Computer Systems and Professional Practice

Professor Matthew Leeke
School of Computer Science
University of Birmingham

Professional Technologies

What Professional Technologies?

In Object Oriented Programming you're studying a programming language and adjacent technologies, including Git and virtualisation (if you're using the online lab environment)

It's not particularly useful for us to gaze into a crystal ball to predict what technologies will be more important for your collective careers

Let's think about two specific technologies, with which you'll likely have different levels of engagement, and think about how we might build your skills over time

LaTeX and BibTeX

Generative AI

LaTeX and BibTeX

LaTeX and BibTeX

Introduction to LaTeX

Structure and processing of a LaTeX source file

Common commands in LaTeX

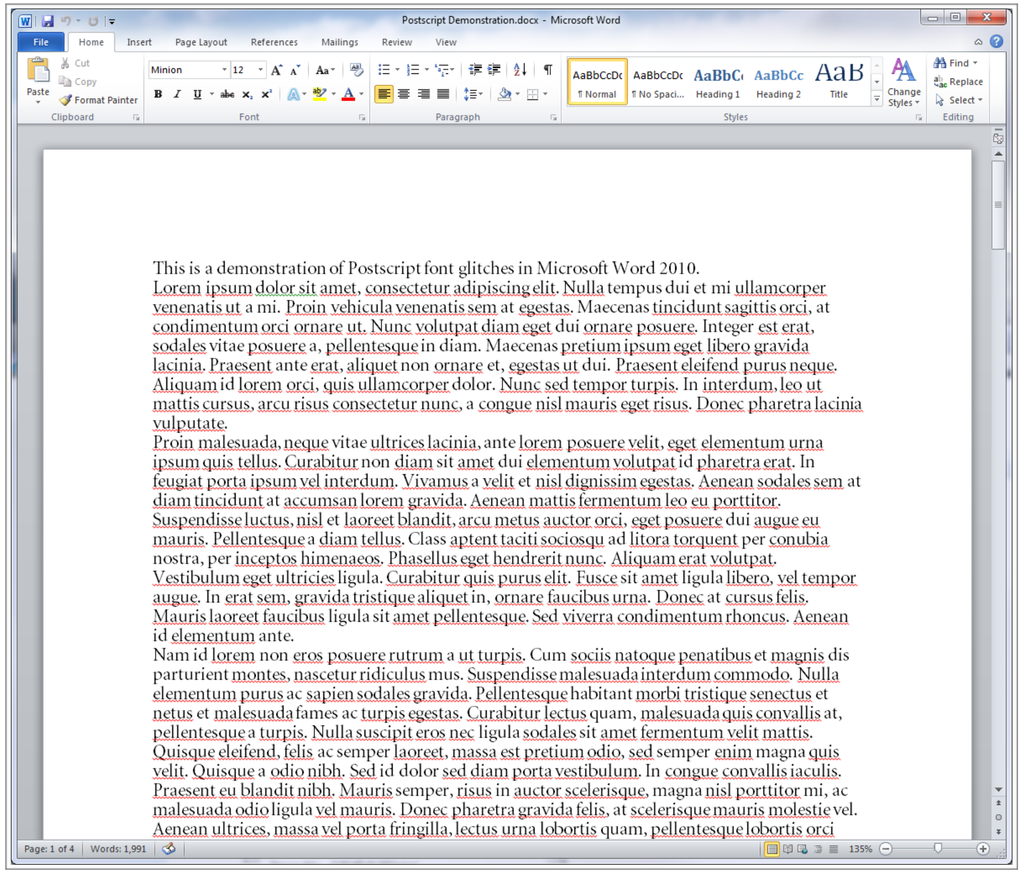
Itemisation, Enumeration, Figures, Tables, Equations and Mathematics

Introduction to BibTeX

Referencing

Bibliography

LaTeX vs Microsoft Word



***Abstract*—The application of machine learning to software fault injection data has been shown to be an effective approach for the generation of efficient error detection mechanisms (EDMs) at arbitrary locations. However, such approaches to the design of EDMs have invariably adopted a fault model with a single-fault assumption, limiting the practical relevance of the detectors and their evaluation. Software containing more than a single fault is commonplace, with prominent safety standards recognise that critical failures are often the result of unlikely or unforeseen combinations of faults. This paper addresses this shortcoming, demonstrating that it is possible to generate similarly efficient EDMs under more realistic fault models. In particular, it is shown that (i) efficient EDMs can be designed using fault data collected under models accounting for the occurrence of simultaneous faults, (ii) exhaustive fault injection under a simultaneous bit flip model can yield improvements to true EDM efficiency, and (iii) exhaustive fault injection under a simultaneous bit flip model can made non-exhaustive, thereby reducing the resource costs of experimentation to practicable levels, without sacrificing the efficiency of the resultant EDMs.**

***Keywords*-Detection; Error; Fault; Injection; Machine Learning**

I. Introduction

The design of error detection mechanisms (EDMs) is integral to the development of dependable software systems [1]. EDMs are fundamentally concerned with the detection of erroneous software states. Once detected by an EDM, erroneous software states can be handled by error recovery mechanisms (ERMs) to maintain proper function. A failure to contain the propagation of erroneous state is known to make recovery more difficult, leading to a focus on the efficiency of EDMs through measures such as coverage and latency [2], [3].

The effectiveness of an EDM has been shown to depend on two factors. These factors are (i) the error detection predicate that it implements and (ii) its location in a software system [4]. This gives rise to two related problems. Firstly, the EDM design problem, which is concerned with the derivation of an error detection predicate over program variables that can be used for the detection of erroneous system states. Secondly, the EDM location problem, which is concerned with the identification of those software locations at which an EDM will be most effective. Though often treated as orthogonal for simplicity, the interaction of the implemented error detection predicate and the software location are demonstrably critical to the efficiency of an EDM [5], [6].

The efficiency of a particular EDM can be characterised by completeness and accuracy [4]. Completeness is the capability of an EDM to detect erroneous states, i.e., its true positive rate, whilst accuracy relates to its ability to avoid incorrectly detecting erroneous states, i.e., its false positive rate. An erroneous state is one that will lead to system failure if the error is not handled, where a failure is characterised as a violation of a system specification. An EDM that is complete and accurate is commonly known as a perfect detector. Due to implementation constraints, it is not generally possible to generate or guarantee the existence of a perfect detector for a software location [7].

The role of a fault model is to provide a means for analysing the response of software to the presence a well defined set of faults, such that appropriate EDMs and ERMs can be designed to impart dependability. The assumption that faults do not occur simultaneously or interact is a limitation of many established fault models and the software fault injection frameworks that implement them, not least because software containing more than a single fault are commonplace [8]. Further, existing safety standards recognise that critical failures are often the result of unlikely or unforeseen interactions combinations of faults [9].

It has been shown that efficient error detection predicates for EDMs can be designed through the application of machine learning algorithms to data sets generated during software fault injection [10]. This approach demonstrated, under a transient data value fault model, that it was possible to generate error detection predicates for specified locations with a true positive rate of nearly 100% and a false positive rate close to 0% for the detection of failure-inducing states. Consistent with the overwhelming majority of software fault injection frameworks, this efficiency was achieved under a single-fault assumption, calling into question their suitability for real-world application.

