FoilTEX, A LATEX-like System for Typesetting Foils

James L. Hafner
IBM Research Division
Almaden Research Center, K53/802
650 Harry Road
San Jose, CA 95120-6099
Internet: `hafner@almaden.ibm.com`

Abstract

FoilTEX is a LATEX-like system for typesetting foils. Its features include simplicity of use, compatibility with LATEX, large sans serif font as default, extra macros to start foils with bold headings and special mechanisms to control the footer and header. There are also facilities incorporated into FoilTEX, when used with compatible drivers, for one-pass multi-color printing. This article describes the basic features and components of FoilTEX.

The system FoilTEX for typesetting foils with TEX is a LATEX-like system designed with a number of goals in mind. The first was simplicity for the user and, in conjunction with this, compatibility with LATEX. It was part of the design plan that one could take an article written in LATEX, delete large sections of the text and with minor modifications make foils from this. One of the special features that was incorporated into FoilTEX, in conjunction with some output drivers, was the ability to get one-pass, multi-color output. In this article, we will describe the basic features of FoilTEX, how some of the features were implemented and some aspects of the use of color.

We should mention that "foils" is an IBMism for transparencies or slides, in the sense of SLITEX. FoilTEX then is intended for preparation of video materials for talks or other presentations (including poster boards). In testing within IBM, FoilTEX has been used to prepare transparencies for overhead projectors, 35mm slides and even materials for teleconferencing.

The FoilTEX Package

The basic FoilTEX package consists of the following files on top of the basic implementation of LATEX.

```
fltplain.tex  foildoc.tex
fltfonts.tex  sampfoil.tex

foils.sty     colordvi.[tex,sty]
foil17.sty    blackdvi.[tex,sty]
foil20.sty
foil25.sty    foilfont.tex
foil30.sty
```

In the first column, the first two files are the heart of FoilTEX. The file `fltplain.tex` defines the basic set of macros (and includes a request to input `latex.tex`) and the second defines all the fonts used by FoilTEX. These are used to build a format (`.fmt`) file.

We remark that FoilTEX was built in the form of a format file primarily because the basic font set used is very different from either LATEX or SLITEX. For example, unlike LATEX, FoilTEX does not load any font smaller than 12pt and unlike SLITEX, it uses scaled 10pt fonts rather than scaled 8pt fonts. It also has fonts available at larger point sizes than both of these other packages. Consequently, to achieve the same effect simply with style files would force almost doubling the preloaded fonts and the *redefining* of a large collection of macros. It was more efficient and simplier to create a new format file.

The next group of files in the first column are the style files that are used with FoilTEX. `foils.sty` is the basic style file used for all foils. The other `.sty` files are used to change default font sizes. See the sections on the `\documentstyle` command and on fonts for more information about these files. There is no `foils.doc` file because the `.sty` file is already relatively well documented.

In the second column, `foildoc.tex` is the source for the documentation which gives more details about the use and installation of FoilTEX. `sampfoil.tex` is a fairly detailed sample foils document.

When FoilTEX was conceived, a suggestion was made to add one-pass color printing capability. This is more related to drivers, but we developed a device independent (but driver dependent) scheme for doing this. The necessary files are included in this package. The files `colordvi.[tex,sty]` and `blackdvi.[tex,sty]` contain device-independent macros for using color in FoilTEX (or any other TEX). The `.tex` and `.sty` files are identical.

Finally, the file `foilfont.tex` is a FoilTEX file which can be used to test an installation's font availability.

There are other files in the FoilTEX package that we have not listed here. For example, there are system dependent scripts for invoking FoilTEX and some on-line manual or help files. There are also some files for substituting PostScript fonts for the CM fonts or for using some AMS-TEX fonts, `msam`, `msbm` and `eufm`, with automatic scaling. These will not be described here.

The \documentstyle Command

To create a FoilTEX document, the user edits a file very much like a LATEX file. Instead of the standard LATEX options specified in the `\documentstyle` command, they would use

`\documentstyle[`*opts*`]{foils}`

Here, the *opts* list can include any style option that one would normally use (and that doesn't corrupt any macros defined by `foils.sty`).

