Joshua Send

Conflict Free Document Editing with Different Technologies

Computer Science Tripos – Part II

Trinity Hall

8th March 2017

Proforma

Name:	**Joshua Send**
College:	**Trinity Hall**
Project Title:	**Conflict Free Document Editing with Different Technol**
Examination:	**Computer Science Tripos – Part II, June 2017**
Word Count:	**1587**[1]
Project Originator:	Joshua Send
Supervisor:	Stephan Kollmann

Original Aims of the Project

TODO[2]

Work Completed

TODO

Special Difficulties

TODO

[1]This word count was computed by `detex diss.tex | tr -cd '0-9A-Za-z \n' | wc -w`

[2]A normal footnote without the complication of being in a table.

Declaration

I, Joshua Send of Trinity Hall, being a candidate for Part II of the Computer Science Tripos [or the Diploma in Computer Science], hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed TODO [signature]

Date TODO [date]

Contents

List of Figures

Acknowledgements

TODO

Chapter 1

Introduction

Real time interaction between users is becoming an increasingly important feature to many applications, from word processing to CAD to social networking. This dissertation examines trade offs that should be considered when applying the prevailing technologies that enable concurrent use of data in applications. More specifically, this project implements and analyzes a concurrent text editor based on Convergent Replicated Data Types (also known as Conflict-free Replicated Data Types), CRDT in short, in comparison to an editor exploiting Operational Transformations (OT) as its core technology.

1.1 Motivation

Realtime collaborative editing was first motivated by a demonstration in the Mother of All Demos by Douglas Engelbart in 1968 [**MotherDemo**]. From that time, it took several decades for implementations of such editing systems to appear. Early products appeared in the 1990's, and the 1989 paper by Gibbs and Ellis [**Ellis1989**] marked the beginning of extended research into operational transformations. Due to almost 20 years of research, OT is a relatively developed field of research and has been applied to products that are commonly used. The most familiar of these is likely to be Google Docs, which seems to behave in a predictable and well understood way. One reason Google Docs is so widely used might be that it follows users' expectations for how a concurrent, multi-user document editor should work. Importantly, this includes lock-free editing and independence on a fast connection, no loss of data, and the guarantee that everyone ends up with the same document when changes are complete. These are in fact the goals around which OT and CRDTs have developed.

The convergence, or consistency, property above is the hardest to provide it is easy to create a system where the last writer wins, but data is lost in the process. In a distributed system such as a shared text editor, the CAP theorem tells us we cannot guarantee all three of consistency, availability, and partition-tolerance [**Gilbert2005**]. However, if we forgo strong consistency guarantees and settle for eventual consistency, we are able provide all three [http://gun.js.org/distributed/matters.html]. However, achieving eventual consistency is non-trivial. Operational transformations, if designed and implemented correctly, accomplish this by recording changes made locally to the data at the moment the user submits them. These operations are sent to the server, which linearizes [GB?] and forwards them to other clients. These clients transform incoming operations against locally made changes in a way that guarantees both the sender and the receiver produce the same data. A simple example can be seen in Figure 1.

This document was produced using LaTeX 2$_\varepsilon$which is based upon LaTeX[**Lamport86**]. To build the document you first need to generate `diss.aux` which, amongst other things, contains the references used. This if done by executing the command:

```
pdflatex diss
```

Then the bibliography can be generated from `refs.bib` using:

```
bibtex diss
```

Finally, to ensure all the page numbering is correct run `pdflatex` on `diss.tex` until the `.aux` files do not change. This usually takes 2 more runs.

1.1.1 The makefile

To simplify the calls to `pdflatex` and `bibtex`, a makefile has been provided, see Appendix **??**. It provides the following facilities:

`make`
Display help information.

`make proposal.pdf`
Format the proposal document as a PDF.

`make view-proposal`
Run `make proposal.pdf` and then display it with a Linux PDF viewer (preferably "okular", if that is not available fall back to "evince").

`make diss.pdf`
Format the dissertation document as a PDF.

`make count`
Display an estimate of the word count.

`make all`
Construct `proposal.pdf` and `diss.pdf`.

`make pub`
Make `diss.pdf` and place it in my `public_html` directory.

`make clean`
Delete all intermediate files except the source files and the resulting PDFs. All these deleted files can be reconstructed by typing `make all`.

1.2 Counting words

An approximate word count of the body of the dissertation may be obtained using:

```
wc diss.tex
```

Alternatively, try something like:

```
detex diss.tex | tr -cd '0-9A-Z a-z\n' | wc -w
```

Chapter 2

Preparation

This chapter is empty!

