

A Theoretical Analysis of Hierarchical Proofs

Paul Cairns and Jeremy Gow

UCL Interaction Centre, University College London
26 Bedford Way, London WC1H 0AP, UK
{p.cairns, j.gow}@ucl.ac.uk
www.ucl.ac.uk/imp

Abstract. Hierarchical proof presentations are ubiquitous within logic and computer science, but have made little impact on mathematics in general. The reasons for this are not currently known, and need to be understood if mathematical knowledge management systems are to gain acceptance in the mathematical community. We report on some initial experiments with three users of a set of web-based hierarchical proofs, which suggest that usability problems could be a factor. We then highlight the problems with such experiments, caused by limited access to working mathematicians. In order to address these problems we present theoretical analysis of hierarchical proofs using *Cognitive Dimensions* [10]. The analysis allows us to formulate some concrete hypotheses about the usability of hierarchical proof presentations.

1 Introduction

Rigorous proof is a central aspect of modern mathematics. In providing a coherent organisation of mathematical knowledge, any human-oriented system must consider not only how to organise, reference and retrieve proofs but also how to present the retrieved proofs to the working mathematician.

In many automated or interactive theorem proving systems, proofs are represented using hierarchical structures. This reveals the ‘true structure’ of a proof that is suppressed in linear presentations of rigorous deductive proofs, such as those in textbooks [12]. In addition, there are well established data structures for internally storing tree structures. These considerations suggest that a hierarchical rather than linear proof presentation would be more appropriate from a system perspective and that, by revealing the logical structure, could actually be of more benefit to the reader as well.

However, there is no clear hypothesis with regards to the usability of hierarchical proofs. This paper develops hypotheses using two tools. The first is a small pilot study of the Polya-Lampert framework [3] that used hierarchical proofs as the main way to present proofs to mathematical readers. Secondly, hierarchical proofs are analysed in comparison with more traditional proofs using Cognitive Dimensions [8, 10]. The contribution of the paper is therefore testable ideas on the usefulness and usability of hierarchical presentations of mathematical proofs.

2 Hierarchical Proofs

It has long been recognised within the mathematical community that rigorous logical proofs can be represented hierarchically. That is, since statements of a proof depend logically on axioms or previous statements in the proof, the structure of the proof can be understood as a directed acyclic graph (or tree). However, in practice, the full logical structure is rarely presented, or even fully considered, and most published proofs take a sequential form like most printed material, e.g. [20, 9]. The few explicit logical links are made through the numbering of lemmas and definitions and the labelling of formulae rather than by any graphical means — most are left implicit, to be reconstructed by the reader.

Research into proofs with an explicit hierarchical structure falls into two broad categories: attempts to introduce hierarchical proofs into mathematics, and work on logic, including theorem provers and related systems.

Lampert proposed a standard for presenting hierarchical proof structure that employs indentation and labelling of proof statements, without resorting to graphical methods [15]. A proof is either a list of labelled statements, each with its own proof, or a list of references to previous statements. Thus, a proof may be broken down progressively into finer and finer detail until the bottom-most steps are trivial. Further annotations are used to indicate definitions and case analyses within the proof. An example proof is given in Figure 1. Lampert recommends the hierarchical format, as it ‘makes it much harder to prove things that are not true’. He also notes its suitability for hypertext systems.

PROOF SKETCH: Given a finite set of triadic primes, we construct a number that has a triadic prime factor not in the set.

ASSUME: There are only finitely many triadic primes $\{p_1, \dots, p_n\}$

PROVE: A contradiction

⟨1⟩1. 3 is a triadic prime, say $p_1 = 3$

PROOF: By definitions of triadic and prime.

LET: $M \doteq 4p_2 \dots p_n + 3$

⟨1⟩2. No p_1, p_2, \dots, p_n divides M

⟨2⟩1. p_1 does not divide $4p_2 \dots p_n$

⟨2⟩2. p_2, \dots, p_n have remainder 3 from M

⟨2⟩3. Q.E.D.

PROOF: By ⟨2⟩1 and ⟨2⟩2.

