The Atavachron

for Pre-Prints

version 1.1

by James E. Hetrick <hetrick@phys.uva.nl>

The **Atavachron** is an interface to the *Astro/Math/Physics* preprint archives from within the GNU Emacs editor. It gives the user a simple set of commands to get, store, process, and list papers with minimal effort.

This TEXinfo documentation was compiled
by James Hetrick and Marcus Speh <marcus@x4u.desy.de>

Last change: 1/19/93.

1 Copying Conditions

1.1 Obtaining This Package

At the time of writing (January 1993), the `atavachron` package is intended to be placed in the `macros` directory of the supported archives. Thus it should hopefully be accessible by February 1993 through anonymous ftp or by electronic mail from your favorite preprint site.

Copies will be submitted to *`prep.ai.mit.edu`* and *`archive.cis.ohio-state.edu`* for permenant archiving, and announcements will be made to the relevant Newsgroups (*`gnu.emacs.source`* and *`sci.phys`*, at least).

Of course, the latest version can always be obtained from the author directly at the address given in see Chapter 2 [General Description], page 2.

2 General Description

The **Atavachron** (pronounced: a · **tav**′· a · kron) is an intelligent interface or "Front End" to the scientific preprint archives from within emacs. The preprint archives supported are only those running Paul Ginsparg's (<ginsparg@qfwfq.lanl.gov>) Preprint Bulletin Board software. At present the supported archives are:

alg-geom: Algebraic Geometry

astro-ph: Astrophysics

cond-mat: Condensed Matter Physics

funct-an: Functional Analysis

gr-qc: General Relativity and Quantum Cosmology

hep-lat: High Energy Physics — Lattice

hep-ph: High Energy Physics — Phenomenology

hep-th: High Energy Physics — Theory

lc-om: Liquid Crystals

math-ph: Mathematical Physics (at babbage.sissa.it)

nucl-th: Nuclear Theory

Should you wish to add another archive please contact the author (via *atavachron-mail* if you like see Section 5.5 [Utilities], page 14). Supporting other disciplines which use TEX as their main typesetting format would be very simple and rewarding.

The **Atavachron** gives its user a simple set of commands to get, store, process, and list papers which can be called from the *mini-buffer* (the bottom most, one line window in Emacs used for command input and message output). The key strokes: *M-x* invoke Emacs functions; on most keyboards, *M* is *ESC*. The package is easily loaded via your `.emacs` file (see Chapter 3 [Configuration], page 4) and then provides several functions which can be called at any time.

The **Atavachron**'s user functions are listed here and described below (see Chapter 5 [Functions], page 9):

```
get-paper
get-paper-from-data
store-paper
store-paper-from-data
process-paper
slice-and-dice
shell-paper
get-listing-from-data
store-listing-from-data
get-abstract
get-abstract-from-data
interesting
atavachron-mail
```

You must go through Chapter 3 [Configuration], page 4, and Chapter 4 [Batteries Not Included], page 7, and set things appropriately for your system. The variables to be set

govern such things as where papers gotten with *get-paper* are put, whether they are renamed, and the address of your default archive.

Note that the documentation here is meant to be somewhat pedagogical for new users to Emacs. If you are such a user, your comments on the accessibility of this manual would be most appreciated.

`atavachron.el` requires `ange-ftp.el` for retrieving files across the network. In addition there are auxiliary files used by the **Atavachron** which greatly enhance its abilities. The purpose of these files is described in Chapter 4 [Batteries Not Included], page 7.

If you find bugs, please report them; this is very easy with the function '`atavachron-mail`'. You're welcome to use this function for other correspondence as well (See Section 5.5 [Utilities], page 14).

Why "Atavachron"?

This package was originally named `pp-tools.el` in its development stage, when a previously existing "pp-tools" package turned up in a network search. Thus the name had to be changed.

The name **Atavachron** comes from the original *Star Trek* series, in the episode "All Our Yesterdays". Kirk, Spock, and McCoy beam down to a planet with this wonderful library. Library users place disks into viewers which show images from periods of the planet's past. The library also has a portal through which users can travel backwards in time (for various reasons which probably included research).

All the disk viewers are interfaced to a machine called the **Atavachron** which senses which disk the user was viewing and configures the portal such that when the user steps through, he passes into the corresponding period of time.

Due to the similarities in functionality between the present package and the 23rd century's archive interface (at least to the imaginative mind), the name **Atavachron** seemed appropriate.

