Predicting SMT solver performance for software verification

Andrew HEALY

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

MAYNOOTH UNIVERSITY

Department of Computer Science
Faculty of Science and Engineering

October 2016

Head of Department:	*Supervisors:*
Dr. Adam WINSTANLEY	Dr. James F. POWER
	Dr. Rosemary MONAHAN

Contents

MAYNOOTH UNIVERSITY

Abstract

Faculty of Science and Engineering
Department of Computer Science

Master of Science

Predicting SMT solver performance for software verification

by Andrew HEALY

The approach Why3 takes to interfacing with a wide variety of interactive and automatic theorem provers works well: it is designed to overcome limitations on what can be proved by a system which relies on a single tightly-integrated solver. In common with other systems, however, the degree to which proof obligations (or "goals") are proved depends as much on the SMT solver as the properties of the goal itself. In this work, we present a method to use syntactic analysis to characterise goals and predict the most appropriate solver via machine-learning techniques.

Combining solvers in this way - a *portfolio-solving* approach - maximises the number of goals which can be proved. The driver-based architecture of Why3 presents a unique opportunity to use a portfolio of SMT solvers for software verification. The intelligent scheduling of solvers minimises the time it takes to prove these goals by avoiding solvers which return *Timeout* and *Unknown* responses. We assess the suitability of a number of machine-learning algorithms for this scheduling task.

The performance of our tool Where4 is evaluated on a large dataset of proof obligations. We compare Where4 to a range of SMT solvers and theoretical scheduling strategies. We find that Where4 can prove a greater number of goals in a shorter amount of time than any single solver. Furthermore, Where4 can integrate into a Why3 user's normal workflow - simplifying and automating the use of SMT solvers for software verification.

Acknowledgements

The acknowledgments and the people to thank go here, don’t forget to include your project adviser…

List of Abbreviations

SMT	**S**atisfiability **M**odulo **T**heories
WSF	**W**hat (it) **S**tands **F**or

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Software Verification Background

1.1.1 Software Verification Systems as Composed of Modular Components

1.1.2 The Why3 system

[cesare:xiang]

Front End: Why3 as an intermediary language

Back End: Driver-based interface to APT, ITP and SMT tools

1.2 An overview of various SMT solvers and their capabilities

1.3 The problem we want to solve: the efficient allocation of SMT-solving resources for software verification tasks

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just *lshort*. You can download the latest version (as it is occasionally updated) from here: http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

It is also available in several other languages. Find yours from the list on this page: http://www.ctan.org/tex-archive/info/lshort/

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents, or having a close look at several templates on:
http://www.LaTeXTemplates.com
Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

If you are writing a technical or mathematical thesis, then you may want to read the document by the AMS (American Mathematical Society) called, "A Short Math Guide for LaTeX". It can be found online here: http://www.ams.org/tex/amslatex.html under the "Additional Documentation" section towards the bottom of the page.

There are a multitude of mathematical symbols available for LaTeX and it would take a great effort to learn the commands for them all. The most common ones you are likely to use are shown on this page: http://www.sunilpatel.co.uk/latex-type/latex-math-symbols/

You can use this page as a reference or crib sheet, the symbols are rendered as large, high quality images so you can quickly find the LaTeX command for the symbol you need.

The LaTeX distribution is available for many systems including Windows, Linux and Mac OS X. The package for OS X is called MacTeX and it contains all the applications you need – bundled together and pre-customized – for a fully working LaTeX environment and work flow.

MacTeX includes a custom dedicated LaTeX editor called TeXShop for writing your '**`.tex`**' files and BibDesk: a program to manage your references and create your bibliography section just as easily as managing songs and creating playlists in iTunes.

If you are familiar with LaTeX, then you should explore the directory structure of the template and then proceed to place your own information into the *THESIS INFORMATION* block of the **`main.tex`** file. You can then modify the rest of this file to your unique specifications based on your degree/university. Section **??** on page **??** will help you do this. Make sure you also read section **??** about thesis conventions to get the most out of this template.

If you are new to LaTeX it is recommended that you carry on reading through the rest of the information in this document.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change to bring the template in line with your institution's recommendations. These modifications will need to be done on the **`MastersDoctoralThesis.cls`** file.