More generally, when generating error detection predicates for EDMs through the application of machine learning to fault injection data, an implication of the single-fault assumption is that the efficiency properties of the EDMs are principally relevant in the context of a single fault, thus limiting their application in practical software systems. This paper directly addresses this issue by demonstrating that efficient EDMs can be designed using fault injection data collected under models accounting for the occurrence of simultaneous faults. Moreover, it is shown that the adoption of such fault models in fault injection can yield EDMs with improved efficiency, including where non-exhaustive fault injection is performed

What is TeX

Donald Knuth began work on TeX in 1977 in response to the deteriorating typographical quality of scientific articles

TeX is a low-level markup and programming language

Powerful and expressive but complex and time-consuming

TeX is renowned for stability and platform compatibility

What is LaTeX?

LaTeX is a markup tool and document preparation system for the TeX typesetting system

A language that describes the document structure and format, not a text editor

Works in a similar way to static web pages, e.g., HTML pages

Widely used in scientific related documents, particularly academic papers

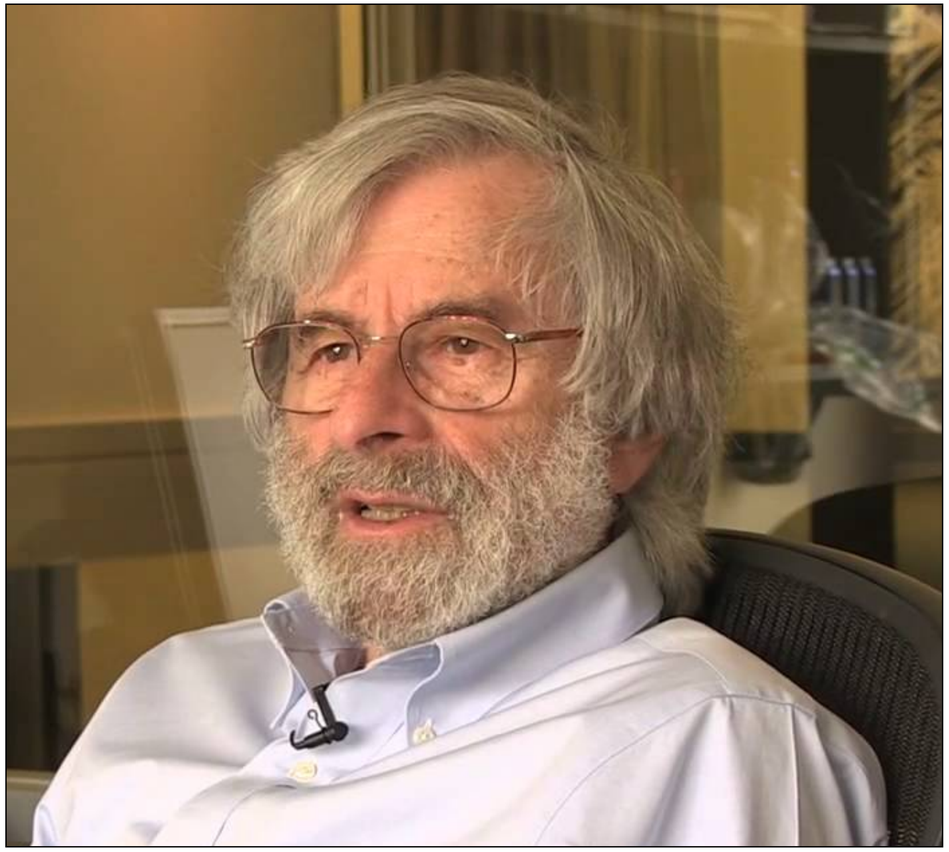
LaTeX and TeX

You can think of LaTeX as a high-level interface to TeX

LaTeX is actually a TeX-based macro package

Originally developed by Leslie Lamport

Can consider LaTeX to be a little like programming

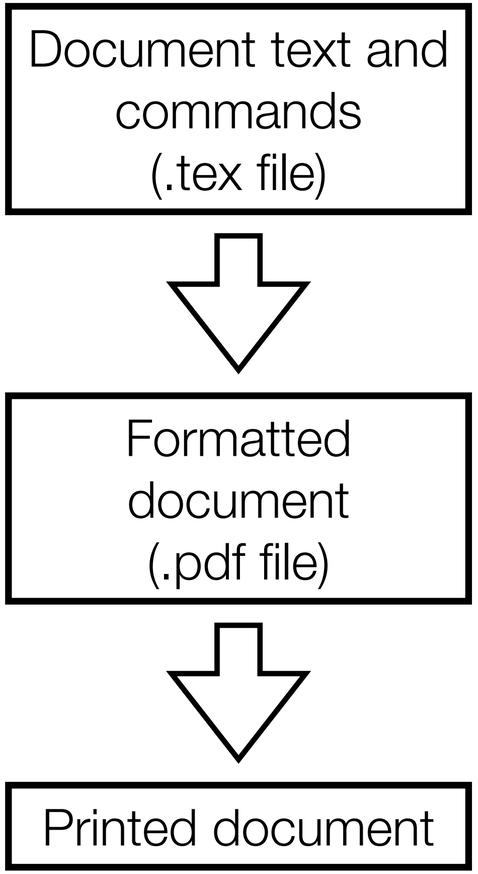


Typesetting

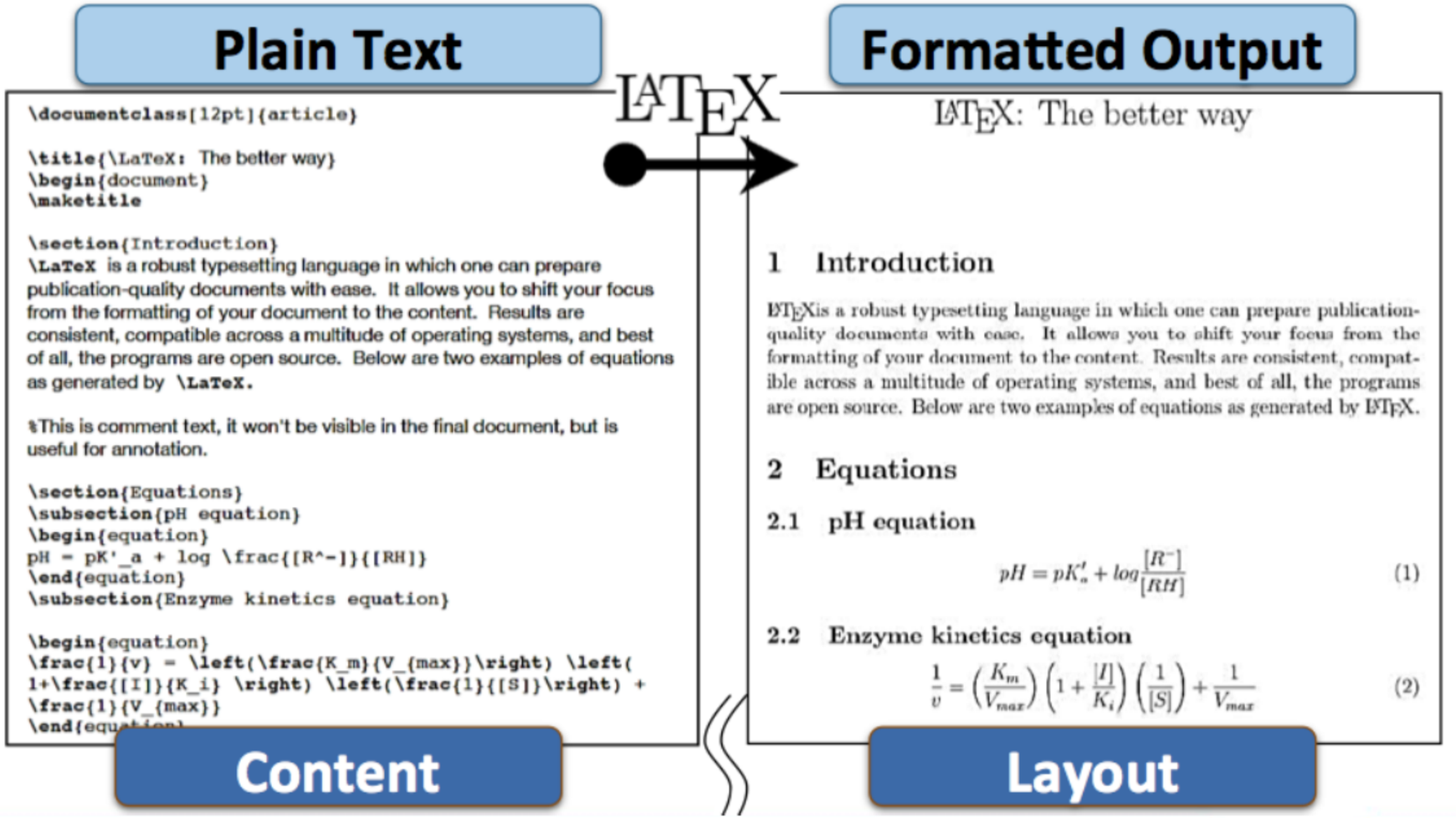
Separation of content from its visual appearance