Acknowledgements

Many thanks to Paul Ginsparg, Greg Kilcup, Marcus Speh, and Bas de Bakker for ongoing discussions, suggestions, and beta tests.

The text for the first edition of this document was taken from the text description distributed with version 1.3 of `pp-tools` and set into Texinfo by Marcus Speh (If you don't have Texinfo, you can obtain it from *prep.ai.mit.edu* via anonymous FTP).

The change over to the **Atavachron** package version 1.1 and documentation of subsequent additions to the source has been mostly done by Jim Hetrick.

Author's address:

James E. Hetrick Institute for Theoretical Physics
hetrick@phys.uva.nl -\|/-
University of Amsterdam
Valckenierstraat 65, 1018 XE Amsterdam
Telephone: +31 20 525 5772 Fax: +31 20 525 5788

3 Configuration

This node describes the variables which configure the **Atavachron** within Emacs. *Configuring* means setting environment variables for Emacs. These definitions are usually put into Emacs' initialization file `.emacs`. If you want to try them out, you may use any buffer in '`Emacs-lisp`' mode, then use *`(M-x) eval-buffer`*, but remember that `atavachron.el` should really be reloaded everytime you change the configuration.

For users new to Emacs, there are two types of variables used below. Strings, which must be enclosed by `"`'s, are used for directories, archive sites, and so forth. The other kind is a logical variable which is either *t* (true or ON) or *nil* (false or OFF). Logical variables are **not** enclosed in quotations.

3.1 Example `.emacs` File

Below is an example `.emacs` file which loads `atavachron.el` and the batteries, then sets the *`atv`*-variables to their defaults. **Please** go through each variable and set it according to your own directory structure and preferences.

Regarding `ange-ftp.el`, the simplest method of loading has been given here, for the case in which it has not been loaded previously in your `.emacs` file. If it is loaded previously be sure to delete the *`(load-library '"ange-ftp")`* line here, so as not to load it twice. Since `atavachron.el` contains the `require 'ange-ftp`, users of more advanced auto-loading techniques may keep their mechanisms intact.

Although not discussed here, any **Atavachron** function can be bound to a key sequence or mouse button action. Such bindings should go here as well. See the Emacs manual on key rebinding and the documentation of your mouse interface for more details.

```
; ====================.emacs stuff ========================

(load-library '"ange-ftp")  ;see ange-ftp docs for more details
(setq ange-ftp-generate-anonymous-password t) ; sends email address
                                              ; for password
(load-library '"zcat")
(load-library '"atavachron")

(setq atv-main-ftp-site "/anonymous\@babbage.sissa.it:") ;Europeans
;;(setq atv-main-ftp-site "/anonymous\@xxx.lanl.gov:")  ;N+S Americans

(setq atv-download-dir "~/tex/preprints/")   ;default download directory
(setq atv-uufiles-dir "~/tex/preprints/")    ;default place for uu output
(setq atv-prompt-filename t)              ;prompt to rename paper locally
(setq atv-interesting-file "~/interesting.atv") ;file for interesting abs.
(setq atv-auto-tex-paper nil)     ;run tex on paper grabbed automatically
(setq atv-auto-tex-command 'TeX-buffer)        ;command to run on buffer
(setq atv-no-csh nil)                     ;set to t if you don't have csh

; ========================================================
```

3.2 'atv-main-ftp-site'

```
; ====================.emacs stuff ======================
 (setq atv-main-ftp-site "/anonymous\@babbage\.sissa\.it:"); Europeans
;; (setq atv-main-ftp-site "/anonymous\@xxx\.lanl\.gov:")  ; N+S Americans
; ========================================================= ; Others chose one
```

Uncomment one of these as your main archive. Branches not supported at '`xxx.lanl.gov`', such as *cond-mat*, are retrieved from '`babbage.sissa.it`'. Papers on *hep-lat*, *alg-geom*, etc. which are not mirrored at the above two sites are automatically gotten >from their respective archives at present.

3.3 'atv-download-dir'

```
; ====================.emacs stuff ======================
(setq atv-download-dir "~/tex/preprints/")    ;default download directory
(setq atv-uufiles-dir "~/tex/preprints/")     ;default place for uu output
; ========================================================
```

This is your local download workspace. Upon "getting" a paper, the default directory is reset to '`atv-download-dir`' so that saves put files there. In addition tarred papers (archived and compressed as .tar.Z files), and script papers (papers which are shell scripts to be executed) are processed in this directory so that any files unpacked end up there.

