_1996 _manolios_moore_2000 _2000

An Example of a Paper with a Rather Large Title-to-Content Ratio

Rob van Glabbeek
NICTA
Sydney, Australia
School of Computer Science and Engineering
University of New South Wales*
Sydney, Australia
rvg@cs.stanford.edu

Co Author   Yet S. Else
Stanford Univeristy
California, USA
is@gmail.com   somebody@else.org

This is a sentence in the abstract. This is another sentence in the abstract. This is yet another sentence in the abstract. This is the final sentence in the abstract.

1 Introduction

Filesystems provide a convenient interface for the named storage of data, and as such are ubiquitous in application programs. Filesystem bugs expose users to the risks of data breach, corruption, loss; thus, verifying their correctness is essential. In this work, we present an approach towards constructing executable specifications of existing filesystems and verifying their functional properties in a theorem proving environment. We detail an application of this approach to the FAT32 filesystem.

We also detail the methodology used to build up this type of executable specification through a series of models which incrementally add features of the target filesystem. This methodology has the benefit of allowing the verification effort to start from simple models which encapsulate features common to many filesystems and are thus suitable for re-use.

2 Related work

Filesystem verification research has largely followed a pattern of synthesising a new filesystem based on a specification chosen for its ease in proving properties of interest, rather than similarity to an existing filesystem. Our work, in contrast, follows the FAT32 specification closely. In spirit, our work is closer to previous work which uses interactive theorem provers and explores deep functional properties than to efforts which use non-interactive theorem provers such as Z3 to produce fully automated proofs of simpler properties.

2.1 Interactive theorem provers

An early effort in the filesystem verification domain was by Bevier and Cohen [5], who specified the Synergy filesystem and created an executable model of the same in ACL2 [**?**], down to the level of processes and file descriptors. On the proof front, they certified their model to preserve well-formedness of their data structures through their various file operations; however, they did not attempt to prove, for instance, read-over-write properties or crash consistency. Later, Klein et al with the SeL4 project [16]

*A fine university.

Submitted to:
SOS 2007

used Isabelle/HOL [21] to verify a microkernel; while their design abstracted away file operations in order to keep their trusted computing base small, it did serve as a precursor to their more recent COGENT project [2]. Here the authors built a ”verified compiler” of sorts, generating C-language code from specifications in their domain-specific in a manner guaranteed to avoid many common filesystem bugs. Elsewhere, the SibylFS project [22], again using Isabelle/HOL, provided an executable specification for filesystems at a level of abstraction that could function across multiple operating systems including OSX and Unix. The Coq prover [4] has also been used, for instance, for FSCQ [6], a state-of-the art filesystem which was built to have high performance and formally verified crash consistency properties.

2.2 Non-interactive theorem provers

Non-interactive theorem provers such as Z3 [7] have also been used; Hyperkernel [20] is a recent effort which focusses on simplifying the xv6 microkernel until the point that Z3 can verify it with its SMT solving techniques. However, towards this end, all system calls in Hyperkernel are replaced with analogs which can terminate in constant time; while this approach is theoretically sound, it increases the chances of discrepancies between the model and the implementation which may diminish the utility of the proofs or even render them moot. A stronger effort in the same domain is Yggdrasil [24], which focusses on verifying filesystems with the use of Z3. While the authors make substantial progress in terms of the number of filesystem calls they support and the crash consistency guarantees they provide, they are subject to the limits of SMT solving which prevent them from modelling essential filesystem features such as extents, which are central to many filesystems including FAT32.

3 Ancillary files

Authors may upload ancillary files to be linked alongside their paper. These can for instance contain raw data for tables and plots in the article, or program code. Ancillary files are included with an EPTCS submission by placing them in a directory `anc` next to the main latex file. See also `https://arxiv.org/help/ancillary_files`. Please add a file README in the directory `anc`, explaining the nature of the ancillary files, as in `http://eptcs.org/paper.cgi?226.21`.

4 Prefaces

Volume editors may create prefaces using this very template, with `\title{Preface}` and `\author{}`.

5 Bibliography

We request that you use `\bibliographystyle{eptcs}` [**?**], or one of its variants eptcsalpha, eptcsini or eptcsalphaini [**?**]. Compared to the original LaTeX `\bibliographystyle{plain}`, it ignores the field `month`, and uses the extra bibtex fields `eid`, `doi`, `ee` and `url`. The first is for electronic identifiers (typically the number n indicating the n^{th} paper in an issue) of papers in electronic journals that do not use page numbers. The other three are to refer, with life links, to electronic incarnations of the paper.