This LaTeX Thesis Template is originally based and created around a LaTeX style file created by Steve R. Gunn from the University of Southampton (UK), department of Electronics and Computer Science. You can find his original thesis style file at his site, here: `http://www.ecs.soton.ac.uk/~srg/softwaretools/document/templates/`

Steve's **`ecsthesis.cls`** was then taken by Sunil Patel who modified it by creating a skeleton framework and folder structure to place the thesis files in. The resulting template can be found on Sunil's site here: `http://www.sunilpatel.co.uk/thesis-template`

This template comes as a single zip file that expands out to several files and folders. The folder names are mostly self-explanatory:

Appendices – this is the folder where you put the appendices. Each appendix should go into its own separate **`.tex`** file. An example and template are included in the directory.

Chapters – this is the folder where you put the thesis chapters. A thesis usually has about six chapters, though there is no hard rule on this. Each chapter should go in its own separate **`.tex`** file and they can be split as:

- Chapter 1: Introduction to the thesis topic
- Chapter 2: Background information and theory
- Chapter 3: (Laboratory) experimental setup
- Chapter 4: Details of experiment 1
- Chapter 5: Details of experiment 2

- Chapter 6: Discussion of the experimental results
- Chapter 7: Conclusion and future directions

This chapter layout is specialised for the experimental sciences.

Figures – this folder contains all figures for the thesis. These are the final images that will go into the thesis document.

Included are also several files, most of them are plain text and you can see their contents in a text editor. After initial compilation, you will see that more auxiliary files are created by LaTeX or BibTeX and which you don't need to delete or worry about:

example.bib – this is an important file that contains all the bibliographic information and references that you will be citing in the thesis for use with BibTeX. You can write it manually, but there are reference manager programs available that will create and manage it for you. Bibliographies in LaTeX are a large subject and you may need to read about BibTeX before starting with this. Many modern reference managers will allow you to export your references in BibTeX format which greatly eases the amount of work you have to do.

MastersDoctoralThesis.cls – this is an important file. It is the class file that tells LaTeX how to format the thesis.

main.pdf – this is your beautifully typeset thesis (in the PDF file format) created by LaTeX. It is supplied in the PDF with the template and after you compile the template you should get an identical version.

main.tex – this is an important file. This is the file that you tell LaTeX to compile to produce your thesis as a PDF file. It contains the framework and constructs that tell LaTeX how to layout the thesis. It is heavily commented so you can read exactly what each line of code does and why it is there. After you put your own information into the *THESIS INFORMATION* block – you have now started your thesis!

Files that are *not* included, but are created by LaTeX as auxiliary files include:

main.aux – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main **`.tex`** file.

main.bbl – this is an auxiliary file generated by BibTeX, if it is deleted, BibTeX simply regenerates it when you run the **`main.aux`** file. Whereas the **`.bib`** file contains all the references you have, this **`.bbl`** file contains the references you have actually cited in the thesis and is used to build the bibliography section of the thesis.

main.blg – this is an auxiliary file generated by BibTeX, if it is deleted BibTeX simply regenerates it when you run the main **`.aux`** file.

main.lof – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file. It tells LaTeX how to build the *List of Figures* section.

main.log – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file. It contains messages from LaTeX, if you receive errors and warnings from LaTeX, they will be in this `.log` file.

main.lot – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file. It tells LaTeX how to build the *List of Tables* section.

main.out – this is an auxiliary file generated by LaTeX, if it is deleted LaTeX simply regenerates it when you run the main `.tex` file.

So from this long list, only the files with the `.bib`, `.cls` and `.tex` extensions are the most important ones. The other auxiliary files can be ignored or deleted as LaTeX and BibTeX will regenerate them.

You will need to personalise the thesis template and make it your own by filling in your own information. This is done by editing the `main.tex` file in a text editor or your favourite LaTeX environment.

Open the file and scroll down to the second large block titled *THESIS INFORMATION* where you can see the entries for *University Name*, *Department Name*, etc …

Fill out the information about yourself, your group and institution. You can also insert web links, if you do, make sure you use the full URL, including the `http://` for this. If you don't want these to be linked, simply remove the `\href{url}{name}` and only leave the name.

When you have done this, save the file and recompile `main.tex`. All the information you filled in should now be in the PDF, complete with web links. You can now begin your thesis proper!