⟨1⟩3. M has a triadic prime factor

ASSUME: M 's prime factors are monadic

PROVE: A contradiction

⟨2⟩1. A product of monadic numbers is monadic

⟨2⟩2. M is monadic

⟨2⟩3. Q.E.D.

PROOF: By ⟨2⟩2 and definition of M .

⟨1⟩4. Q.E.D.

Fig. 1. A partial Lampert-style proof of the infinity of the triadic primes are infinite

Lampert's approach has been implemented in HTML for calculational style proofs (i.e. linear derivations) [11]. The hierarchical nature of the proof gives a natural way to hide and reveal details in a proof: substeps in the proof can be hidden or revealed on a mouse click.

An alternative way of hierarchically structuring proofs was proposed by Leron [16]. Whereas Lampert emphasises the deductive structure of a proof, Leron advocates using a hierarchy to highlight the conceptual structure. The top level of the proof explains the key ideas and proof outline, with lower levels making the ideas more concrete and filling in the details, until the author has nothing more to add. Furthermore, Leron's format is less structured, with each node¹ of the hierarchy corresponding to a collection of paragraphs, rather than an individual assertion. The proof presentation also incorporates heuristic and informal knowledge, with 'elevator' sections discussing the transition between levels.

The difference in emphasis can clearly be seen by comparing Figures 2 and 1. Lampert's proof hierarchy draws attention to the argument structure, and omits heuristic information. Leron's proof hierarchy highlights the key ideas and refines them. The argument is described in much the same way as traditional proofs.

Level 1 Suppose the theorem is false and let p_1, p_2, \dots, p_n be all the triadic primes. We construct in (Level 2) a number M having the following two properties:

- (a) M as well as its factors are different from p_1, p_2, \dots, p_n ;
- (b) M has a triadic prime factor.

These two properties clearly produce a contradiction, as we get a triadic prime which is not one of p_1, p_2, \dots, p_n . Thus the theorem is proved.

In The Elevator How shall we approach the definition of M ? In light of Euclid's classical proof, it is natural to try $M = p_1 p_2 \dots p_n + 1$. This indeed...

Level 2 Let $M = 4p_2 \dots p_n + 3$ (we assume $p_1 = 3$). We show that M satisfies the two requirements from Level 1. Requirement (a) means...

Level 3 Lemma: A product of monadic numbers is again a monadic number.

Fig. 2. Extracts from a Leron-style proof of the infinity of triadic primes .

Amongst the automated and interactive theorem proving community, there are numerous examples of user interfaces that use a hierarchical representation. XIsabelle was developed as an interface for the Isabelle interactive proving system [6]. The steps of the proof are graphically linked when there is a tactic used to move from one step to the other. This interface also supports the generation of proofs in that steps can be selected from the tree and then further actions will apply different tactics to the step.

The Ω MEGA and λ Clam proof planning systems both have graphical user interfaces that can represent proofs as trees, respectively, L Ω UI [21] and XBar-

¹ Confusingly, he calls nodes 'levels'. Hence they may be several 'levels' of depth 2, called 2.1, 2.2,...

nacle [14]. Proofs in proof planning are not always proofs in a particular logic. However, the same principles apply in that the links between steps represent that two steps are related by a method application. And most proof plans are considerably more logically rigorous than most mathematicians so deserve the term proof every bit as much.

The JAPE system is intended as a tool for teaching logic to computing science students [1]. As such it is able to work in a variety of logics and with a variety of representations. One of the representations is very much like the paper notation of tableau proofs. Another is more like a written page and the proof a step has a subproof made up of substeps and so on. In both cases though, the interface is representing the logical connection between steps and allowing users to select the steps for further development of the proof.

Thus, there are many ways in which hierarchical proofs have been used both in mathematics and in computer science. Yet despite the naturalness and obviousness of the representation, it is notable that none have wide use in the general mathematical community. This suggests that there is some factor working against the use of hierarchical presentations.