If you have branch directories *hep-th*, *hep-lat*, *astro-ph*, etc. below '`atv-download-dir`', papers, when saved, will automatically be put there, sorted by branch. For example, your '`atv-download-dir`' is set to `~/tex/preprints/` and you have the subdirs `~/tex/preprints/hep-th/` and `~/tex/preprints/cond-mat/`. Then getting *hep-th/91120013* will put the file in `tex/preprints/hep-th/`. Getting *cond-mat/920106* will put it in `tex/preprints/cond-mat/`. If these subdirs do not exist below `tex/preprints/`, both papers would be put in `tex/preprints/`. The recognition of branched subdirectories is mainly provided for very heavy or library related use.

Be sure to put this directory (and branches) in your *TEXINPUTS* environment variable so that tex can get at your newly gotten files.

3.4 'atv-uufiles-dir'

This is where uuencoded files will be unpacked. It should usually be the same as '`atv-download-dir`', unless you have a good reason to put the figures elsewhere [your *dvi* —> *ps* processor can't get at them in '`atv-download-dir`', for instance. (in which case you should get '`dvips`')]. If it's the same as '`atv-download-dir`', it will adjust itself to the branched subdirectory structure following '`atv-download-dir`'.

3.5 'atv-prompt-filename'

```
; ====================.emacs stuff ======================
(setq atv-prompt-filename t)        ; prompt to rename paper locally
; ========================================================
```

This one toggles prompting for a local name for the grabbed paper or listing. When *t* (true), you are prompted to name the incoming archive file which is then appended with

`.tex`. Hitting return at the prompt will accept the default given in parenthesis. If *nil* the paper is named by its paper number.

3.6 'atv-interesting-file'

```
; ========================================
(setq atv-interesting-file "~/interesting.atv")  ; file for interesting
; ========================================  ; abstracts.
```

The default file to which abstracts stored with *interesting* function are appended (see Section 5.5 [Utilities], page 14). You are prompted for a file name to append the interesting abstract to, and offered '`atv-interesting-file`' as the default.

I find it useful to put this file in a directory other than '`atv-download-dir`' since I regularly clean '`atv-download-dir`' with *rm **.

3.7 'atv-auto-tex-paper'

```
; ====================.emacs stuff ======================
(setq atv-auto-tex-paper nil)    ; run tex on paper grabbed automatically
(setq atv-auto-tex-command 'TeX-buffer)  ; command to run on buffer
; ========================================================
```

Setting '`atv-auto-tex-paper`' to *t*, automatically starts the tex processor as given by the function '`atv-auto-tex-command`' on the paper once it's loaded into a local buffer, provided no shell command signal is found. The default is that auto-texxing is *OFF*.

Note that adding auto-previewing is quite simple as well (e-mail for details), and will impress your friends if the paper texs, but is not really all that useful since many papers don't pass the texxing stage, as you probably know! As this changes, and authors submit standardized TEX, auto-tex/preview could become useful.

Originally provided as extra frill, these variables can actually be a powerful expansion slot. You can supply any function you like for '`atv-auto-tex-command`', in particular your own custom ones. Thus after processing all auxillary files and getting the primary TEX file, '`atv-auto-tex-command`' provides a final hook to execute custom processing.

3.8 'atv-no-csh'

```
; ====================.emacs stuff ======================
(setq atv-no-csh nil)            ; set to t if you don't have csh
; ========================================================
```

Setting this variable to *t* will replace the '`/bin/csh`' in the grabbed uufile to '`/bin/sh`', making it possible to unpack the figure files even if you don't have '`csh`' on your machine. If you use this feature, please read the copyright Chapter 1 [Copying Conditions], page 1, now, especially where it says:

"This program is distributed in the hope that it will be useful, but **without any warranty**..."

4 Batteries Not Included

Here we discuss the other files needed or recommended for smooth operation of `atavachron.el`. The only one absolutely required is `ange-ftp.el`, however you'll have to decipher a compressed binary file if you get a month old paper without `zcat.el`. The files `.netrc` and `.dvipsrc` eliminate the seams at the connections to the network and TEX.

Loading the elisp (`.el`) files in your `.emacs` file is discussed in Chapter 3 [Configuration], page 4.

4.1 ange-ftp.el

• This file is available via anonymous FTP from '`archive.cis.ohio-state.edu`' in directory `/pub/gnu/emacs/elisp-archive/packages/`.