By default, `foils.sty` loads `foil20.sty` and sets up the normal size fonts at 20pt. Analogous to LATEX's `11pt` and `12pt` style options, FoilTEX has `25pt`, `30pt` and `17pt` options. For example, to make normal size at 25pt the command

`\documentstyle[25pt,`*opts*`]{foils}`

will do the trick. Contrary to LATEX, the default `20pt` *is* an acceptable option, though it is redundant.

The Basic Features

The current version of FoilTEX has the following built-in features. The first is that the basic fonts are in large size, approximately 20pt, (so you do not need to do fontsize changing to get large type). The default font is also sans serif as this look better on foils than serif fonts like roman. We have implemented LATEX's font and font size changing commands, relative to this default. More information about fonts and fontsize changing commands can be found in the section "Fonts and Their Sizes".

In spite of the fact that the basic font is sans serif, the numerals and other symbols from the roman font used in math mode are still in the roman font. Thus mathematics will look exactly the same as in LATEX (only larger) but numerals in text will appear in sans serif.

In addition, almost all LATEX macros are available including automatic referencing and citation, table of contents, footnotes, and itemize (which will probably be very popular for foils). The user is not expected to have to do anything to control font types or size changing, except as might be expected in a typical LATEX document.

The following subsections describe a number of additional macros and features that have been defined to make foilmaking easier. In the appendix is a small sample foil document in FoilTEX source and final output form to demonstrate the simplicity and the beauty of the output. In the final few subsections of this section we mention a few of the differences between FoilTEX and LATEX/SLITEX.

The `\maketitle` command. The use of FoilTEX's `\maketitle` command is the same as for LATEX. That is, it reads the contents of `\title{}`, `\author{}`, etc., and produces a titlepage, actually a title foil. The title itself appears centered and down a small space from the top, in a `\Large` bold sans serif font. The author's name with address and date appear under the title, centered and in the `\normalsize` font. If desired, this can be followed by a (necessarily short) abstract with the word "Abstract" appearing in bold and centered above the text of the abstract. See the appendix for a sample. The footer of the title page will contain some special text. See the section on `\MyLogo` for more details.

The new `\foilhead` macro. The first new macro is called `\foilhead`. Its use is described by

`\foilhead[`*length*`]{`*text*`}`

This macro starts a new page and puts *text* in `\large` bold type at the top center of the new page. After the header, a vertical space of approximately 1.0 inch is added providing an automatic cushion between the header and the body of the foil. This vertical space can be adjusted either up or down by putting in the optional argument a TEX *length.* For example, if the body of the foil should sit closer to the header, the command

`\foilhead[-.5in]{This is the Header}`

would suffice.

This macro should be used to start any new foil, especially if a new heading is needed. If too much text is intended for a single foil, FoilTEX will do its own page break. This could cause some odd vertical spacing since there is a fair amount of stretchability in vertical glue, particularly in list environments.

This can easily be fixed simply by forcing a page break with an empty `\foilhead{}` command.

The new `\MyLogo` and `\Restriction` macros. Another new pair of macros, `\MyLogo` and `\Restriction`, each of which takes a single argument, are used to control the contents of part of the footline. By default, the footline consists of the contents of `\MyLogo` followed by the contents of `\Restriction` both left justified, with the page number right justified[1]. On the main foils, the default font size is `\tiny`. The contents of these macros can be an empty box as well. By default, `\Restriction` is empty and `\MyLogo` is the phrase "– Typeset by FoilTEX –".

The declarations for these macros would normally be placed in the preamble to the document; i.e., before the `\begin{document}` command. However, these macros can be declared or redeclared at any place in the document. They (and all the other commands that control the footer and header) are sensitive to FoilTEX's output routine, which is essentially unchanged from LATEX's. Consequently, care must be taken in their placement to be sure they act on the correct pages. In the preamble or immediately after the `\foilhead` command are best. In addition, there are macro switches that can be used to easily turn on or off the logo, without having to do any redeclarations.