3 Initial Experiments

We developed the Polya-Lamport framework to investigate using interaction to improve presentations of mathematical proofs. The Polya part of the framework built on the heuristic problem solving approach proposed by Polya [19] to provide users with relationships to the wider context of particular proof or theorem. The Lamport part was based on the method of structuring proofs to provide a way of providing users control over the amount of the detail that they see in a proof. We will only discuss the actual proof presentation here. The full framework is described elsewhere [3].

3.1 The Web-Based Hierarchical Proofs

The presentation of proofs in our framework is very similar to that proposed by Lamport. A proof is made up of a sequence of steps and each step may have its own proof made up of further steps. Though Lamport only demonstrated the structure of proofs through examples, it was relatively easy to formalise the structure into a grammar for an XML DTD [2]. Thus, all proofs were actually written by hand in XML and converted using XSLT [5] to a combination of HTML, GIFs and JavaScript that could be viewed on the two most common browser platforms, Netscape and Internet Explorer.

Interaction with a proof came in two main ways. First, steps that had their own proofs had squares next to them for toggling between states. Clicking the squares revealed or hid the proof as appropriate. When the proofs were hidden, the hidden steps were “summarised”. That is, any statements that were referred to in the hidden steps but were not part of the hidden steps were collected

together and made a summary such as “By background 2, assumption 2.1”. This was called the summary of the proof step.

Secondly, there were a number of links in the proof, just as in Lamport, that were to assumptions, other proof steps or background theory. Hovering over these links revealed the statement that was being linked to in a window at the bottom of the page². Clicking on the links brought the target statement to the top of the page. Thus, the proof presentation was a combination of common web and user interface interactions built into Lamport’s proof structure.

The new presentation was intended to be compared with more traditional proof presentations on paper. Thus, the content of the hierarchical proofs was made to resemble the normal language of the mathematical vernacular [13]. Logically necessary information was often omitted because it was irrelevant or could be easily reconstructed from the context. For example, we say X is a topological space even though this is formally defined as a pair (X, \mathcal{T}) .

3.2 A Limited User Study

A pilot user trial was used to elicit feedback on the Polya-Lamport framework as a method for presenting proofs. The trial used as the basic content an undergraduate course in topology written by Peter Collins. This course is taken by all first year mathematicians at Oxford University. The course notes were adapted into the Polya-Lamport framework using the original examples, diagrams and language as far as possible. Details of the conversion of the materials is given elsewhere [4]. These formed a set of online materials that were used in the trial.

Three users were given access to the Polya-Lamport version of the notes. These users were first year mathematicians at St. Edmund Hall, Oxford University who were currently studying the topology course and so who would have a real motivation to use the online materials made available to them.

The users had access to the online materials for three weeks before they were then interviewed. During the interview, the online materials were available on a lap-top as prompt because users are not always explicitly able to remember software during interviews [7]. The interview was intended to elicit qualitative data on the experience of using the Polya-Lamport framework, including the hierarchical proofs. The interviews were recorded on MiniDisc for subsequent analysis. Though three users is not enough for any rigorous trial, it is enough to begin a qualitative analysis of the data and to start the formation of a grounded theory as part of a larger investigation [22]. Specifically, we can formulate hypotheses to test in more focused experiments.

The results presented here are just the analysis of the online materials that involved the interactive, hierarchical proofs.

One of the most noticeable outcomes of the interviews were that the users quickly recognised the hierarchical structure even though they had not been primed to think of it as such:

² Except in Internet Explorer — as usual, Microsoft’s interpretation of a standard (the HTML DOM) differed from everyone else’s.

INTERVIEWER: In terms of the proofs you've already mentioned . . . what do you think about how they are laid out?

SUBJECT 1: They seem laid out quite well. Very easy to see where it is going.

And even one student who declared himself as “not very good with computers” recognised what was going on:

INTERVIEWER: So you are not used to seeing that [format]?

SUBJECT 2: No, but once I worked it out, it seemed okay.

Interestingly, though Lamport has carefully developed a numbering system for the steps of the proof, none of the students particularly noticed it or felt it was useful. This may be that the hierarchy implied by the pretty printing of the proofs was sufficient to get the idea and the numbering did not add any further information.