Chapter 3

Implementation

3.1 Verbatim text

Verbatim text can be included using `\begin{verbatim}` and `\end{verbatim}`. I normally use a slightly smaller font and often squeeze the lines a little closer together, as in:

```
GET "libhdr"

GLOBAL { count:200; all  }

LET try(ld, row, rd) BE TEST row=all
                        THEN count := count + 1
                        ELSE { LET poss = all & ~(ld | row | rd)
                               UNTIL poss=0 DO
                               { LET p = poss & -poss
                                 poss := poss - p
                                 try(ld+p << 1, row+p, rd+p >> 1)
                               }
                             }
LET start() = VALOF
{ all := 1
  FOR i = 1 TO 12 DO
  { count := 0
    try(0, 0, 0)
    writef("Number of solutions to %i2-queens is %i5*n", i, count)
    all := 2*all + 1
  }
  RESULTIS 0
}
```

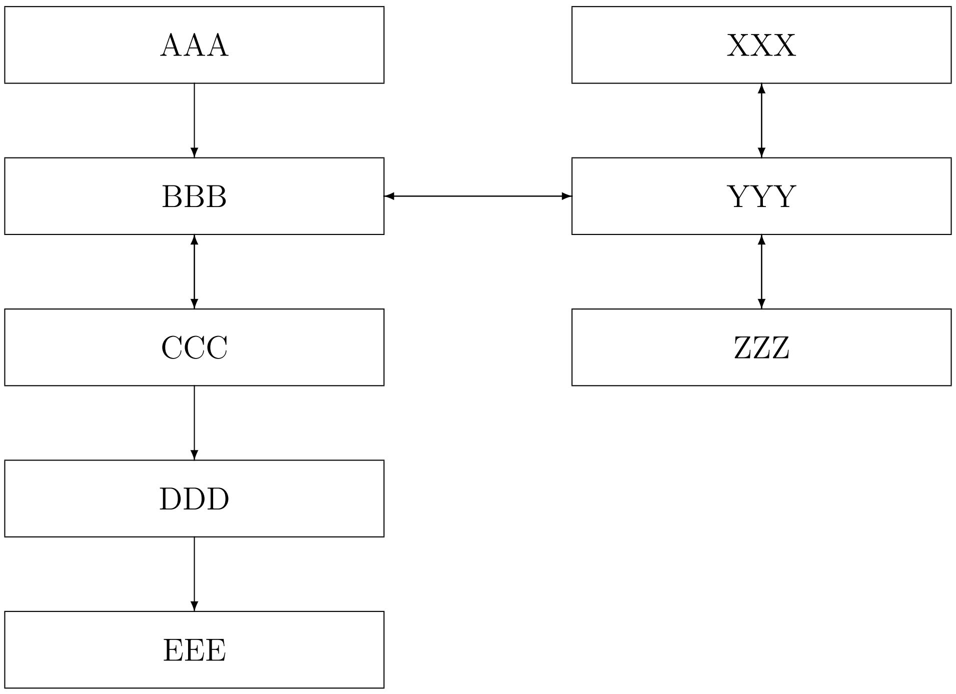

Figure 3.1: A picture composed of boxes and vectors.

3.2 Tables

Here is a simple example[1] of a table.

Left Justified	Centred	Right Justified
First	A	XXX
Second	AA	XX
Last	AAA	X

There is another example table in the proforma.

3.3 Simple diagrams

Simple diagrams can be written directly in LaTeX. For example, see figure **??** on page **??** and see figure **??** on page **??**.

3.4 Adding more complicated graphics

The use of LaTeX format can be tedious and it is often better to use encapsulated postscript (EPS) or PDF to represent complicated graphics. Figure **??** and **??** on page **??** are

[1]A footnote

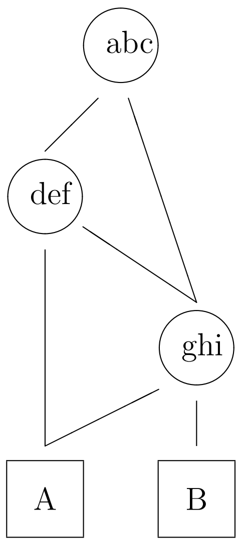

Figure 3.2: A diagram composed of circles, lines and boxes.

examples. The second figure was drawn using `xfig` and exported in `.eps` format. This is my recommended way of drawing all diagrams.