Enables users to focus on the content without having to handle document layout simultaneously

However, it allows for manual typesetting adjustments, i.e., formatting of standard elements such as tables, figures



Typesetting



LaTeX Advantages

Allows to focus on content and structure rather than layout

Powerful formatting

Standard layout of common elements, e.g, tables, figures

Source files are not version / editor dependent

Equations and scientific notations are easily typeset

Easy referencing

LaTeX Disadvantages

You can’t see the typeset document in real-time

Word processors generally allow document viewing

You need to know specific commands to achieve a desired look for a document

Many templates available

It can be time consuming to develop new document types

Typical Structure of a LaTeX source file

Document Class

First line of a LaTeX source file that specifies the overall style and structure, but also general properties such as font size, paper size etc.

Preamble

Responsible for importing features and document properties contained within packages. Author, title and date are also specified at this point.

Content

Document content. Environment marked by \begin and \end commands.

LaTeX - Document Class

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\usepackage{graphicx}

\ti

\beg

  \r

  \s
  \subsection{What is \LaTeX?}
  % this is a comment
  \LaTeX{} is a document preparation system for the \TeX{}
  typesetting program. It offers programmable desktop publishing
  features and extensive facilities for automating most aspects of
  typesetting and desktop publishing, including numbering and
  cross-referencing, tables and figures, page layout, bibliographies,
  and much more.

\end{document}
```

Document Type

\documentclass[*options*]{class}

predefined formats (article, report, book, letter, ...)

options = a4paper, 11pt, 12pt, 10pt, twocolumn, landscape,...
class = article, report, book,...

LaTeX - Preamble (1)

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\usepackage{graphicx}

\title{Introduction to \

\begin{document}

  \maketitle

  \section{\LaTeX}
  \subsection{What is \LaTeX?}
  % this is a comment
  \LaTeX{} is a document preparation system for the \TeX{}
  typesetting program. It offers programmable desktop publishing
  features and extensive facilities for automating most aspects of
  typesetting and desktop publishing, including numbering and
  cross-referencing, tables and figures, page layout, bibliographies,
  and much more.

\end{document}
```

Packages

\usepackage{package name}

added functionality (graphics, reference style,...).

LaTeX - Preamble (2)

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\usepackage{graphicx}

\title{Introduction to \LaTeX}

\begin{document}

  \maketitle

  \section{\LaTeX}
  \subsection{What is \LaTeX?}
  % this is a comment
  \LaTeX{} is a document preparation system for the \TeX{}
  typesetting program. It offers programmable desktop publishing
  features and extensive facilities for automating most aspects of
  typesetting and desktop publishing, including numbering and
  cross-referencing, tables and figures, page layout, bibliographies,
  and much more.

\end{document}
```

Title

LaTeX - Content

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\usepackage{graphicx}

\title{Introduction to \LaTeX}

\begin{document}

\maketitle

\section{\LaTeX}
\subsection{What is \LaTeX?}
% this is a comment
\LaTeX{} is a document preparation system for the \TeX{}
typesetting program. It offers programmable desktop publishing
features and extensive facilities for automating most aspects of
typesetting and desktop publishing, including numbering and
cross-referencing, tables and figures, page layout, bibliographies,
and much more.

\end{document}
```

Body

LaTeX - Sectioning Content

Sectioning commands available in LaTeX allow to breakdown the document in smaller parts

These commands are mostly dependent on the document class used

Common sectioning options include:

- Chapter
- Section
- Subsection
- Subsubsection

LaTeX - Itemisation and Enumeration

The *enumerate* environment can be used to create a numbered list

```
\begin{enumerate}
\item This is the first item.
\item This is the second.
\item These items are even numbered!
\end{enumerate}
```

1. This is the first item.
2. This is the second.
3. These items are even numbered!

LaTeX - Itemisation and Enumeration

The *itemize* environment can be used to create a bullet point list

```
\begin{itemize}
\item Item 1.
\item Item 2.
\end{itemize}
```

- Item 1.
- Item 2.

LaTeX - Figures

The *figure* environment is used to insert pictures and graphs into documents

To this end, it imports *graphicx* package, i.e., *\usepackage{graphicx}*

```
\includegraphics[width=80mm]{path/to/file.png}
```

```
\begin{figure}[h]
The contents of the figure are placed here.
\caption{This is the caption that will
 appear under the figure}
\label{reflabel} % More on this later
\end{figure}
```

LaTeX - Figures

It can sometimes be tricky to have figures present

Experience working with LaTeX will go a long way
how you want

The same applied to tables!

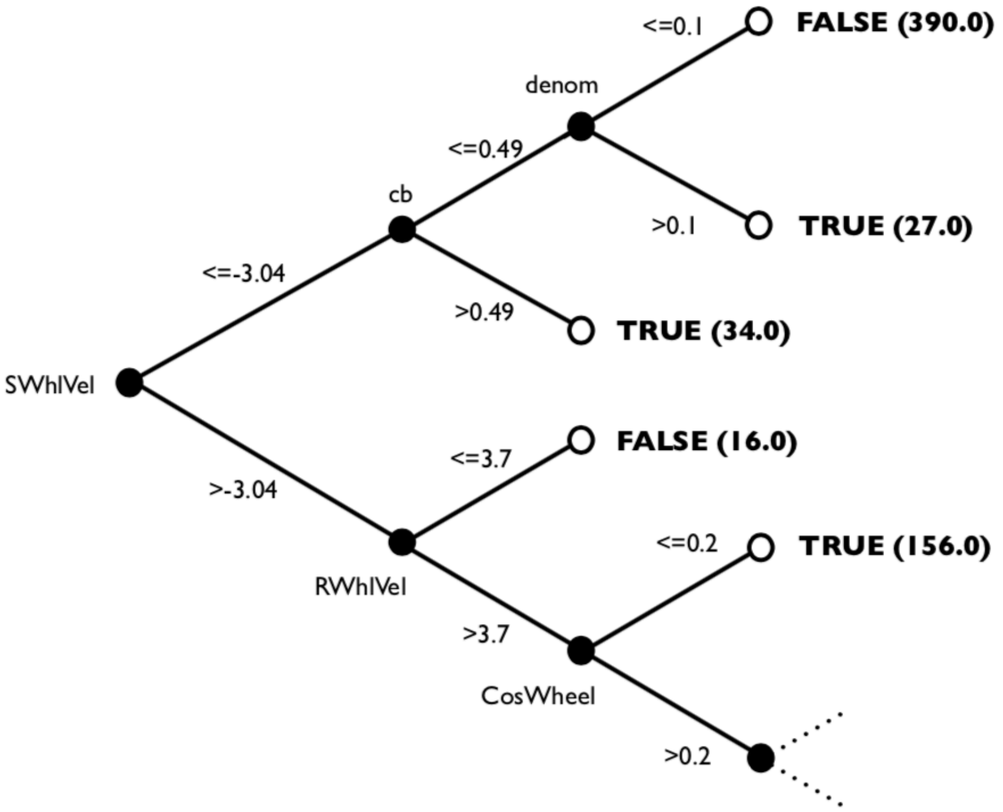


Figure 2. Decision Tree Predicate Example

LaTeX - Tables

The *tabular* environment is used to create tables in LaTeX

Different types of alignment can be achieved

l: left-justified column

c: centered column

r: right-justified column

```
\begin{tabular}{l c r}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{tabular}
```

