

THE SFD DATASET BROWSER

<http://www.dina.kvl.dk/~sestoft/sfdbrowser>

Ulla Dindorp and Peter Sestoft

*Department of Mathematics and Physics
Royal Veterinary and Agricultural University, Denmark
e-mail: ud@dina.kvl.dk, sestoft@dina.kvl.dk*

Abstract: The SFD dataset browser is a new tool for viewing and browsing SFD datasets on the World Wide Web, using a standard web-browser such as Netscape Navigator. SFD is a data storage format and associated tools developed for use within agricultural research.

Keywords: Data storage, browser, Internet, CGI programs, HTML, functional programming.

1 Introduction

SFD [2] is a system for organizing experimental data. The purpose of SFD is to store data in a form suitable for communication between scientists and for exchange between programs for data processing.

The SFD system contains an editor for defining a *data model* written in the SFD language. The model determines the data *storage format* and may include annotations which provide an *interpretation* of the data. The SFD system permits automatic creation of data entry modules for a given data model, and permits data import from and export to ASCII files. The data model and data themselves are stored as a structured ASCII file.

To effectively exchange and reuse data, a description of the format and interpretation of the data must be available. These are provided by the SFD data model and the annotations. SFD has been used within the EU-project ADDA¹, dealing with exchange of agricultural data between EU member states.

The new SFD dataset browser is a World Wide Web front-end for viewing SFD datasets. It facilitates remote access to experimental data, since any SFD dataset can be displayed using a frames-capable HTML-browser. This allows researchers to make their data available to colleagues in a well-defined and yet easily accessible and understandable format.

Technically, the SFD dataset browser is a collection of so-called CGI programs for creating HTML code, to be interpreted by the web-browser. The SFD dataset browser has been programmed in the functional programming language Standard ML; this greatly reduced the time spent on implementation.

Section 2 describes the browser from the user's point of view. Sections 3 to 5 describe technical aspects of the browser.

¹This work was supported in part by the ADDA project (AIR-CT94-1330).

2 The SFD dataset browser

The SFD dataset browser has two kinds of windows. A *choose dataset* window, and a *show dataset* window. The SFD dataset to view is chosen from a list of available datasets in the *choose dataset* window². To make the system fast, the datasets have been converted to an internal format, stored in .sfi files. The system includes a program for creating .sfi files from ordinary SFD data files.