The `main.tex` file contains the structure of the thesis. There are plenty of written comments that explain what pages, sections and formatting the LaTeX code is creating. Each major document element is divided into commented blocks with titles in all capitals to make it obvious what the following bit of code is doing. Initially there seems to be a lot of LaTeX code, but this is all formatting, and it has all been taken care of so you don't have to do it.

Begin by checking that your information on the title page is correct. For the thesis declaration, your institution may insist on something different than the text given. If this is the case, just replace what you see with what is required in the *DECLARATION PAGE* block.

Then comes a page which contains a funny quote. You can put your own, or quote your favourite scientist, author, person, and so on. Make sure to put the name of the person who you took the quote from.

Following this is the abstract page which summarises your work in a condensed way and can almost be used as a standalone document to describe what you have done. The text you write will cause the heading to move up so don't worry about running out of space.

Next come the acknowledgements. On this page, write about all the people who you wish to thank (not forgetting parents, partners and your advisor/supervisor).

The contents pages, list of figures and tables are all taken care of for you and do not need to be manually created or edited. The next set of pages are more likely to be optional and can be deleted since they are for a more technical thesis: insert a list of abbreviations you have used in the thesis, then a list of the physical constants and numbers you refer to and finally, a list of mathematical symbols used in any formulae. Making the effort to fill these tables means the reader has a one-stop place to refer to instead of searching the internet and references to try and find out what you meant by certain abbreviations or symbols.

The list of symbols is split into the Roman and Greek alphabets. Whereas the abbreviations and symbols ought to be listed in alphabetical order (and this is *not* done automatically for you) the list of physical constants should be grouped into similar themes.

The next page contains a one line dedication. Who will you dedicate your thesis to?

Finally, there is the block where the chapters are included. Uncomment the lines (delete the `%` character) as you write the chapters. Each chapter should be written in its own file and put into the *Chapters* folder and named **`Chapter1`**, **`Chapter2`**, etc... Similarly for the appendices, uncomment the lines as you need them. Each appendix should go into its own file and placed in the *Appendices* folder.

After the preamble, chapters and appendices finally comes the bibliography. The bibliography style (called *`authoryear`*) is used for the bibliography and is a fully featured style that will even include links to where the referenced paper can be found online. Do not underestimate how grateful your reader will be to find that a reference to a paper is just a click away. Of course, this relies on you putting the URL information into the BibTeX file in the first place.

To get the best out of this template, there are a few conventions that you may want to follow.

One of the most important (and most difficult) things to keep track of in such a long document as a thesis is consistency. Using certain conventions and ways of doing things (such as using a Todo list) makes the job easier. Of course, all of these are optional and you can adopt your own method.

This thesis template is designed for double sided printing (i.e. content on the front and back of pages) as most theses are printed and bound this way. Switching to one sided printing is as simple as uncommenting the *`oneside`* option of the `documentclass` command at the top of the **`main.tex`** file. You may then wish to adjust the margins to suit specifications from your institution.

The headers for the pages contain the page number on the outer side (so it is easy to flick through to the page you want) and the chapter name on the inner side.

The text is set to 11 point by default with single line spacing, again, you can tune the text size and spacing should you want or need to using the options at the very start of **`main.tex`**. The spacing can be changed similarly by replacing the *`singlespacing`* with *`onehalfspacing`* or *`doublespacing`*.

The paper size used in the template is A4, which is the standard size in Europe. If you are using this thesis template elsewhere and particularly in the United States, then you may have to change the A4 paper size to the US Letter size. This can be done in the margins settings section in **`main.tex`**.

Due to the differences in the paper size, the resulting margins may be different to what you like or require (as it is common for institutions to dictate certain margin sizes). If this is the case, then the margin sizes can be tweaked by modifying the values in the same block as where you set the paper size. Now your document should be set up for US Letter paper size with suitable margins.

The `biblatex` package is used to format the bibliography and inserts references such as this one [**Reference1**]. The options used in the **`main.tex`** file mean that the in-text citations of references are formatted with the author(s) listed with the date of the publication. Multiple references are separated by semicolons (e.g. [**Reference2**, **Reference1**]) and references with more than three authors only show the first author with *et al.* indicating there are more authors (e.g. [**Reference3**]). This is done automatically for you. To see how you use references, have a look at the **`Chapter1.tex`** source file. Many reference managers allow you to simply drag the reference into the document as you type.