`\MyLogo` is really intended for something idiosyncratic to the speaker or his organization. `\Restriction` was included in case you want to have each foil identified for a particular audience. For example, at IBM, we have the option of displaying the IBM logo and words like "Confidential" or "Internal Use Only". The defaults are set in `foils.sty`.

The other three corners of the page. Since the macros `\Restriction` and `\MyLogo` control the bottom left corner of the page, there are other macros for putting text in the other three corners. These are, not surprisingly,

`\rightfooter{`*text*`}`
`\leftheader{`*text*`}`
`\rightheader{`*text*`}`

They each take one argument, the text you want to place in the associated corner of the page. These can also be redeclared within the document with the appropriate attention paid to the output routine.

By default the headers are empty and the lower right footer is just the page number:

```
\rightheader{}
\leftheader{}
\rightfooter{\quad{\sf\thepage}}
```

except on the title page where they are all suppressed. You can easily suppress page numbering by declaring `\rightfooter{}`.

We did not add macros for centering text in the header or footer because we felt this simply add unnecessary clutter to the foils.

New `Theorem` and `Proof` environments. There are a number of (both starred and unstarred) `\newtheorem` environments built in. These are for `Theorem`, `Lemma`, `Corollary`, `Proposition` and `Definition`. Note the uppercased first letter (to avoid possible collisions with user-defined environments of this type). Each must begin and end with `\begin` and `\end` commands as usual. Their text begins with a bold sans serif label like **Theorem** and the content of each is typeset in *slanted sans serif*. The unstarred forms are sequentially numbered and support automatic referencing. The starred forms suppress the numbering and referencing.

There is a `Proof` environment which opens with the word **Proof** and ends with a □. The contents are printed in the normal font.

Mathematics in bold typeface. FoilTEX uses a modified form of LATEX's font definitions for bold typefaced mathematics. In particular, a `\bf` command in math mode will switch to a bold sans serif font (probably not desirable in mathematics since the rest of mathematics is in serifed fonts). In FoilTEX, LATEX's `\boldmath` command has been modified also. Here, characters from the roman font are emboldened by switching to the bold roman font (`cmbx` family), not the bold math symbol font as in LATEX.

To make using bold mathematics easier some new macros have been defined. The first is

`\bm{`*formula*`}`

This takes its argument (within mathematics mode) and replaces it with the emboldened version. Unfortunately, it acts a little funny on characters like summation signs and in super- or subscripts since it reverts to TEX's text style (style T) first. Consequently, this command should be used primarily on individual characters or small parts of formulas.

The second method for getting bold mathematics is a pair of environments

`\begin{boldequation}`
formula (*number*)
`\end{boldequation}`

[1] For the title foil, there is no page number; `\MyLogo` and `\Restesizion` are centered and appear in `\footnotesize` font.

```
\begin{boldequation*}
formula
\end{boldequation*}
```

They both set *formula* in bold (except for super- and subscripts). The unstarred form has automatic referencing and is numbered; the starred form inhibits the numbering and referencing.

The limitation on the super- and subscripts not appearing in bold face is strictly to limit the number of fonts loaded by FoilTEX. It was felt this bold math feature would have limited use and so it is not fully supported. It could easily be extended if there is sufficient demand.

List environments. The vertical spacing of items in list environments is controlled by exactly the same mechanisms as in LATEX. We have set the defaults, however, so that at the highest level there is a fair amount of vertical space, but at lower levels this shrinks to nothing. This seemed to produce the best and most pleasing results, at least to the author's personal taste.

This is *not* LATEX. At the heart of FoilTEX is a format file. Consequently, there is usually a system dependent exec (or script or batch program) which calls the main TEX program with the necessary FoilTEX format file, `fltplain.fmt`. Testing showed that users (especially hard-core LATEX users) tended to run LATEX instead out of habit. As a result, a special feature was implemented in which, if LATEX is called on a FoilTEX file, the user is prompted with a warning and given a choice of continuing with some unpredictable consequences or aborting.