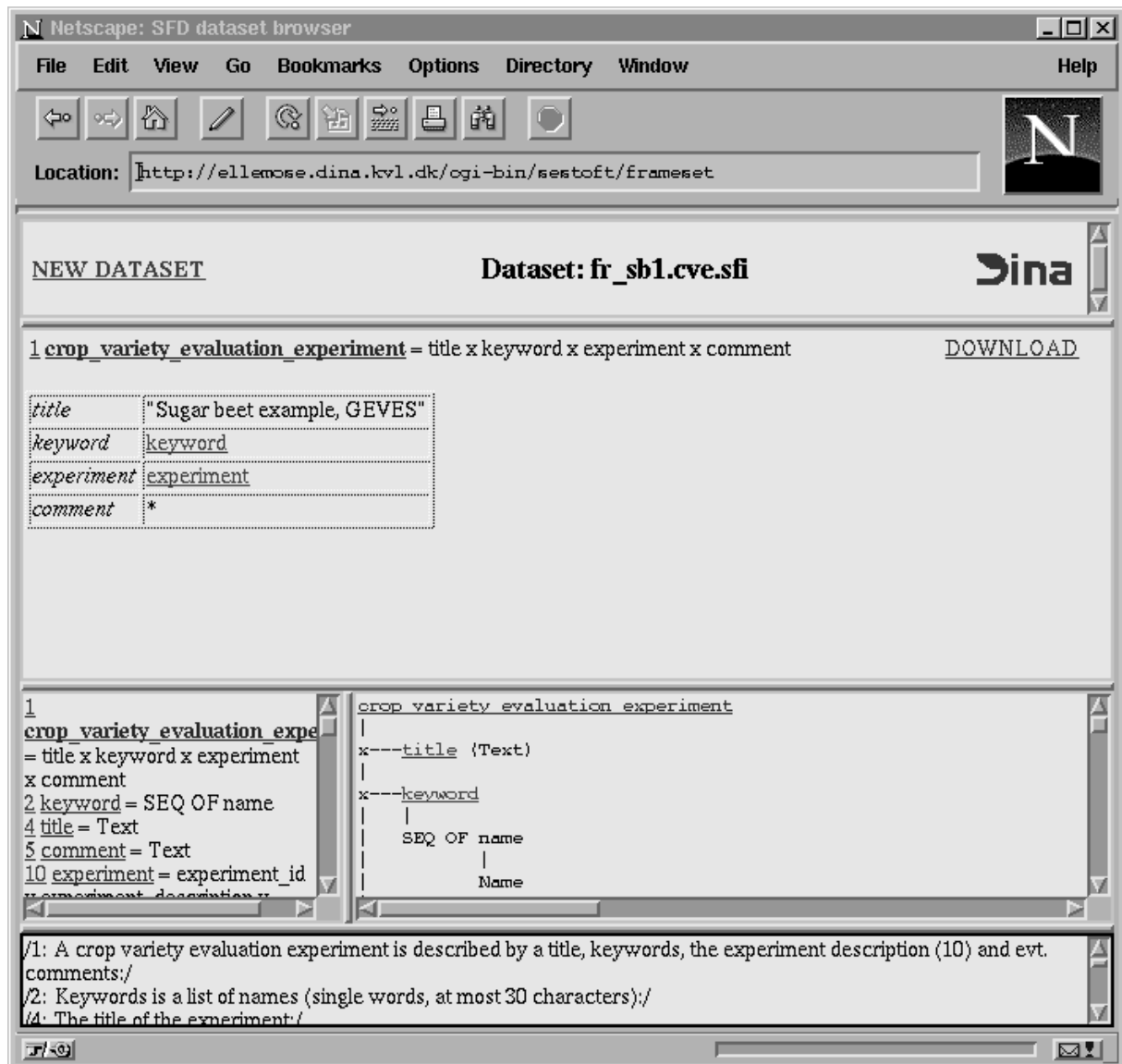


Figure 1: The show dataset window

2.1. Show dataset

The *show dataset* window is divided into five frames; see Figure 1. In order from top to bottom and left to right, the frames are

- The *header frame*, which displays the name of the SFD dataset.

² <http://www.dina.kvl.dk/~sestoft/sfdbrowser/>

- The *value frame*, which displays values and subvalues of the SFD dataset.
- The *model frame*, which displays the SFD model as a sequence of type declarations.
- The *type graph frame*, which displays the SFD model, or a submodel thereof, as a tree in a two-dimensional lay-out.
- The *annotation frame*, which displays the annotations of the SFD model.

The borders between the frames can be moved by the cursor to enlarge or shrink the individual frames.

The screenshot shows a web interface with three frames. The top frame, labeled 'TOP', contains a 'DOWNLOAD' link and a table of data. The middle frame, labeled 'UP', contains a type graph for 'data'. The bottom frame contains annotations for the data type.

trial_unit		measurements	
bloc	facteur1level	pdr	rs
1	'ALLYX'	62.9	18.95
1	'BETTIMO'	63.4	19.05
1	'KAWERENTA'	67.8	18.7
1	'REGINA'	61.2	18.85
1	'AMIDOR'	66.1	19.25

```

graph TD
    data[MAP trial_unit TO measurements]
    data --- pdr[x---pdr (Real)]
    data --- rs[x---rs (Real)]
    data --- bloc[x---bloc { [1;5] } ]
  
```

Annotations:

```

/500: Data is a map from the trialunit to the measurements:/
/510: The trialunit identifies the observation - here bloc x facteur1:/
/520: The block for the observation:/
/530: The measurements for the observation:/
  
```

Figure 2: The value frame (top) displays a table of type *data*, the graph frame (middle right) displays the type graph for *data*, and the annotation frame (bottom) displays the annotations about *data*.

