

# **Developing a New Web Application for the Archive of Formal Proofs**

*Carlin MacKenzie*

**MInf Project (Part 1) Report**

Master of Informatics  
School of Informatics  
University of Edinburgh

2021

# Abstract

Formal proofs allow us to prove that a theorem is true over a domain. They can be built from axioms or built on top of other theorems. Computers can mechanically verify these proofs if they are written in a language such as Isar, which is the language of the proof assistant Isabelle. As proofs can be built on top of other theorems, it is useful to have a central repository which collects theorems that are free to use. Since 2004, this service has been provided for Isabelle by the Archive of Formal Proofs (AFP). The AFP is functional, however it is lacking in key areas such as navigation and search, as it has not been significantly updated since its inception.

We first reimplement the site generation using Hugo, before evaluating the current AFP with long term users. This helps us to understand what features matter to them and the problems that they have. Using this information, the site is redesigned using paper prototypes before being implemented with SCSS and Hugo templates. The functionality of the site is also extended with the addition of reactive search, related items, author pages and improved code navigation. Finally, the redesigned AFP is evaluated with users to discover whether it meets their needs. We find that all participants agree that the redesign is an improvement.

The result of this project is a website that is more useful to users, while being easy to maintain, and a published formal evaluation of the current AFP.

## **Acknowledgements**

I would like to thank Jacques Fleuriot and James Vaughan for their invaluable support and guidance with this project.

I would also like to thank my friends for supporting me throughout my time at university and my family for sowing the seeds that allowed me to be where I am today.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>3</b>
2.1	Formalization of Mathematics . . . . .	3
2.1.1	QED Manifesto . . . . .	3
2.2	Formal Proof Assistants . . . . .	3
2.2.1	Mizar . . . . .	4
2.2.2	Isabelle . . . . .	4
2.2.3	Coq . . . . .	4
2.2.4	Lean . . . . .	5
2.3	The Archive of Formal Proofs . . . . .	5
2.3.1	Features . . . . .	5
2.3.2	Design . . . . .	6
2.3.3	Directory Structure . . . . .	7
2.3.4	Entry Information . . . . .	7
2.4	Previous Work Involving the AFP . . . . .	7
<b>3</b>	<b>Evaluation of the Current Archive</b>	<b>9</b>
3.1	User Survey . . . . .	9
3.1.1	Pre-study . . . . .	9
3.1.2	Study . . . . .	9
3.2	Automated Audits . . . . .	11
3.2.1	W3C Validation . . . . .	12
3.2.2	Google Lighthouse . . . . .	12
3.3	Conclusion . . . . .	13
<b>4</b>	<b>Design</b>	<b>14</b>
4.1	Paper Prototypes . . . . .	14
4.1.1	Theme Colour . . . . .	14
4.1.2	Menu . . . . .	15
4.1.3	Home page . . . . .	15
4.1.4	Entry page . . . . .	16
4.2	Interactive Prototype . . . . .	17
4.3	Design Philosophy . . . . .	17
<b>5</b>	<b>Implementation</b>	<b>18</b>

5.1	Site Generation . . . . .	18
5.1.1	Overview . . . . .	19
5.1.2	Python Scripts . . . . .	19
5.1.3	Directory Structure . . . . .	20
5.1.4	Entry Information . . . . .	21
5.1.5	URL Structure . . . . .	21
5.2	Search . . . . .	22
5.2.1	FlexSearch.js . . . . .	22
5.2.2	FindFacts Integration . . . . .	23
5.2.3	Autocomplete Suggestions . . . . .	24
5.2.4	Search on Other Pages . . . . .	25
5.3	Navigation . . . . .	25
5.3.1	Taxonomies . . . . .	25
5.3.2	Related Entries . . . . .	26
5.4	Script Browsing . . . . .	26
5.4.1	SideKick . . . . .	27
5.5	Styling . . . . .	28
5.5.1	Validation . . . . .	28
5.5.2	Avoiding Tables for Layout . . . . .	28
5.5.3	Redesign . . . . .	29
5.5.4	Fine-tuning Cohesion . . . . .	30
5.6	Hosting . . . . .	30
5.6.1	Autogeneration . . . . .	31
5.7	The AFP in Machine Readable Format . . . . .	32
5.8	Conclusion . . . . .	32
<b>6</b>	<b>Evaluation of the New Archive</b>	<b>33</b>
6.1	User Evaluation . . . . .	33
6.1.1	Design . . . . .	33
6.1.2	Results . . . . .	34
6.2	Automated Audits . . . . .	35
6.3	Performance . . . . .	36
6.4	Maintenance . . . . .	36
6.4.1	Software . . . . .	37
6.4.2	Hugo . . . . .	37
6.5	Conclusion . . . . .	37
<b>7</b>	<b>Conclusion</b>	<b>38</b>
7.1	Suitability for Production . . . . .	38
7.1.1	Site Generation . . . . .	39
7.1.2	Continuous Integration . . . . .	39
7.1.3	Documentation . . . . .	39
7.1.4	Testing . . . . .	39
7.2	Future Work . . . . .	39
	<b>Bibliography</b>	<b>41</b>

<b>A</b>	<b>Screenshots of the Current AFP</b>	<b>44</b>
<b>B</b>	<b>Evaluation of the Current Archive—Pre-study</b>	<b>48</b>
	B.1 Design . . . . .	48
	B.2 Results . . . . .	49
<b>C</b>	<b>Evaluation of the Current Archive—Study Results</b>	<b>51</b>
<b>D</b>	<b>Paper Prototypes</b>	<b>62</b>
<b>E</b>	<b>Related Entry Graphs</b>	<b>66</b>
<b>F</b>	<b>Screenshots of the Redesigned AFP</b>	<b>69</b>
<b>G</b>	<b>Poster</b>	<b>73</b>
<b>H</b>	<b>Script for the Second Evaluation</b>	<b>74</b>

# Chapter 1

## Introduction

Isabelle [35] is an interactive proof assistant which allows users to write and prove formal proofs. As proofs can build on top of other proofs, the value of a theorem prover lies in the size of its library. Isabelle has a standard library<sup>1</sup> as well as collecting user submitted proofs in the Archive of Formal Proofs (AFP).

The entries of the AFP are reviewed similarly to a journal and there are annual releases of the AFP (which correspond with new versions of Isabelle). To date over 375 authors have contributed over 590 entries [10].

**Motivation** Unfortunately, the AFP has not been significantly updated since it first appeared online in 2004. As such there are many areas such as search, navigation and code browsing which we believed might require attention. Additionally, it has a non-responsive table-based layout which is typical of early 2000s web design. Finally, it uses a custom site generator which means that it is hard for outside contributors to improve the site.

**Objective** The goal of this project was to redesign and improve the AFP, guided by the priorities of the users. It should have feature parity with the current AFP, besides from submission, which is outside the scope of this project, plus new features which aid the users. The redesign should follow modern design conventions.

**Contribution** This project covers the following contributions:

- *Evaluation:* The current AFP was assessed with a structured survey by both pre-study and study groups. The latter survey was formally written up and published as a pre-print [22]. Additionally, the redesigned AFP was evaluated by a study group to understand if it meets their needs.
- *Site Generation:* To provide a foundation for the redesign, the site generation was recreated in Hugo by converting the site's data and creating templates to match the current website. On top of this, a continuous integration script was created to update the site daily.

---

<sup>1</sup><https://isabelle.in.tum.de/library/>

- *Redesign*: The site was redesigned using paper prototypes, before being implemented with Hugo templates and SCSS.
- *Search*: A new client-side search functionality was created which is responsive and has autocomplete suggestions. The search was then integrated with an external service, FindFacts, which provides additional results from the code of the AFP.
- *Script Browsing*: The script browsing experience was improved by allowing users to view all scripts for an entry on one page. Navigation was further improved with the addition of links to the lemmas.
- *Machine Readable Format*: The metadata of the entries was released so that it is accessible for future researchers.

**Organisation** Chapter 2 introduces the background of the current AFP and it is evaluated in Chapter 3. Following this, the redesign of the AFP is described in Chapter 4. Next, the implementation of the AFP is described in Chapter 5 and it is subsequently evaluated in Chapter 6. Finally, Chapter 7 concludes this project by summarising the results and providing an outline of future work.



# Chapter 2

## Background

This chapter contextualises my work by giving an overview of formal mathematics and proof assistants. The corresponding archives for each proof assistant mentioned is elaborated on. Finally the AFP is described, detailing its features and its site generation, as well as an overview of the existing literature.

### 2.1 Formalization of Mathematics

Humans have been reasoning about formal sciences, including mathematics, for thousands of years [7]. This is the process of creating logical systems in which axioms can be acted upon by rules. In this way, theorems can be guaranteed to be true based upon the logic of the system, rather than relying on evidence from the world. As these systems are based upon applying rules, it is possible for computers to validate these theories by applying the same rules systematically.

#### 2.1.1 QED Manifesto

The QED Manifesto [1] sketched out a project that aimed to formalise all of mathematics. This would mean that one could create new theorems which are rigorously true, without having to understand the minutia of what they are building upon. The resulting archive would be an open access and rigorously true set of all mathematical lemmas and techniques. Unfortunately, the project only lasted for 3 years [36], but the goals that it laid out live on in the AFP and other proof archives.

### 2.2 Formal Proof Assistants

Over the past 50 years, many formal proof assistants have been created in different mathematical systems and styles [12]. This section provides an overview of four major assistants that have large or rapidly growing proof libraries.

### 2.2.1 Mizar

The Mizar System [27] was one of the first proof assistants and was created in 1973. Proofs are written in a single script file in the Mizar language which is based on set theory. Proofs are mainly developed in MizarMode, an authoring environment for Emacs.

Mizar proofs are collected in the Mizar Mathematical Library (MML)<sup>1</sup> which was the largest formal maths library, as of 2009. It currently features 1,357 articles by 263 authors. Submissions are reviewed by three experts in a double-blind process. The MML is served as a downloadable archive and a quarterly journal, *Formalized Mathematics*. Searching of the library is provided by MML Query<sup>2</sup>, but it is in beta and currently seems to be broken. Each entry of the MML displays the author, summary, and the script file.

### 2.2.2 Isabelle

Isabelle [34] is a theorem prover that was first released in 1986. It is written in Standard ML [26] and users write their proofs in the structured proof language Isar [35], which is inspired by Mizar. Development of proofs is primarily executed through Isabelle/jEdit [33], and an extension is also available for VS Code. In comparison to most proof assistants, Isabelle is generic and allows for many different object logics such as Zermelo–Fraenkel (ZF) set theory or Higher Order Logic (HOL), the most popular.

Entries are collected in the Archive of Formal Proofs<sup>3</sup> and so far over 375 authors have contributed 590 entries. Submission to the AFP is dependent on review from one of the editors of the project. A thorough description of the Archive can be found in Section 2.3

### 2.2.3 Coq

Coq is written in OCaml and was released in 1989. Users write proofs in the Gallina language, which is based on the Calculus of Inductive Constructions [20], a type theory. Creation of Coq proofs are performed through the CoqIDE which is a GTK based editor.

Submission to the Coq Package Index<sup>4</sup> (CPI) is performed through GitHub pull requests and each package is reviewed by a developer of Coq before accepting. So far 308 people have contributed to 326 packages. The CPI has a responsive search interface which can be filtered with categories and keywords. Each entry of Coq is an independent GitHub repository owned by the “coq-community” organisation.

---

<sup>1</sup><http://mizar.org/library/>

<sup>2</sup><http://mmlquery.mizar.org>

<sup>3</sup><https://isa-afp.org>

<sup>4</sup><https://coq.inria.fr/opam/www/>

## 2.2.4 Lean

A new research project from Microsoft, Lean<sup>5</sup> is a theorem prover that was created in 2013 and is based on the Calculus of Constructions [9], a predecessor to the calculus used by Coq. It is written in C++ and the Lean language, which can be compiled to JavaScript, is used to write proofs. Extensions to aid creating proofs are available for Emacs and VS Code.

To date 157 people have contributed to the proof library, mathlib [24]. Contributing to mathlib is also managed through GitHub pull requests and each proof must be approved by a reviewer. Each entry is visible on the website as well as the GitHub repository which holds the entire mathlib. Search is provided by a Google SiteSearch, with the results rendered inline on the page.

## 2.3 The Archive of Formal Proofs

The Archive of Formal Proofs (AFP) is the online repository for Isabelle proofs. It first appeared on the Internet in 2004, hosted as a static site on SourceForge at <https://afp.sourceforge.net>. Since then it has taken residence on its own domain at <https://www.isa-afp.org>, however the visuals and functionality of the site have not been significantly updated since.

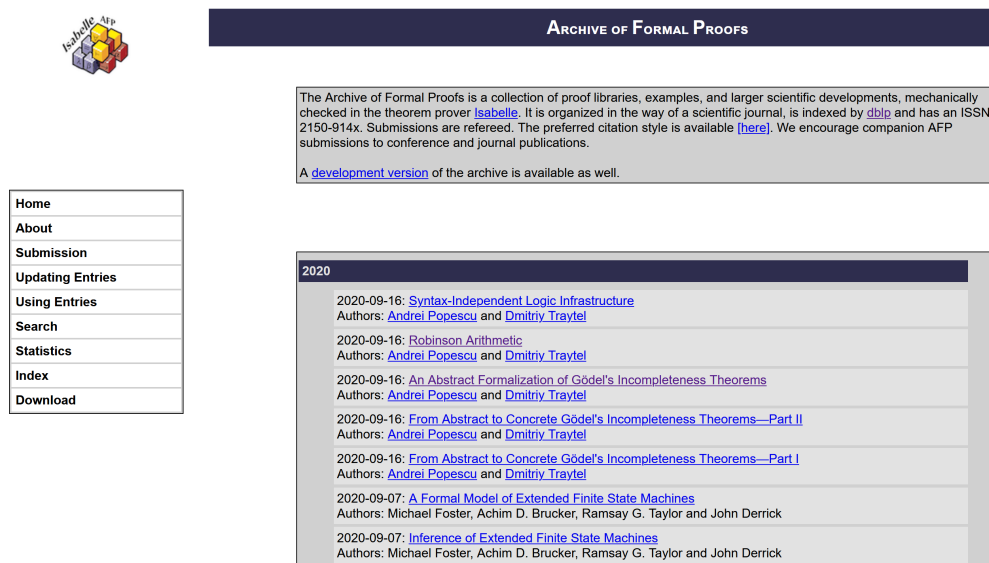


Figure 2.1: Archive of Formal Proofs homepage

### 2.3.1 Features

The home page of the AFP, depicted in Figure 2.1, lists all the entries with their authors in reverse chronological order. Each entry has its own page (as in Figure A.2) listing the abstract, license, entries it depends on, etc. Additionally, there are links to PDFs

<sup>5</sup><https://leanprover.github.io>

of the lemmas and code, links to download the entry and its previous releases, as well as a link to the HTML directory where the proof scripts can be browsed. These pages display the scripts with syntax highlighting applied (Figure A.3). There are no links on this page to other pages or back to its entry. No outline of the lemmas is available, so navigation is performed by either manual scrolling or using the browser’s “Find” feature.

Searching for entries, Figure A.4, is provided by a Google SiteSearch, which is a Google search with “site:isa-afp.org” appended. This experience is functional but relies on Google’s indexing of content which may be outdated or incomplete.

An index of topics is available which lists the entries in a hierarchy by topic. The topic of each entry can have up to three levels, for example “*Computer science/Algorithms/Distributed*”. Entries can be listed under multiple topics and so will appear multiple times. Unfortunately, the topic of an entry is not listed on its page. This means that if a user wanted to see other entries under the same topic as the one they are looking at, they would need to remember its name and find it on the index page.

Submission to the Archive is simple. Information about the entry is documented in a form, and the entry is attached as a .zip or .tar.gz archive. If review by an editor is successful, the entry is added to the Archive. However, from the maintainer’s perspective, the addition of an entry is a manual 11 step process.

### 2.3.2 Design

When the AFP was created, the only non-JavaScript way to create complex, structured layouts was to use tables. These layouts feature intricate nested HTML to define the structure of the page. For example, A very basic 2 row and 2 column table would have the following mark up:

```
<table>
  <tbody>
    <tr>
      <td>One</td>
      <td>Two</td>
    </tr>
    <tr>
      <td>Three</td>
      <td>Four</td>
    </tr>
  </tbody>
</table>
```

Care must be taken when changing the layout of the table to ensure that the number of columns are consistent across the rows. This means that it is not possible to make these tables responsive to the available screen width. Fortunately, more responsive and cleaner layouts are now achievable with CSS grid [32].

The structure of the site itself can also prevent users from engaging with the content fully. It is not possible to see all the proofs by an author, other proofs in this topic or the most frequently accessed proofs. Additionally, by directing users to search with

Google instead of a native solution, users cannot be sure that the results are complete— if search results are missed duplicate work could be unnecessarily performed.

These issues are likely due to the prioritisation of development time going towards Isabelle, and so AFP development is kept to maintenance work. Additionally, as the site is generated with custom Python scripts, it is difficult for people outside the development team to contribute.

It is important that the features of the AFP are improved so that users can be more productive and engage better with the contents of the Archive. Additionally, if the user experience and interface were to be improved, it is hoped that engagement with the Archive would increase, and so, encourage more proofs to be contributed.

### 2.3.3 Directory Structure

The Archive of Formal Proof follows the Unix directory structure and consists of the following:

- `admin` Site generation scripts and continuous integration configs.
- `doc` Documentation for managing the AFP.
- `etc` Various data files.
- `metadata` Jinja templates and data files for entries, topics, and release dates.
- `thys` Directories containing the session for every entry of the AFP.
- `tools` Various tools for checking and building the AFP (non-site-generation).
- `web` The generated static AFP website.

Site generation is performed by `admin/sitegen-lib/sitegen.py` which is a hand-written Python static site generator. It builds various Python objects for each page, which is then rendered with Jinja<sup>6</sup> templates.

### 2.3.4 Entry Information

The information about each entry can be found in `metadata/metadata`. This is an INI file which has a simple format with only two elements, `[sections]` and `key = value` pairs. Each entry of the AFP stores its information (apart from previous releases) in this 10,500-line file, which is used to generate the site. Figure 2.2 shows an example section of this file.

## 2.4 Previous Work Involving the AFP

This is the second undergraduate project from the University of Edinburgh which aims to improve the AFP. Goodwin [13] outlined the development life cycle and re-implementation of the Archive with modern frameworks. The project completely

---

<sup>6</sup><https://jinja.palletsprojects.com/>

```

-----
10004 [Relational_Disjoint_Set_Forests]
10005 title = Relational Disjoint-Set Forests
10006 author = Walter Guttman <http://www.cosc.canterbury.ac.nz/walter.guttman/>
10007 topic = Computer science/Data structures
10008 date = 2020-08-26
10009 notify = walter.guttman@canterbury.ac.nz
10010 abstract =
10011     We give a simple relation-algebraic semantics of read and write
10012     operations on associative arrays. The array operations seamlessly
10013     integrate with assignments in the Hoare-logic library. Using relation
10014     algebras and Kleene algebras we verify the correctness of an
10015     array-based implementation of disjoint-set forests with a naive union
10016     operation and a find operation with path compression.
10017
10018 [PAC_Checker]
10019 title = Practical Algebraic Calculus Checker
10020 author = Mathias Fleury <http://fmv.jku.at/fleury/>, Daniela Kaufmann <http://fmv.jku.at/kaufmann/>
10021 topic = Computer science/Algorithms
10022 date = 2020-08-31
10023 notify = mathias.fleury@jku.at
10024 abstract =
10025     Generating and checking proof certificates is important to increase
10026     the trust in automated reasoning tools. In recent years formal
10027     verification using computer algebra became more important and is
10028     heavily used in automated circuit verification. An existing proof
10029     format which covers algebraic reasoning and allows efficient proof
10030     checking is the practical algebraic calculus (PAC). In this
10031     development, we present the verified checker Pastèque that is obtained
10032     by synthesis via the Refinement Framework. This is the formalization
10033     going with our FMCAD'20 tool presentation.

```

Figure 2.2: Example section of the metadata file

overhauled the functionality and created a single page application with a database, log in and search. The final system is impressive and allows for entries to be submitted and changed in the browser. This system was not used as the foundation for this project as we did not want to use the client-server model they introduced because it would increase the maintenance load of the AFP.

Elsewhere, Huch and Krauss [19] have tried to improve the search facilities of the AFP (see Section 2.3.1). They recognised that searching for lemmas among all entries was impractical and aimed to provide this functionality. Consequently, they created an external website which allows users to query a search index of all code in the AFP. Queries can have complex filters and additional facets which reduces the search space of the query. As such, users can find specific lemmas of interest from the 2.9 million lines of Isar code which comprises the AFP.

Blanchette et al. [4] analysed the metrics of the AFP in response to questions about many areas such as entry reuse, composition of proofs and the impact of contributors. They gave a thorough overview of the AFP through mining its data and answer the aforementioned questions. For example, they discover 58% of the text of the AFP is taken up by proofs, 19% by lemma statements and 8% by definitions.

# Chapter 3

## Evaluation of the Current Archive

The evaluation of the current AFP serves two purposes. First, it informs us of its users' needs so that the redesign increases the sites utility. Second, it gives us a baseline to evaluate any redesign against, so that we can tell if improvements were made. This chapter covers both, the former in Section 3.1 and the latter in Section 3.2.

### 3.1 User Survey

There are many ways to elicit user feedback on an interface [16], however as we wanted to create a design which is useful for most users, the aggregate experience was of greater interest to us than the individual. Due to this, and the relatively large pool of users that could be participants, questionnaires were chosen as the method of evaluation.