`atavachron.el` requires Andy Norman's essential ftp interface `ange-ftp.el` to do the remote file teleportation If you don't have this loaded in your `.emacs` file, you must get it. Check locally if you have it (perhaps in `/usr/local/emacs/lisp` or there abouts?), or convince your system administrator to install it (or do it yourself).

4.2 zcat.el

• This file is included in the `atavachron` distribution package.

Another file needed is Graham Gough's `zcat.el` which uncompresses `.Z` files on the fly. This makes loading older papers which are compressed at the archive transparent. Since it's short it's included with the distribution package of `atavachron`.

4.3 .netrc

• You can easily write your own `.netrc` file, see example below.

Ange-ftp will work much smoother if you keep a `.netrc` file in your home directory, thus bypassing the need to send the *anonymous* login and email address when ftping into the archive. Simply clip out the following and name it `~/.netrc`. Remember to put your email address in for you@where.you.are).

sample .netrc

```
#   --------------------.netrc sample -----------------------
machine ftp.scri.fsu.edu login anonymous password you@where.you.are
machine xxx.lanl.gov login anonymous password you@where.you.are
machine babbage.sissa.it login anonymous password you@where.you.are
machine publications.math.duke.edu login anonymous password you@where.you.are
machine alcom-p.cwru.edu login anonymous password you@where.you.are
#   ------------------------------------------------------------
```

Furthermore, it must be a protected file. Run: '`chmod go-rwx .netrc`' on it once it has been created.

4.4 .dvipsrc

• You can easily write your own `.dvipsrc` file, see example below.

Finally, if you use `dvips` as your *.dvi* —> *.ps* processor, put your '`atv-download-dir`' and '`atv-uufiles-dir`' (see Chapter 3 [Configuration], page 4) in its path by keeping a `.dvipsrc` file in your home directory with the following lines:

sample .dvipsrc

```
# -------------------.dvipsrc sample ---------------------
S ~/tex/preprints:
H ~/tex/preprints:
W "reading your .dvipsrc file"
# ------------------------------------------------------
```

The trailing `:`'s above add the default paths to the path list. A line beginning with '`W`' echos the following string.

Having '`atv-download-dir`' and '`atv-uufiles-dir`' in both your '`dvips`' path and your *TEXINPUTS* environment variable (or whatever the path to TEX input and style files is named on your local machine) allows '`dvips`' and TEX to access your new figure files.

For more info, see your man pages on TEX and '`dvips`' or your system administrator, whichever is friendlier.

5 Functions

This section describes the functions available to the user for interacting with the archive. In addition three example sessions are displayed in Chapter 7 [Example Sessions], page 17, where the reader can get a quick feel for the operation of the **Atavachron** on different types of papers.

5.1 Getting Papers

The following functions get an archive paper to an Emacs buffer (or buffers if it is distributed in several files). After the paper is in a local buffer, action appropriate to the format of the paper is taken. For instance, a tar file is uncompressed and unpacked, or uu encoded files are decoded, as described below in Chapter 6 [Paper Formats], page 15.

- `Function: (M-x)  get-paper`

When reading an abstract listing in emacs, either in one of the mail readers or as a previously saved file (output by Unix Mail perhaps), placing the point (cursor) anywhere in the abstract of choice (after the line beginning with '`Paper`'), and typing: *`M-x get-paper`* automatically opens an ftp socket into the right archive (via `ange-ftp.el`), transfers the paper, uncompresses it if necessary (via `zcat.el`), and loads it into a '`tex-mode`' buffer for [La]TEXing, previewing, and printing by the usual '`tex-mode`' facilities.

First though, the **Atavachron** performs a number of tests to determine the format in which the paper is stored at the archive. These steps are also performed by '`process-paper`' if it is called on a stored paper (See Section 5.3 [Processing Functions], page 10, Section 5.2 [Storing Papers], page 9.) Once its format is determined, appropriate action is taken based on the configuration variables (see Chapter 3 [Configuration], page 4). The description of the paper formats supported in the present version (1.1) is given below (see see Chapter 6 [Paper Formats], page 15).

If '`atv-prompt-filename`' is *t* (true), one is prompted to give a local name to the incoming paper, and the paper is renamed `localname.ext`, where `.ext` depends on the format of the paper.

The buffer(s) containing the paper are marked as changed so that you will be prompted to save it if you try to delete it.