2.1.1 The header frame

The header frame contains the name of the dataset currently viewed, and a link for picking another dataset; see Figure 1.

2.1.2 The value frame

The value frame is used for showing data from the dataset. Upon invocation it will show the top type in the data model, and the top of the data value; see Figure 1. The type equation is shown first, followed by the corresponding data. The data being displayed may be downloaded (in SFD format) by clicking on the **DOWNLOAD** link.

Values such as numbers, dates, and identifiers are simple SFD values. Simple values, and tables of simple values, are displayed as compactly as possible. Compound components of products and tables are replaced by hyperlinks which target the value frame with a new call to the CGI program, requesting to show the compound type, as in Figure 1. By using these links one can descend the value tree.

Sequences, sets and maps are composite SFD values. These are displayed as tables with a row for each element; see Figure 2. Products are displayed with a row for each element of the product. The table

header indicates the structure of the table. For instance, the header in Figure 2 shows that the map is from `trial_unit` to `measurements` and that `trial_unit` in turn is a product of `bloc` and `facteur1level`.

Type equations in the value frame have links to the annotation frame and the type graph frame: clicking on the line number in a type equation causes the annotation frame to display the annotation corresponding to the line number. Clicking on the type name on the left hand side of an equation will cause the type graph frame to display the subgraph rooted at that type name; see Figure 1.

When the value frame is showing a sub-value, two links appear in the top part of the frame, as shown in Figure 2. The links are used to climb up the value tree. The TOP link will return to the top-most part of the value. The UP link will return to the parent value of the current sub-value. For instance, the parent value of a set element is the entire set, and the parent value of a product component is the entire product. For instance, if the value frame is displaying the `data` table as in Figure 2, then pressing UP will cause the frame to show the `experiment`, which is a part of `crop_variety_evaluation_experiment` in Figure 1.

2.1.3 The model frame

The model frame contains the entire model from the dataset in the SFD language. Line numbers link to annotations in the annotation frame, and typenames link to the type graph frame.

2.1.4 The type graph frame

Initially, the type graph frame displays the entire type tree, drawn with lines and the symbols ‘x’ for products and ‘+’ for unions. A subtree of the type tree may be displayed by clicking on the typename in the value frame or model frame. If a typename has a corresponding annotation, then the typename in the graph frame will contain a link to that annotation in the annotation frame.

2.1.5 The annotation frame

The annotation frame contains the annotations from the dataset. From the value, model and the graph frames one can make the annotation window show the annotation for a given type or for a given type equation number.

3 Main components

There is a particular CGI program responsible for creating each of the five frames shown in Figure 1. These CGI programs are called `SFDheader`, `SFDvalue`, `SFDmodel`, `SFDgraph`, and `SFDannot`, respectively.

In addition, there is a CGI program `frameset`, which creates the entire frame set of the `show dataset` window, and invokes the above-mentioned programs to fill in the five frames. A CGI program `SFDretrieve` performs the DOWNLOAD in the `value` frame.

Finally, there is a utility program `internal` for parsing and checking an SFD source file and saving it on an internal format, with filename suffix ‘.sfi’.

The CGI programs are invoked by the web-browser to compute the HTML code describing the contents of a frame; the HTML code returned by the CGI program is then interpreted by the web-browser which fills in the frame. The CGI programs are invoked with an argument specifying the name of the SFD dataset, and possibly further arguments. The CGI programs themselves are quite simple; their main work is to obtain the CGI arguments, read the specified SFD dataset from the file system, and invoke utility functions for generating HTML code.

Each CGI program reads the entire SFD dataset on invocation. This is affordable because the datasets tend to be smaller than 1000 KB, and because they are stored in a pre-checked internal format, so no parsing and checking needs to be done.

4 Some central ideas

The SFD dataset browser is implemented as a functional program, with utility functions for formatting a CGI call, for lay-out of trees, for formatting an HTML table, etc. This programming style leads to general and widely reusable subprograms, minimizes (re)programming effort, and thereby facilitates experimentation.