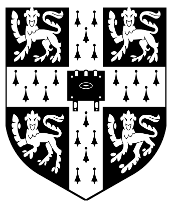

Figure 3.3: Example figure using encapsulated postscript

Figure 3.4: Example figure where a picture can be pasted in

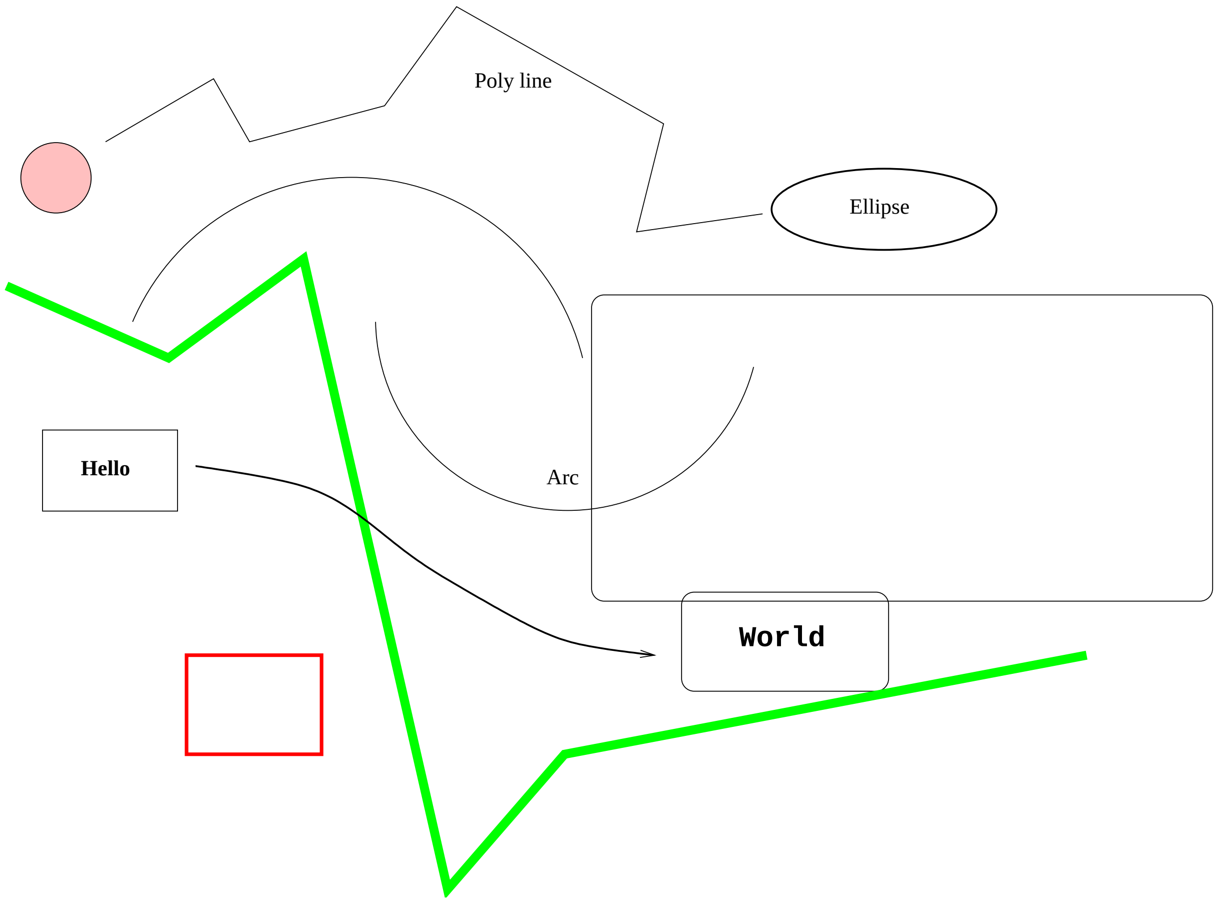

Figure 3.5: Example diagram drawn using `xfig`

Chapter 4

Evaluation

4.1 Printing and binding

Use a “duplex” laser printer that can print on both sides to print two copies of your dissertation. Then bind them, for example using the comb binder in the Computer Laboratory Library.

4.2 Further information

See the Unix Tools notes at
`http://www.cl.cam.ac.uk/teaching/current-1/UnixTools/materials.html`

Chapter 5

Conclusion

I hope that this rough guide to writing a dissertation is LaTeX has been helpful and saved you time.