- `Function: (M-x)  get-paper-from-data`

This function works exactly like *`get-paper`* except that it promps the user to input the paper's data in the usual format:

```
cond-mat/9212003
hep-lat/9212005
etc.
```

5.2 Storing Papers

- `Function: (M-x)  store-paper`
- `Function: (M-x)  store-paper-from-data`

'`Store-paper`', and '`store-paper-from-data`', work the same as '`get-paper`' and '`get-paper-from-data`', except that the paper is stored directly to the user's default

download directory (see Chapter 3 [Configuration], page 4) for later processing. This is convenient when reading the abstracts remotely (on the weekend from home, say) when one wants to save processing and printing the tex file for later, or so that the file can be further downloaded via modem. (see Chapter 7 [Example Sessions], page 17)

5.3 Processing Functions

Three functions are presently provided for operating on preprint files individually. These are stand alone versions of functions which are called by *get-paper* and are useful for processing stored papers or individual buffers of a multifile paper.

In addition, the descriptions of the Emacs functions *buffer-menu* and *insert-buffer* are included for completeness; they are quite useful when a retrieved paper consists of several files which must be processed.

5.3.1 Stand Alone Internal Functions

- `Function: (M-x) process-paper`

Use this function on a paper that has been stored locally with *`store-paper`* (see Section 5.2 [Storing Papers], page 9) to invoke the functions which determine the paper's format (see Chapter 6 [Paper Formats], page 15) and execute processing accordingly.

For example, you have stored a particular paper with the local name `jones` earlier. *`process-paper`* will ask for the file name to be processed, `jones` in this case, then examine this file (or files) in '`atv-download-dir`' to determine its format. Having deduced this it takes appropriate action such as stripping appended PostScript figures, executing shell scripts, etc. For more details see See Chapter 6 [Paper Formats], page 15, and Chapter 7 [Example Sessions], page 17.

- `Function: (M-x) slice-and-dice`

`slice-and-dice` extracts appended PostScript figures from a file which is loaded in the current buffer. It will search for each occurance of `"^%!"` and strip the region between this line and the next `"^%!"` or the end of file. For each such region it pauses for the filename to be given to the figure, offering a default (in parenthesis) gathered from the comment area preceeding the PostScript signal.

- `Function: (M-x) shell-paper`

This function is the one called when the retrieved paper is found to contain the regexp `"^#![ \t]*\\(.*/bin/.*\\)$"`. Everything below and including this line is extracted to a file `localname.sh` which is then executed by Emacs with *`shell-command`* `"(match) localname.sh"`, where (match) is the executable shell name and path extracted from the file. Thus a variety of shells are supported, includng for instance, *Perl* scripts. A file `localname.rpt` is made which catches any standard output of the execution.

You can use this function on any buffer containing a non-interactive shell script, like a shar file sent to you in the mail. *`shell-paper`* strips the header and executes the shar file without leaving mail. Remember that the script is executed in *`atv-download-dir`*.

5.3.2 Operating on Several Buffers in Emacs

See: See Info file `emacs`, node '`Several Buffers`'.

- `Function: (M-x) buffer-menu`

Begin editing a buffer listing all Emacs buffers.

The "buffer-menu" facility is like a "Dired for buffers"; it allows you to request operations on various Emacs buffers by editing an Emacs buffer containing a list of them. You can save buffers, kill them (here called "deleting" them, for consistency with Dired), or display them.

The command *`buffer-menu`* writes a list of all Emacs buffers into the buffer `*Buffer List*`, and selects that buffer in Buffer Menu mode. The buffer is read-only, and can only be changed through the special commands described in this section. Most of these commands are graphic characters. The usual Emacs cursor motion commands can be used in the `*Buffer List*` buffer. The following special commands apply to the buffer described on the current line.

`d` Request to delete (kill) the buffer, then move down. The request shows as a `D` on the line, before the buffer name. Requested deletions take place when the `x` command is used.

`k` Synonym for `d`.

`C-d` Like `d` but move up afterwards instead of down.

`s` Request to save the buffer. The request shows as an `S` on the line. Requested saves take place when the `x` command is used. You may request both saving and deletion for the same buffer.

`~` Mark buffer "unmodified". The command `~` does this immediately when typed.

`x` Perform previously requested deletions and saves.

`u` Remove any request made for the current line, and move down.

`DEL` Move to previous line and remove any request made for that line.

All the commands that put in or remove flags to request later operations also move down a line, and accept a numeric argument as a repeat count, unless otherwise specified.

There are also special commands to use the buffer list to select another buffer, and to specify one or more other buffers for display in additional windows.