#### 3.1.1 Pre-study

The survey was designed and validated on a smaller group from the *Artificial Intelligence Modelling Lab* in the School of Informatics. This group was chosen as the members are familiar with Isabelle across a variety of use cases and workflows. A full write up of this pre-study can be found in Appendix B. In summary, six people responded to the survey and they were not satisfied with the current AFP. Their largest problem was with searching for entries and theorems.

#### 3.1.2 Study

This *Isabelle Mailing List* was chosen as its members were likely to be users of the AFP, thereby increasing the likelihood of achieving a comprehensive evaluation of the website. The survey was advertised on the mailing list as *Survey on the AFP* with an estimated time of 10–20 minutes<sup>1</sup>. The time estimate was calculated based upon the average completion time of the pre-study. No compensation was advertised, and the

---

<sup>1</sup><https://lists.cam.ac.uk/pipermail/cl-isabelle-users/2020-November/msg00036.html>

main benefit of the study was to *help guide my research in evaluating the AFP*. The survey was open from 10 to 30 November 2020.

This study was published as a pre-print on ArXiv [22].

### 3.1.2.1 Design

The first section in the survey was “Demographics” so that the rest of the survey could be skipped if they were not eligible. The next two sections were “SUS” and “Pain Point” as these questions help us understand the user’s general thoughts and the most pressing issue that comes to mind. These were placed early in the survey so that respondents were not prompted about any specific area before answering. However, after reviewing the answers of the pre-study, several adjustments were made to address limitations of the initial design and to account for the main audience. This meant introducing the “Submission” questions which had to be asked immediately after the “Demographics” questions, due to limitations in the distribution platform, Microsoft Forms. Following this were several topics which had no constraint in ordering as they were independent to each other. The “Ranking” question was placed at the end of the survey as we wanted users to reflect on the areas which they had just considered. The final iteration of the survey was organised as follows:

1. **Demographics:** 4 questions to filter users into different groups depending on their experience with the AFP.
2. **Submission:** 2 questions to assess the submission process.
3. **System Usability Scale (SUS):** 9 SUS [5] questions to act as an indicator of the usability of the AFP.
4. **Biggest pain point:** 1 long answer question asking users to identify their biggest pain point.
5. **Navigation:** 4 questions related to the ease of finding pages and page visit frequency.
6. **Design:** 2 questions on the user interface and user experience.
7. **Browsing session scripts:** 2 questions about the browsing experience and 1 short answer question about missing features.
8. **Ranking priorities:** 1 question asking users to rank several areas in order of importance to guide development priorities.

### 3.1.2.2 Results

The survey was completed by 29 members of the mailing list who skewed towards long term and active users of the AFP. They were satisfied with the AFP in general, however they had specific criticisms about navigation, search, and theory browsing. Most respondents were neutral or negative towards a redesign of the user interface and user experience. Full results are available in Appendix C.



### 3.1.2.3 Analysis

It is likely that people who are subscribed to the Isabelle mailing list and willing to answer the survey are active AFP users. This was reflected in the demographics and the familiarity of the audience should be kept in mind when interpreting the results.

The survey was taken by 30 participants and 29 of them answered all the parts. From Fleuriot et al. [11], there were around 600 contributing users on the mailing list in the seven-year period of 2008 to 2015. We cannot tell whether the mailing list has grown or shrunk in the years since, but the number of responses seems adequate for the order of magnitude of the mailing list.

As the participants were mostly contributors to the AFP, their opinions are highly valued. The SUS score of 72 implies that they are generally satisfied with the AFP, which is a testament to the design decisions that have lasted almost 20 years. The pre-study score was much lower, 46, which seems to imply that non-contributors of entries to the AFP might be less satisfied. However a further study with a larger group would be needed to confirm this.

Three respondents described difficulty in finding existing functionalities and seven requested improvements to script search capabilities. Additionally, most participants struggle to find specific content in the AFP. As it was the second highest priority for users, the AFP would be more useful if the search capabilities were improved.

The most important thing for participants was navigation and the results of the survey imply that it does not currently meet their needs. The sidebar is the main navigation area and it is not ordered by frequency of use (Figure C.5), audience (contributor *vs* non-contributor) or content (i.e., “Home” and “Index” are the only pages which list entries and they are separate). Participants also report mis-clicking, which could suggest link labelling is not clear or links are too small. It is also hard to find many different types of content as shown in Figure C.4. Navigation is closely related to search, however, and many of these issues could be solved in other ways.

Finally, navigation improvements to script browsing were requested frequently—over half the respondents requested in-place links to entity definitions, i.e., to directly navigate to specific content. Similarly having an outline of the theory file, as SideKick provides in Isabelle/jEdit, was highly requested.

Whilst a significant minority of responses hold that a redesign is unnecessary, there were many specific criticisms with the current design as well as a general sentiment that several core features (specifically navigation, search, and theory browsing) could be improved.

## 3.2 Automated Audits

Many structural website issues can be detected automatically by validators and auditors [21]. They cannot detect all issues, nor large structural problems, however they are a good bellwether for detecting if best practices are followed. For the following audits,

each will be tested on the home page and an entry page, as these are the most frequently accessed pages. The entry used was *Separata* as MathJax is used in its abstract.

### 3.2.1 W3C Validation

The W3C Validator [31] is very basic and only checks whether the HTML syntax is correct. It is maintained by the World Wide Web Consortium, which is responsible for the HTML standard among many others.

#### 3.2.1.1 HTML Results

The errors found by the validator can be seen in Table 3.2

Home	Entry	
3	4	Uses of obsolete <code>font</code> element
6	10	Obsolete attributes on the <code>td</code> element
20	3	Obsolete attributes on the <code>table</code> element
1	1	Use of obsolete <code>align</code> attribute on the <code>div</code> element
1	1	Use of obsolete <code>border</code> attribute on the <code>img</code> element
1	1	Lack of <code>alt</code> attribute on the <code>img</code> element
1	0	Extra unopened <code>h1</code> tag
33	20	Total

Table 3.2: Issues with the AFP found by the W3C validator

The validator advised using CSS to fix all but the last two issues, which could instead be solved by fixing the HTML.

#### 3.2.1.2 CSS Results

All pages of the AFP use the same style sheet and the validator found 10 errors in it. Of these:

- 2 value errors for the `font` property
- 8 for non-existent values on properties (`vertical-alignment`, `text-transformation`, `text-alignment`, `text-indentation`)

All issues can be fixed by inferring the intent of the CSS and correcting it.

### 3.2.2 Google Lighthouse

Lighthouse [14] is a tool created by Google to help web developers assess their web pages. It can be run from the Chrome developer tools, or the command line, and generates a report for each URL that is provided. The report lists the result of five categories of automated tests, giving an overall score for how the website performed. In addition to this, it suggests several manual checks which should be performed to

cover aspects which cannot be automatically tested for. It should be noted that a perfect lighthouse score does not indicate that the website is fully accessible.<sup>2</sup> The Lighthouse report for the AFP is as follows:

	Home		Entry	
	Desktop	Mobile	Desktop	Mobile
Performance	80	73	80	78
Accessibility	70	70	87	87
Best Practices	93	93	87	87
SEO	70	58	70	58
Progressive Web App (PWA)	–	–	–	–

Table 3.4: Google Lighthouse metrics for the current AFP, out of 100

The high “Performance” score is coherent, as the page is very minimal with few external libraries and no tracking or ads. The score is not 100 due to the large DOM size, i.e., the page is very long and some nodes are deeply nested. The high “Best Practices” score is surprising but, upon reviewing, we found that the checks are mainly for correct HTML and for responsible JavaScript use, which the AFP conforms to.

The lower scores for Accessibility and SEO are less surprising due to many new guidelines being standardised for these in the years after the site’s creation.

The mobile scores are similar to the desktop scores despite there being no consideration for mobile devices.

PWAs are websites which are designed to function like apps on mobile phones. There is no score for this as the AFP cannot be installed, however this should not be seen as a negative as this technology does not have wide adoption.

In all, the issues are relatively minor and are fixed as noted in Section 6.2, or not relevant in the case of PWA.

### 3.3 Conclusion

From user and automated evaluation, we can see that there are flaws with the current AFP. These issues range from relatively minor invalid CSS issues, to lacking adequate search facilities which are crucial for user productivity and not duplicating work done previously. In the next chapter, we redesign the AFP to match modern design conventions before fixing these issues in the following implementation chapter.

<sup>2</sup><https://www.matuzo.at/blog/building-the-most-inaccessible-site-possible-with-a-perfect-lighthouse-score/>

# Chapter 4

## Design

As shown in Section 3.2 there are a number of structural flaws with the website, and based upon our subjective evaluation of the interface as being outdated, we chose to redesign the AFP. The survey results in Section 3.1.2.2 inform us that a complete redesign would not be welcome by the users and therefore creating a new, but familiar, interface would be more successful. Therefore we chose to create an interface which is faithful to the current design while using modern design conventions.

### 4.1 Paper Prototypes

Paper prototyping was chosen to test designs as it allows for quick iterations and easy modification. First, the original design was recreated in paper, and then the placement and form of each component was considered in turn.

#### 4.1.1 Theme Colour

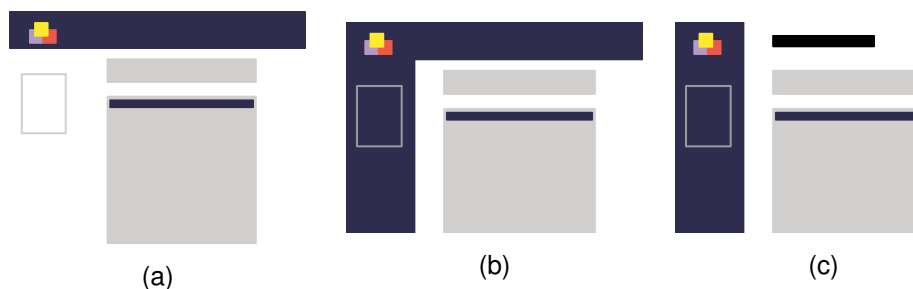


Figure 4.1: Mock-ups showing the three options for theme colour placement

The AFP uses a dark navy blue as its main theme colour which is present as the background of the page title (Figure 2.1). While iterating, the placement and size of this was considered.

At first, the banner was extended across to the left side of the page (Figure 4.1a), however it would visually separate the logo from the sidebar. This would leave the sidebar floating in space as it is currently. Hypothetically the sidebar could also have

the theme colour background (Figure 4.1b), however this would make the theme colour overwhelming.

Thus the chosen placement for the theme colour was the background of the sidebar (Figure 4.1c). This has several advantages as the dark colour ensures that your eyes stay on the content of the page rather than the side bar. Also, it links the sidebar to the logo ensuring that there is a strong connection between these items. One consideration is that the logo will need to have white text instead, but this was easy to create.

### 4.1.2 Menu

The Archive of Formal Proofs has always featured side navigation. As users disagreed with changing the user experience (Section 3.1.2.2), and to prevent users having to relearn the interface, we chose to preserve it.

As the side bar is being kept, greater focus was placed on the order and placement of the menu items. The current menu items and their attributes are shown in Table 4.1 and it is clear that there is little logic in the order of these items, as none of the attributes are grouped together. To resolve this, items relevant to contributors were separated from the main group. Then the remaining items were grouped by their content, and then arranged in descending order by their usage frequency, as people read menus from top to bottom [6]. Finally, the search page was imagined as a direct input and separated from the menu. The final menu groupings are shown in Table 4.2.

The labelling of the menu items was then considered. The “Index” page is the index of the topics of the entries, and the label “Topics” was chosen to reflect this. “Submitting” and “Updating Entries” are both items relating to “Contribution” so these were combined into one page. This centralises the information and preserves the number of clicks to reach either of these pages. Finally, “Using Entries” is descriptive, however some long form survey feedback expressed that there is a lack of help and documentation. This was renamed to “Help” and the content of the page would now cover many topics and link to external Isabelle resources.

	<b>Content</b>	<b>Audience</b>	<b>Use Frequency</b>
Home	List of entries	Everyone	Common
About	Facts	Everyone	Rare
Submission	Instructions	Contributors	Rare
Updating Entries	Instructions	Contributors	Rare
Using Entries	Instructions	Everyone	Sometimes
Search	Tool	Everyone	Common
Statistics	Facts	Everyone	Rare
Index	List of entries	Everyone	Common
Download	Links	Everyone	Sometimes

Table 4.1: Original menu

### 4.1.3 Home page

Most elements of the home page were preserved as it is functional. For example, the lack of pagination allows for the use of the browsers “Find” functionality.

	Content	Audience	Use Frequency
Home	List of entries	Everyone	Common
Index	List of entries	Everyone	Common
Download	Links	Everyone	Sometimes
Using Entries	Instructions	Everyone	Sometimes
Statistics	Facts	Everyone	Rare
About	Facts	Everyone	Rare
Search	Tool	Everyone	Common
Submission	Instructions	Contributors	Rare
Updating Entries	Instructions	Contributors	Rare

Table 4.2: New menus

Most improvements to this page were in simplifying the elements and reducing the visual clutter. For example, removing the table borders and the border around each entry opens the design up, and makes it easier to scan. To ensure that entries were still distinguishable without borders, white space was added. The date was simplified from a 2001-02-03 format to a 03 Feb format so that the year is not duplicated for every entry. Finally, the “Author: ” label was replaced with “by ” as this is more natural. The result of this can be found in Figure 4.2

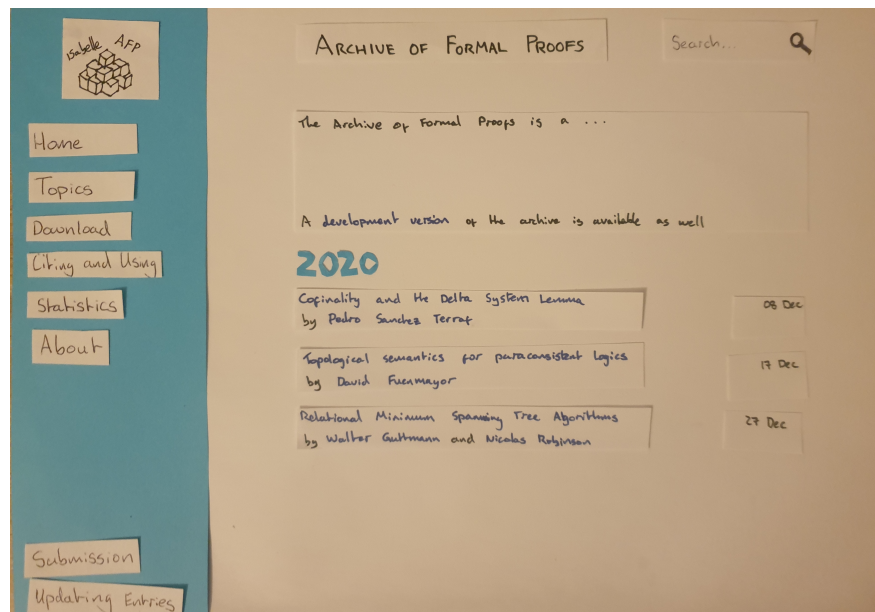


Figure 4.2: Final paper prototype of the home page

#### 4.1.4 Entry page

In contrast with the home page, more changes needed to be made to the entry page. The current interface is a very simple key-value table and all content has the same basic styling. Emphasis is given to the page title, but it is duplicated in the table directly beneath it. There are also associated links on this page however they are placed in a

table with a single column. The result is that a user cannot infer the content of the page from the structure, and instead must read to find the information that is needed.

The first changes were unpacking the table information and placing it where users would expect to find them. For example, the title is written in large bold text at the top of the page with the authors just underneath. The abstract becomes the main body text, and the license is written just below. The table of links is transformed into a bullet point list, while cite and download are highlighted with large buttons, signalling they are actions. The final design can be seen in Figure D.4.

## 4.2 Interactive Prototype

After the paper prototypes were satisfactory, we wanted to receive feedback on the designs before implementation. Rather than sending a PDF and describing how to use the interface, it was decided that an interactive prototype would be created which would be easier to share and demonstrate.

Figma<sup>1</sup> was used to create the prototype, as the main priority was speed of creation and it was already familiar. The interactive prototype was evaluated internally with my supervisors as the functionality of the new design was largely the same. A second iteration was performed to fix the issues raised in their feedback.

## 4.3 Design Philosophy

In order to create a familiar but new interface, we must consider the qualities of the current one which can be preserved—such as the tables. Table 4.4 shows how the qualities of a table are persisted into the redesign.

<i>Qualities of a table</i>	<i>How they are preserved in the redesign</i>
Horizontal lines	Horizontal lines underline <h1> headings to emphasise them. Em dashes “—” are used instead of bullet points
Vertical lines	The sidebar creates a vertical line which separates it from the content
Corners and intersections	Buttons and borders have sharp right angle corners
Alignment of content in rows and columns	All content is aligned on an underlying grid

Table 4.4: Preserving the design philosophy of the current AFP in the redesign

<sup>1</sup><https://www.figma.com>

# Chapter 5

## Implementation

The redesigned AFP features a new site generator, user interface, search interface, script browsing interface and new ways to navigate. This chapter describes how the redesigned AFP was created in detail, justifying design decisions that were made.

### 5.1 Site Generation

As Goodwin [13] demonstrated, the client-server model can successfully distribute the AFP. This architecture, however, would conflict with the goals of this project because the introduction of a database and a web server would increase the maintenance load. Consequently it was decided that the site generation for the redesign should continue to be static.

Before the AFP could be redesigned, site generation must be understood so that changes can be made. The publicly available codebase was briefly examined to see how extensible and maintainable the current solution was. It was clear that the code, while well structured, was not capable of competing with existing static site generators (SSGs). Additionally, the AFP's custom-based site generator is a likely source of overhead when it comes to contributing to the project, as people would need to understand it before they can contribute. This is especially exacerbated by the current generator integrating with Scala and Isabelle for dependency generation—however this was found to be unnecessary, and the functionality was re-implemented without Isabelle in Section 5.1.2. Thus, it was felt that migrating to a standard SSG would allow people to bring their existing knowledge to the project more readily.

As of 2021, there are many competing static site generators which can be broken up into two main categories. On the one hand, a popular paradigm resides in JavaScript based site generators, such as Next.js<sup>1</sup> and Gatsby<sup>2</sup>. These are powerful and are commonly combined with APIs, to allow for logins and payment, in what is known as the JAMstack [3]. On the other hand, there are SSGs in other languages with more tradi-

---

<sup>1</sup><https://nextjs.org>

<sup>2</sup><https://www.gatsbyjs.com>



tional support for templating, like Hugo (Go)<sup>3</sup> and Jekyll (Ruby)<sup>4</sup>. These generators take Markdown [15] content files and insert the content into templates. Jekyll is championed for being easy to learn, whereas Hugo takes longer to learn but is more powerful [23]. Additionally, Hugo excels in its speed due to being written and templated in Go, which is statically typed and compiled. For these reasons Hugo was chosen as the SSG to re-implement the AFP in.

The first step of the project was to re-implement the site generation in Hugo, in preparation for the redesign. An initial prototype with very few entries was first created as a proof of concept to figure out the best structure for the site. Many of the Jinja templates were partially reused as they have a similar syntax to Hugo templates. Subsequently, Python scripts were created to generate the content files for each entry.

### 5.1.1 Overview

The original site generation is described in Section 2.3.3, and the following structure is used to generate the new AFP:

1. Update the `thys/` directory and `metadata/metadata` file—these are the only files from the upstream repository that are needed to update the Hugo site.
2. Run `exportMetadata.py` to update the Hugo content files—Section 5.1.2.
3. Build the site using Hugo.

### 5.1.2 Python Scripts

Python was used to convert the original sites data into files suitable for Hugo generation. Most of the scripts iterate over the list of entries in the `thys/` directory, and so, they could be refactored into one script so that this iteration is not repeated multiple times. This design was not chosen however, as we wanted to keep the scripts modular so that they were self-contained and did not have side effects. As such, each script can be run by itself if required. Brief details for each are given below:

**exportMetadata.py** The purpose of this script is to run the rest of the scripts in the correct order and provide feedback in the form of a progress bar.

**iniToJson.py** This script primarily converts the metadata stored in an INI file into individual JSON files. The shortname, title, date and abstract are preserved as is, but the other attributes are transformed into more appropriate formats like arrays and objects. Author emails are extracted from the entry data and are collated into an `authors.json` file.

**addOlderReleases.py** This script traverses the `metadata/release-dates` and `metadata/releases` files and adds all the releases (except the most recent) of each entry to its JSON file.

---

<sup>3</sup><https://gohugo.io>

<sup>4</sup><https://jekyllrb.com>

**addDependencies.py** The dependencies of an AFP entry are listed in the ROOT file, and as it is regular [34], this script uses a regular expression to extract the dependencies and adds them to the JSON file of the entry.

**addRelatedEntries.py** This script generates related entries, using three metrics as described in Section 5.3.2, and adds these to the entries to improve site navigation.

**generateKeywords.py** This script generates the list of keywords for the search auto-complete. Each entry's abstract is sanitised and then the keywords are extracted with the RAKE algorithm. This script is described in detail in Section 5.2.3

**exportJsonMetadata.py** `metadata.json` is a JSON release of the AFP's metadata which is generated by this script and is described in Section 5.7.

**addStatistics.py** Most of the statistics for the site, like number of authors and most used entries, are generated by Hugo. However, some statistics like number of lines in the AFP are generated by the scripts from the current AFP. This is because the implementation of these is non-trivial and was not a priority in the time available. Unlike the previously mentioned scripts which take 5–10 seconds to execute, this takes 60 seconds due to duplicate computation that is discarded.

Finally, `getTheories.py` is instead run rarely due to high network load on the upstream site.