1	2	3
4	5	6
7	8	9

LaTeX - Floats

Floats are containers that do not allow elements to be split at page break. Thus, if there is not enough space on the remainder of a given page, this element will float on the next page.

Tables and figures are by default floats

Placement specifiers are used to position float elements. These include:

t - Position at the *top* of the page.

b - Position at the *bottom* of page

h - Position float element *here*

LaTeX - Equations and Mathematics (1)

Typeset of mathematics is one of LaTeX strongest advantage

Two main modes for presenting mathematical formulas

Free-form text:

formulas are wrapped in the main text in $ symbols

Displayed:

formulas are contained in an equation environment

LaTeX - Equations and Mathematics (2)

Example of displayed mode for equations:

```
\begin{equation}
a^{2} - b^{2} = (a - b)(a + b)
\label{diff2squares}
\end{equation}
```

$$a^2 - b^2 = (a - b)(a + b)$$

LaTeX – Algorithm Listings

```
\usepackage{algorithmic}
```

```
\begin{algorithmic}
\STATE $i\gets 10$
\IF {$i\geq 5$}
        \STATE $i\gets i-1$
\ELSE
        \IF {$i\leq 3$}
                \STATE $i\gets i+2$
        \ENDIF
\ENDIF
\end{algorithmic}
```

$i \leftarrow 10$
if $i \geq 5$ **then**
  $i \leftarrow i - 1$
else
  if $i \leq 3$ **then**
    $i \leftarrow i + 2$
  end if
end if

LaTeX – Code Listings

```
\usepackage{minted}
```

```
\begin{minted}{python}
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
\end{minted}
```

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

LaTeX - Cross Referencing

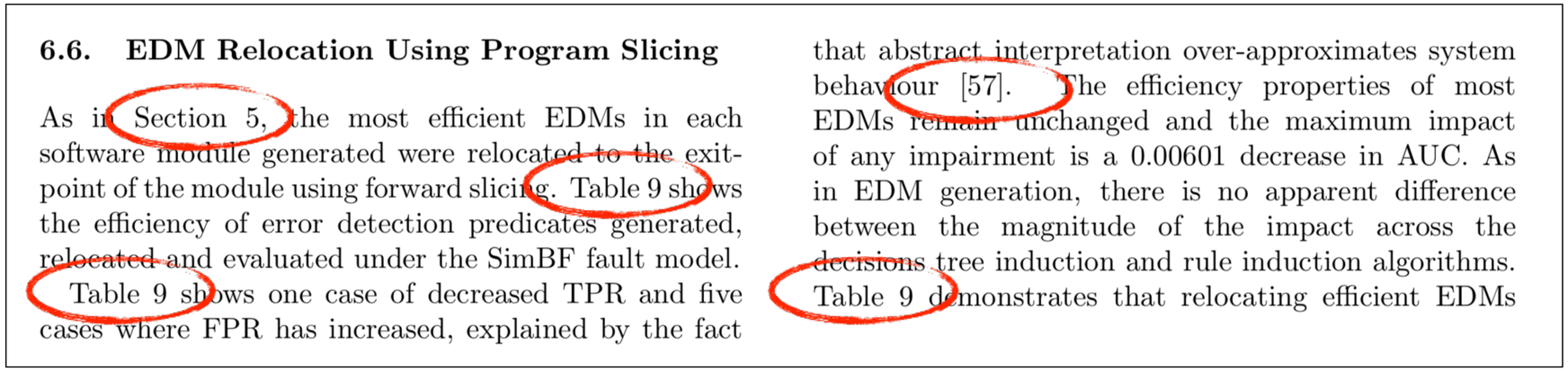
Cross references are often employed for segments of text, images or tables

The following LaTeX commands are available for cross referencing:

\label{identifier}

\ref{identifier}

\pageref{identifier}



6.6. EDM Relocation Using Program Slicing

As in Section 5, the most efficient EDMs in each software module generated were relocated to the exit-point of the module using forward slicing. Table 9 shows the efficiency of error detection predicates generated, relocated and evaluated under the SimBF fault model. Table 9 shows one case of decreased TPR and five cases where FPR has increased, explained by the fact that abstract interpretation over-approximates system behaviour [57]. The efficiency properties of most EDMs remain unchanged and the maximum impact of any impairment is a 0.00601 decrease in AUC. As in EDM generation, there is no apparent difference between the magnitude of the impact across the decisions tree induction and rule induction algorithms. Table 9 demonstrates that relocating efficient EDMs

BibTeX

Extension to LaTeX that allows to keep reference information in a separate file

Simple commands are used to produce citations

Reference listings are automatically generated based on citations made in a document

BibTeX and Bibliography

Bibliography files, i.e., .bib, are created to include a list of entries. These are plain text files that can be easily viewed and edited

Different types if entries have different fields that must be completed in order for an entry to be complete

Examples include *article (journal publications), inproceedings (conference publications), book, report* or *thesis*

```
@article{label91,
author = "A. N. Other",
title = " Article 91",
year = " 1991",
journal = "IEEE Transactions",
volume = "14",
number = "3",
pages = "342–351"
}
```

Referencing using the BibTeX file

Right before the \end{document} the following LaTeX commands should be added

\bibliographystyle{plain}

\bibliography{bibfile_name}

The following LaTeX command is used to cite the reference in the main text

\cite{ref_key}

Processing LaTeX and BibTeX files

The follwing commands are used to process the LaTeX and BibTeX files together

latex example (Compiles .tex file)

bibtex example (Compiles .bib file)

latex example (Adds references at the end of the document)

latex examples (Adds in-text document references)

Generating PDFs Using pdflatex

To compile with citations you will need to compile the LaTeX, run bibtex, and then LaTeX twice more