DOIs Almost all publishers use digital object identifiers (DOIs) as a persistent way to locate electronic publications. Prefixing the DOI of any paper with `http://dx.doi.org/` yields a URI that resolves to the current location (URL) of the response page[1] of that paper. When the location of the response page changes (for instance through a merge of publishers), the DOI of the paper remains the same and (through an update by the publisher) the corresponding URI will then resolve to the new location. For that reason a reference ought to contain the DOI of a paper, with a life link to the corresponding URI, rather than a direct reference or link to the current URL of publisher's response page. This is the rôle of the bibtex field `doi`. **EPTCS requires the inclusion of a DOI in each cited paper, when available.**

DOIs of papers can often be found through `http://www.crossref.org/guestquery`;[2] the second method *Search on article title*, only using the **surname** of the first-listed author, works best. Other places to find DOIs are DBLP and the response pages for cited papers (maintained by their publishers).

The bibtex fields `ee` and `url` Often an official publication is only available against payment, but as a courtesy to readers that do not wish to pay, the authors also make the paper available free of charge at a repository such as `arXiv.org`. In such a case it is recommended to also refer and link to the URL of the response page of the paper in such a repository. This can be done using the bibtex fields `ee` or `url`, which are treated as synonyms. These fields should **not** be used to duplicate information that is already provided through the DOI of the paper. You can find archival-quality URL's for most recently published papers in DBLP—they are in the bibtex-field `ee`—but please suppress repetition of DOI information. In fact, it is often useful to check your references against DBLP records anyway, or just find them there in the first place.

Typesetting DOIs and URLs When using LaTeX rather than `pdflatex` to typeset your paper, by default no linebreaking within long URLs is allowed. This leads often to very ugly output, that moreover is different from the output generated when using `pdflatex`. This problem is repaired when invoking `\usepackage{breakurl}`: it allows linebreaking within links and yield the same output as obtained by default with `pdflatex`. When invoking `pdflatex`, the package `breakurl` is ignored.

Please avoid using `\usepackage{doi}`, or `\newcommand{\doi}`. If you really need to redefine the command `doi` use `\providecommand{\doi}`.

The package `\usepackage{underscore}` is recommended to deal with underscores in DOIs. This is not needed when using `\usepackage{breakurl}` and not `pdflatex`.

References to papers in the same EPTCS volume To refer to another paper in the same volume as your own contribution, use a bibtex entry with

```
series = {\thisvolume{5}},
```

where 5 is the submission number of the paper you want to cite. You may need to contact the author, volume editors or EPTCS staff to find that submission number; it becomes known (and unchangeable) as soon as the cited paper is first uploaded at EPTCS. Furthermore, omit the fields `publisher` and `volume`. Then in your main paper you put something like:

```
\providecommand{\thisvolume}[1]{this volume of EPTCS, Open Publishing Association}
```

This acts as a placeholder macro-expansion until EPTCS software adds something like

[1]Nowadays, papers that are published electronically tend to have a *response page* that lists the title, authors and abstract of the paper, and links to the actual manifestations of the paper (e.g. as `dvi`- or `pdf`-file). Sometimes publishers charge money to access the paper itself, but the response page is always freely available.

[2]For papers that will appear in EPTCS and use `\bibliographystyle{eptcs}` there is no need to find DOIs on this website, as EPTCS will look them up for you automatically upon submission of a first version of your paper; these DOIs can then be incorporated in the final version, together with the remaining DOIs that need to found at DBLP or publisher's webpages.

`\newcommand{\thisvolume}[1]{{\eptcs} 157\opa, pp 45--56, doi:...},`
where the relevant numbers are pulled out of the database at publication time. Here the newcommand wins from the providecommand, and `\eptcs` resp. `\opa` expand to
`\sl Electronic Proceedings in Theoretical Computer Science` and
`, Open Publishing Association` .
Hence putting `\def\opa{}` in your paper suppresses the addition of a publisher upon expansion of the citation by EPTCS. An optional argument like

`series = {\thisvolume{5}[EPTCS]},`

overwrites the value of `\eptcs`.

References

[1] Martín Abadi & Leslie Lamport (1991): *The existence of refinement mappings*. Theoretical Computer Science 82(2), pp. 253–284.

[2] Sidney Amani, Alex Hixon, Zilin Chen, Christine Rizkallah, Peter Chubb, Liam O'Connor, Joel Beeren, Yutaka Nagashima, Japheth Lim, Thomas Sewell et al. (2016): *Cogent: Verifying high-assurance file system implementations*. In: ACM SIGPLAN Notices, 51, ACM, pp. 175–188.

[3] Konstantine Arkoudas, Karen Zee, Viktor Kuncak & Martin Rinard (2004): *Verifying a file system implementation*. In: International Conference on Formal Engineering Methods, Springer, pp. 373–390.

[4] Yves Bertot & Pierre Castéran (2013): *Interactive theorem proving and program development: CoqArt: the calculus of inductive constructions*. Springer Science & Business Media.