**getTheories.py** This script downloads and transforms the HTML documents for the theory browsing, which is detailed in Section 5.4.1.

### 5.1.3 Directory Structure

The new Hugo site generator has the following structure:

- `archetypes/default.md` A file which describes the structure for pages created with `hugo new entry`. This is optional and pages can be created manually.
- `assets/theories/` HTML files for each entry which contain the concatenated theories.
- `content/` Markdown files for the non-entry pages (home, about, search, etc.).
  - `entries/` Markdown files for each entry in the AFP, described in Section 5.1.4.
  - `theories/` Markdown files which list the lemmas of each theory for generating the menu, described in Section 5.4.1.
- `data/` JSON files which contain data about the authors, topics, and statistics.
- `static/metadata.json` The JSON release of the Archive's metadata.
- `themes/afp/` Where the site's theme is stored. Hugo allows for multiple themes, but we only use one for the redesign.
  - `assets/sass/main.scss` The SASS for the website which is compiled to CSS upon build.

- layouts/ HTML templates for each section and page type.
- static/ JavaScript, fonts and images for the website.
- config.json The Hugo config contains the site metadata and describes how the site should be built.

### 5.1.4 Entry Information

Hugo has support for structured data to be associated with a Markdown file in the form of *frontmatter*. This structured data is a dictionary of key-value pairs and can be one of three formats: YAML, TOML, or JSON. When the site is generated, this metadata can be referenced by key and rendered on the page. This is a natural choice for representing the entries of the AFP as it is already stored in a key-value format. This means that while entries are stored in Markdown files, they only contain a JSON blob which contains the information for the entry. For example, this entry from metadata/metadata on the left is stored in the file content/AVL-Trees.md:

```
[AVL-Trees]
title = AVL Trees
author = Tobias Nipkow <http://www...
date = 2004-03-19
topic = Computer science/Data structures
abstract = Two formalizations of AVL t...
extra-history =
  Change history:
  [2011-04-11]: Ondrej Kuncar added ...
notify = kleing@cse.unsw.edu.au

{"title": "AVL Trees",
 "authors": [
   "Tobias Nipkow",
   "Cornelia Pusch"
 ],
 "date": "2004-03-19",
 "topics": [
   "Computer science/Data structures"
 ],
 "abstract": "Two formalizations of A...",
 "extra": {
   "Change history": "[2011-04-11] ..."
 },
 "notify": [
   "kleing@cse.unsw.edu.au"
 ],}
```

There are many advantages to this. Each entry is self-contained and can store all the information related to that entry. We can extract author information to its own JSON file reducing duplication and inconsistencies. The three formats available are more powerful than the INI format that is currently used. This means we can store arrays instead of strings which need to be parsed into lists. Of the three formats, JSON was chosen as it is the only one of these which has a module in the Python Standard Library.

### 5.1.5 URL Structure

It is good practice to write URLs in such a way so that they never change [2]. For instance, information that can change such as authorship, file name extension and status should not be included in the URL. Most URLs in the current AFP violate the file name extension rule, and other URLs are structured in unintuitive ways. An overview of the current and redesigned AFP's URLs is shown in Table 5.1

All the URLs in the new AFP have lowercase, hyphenated URLs without file exten-

Current Home Path	/
New Home Path	/
Current Entry Path	/entries/AVL-Trees.html
New Entry Path	/entries/avl-trees/
Current Browse Theories Path	/browser_info/current/AFP/AVL-Trees/
New Browse Theories Path	/entries/avl-trees/theories/
Current Theory Path	/browser_info/current/AFP/AVL-Trees/AVL.html
New Theory Path	/entries/avl-trees/theories/#AVL

Table 5.1: Comparison of the URL paths in the current and redesigned AFP.

sions as this is the default behaviour of Hugo<sup>5</sup>. File extensions are avoided by serving all pages at `page/index.html` rather than `page.html`.

In comparison to the current AFP, the theory pages can be found at an obvious sub-directory of the entry page. This was relatively complex to set up and a discussion was made in the Hugo forum to figure out how this could be done<sup>6</sup>. In the end, the URL for each theory page is set manually in the frontmatter by `getTheories.py`.

## 5.2 Search

The current search facility of the AFP relies on a Google SiteSearch, which, as mentioned previously, is a Google search for the phrase with “`site:www.isa-afp.org`” appended. This is an example of a server-side search as the user sends a request to the server, the server searches the index, and returns the result. This is the most common search paradigm as it enables searching large indices. This search is useful, however could be incomplete or outdated depending on Google’s indexing. Due to this, a new native search facility for the AFP was created.

In contrast, a client-side search will instead have the server send the index to the client and searches are performed locally. The benefits of this are faster results due to the lack of network requests, and this has become a popular method as smartphones and computers have become more powerful. As the number of entries in the AFP is relatively small, we chose to implement a search on the client that is responsive and fast.

### 5.2.1 FlexSearch.js

`FlexSearch.js`<sup>7</sup> was the chosen search framework as it provided features such as tokenization, stemming, and autocomplete.

On page load, several JSON files containing the search index are fetched and then loaded into their own individual FlexSearch instance, as shown below. This allows for

<sup>5</sup><https://gohugo.io/content-management/urls/#pretty-urls>

<sup>6</sup><https://discourse.gohugo.io/t/complex-use-of-taxonomy-or-subpage-generation/30692>

<sup>7</sup><https://github.com/nextapps-de/flexsearch>

custom parameters to be set per index, as well as faster search as they can be searched in parallel.

```
var entryIndex = new FlexSearch({
  encode: "advanced",
  tokenize: "forward",
  doc: {
    id: "id",
    field: ["title", "abstract"],
  },
});

var topicIndex = new FlexSearch({
  encode: "icase",
  tokenize: "forward",
  doc: {
    id: "id",
    field: "name",
  },
});
```

**Highlighting Results** Pre-attentiveness is the quality of being able to very rapidly and accurately detect visual stimulus that “pops out” of the page [17]. By changing the background colour of the search term in the results, we can use this effect to make it easier to scan and check for relevant results (Figure 5.1). This is implemented using `mark.js`<sup>8</sup> which is an 18kb library which highlights matching text in a container.

## 5.2.2 FindFacts Integration

Huch and Krauss created the FindFacts service [19] which allows for searching definitions, lemmas, and constants across all 2.5 million lines of Isabelle code which form the AFP. Logical operators can be used, as well as filters on the expected type, to narrow the search.

This is very useful although it is not advertised anywhere on the current AFP. Rather than just linking to it, it would be beneficial if this service were available on the AFP itself. However as the current FindFacts implementation is satisfactory, we chose to integrate it into the search results as another index—showing the number of results and linking to them in the AFP interface.

A prototype was made to do this, but the FindFacts server initially responded with an error due to its *Cross Origin Request Security* policy. Consequently, a request was made to the developers to adjust this policy to allow queries from other websites. This request was successful, and we subsequently implemented the search as shown in Figure 5.1.

Debouncing is applied to the search to prevent overloading the server with queries. This means that the script waits for the user to stop typing for 300 milliseconds before sending it. The trade-off is that the search appears slower due to the delay, however this is not an inconvenience compared to accidentally overloading the server.

To further reduce server load, requests are cached client-side by memoising the query. This means that every query is checked against the local cache before making a request to the server. If it is not found, then the request is made, and the returned result is added to the cache so that the same request is not made again in the future.

---

<sup>8</sup><https://markjs.io>

## Search the Archive

graph | Search

- graph theory
- directed graph
- random graphs
- undirected graphs
- weighted graphs

Entries

**Graph Saturation** 2018  
 Sebastiaan J. C. Joosten  
 This is an Isabelle/HOL formalisation of graph saturation, closely following a paper by the author on graph saturation. Nine out of ten lemmas of the original paper are proven in this formalisation. The formalisation additionally includes two theorems that show the main premise of the paper: that consistency and entailment are decided through graph saturation. This formalisation does not give executable code, and it did not implement any of the optimisations suggested in the paper.  
[Logic/Rewriting](#) and [Mathematics/Graph theory](#)

**Graph Theory** 2013  
 Lars Noschinski  
 This development provides a formalization of directed graphs, supporting (labelled) multi-

Topics

- [Mathematics/Graph theory](#)
- [Computer science/Algorithms/Graph](#)

FindFacts Results

- [1081 Constants](#)
- [2384 Facts](#)
- [63 Types](#)

Figure 5.1: Search page of the redesigned AFP

### 5.2.3 Autocomplete Suggestions

FlexSearch.js does provide autocomplete functionality however it does not work as expected. It generates matches in the index up to a limit<sup>9</sup> but does not guarantee that these words will be keywords across many documents. Therefore, a novel solution had to be created to generate a list of keywords to offer as suggestions.

Rapid Automatic Keyword Extraction (RAKE) [29] was used to extract keywords. In comparison to other methods, which discard stop words, RAKE instead splits the text on the stop words, as keywords often do not contain stop words themselves. The algorithm then rates the words on how often they co-occur and favours keywords that appear in longer phrases.

For each entry in the AFP, the abstract is sanitised and then RAKE is used to get a list of keywords. The parameters used with RAKE were a 3-letter minimum character length and a 2-word maximum for keywords. These were chosen as 1–2 letter keywords are faster to type than to read from a suggestion, and keywords with more than 2 words will match few documents. The top 8 keywords from the abstract are added to a list of keywords—8 was chosen as it preserved infrequent words like “Godel” but rejected keywords like “accompanying paper”. After this we have a list of 3,741 keywords from all abstracts. We then filter this list and remove all keywords that only appear once, as we do not want to suggest a term with only one result. This reduces the list to 460 keywords. Finally, we remove plural keywords where we also have the singular version, as the singular will return the plural. The final list has 435 keywords.

The list of generated keywords was added as a search index to FlexSearch. The results

<sup>9</sup><https://github.com/nextapps-de/flexsearch#suggestions>

from searching this are then added to a `<div>` which appears below the search box and supports keyboard interaction.

## 5.2.4 Search on Other Pages

The paper prototypes included a search bar on every page of the AFP to make searching easier (Section 4.1.2). As such, a trimmed down version of the search script was created. This script would only give autocomplete suggestions and redirect users to the search results when they pressed enter. Thus, there are two search scripts `search.js` and `header-search.js`, the former being loaded only on the search page, and the latter being loaded on every other page.

## 5.3 Navigation

The current ways to navigate the Archive are as follows:

- List of entries on the homepage.
- List of entries by topic on the “Index” page.
- Links in the sidebar to several single pages.
- Authors will be linked to their website if they have one.
- Entries will link to entries they depend on or entries which depend on them.

The redesign preserves all of these, however adds several new ways to navigate via Hugo taxonomies and related entries.

### 5.3.1 Taxonomies

In Hugo, taxonomies generate pages which are indexed on a value for a key in the frontmatter. In other words, for a list of entries as on the left, we can set authors to be a taxonomy and additionally generate pages which list the entries like on the right:

```
entries/
  entry-1.html
  authors: author-1, author-2
  entry-2.html
  authors: author-1
authors/
  author-1.html
  entries: entry-1, entry-2
  author-2.html
  entries: entry-1
```

The redesigned AFP adds three taxonomies: authors, topics, and dependencies. Each one of these lists all the entries that have the same author, topic, or dependency. Links to these pages can be found on the entry pages, as well as the home page in the case of authors. Author pages also link to the author’s website if they have provided it.

The root of the taxonomy (i.e., `/authors/index.html`) lists all items in the taxonomy, i.e., a list of authors, topics, or dependencies. It is using this list of authors that the number of authors in the statistics is generated.

### 5.3.2 Related Entries

As well as the taxonomies, the redesigned AFP can be navigated via related entries. We see entries as related if they share dependencies, topics, or keywords.

To generate the related entries, each entry of the AFP is iterated over to create three dictionaries as follows:

```
dependencies = {"dependency": [list-of-entries, ...], ...}
keywords = {"keyword": [list-of-entries, ...], ...}
topics = {"topic": [list-of-entries, ...], ...}
```

All keywords which feature in more than 10 entries are dropped as these keywords are seen as too general for this purpose. For the next step, each dictionary is assigned a weighting of how strongly it indicates relatedness. These scores were chosen based upon the specificity of the category—dependencies are highly specific to an entry, keywords are limited to 10 entries, and topics have no limit.

```
dependencyModifier = 1.5
keywordModifier = 1
topicModifier = 0.5
```

Next a dictionary is created with the structure shown below. For each of the categories, the list of entries associated with each key is iterated over twice and, if the entries are not the same, the modifier of that category is added to the relatedness score between the two entries in the dictionary. As the loop iterates twice over the value set, the resulting dictionary is bijective—i.e., the scoreValues below will be the equal.

```
relatedEntries = {
  "entry-1" : {"entry-2": scoreValue, ...},
  "entry-2" : {"entry-1": scoreValue, ...},
  ...
}
```

Once this dictionary is created, all the relations which have score less than or equal 2.5 are dropped. This means that for entries to be considered related, they must at least have 2 shared dependencies; a dependency, keyword, and topic; etc.

Finally, the top three highest scoring relations for each entry are chosen to be its related entries and these are written to each entry file. The result of this is 194 relations between entries, and a selection of these are visualised as a graph in Figure 5.2

## 5.4 Script Browsing

The current script browsing experience, shown in Figure A.3, is basic and consists of a directory page and pages for each script. If a user is to find something specific in the files, they must open each one individually and use the browsers “Find” feature.

The new script browsing experience instead concatenates the theory files together so that they can be displayed on one page. The code which does this downloads each theory page for each entry from the live website, as the HTML pages are not stored in a public repository.



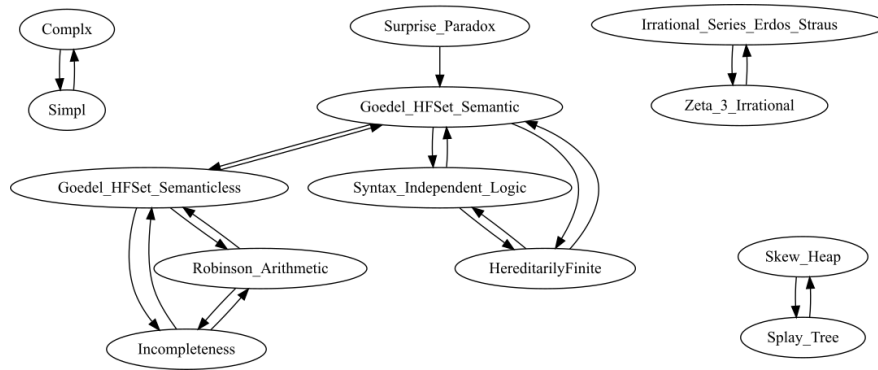


Figure 5.2: A selection of four related entry clusters. They are all visible in Appendix E

### 5.4.1 SideKick

One of the most highly requested features of the AFP is the addition of “SideKick”. This is a plugin for Isabelle/JEdit which surfaces the outline of the currently open script file for navigation, visible in Figure 5.3.

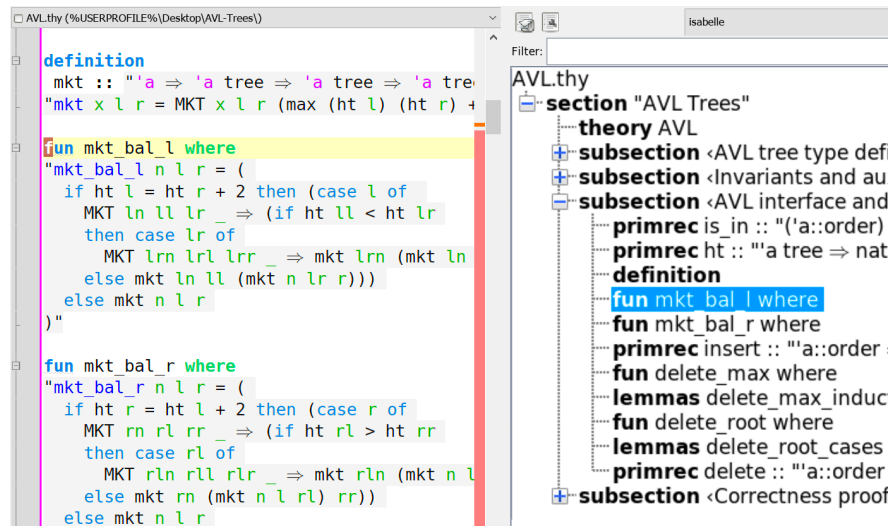


Figure 5.3: The SideKick available in Isabelle/JEdit

To recreate this functionality, a script was created which attaches unique IDs to the lemma elements of the theory.

For every entry, the script downloads the “Browse theories” page to get a list of theories. The theories are then downloaded, transformed, and concatenated together. The first transformation is to keep the `<body>` and change it to be a `<div>`, as there can only be one body tag in a document. The next transformation is to select all lemmas in the document and add unique IDs to them. The resulting HTML and lemma names are returned to be added to the theory’s front matter.

Consequently, Hugo generates the menu with theories and lemmas from the theory front matter, and users can scroll to them by clicking the links. This is visible in Figure 5.4.

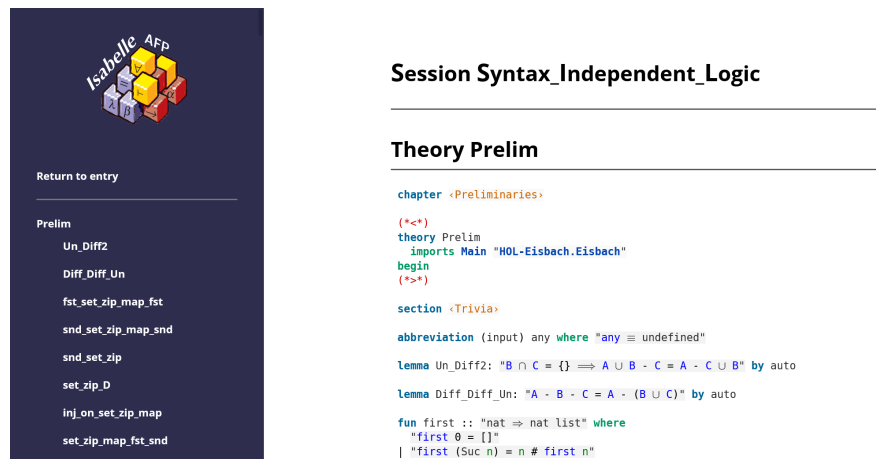


Figure 5.4: Example theory page showing the new SideKick style navigation

## 5.5 Styling

This section discusses the approach taken when implementing the design created in Chapter 4. First, structural issues with the CSS were resolved, then the HTML tables were swapped out for CSS grid. At this point, the website was redesigned one component at a time, before final design tweaks were made to improve cohesion.

### 5.5.1 Validation

As mentioned in Section 3.2.1.2, there were many errors in the CSS syntax itself, hence the first priority was to correct these. When this was being fixed, it was realised that a lot of the styling was overly verbose and unnecessary—i.e., explicitly setting values to their default. These were removed or simplified, but it required full concentration to ensure that the rules were preserved correctly.

It was at this point that it was realised this could be automated with a CSS preprocessor, which could standardise and minify the CSS [28]. After some research, the `cssnano` module of the PostCSS<sup>10</sup> tool was used to produce valid CSS which was simplified and standardised.

### 5.5.2 Avoiding Tables for Layout

The current AFP is composed of nested tables. There is a table which holds the sidebar and the content, and they are themselves composed of tables. This was a very common design pattern<sup>11</sup> before CSS3 introduced flexbox (2014) and grid (2018).

Thus, as grid allows for greater flexibility than tables, the current table-based design was converted to use grid instead. This was easy and provided many benefits, like simplifying the HTML markup and making maintenance easier.

<sup>10</sup><https://postcss.org/>

<sup>11</sup>[https://en.wikipedia.org/wiki/Holy\\_grail\\_\(web\\_design\)](https://en.wikipedia.org/wiki/Holy_grail_(web_design))

Both this and the previous section would be recommended, simple improvements to the AFP even if a redesign were rejected by the maintainers.

### 5.5.3 Redesign

After finalising the paper prototypes from Section 4.1, each component of the website was implemented in turn, in the same way as the paper prototypes had been iterated. The HTML was simplified to match the semantics of the layout, as this results in more accessible pages by default [8]. The CSS was altered to style the components to match the prototype. An early example of the home page can be seen in Figure 5.5 and the final version can be seen in Figure 5.6.

