarrow B \supseteq D)$$

Attribute hierarchies: A useful partial order ($\leq$) on the set M can be defined by this compatibility condition:

$$\forall g \in G, m, n \in M : (g,m) \in I, m \leq n \Rightarrow (g,n) \in I$$

This can be read as saying that if this relation holds $m \leq n$ then anything that has the attribute m also has the attribute n. This partial order is a powerful abstraction upon which to build algorithms to classify data sets and infer functional dependencies.

2.2 Ontology

FCA facilitates the collection of objects via shared attributes ("concepts") but the problem remains of how to describe or name the concepts compactly. The naive approach of using attribute names directly leads in general to verbose names that do not really convey the character of the concept. So how can we name each concept? Every concept consists of a set of objects which have the same set of attributes. How do we make use of this?

For genes which are the data set used throughout this paper there is a well developed knowledge structure that can help, the Gene Ontology (GO)[ABB$^+$00]. The GO is a controlled set of vocabularies that describe gene products, biological processes and molecular function. It naturally forms a hierarchical, taxonomic categorization of biological terms. For example given the terms 'DNA replication' and 'DNA repair' it can tell us they are both subclasses of 'DNA metabolism'. Since there exists mappings from genes to the GO terms it is thus possible to utilise the GO in helping to describe the structure of the data from L2L.

3 Review of semantic file systems based on FCA

The notion of a *semantic file system* was introduced by Jouvelot in [DJ91]. It is defined as a virtual arrangement of directories that allows flexible associative access to the contents of files by automatically extracting attributes from files with transducers. Transducers are user defined programs that can derive attributes from a file. These attributes are then used to build corresponding indexes. This generalises the model commonly used in file-systems where only a fixed set of attributes (time of modification, user permissions, size of file etc) are available for each file. In semantic file systems, the attributes are dynamic entities that can be created at will. Semantic file systems are thus more generic examples of file systems.

The motivation to base a semantic file system on FCA comes from the ability for the system to handle under-specified queries, the provision of an ordered grouping on query results and the ability to switch between query and navigation. Other semantic file-systems like SFS[DJ91], HAC[GM99] and BeFS[Gia93] for example do not permit the navigation of arbitrary queries.

It is unsurprising for the Plan 9 community that the developers of both libferris[Mar06] and LISFS have realised the power of having a consistent file interface to dynamic structures.

3.1 libferris

The libferris[Mar04] file system transforms extended attributes about a file into binary attributes for which a canonical algorithm from FCA is used to find concepts which are represented as directories. Moving files between the virtual directories can change the attributes that define a file. To take advantage of libferris requires applications that are aware of it. Hence it includes rewritten versions of basic UNIX programs like ls, cp, mv, rm, mkdir, cat, find, touch and IO redirection. It can currently only be used in GNU/Linux.

3.2 LISFS

The Logical File [FR00]system is based on Boolean logic together with Logical Concept Analysis (a development of FCA where the result of logical formulae are used in place of binary attributes). Paths are formulae, directories represent queries and determine set of files and parts of file whose description satisfies the formulae. The root directory represents the formula "true", and sub-directories of a directory are determined by the most general properties refining the query. The *ls* command gives hints about further queries that can be made. The file system is thus generated as a user explores it. Each directory is created by users' queries, the file system can thus be used for both navigation and querying. The names of the directories are taken from the queries used to form them.

So an example interaction taken from [PSR06]:

```
[1] cd /lfs/music/year:[1980..1990]
[2] cd !genre:Comedy
[3] cd time:<7min
[4] cd .ext
[5] playmp3 *
...
[6] cd /lfs/music/genre:Disco/
```

```
[7] ls
artist:BeeGees/ artist:DonnaSummer/
[...]
year:1976/ year:1977/
[...]
```

Command 1 selects music files from the '80s, command 2 excludes comedy files from the selected files and command 3 selects only short songs. For each one of the commands 1, 2 and 3 a different transducer (or attribute extractor) is used. From then, the user can switch to the extension of its query, which actually contains the files that match the query that has been built up (command 4) and listen to it (command 5). Command 6 selects disco music and command 7 then asks for possible navigation links. It can be seen that they can be used as possible queries.

Some of what LISFS accomplishes could be done with appropriate use of the shell, however it is arguably simpler. In addition all this navigate and query functionality can easily be used within applications (and also keeps those applications simple). Unlike libferris it uses standard UNIX utilities (all the intelligence is kept in the file system) but can only be run in GNU/Linux.