A commonly used program for LaTeX compilation is pdflatex

First pdflatex generates an initial document and files, e.g., table of contents

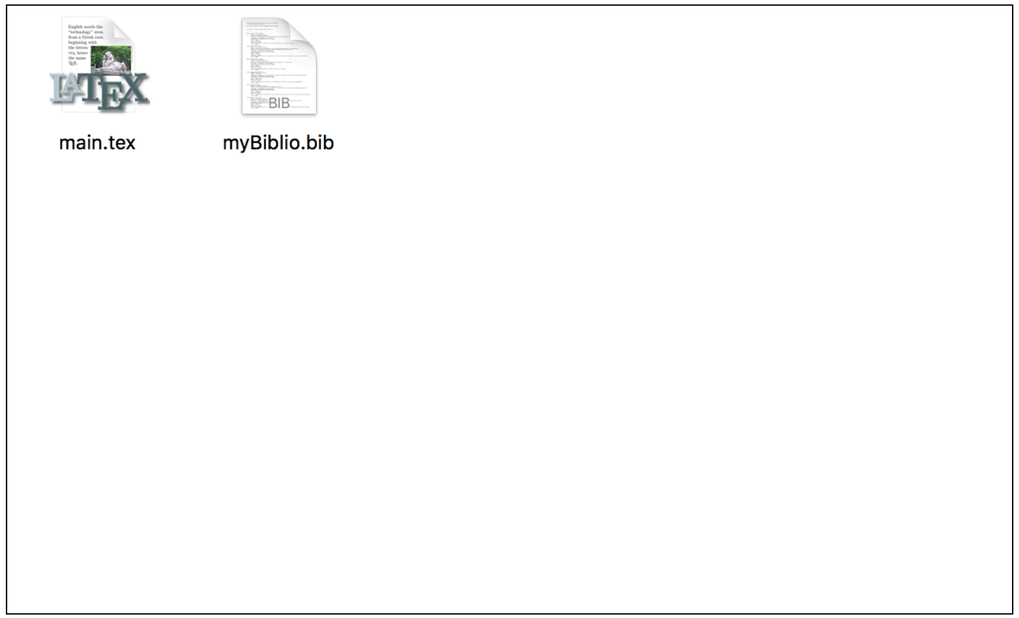
The bibtex uses initial document to produce bibliography information

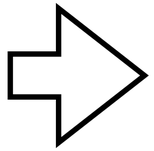
Second pdflatex fills in information

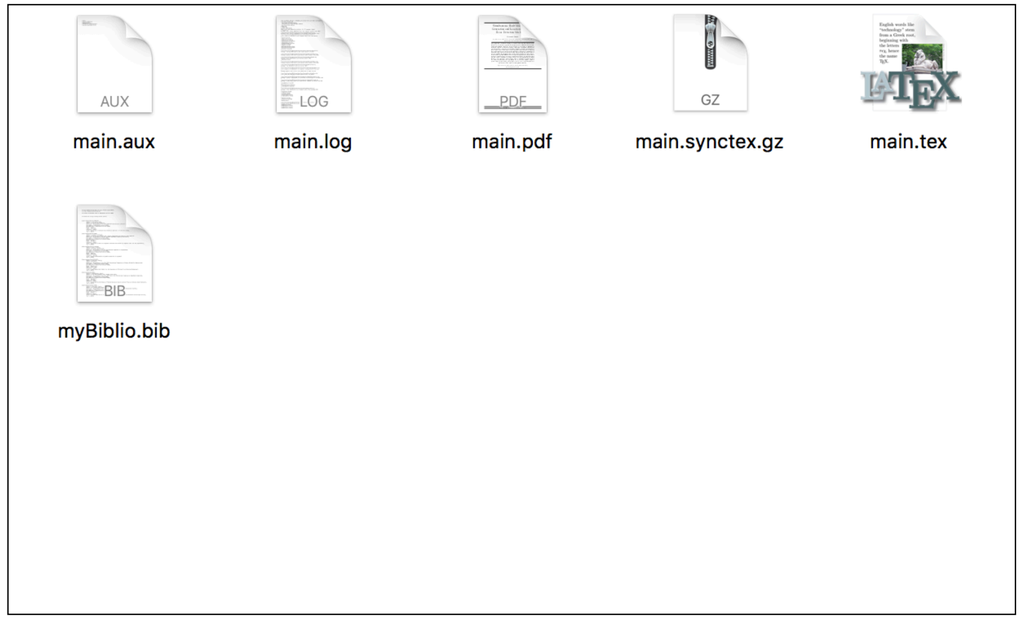
Third pdflatex resolves numbering issues and page arrangement

```
bash$ pdflatex mylatexdoc
bash$ bibtex mylatexdoc
bash$ pdflatex mylatexdoc
bash$ pdflatex mylatexdoc
```

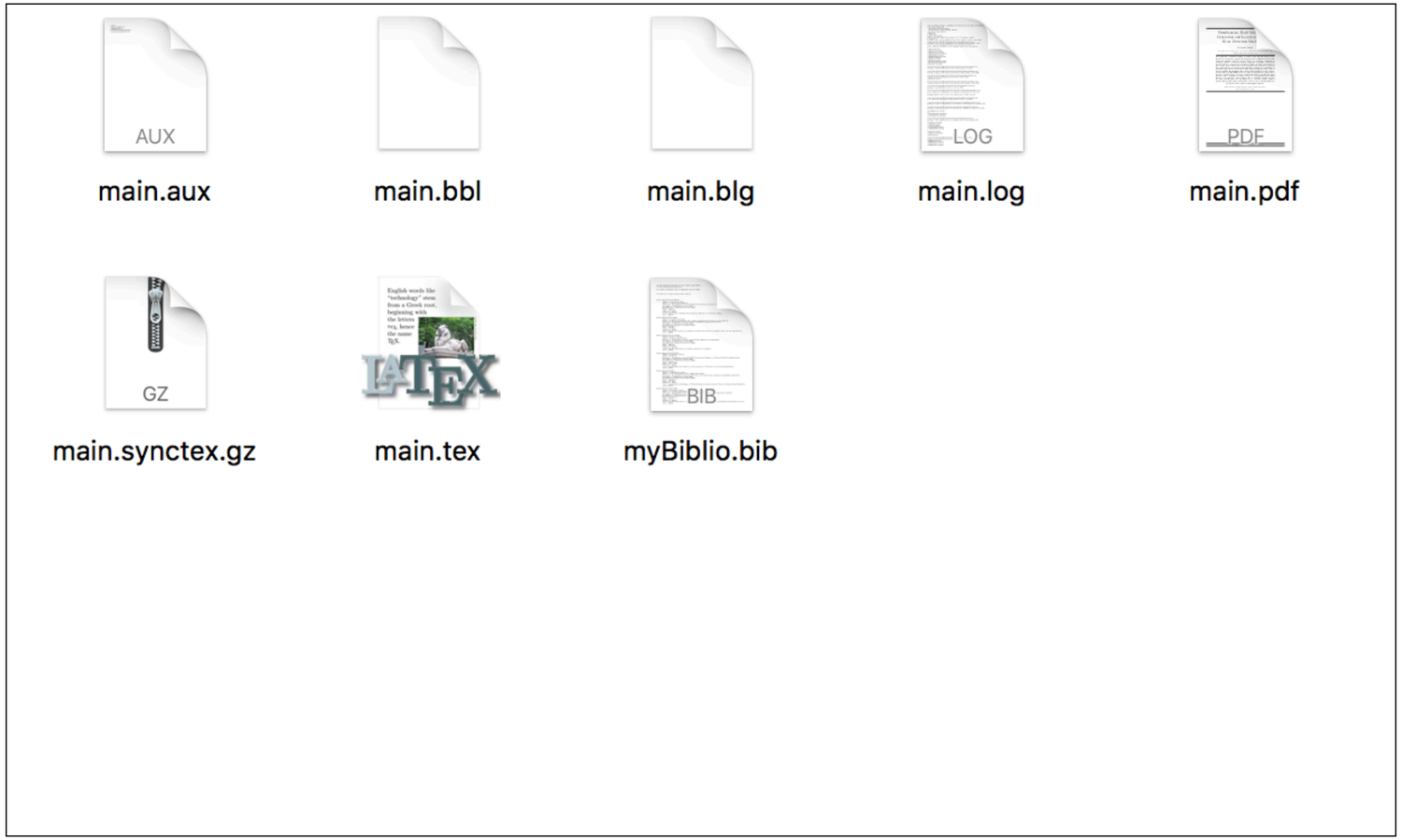
Why So Many Files?