Appendix A

Latex source

A.1 diss.tex

```
% Template for a Computer Science Tripos Part II project dissertation
\documentclass[12pt,a4paper,twoside,openright]{report}


\usepackage[pdfborder={0 0 0}]{hyperref}    % turns references into hyperlinks
\usepackage[margin=25mm]{geometry}  % adjusts page layout
\usepackage{graphicx}  % allows inclusion of PDF, PNG and JPG images
\usepackage{verbatim}
\usepackage{docmute}   % only needed to allow inclusion of proposal.tex
\usepackage{url}




\usepackage[UKenglish]{babel}% Recommended
\usepackage[bibstyle=numeric,citestyle=numeric,backend=biber,natbib=true]{biblatex}

\addbibresource{refs.bib}% Syntax for version >= 1.2


\raggedbottom                           % try to avoid widows and orphans
\sloppy
\clubpenalty1000%
\widowpenalty1000%

\renewcommand{\baselinestretch}{1.1}    % adjust line spacing to make
                                        % more readable

\begin{document}




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Title


\pagestyle{empty}

\rightline{\LARGE \textbf{Joshua Send}}

\vspace*{60mm}
\begin{center}
\Huge
\textbf{Conflict Free Document Editing with Different Technologies} \\[5mm]
```

```
Computer Science Tripos -- Part II \\[5mm]
Trinity Hall \\[5mm]
\today  % today's date
\end{center}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Proforma, table of contents and list of figures

\pagestyle{plain}

\chapter*{Proforma}

{\large
\begin{tabular}{ll}
Name:               & \bf Joshua Send                       \\
College:            & \bf Trinity Hall                      \\
Project Title:      & \bf Conflict Free Document Editing with Different Technologies \\
Examination:        & \bf Computer Science Tripos -- Part II, June 2017  \\
Word Count:         & \bf 1587\footnotemark[1] \\
Project Originator: & Joshua Send                    \\
Supervisor:         & Stephan Kollmann                    \\
\end{tabular}
}
\footnotetext[1]{This word count was computed
by \texttt{detex diss.tex | tr -cd '0-9A-Za-z $\tt\backslash$n' | wc -w}
}
\stepcounter{footnote}


\section*{Original Aims of the Project}


TODO\footnote{A normal footnote without the
complication of being in a table.}


\section*{Work Completed}

TODO

\section*{Special Difficulties}

TODO

\newpage
\section*{Declaration}

I, Joshua Send of Trinity Hall, being a candidate for Part II of the Computer
Science Tripos [or the Diploma in Computer Science], hereby declare
that this dissertation and the work described in it are my own work,
unaided except as may be specified below, and that the dissertation
does not contain material that has already been used to any substantial
extent for a comparable purpose.

\bigskip
\leftline{Signed TODO [signature]}

\medskip
\leftline{Date TODO [date]}

\tableofcontents

\listoffigures

\newpage
\section*{Acknowledgements}
```

```
TODO

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% now for the chapters

\pagestyle{headings}

\chapter{Introduction}

Real time interaction between users is becoming an increasingly important feature to many applications, from word processing


\section{Motivation}

Realtime collaborative editing was first motivated by a demonstration in the Mother of All Demos by Douglas Engelbart in 196

The convergence, or consistency, property above is the hardest to provide  it is easy to create a system where the last writ

This document was produced using \LaTeXe which is based upon
\LaTeX\cite{Lamport86}.  To build the document you first need to
generate \texttt{diss.aux} which, amongst other things, contains the
references used.  This if done by executing the command:

\texttt{pdflatex diss}

\noindent
Then the bibliography can be generated from \texttt{refs.bib} using:

\texttt{bibtex diss}

\noindent
Finally, to ensure all the page numbering is correct run \texttt{pdflatex}
on \texttt{diss.tex} until the \texttt{.aux} files do not change.  This
usually takes 2 more runs.

\subsection{The makefile}

To simplify the calls to \texttt{pdflatex} and \texttt{bibtex},
a makefile has been provided, see Appendix~\ref{makefile}.
It provides the following facilities:

\begin{description}

\item\texttt{make} \\
 Display help information.

\item\texttt{make proposal.pdf} \\
 Format the proposal document as a PDF.

\item\texttt{make view-proposal} \\
 Run \texttt{make proposal.pdf} and then display it with a Linux PDF viewer
 (preferably ``okular'', if that is not available fall back to ``evince'').

\item\texttt{make diss.pdf} \\
 Format the dissertation document as a PDF.

\item\texttt{make count} \\
Display an estimate of the word count.

\item\texttt{make all} \\
Construct \texttt{proposal.pdf} and \texttt{diss.pdf}.

\item\texttt{make pub} \\ Make \texttt{diss.pdf}
and place it in my \texttt{public\_html} directory.

\item\texttt{make clean} \\ Delete all intermediate files except the
source files and the resulting PDFs. All these deleted files can
```

```
be reconstructed by typing \texttt{make all}.

\end{description}


\section{Counting words}

An approximate word count of the body of the dissertation may be