`1` Select the buffer in a full-screen window. This command takes effect immediately.

`2` Immediately set up two windows, with this buffer in one, and the previously selected buffer (aside from the buffer '*Buffer List*') in the other.

`f` Immediately select the buffer in place of the `*Buffer List*` buffer.

`o` Immediately select the buffer in another window as if by `C-x 4 b`, leaving `*Buffer List*` visible.

`q` Immediately select this buffer, and also display in other windows any buffers previously flagged with the `m` command. If there are no such buffers, this command is equivalent to `1`.

`m` Flag this buffer to be displayed in another window if the `q` command is used. The request shows as a `>` at the beginning of the line. The same buffer may not have both a delete request and a display request.

All that `buffer-menu` does directly is create and select a suitable buffer, and turn on Buffer Menu mode. Everything else described above is implemented by the special commands provided in Buffer Menu mode. One consequence of this is that you can switch from the `*Buffer List*` buffer to another Emacs buffer, and edit there. You can reselect the `buffer-menu` buffer later, to perform the operations already requested, or you can kill it, or pay no further attention to it.

The only difference between `buffer-menu` and `list-buffers` is that `buffer-menu` selects the `*Buffer List*` buffer and `list-buffers` does not. If you run `list-buffers` (that is, type `C-x C-b`) and select the buffer list manually, you can use all of the commands described here.

Also the command *insert-buffer* is very useful for concatenating buffers of multifile papers.

See: See Info file elisp, node 'Commands for Insertion'.

- Function: (M-x) insert-buffer FROM-BUFFER-OR-NAME

This function inserts the entire contents of FROM-BUFFER-OR-NAME (which must exist) into the current buffer after point. It leaves the mark after the inserted text. The value is unpredictable.

5.4 Title Listings and Abstracts

- `Function: (M-x) get-listing-from-data`
- `Function: (M-x) store-listing-from-data`

'`get-listing-from-data`' and '`store-listing-from-data`' work the same way, except on title listings. They prompt the user for the listing in the usual format: '`hep-th/9204`', then *get* (to a buffer), or *store* (to a file) the given month's listing.

Note that if the listing is gotten with *`get-listing-from-data`* into a buffer, then *`get-paper`* will work in that buffer as in the daily abstract list. Simply put the cursor on the paper of interest (**after** the line beginning with *Paper:* and type *`(M-x:) get-paper`*.

- `Function: (M-x) get-abstract`
- `Function: (M-x) get-abstract-from-data`

These two functions get abstracts to a new window and are useful in two situations mainly: use *`get-abstract`* when reading a monthly title listing where just the titles are given, to get the paper's abstract. *`get-abstract-from-data`* is handy to view the abstract of a reference for which you have the archive number.

5.5 Utilities

- `Function: (M-x) interesting`

This function is a tool for saving interesting abstracts to a file which can then be examined sometime later. It is used to avoid having to save the entire Mail abstract posting, and then later sifting through all abstracts. It can also be used to save abstracts to different files according to subject, etc.

When the point is placed in an abstract, type *`M-x: interesting`*. You are presented with the prompt to save the abstract to the default interesting file,

```
>>> Append to? (interesting.atv):
```

`[RETURN]` saves to the default, set by '`atv-interesting-file`' (see Section 3.6 [atv-interesting-file], page 6), or you may enter another filename.

- `Function: (M-x) atavachron-mail`

This handy function allows you to easily send a bug report, comment, or question to the author. It opens a mail buffer addressed to me (*Jim Hetrick)* and dumps the current values of all **Atavachron** variables, which can be used to reconstruct your situation. Please feel free to use this function any time you would like to contact me with questions, suggestions, or other correspondence.

6 Paper Formats

The paper formats presently supported by the **Atavachron** version `1.1` are listed here with a brief desription of the action taken for each (see Section 5.3 [Processing Functions], page 10, for more). The paper's format is determined by listing all files which match the paper's number in the correct branch directory at the archive.

6.1 Single TEX File

If the paper is a single file and no *PostScript* signal or shell commands are found, the paper is renamed `localname.tex` upon transfer to a local buffer, and '`tex-mode`' is loaded. It is then ready for processing via the usual '`tex-mode`' facilities,

- *C-c C-b* for [La]TEXing,
- *C-c C-p* for printing,

or any other local tex or '`auc-tex`' features, such as previewing. If '`atv-auto-tex-paper`' is *t*, [La]TEX is automatically started at this point.