Even More Files!



Installing LaTeX on Linux (Ubuntu)

```
sudo apt install texlive-latex-extra
```

Lots of documentation and installation instructions online

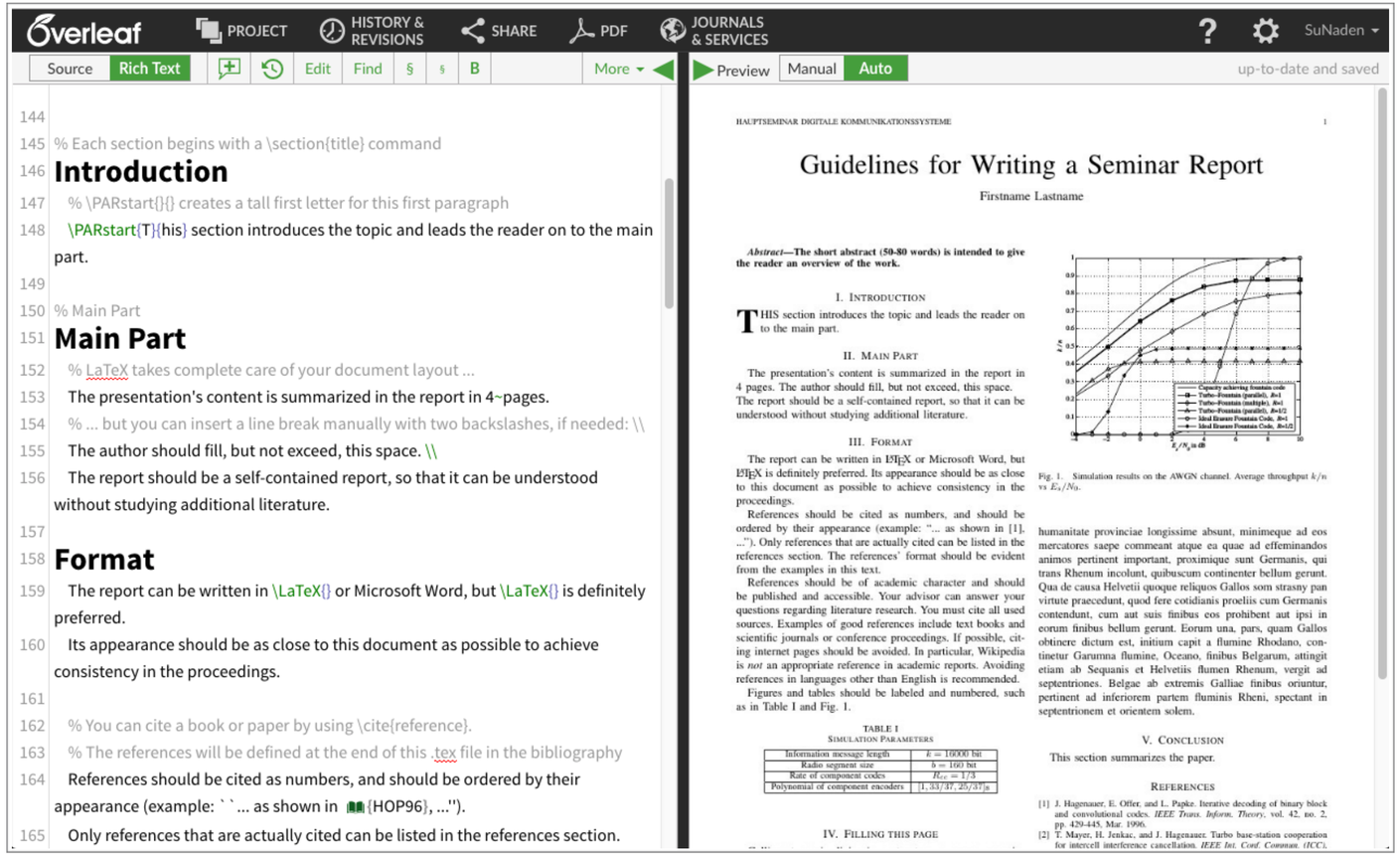
Tools for Writing LaTeX Documents

You can use any editor to write LaTeX but many people prefer to use a dedicated a development tool

Overleaf is among the best online LaTeX platforms

We have an institutional licence

Try it!



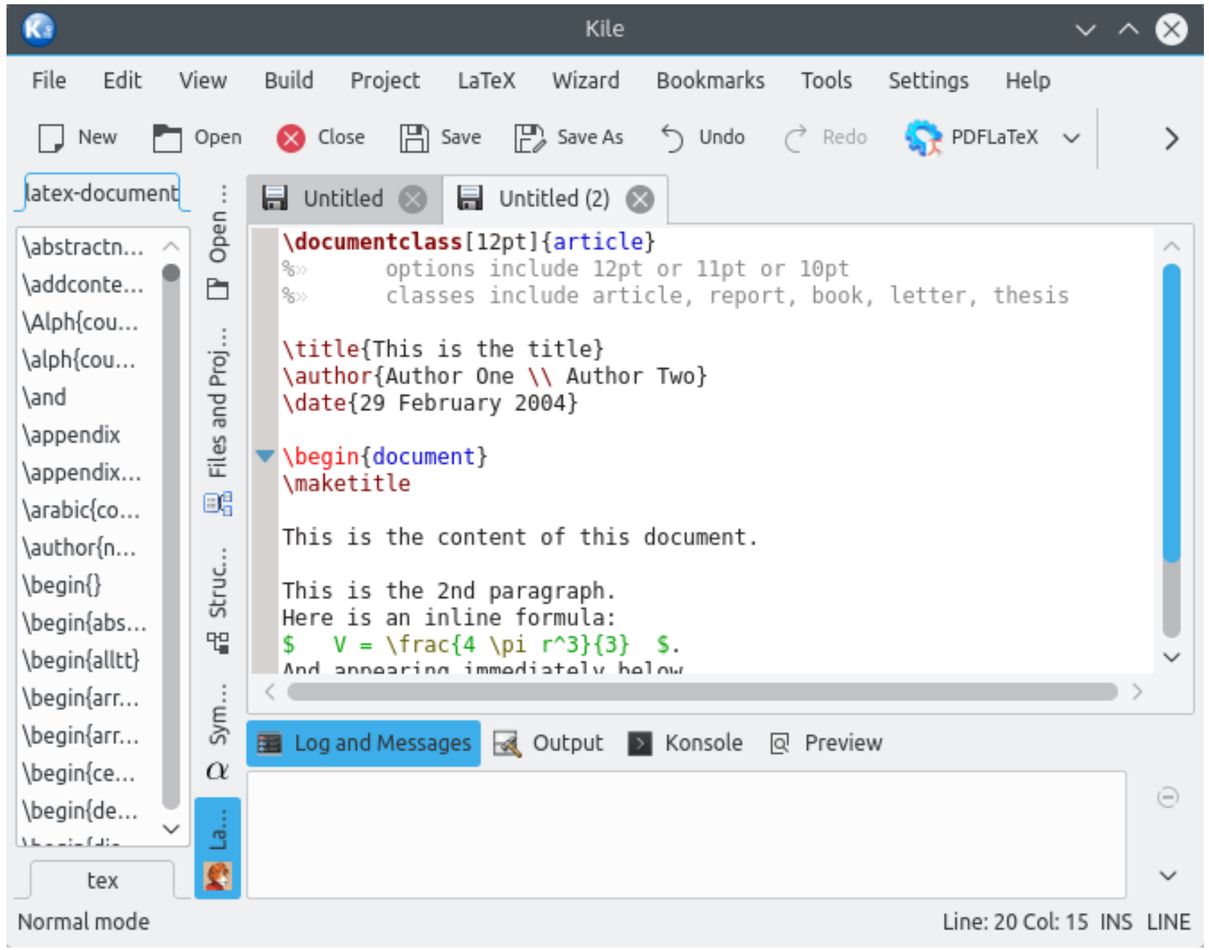
Tools for Writing LaTeX Documents

You can use any editor to write LaTeX but many people prefer to use a dedicated a development tool