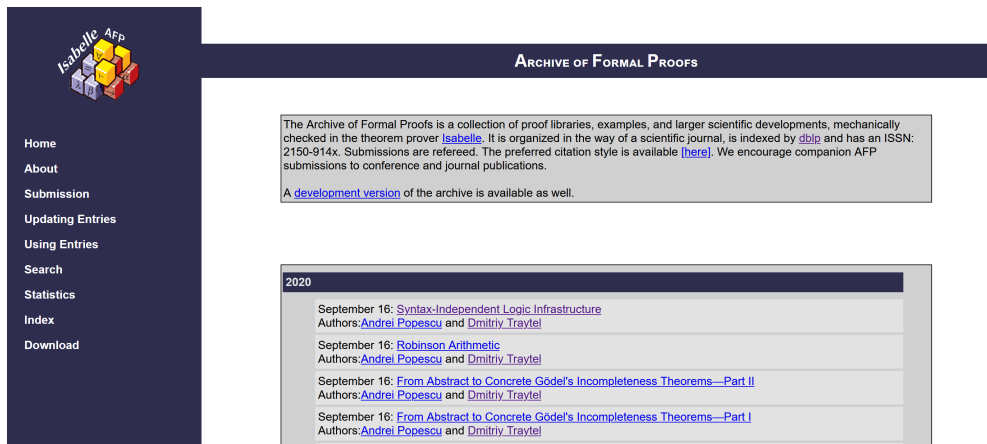


Figure 5.5: First iteration of the home page redesign

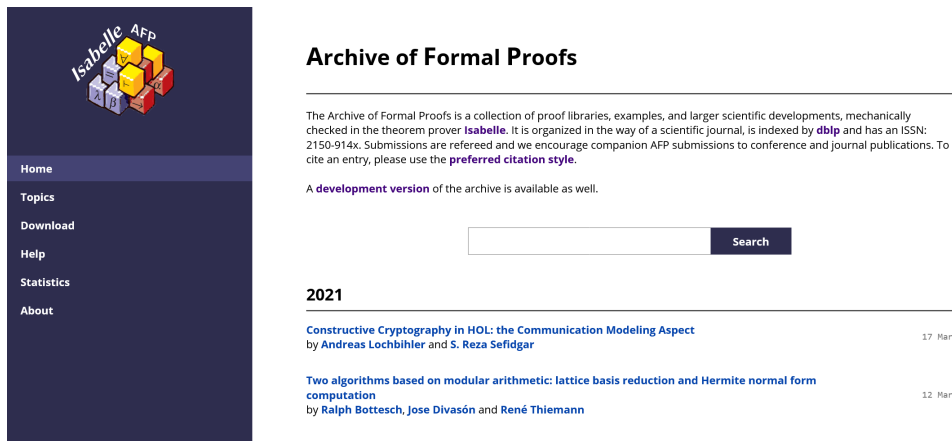


Figure 5.6: Final iteration of the home page redesign

#### 5.5.3.1 Font

When testing the AFP across browsers and systems, it was realised that the font choice was platform dependent. The AFP only specifies a `sans-serif` font, so the default sans-serif for that browser/operating system is used. Cross-platform inconsistencies can lead to hard to diagnose problems and degraded design depending on the fonts

available. Therefore, it is best to define one font so that the experience is the same everywhere.

The font was chosen to be kept similar to the current neutral style that operating systems' default fonts have. Fonts that were heavily associated with a brand were discarded, such as Roboto by Google or Fira Sans by Mozilla. In the end, Open Sans was chosen as it is a neutral and easy to read font which is commonly used as a default on the web [30].

## 5.5.4 Fine-tuning Cohesion

After the redesign, there were some outstanding inconsistencies in the look of buttons and headings. For example, download links, buttons with images and buttons with text should have looked similar but did not.

Test pages were created with these elements so they could be styled more cohesively. However, the CSS became hard to work with as these elements shared styles, but it was tedious to compose them while preserving a coherent source order in the CSS.

### 5.5.4.1 SASS

SASS<sup>12</sup> is a preprocessor language for CSS which enables developers to write succinct and powerful CSS [25]. There are two syntaxes available, SASS and SCSS, the former removing the need for parentheses and semi-colons and the latter being most like CSS, which is why it was chosen. The extra utility provided by the language is that it allows for nested CSS rules, which keeps all the styling of each element, and its children, in one place.

To ease the transition from CSS, `css2scss`<sup>13</sup> was used to convert it into SCSS. It was chosen as it allowed for customisation of the output, especially creating variables for colours and formatting them. The result was a SCSS file which compiled to the same CSS but was easier to reason about. For example:

- It reduced the number of colours by combining similar colours under one intuitive name.
- It becomes obvious that some properties only need to be set in one place, like `text-decoration`
- Source order is preserved, but rules are grouped by the first selector—i.e., `header h1` is grouped with the `header` rules rather than the `h1` rules.

## 5.6 Hosting

Originally, an early version of the website was hosted on a sub-domain of my own website<sup>14</sup> as this sub-domain was configured, not being used and a way to demonstrate

---

<sup>12</sup><https://sass-lang.com>

<sup>13</sup><https://sebastianpontow.de/css2compass/>

<sup>14</sup><https://beta.carlinmack.com>

my progress was needed. However, as more and more content was added, worries arose about the cost of hosting and it became harder to upload to my server.

GitHub pages give free hosting for any public repository which has it enabled, so it was a natural choice for my project as it was already hosted on GitHub. Originally a repository was created, *afp-alpha*, but unfortunately the URL of the index page would then be `https://carlinmack.github.io/afp-alpha/index.html`. This would mean that absolute links could not be used unless they were prefixed with `/afp-alpha/`. The site root can be set to a path in Hugo, however we did not want to add more complexity that would have to be undone if the site was hosted on its own domain. Fortunately, GitHub allows for users to have a repository that serves pages from the root rather than a path. Thus the website of the new redesign is simply `https://carlinmack.github.io/`.

### 5.6.1 Autogeneration

To keep the site up to date, an automated workflow was created to monitor the upstream repository and generate the new site as appropriate. This was implemented as a GitHub Action and is triggered daily as follows:

1. Check out the static site repository, set up Python and install dependencies.
2. Get the SHA of the `metadata/metadata` file of the upstream repository. If this is different to the stored SHA, continue, else exit.
3. Checkout the site generator repository.
4. Download the `thys/` and the `metadata/metadata` file. This is all the files required to update the site, so the repository does not need to be cloned.
5. Overwrite `thys/` and `metadata/metadata` in the site generator repository.
6. Install dependencies for site generation script.
7. Run site generator.
8. Commit changes to the site generator repository.
9. Set up Hugo.
10. Build static site and output in the static site directory from step 1.
11. Commit changes to the static site repository.
12. Clean up files.

When the upstream repository has been updated, the workflow takes around 3.5 minutes, and the largest proportion of this time (1.5 minutes) is the unavoidable wait to download the `thys/` directory. If the new site generation were merged into the upstream repository these files would already be available, and thus large time savings could be made. Exporting the metadata takes roughly 7 seconds and building the site takes 50 seconds. The performance of the AFP is compared to the original in Section 6.3.

## 5.7 The AFP in Machine Readable Format

The content of the AFP can be downloaded wholesale from the website, however the metadata is only available in the HTML pages. To fix this discrepancy, a JSON release of the metadata was created. This is an array of JSON objects with the authors' emails removed for privacy and the related entries removed as they are not found in the original data. A static version of the metadata, which corresponds to Isabelle2021, will soon be released on Zenodo. Additionally, the most recent version of the dataset can be downloaded from the downloads page. Consequently, the data can be used more readily for research or other use cases.

## 5.8 Conclusion

This chapter presented the implementation of the redesign and new features such as search, navigation and code browsing. In the next chapter we evaluate this implementation to ascertain whether it is an improvement over the current AFP, and if it meets the needs of the users.

# Chapter 6

## Evaluation of the New Archive

The success of this project is gauged upon whether it meets the original goal set out: To create a website that is easier to use, guided by the priorities of the users. This chapter evaluates the usability with users, structural issues with automated audits, performance compared to the current Archive and, finally, an overview of what needs to be maintained.

### 6.1 User Evaluation

The success of the redesign can only be evaluated by users of the AFP. Due to the short time available we could not distribute the original survey from Chapter 3 with a new focus on the redesign. Instead, we chose to perform a lab study with a small number of people from the *Artificial Intelligence Modelling Lab*.

#### 6.1.1 Design

A mixed approach was used to get a variety of information. The first part of the study was a think-aloud as it allowed the participants to get acquainted with the AFP and allowed me to see how users naturally used the new design. It contained 6 tasks which each asked the participant to use a different facet of the AFP. For example, the first task was “Visit the “Ordinal Partitions” entry and copy its BibTeX citation.”. This entry was chosen as it was not published recently and we wanted to see if they would use the browsers “Find” functionality or if they would use the search. The second part of the survey was a 7-question multiple-choice to gain quantitative feedback about whether the design is successful. Finally, open ended questions were asked to prompt a discussion of the new design, regarding whether it is an improvement and if there is anything missing.

A script (Appendix H), questionnaire, participant information sheet and consent form were prepared and reviewed by my supervisor. A quick pre-study was performed with a fellow undergraduate student to ensure that the study would go smoothly. Although they were unfamiliar with the AFP and the redesign, they were able to easily complete all the tasks. This is not generalisable, however it is a good indicator of clear design.

## 6.1.2 Results

Five people were individually asked to take part, four responded and were subsequently interviewed. The interviews were performed on 17–18 March 2021 and lasted 21, 12, 28 and 26 minutes in order.

No participant struggled with the think-aloud tasks and they almost always completed the tasks in the way they were designed to be completed—i.e., by using the search feature instead of the browsers “Find” feature; by using the download button in the search results rather than on the entries page. All participants were comfortable using the top search box. There were a few things of note during the think-aloud:

- One participant was pleasantly surprised that the search was responsive.
- One participant tried using the prefix “author:” to search for the author, which is assumed to be learned behaviour from similar sites.
- Three of four participants did not notice the FindFacts search results in the sidebar on the right at first.
- One participant was confused that all the theories were on the same page, and wanted to be able to pop the theories out into their own page. This is because they are used to using the keyboard navigation keys, like “Home”, to navigate in the theory files.

1. For each of the following, please mark the box that best describes your reaction to the statement

[More Details](#)

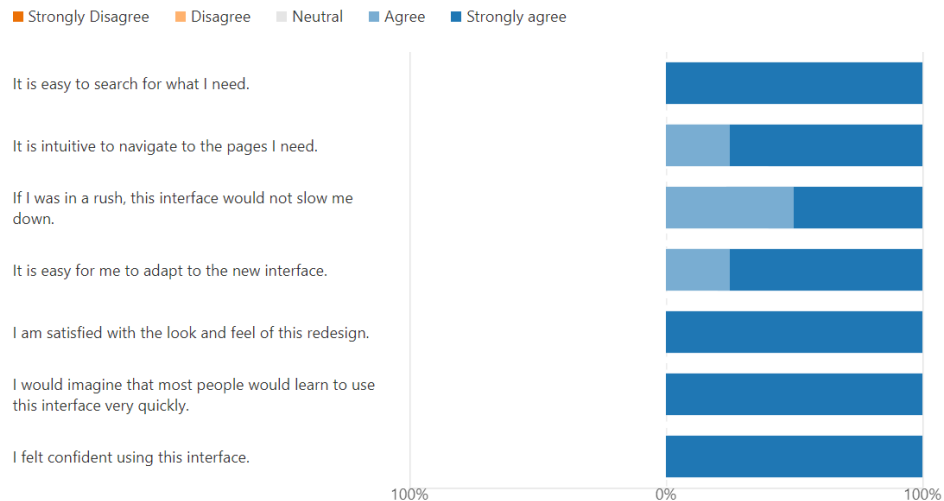


Figure 6.1: Short survey results

The short survey was answered during the call and participants stopped sharing their screen before filling it in. As shown in Figure 6.1, the participants agree or strongly agree with all the questions asked. Of note, all participants agree that it is easy to adapt to the new interface and it is quick to learn to use it.

Finally, the participants answered the long answer questions. The first question was:



*Is this an improvement over the current AFP? How so or how not?*

All participants thought the redesign was an improvement.

**Participant 1** They particularly highlighted the prominence of the search in the redesign and that it was good to not rely on Google. They felt the redesign was more “streamlined” and that the cite and download buttons “pop out”.

**Participant 2** They enjoyed the search bar on the landing page and that it is interactive. Additionally, they appreciated the ability to click through to theories.

**Participant 3** They felt that the redesign was a “massive improvement” and “fantastic”. They commented that there is “no background pollution”, the interface is “friendlier” and “a bit more professional”.

**Participant 4** They felt it was “undoubtedly” an improvement and expressed that searching is a “pain” in the current AFP.

*Does this redesign meet your needs? Is there anything lacking or missing?*

All participants thought their needs were met overall. There were several features related to search which would be appreciated in future work.

**Participant 1** Their main need was to find entries on a specific topic, and felt that Google may provide better results for things which were not exact text matches, but still related.

**Participant 2** Overall their needs are met, as they mainly use the search to do literature reviews. They like the new author pages, but would like to search within theory files.

**Participant 3** They felt that their needs were met as they mostly just use the search and it is “good”. They would like for entries to also match on the author name, rather than just showing the authors in the sidebar.

**Participant 4** Their main need is search and this is mostly met, however they recognised that searching for lemmas is beyond the scope of this project.

## 6.2 Automated Audits

To ensure that there is no degradation of the website, the same automated audits from Section 3.2 were performed on the redesign.

**W3C Validation** Both the home and entry pages have valid HTML and CSS.

**Google Lighthouse** The Lighthouse results for the redesigned AFP are shown in Table 6.2 and there is no degradation in the scores compared to the current AFP. .

The accessibility score is just off perfect due to having many heading elements with the same level. This is because all the entry titles have h5 elements for the title, as this is recommended by the W3C.

	Home		Entry	
	Desktop	Mobile	Desktop	Mobile
Performance	82	73	100	98
Accessibility	98	98	99	99
Best Practices	100	100	100	100
SEO	100	100	100	98
Progressive Web App (PWA)	–	–	–	–

Table 6.2: Google Lighthouse metrics for the redesigned AFP, out of 100

The performance score stays low due to the large size of the home page, listing every entry in the AFP. As the search function is lacking in the current website, the exhaustive listing is beneficial so that the browsers “Find” function can be used. However, as the redesign features the search input prominently on the front page, it may be possible to introduce pagination in the future.

The redesigned AFP is still not a PWA, so this score is still blank.

### 6.3 Performance

The performance of the current and redesigned AFP can be seen in Table 6.4. It is a better comparison of performance when the theory pages are not generated, as the current generator does not output them. Significant time is added by generating these pages as 143 of them are greater than 2MB. The metrics show that generation with Hugo is at least 4 times faster.

	Time (sec)	# Pages	Pages/sec	Size (MB)	MB/sec
Current AFP	48–79	602	8–13	4	0.05–0.09
Redesigned AFP	44–53	2,506	47–57	951	17.9–26.1
<i>without theories</i>	20–22	1,913	80–96	26	1.2–1.3

Table 6.4: Comparison of the performance of site generation in the current and redesigned AFP.

The file size of the pages of the AFP are also generally smaller. The current homepage is 191KB versus 168KB for the redesign. This is due to the move away from table-based layouts and Hugo generating minified HTML files.

### 6.4 Maintenance

If the redesign replaces the current design, it is important that someone will be able to fix the site if it breaks. In this section, we outline the amount of work the redesign requires to be maintained.

### 6.4.1 Software

The redesigned AFP depends on the following programming languages and libraries.

- Hugo  $\geq$  0.81
- Python  $\geq$  3.3: requests, tqdm, bs4, unidecode
- JavaScript: mark.js, FlexSearch.js

All these requirements are easy to satisfy as none require compilation or version management.

### 6.4.2 Hugo

In comparison to the previous site generator, Hugo brings several benefits to maintainability. First, the template syntax is very similar to Jinja which is used in the current AFP. Second, people can bring their outside knowledge to help improve the site, as it is a common tool which uses familiar paradigms. Finally, the generator itself is unlikely to break, as maintenance is handled by the Hugo developers.

One caveat with Hugo is that there is a history of updates with substantial breaking changes. For example, the 0.60.0 update changed the default behaviour of Markdown pages to omit included HTML instead of rendering it<sup>1</sup>. They made this change to close a potential security liability, and you could disable the new behaviour by adding a line in the config file. Unfortunately, this caused ire in the community as the new behaviour was not made clear enough. This means that maintainers should be aware when upgrading Hugo and read the changelog if any warnings or errors appear on the first build.

## 6.5 Conclusion

From our evaluation with users and automated audits, we can see that the redesign of the AFP has been successful. All participants agree that the redesign is an improvement and there is no regression in the scores of any automated audit. The performance of site generation has also been improved and the maintenance load is not greater than the current AFP. In the next chapter we conclude the project, summarising and evaluating our contributions and outlining the scope of future work.

---

<sup>1</sup><https://discourse.gohugo.io/t/raw-html-getting-omitted-in-0-60-0/22032/11>

# Chapter 7

## Conclusion

In this report a new design of the AFP has been created in response to user feedback. This involved re-implementing the site generation, paper prototyping and redesigning the website. In addition, several features were added such as improved code navigation and responsive search. These features increase the utility of the website for users. Furthermore, the site auto-updates with each change to the AFP and thus can replace the existing website.

As Hugo is used to generate the site, the maintenance of the generator is off-loaded to the Hugo community. This site generator is performant and builds the 2,500 pages of the site in 50 seconds—which is 4–12x faster than the current generator (Section 6.3). Upon evaluation, it was found that this new design met the needs of the users and was a major improvement upon the current website. As such, this project met the goal set out.

This project helped me to improve as a software developer, especially regarding management of workload, delivering on time and in communicating the outcome of my work. Since this project was started from scratch, it required knowledge in many areas and helped me to develop my skills and learn new ones. In particular, Hugo is now something which I am proficient in and I feel comfortable using the most advanced features which it has to offer.

Finally, this project was presented at the *Honours Project Day* and the poster that was created for it can be found in Appendix G.

### 7.1 Suitability for Production

While it is hoped that the redesign will one day replace the current AFP (as users suggest that it is an improvement, Section 6.1.2) there are several areas which need attention before it can be released.

### 7.1.1 Site Generation

Due to backwards compatibility, the site is generated from the previous structure of the AFP as detailed in Section 5.1.2. This is so that the site can be updated as needed, however it means that there are several pre-processing steps which are unnecessary. If the new site were to replace the current AFP, users would still have to edit the previous metadata files to update their entries—negating the value of having separate JSON files.

Before deployment, we would need to check that the browsers of the target audience are still supported. If this is a problem, build scripts can be used to replace the new features with backwards compatible and prefixed versions for older browsers.

### 7.1.2 Continuous Integration

The site is currently generated with a build script that checks out the various repositories and generates the new site. This script is brittle, due to the specificity of generating the site, but is functional for demonstration purposes. In a production scenario, the generation should be integrated with the upstream repository. This would allow it to be less vulnerable to errors, as checks could be added to ensure that site generation is not broken by any commit.

### 7.1.3 Documentation

Currently there is not enough documentation to hand off maintenance confidently.

### 7.1.4 Testing

Unfortunately, there is currently minimal testing of the software. If the Python scripts are continued to be used, it would be good to introduce unit testing to ensure they are working correctly. In terms of JavaScript, it would be beneficial to convert it to TypeScript so that automatic verification can surface errors that would otherwise go unnoticed.

## 7.2 Future Work

The redesigned AFP has feature parity with the current AFP, however there are notable extensions that would elevate this project. In order of increasing complexity we have:

**Design Improvements** As the current AFP only has a desktop design, mobile was not accounted for in the redesign. It will be quite natural to convert the sidebar into a “hamburger” menu on mobile.

Similarly, the current AFP has one colour mode and so there is only one colour scheme in the redesign. It would be preferable to many people to add a dark mode to the site, including script browsing pages.

The page which lists the topics should be redesigned to be clearer by decreasing the density of the information and making the hierarchy clearer.

**Web Feeds** RSS and Atom feeds allow users to subscribe to updates for a page on the web. Regarding the AFP, these could allow an academic to subscribe to a feed of new entries under a topic, or an author could subscribe to a feed to be notified when someone uses their theorem.

**Accessibility** Accessibility is necessary for any professional website [18] and it was considered during implementation. For example, semantic HTML was chosen so that the website is more accessible by default, and background colours were chosen to have enough contrast. Unfortunately, an accessibility audit was not completed on the website so there are most likely outstanding accessibility issues.

**Functional Improvements** The code browsing feature currently lists all theories and lemmas in the side bar, however it becomes less useful as the number of lemmas increases. Instead, it would be helpful if the lemmas were initially collapsed under the theories and could be toggled when needed.

The search experience could be made more useful by providing search results which do not match the input exactly. For example, adding to the search will currently always decrease the number of returned results. This is beneficial for highly specific searches, but less so for finding related content. Surfacing the FindFacts results in a more obvious or useful way would also be appreciated, as the participants in the evaluation did not seem to be aware of its function.

**Improving Entry Maintenance** One of the benefits of using JSON as the entry format, is that it opens the door to editing metadata through a form in the browser. This should not be too difficult, however users would need to sign in and authenticate themselves before they receive editing permissions for current entries. This necessitates a web server to receive requests, and access control to define different categories of users. This extension would therefore be a substantial undertaking.

# Bibliography

- [1] Anon. The QED manifesto. In Alan Bundy, editor, *Automated Deduction - CADE-12, 12th International Conference on Automated Deduction, Nancy, France, June 26 - July 1, 1994, Proceedings*, volume 814 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 1994.
- [2] Tim Berners-Lee. Cool URIs don't change. <http://www.w3.org/Provider/Style/URI>, 1998. Accessed: 2021-04-05.
- [3] Mathias Biilmann. The new front-end stack. Javascript, APIs and markup. SmashingConf San Francisco 2016, 2016.
- [4] Jasmin Christian Blanchette, Max W. Haslbeck, Daniel Matichuk, and Tobias Nipkow. Mining the archive of formal proofs. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings*, volume 9150 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2015.
- [5] John Brooke. SUS: A quick and dirty usability scale. *Usability evaluation in industry*, page 189, 1996.
- [6] Michael D. Byrne, John R. Anderson, Scott Douglass, and Michael Matessa. Eye tracking the visual search of click-down menus. In Marian G. Williams and Mark W. Altom, editors, *Proceeding of the CHI '99 Conference on Human Factors in Computing Systems: The CHI is the Limit, Pittsburgh, PA, USA, May 15-20, 1999*, pages 402–409. ACM, 1999.
- [7] Karine Chemla, editor. *The history of mathematical proof in ancient traditions*. Cambridge University Press, Cambridge, 2012. OCLC: ocn779264820.
- [8] MDN contributors. HTML: A good basis for accessibility. Available at <https://developer.mozilla.org/en-US/docs/Learn/Accessibility/HTML>, March 2021. Accessed: 2021-04-10.
- [9] Thierry Coquand and Gérard P. Huet. The calculus of constructions. *Inf. Comput.*, 76(2/3):95–120, 1988.
- [10] Manuel Eberl, Gerwin Klein, Tobias Nipkow, Lawrence Paulson, and René Thiemann. Archive of Formal Proofs - statistics. Available at <https://www.isa-afp.org/statistics.html>. Accessed: 2021-01-12.