Lamport also used the numbering to refer to different parts of the proof within a given proof step and these corresponded to links in our proofs as described earlier. One user thought that this could be confusing. Another user experienced that confusion:

SUBJECT 3: If you need to click that to [go] back to the actual material. I just can't concentrate... to the program.

This wasn't a particular feature of the hierarchical proofs. Any links in the framework which took the user away from the focus of their attention tended to be considered negatively.

As for the hiding and revealing of proof substeps, the response was rather mixed because of the summaries of the proofs. The summary proofs did collect together all the statements that the hidden proof steps used and referred to them by the numbers that the rest of the proof used. However, this could easily turn into a list of numbered statements and, without the actual statements there, tended to be off-putting:

INTERVIEWER: What was difficult about seeing it online? . . .

SUBJECT 2: I would like to have... more words.

In addition, because of problems with the XSLT used to generate the summaries, the numbering of the statements came up in the order in which they were used in the proof. This meant that a summary might appear as “By background 3, background 2, background 1” which had a tendency to distract the users:

INTERVIEWER: How did you find the numbering of the theorem? . . . SUBJECT 1: I found it a bit bizarre...

There also seemed to be a deeper problem with the content of the hierarchical proofs themselves. The aim of the hierarchical structure was to reveal the more detailed steps as the reader descended the hierarchy. However, though the principle was recognised it was not understood that way:

SUBJECT 3: It is not very detailed, I think. . . [I found] more details but not very clear.

SUBJECT 2: ... it is not all out there.

It seems that the details were not in a recognisable form or that what was there was not the “whole proof” despite being logically more complete. This gave the tendency to “skim-read” and “go through it fast”.

On the positive side though, the hope of giving the user control over how much detail they read was realised to some extent. One user liked it:

SUBJECT 2: It was helpful to shrink it down . . . Because you could expand the parts to see exactly what you wanted. It is good to have that flexibility.

In summary then, the second user trial gave some initial pointers to problems of hierarchical proofs. Though the structure of the proof is easily recognised and manipulated, it does not seem to be easily understood. Some of this may be due to artificial elements introduced such as the numbering and summaries of proofs. But another factor is somewhat deeper than this. The steps of the proof given do not seem to have the right sort of details even though they are logically more detailed than traditional proofs. It seems to be the quality of the details and not the quantity of the details that is important for the readers. However, this conclusion can only be tentative and we turn to a more theoretical approach to look for further insights.

4 Theoretical Analysis Over Cognitive Dimensions

Cognitive dimensions were developed by Thomas Green and others to reflect aspects of information representations that can be considered independently, though in a given situation may be mutually constraining. Where they differ from guidelines and heuristic evaluation [17] is that they consider inherent features of working with an information representation, rather than the more superficial aspects of user interfaces. Moreover, they are motivated and inspired by findings in cognitive science, though the logical path from cognitive science to the cognitive dimensions is tenuous. A detailed description of the technique and its application to evaluating visual programming languages can be found in [10].

In the analysis, we consider both the structure and the interaction of the hierarchical proofs, often by comparison to traditional proof presentations. When performing the analysis, it might be that we found using cognitive dimensions what we were looking for from the results of the interviews. To avoid this, the second author separately analysed the proofs using the cognitive dimensions before learning either of the findings from the interviews or the first author's analysis. The analyses corresponded closely.

4.1 Applying the Dimensions

In the sections below we analyse hierarchical proofs over most of the cognitive dimensions. Four of the dimensions are omitted as they would refer to editing proofs which is not the current focus. The fifth, role-expressiveness is also omitted as it is not clear what it means — a view that Green himself also holds.

Abstraction Gradient: Abstraction gradient refers to how much abstraction needs to be learnt by the user before they can understand the representation of the proof. This is independent of the amount of abstraction used in the proof or the underlying mathematics.

Any hierarchical proof requires some abstraction as the breakdown of the proof into a hierarchy is any abstraction of the logical process. Furthermore, the steps also have individual labels such as GIVEN for assumption and LET for definitions in the proof. These must all be learned before the proof can be understood.