Differences with LATEX. One simple difference is that the LATEX command `\em` switches from any unslanted font to *slanted sans serif* and from any slanted font to unslanted sans serif, not to *text italics* and roman, respectively.

Unlike TEX/LATEX, numerals in FoilTEX look different when they are in ordinary text from when they are in math-mode. This means that `12345` in text will print as 12345 and `$12345$` prints as 12345.

Hyphenation has been eliminated from FoilTEX. It was felt that this improves readability. Because of this, FoilTEX might have problems fitting things nicely on a line. Overfull and underfull `\hbox`es might occur more often than in LATEX but the tolerances are set to reduce their frequency. Since the fonts are so large, FoilTEX can be more tolerant of white space without being unaesthetic. If they do occur with no obvious fix, a discretionary hyphen strategically placed can resolve the problem.

Some user's felt that `\raggedright` is preferable for foils. It was decided not to make this the default, but to leave this to the user's discretion.

The following features of LATEX have been disabled in FoilTEX because they seemed unnecessary. They can easily be added if there is sufficient demand: lists of figures, indexing, glossary.

Major differences with SLITEX. There are many differences between SLITEX and FoilTEX. The most glaring feature not supported in FoilTEX is invisible fonts. Also, as indicated in Table 1, `\rm` and `\sf` do what you expect, that is switch to roman and sans serif, respectively. In SLITEX, they reverse roles.

The basic features: future versions. A possible new feature might be an automatically-generated "Summary of the Talk", akin to a table of contents, where the user could tag some of the `\foilhead` macros and have them collected in a special foil following the title foil.

Fonts and Their Sizes

As noted earlier, the default font at `\normalsize` is a sans serif font at size 20pt, unless one of the `[17pt]`, `[25pt]`, or `[30pt]` options have been declared in the `\documentstyle` command. Table 1 shows the control sequences for other accessible text fonts and the name of the font in a sample of its type. These control sequences give the font at the current size. Font size changing commands for each of the normal point size options are described by Table 2. Note that `\bf` and `\sl` yield sans serif fonts, not the usual variations on roman.

Mathematics is also automatically displayed at normal size unless magnified by a size changing declaration. Table 3 describes the font point sizes for TEX's mathematics styles at each of the normal point size options. FoilTEX loads or knows about enough fonts, particularly symbol fonts, that there should never be a discrepancy between the size of text and mathematics at any of the different sizes (unlike LATEX where some fonts at `xxvpt` are actually only 20pt fonts).

Since many of FoilTEX's fonts are not in the standard distribution, and so not available on most systems, the installer will probably have to run METAFONT to generate the necessary files. The file `foilfont.tex` requires a sample of every preloaded or load-on-demand font and so can be used to test an installation's font availability. (Some drivers, like Tom Rokicki's `dvips` program, will generate all the missing fonts just by trying to process this file.)

The LATEX circle and line fonts have been preloaded at `magstep4` so that small LATEX pictures should scale naturally to a foil.

Table 1: Available fonts and their names.

command	font names
`\sf`	Sans Serif
`\it`	*Text Italic*
`\sl`	*Slanted Sans Serif*
`\bf`	**Bold Sans Serif**
`\tt`	`Typewriter`
`\rm`	Roman
`\sc`	Small Caps

Table 2: Type sizes for FoilTEX size-changing commands for the different documentstyle options.

`size/doc-opt`	20pt	17pt	25pt	30pt
`\tiny`	12pt	12pt	12pt	14pt
`\scriptsize`	12pt	12pt	14pt	17pt
`\footnotesize`	14pt	12pt	17pt	20pt
`\small`	17pt	14pt	20pt	25pt
`\normalsize`	20pt	17pt	25pt	30pt
`\large`	25pt	20pt	30pt	36pt
`\Large`	30pt	25pt	36pt	43pt
`\LARGE`	36pt	30pt	43pt	43pt
`\huge`	43pt	36pt	43pt	43pt
`\Huge`	43pt	43pt	43pt	43pt

Table 3: Mathematics type styles and their point sizes at `\normalsize` for the different documentstyle options.