- [11] Jacques Fleuriot, Steven Obua, and Phil Scott. Social network processes in the isabelle and coq theorem proving communities, 2016.
- [12] Herman Geuvers. Proof assistants: History, ideas and future. *Sadhana*, 34(1):3–25, February 2009.
- [13] Jaydn Goodwin. Developing a New Web Application for the Archive of Formal Proofs. UG4 dissertation, School of Informatics, University of Edinburgh, 2020.
- [14] Google. Lighthouse | Tools for Web Developers. Available at <https://developers.google.com/web/tools/lighthouse>, 2021. Accessed: 2021-04-10.
- [15] John Gruber. Markdown Syntax Documentation. Available at <https://daringfireball.net/projects/markdown/syntax>, March 2004. Accessed: 2021-04-10.
- [16] Bruce Hanington and Bella Martin. *Universal Methods of Design: 100 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions*. Rockport Publishers, 2012.
- [17] Christophe G. Healey and James T. Enns. Attention and visual memory in visualization and computer graphics. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1170–1188, 2012.
- [18] Shawn Lawton Henry and Liam McGee. Accessibility - W3C. Available at <https://www.w3.org/standards/webdesign/accessibility>, January 2021. Accessed: 2021-04-10.
- [19] Fabian Huch and Alexander Krauss. FindFacts: A scalable theorem search, June 2020. Isabelle Workshop 2020.
- [20] Gérard P. Huet. Induction principles formalized in the calculus of constructions. In Hartmut Ehrig, Robert A. Kowalski, Giorgio Levi, and Ugo Montanari, editors, *TAPSOFT'87: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Pisa, Italy, March 23-27, 1987, Volume 1: Advanced Seminar on Foundations of Innovative Software Development I and Colloquium on Trees in Algebra and Programming (CAAP'87)*, volume 249 of *Lecture Notes in Computer Science*, pages 276–286. Springer, 1987.
- [21] M.Y. Ivory. *Automated Web Site Evaluation: Researchers' and Practitioners' Perspectives*. Human–Computer Interaction Series. Springer Netherlands, 2013.
- [22] Carlin MacKenzie, Jacques Fleuriot, and James Vaughan. An evaluation of the Archive of Formal Proofs. Available at <https://arxiv.org/abs/2104.01052>, 2021.
- [23] Chris Macrae. Hugo or Jekyll? Available at <https://forestry.io/blog/hugo-and-jekyll-compared>, April 2018. Accessed: 2021-04-08.
- [24] The mathlib Community. The lean mathematical library. *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, Jan 2020.

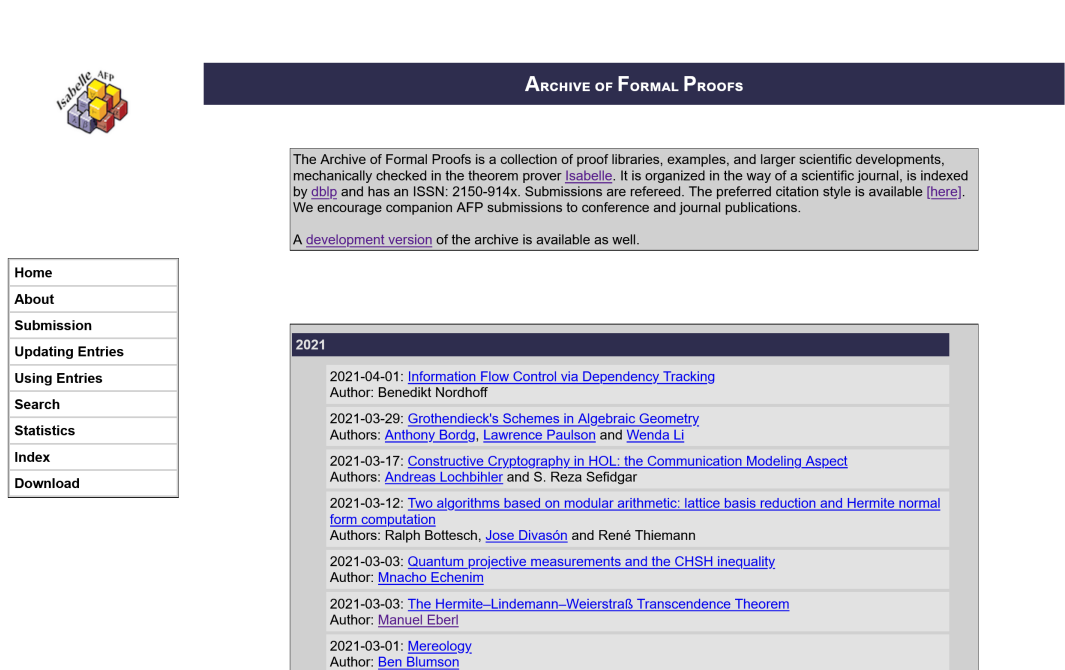


- [25] Davood Mazinianian and Nikolaos Tsantalis. An empirical study on the use of css preprocessors. In *2016 IEEE 23rd international conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 168–178. IEEE, 2016.
- [26] Robin Milner, Mads Tofte, and Robert Harper. *Definition of standard ML*. MIT Press, 1990.
- [27] Adam Naumowicz and Artur Korniłowicz. A brief overview of mizar. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Theorem Proving in Higher Order Logics*, pages 67–72, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [28] Ricardo Queirós. A survey on CSS preprocessors. In Ricardo Queirós, Mário Pinto, Alberto Simões, José Paulo Leal, and Maria João Varanda Pereira, editors, *6th Symposium on Languages, Applications and Technologies, SLATE 2017, June 26-27, 2017, Vila do Conde, Portugal*, volume 56 of *OASICS*, pages 8:1–8:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
- [29] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. *Automatic Keyword Extraction from Individual Documents*, chapter 1, pages 1–20. John Wiley & Sons, Ltd, 2010.
- [30] Krista Stevens. Open Sans, how do we love thee? — wordpress. Available at <https://wordpress.com/blog/2012/10/09/open-sans-how-do-we-love-thee-let-us-count-the-ways/>, October 2012. Accessed: 2021-04-08.
- [31] W3C. The W3C Markup Validation Service. Available at <http://validator.w3.org/>. Accessed: 2021-03-09.
- [32] W3C. CSS Grid Layout Module Level 1. Available at <https://www.w3.org/TR/2020/CRD-css-grid-1-20201218/>, December 2020. Accessed: 2021-04-09.
- [33] Makarius Wenzel. Isabelle/jEdit—a prover IDE within the PIDE framework. In *International Conference on Intelligent Computer Mathematics*, pages 468–471. Springer, 2012.
- [34] Makarius Wenzel. The Isabelle System Manual. Available at <https://isabelle.in.tum.de/doc/system.pdf>, February 2021. Accessed: 2020-11-18.
- [35] Makarius Wenzel. The Isabelle/Isar Reference Manual. Available at <https://isabelle.in.tum.de/dist/Isabelle2021/doc/isar-ref.pdf>, February 2021. Accessed: 2020-11-18.
- [36] Freek Wiedijk. The QED manifesto revisited. *Studies in Logic, Grammar and Rhetoric*, 10(23):121–133, 2007.

# Appendix A

## Screenshots of the Current AFP


At time of submission, the current AFP shown below can be viewed at <https://www.isa-afp.org/>



The screenshot shows the home page of the Archive of Formal Proofs (AFP). On the left is a navigation menu with the following items: Home, About, Submission, Updating Entries, Using Entries, Search, Statistics, Index, and Download. The main content area has a dark blue header with the AFP logo and the text "ARCHIVE OF FORMAL PROOFS". Below the header is a paragraph describing the archive: "The Archive of Formal Proofs is a collection of proof libraries, examples, and larger scientific developments, mechanically checked in the theorem prover Isabelle. It is organized in the way of a scientific journal, is indexed by dblp and has an ISSN: 2150-914x. Submissions are refereed. The preferred citation style is available [here]. We encourage companion AFP submissions to conference and journal publications. A development version of the archive is available as well." Below this is a list of entries for the year 2021:

2021
2021-04-01: <a href="#">Information Flow Control via Dependency Tracking</a> Author: <a href="#">Benedikt Nordhoff</a>
2021-03-29: <a href="#">Grothendieck's Schemes in Algebraic Geometry</a> Authors: <a href="#">Anthony Bordg</a> , <a href="#">Lawrence Paulson</a> and <a href="#">Wenda Li</a>
2021-03-17: <a href="#">Constructive Cryptography in HOL: the Communication Modeling Aspect</a> Authors: <a href="#">Andreas Lochbihler</a> and <a href="#">S. Reza Sefidgar</a>
2021-03-12: <a href="#">Two algorithms based on modular arithmetic: lattice basis reduction and Hermite normal form computation</a> Authors: <a href="#">Ralph Bottesch</a> , <a href="#">Jose Divasón</a> and <a href="#">René Thiemann</a>
2021-03-03: <a href="#">Quantum projective measurements and the CHSH inequality</a> Author: <a href="#">Mnacho Echenim</a>
2021-03-03: <a href="#">The Hermite–Lindemann–Weierstraß Transcendence Theorem</a> Author: <a href="#">Manuel Eberl</a>
2021-03-01: <a href="#">Mereology</a> Author: <a href="#">Ben Blumson</a>

Figure A.1: Current home page of the AFP



SYNTAX-INDEPENDENT LOGIC INFRASTRUCTURE

Home
About
Submission
Updating Entries
Using Entries
Search
Statistics
Index
Download

<b>Title:</b>	Syntax-Independent Logic Infrastructure
<b>Authors:</b>	<a href="#">Andrei Popescu</a> and <a href="#">Dmitry Traytel</a>
<b>Submission date:</b>	2020-09-16
<b>Abstract:</b>	We formalize a notion of logic whose terms and formulas are kept abstract. In particular, logical connectives, substitution, free variables, and provability are not defined, but characterized by their general properties as locale assumptions. Based on this abstract characterization, we develop further reusable reasoning infrastructure. For example, we define parallel substitution (along with proving its characterizing theorems) from single-point substitution. Similarly, we develop a natural deduction style proof system starting from the abstract Hilbert-style one. These one-time efforts benefit different concrete logics satisfying our locales' assumptions. We instantiate the syntax-independent logic infrastructure to Robinson arithmetic (also known as Q) in the AFP entry <a href="#">Robinson Arithmetic</a> and to hereditarily finite set theory in the AFP entries <a href="#">Goedel HFSet Semantic</a> and <a href="#">Goedel HFSet Semanticless</a> , which are part of our formalization of Gödel's Incompleteness Theorems described in our CADE-27 paper <a href="#">A Formally Verified Abstract Account of Gödel's Incompleteness Theorems</a> .
<b>BibTeX:</b>	@article{Syntax_Independent_Logic-AFP, [...]}
<b>License:</b>	<a href="#">BSD License</a>
<b>Used by:</b>	<a href="#">Goedel Incompleteness</a> , <a href="#">Robinson Arithmetic</a>

<a href="#">Proof outline</a>
<a href="#">Proof document</a>
<a href="#">Browse theories</a>
<a href="#">Download this entry</a>
Older releases:

Figure A.2: Example entry page from the current AFP

## Theory Prelim

```
chapter <Preliminaries>

(*<*)
theory Prelim
  imports Main "HOL-Eisbach.Eisbach"
begin
(*>*)

section <Trivia>

abbreviation (input) any where "any ≡ undefined"

lemma Un_Diff2: "B ∩ C = {} ⇒ A ∪ B - C = A - C ∪ B" by auto

lemma Diff_Diff_Un: "A - B - C = A - (B ∪ C)" by auto

fun first :: "nat ⇒ nat list" where
  "first 0 = []"
| "first (Suc n) = n # first n"

text <Facts about zipping lists:>


lemma fst_set_zip_map_fst:
  "length xs = length ys ⇒ fst `(set (zip (map fst xs) ys)) = fst `(set xs)"
  by (induct xs ys rule: list_induct2) auto

lemma snd_set_zip_map_snd:
  "length xs = length ys ⇒ snd `(set (zip xs (map snd ys))) = snd `(set ys)"
  by (induct xs ys rule: list_induct2) auto

lemma snd_set_zip:
  "length xs = length ys ⇒ snd `(set (zip xs ys)) = set ys"
  by (induct xs ys rule: list_induct2) auto


lemma set_zip_D: "(x, y) ∈ set (zip xs ys) ⇒ x ∈ set xs ∧ y ∈ set ys"
```

Figure A.3: Example theory page from the current AFP



SEARCH THE ARCHIVE

Use Google to search the archive. It will look in entry descriptions as well as in the Isabelle theories and PDF proof documents.




Web
  Archive of Formal Proofs

Google may take some time to index new pages. Very new Submissions might be missed.

Home
About
Submission
Updating Entries
Using Entries
Search
Statistics
Index
Download

Figure A.4: Current search page of the AFP



SUBMISSION GUIDELINES

Please send your submission [via this web page](#).


**The submission must follow the following Isabelle style rules.** For additional guidelines on Isabelle proofs, also see the [this guide](#) (feel free to follow all of these; only the below are mandatory). **Technical details about the submission process and the format of the submission are explained on the submission site.**

- No use of the commands `sorry` or `back`.
- Instantiations must not use Isabelle-generated names such as `xa` — use Isar, the `subgoal` command or `rename_tac` to avoid such names.
- No use of the command `smt_oracle`.
- If your theories contain calls to `nitpick`, `quickcheck`, or `munchaku` those calls must include the `expect` parameter. Alternatively the `expect` parameter must be set globally via, e.g. `nitpick_params`.
- `apply` scripts should be indented by `subgoal` as in the Isabelle distribution. If an `apply` command is applied to a state with `n+1` subgoals, it must be indented by `n` spaces relative to the first `apply` in the sequence.
- Only named lemmas should carry attributes such as `[simp]`.
- We prefer structured Isar proofs over `apply` style, but do not mandate them.
- If there are proof steps that take significant time, i.e. longer than roughly 1 min, please add a short comment to that step, so maintainers will know what to expect.
- The entry must contain a `ROOT` file with one session that has the name of the entry. We strongly encourage precisely one session per entry, but exceptions can be made. All sessions must be in group (AFP), and all theory files of the submission must be contained in at least one session. See also the example `ROOT` file in the [Example submission](#).
- The entry should cite all sources that the theories are based on, for example textbooks or research articles containing informal versions of the proofs.

Your submission must contain an abstract to be displayed on the web site — usually this will be the same as the abstract of your proof document in the `root.tex` file. You can use LaTeX formulae in this web site abstract, either inline formulae in the form `sa+bs` or `\(s+b\)` or display formulae in the form  $s_a + b_s$  or  $\left[ a + b \right]$ . Other occurrences of these characters must be escaped (e.g. `\$` or `\(`). Note that LaTeX in the title

Home
About
Submission
Updating Entries
Using Entries
Search
Statistics
Index
Download

Figure A.5: Current help page of the AFP



Home
About
Submission
Updating Entries
Using Entries
Search
Statistics
Index
Download

REFERRING TO AFP ENTRIES

Once you have downloaded the AFP, you can include its articles and theories in your own developments. If you would like to make your work available to others *without* having to include the AFP articles you depend on, here is how to do it.

If you are using Isabelle2021, and have downloaded your AFP directory to `/home/myself/afp`, for Linux/Mac you can run the following command to make the AFP session ROOTS available to Isabelle:

```
echo "/home/myself/afp/thys" >> ~/.isabelle/Isabelle2021/ROOTS
```

This adds the path `/home/myself/afp/thys/` to the ROOTS file, which Isabelle will scan by default. You can also manually edit and/or create that ROOTS file. There are many other ways to achieve the same outcome, this is just one option.

For Windows, the idea is the same just the path is slightly different. If the AFP is in `c:\afp`, you should be able to run the following in a Cygwin terminal.

```
echo "/cygdrive/c/afp/thys" >> ~/.isabelle/Isabelle2021/ROOTS
```

You can now refer to article `ABC` from the AFP in some theory of yours via

```
imports "ABC.Some_ABC_Theory"
```

This allows you to distribute your material separately from any AFP theories. Users of your distribution also need to install the AFP in the above manner.

Figure A.6: Current contribution page of the AFP

# Appendix B

## Evaluation of the Current Archive—Pre-study

To understand users and their requirements, a structured survey was created to poll the *Artificial Intelligence Modelling Lab* at the *University of Edinburgh*. This group was chosen as the members are familiar with Isabelle across a variety of use cases and workflows.

### B.1 Design

The survey had the following six sections in order:

- Familiarity questions
  - The first five questions of the survey filter users into different groups depending on their experience with the AFP.
- SUS questions
  - The 10 standard SUS questions were asked as an indicator of the usability of the current AFP.
- Navigation questions
  - Investigate how easy it is to find pages and which pages are accessed most.
- Design questions
  - Simple ratings of the look and feel and if it is intuitive.
- Browsing code within theories questions
  - Rating the experience and a short answer question about features.
- Ranking priorities question
  - Ranking which areas are most important to the user.

The survey was distributed via Microsoft Forms as it allows for complex surveys to be created and answered easily.

## B.2 Results

The survey had 10 total respondents, and 6 respondents who use the Archive. Of them, most were long term users of the AFP. However, only a third of them access the AFP frequently but almost all of them have downloaded an entry from the Archive.

The SUS score for the AFP is 46, which is below the average SUS score of 68 and suggests the AFP needs serious usability improvements.

Next respondents answered the first of two long answer questions:

What is your biggest pain point with the Archive? This could be with browsing entries, browsing code within entries, or any other feature of the AFP.

Five of six responded to this question with problems searching for entries or theorems. They described difficulty of not being sure of what to search for, or not being sure that they have found all the relevant entries. The remaining participant mainly had difficulty with the documentation for using entries and feel like some of the steps could be automated in some way. Interestingly, one user finds the AFP so painful to use that they download the entire Archive and manually search for things in jEdit as it provides more functionality.

Responses were split over whether it was easy to find specific entries in the AFP. On the other hand, everyone agreed that it was not easy to find entries on a topic or entries related to a topic.

The most accessed pages in order were: Search, Index (list of entries and topics), Citing Entries, Home, Using Entries and Download. The other five pages were never or rarely accessed.

All the other links were accessed at least sometimes, except for the Older Releases. Surprisingly, the only link which everyone accessed at least sometimes was the Proof Document page which is a PDF of the Isabelle code. This is interesting as it was assumed that people would prefer to access the syntax highlighted HTML version of the Isabelle content. It may be so frequently accessed as this is the only listing of all the code of an entry on one page.

In general, people did not mis-click when navigating, which suggests that the text is clear for links that people access.

The look and feel of the AFP received a 2.3 star rating out of 5. The intuitiveness of the layout received a higher score at 2.7, however this is still lacking.

Everyone who took the survey browses entry code and they rated the experience a 2.8. However they rated finding specific entries 1.3 out of 5, which is very poor. They then answered the second long answer question:

What feature would make browsing this code better for you?

Half of the respondents wanted functionality which would allow them to search by approximate/fuzzy statement, such as provided by the FindFacts tool [19]. Other features that were suggested were being able to click to go to the definition of an item, being able to see an outline of sections, searching across all theories and intra-page links between lemmas and others they are used in.

The final question of the survey was a ranking question of priorities. The results are very consistent, everyone ranked the same 3 in the top 3 priorities and the same for the bottom 3. In order:

1. Searching the archive.
2. Navigation, like finding related entries on a topic.
2. Browsing code within theories.
3. Submitting entries to the archive.
4. Look and feel.
5. Statistics about the archive.

Therefore, the most important priority is searching the archive.



## **Appendix C**

### **Evaluation of the Current Archive—Study Results**

## 6. What is your biggest pain point when submitting entries to the Archive?

Sometimes you get some errors from the system after submitting. And if I remember correctly, one time an entry didn't arrive because of a non-ASCII letter in an author name, but AFAIK this has been fixed now.

Whether the entry will be accepted or not.

In 2017, there was no "preview" feature for the entry description.

Converting apply-style proofs to Isar (not necessarily required by the AFP, but recommended)

Forgetting to update something about a theorem before submission.

Compared to a pull request on Github it is a bit more tedious and less transparent.

Building of submission failing due to LaTeX issues without helpful error messages.

Need to make sure the LaTeX part compile.

To bring a submission into format. Sometimes this needs 5–6 times to make a submission attempt and to finally complete it.

Having to run the new entry with the Isabelle development version if the new entry imports an entry which has been updated since the last release.

Checking the Isabelle style rules.

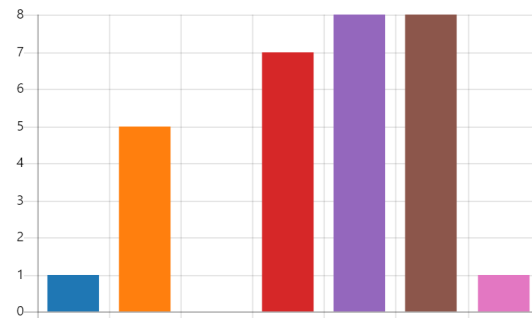
Getting the ROOT file right.