Kile is popular on Linux

```
sudo apt install kile
```

Kile isn't my favourite but it's worth a go if you are a Linux user



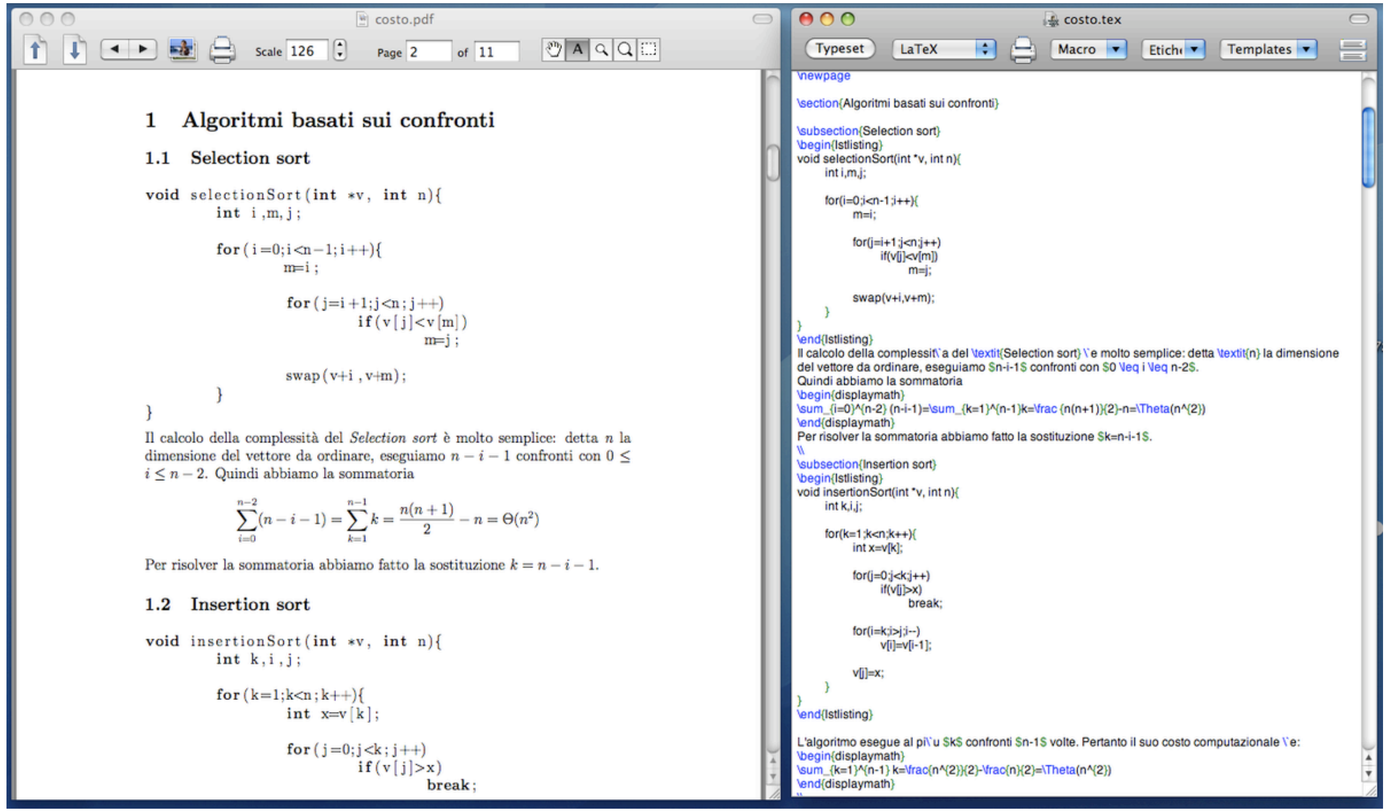
Tools for Writing LaTeX Documents

You can use any editor to write LaTeX but many people prefer to use a dedicated a development tool

TexShop is particularly great for Mac users - I use it a lot!

Free and comprehensive

Definitely worth a try!

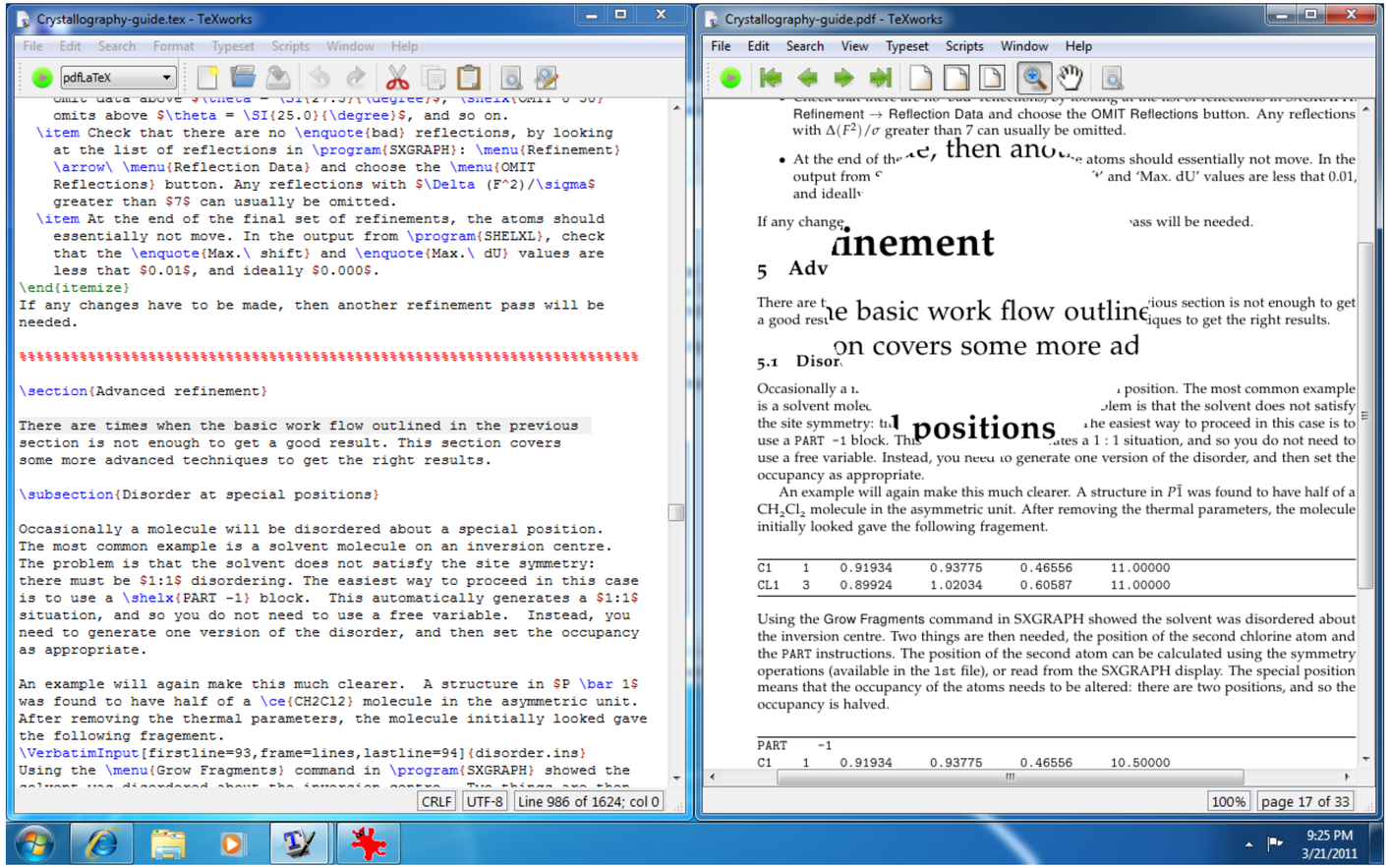


Tools for Writing LaTeX Documents

You can use any editor to write LaTeX but many people prefer to use a dedicated a development tool

TexWorks is a classic solution for Windows, but there are many, many alternatives

I have no strong recommendations for Windows - sorry!