obtained using:

\texttt{wc diss.tex}

\noindent
Alternatively, try something like:

\verb/detex diss.tex | tr -cd '0-9A-Z a-z\n' | wc -w/


\chapter{Preparation}

This chapter is empty!


\chapter{Implementation}

\section{Verbatim text}

Verbatim text can be included using \verb|\begin{verbatim}| and
\verb|\end{verbatim}|. I normally use a slightly smaller font and
often squeeze the lines a little closer together, as in:

{\renewcommand{\baselinestretch}{0.8}\small
\begin{verbatim}
GET "libhdr"

GLOBAL { count:200; all  }

LET try(ld, row, rd) BE TEST row=all
                        THEN count := count + 1
                        ELSE { LET poss = all & ~(ld | row | rd)
                               UNTIL poss=0 DO
                               { LET p = poss & -poss
                                 poss := poss - p
                                 try(ld+p << 1, row+p, rd+p >> 1)
                               }
                             }
LET start() = VALOF
{ all := 1
  FOR i = 1 TO 12 DO
  { count := 0
    try(0, 0, 0)
    writef("Number of solutions to %i2-queens is %i5*n", i, count)
    all := 2*all + 1
  }
  RESULTIS 0
}
\end{verbatim}
}

\section{Tables}

\begin{samepage}
Here is a simple example\footnote{A footnote} of a table.

\begin{center}
\begin{tabular}{l|c|r}
Left      & Centred & Right \\
Justified &         & Justified \\[3mm]
%\hline\\%[-2mm]
First     & A       & XXX \\
Second    & AA      & XX  \\
Last      & AAA     & X   \\
\end{tabular}
\end{center}

\noindent
There is another example table in the proforma.
\end{samepage}

\section{Simple diagrams}

Simple diagrams can be written directly in \LaTeX.  For example, see
figure~\ref{latexpic1} on page~\pageref{latexpic1} and see
figure~\ref{latexpic2} on page~\pageref{latexpic2}.

\begin{figure}
\setlength{\unitlength}{1mm}
\begin{center}
\begin{picture}(125,100)
\put(0,80){\framebox(50,10){AAA}}
\put(0,60){\framebox(50,10){BBB}}
\put(0,40){\framebox(50,10){CCC}}
\put(0,20){\framebox(50,10){DDD}}
\put(0,00){\framebox(50,10){EEE}}

\put(75,80){\framebox(50,10){XXX}}
\put(75,60){\framebox(50,10){YYY}}
\put(75,40){\framebox(50,10){ZZZ}}

\put(25,80){\vector(0,-1){10}}
\put(25,60){\vector(0,-1){10}}
\put(25,50){\vector(0,1){10}}
\put(25,40){\vector(0,-1){10}}
\put(25,20){\vector(0,-1){10}}

\put(100,80){\vector(0,-1){10}}
\put(100,70){\vector(0,1){10}}
\put(100,60){\vector(0,-1){10}}
\put(100,50){\vector(0,1){10}}

\put(50,65){\vector(1,0){25}}
\put(75,65){\vector(-1,0){25}}
\end{picture}
\end{center}
\caption{A picture composed of boxes and vectors.}
\label{latexpic1}
\end{figure}

\begin{figure}
\setlength{\unitlength}{1mm}
\begin{center}

\begin{picture}(100,70)
\put(47,65){\circle{10}}
\put(45,64){abc}

\put(37,45){\circle{10}}
\put(37,51){\line(1,1){7}}
\put(35,44){def}

\put(57,25){\circle{10}}
\put(57,31){\line(-1,3){9}}
\put(57,31){\line(-3,2){15}}
\put(55,24){ghi}
```

```
\put(32,0){\framebox(10,10){A}}
\put(52,0){\framebox(10,10){B}}
\put(37,12){\line(0,1){26}}
\put(37,12){\line(2,1){15}}
\put(57,12){\line(0,2){6}}
\end{picture}

\end{center}
\caption{A diagram composed of circles, lines and boxes.}
\label{latexpic2}
\end{figure}



\section{Adding more complicated graphics}

The use of \LaTeX\ format can be tedious and it is often better to use
encapsulated postscript (EPS) or PDF to represent complicated graphics.
Figure~\ref{epsfig} and~\ref{xfig} on page \pageref{xfig} are
examples. The second figure was drawn using \texttt{xfig} and exported in
{\tt.eps} format. This is my recommended way of drawing all diagrams.


\begin{figure}[tbh]
\centerline{\includegraphics{figs/cuarms.pdf}}
\caption{Example figure using encapsulated postscript}
\label{epsfig}
\end{figure}

\begin{figure}[tbh]
\vspace{4in}
\caption{Example figure where a picture can be pasted in}
\label{pastedfig}
\end{figure}


\begin{figure}[tbh]
\centerline{\includegraphics{figs/diagram.pdf}}
\caption{Example diagram drawn using \texttt{xfig}}
\label{xfig}
\end{figure}


\chapter{Evaluation}

\section{Printing and binding}

Use a ``duplex'' laser printer that can print on both sides to print
two copies of your dissertation. Then bind them, for example using the
comb binder in the Computer Laboratory Library.

\section{Further information}

See the Unix Tools notes at

\url{http://www.cl.cam.ac.uk/teaching/current-1/UnixTools/materials.html}


\chapter{Conclusion}

I hope that this rough guide to writing a dissertation is \LaTeX\ has
been helpful and saved you time.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the bibliography
```

```
\addcontentsline{toc}{chapter}{Bibliography}
\printbibliography

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the appendices
\appendix

\chapter{Latex source}

\section{diss.tex}
{\scriptsize\verbatiminput{diss.tex}}

\section{proposal.tex}
{\scriptsize\verbatiminput{proposal.tex}}

\chapter{Makefile}

\section{makefile}\label{makefile}
{\scriptsize\verbatiminput{makefile.txt}}

\section{refs.bib}
{\scriptsize\verbatiminput{refs.bib}}


\chapter{Project Proposal}

\input{proposal}

\end{document}
```