Table C.2: **Submission.** Six of the comments were related to formatting of the ROOT file and script files. The most actionable feedback from this section was that error messages are often unhelpful and that there is no preview for the abstract section. Three participants had no discernible pain point with submission and are not included in the table

1. How often do you access the Archive of Formal Proofs?

[More Details](#)

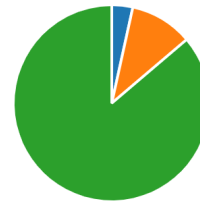
● Never	1
● When there is a new submission	5
● When there is a new release o...	0
● Weekly	7
● Monthly	8
● Few times per year	8
● Yearly	1



2. How long have you been using the Archive?

[More Details](#)

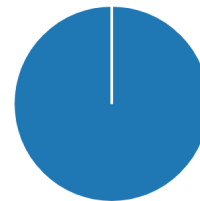
● three months or less	1
● one year or less	3
● more than a year	25



3. Have you ever downloaded an Archive entry?

[More Details](#)

● Yes	29
● No	0



4. Have you ever submitted an entry to the Archive?

[More Details](#)

● Yes	20
● No	9

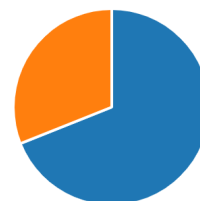


Figure C.1: **Demographics.** The demographics of the respondents is skewed towards very active and long-term users.

5. How much do you agree with the following

[More Details](#)

Strongly disagree Disagree Neutral Agree Strongly agree

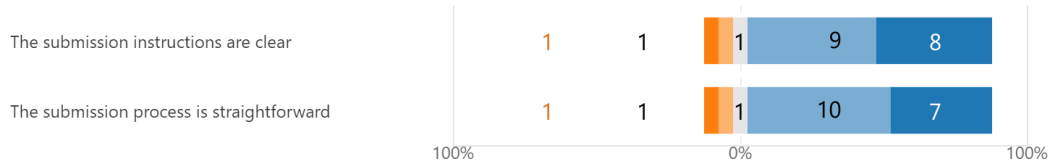


Figure C.2: **Submission.** The vast majority of people who have submitted find the process clear and straightforward.

7. For each of the following statements, please mark one box that best describes your reactions to the Archive of Formal Proofs as it is today.

[More Details](#)

Strongly disagree Disagree Neutral Agree Strongly agree

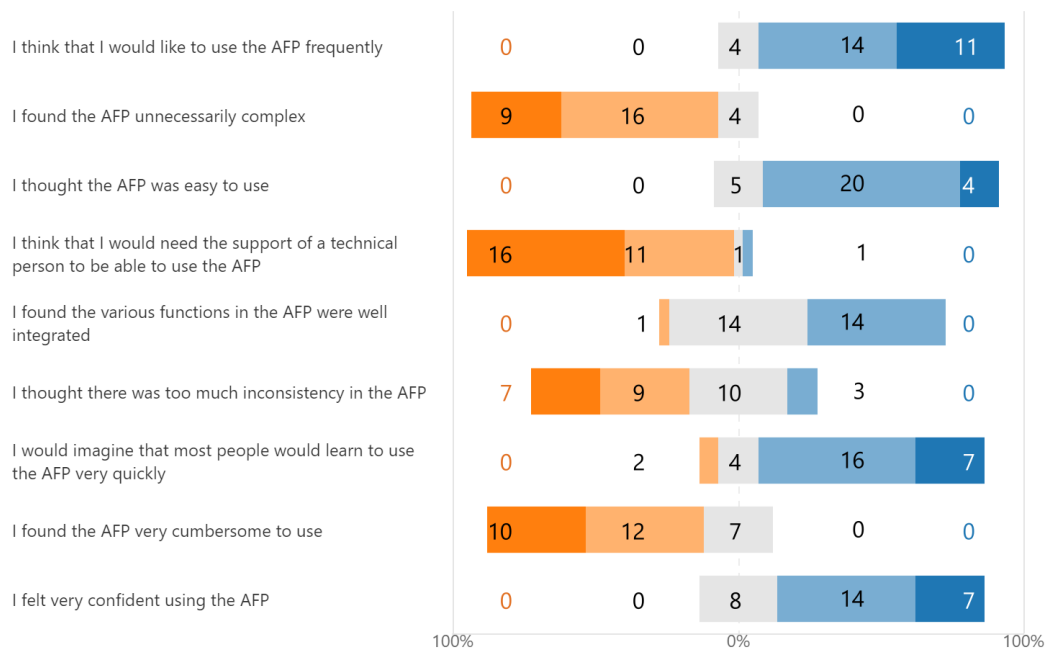


Figure C.3: **SUS Questions.** The SUS score for the AFP is 72, which is above the average SUS score of 68 and suggests that the participants are satisfied by the AFP.

8. What is your biggest pain point with the Archive? This could be with browsing entries, browsing scripts within entries, or any other feature of the AFP.

I think the biggest problem is that <https://www.isa-afp.org/using.html> is not explained well for Microsoft Windows.

Use downloaded entries (e.g.integrate in a new development). (This might be more an issue with Isabelle itself than AFP, I do not use Isabelle frequently.)

Finding the correct Theory to import in Isabelle for a given Entry.

I cannot online download and integrate the libs of AFP into Isabelle/HOL in the Isabelle/jedit UI.

A lot of redundant formalizations (like graphs), making it non-obvious which to use.

Problems with installing and using the new AFP version with every new Isabelle release.

Rather weak HTML presentation.

It's sometimes hard to find what you're looking for when you're just in search of "a development that does X".

Learning what is there. As it grows, I do not know if my contributions are reinventing the wheel or if any theory for an entry in a different topic can help with my developments.

The Proof Document contains all the proof, but the research value of the entry is usually in the published paper. A direct link would be useful.

The scope could be clearer. In particular: What do I do with work in progress? Are many small libraries or one big library preferred? What about new tools, i.e. new tactics implemented in ML without any new proofs? How do I add a library from the AFP as a dependency to my project? (The method described at <https://www.isa-afp.org/using.html> lacks basic functionalities like versioning or automatic downloading of dependencies and is a system wide setting instead of a per-project setting.)

Searching if a theory already does something I need.

Table C.4: **Biggest Pain Point.** The most common response was problems using AFP entries with Isabelle/jEdit [33]. In total, 6 people had this problem, from lacking instructions for Windows to finding the correct theory to import from an entry. The next largest area was search, with 3 respondents describing issues relating to finding whether there is an entry which does what they need. There were four specific asks: one would like a direct link to the corresponding paper about the entry if applicable; another finds the HTML presentation weak; yet another finds it difficult to choose between many similar entries; finally, one user is confused of the scope of the project and what entries are worthy of entering. Three respondents had no pain point with the AFP and their responses are not included in the table.

9. For each of the following, please mark one box that best describes your reaction to the statement

[More Details](#)

Disagree Neutral Agree

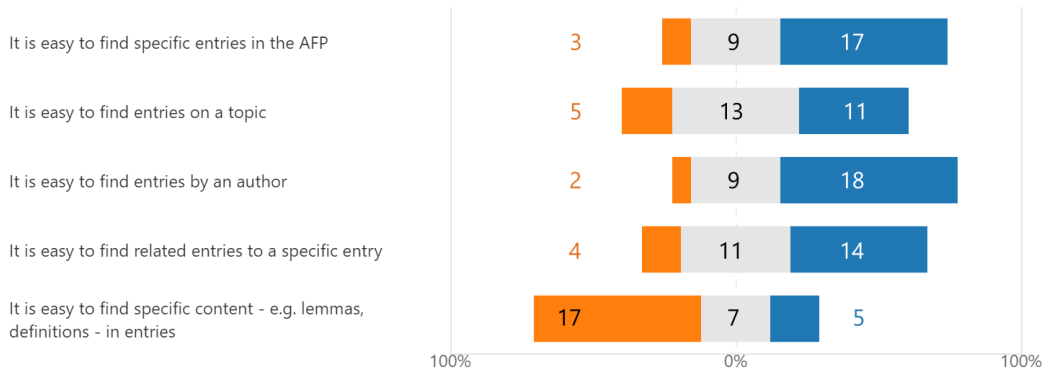


Figure C.4: **Navigating to Specific Content.** Most content is easy to navigate to, except for specific content in entries. Notably, there is no category in which everyone is neutral or agrees implying that navigation can be improved in all areas.

10. How often do you access the following pages:

[More Details](#)

■ Never
 ■ Rarely
 ■ Sometimes
 ■ Frequently

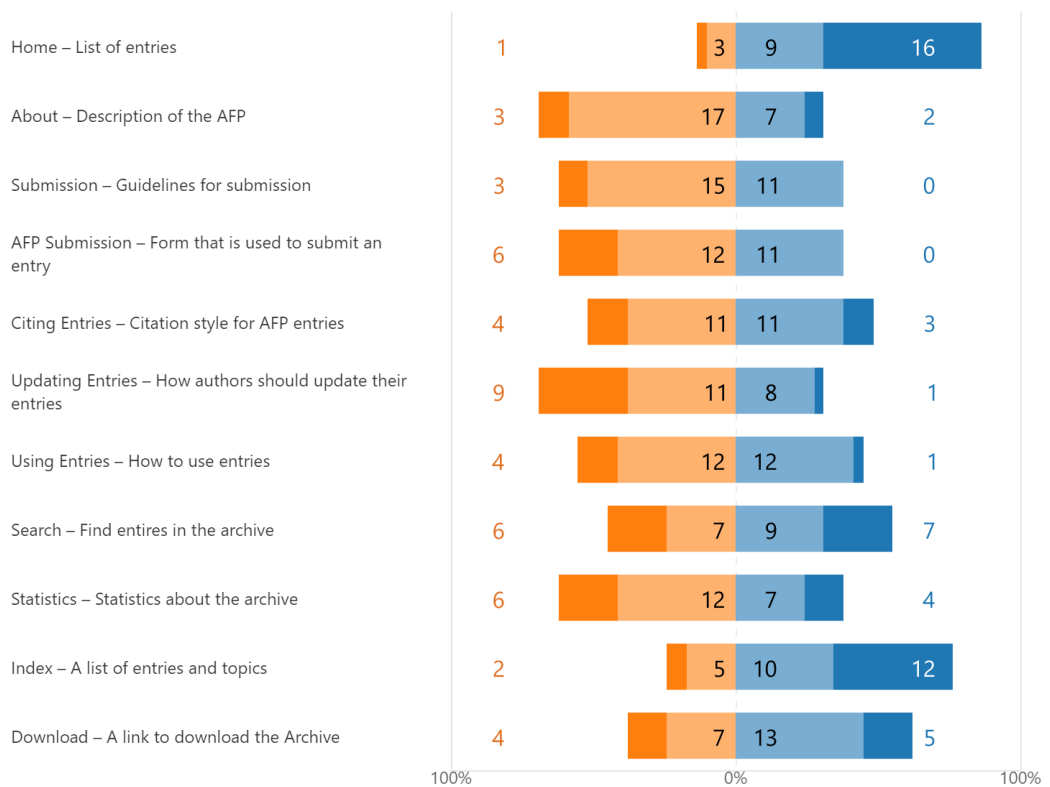


Figure C.5: **Navigating to Pages.** There are very different access requirements for pages of the AFP even though all but two of the eleven pages feature in the sidebar.

11. For an entry of the AFP, how often do you access the following:

[More Details](#)

■ Never
 ■ Rarely
 ■ Sometimes
 ■ Frequently

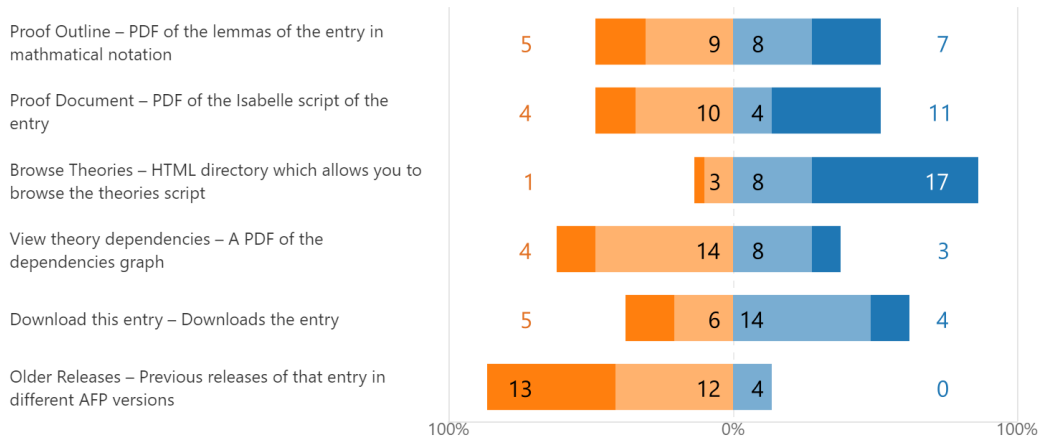


Figure C.6: **Navigating to Pages Related to the Entry.** Each entry of the AFP has several links to pages related to it. “Browse Theories” and “Download” are the most accessed while “Older Releases” is rarely accessed.

12. How often do you mis-click when navigating between pages?

[More Details](#)

[Insights](#)

- Frequently 0
- Sometimes 4
- Rarely 13
- Never 12

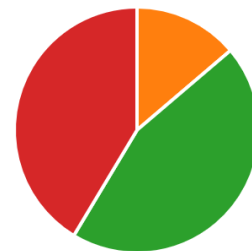


Figure C.7: **Clarity of Link Text.** Over half the participants mis-click rarely or some-times.



13. For each of the following, please mark the box that best describes your reaction to the statement

[More Details](#)

Strongly disagree Disagree Neutral Agree Strongly agree

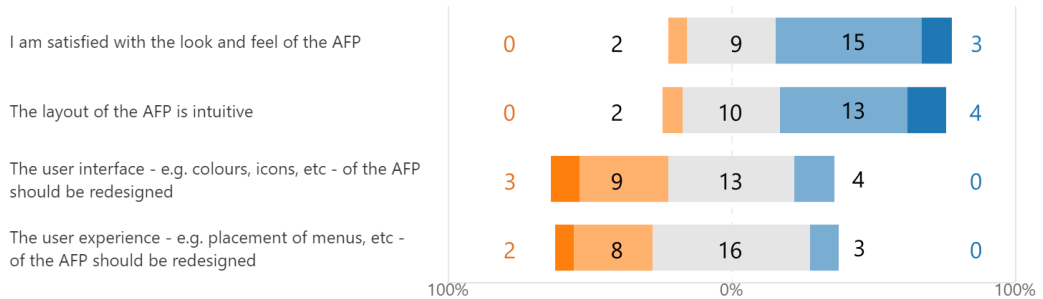


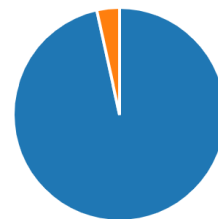
Figure C.8: **Design and User Experience.** Most users are satisfied with the UI and UX but are neutral towards a redesign of either.

14. Do you browse theories within entries?

[More Details](#)

[Insights](#)

Yes 28  
No 1



15. How satisfied are you with:

[More Details](#)

Very dissatisfied Somewhat dissatisfied Neither satisfied nor dissatisfied Somewhat satisfied Very satisfied

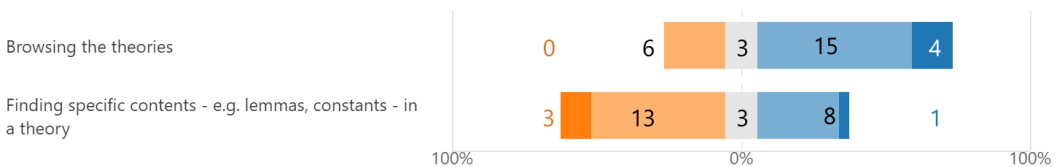


Figure C.9: **Browsing Theories.** Almost all participants browse theories and are mostly satisfied with the experience. However, they are largely unsatisfied with finding contents within theories.

## 16. What feature would improve browsing theory scripts?

The ctrl-click/cmd-click option of JEdit to find theorems and constants available in the online version.

If I could navigate to the definition of a type or a constant by clicking on it.

Summary/Outline Feature. Goto Definition/Usage Statistics about frequently used theorems.

Search for a lemma and a definition. Click & jump like in jEdit when navigating theory Maybe something like “sidekick” from Isabelle/jEdit. Maybe a better search.

Linking <https://search.isabelle.in.tum.de/> would improve the search experience.

More structure and links in the HTML output.

Links from entities to where they are defined or proved.

Index of lemmas.

A proper search function.

A Sidekick of the theory.

Semantic search.

Ontology and ontology based search.

I don't know

Maybe something like “sidekick” from Isabelle/jEdit. Maybe a better search.

Clickable terms with a link that leads to the definition!!! that would be awesome!; Crossreferences; overlays that show information about terms.

Add some features from jEdit: highlighting of inner syntax, go to definition hyperlinks, search theorems and search constants functionality, text search across all files. Also: option to find all uses of a constant or lemma.

**Table C.6: Browsing Theory Scripts.** 16 people responded to this question and half of them requested the ability to be able to click on links to definitions, as available in Isabelle/jEdit. Following this was 7 requests for better search capabilities and 5 requests for SideKick functionality (an outline of the sections, lemmas, etc). One respondent suggested usage statistics of frequently used theorems.

17. Please rank the following in order of importance to you

[More Details](#)

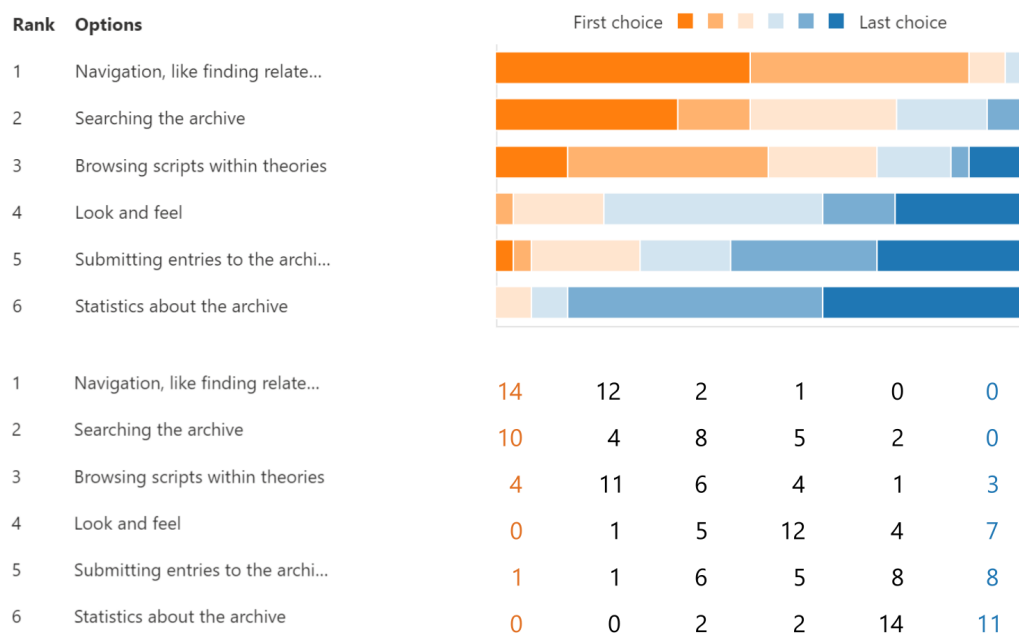


Figure C.10: **Ranking Priorities.** The ordering of priorities is consistent across the participants. Look and feel is a low priority which is congruous with the neutrality towards a redesign.