However, the abstractions used rely on other notations either within computing or mathematics. The hierarchical structure is common and the terms used to denote elements of the hierarchy have been taken from mathematics. Thus, though there are some abstractions to be learnt and these can be off-putting to new users, they are not so far removed from common mathematical experience and so should not pose any particular barrier.

Closeness of Mapping: The closeness of mapping is about mapping the representation of the proof to the process of proving. The closer the mapping the easier it is for users to relate the proof to their understanding of the problem.

This may be a serious flaw. Hierarchical proofs do not closely map to traditional proofs — even when they have the same content they can be considerably re-organised from the original traditional format. Or the closeness of mapping may in fact relate to how closely the proof represents the process of proving. In this case, both hierarchical proofs and traditional proofs can have no claim to demonstrate the process of proving other than presenting the logic of the proof. Traditional proofs may highlight key steps and so help guide the reader to the process but this element is absent from hierarchical proofs.

Hence, in both ways of understanding this dimension, hierarchical proofs do not map closely to the user's understanding of the problem. This may only be a matter of change of culture from traditional proofs or it may emphasise a problem that is already met in traditional proofs.

Consistency: Consistency is the extent to which users can transfer their experience in one part of the proof to the understanding another part.

Inasmuch as a hierarchy imposes a regular structure on a proof, hierarchical proofs could be said to be very consistent. Certainly much more so than traditional proofs which have no generally accepted structure.

The consistency of hierarchical proofs could help users to learn the representation more quickly and to extract information more quickly from unfamiliar proofs.

Diffuseness/Terseness: Diffuseness and terseness are the quantity of symbols and space needed to represent an element of the hierarchical proof. Because hierarchical proofs use space to represent the hierarchy and special tags to denote type of proof step, the representation is considerably more diffuse than traditional proofs. This can be a problem when screen space is limited.

It could be that hiding parts of the proof reduces the diffuseness imposed on the reader. However, on revealing hidden steps, the diffuseness is also revealed.

Conversely though, if the representation were very terse, the reader must hold more of the proof in the mind and so increase their mental effort. This is often seen in traditional proofs where very few words can be used to mean extremely complex steps [18]. There are elements of terseness in the summaries and

references within the proof where statements are referred to only by numbers. This could give users problems.

Hierarchical proofs, then, are reasonably diffuse but probably not too diffuse to be frustrating to users. However, terseness at certain points in the proofs particularly references to other steps could cause the user a cognitive workload and so hinder them from working on the proof itself.

Hard Mental Operations: Hard mental operations refer to the user having to spend time working to understand the proof because of the hierarchy. Obviously, a complex mathematical proof is complex no matter how it is presented and the user would have to work at it. This dimension is rather concerned with any mental workload placed on the user by the presentation itself.

There is some mental workload due to having to remember statement numbers as mentioned above. However, in addition to that, we suspect that there may be some extra workload as a consequence of reconstructing the proof from a hierarchical structure. Users are used to reading text and so constructing a proof from a sequential presentation. The hierarchical structure does not provide such easy navigation through the proof and consequently the user may need to work in order to put the proof in a better form.

This is only conjecture and would need to be tested through psychological experiments about how a person understands a proof.

Progressive Evaluation: Progressive evaluation is the ability of the user to assess their progress through the proof either in terms of how much they still have to read or how much they still have to understand. For novice users, progressive evaluation is essential and it can still be useful to expert users.

Hierarchical proofs, by hiding parts of the proof, certainly prevent the user from seeing the amount of proof that is still left to read. Yet the aim of the hiding is to allow users to choose how much they wish to see. Thus, hierarchical proofs prevent progressive evaluation in one sense but then if the user is satisfied with their progress, they do not need to know how much more they might have progressed had they revealed even more proof.

In contrast, it is easier to see how much of a traditional proof there is and therefore how far the user has progressed throughout. However, in neither case do the proofs help a reader identify how much understanding they still have to do. But then as this is most probably different for every reader, this is almost certainly impossible.