A.2 proposal.tex

```
% Note: this file can be compiled on its own, but is also included by
% diss.tex (using the docmute.sty package to ignore the preamble)
\documentclass[12pt,a4paper,twoside]{article}
\usepackage[pdfborder={0 0 0}]{hyperref}
\usepackage[margin=25mm]{geometry}
\usepackage{graphicx}
\usepackage{parskip}


\usepackage[UKenglish]{babel}% Recommended
\usepackage[bibstyle=numeric,citestyle=numeric,backend=biber,natbib=true]{biblatex}

\addbibresource{proposal_bibliography.bib}


\begin{document}

\begin{center}
\Large
Computer Science Tripos -- Part II -- Project Proposal\\[4mm]
\LARGE
Conflict Free Document Editing with Different Technologies\\[4mm]

\large
J.~Send, Trinity Hall

Originator: J.~Send

10 October 2016
\end{center}

\vspace{5mm}
```

```
\textbf{Project Supervisor:} S.~Kollmann

\textbf{Director of Studies:} Prof.~S.~Moore

\textbf{Project Overseers:} Prof.~T.~Griffin \& Prof.~P.~Lio

% Main document

\section*{Introduction}

\subsection*{Background}

In a world of ever increasing connectivity, collaborative features of applications
will take on greater and greater roles. Popular services such as Google Docs offer real-time
editing of documents by multiple users, a type of interaction that will move from being
a special offering by few applications to a common and expected interface.

The key property that must be implemented to achieve concurrent editing is eventual consistency,
meaning that all connected users should end up with the same result after receiving all changes to the document
--- even if edits conflict~\cite{Technion}. There are two main technologies that are used to enable concurrent editing of a
which that generally relies  on having a central server receive, serialize, transform, and
relay edits occurring simultaneously to each client. OT is notoriously difficult to implement
correctly as incoming operations have to be transformed against preceding ones on each client,
such that the result converges~\cite{sun1998operational}. The server may also be required to make some transformations.
Due to this, central server must be able to read all the operations being performed by clients.
Thus, unless the server is trusted and secure, OT-based services cannot provide any security or privacy guarantees.

The alternative, newer technology uses Conflict Free Replicated Datatypes (CRDTs). Instead of resolving conflicts and guaran
and possibly more scalability and efficiency.

This project is first concerned with exploring and developing a P2P CRDT concurrent text editor, and secondly comparing it t
listed in later sections.

\subsection*{Resources required}

The primary external resource I will need is the Javascript library ShareJS, which is published under the MIT license on Git

Additionally, I am developing on my personal computer, a Thinkpad T440s with 8 GiB of RAM,
128 GB of hard drive space, and a low wattage dual core Intel CPU running at 1.60GHz.
The primary development environment is Ubuntu Linux, though Windows 10 is also available
on the same machine.

Git with Github is used as both a version control system and a cloud backup. Dropbox
provides continuous cloud backups as well. Secondary development machines are any of the MCS computers.

\subsection*{Starting point}

I have some knowledge of the open source library ShareJS from a past internship, which I aim to leverage when
evaluating and comparing it to my system. My knowledge of CRDTs and the relevant adding/insert/merge
algorithms stems mostly from a high level explanation provided by Martin Kleppmann, along with a diagram.
This will be the starting point for my from-scratch implementation of the concurrent text editor.

Since I have no experience writing test cases and performance profiling, nor network simulation,
I will have to learn how to do these.

Lastly, I may consult various papers on CRDTs, as well as my supervisor's work in the area, if required.

\section*{Work to be done}

\subsection*{Overview}
I plan to implement a simulation of P2P CRDT text editing using Typescript. Following this, my project will focus on
comparing an existing OT-based concurrent document editing library (ShareJS) to my implementation, in order to draw conclusi
about their relative network and memory efficiency, and scalability. It is highly likely that my
system will need some optimization, which can feed back into my evaluation and comparison process. In the case that
these phases do not take too long, there are several possible extensions. The first would be to add
a networking layer to the simulation - in effect turning the it into a usable library. The second would be
researching and implementing 'undo' and 'move' operations, which are relatively open research problems.
```

```
\subsection*{Detailed Project Structure}

\begin{enumerate}

\item \textbf{Core CRDT Development:} Consider and decide CRDT datastructures. Then detail how I expect
the insert/delete/merge algorithms to work on paper, followed by implementing these. Lastly, I need to
learn frameworks for testing my implementation. The tests for correctness should include hand-crafted unit tests
to confirm expected behavior of intermediate execution steps and convergence of results across clients,
along with generated test loads to check correct convergence on all clients.


\item \textbf{Implement Simulation:} Model having an arbitrary number of clients each running the CRDT algorithms, and simul
networking between these clients. Because this is P2P, it may be worth adding functionality for a variety of network
topologies.

\item \textbf{Set up ShareJS and Compare:} Set up the ShareJS environment, mirror functionality and setups
between two systems as much as possible, and create corresponding performance profiling tests for both
systems. These will focus on network efficiency, memory usage, and scalability.

\item \textbf{Tune Implementation:} There will likely be opportunity for some optimization, which will
feed back into the performance comparisons in the previous step and help evaluate the optimizations themselves.

\item \textbf{Extensions:} The first extension is implementing a proper P2P network stack and remove the simulated
networking. Next would be researching undo and move operations and perhaps try to implement one or both of these.

\end{enumerate}



\subsection*{Possible extensions}

There are two extensions of varying difficulty:

\begin{itemize}

\item (Easier) Replace networking simulation with a P2P networking library. The end result of this extension should be a rea

\item (Difficult) Research prior work on undo and move functionality using CRDTs. If something suitable is found, implement

\end{itemize}


\section*{Success criteria}

These are the main success criteria associated with my project

\begin{enumerate}
\item A concurrent text editor based on CRDTs has been implemented.

\item The concurrent text editor passes all correctness tests.

\item Quantitative results comparing ShareJS and the CRDT based system have been obtained and analyzed.
\end{enumerate}


\section*{Timetable}

Planned starting date is 16/10/2011.

\begin{enumerate}

\item \textbf{Michaelmas weeks 2--3} Develop CRDT datastructure and algorithms on paper. Read into P2P networks
and simulating them.

\item \textbf{Michaelmas weeks 4--5} Lay out project files and implement network simulation with support for different
P2P topologies.
```

```
\item \textbf{Michaelmas weeks 6--8} Implement CRDT datastructures and algorithms, and connect these to network simulation.

\item \textbf{Michaelmas vacation} Learn an appropriate testing framework, write and generate unit tests for correctness of

\item \textbf{Lent weeks 0--1} Complete progress report. Mirror functionality of ShareJS to the setup
of my system. Start writing performance benchmarks and scalability tests for both systems.

\item \textbf{Lent weeks 2--4} Execute tests and analyze results. Try to explain differences and similarities observed.
Tune my implementation and evaluate various optimizations. Begin writing dissertation.

\item \textbf{Lent weeks 5--6} Continue writing dissertation and optimizing system. Begin research for extension
which implements proper networking stack.

\item \textbf{Lent weeks 7--8} Continue writing dissertation. Before terms ends, review and peer-review (including superviso
incomplete draft. Implement networking extension. Research undo and move operations with CRDTs.

\item \textbf{Easter vacation:} Finish dissertation draft. Work on undo and move extensions for system.

\item \textbf{Easter term 0--2:} Edit and proof read dissertation. Work on extensions.

\item \textbf{Easter term 3:} Proof read and then submit early to concentrate on exams.

\end{enumerate}

\newpage

\printbibliography

\end{document}


\end{filecontents}
```