# Appendix D

## Paper Prototypes

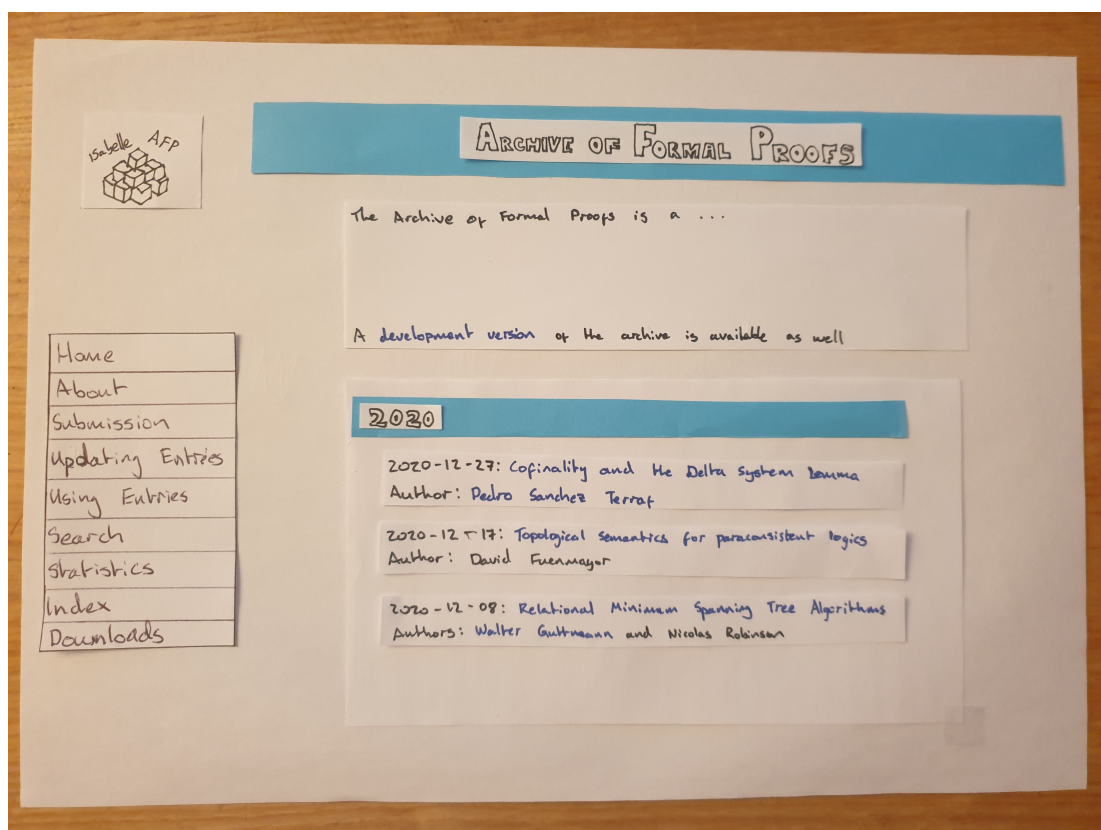


Figure D.1: First paper prototype of the AFP home page

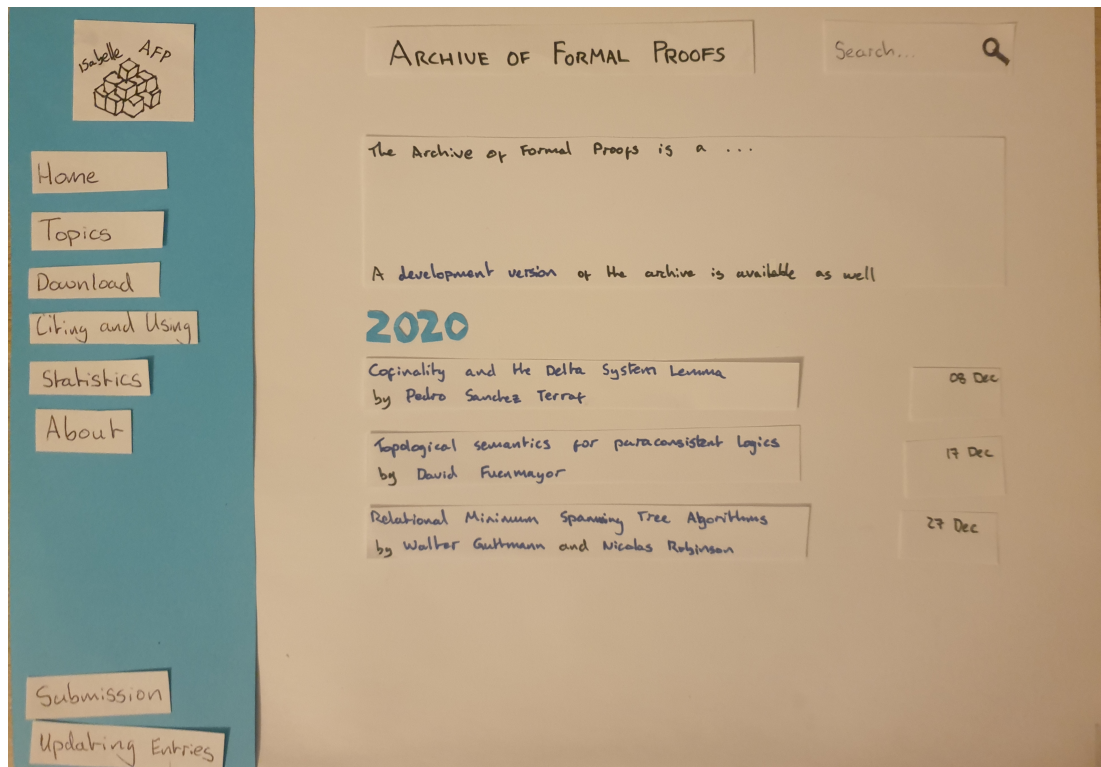


Figure D.2: Final paper prototype of the AFP home page

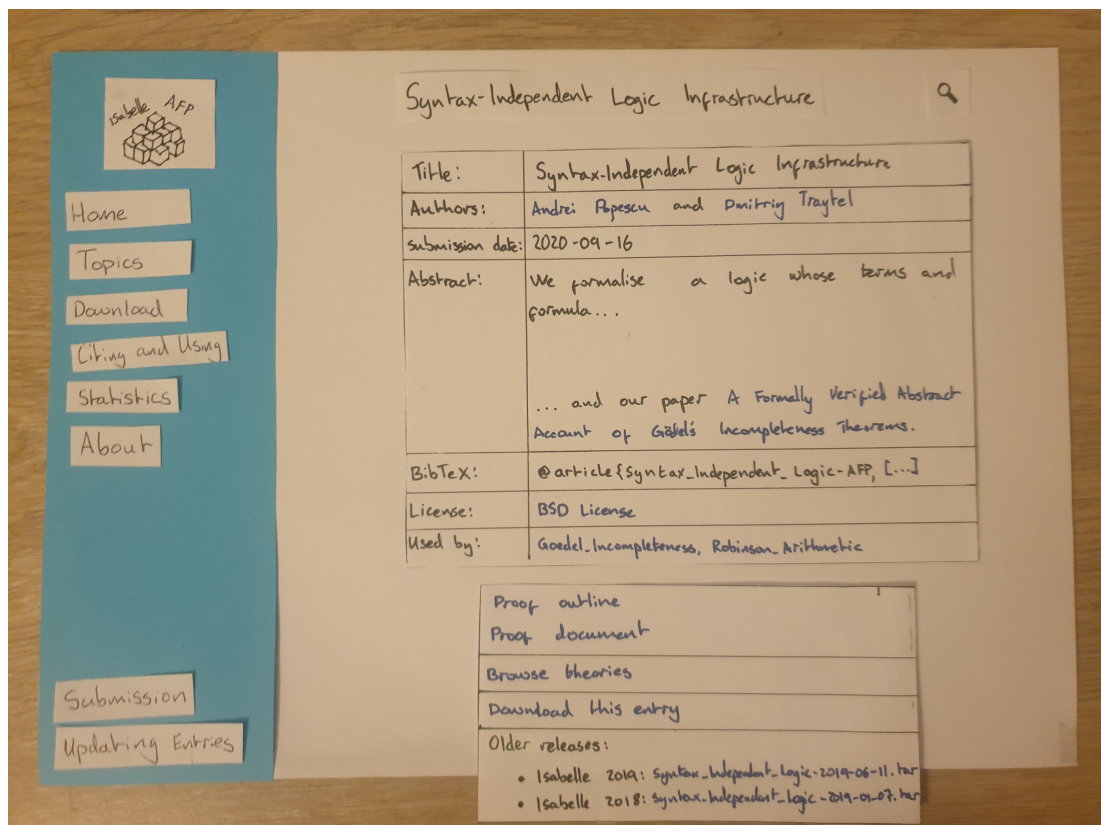


Figure D.3: First paper prototype of an AFP entry page

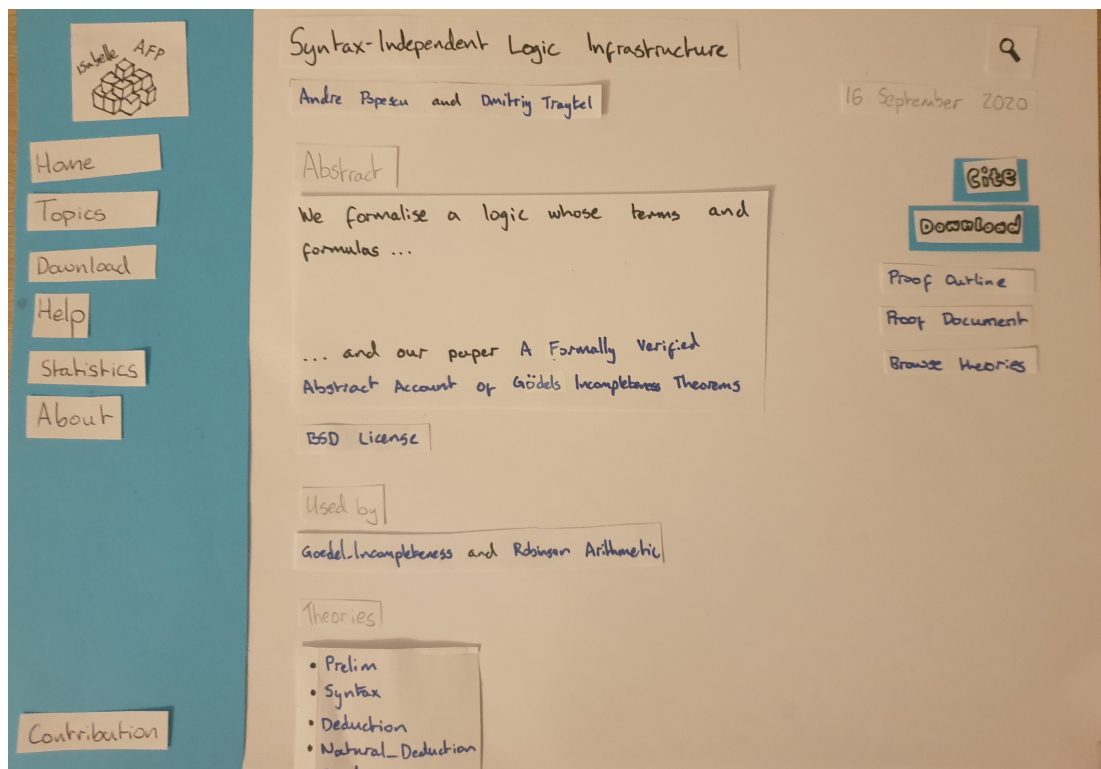


Figure D.4: Final paper prototype of an AFP entry page

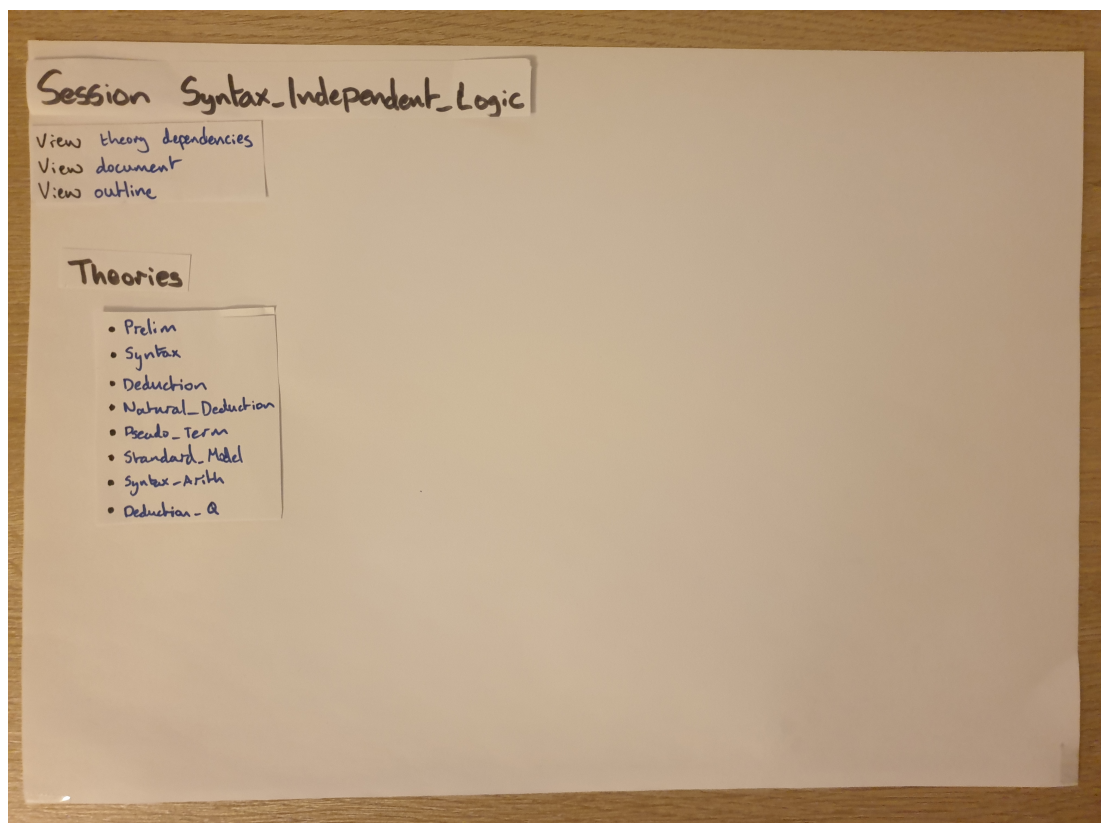


Figure D.5: First paper prototype of an AFP theory page

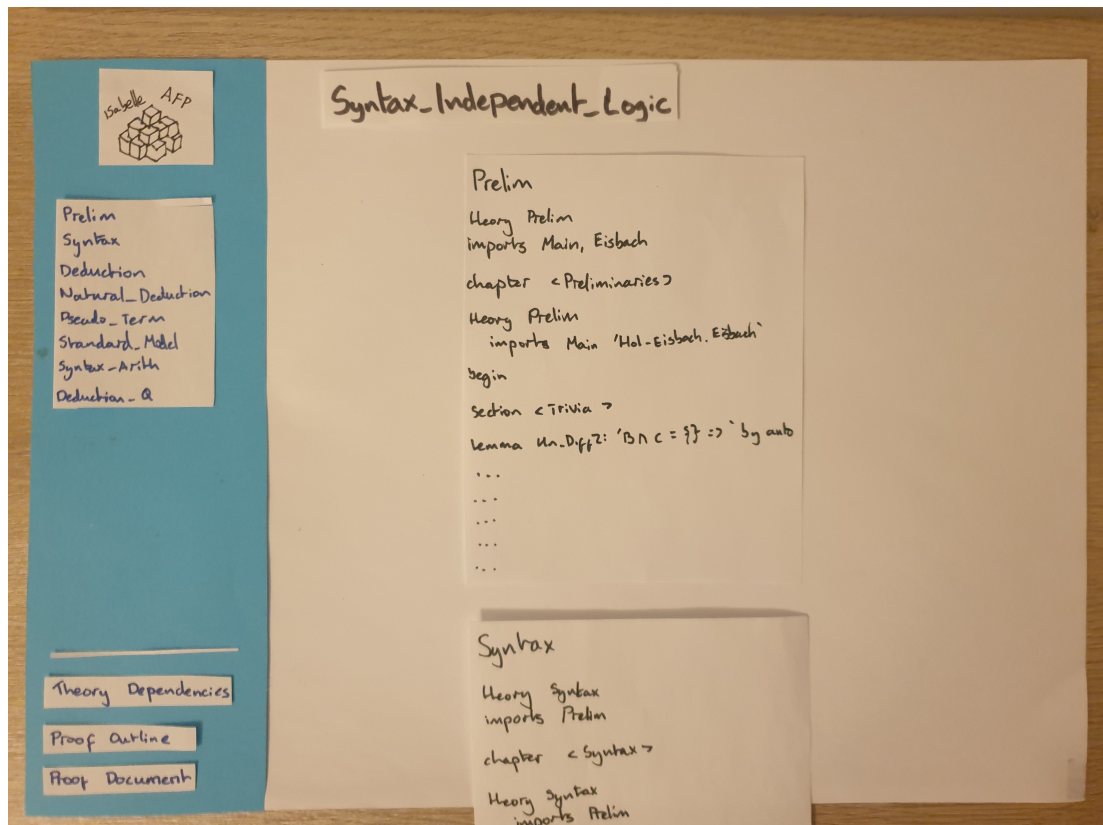
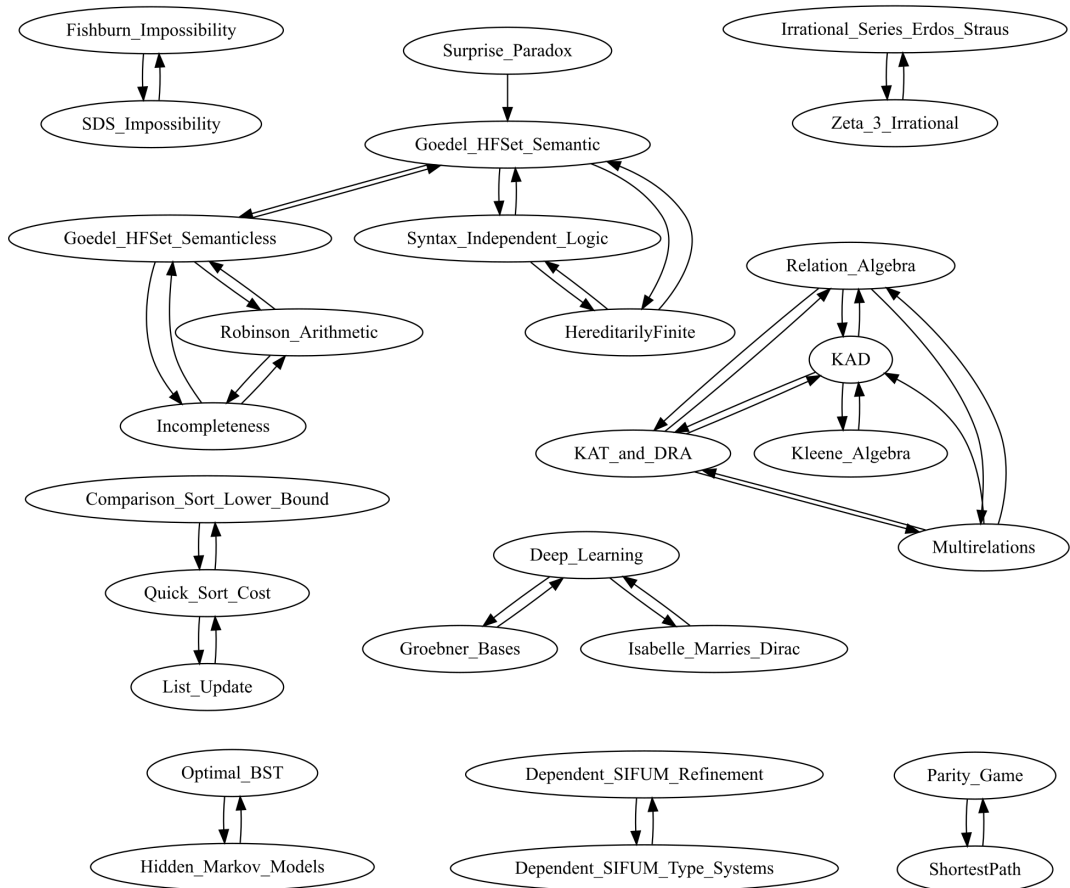