`style/doc-opt`	20pt	17pt	25pt	30pt
D, D',T,T'	20pt	17pt	25pt	30pt
S,S'	14pt	12pt	17pt	20pt
SS,SS'	12pt	12pt	14pt	17pt

Making Color Foils

This feature is still in the development stage and is *very* device-driver dependent. This last problem is regrettable because it severely limits portability, but this cannot be helped at the moment because TEX was not designed with color in mind. In any case, what we have incorporated into FoilTEX are sets of macros which are device independent. They of course use TEX's `\special` command, but do not use any syntax that is dependent on the physical device or output data stream. In this way, it is hoped that more drivers can take advantage of the same set of macros for color printing or display. The only drivers we are aware of that fully support the macros we provide here are Tom Rokicki's `dvips` program (version 5.478 or later) and TEXview on the NeXT. In the next few sections we will describe this implementation of color.

One other comment: these color macros are not necessarily limited to FoilTEX but can run under any other TEX. However there are subtleties about how footlines, headlines, and other special regions of the text will handle the color changes. For very successful use, some macros need to be modified with implied color. We have not tested this explicitly but foresee no special difficulties (provided the driver operates compatibly). The relevant macros in FoilTEX already have these features built in. For example, the footer and header macros wrap everything in `\Black` so colors in the text that cross a page boundary will not affect these regions.

The style file `colordvi.sty` and `dvips`. As we see it, the "right" way to use color in FoilTEX (or other TEX) files is with the `colordvi.sty` file. These macros can be included in FoilTEX, for example, by simply adding `colordvi` to the `\documentstyle` command:

```
\documentstyle[colordvi]{foils}
```

(In TEXs that don't have documentstyles, the appropriate `\input` command will work as well.) This file defines all the color macros using TEX's `\special` command. The internal syntax has forms like

```
\special{color push Red}
```
Nested Red text.
```
\special{color pop}

\special{color Blue}
```
Base color now Blue.

depending on whether this is nested color or global color changes (see the section on color macros). Consequently, the driver must be able to recognize the `\special` keyword `color` and process something to the output file that signals the color change, tracking the nesting level, etc. It is also important that the driver be able to track the color state across page boundaries or any other boundary where the output state can change. The driver should ideally also produce output where each page has self-contained color state information, so that pages can be printed in different orders, or by selected pages.

An additional macro in `colordvi.ps` can be used to set the background color. For this macro, the driver needs to recognize the `\special` keyword `background` and must be able to set the specified background color on the *current* page and remember that color until changed explicitly.

Furthermore, the actual color parameters need to be set in some device dependent way, say with a special prolog file that defines the color `Red` in terms the output device can understand, and in such a way that the parameters are tuned to the particular device. (Each output mechanism uses different color renditions which makes it very difficult to set a universal standard.)

For Rokicki's `dvips`, we have done all of the above. There is a color prolog file which `dvips` includes in its header list whenever it encounters the keywords `color` or `background`. The particular one we wrote has the color parameters tuned to the Tektronix PHASER printer. We added code to `dvips` to track the color history and states during the prescan. In this way, it can initialize the color state on each page of the output file during the final scan. (We should mention that our original code for `dvips` and our original set of macros where greatly improved by Tom Rokicki. We are grateful for his help and for including these features in his driver.)

Finally, we remark that we have used the names `color` and `black` suffixed by `dvi` so as not to conflict with Leslie Lamport's `color.sty` which has become somewhat wide-spread. We chose the suffix `dvi` because it reflects the device independent nature of the macros.

Printing in black/white, with or without `blackdvi.sty`. A FoilTEX (or other TEX) document written with color macros can be printed in black and white in two ways. If the device is a black and white version of a color device (e.g., display or PostScript printer) then it should print in corresponding grey-levels. This is useful since in this way one can get a rough idea of where the colors are changing without using expensive color printing devices. The second option is to replace the call to input `colordvi` with `blackdvi`. This "black" style file turns all the color macros into no-ops, and so will produce normal black/white printing without the user having to ferret out the color commands. Also, most device drivers will simply ignore the color commands and so print in normal black and white.