As a consequence of this approach, the entire HTML text is built in memory before any part of it is printed to the standard output. Since the HTML text generated to describe a frame rarely exceeds 50 KB, this works well. All noticeable delay seems to be caused by the web-browser interpreting the HTML text, not by our programs generating it.

5 Implementation and experience

The CGI programs of the SFD dataset browser were written in Standard ML [3], a functional programming language. It has become customary to use Perl scripts for CGI programming, but Standard ML appears to be a better choice when non-trivial data processing is required. This idea came from Jonas Barklund's `Mosmlcgi` interface³.

Perl has elaborate support for string manipulation and for interacting with the operating system. Perl is an interpreted language, which makes experimentation easy.

However, Standard ML can claim most of these advantages, in addition to others. The new Basis Library [1] provides support for string and filename processing. The support for modularization is much better than in Perl, and separate recompilation control means that experimentation is as fast as for Perl, once a Makefile has been set up.

Moreover, Standard ML's static polymorphic type checking means that most programming errors are discovered automatically by the compiler, not by the user of the program, which is important for quality programming. By contrast, Perl is an interpreted and untyped language, so errors are discovered late.

The Standard ML programs were compiled with the Moscow ML system [4] which generates compact bytecode files. The largest CGI program is `SFDvalue`, for which the compiled bytecode is 25 KB. To this we must add the shared runtime system, which is 75 KB. By contrast, the Perl interpreter itself is 523 KB, to which must be added the size of the scripts (programs). Moreover, a Perl script must be compiled every time it is invoked, which takes time, whereas ML programs are compiled once and for all. The upshot is that compiled Moscow ML programs load and execute at least as fast as Perl scripts.

The programs and the lexer and parser specifications making up the SFD dataset browser are quite compact; their total size is approximately 2100 lines.

Using appropriate buzzwords, we have found Standard ML to be well-suited for 'rapid prototyping' and 'rapid application development'. The HTML standard and the excellent web-browser from Netscape made the development of window-based interactive programs easy.

6 Plans and further work

Some ideas for further extensions to the browser is:

- From a given model, automatically create HTML forms for data entry, corresponding to the data entry modules generated by the existing SFD system. Allow such newly created datasets to be saved on the user's machine.
- Permit export of SFD (sub)datasets from the server to the user's machine in a general format, such as 'comma separated variables', readable by the SAS system, by commonly used spreadsheets, etc.

³<http://www.csd.uu.se/~jonas/mosmlcgi/> or <http://www.dina.kvl.dk/~sestoft/mosmlib/Mosmlcgi.html>

7 Conclusion

Within agricultural research an immense number of experiments are performed each year, producing large amounts of primary data. Money, time, and effort are spent producing these data; reusing such primary data in other investigations makes good sense from an economical point of view.

SFD has been developed as a data format which includes documentation of the data's structure and interpretation. By storing data in the SFD format, researchers can make them available to colleagues in a well-defined and easily accessible format.

The new SFD browser facilitates remote access to experimental data. With the SFD browser any SFD dataset can be displayed anywhere on the World Wide Web using a standard web-browser. Anyone interested can download all or parts of the data as SFD datasets with data model, annotations and data in the same file. The datasets are ASCII files, and data are easy to extract either directly from the files or via the freely available SFD system⁴.

3 References

- [1] E. Gansner and J. Reppy. Standard ML Basis Library. Technical report, AT&T Research, 1996.
- [2] J.T. Kristensen, M. Gudmand-Høyer, and L. Thygesen. SFD standard format for data exchange. User's manual. Dina Publication 2, Royal Veterinary and Agricultural University, Copenhagen, Denmark, 1993. Available as <ftp://ftp.dina.kvl.dk/pub/sfd/sfd-man.zip>.
- [3] R. Milner, M. Tofte, and R. Harper. *The Definition of Standard ML*. The MIT Press, 1990.
- [4] S. Romanenko and P. Sestoft. *Moscow ML Owner's Manual, version 1.41*, October 1996. Available at <http://www.dina.kvl.dk/~sestoft/mosml.html>.

⁴<ftp://ftp.dina.kvl.dk/pub/SFD/sfd12.zip>