Figure D.6: Final paper prototype of an AFP theory page

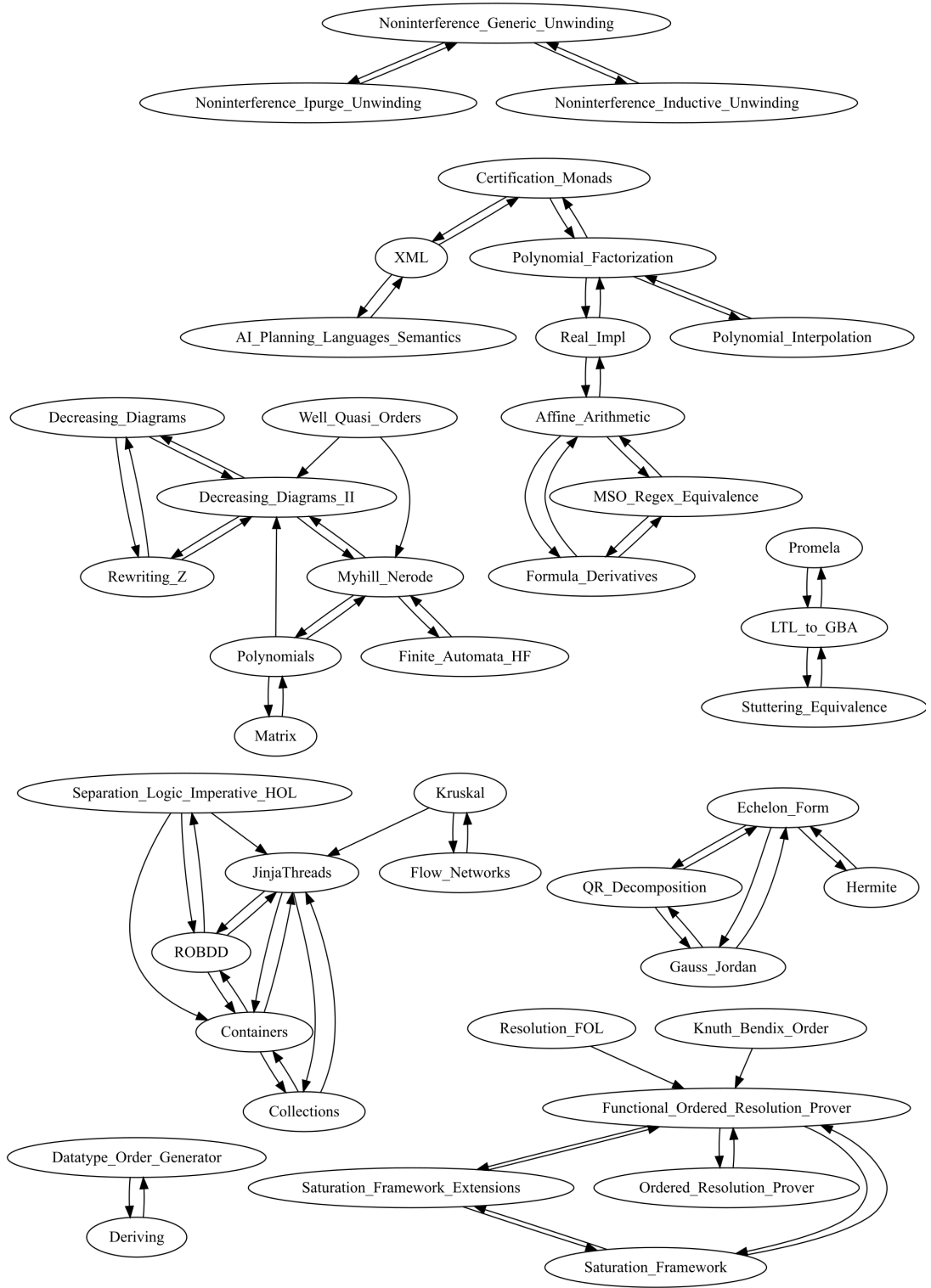
# Appendix E

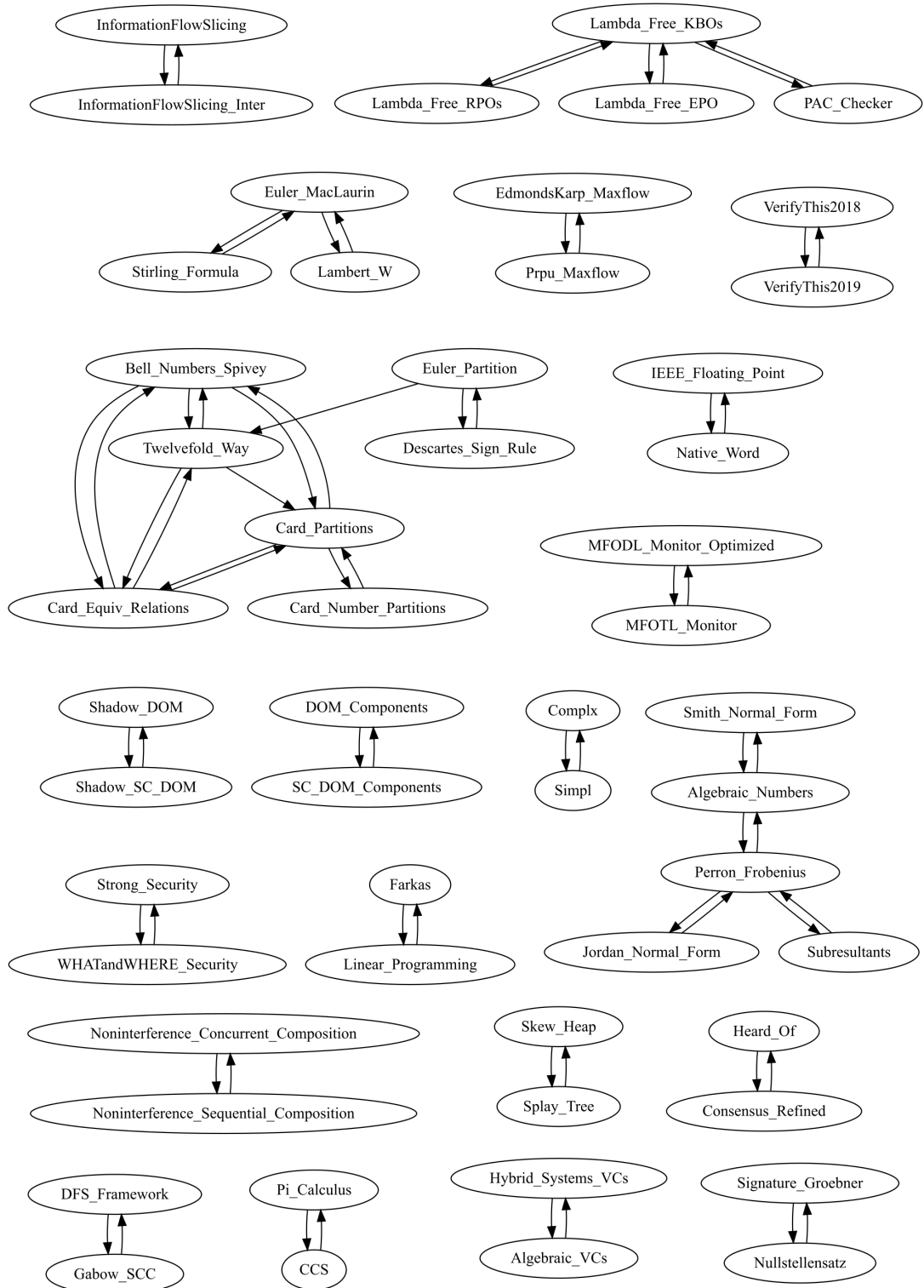
## Related Entry Graphs

Graphs which visualise the related entries of the redesigned AFP. A line from A to B implies A is related to B. Each node will only have up to three outgoing edges, but there is no limit on incoming edges. There is no ordering of the clusters.





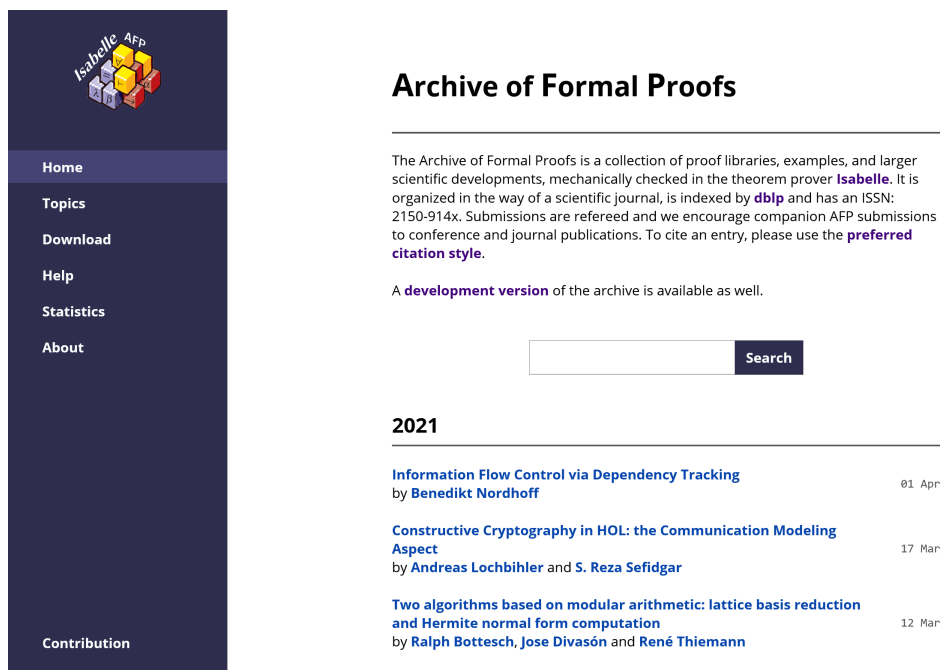




# Appendix F

## Screenshots of the Redesigned AFP

At time of submission, the redesigned AFP shown below can be viewed at <https://carlinmack.github.io/>



**Archive of Formal Proofs**


The Archive of Formal Proofs is a collection of proof libraries, examples, and larger scientific developments, mechanically checked in the theorem prover **Isabelle**. It is organized in the way of a scientific journal, is indexed by **dblp** and has an ISSN: 2150-914x. Submissions are refereed and we encourage companion AFP submissions to conference and journal publications. To cite an entry, please use the **preferred citation style**.

A **development version** of the archive is available as well.

### 2021

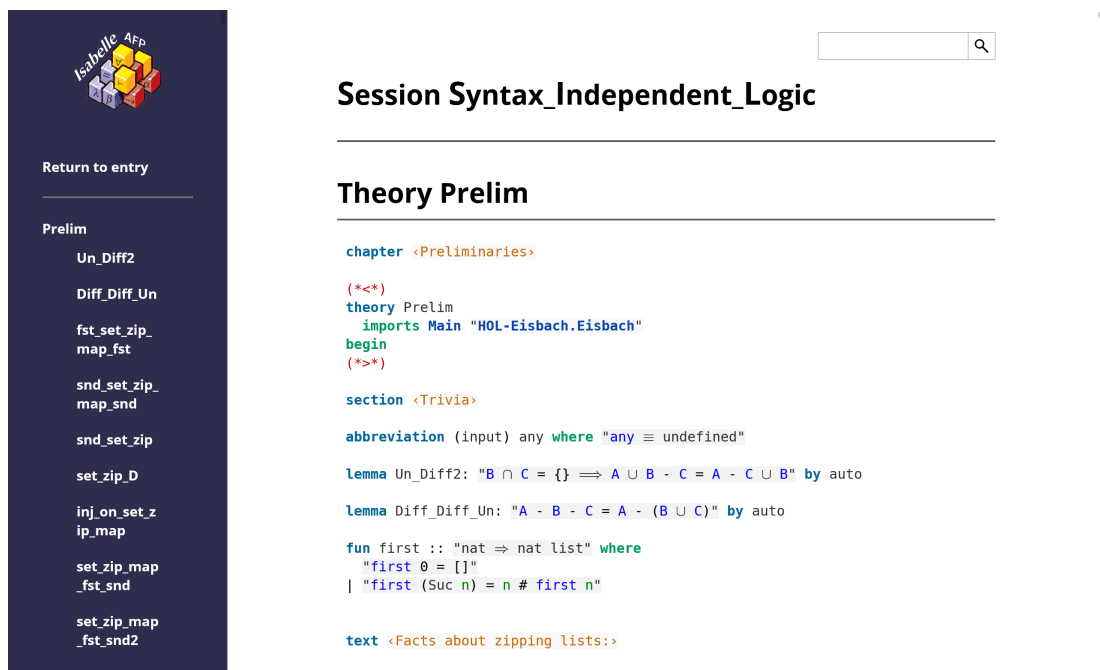
<b>Information Flow Control via Dependency Tracking</b> by <b>Benedikt Nordhoff</b>	01 Apr
<b>Constructive Cryptography in HOL: the Communication Modeling Aspect</b> by <b>Andreas Lochbihler</b> and <b>S. Reza Sefidgar</b>	17 Mar
<b>Two algorithms based on modular arithmetic: lattice basis reduction and Hermite normal form computation</b> by <b>Ralph Bottesch</b> , <b>Jose Divasón</b> and <b>René Thiemann</b>	12 Mar

Figure F.1: Home page of the redesigned AFP



The screenshot shows the Isabelle AFP website interface. On the left is a dark blue sidebar with navigation links: Home, Topics, Download, Help, Statistics, About, and Contribution. The main content area has a search bar at the top right. The title is "Syntax-Independent Logic Infrastructure" by Andrei Popescu and Dmitriy Traytel, dated 16 September 2020. Below the title is an "Abstract" section containing a paragraph of text. To the right of the abstract are buttons for "Cite" and "Download". Below the abstract are links for "PDFs", "Proof outline", "Proof document", and "Dependencies". At the bottom of the abstract section is a link for "BSD License". Below the abstract section is a "Theories" section.

Figure F.2: Example entry page from the redesigned AFP



The screenshot shows the Isabelle AFP website interface for a theory page. On the left is a dark blue sidebar with a "Return to entry" link and a "Prelim" section listing various theory names. The main content area has a search bar at the top right. The title is "Session Syntax\_Independent\_Logic". Below the title is a "Theory Prelim" section containing a code block with Isabelle theory definitions. The code includes a chapter header, a theory definition with imports, a section header, an abbreviation, two lemmas, a function definition, and a text block.

```
chapter <Preliminaries>

(*<*)
theory Prelim
  imports Main "HOL-Eisbach.Eisbach"
begin
(*>*)

section <Trivia>

abbreviation (input) any where "any ≡ undefined"

lemma Un_Diff2: "B ∩ C = {} ⇒ A ∪ B - C = A - C ∪ B" by auto

lemma Diff_Diff_Un: "A - B - C = A - (B ∪ C)" by auto

fun first :: "nat ⇒ nat list" where
  "first 0 = []"
| "first (Suc n) = n # first n"

text <Facts about zipping lists:>
```

Figure F.3: Example theory page from the redesigned AFP

Isabelle AFP

Home  
Topics  
Download  
Help  
Statistics  
About  
Contribution

## Search the Archive

graph

graph theory  
directed graph  
random graphs  
undirected graphs  
weighted graphs

Entries

**Graph Saturation**

Sebastiaan J. C. Joosten 2018

This is an Isabelle/HOL formalisation of **graph** saturation, closely following a **paper by the author on graph** saturation. Nine out of ten lemmas of the original paper are proven in this formalisation. The formalisation additionally includes two theorems that show the main premise of the paper: that consistency and entailment are decided through **graph** saturation. This formalisation does not give executable code, and it did not implement any of the optimisations suggested in the paper.

[Logic/Rewriting](#) and [Mathematics/Graph theory](#)

**Graph Theory**

Topics

- [Mathematics/Graph theory](#)
- [Computer science/Algorithms/Graph theory](#)

FindFacts Results

- [1081 Constants](#)
- [2384 Facts](#)
- [63 Types](#)

Figure F.4: Search page of the redesigned AFP

Isabelle AFP

Home  
Topics  
Download  
Help  
Statistics  
About  
Contribution

## Help

This section focuses on the Archive of Formal Proofs. For help with Isabelle, see the [Isabelle Wiki](#) and [Documentation](#)

### Referring to AFP Entries in Isabelle

Once you have downloaded the AFP, you can include its articles and theories in your own developments. If you would like to make your work available to others *without* having to include the AFP articles you depend on, here is how to do it.

#### Linux and Mac

If you are using Isabelle2020, and have downloaded your AFP directory to `/home/myself/afp`, you can run the following command to make the AFP session ROOTS available to Isabelle:

```
echo "/home/myself/afp/thys" >> ~/.isabelle/Isabelle2020/ROOTS
```

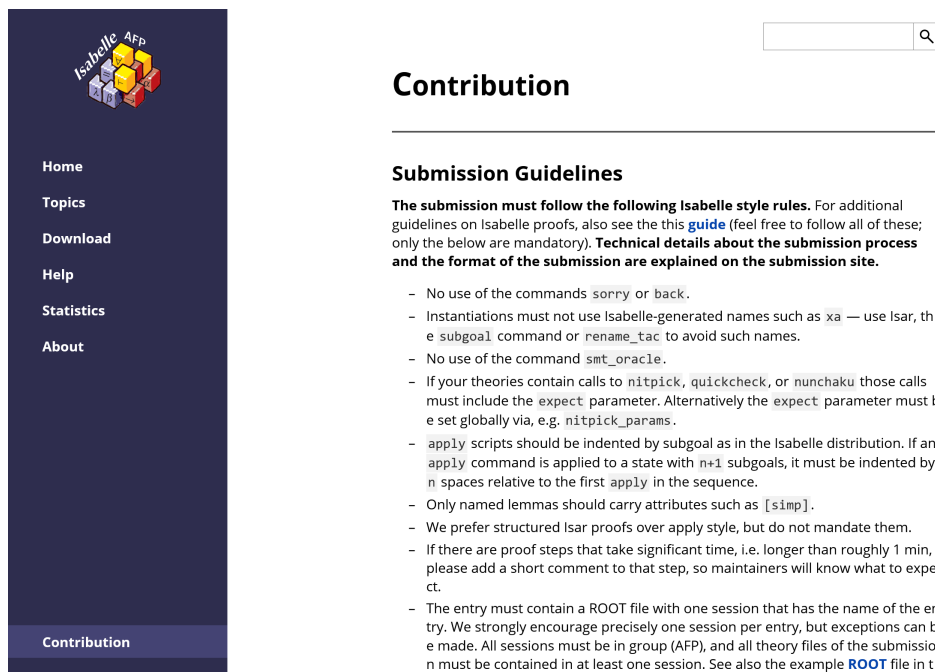
This adds the path `/home/myself/afp/thys/` to the ROOTS file, which Isabelle will scan by default. You can also manually edit and/or create that ROOTS file. There are many other ways to achieve the same outcome, this is just one option.

#### Windows

The idea is the same just the path is slightly different. If the AFP is in `C:\afp`, you should be able to run the following in a Cygwin terminal.

```
echo "/c:/drive/c/afp/thys" >> ~/.isabelle/Isabelle2021/ROOTS
```

Figure F.5: Help page of the redesigned AFP



Isabelle AFP

- Home
- Topics
- Download
- Help
- Statistics
- About
- Contribution**

Contribution

### Submission Guidelines

The submission must follow the following Isabelle style rules. For additional guidelines on Isabelle proofs, also see the [this guide](#) (feel free to follow all of these; only the below are mandatory). **Technical details about the submission process and the format of the submission are explained on the submission site.**

- No use of the commands `sorry` or `back`.
- Instantiations must not use Isabelle-generated names such as `xa` — use Isar, the `subgoal` command or `rename_tac` to avoid such names.
- No use of the command `smt_oracle`.
- If your theories contain calls to `nitpick`, `quickcheck`, or `nunchaku` those calls must include the `expect` parameter. Alternatively the `expect` parameter must be set globally via, e.g. `nitpick_params`.
- `apply` scripts should be indented by subgoal as in the Isabelle distribution. If an `apply` command is applied to a state with `n+1` subgoals, it must be indented by `n` spaces relative to the first `apply` in the sequence.
- Only named lemmas should carry attributes such as `[simp]`.
- We prefer structured Isar proofs over `apply` style, but do not mandate them.
- If there are proof steps that take significant time, i.e. longer than roughly 1 min, please add a short comment to that step, so maintainers will know what to expect.
- The entry must contain a `ROOT` file with one session that has the name of the entry. We strongly encourage precisely one session per entry, but exceptions can be made. All sessions must be in group (AFP), and all theory files of the submission must be contained in at least one session. See also the example `ROOT` file in t

Figure F.6: Contribution page of the redesigned AFP

# Appendix G

## Poster

The poster is titled "Re-imagining the Archive of Formal Proofs" and features the Isabelle AFP logo on the top left and the University of Edinburgh logo on the top right. The main content is organized into several sections:

- Summary:** Describes the Archive of Formal Proofs as the central repository for Isabelle formal proofs, updated in 20 years. The project re-implements it as a Hugo static site with responsive search and code browsing.
- Background:** Explains that Isabelle is a language for writing and validating formal mathematical proofs. The AFP first appeared in 2004 and hasn't been updated since. The site is generated using a handwritten Python static site generator that interfaces with Isabelle and Scala. Search is provided by directing users to a Google search with "site:isa-afp.org" appended. Script browsing is very plain, and entries are categorized by topics.
- Evaluation:** A survey was conducted to understand user needs. The first version was distributed to the Artificial Intelligence Modelling Lab. The second version was distributed to the Isabelle Mailing list and received 29 responses. Results will be published on ArXiv.
- Design:** The original design was a paper prototype. The new design is an interactive prototype created in Figma, showing how pages would link together.
- Implementation:** Hugo was chosen for site generation. Python scripts were used to transform content for compatibility with Hugo. The final site builds 2,500 pages in ~4 seconds. Search is implemented using FlexSearch.js. Script browsing uses a Python script to download theory pages and concatenate them. Related entries are generated using a similarity score between documents.

At the bottom, the authors are listed: By Carlin MacKenzie, Supervised by Jacques Fleuriot, Co-supervised by James Vaughan.

Figure G.1: Honours project day poster

# Appendix H

## Script for the Second Evaluation

Hello, I'm Carlin and today we will be evaluating a redesign of the Archive of Formal Proofs. Your participation today is purely voluntary, you may stop at any time.

Before we start, I just want to confirm that you've read the participation sheet and signed the consent form. If not, you can do that now. [After they have confirmed/signed] Is it okay for me to start recording the call now?

In this observation, I am interested in what you think about, as you perform the tasks you're asked to do. To do this, I am going to ask you to talk aloud as you work on the task. What I mean by "talk aloud" is that I want you to tell me everything you are thinking from the first time you see the statement of the task till you finish the task. I would like you to talk aloud constantly from the time I give you the task till you have completed it. I do not want you to try and plan out what you say or try to explain to me what you are saying. Just act as if you were alone, speaking to yourself. It is most important that you keep talking and I will prompt you if you are silent for a long period of time. Do you understand what I want you to do?

Good. We'll start with a simple practice problem first. I will demonstrate by thinking aloud while I solve a simple problem: "How many pillows are there in my parents' house?" [Demonstrate thinking aloud.] Please verbalise like this as you are doing the tasks. I will not be able to answer any questions, however, please ask them anyway and I will answer them after the session. Is this clear?

First, I would like you to open a browser and go to the link which I will send in the chat. [When they confirm have done so] Thank you, could you now share your screen?

<https://carlinmack.github.io>

I have prepared six tasks for you to do which I'll send over Teams. For each one please read it aloud, complete it to the best of your ability and to say "done" when you feel that you have completed the task. Lastly take your time, remember that I'm testing the interface, not you!

1. Visit the "Ordinal Partitions" entry and copy its BibTeX citation.
2. Download the "AVL Trees" entry



3. Search the AFP for “lemma” then “graph theory”.
4. Find how many submissions “Bohua Zhan” has authored.
5. Find the link to the submission form and return to the home page.
6. View the “Type” and “Instance” theories of the “Mini ML” entry and return to the home page.

Now that you have completed the tasks, I will send you a link to a survey which I would like you to answer. You can stop sharing your screen now, and please feel free to take your time and click around the website if you need a reminder. Let me know when you have completed it.

1. For each of the following, please mark the box that best describes your reaction to the statement

	Strongly Disagree	Disagree	Neutral	Agree	Strongly agree
It is easy to search for what I need.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
It is intuitive to navigate to the pages I need.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
If I was in a rush, this interface would not slow me down.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
It is easy for me to adapt to the new interface.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I am satisfied with the look and feel of this redesign.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I would imagine that most people would learn to use this interface very quickly.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I felt confident using this interface.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Lastly, I'd like you to visit the old website before I ask some final open-ended questions. I'll send a link in the chat to [www.isa-afp.org](http://www.isa-afp.org)

1. Is this an improvement over the current AFP? How so/how not?
2. Does this redesign meet your needs? Is there anything lacking or missing?

This is the end of experiment, thank you so much for your time, it was really appreciated.