4 ConceptOntoFs

ConceptOntoFs is geared around using small programs and existing tools and utilities wherever possible.

4.1 Implementation Details

The basic process of directory generation requires 3 programs (L2l2FCA, Conceptderive and FCA2fs), and in addition some resources necessary to provide usage of the GO, all are described below:

L2l2FCA The first program is a transducer that is used to generate a table of attributes, this is a table representation of the formal context described above in 2.1. The format of this context table is designed to be human readable by having attributes as columns and objects as rows; and an x where every object has a attribute and a 0 otherwise. Large context tables can be made simpler by either removing some attributes or by ORing some of them together.

Conceptderive This takes a context table in the format described above (i.e. taken from L2L2FCA) and applies the standard FCA next closure algorithm[1] to produce output listing the concepts (this is the objects and the attributes that belong to that concept) for convenience it also lists the parents and children for each concept.

FCA2fs This takes the output from Conceptderive to generate and serve a synthetic file system (an Inferno server). Directories are formed from the concepts, however the concepts form a *lattice* which means that concepts can and do have multiple parents. To implement this poly-hierarchy every path through the lattice is

[1]It can be found in Section 2 of [GW99].

traversed and for each path a directory is created. This means the order of names that make up a path from the root of the mounted file system is irrelevant and there are multiple equivalent directories, each of which point to the same contents and equivalent child directories. These multiple directories need to be identified as equivalent, later we describe the mechanism behind how Inferno servers operate to show how this is accomplished.

FCA2fs alters the namespace by mounting itself over the existing directory it is run in, however it spawns a thread that runs within the old namespace.

GO Utilities The Gene Ontology has multiple uses in this set of tools. The look up of GO terms from gene names is done via a process modelled on how the DNS (Domain Name System) service is implemented in Inferno, that is it serves one file (/info/g2go) that applications can write to with their request and read back answers. The obvious benefit is that this functionality is easily shareable amongst applications. Actually working with GO terms and their structure (e.g. finding hypernyms, calculating significance scores) is done via another file (/info/go).

4.2 Handling poly-hierarchy

The architecture for file systems within Inferno is arranged so that all system calls on files on are translated into requests and responses that are handled by a server, where a server can be a user program such as FCA2fs. Each instance of a file or a directory within the synthetic file system has a unique QID path number allocated to it. This number is used by the file server to identify requests for that file.

When FCA2fs turns the lattice structure into a tree, the concepts are enumerated and each directory (and it's contents) corresponding to a particular concept has the concept number encoded as part of the top 2 bytes of the 8 byte QID path number.

Thus while each instance of a file and a directory has a unique QID, equivalent files and directories can be identified. For example Qidc is a gate for the fs[Inf] system in Inferno that takes an integer, it is open if the entry has a QID that contains the given concept number, so

```
%fs select {and {mode +d} {qidc 5} } $conceptserver
```

would return all directories that are part of concept 5.

The system serves a read only parents file within each directory that can be read to obtain information about the parental concepts of that directory. With these elements, handling going up (getting dot-dot right) in this lattice file system can be accomplished with a convoluted and somewhat ugly solution described in the next section.

4.3 File System Traversal and Naming

The above three commands are typically run altogether with a pipe connecting them, and are packaged for convenience into a shell script, *crd* (CReate Directory). As mentioned above, FCA2fs alters the namespace by mounting itself over the existing directory it is run in, while still running a thread with the old namespace. File requests and responses are connected between the two running threads such that all reads and writes to files get passed through directly to the original files. The net effect is that a flat directory gains structure. Crd can be run again with alternate parameters and by spawning new namespaces for each new query this system produces a similar style of querying and navigation as LISFS, however in this system querying and navigation are explicitly two different commands.

Each entry in the file system has a definite path (that which was used to get to that entry), this is used when "cd .. " is executed. However while each directory has only one parent as far as ".. " is concerned there are multiple concepts and parental directories that are actually available. In order to get to a more general cd_dotdot command the following procedure is used:

1. The numbers of the parental concept can be found by reading the parents file.
2. These entries can then be passed to a shell command which uses qidc in a similar manner as above to list all directories of a concept. They are all listed with their full path from the root of the synthesised file system.
3. From this list an entry can be chosen for navigation.