6.2 Single TEX File with Appended *PostScript* Figures

A single paper file is first checked for any appended *PostScript* figures, which are then surgically removed to `.ps` files at the user's request (see Section 5.3 [Processing Functions], page 10). The whole file is first saved though, with extension `.tpl`. The `slice-and-dice` function (which removes the figures) tries to find a name for the figure file from the comments near the top of the *PostScript* header.

With the advent of the bulletin board macro '`uufiles`' to package figures, data files, or other auxiliary files, the occurance of *all-in-one* papers will hopefully deminish.

6.3 Shell Script

After checking for *PostScript* signs, the **Atavachron** checks for shell commands, in particular for `"#!.*/bin/"`. If this string is found, the region below `#!.*/bin/` is saved to a file `localname.sh` which is then executed (see Section 5.3 [Processing Functions], page 10). A report of the execution (including any terminal output made by the script) is made and named `localname.rpt`.

6.4 Tar File

If the archive paper is a tarfile, it is renamed `localname.tar.Z` and, upon confirmation, unpacked in '`atv-download-dir`'. A report of any output during the unpacking process is presented in the buffer `localname.rpt` which is then searched for a unique TEX file. If found this file is loaded to a buffer.

6.5 Multiple Archive Files for a Paper

- UUENCODED Auxillary Files (Figures, Data, etc.)

When the **Atavachron** finds multiple files for a given paper, it first checks for the extention `.2` and if found, assumes sequential integer extensions for all files. The paper files

`localname.2` through `localname.N` are copied and checked as to whether they are **uuencoded** files produced by Paul Ginsparg's preprint bulliten board macro `uufiles`. If so, they are **uudecoded**, ie. their contents is unpacked in '`atv-uufiles-dir`'. The figure files should then be accessible to TEX and '`dvips`' via the paths specified in the environment variable *TEXINPUTS* and in the file `.dvipsrc` (see see Section 4.4 [.dvipsrc], page 8). If '`csh`' is not available on your machine, you can set '`atv-no-csh`' to *t* (see Chapter 3 [Configuration], page 4) and still unpack the uufiles.

A file named `localnameN.rpt` is produced in '`atv-uufiles-dir`' for each auxillary file, which records the output of the command '`tar -xvf -`' when unpacking (each of the N) uufiles. This file is then a manifest of figures/files included with this paper.

• NON-UUENCODED Files

If the extra archive files are not found to be **uufiles**, or they do not have sequential integer extensions, they are simply copied to separate Emacs buffers to be processed by hand. To facilitate this, *`list-buffers`* is called which lists all the buffers of the current session in *buffer-menu-mode* which makes working with several buffers much easier (see Section 5.3 [Processing Functions], page 10).

Finally, `localname.1` is transfered and treated as the primary file of the paper. Thus it is assumed to be a TEX document (although tests for *PostScript* and shell commands are automatically done), and will either make use of the decoded uufiles, or hopefully contain instructions for manually processing the set of files displayed by *`list-buffers`*.

6.6 Multiple Non-Integer Extensions

If multiple files with non-integer extensions are found or if the **Atavachron** is confused about the nature of the files on the remote archive, the files are stored directly on the user's system in '`atv-download-dir`'; these files will in general require conscious processing.

7 Example Sessions

This section shows some sample sessions of what should happen when getting specific papers of various formats. These are intended to illustrate the most common formats and certainly do not exaust the permutations of papers and functions. Below, `>>>` means that the following prompt or message appears in the mini-buffer.

7.1 Example: Single TEX File

The point is located in the abstract of Witten's paper *hep-th/9204083*. If '`atv-prompt-filename`' is *t*, doing

```
M-x get-paper
```

asks:

```
>>>   Local name for paper? [.tex appended] (9204083):
```

Typing *witten*, renames this paper to `witten.tex`, while simply hitting *[RETURN]*, preserves it's archive name and adds `.tex` yielding the buffer `9204083.tex`. Note that as an older paper it is compressed at the archive, so `9204083.Z` is transferred and uncompressed on the fly.

The file is gotten to a new window in '`tex-mode`'. Now typing *C-c C-b* (*Control and c* then *Control and b*), invokes TEX on the file, and *C-c C-p* prints it.

Thus in four simple of keystrokes, after reading the abstract in the mail, the paper is in the printer bin. This could even be reduced to a mouse click and the print command.