Appendix B

Makefile

B.1 makefile

B.2 refs.bib

```
@misc{MotherDemo,
  title = {The Mother of All Demos, Reel 3},
  howpublished = {\url{https://archive.org/details/XD300-25_68HighlightsAResearchCntAugHumanIntellect&start=286}},
  note = {Accessed: 2017-03-01}
}

@article{Ellis1989,
author = {Ellis, C A and Gibbs, S J},
title = {{Concurrency Control in Groupware Systems}},
year = {1989}
}

@article{Gilbert2005,
author = {Gilbert, Seth and Lynch, Nancy},
pages = {51--59},
title = {{Brewer ' s Conjecture and the Feasibility of Consistent , Available , Partition-Tolerant Web Services}},
year = {2005}
}

@BOOK{Lamport86,
TITLE = "{LaTeX} --- a document preparation system --- user's guide
and reference manual",
AUTHOR = "Lamport, L.",
PUBLISHER = "Addison-Wesley",
YEAR = "1986"}

@REPORT{Moore95,
TITLE = "How to prepare a dissertation in LaTeX",
AUTHOR = "Moore, S.W.",
YEAR = "1995"}
```

Appendix C

Project Proposal

Computer Science Tripos – Part II – Project Proposal

Conflict Free Document Editing with Different Technologies

J. Send, Trinity Hall
Originator: J. Send
10 October 2016

Project Supervisor: S. Kollmann
Director of Studies: Prof. S. Moore
Project Overseers: Prof. T. Griffin & Prof. P. Lio

Introduction

Background

In a world of ever increasing connectivity, collaborative features of applications will take on greater and greater roles. Popular services such as Google Docs offer real-time editing of documents by multiple users, a type of interaction that will move from being a special offering by few applications to a common and expected interface.

The key property that must be implemented to achieve concurrent editing is eventual consistency, meaning that all connected users should end up with the same result after receiving all changes to the document — even if edits conflict [**Technion**]. There are two main technologies that are used to enable concurrent editing of a document (plain text or otherwise). One approach is Operational Transforms (OT), which that generally relies on having a central server receive, serialize, transform, and relay edits occurring simultaneously to each client. OT is notoriously difficult to implement correctly as incoming operations have to be transformed against preceding ones on each client, such that the result converges [**sun1998operational**]. The server may also be required to make some transformations. Due to this, central server must be able to read all the operations being performed by clients. Thus, unless the server is trusted and secure, OT-based services cannot provide any security or privacy guarantees.

The alternative, newer technology uses Conflict Free Replicated Datatypes (CRDTs). Instead of resolving conflicts and guaranteeing eventual consistency by transforming operations against each other, CRDTs use special datastructures that guarantee that no operations will conflict [**preguica2009commutative**]. There are many types of CRDTs that are tailored for different situations. One example is a simple up-down counter which could be implemented as two locally replicated registers, one for increments and one for decrements, where the current state is their difference[**Shapiro2011**]. Compared to OT, there is no interdependence between edits (as long as the network protocol can guarantee in-order delivery), which means CRDT-based systems can do away with the server and be implemented using peer to peer (P2P) protocols. This lends itself to security (encryption is now possible between endpoints), and possibly more scalability and efficiency.