Scientific references should come *before* the punctuation mark if there is one (such as a comma or period). The same goes for footnotes[1]. You can change this but the most important thing is to keep the convention consistent throughout the thesis. Footnotes themselves should be full, descriptive sentences (beginning with a capital letter and ending with a full stop). The APA6 states: "Footnote numbers should be superscripted, [...], following any punctuation mark except a dash." The Chicago manual of style states: "A note number should be placed at the end of a sentence or clause. The

[1]Such as this footnote, here down at the bottom of the page.

number follows any punctuation mark except the dash, which it precedes. It follows a closing parenthesis."

The bibliography is typeset with references listed in alphabetical order by the first author's last name. This is similar to the APA referencing style. To see how LaTeX typesets the bibliography, have a look at the very end of this document (or just click on the reference number links in in-text citations).

The bibtex backend used in the template by default does not correctly handle unicode character encoding (i.e. "international" characters). You may see a warning about this in the compilation log and, if your references contain unicode characters, they may not show up correctly or at all. The solution to this is to use the biber backend instead of the outdated bibtex backend. This is done by finding this in **`main.tex`**: *`backend=bibtex`* and changing it to *`backend=biber`*. You will then need to delete all auxiliary BibTeX files and navigate to the template directory in your terminal (command prompt). Once there, simply type `biber main` and biber will compile your bibliography. You can then compile **`main.tex`** as normal and your bibliography will be updated. An alternative is to set up your LaTeX editor to compile with biber instead of bibtex, see here for how to do this for various editors.

Tables are an important way of displaying your results, below is an example table which was generated with this code:

```
\begin{table}
\caption{The effects of treatments X and Y on the four groups studied.}
\label{tab:treatments}
\centering
\begin{tabular}{l l l}
\toprule
\tabhead{Groups} & \tabhead{Treatment X} & \tabhead{Treatment Y} \\
\midrule
1 & 0.2 & 0.8\\
2 & 0.17 & 0.7\\
3 & 0.24 & 0.75\\
4 & 0.68 & 0.3\\
\bottomrule\\
\end{tabular}
\end{table}
```

You can reference tables with `\ref{<label>}` where the label is defined within the table environment. See **`Chapter1.tex`** for an example of the label and citation (e.g. Table 1.1).

There will hopefully be many figures in your thesis (that should be placed in the *Figures* folder). The way to insert figures into your thesis is to use a code template like this:

TABLE 1.1: The effects of treatments X and Y on the four groups studied.

Groups	Treatment X	Treatment Y
1	0.2	0.8
2	0.17	0.7
3	0.24	0.75
4	0.68	0.3

```
\begin{figure}
\centering
\includegraphics{Figures/Electron}
\decoRule
\caption[An Electron]{An electron (artist's impression).}
\label{fig:Electron}
\end{figure}
```

Also look in the source file. Putting this code into the source file produces the picture of the electron that you can see in the figure below.

FIGURE 1.1: An electron (artist's impression).

Sometimes figures don't always appear where you write them in the source. The placement depends on how much space there is on the page for the figure. Sometimes there is not enough room to fit a figure directly where it should go (in relation to the text) and so LaTeX puts it at the top of the next page. Positioning figures is the job of LaTeX and so you should only worry about making them look good!

Figures usually should have captions just in case you need to refer to them (such as in Figure 1.1). The `\caption` command contains two parts, the first part, inside the square brackets is the title that will appear in the *List of Figures*, and so should be short. The second part in the curly brackets should contain the longer and more descriptive caption text.

The `\decoRule` command is optional and simply puts an aesthetic horizontal line below the image. If you do this for one image, do it for all of them.

LaTeX is capable of using images in pdf, jpg and png format.

If your thesis is going to contain heavy mathematical content, be sure that LaTeX will make it look beautiful, even though it won't be able to solve the equations for you.

The "Not So Short Introduction to LaTeX" (available on CTAN) should tell you everything you need to know for most cases of typesetting mathematics. If you need more information, a much more thorough mathematical guide is available from the AMS called, "A Short Math Guide to LaTeX" and can be downloaded from: `ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf`

There are many different LaTeX symbols to remember, luckily you can find the most common symbols in The Comprehensive LaTeX Symbol List.