7.2 Example: TEX File with Appended *PostScript*

Reading a reference to the paper *hep-lat/9212017* we type:

```
M-x get-paper-from-data
```

the **Atavachron** asks:

```
>>>   Local name for paper? [.tex appended] (9212017):
```

We hit *[RETURN]* to accept the default name '`9212017`' and the paper is retrieved. We are notified of appended *PostScript* figures

```
>>>   Looks like there are appended PostScript figures...
```

then asked by the **Atavachron**

```
>>>   Shall I surgically remove the figures? (y or n)
```

Entering *y* gets the reply "'`Entering slice and dice mode...`'". First a backup is made called `9212017.tpl`. Then, upon demarking the first figure, the **Atavachron** asks

```
>>>   Name for figure file below this point? (bumpy-operator.ps):
```

Hitting *[RETURN]* accepts the default name `bumpy-operator.ps` which was found in the nearby comments. The *PostScript* figure `bumpy-operator.ps` is extracted; this process is repeated four times extracting each of the four appended figures. On the last figure the **Atavachron** asks:

```
>>>   Name for figure file below this point? (* NONE FOUND *):
```

Note the '`* NONE FOUND *`'; this is because of a typing error in the original paper file where the figure is labeled *binning-error.p*, and the *.ps* search fails. In this case `binning-error.ps` (or some other name) **must** be entered at the above prompt by hand.

Slice and dice mode is now complete and the TEX portion of the original paper is returned, ready for '`tex-mode`' processing. In addition, a report of the figure extraction is made in the file `9212017.figs`:

```
 * -\|/- The Atavachron  (1.1) *
Below are the PostScript figures extracted from: 9212017.tex
-----------------------------------------------
bumpy-operator.ps
interpolation.ps
surface-energy.ps
binning-error.ps
```

7.3 Example: Storing and Processing a Paper (with uuencoded Figures)

While logged in from home, you read the abstract of Distler's paper *hep-th/9212062*. You type:

```
M-x store-paper
```

with the point in the abstract. The **Atavachron** asks the usual:

```
>>>   Local name for paper? (9212062):
```

to which you enter *`distler`*. Two files are retrieved with the message: '`Use "process-paper" on "distler" at your convenience..."`'

The next morning you find the files `distler.1` and `distler.2` stored in your '`atv-download-dir`', thus you enter Emacs and do *`M-x process-paper`*. The response is

```
>>>   Name of stored paper? [without suffix]:
```

You enter `distler` and the **Atavachron** begins to process the paper. Since there are multiple files to this paper, the **Atavachron** checks for **uuencoded** figures. Finding them, it unpacks the figures into the '`atv-uufiles-dir`' where TEX and `dvips` can get at them. Finally it returns the TEX file of the paper in `tex-mode` and a manifest `distler2.rpt` of the uudecoding:

```
 * -\|/- The Atavachron  (1.1) *
This is the output of: sh hep-th/9212062.2
Below should be a manifest of the enclosed files:
-----------------------------------------------
x grapha.eps, 3080 bytes, 7 media blocks.
x graphb.eps, 3080 bytes, 7 media blocks.
x fig.pro, 3089 bytes, 7 media blocks.
```

8 Things To Come

- The first thing that should be done is clean the code a bit. **The Atavachron** grew out of a 20 line *(defun get-paper ())* to teach myself some Lisp, into the present package by Lamarkian evolution. Certainly there are more efficient forms of various functions, and many of the large modules could be broken down into reasonable blocks. Along these lines is a redefinition of the package as a "semi-major mode" of some kind, instead of a non-linear collection of functions.
- Improved error trapping; as *atavachron-mail* comes in, these will be fixed. At present **The Atavachron** will crash if you don't give an expected response, say *[RETURN]* when it really needs a string input, however these seem to just abort without fatalities.
- At present the macro directories at the archives are ignored, meaning you have to get macros via the usual routes of mail or ftp. A nice feature would be macro (ie. *\input* and *\document{}*) checking, by which **The Atavachron** would search the gotten [La]TEX file for all auxillary files and macro packages needed, check the path for these, then report and get the missing macros.
- Finally, more consistent [La]TEX in preprint submissions will make possible the assumption that the preprint will actually pass the texxing stage, thereby making auto-previewing (at the click of the mouse on an abstract), and similar futuristic fantasies possible. This should be provided soon, along with auto-printing, as hook functions for '`atv-auto-tex-command`' which sense the exit of the TEX shell, and determine if it is safe to send the results to the printer.

Concept Index

R

S

T

U

W

Table of Contents