Writing an Scientific Paper

LaTeX templates are often made available by professional organisations

Templates provide a set up format

IEEE Style template examples at:

http://ieeeauthorcenter.ieee.org/create-your-ieee-article/use-authoring-tools-and-ieee-article-templates/ieee-article-templates/

Generative AI

What Is Generative AI?

Algorithms that create new data based on the patterns learned from existing data

Capable of producing text, images, music, and much more

Uses probability distributions to generate new outputs rather than only existing recognising patterns

The Early Days

1950s-1960s:

Introduction of artificial neural networks (ANNs)

1956 - Dartmouth Conference marked the birth of AI

Early ANNs were limited in capability but laid the groundwork

Alan Turing's 1950 Paper: Posed the question, “Can machines think?”

The Rise Of Generative Models

1970s-1990s:

Development of foundational machine learning techniques

Introduction of Bayesian networks and hidden Markov models (HMMs)

Boltzmann Machines (1985) provide early probabilistic graphical models

Generative Models use model data distributions to create new samples

Breakthroughs - Variational Autoencoders and GANs

2013 - Diederik Kingma and Max Welling proposed Variational Autoencoders (VAEs) as a way to generate data with variational inference.

Good for continuous data generation and latent space representation.

Face generation, image synthesis.

2014 - Ian Goodfellow et al. introduced GANs, revolutionizing generative AI.

Comprised of a Generator and a Discriminator in a competitive setting.

Enabled realistic image and video generation.

From art to medical imaging, GANs paved the way for new applications.

Breakthroughs - Transformer Models

2017 - Transformer paper by Vaswani et al. revolutionised natural language processing

GPT (Generative Pre-trained Transformer)

GPT-1 (2018) to GPT-4 (2023) demonstrate progression of increasingly complex, realistic text generation

Chatbots, content creation, code generation

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Breakthroughs - Diffusion Models

2021 - Diffusion models as a fresh paradigm in image generation

Process to gradually "denoises" data to generate high-quality images

Examples include DALL-E 2, Stable Diffusion, etc.

Generate more fine-grained details, especially in visual data

What Questions Does It Raise For You?

Student

- How can I leverage the technology to improve the quality of my work?
- How can I avoid committing or being suspected of committing plagiarism?

Professional

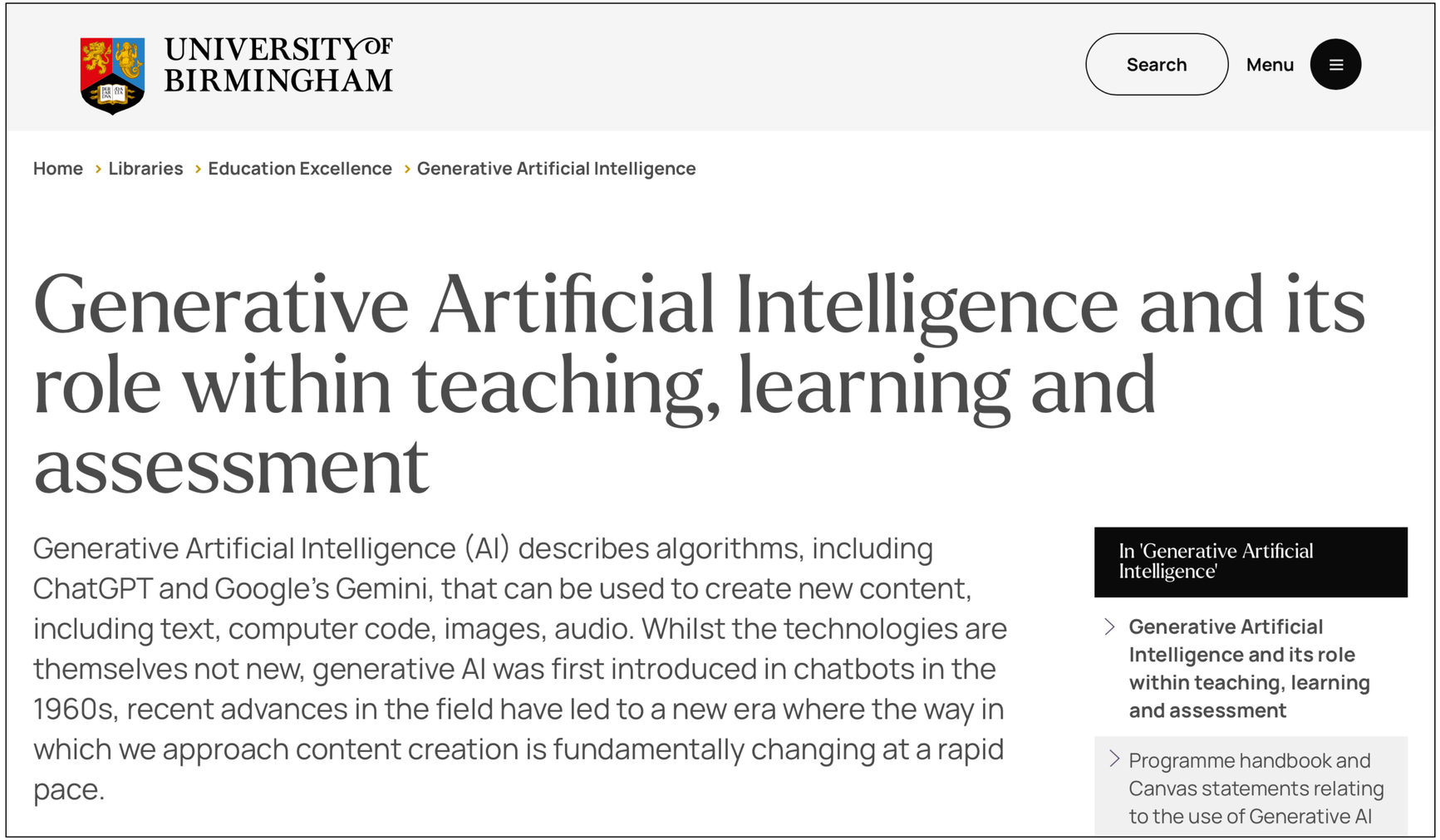
- How can I leverage the technology to improve the quality of my work?
- How can I avoid committing or being suspected of exercising poor professional standards?

UoB and AI

University guidance on Generative AI

Provides guiding principles for staff and students

The Frequently Asked Questions section can help you tackle coursework and navigate the pitfalls



https://www.birmingham.ac.uk/libraries/education-excellence/gai