You can write an equation, which is automatically given an equation number by LaTeX like this:

```
\begin{equation}
E = mc^{2}
\label{eqn:Einstein}
\end{equation}
```

This will produce Einstein's famous energy-matter equivalence equation:

$$E = mc^2 \tag{1.1}$$

All equations you write (which are not in the middle of paragraph text) are automatically given equation numbers by LaTeX. If you don't want a particular equation numbered, use the unnumbered form:

```
\[ a^{2}=4 \]
```

You should break your thesis up into nice, bite-sized sections and subsections. LaTeX automatically builds a table of Contents by looking at all the `\chapter{}`, `\section{}` and `\subsection{}` commands you write in the source.

The Table of Contents should only list the sections to three (3) levels. A `chapter{}` is level zero (0). A `\section{}` is level one (1) and so a `\subsection{}` is level two (2). In your thesis it is likely that you will even use a `subsubsection{}`, which is level three (3). The depth to which the Table of Contents is formatted is set within **`MastersDoctoralThesis.cls`**. If you need this changed, you can do it in **`main.tex`**.

You have reached the end of this mini-guide. You can now rename or overwrite this pdf file and begin writing your own **`Chapter1.tex`** and the rest of your thesis. The easy work of setting up the structure and framework has been taken care of for you. It's now your job to fill it out!

Good luck and have lots of fun!

Guide written by —
Sunil Patel: www.sunilpatel.co.uk
Vel: LaTeXTemplates.com

Chapter 2

Literature Review

Where4 incorporates ideas from at least three separate disciplines: software verification, machine learning, and software measurement and metrics. Each of these disciplines has a considerable body of research associated with it. This chapter focusses on the intersections of the three areas while addressing to relevant literature from each individual area. A Venn diagram illustrating how the literature was classified is given in Figure 2.1. It can be seen that the intersection of *software verification* and *measurement/metrics* itself contains two separate categories of interest: benchmarks and competitions in formal methods, and the relatively young field of *proof engineering*. The rest of this chapter will review the literature associated with each segment of Figure 2.1 in turn.

2.1 Software Verification Systems

2.1.1 Verification and Measurement

Benchmarks and Competitions

The need for a standard set of benchmarks for the diverse range of software systems is a recurring issue in the literature [6]. The benefits of such a benchmark suite are identified by the SMTLIB [2] project. The performance of SMT solvers has significantly improved in recent years due in part to the standardisation of an input language and the use of standard benchmark programs in competitions [7][5]. The TPTP (Thousands of Problems for Theorem Provers) project [11] has similar aims but a wider scope: targeting theorem provers which specialise in numerical problems as well as general-purpose SAT and SMT solvers. The TPTP library is specifically designed for the rigorous experimental comparison of solvers [10].