[5] William R Bevier & Richard M Cohen (1996): *An executable model of the Synergy file system*. Technical Report, Technical Report 121, Computational Logic, Inc.

[6] Haogang Chen, Daniel Ziegler, Tej Chajed, Adam Chlipala, M. Frans Kaashoek & Nickolai Zeldovich (2016): *Using Crash Hoare Logic for Certifying the FSCQ File System*. In Gulati & Weatherspoon [11]. Available at `https://www.usenix.org/conference/atc16/technical-sessions/presentation/chen_haogang`.

[7] Leonardo De Moura & Nikolaj Bjørner (2008): *Z3: An efficient SMT solver*. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, pp. 337–340.

[8] Philippa Gardner, Gian Ntzik & Adam Wright (2014): *Local reasoning for the POSIX file system*. In: European Symposium on Programming Languages and Systems, Springer, pp. 169–188.

[9] Sanjay Ghemawat, Howard Gobioff & Shun-Tak Leung (2003): *The Google file system*. In: ACM SIGOPS operating systems review, 37, ACM, pp. 29–43.

[10] Shilpi Goel, Warren A Hunt & Matt Kaufmann (2017): *Engineering a formal, executable x86 ISA simulator for software verification*. In: Provably Correct Systems, Springer, pp. 173–209.

[11] Ajay Gulati & Hakim Weatherspoon, editors (2016): *2016 USENIX Annual Technical Conference, USENIX ATC 2016, Denver, CO, USA, June 22-24, 2016*. USENIX Association. Available at `https://www.usenix.org/conference/atc16`.

[12] Michael K Johnson (1996): *A tour of the Linux VFS*. Available at `http://www.tldp.org/LDP/khg/HyperNews/get/fs/vfstour.html`.

[13] Matt Kaufmann, Panagiotis Manolios & J. Strother Moore (2000): *Computer-aided reasoning: an approach*. Kluwer Academic Publishers.

[14] Matt Kaufmann & J Strother Moore (1996): *ACL2: An industrial strength version of Nqthm*. In: Computer Assurance, 1996. COMPASS'96, Systems Integrity. Software Safety. Process Security. Proceedings of the Eleventh Annual Conference on, IEEE, pp. 23–34.

[15] Gabriele Keller, Toby Murray, Sidney Amani, Liam O'Connor, Zilin Chen, Leonid Ryzhyk, Gerwin Klein & Gernot Heiser (2014): *File systems deserve verification too!* *ACM SIGOPS Operating Systems Review* 48(1), pp. 58–64.

[16] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish et al. (2009): *seL4: Formal verification of an OS kernel*. In: *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, ACM, pp. 207–220.

[17] Leslie Lamport (1993): *Verification and specification of concurrent programs*. In: *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, Springer, pp. 347–374.

[18] Microsoft (2000): *Microsoft Extensible Firmware Initiative FAT32 File System Specification*. Available at `www.microsoft.com/hwdev/download/hardware/fatgen103.pdf`.

[19] Toby Murray, Daniel Matichuk, Matthew Brassil, Peter Gammie, Timothy Bourke, Sean Seefried, Corey Lewis, Xin Gao & Gerwin Klein (2013): *seL4: from general purpose to a proof of information flow enforcement*. In: *Security and Privacy (SP), 2013 IEEE Symposium on*, IEEE, pp. 415–429.

[20] Luke Nelson, Helgi Sigurbjarnarson, Kaiyuan Zhang, Dylan Johnson, James Bornholt, Emina Torlak & Xi Wang (2017): *Hyperkernel: Push-Button Verification of an OS Kernel*. In: *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, ACM, New York, NY, USA, pp. 252–269, doi:10.1145/3132747.3132748. Available at `http://doi.acm.org/10.1145/3132747.3132748`.

[21] Tobias Nipkow, Lawrence C Paulson & Markus Wenzel (2002): *Isabelle/HOL: a proof assistant for higher-order logic*. 2283, Springer Science & Business Media.

[22] Tom Ridge, David Sheets, Thomas Tuerk, Andrea Giugliano, Anil Madhavapeddy & Peter Sewell (2015): *SibylFS: formal specification and oracle-based testing for POSIX and real-world file systems*. In: *Proceedings of the 25th Symposium on Operating Systems Principles*, ACM, pp. 38–53.

[23] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh & Bob Lyon (1985): *Design and implementation of the Sun network filesystem*. In: *Proceedings of the Summer USENIX conference*, pp. 119–130.

[24] Helgi Sigurbjarnarson, James Bornholt, Emina Torlak & Xi Wang (2016): *Push-Button Verification of File Systems via Crash Refinement*. In: *OSDI*, 16, pp. 1–16.