Thus, progressive evaluation in one sense is not possible using hierarchical proofs but it is arguable that users have control over the degree to which they wish to progress.

Secondary Notation: Secondary notation addresses the amount to which the proof supports notation that is not formally part of the proof. This can help a reader both to understand the hierarchy and to make their own notations on a proof.

The hierarchical proofs use secondary notation, namely pretty printing, to denote the hierarchy. The tags also indicate the type of statement that is being presented. Thus, these cues should be useful to the reader. However, as imple-

mented, there is no facility for readers to enter their own annotations on a proof. This means that they cannot record their own insights or informally restate the proof. The sketch proof in the Polya Lampert framework is intended to give an informal presentation of the proof but this is not part of the hierarchical proof nor may the reader change it. Furthermore, there is no place for diagrams in the hierarchical proof — they are external to the proof itself and thus cannot be used by the reader without a shift of focus and attention.

The lack of secondary notation under the user's control is a limitation of the implementation as given. It could cause some difficulties in helping users to work through the proof. Despite this, the secondary notation in the layout of the proof could be valuable in helping them to recognise the hierarchical structure.

Visibility and Juxtaposition: Visibility is about how much of a proof can be seen at any one time, that is, without cognitive work. Juxtaposition is allowing different parts of a proof to be placed side by side for comparison and hence makes up a significant component of visibility.

It is possible to reveal all parts of a proof but usually, because of the structure imposed on the proof it is unlikely that it would all fit in one window. It may be that such a full view of the proof would not actually be useful but this may not be an issue. Also with regards to visibility, the links out of the proof using the numbering hides some of the information about what the link is for. This means that users would have to work to remember the numbered statement or to move their attention away from their current focus in order to retrieve the actual statement referred to.

Juxtaposition is not built in to the implementation but it would not be a problem for users to re-open the proof in another window and so bring different parts of the proof together. However, because of pretty printing or slight differences, it may be that the different parts may not be easily compared. This may contrast with a traditional proof where the similarities and differences could be made explicit without the constraints of fitting into a hierarchy.

Hierarchical proofs, then, have good visibility but the links can hide useful information for understanding a proof. The different parts of a proof can be juxtaposed but the hierarchical structure may prevent easy comparison.

4.2 Outcome of the Analysis

The good usability features identified by the cognitive dimensions are listed here:

- A hierarchy improves consistency in the structure of proofs
- A user need not worry about seeing the whole proof only as much as they need (progressive evaluation).
- Hidden dependencies in the proof are made explicit.
- Reduces error. HOW???

Features of hierarchical proofs that might cause problems are:

- Introduces several abstractions which may take time to learn.

- Making the logical structure explicit and using indentation to indicate structure increases the diffuseness of the proof significantly.
- Cross-references become harder as the information is far away from where it is needed, increasing cognitive load. Traditional proofs either place the information nearby or use a memorable reference.

We don't know enough about closeness of mapping — this depends on how people understand proof. It could be better or worse than traditional proofs.

5 Conclusions and Future Work

Comparing the analytical findings with the interview findings, it is noticeable that the hierarchical structure was easy to recognise by users and that users can select what they want to see. However, the problems of referencing statements within the proof is confirmed by the cognitive dimensions. The users also seemed to dislike the steps and details that were given in the hierarchical proofs. From the cognitive dimensions, this may be simply because they do not map to traditional proofs and with time and accustomisation, this problem will go away. However, it may be a deeper problem related to the understanding of a logical argument. Though hierarchical proofs may have this problem, it may well be the case that traditional problems have the same problem only it is not addressed.

A more fundamental problem with hierarchical proofs may be that the content of the steps is not appropriate. As our proofs were based on traditional proofs, using similar language and symbols, the problem is not just one of presentation. It is something to do with how a proof is represented in a hierarchy. It is possibly not about the amount of logical detail in the proofs but rather what is being communicated in the proof.

Of course, the hierarchical structure that we used is just one of several possible even within the Lammport proof style. Other proof styles might be more accessible. In particular, it would be interesting to investigate if a hierarchical proof based on Leron's approach would have the same problems.