This system is not perfect because there are some situations where ".." should be used and others where cd_dotdot is neccessary and arguably the user shouldn't be required to know there is a difference.

The concept directories can be named most naively after the attributes that they are defined by, since a concept is composed of a set of objects and the attributes they have in common. However this system using the L2L database can use an ontology too, when FCA2fs is creating the file system it does this: every concept is composed of a set of attributes that is transformed (via /info/g2go) into a set of GO terms. A heuristic is applied to choose the most appropriate GO term (based on scoring elements of the GO for semantic significance and centrality, via /info/go) and this is then used.

5 Comments

Several features of Inferno helped in this project. Most notably the sheer ease at which file systems can be built. The per process namespaces make querying and navigation a simpler to implement feature.

Whilst developing this it was useful, since it takes a relatively long time to parse and set up the GO database, to use the dynamic module loading to load new code into the program whilst it still retained the GO database in memory. This considerably shortened the development process by encouraging rapid prototyping.

There are some obvious issues with this implementation of ConceptOntoFs - it is a prototype is our defence, here we list some of the more glaring and apparent ones:

- Synthesised file-system structure is read only. Doesn't allow files to move or directory structure to be changed.
- Handling multiple parents is done in an inelegant manner with multiple directories and an ugly cd_dotdot command.
- Bit packing numbers into the QID is only useful if applications are aware of it.

6 Conclusion

There is a broader goal of making efficient use of data and reduce the engineering effort needed to access it, semantic file systems offer a useful paradigm to support this. They can be used as a capable store of persistent complex data and processes and furthermore with Plan 9 and Inferno, easily shared.

A longer term goal would be to integrate more knowledge and tools from computational genomics so that they become immediately and easily available, this includes complex data structures like phylogenetic trees, and tools like sequence comparision and alignment programs.

Acknowledgements

Funding for this work came from the BBSRC and Vita Nuova via a CASE studentship. There were several useful ideas and assistance provided by Charles Forsyth and Roger Peppe.

References

[ABB+00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet*, 25(1):25–9, 2000.

[DJ91] Mark A. Sheldon David K. Gifford, Pierre Jouvelot and James W. O'Toole Jr. Semantic file systems. *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pages 16–25, October 1991.

[DPB+96] J. DeRisi, L. Penland, P. O. Brown, M. L. Bittner, P. S. Meltzer, M. Ray, Y. Chen, Y. A. Su, and J. M. Trent. Use of a cDNA microarray to analyse gene expression patterns in human cancer. *Nat Genet*, 14(4):457–60, 1996. 1061-4036 Journal Article.

[FR00] S*é*bastien Ferre and Olivier Ridoux. A File System Based on Concept Analysis. In *Computational Logic*, pages 1033–1047, 2000.

[Gia93] Dominic Giampaolo. CAT-FS :–a content addressable, typed file system. Master's thesis, Worcester Polytechnic Institute, 1993.

[GM99] Burra Gopal and Udi Manber. Integrating content-based access mechanisms with hierarchical file systems. In *Proceedings of the 3rd Symposium on Operating Systans Design and Implementation (OSDI-99)*, pages 265–278, Berkeley, CA, February 22–25 1999. Usenix Association.

[GW99] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer Verlag, 1999.

[Inf] *Inferno Manual, Fourth Edition (Manual Pages)*. Vita Nuova Ltd.

[Laz02] Y. Lazebnik. Can a biologist fix a radio?–or, what i learned while studying apoptosis. *Cancer Cell*, 2(3):179–182, September 2002.

[Mar04] Ben Martin. Formal Concept Analysis and Semantic File Systems. Springer, 2004.

[Mar06] Ben Martin. The world is a libferris filesystem. *Linux J.*, 2006(146):7, 2006.

[Mor05] Peter Morville. *Findability*. O'Reilly Media, Inc., September 2005.

[NW05] J. C. Newman and A. M. Weiner. L2L: a simple tool for discovering the hidden significance in microarray expression data. *Genome Biol*, 6(9):R81, 2005.

[PSR06] Yoann Padioleau, Benjamin Sigonneau, and Olivier Ridoux. Lisfs: a logical information system as a file system. In *ICSE '06: Proceeding of the 28th international conference on Software engineering*, pages 803–806, New York, NY, USA, 2006. ACM Press.