The color macros: user's viewpoint. There are two kinds of color macros, ones for local color changes to, say, a few words or even one character and one for global color changes. All the color names use a mixed case scheme. There are 68 predefined colors, with names taken primarily from the Crayola 64 crayon box, and one pair of macros for the user to set his own color pattern. More on this extra feature later. Users can browse the file `colordvi.sty` for a list of the predefined color names.

A local color command is in the form

```
\ColorName{this will print in color}
```

As this example shows, this type of command takes one argument which is the text that is to print in the selected color. This can be used for nested color changes since it should restore the original color state when it completes. For example, suppose a user was writing in green and wanted to switch temporarily to red, then blue, back to red and restore green. Here is one way to do this:

```
This text is green but here we are
\Red{switching to red,
\Blue{nesting blue} recovering the
red} and back to original green.
```

In principle the nesting level is unlimited, but it is not advisable to nest too deep lest one loose track of the root color or exceeds the driver's capacity.

The global color command has the form

```
\textColorName
```

This macro takes no arguments and immediately changes the default color from that point on to the specified color. This of course can be overridden globally by another such command or locally by local color commands. For example, expanding on the example above, we might have

```
\textGreen
This text is green but here we are
\Red{switching to red,
\Blue{nesting blue,} recovering the
red} and back to original green.
\textCyan
The text from here on will be cyan
unless \Yellow{locally changed
to yellow}.  Now we are back to cyan.
```

The color commands will even work in math mode and across math mode boundaries. This means that a color state going into math mode will force the mathematics to be set in that color as well. More importantly however, in alignment environments like `tabular` and `eqnarray`, local color commands *cannot* extend beyond the alignment characters.

Because local color commands respect only some environment and deliminator changes besides their own, care must be taken in setting their scope. It is best not to have then stretch too far.

User definable colors. There are two ways for the user to specify colors not already defined. For local changes, there is the command `\Color` which takes two arguments. The first argument is a quadruple of numbers between zero and one and specifies the intensity of cyan, magenta, yellow and black (CMYK) in that order. The second argument is the text that should appear in the given color. For example, if a user wants the words "this color is pretty" to appear in a color which is 50% cyan, 85% magenta, 40% yellow and 20% black, they would use the command

```
\Color{.5 .85 .4 .2}{this color is
pretty}
```

For global color changes, there is a command `\textColor` which takes one argument, the CMYK quadruple of relative color intensities. For example, to make the default color to be as above, then the command

```
\textColor{.5 .85 .4 .2}
The text from now on will be this
pretty color.
```

will suffice.

If the intended output device does not treat color in CMYK terms, then the device *driver* should convert these values to the device dependent parameters, e.g., RGB.

Setting the background color. There is an additional macro for setting the background color. It takes a single argument, which can either be one of the predefined color names or a quadruple of CMYK values. For example,

```
\background{SkyBlue}
```

or

```
\background{.1 .2 .3 .1}
```

These should appear somewhere on the page (preferably near the beginning) where the background color is to change. The background should stay this color until explicitly changed by another such command. It should be remembered that the placement of this is sensitive to the output routine.

Default color regions in FoilTEX. When `colordvi.sty` is loaded by FoilTEX, the default text color is black. The footline and headline get defaulted color values as well. So, the contents of `\MyLogo`, `\Restriction`, `\rightfooter` (the page number by default), `\leftheader` and `\rightheader` will all appear in Black.

Overriding the color selection for any of these regions of the text can be done by using *local* color commands in their declarations. For example, a very sensitive talk requiring the words "Need-To-Know" in red would use the declaration

```
\Restriction{\Red{Need-To-Know}}
```

Installing \FoilTeX

Because installations of TEX/LATEX differ so much from system to system and even within systems, the installation instructions here are mostly just an outline of the general procedure and system requirements.