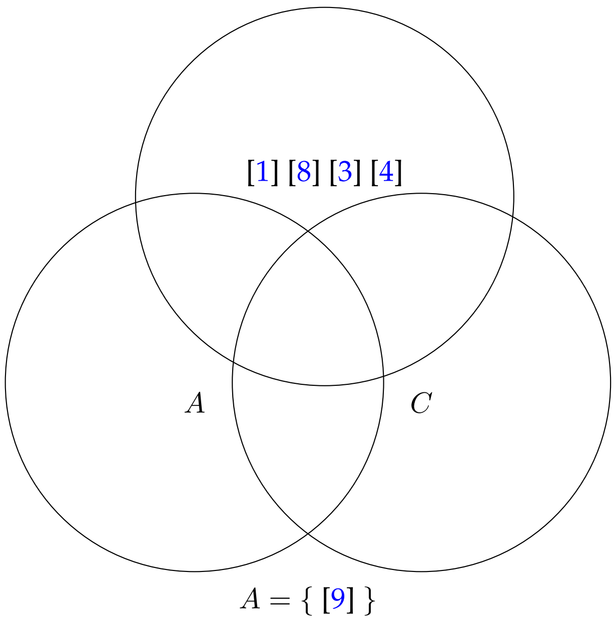

FIGURE 2.1: Three disciplines identifies as being relevant to this project

Proof Engineering

2.1.2 Verification and Machine Learning

2.1.3 Where4 and the intersection of all three disciplines

2.2 Software Measurement and Metrics

2.2.1 Measurement and Machine Learning

2.3 Machine Learning

Chapter 3

Experimental Methodology: Data Collection

3.1 Selection of tools and programs

3.1.1 Selection of SMT solvers

3.1.2 Selection of Why3 programs

The logical theories used

3.2 Measurement of dynamic properties

3.2.1 Execution time

Container-based timings using *BenchExec*

Accounting for randomness with confidence intervals

3.2.2 Prover output

3.3 Static metrics

3.3.1 Using the Why3 notion of "goal shape"

3.3.2 Extracting static metrics from Why3 proof obligation formulæ

Chapter 4

Experimental Methodology: Predicting Response Variables

4.1 The Scoring Function

4.1.1 Favouring *Timeout* over *Unknown*

4.1.2 Favouring *Unknown* over *Timeout*

4.1.3 Favouring shortest time over prover result

4.2 Predicting performance

4.2.1 Statistical Methods versus Machine Learning

4.2.2 Regression and Classification

4.3 Choosing the most effective algorithm

4.3.1 The prediction of score as a regression task

4.3.2 The prediction of *result* as a classification task

4.3.3 The prediction of *time* as a regression task

4.3.4 The prediction of *time* as a classification task

Binarization

Multiclass via binning

4.3.5 The prediction of score as an ensemble task

4.3.6 The prediction of score as a multi-label task

Chapter 5

Experimental Results, Evaluation and System Integration

5.1 Introduction to the evaluation metrics

5.1.1 Normalised Distributed Cumulative Gain

5.1.2 Mean Average Error

5.2 Predicting prover result

5.3 Predicting execution time

5.4 Predicting score directly

5.4.1 Predict each score function individually…

5.5 System integration

5.5.1 Architecture of a Why3 plugin

5.5.2 Timing the integrated system

Chapter 6

Discussion

Chapter 7

Threats to Validity

7.1 Construct

7.2 Internal

7.3 External

7.4 Conclusion

Chapter 8

Conclusions

Appendix A

Python code for data analysis, statistics and learning

Write your Appendix content here.

Appendix B

Ocaml code for interacting with Why3

Write your Appendix content here.

Appendix C

Pattern identification of verification programs using clustering

Bibliography

[1] Mike Barnett et al. “Boogie: A Modular Reusable Verifier for Object-Oriented Programs”. In: *Formal Methods for Components and Objects: 4th International Symposium*. Amsterdam, The Netherlands, Nov 2005, pp. 364–387. ISBN: 978-3-540-36750-5.

[2] Clark Barrett, Aaron Stump, and Cesare Tinelli. *The Satisfiability Modulo Theories Library (SMT-LIB)*. 2010. URL: http://www.smt-lib.org.

[3] Clark Barrett and Cesare Tinelli. “CVC3”. In: *Computer Aided Verification*. Berlin, Germany, July 2007, pp. 298–302.

[4] Clark Barrett et al. “CVC4”. In: *Computer Aided Verification*. Snowbird, UT, USA, July 2011, pp. 171–177. ISBN: 978-3-642-22109-5.

[5] Dirk Beyer. “Status Report on Software Verification”. English. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Grenoble, France, Apr 2014, pp. 373–388. ISBN: 978-3-642-54861-1.

[6] Dirk Beyer et al. “Evaluating Software Verification Systems: Benchmarks and Competitions (Dagstuhl Reports 14171)”. In: *Dagstuhl Reports* 4.4 (2014). ISSN: 2192-5283.

[7] David R. Cok, Aaron Stump, and Tjark Weber. “The 2013 Evaluation of SMT-COMP and SMT-LIB”. In: *Journal of Automated Reasoning* 55.1 (2015), pp. 61–90. ISSN: 1573-0670.

[8] Sylvain Conchon and Évan Contejean. *The Alt-Ergo automatic theorem prover*. 2008. URL: http://alt-ergo.lri.fr/.

[9] Yulia Demyanova et al. “Empirical Software Metrics for Benchmarking of Verification Tools”. English. In: *Computer Aided Verification*. San Francisco, CA, USA, July 2015, pp. 561–579. ISBN: 978-3-319-21689-8.

[10] Geoff Sutcliffe and Christian Suttner. “Evaluating general purpose automated theorem proving systems”. In: *Artificial Intelligence* 131.1-2 (2001), pp. 39–54. ISSN: 0004-3702.

[11] Geoff Sutcliffe and Christian Suttner. “The TPTP Problem Library”. In: *Journal Automated Reasoning* 21.2 (Oct. 1998), pp. 177–203. ISSN: 0168-7433.