This project is first concerned with exploring and developing a P2P CRDT concurrent text editor, and secondly comparing it to the OT-based client/server approach available in the open source library ShareJS. Several extensions are also possible, listed in later sections.

Resources required

The primary external resource I will need is the Javascript library ShareJS, which is published under the MIT license on GitHub [**ShareJS**].

Additionally, I am developing on my personal computer, a Thinkpad T440s with 8 GiB of RAM, 128 GB of hard drive space, and a low wattage dual core Intel CPU running at 1.60GHz. The primary development environment is Ubuntu Linux, though Windows 10 is also available on the same machine.

Git with Github is used as both a version control system and a cloud backup. Dropbox provides continuous cloud backups as well. Secondary development machines are any of the MCS computers.

Starting point

I have some knowledge of the open source library ShareJS from a past internship, which I aim to leverage when evaluating and comparing it to my system. My knowledge of CRDTs and the relevant adding/insert/merge algorithms stems mostly from a high level explanation provided by Martin Kleppmann, along with a diagram. This will be the starting point for my from-scratch implementation of the concurrent text editor.

Since I have no experience writing test cases and performance profiling, nor network simulation, I will have to learn how to do these.

Lastly, I may consult various papers on CRDTs, as well as my supervisor's work in the area, if required.

Work to be done

Overview

I plan to implement a simulation of P2P CRDT text editing using Typescript. Following this, my project will focus on comparing an existing OT-based concurrent document editing library (ShareJS) to my implementation, in order to draw conclusions about their relative network and memory efficiency, and scalability. It is highly likely that my system will need some optimization, which can feed back into my evaluation and comparison process. In the case that these phases do not take too long, there are several possible extensions. The first would be to add a networking layer to the simulation - in effect turning the it into a usable library. The second would be researching and implementing 'undo' and 'move' operations, which are relatively open research problems.

Detailed Project Structure

1. **Core CRDT Development:** Consider and decide CRDT datastructures. Then detail how I expect the insert/delete/merge algorithms to work on paper, followed by implementing these. Lastly, I need to learn frameworks for testing my implementation. The tests for correctness should include hand-crafted unit tests to confirm expected behavior of intermediate execution steps and convergence of results across clients, along with generated test loads to check correct convergence on all clients.

2. **Implement Simulation:** Model having an arbitrary number of clients each running the CRDT algorithms, and simulate networking between these clients. Because this is P2P, it may be worth adding functionality for a variety of network topologies.

3. **Set up ShareJS and Compare:** Set up the ShareJS environment, mirror functionality and setups between two systems as much as possible, and create corresponding performance profiling tests for both systems. These will focus on network efficiency, memory usage, and scalability.

4. **Tune Implementation:** There will likely be opportunity for some optimization, which will feed back into the performance comparisons in the previous step and help evaluate the optimizations themselves.

5. **Extensions:** The first extension is implementing a proper P2P network stack and remove the simulated networking. Next would be researching undo and move operations and perhaps try to implement one or both of these.

Possible extensions

There are two extensions of varying difficulty:

- (Easier) Replace networking simulation with a P2P networking library. The end result of this extension should be a ready to deploy Typescript (compiled to Javascript) library.

- (Difficult) Research prior work on undo and move functionality using CRDTs. If something suitable is found, implement it. Otherwise, attempt to work toward my own solution.

Success criteria

These are the main success criteria associated with my project

1. A concurrent text editor based on CRDTs has been implemented.
2. The concurrent text editor passes all correctness tests.
3. Quantitative results comparing ShareJS and the CRDT based system have been obtained and analyzed.

Timetable

Planned starting date is 16/10/2011.

1. **Michaelmas weeks 2–3** Develop CRDT datastructure and algorithms on paper. Read into P2P networks and simulating them.
2. **Michaelmas weeks 4–5** Lay out project files and implement network simulation with support for different P2P topologies.
3. **Michaelmas weeks 6–8** Implement CRDT datastructures and algorithms, and connect these to network simulation.
4. **Michaelmas vacation** Learn an appropriate testing framework, write and generate unit tests for correctness of implementation. Fix any bugs discovered by the testing process. Set up ShareJS environment. Begin outlining progress report.
5. **Lent weeks 0–1** Complete progress report. Mirror functionality of ShareJS to the setup of my system. Start writing performance benchmarks and scalability tests for both systems.
6. **Lent weeks 2–4** Execute tests and analyze results. Try to explain differences and similarities observed. Tune my implementation and evaluate various optimizations. Begin writing dissertation.
7. **Lent weeks 5–6** Continue writing dissertation and optimizing system. Begin research for extension which implements proper networking stack.
8. **Lent weeks 7–8** Continue writing dissertation. Before terms ends, review and peer-review (including supervisor) incomplete draft. Implement networking extension. Research undo and move operations with CRDTs.

9. **Easter vacation:** Finish dissertation draft. Work on undo and move extensions for system.

10. **Easter term 0–2:** Edit and proof read dissertation. Work on extensions.

11. **Easter term 3:** Proof read and then submit early to concentrate on exams.