To install FoilTEX, the requirements are:

- the TEX program, including a version of `INITEX`;
- LATEX, any version after Nov. 89 (The concern here is access to the font metrics for `lcircle10` and `lcirclew10`, as opposed to `circle10` and `circlew10`. The installer can modify `fltfonts.tex` if necessary.); and
- the METAFONT program and related tools to generate the necessary fonts (FoilTEX uses essentially all the basic CM fonts and some additional LATEX fonts but at magnifications equivalent to `\magstep6, 7` and `8`.).

The installation of FoilTEX then involves

- generating a `fltplain.fmt` format file by running `initex` (and installing this in the appropriate location for the system);
- testing font availability by running TEX with this format against `foilfont.tex` and trying to print this;
- generating all the missing fonts; and
- installing the various style, macro, script and on-line help files in the appropriate location for the system.

Acknowledgements and Requests

We would like to thank and acknowledge the following people in IBM for their great assistance in helping to put FoilTEX together: Katherin Hitchcock, Myron Flickner, Ekkehard Blanz, Melanie Fulgham, Peter Haas, Rocky Bernstein and the many users who contributed their constructive comments on the early test versions within IBM.

A special thanks goes to Tom Rokicki for implementing our color setup in his driver and another to Sheri Gish of IBM for asking the right (or was it wrong?) question that got this project started.

FoilTEX is intended to be easy to use, useful and to produce beautiful foils. Consequently, the author welcomes any comments or suggestions.

Sample Foils

Below is source for a short two page sample foil that demonstrates most of the features of FoilTEX, followed by a facsimile of the output from this source.

```
%%%%%% First we load the correct style file
\documentstyle{foils}
%%%%%% This first section is for a title page; it is typical LaTeX
\title{Rock protocols for binary Quarries}
%
\author{Fred Flintstone\\
Rock Quarry Research Center}
\date{\today}
%%%%%% This next command controls part of the footline.
%%%%%% Note the ‘‘FoilTeX’’ logo will print automatically.
%\yourlogo{-- Typeset by \FoilTeX\ --}
\Restriction{TUG Use Only}
%
\begin{document}
\maketitle                          %
\begin{abstract} This is where an abstract might go.\end{abstract}
%%%%%% This next command starts a new foil with header.
\foilhead{Variability of Rock Quality}
%
What can we prove using only marble rocks?
%%%%%% Itemize, mathematics, auto-referencing and footnotes are built-in.
\begin{itemize}
\item $\Omega(t^2)$ rocks needed \cite{rocky}\footnote{What’s that?}.
\item Worst case structure uses
      \begin{equation} \label{equation}
               O(n+t\sqrt{t})
      \end{equation}
\end{itemize}
%%%%%% Here is a sample theorem with proof.
\begin{Theorem} Everything you know about rocks is false.
\end{Theorem}
%
\begin{Proof} The proof is obvious from equation (\ref{equation}).
\end{Proof}
%%%%%% Bibliographies work even with BibTeX.
\begin{thebibliography}{99}
%
\bibitem{rocky} Rocky and Bullwinkle, Open problems, in {\sl Mr.
Know-it-all’s Rock Encyclopedia}.
%
\end{thebibliography}
\end{document}
```

Rock protocols for binary Quarries

Fred Flintstone
Rock Quarry Research Center

October 8, 2001

Abstract

This is where an abstract might go.

– Typeset by FoilTEX – TUG Use Only

Variability of Rock Quality

What can we prove using only marble rocks?

- $\Omega(t^2)$ rocks needed [1][1].
- Worst case structure uses

$$O(n + t\sqrt{t}) \tag{1}$$

Theorem 1. *Everything you know about rocks is false.*

Proof. The proof is obvious from equation (1). □

References

[1] Rocky and Bullwinkle, Open Problems, in *Mr. Know-it-all's Rock Encyclopedia*.

[1] What's that?

– Typeset by FoilTeX – TUG Use Only 1