It may be that the problem of hierarchical proofs found here is more a problem with individual learning styles. However, the analysis on cognitive dimensions seems to point away from this. They are cognitively motivated, general guides on usability and they indicate problems similar to those actually found with the users.

The only way forward with this is to make further investigations into how users perceive and work with hierarchical proofs and how they compare with traditional proofs. We are planning to do a larger investigation on just this aspect of the Polya-Lammport framework involving both qualitative and quantitative feedback on the usability issues.

Regardless of what might be root cause of the usability problem of hierarchical proofs, the key lesson is that a hierarchical proof is not *a fortiori* the best presentation for proofs. There may be many and complex reasons why they are not desirable at the moment. If it is simply a matter of familiarity then steps

must be taken to introduce them earlier in a mathematical career rather than later. In any case, systems for mathematical knowledge management should be cautious about placing them as the main interface between mathematicians and their proofs.

Acknowledgments

Many thanks to the undergraduate mathematicians at St. Edmund Hall, Oxford University, who took part in the user trials. This work was supported by EPSRC, UK, under grant GR/N29280.

References

1. R. Bornat, B. Sufirin, 'Animating Formal Proofs at the Surface: The Jape Proof Calculator,' *The Computer Journal*, 42(3), 1999, p177-192
2. T. Bray, J. Paoli, C.M. Sperberg-McQueen & E. Maler (eds) , *Extensible Markup Language (XML) 1.0, 2nd edn*, W3C Recommendation October 2000
<http://www.w3.org/TR/REC-xml>
3. P. Cairns, J. Gow, 'On Dynamically Presenting a Topology Course,' *First International Conference on Mathematical Knowledge Management*, 2001
4. P. Cairns, J. Gow, 'On Dynamically Presenting a Topology Course,' submitted to *Annals of Art. Intell. & Math.*, 2002
5. J. Clark (ed), *XSL Transformations 1.0*, W3C Recommendation November 1999
<http://www.w3.org/TR/xslt>
6. K. Easthaughffe, 'Support for Interactive Theorem Proving: Some Design Principles and Their Application,' in R. Backhouse (ed.), *Workshop on User Interfaces for Theorem Provers*, 1998, p96-103
7. X. Faulkner, *Usability Engineering*, Palgrave, 2002
8. Green & Blackwell, A tutorial on cognitive dimensions
9. P. R. Halmos, *Naive Set Theory*, Springer, 1960
10. T. R. G. Green, M. Petre, ' Usability analysis of visual programming environments: a "cognitive dimensions" framework,' *J. Visual Languages and Computing*, 7, 1996, p131-174
11. J. Grundy, 'A Browsable Format for Proof Presentation,' *Math. Universalis*, 2, 1996. <http://www.ant.pl/MathUniversalis/2/grundy/mu.html>
12. A. G. Hamilton, *Logic for Mathematicians, revd edn*, Cambridge University Press, 1988
13. J. Harrison, 'Formalized Mathematics,' *Math. Universalis*, 2, 1996
<http://www.pip.com.pl/MathUniversalis/2/harrison/jrh0100.html>
14. M. Jackson, H. Lowe, 'XBarnacle: Making Theorem Provers More Accessible,' *CADE-17*, LNAI 1831, Springer, 2000, p502-506
15. L. Lamport, 'How to write a proof', *American Mathematical Monthly*, 102(7) p600-608, 1994
16. Uri Leron, 'Structuring Mathematical Proofs,' *American Mathematical Monthly* 90(3), 1983, p174-185
17. J. Nielsen, *Usability Engineering*, Morgan Kaufmann, 1993
18. L. C. Paulson & K. Grabczewski, 'Mechanizing Set Theory,' *Journal of Automated Reasoning*, 17 p291-323, 1996

19. G. Polya, *How to Solve It*, Penguin Books, 1990
20. W. Rudin, *Functional Analysis, 2nd edn*, McGraw-Hill, 1991
21. J. Siekmann, S. Hess, C. Benzmüller, L. Cheikhrouhou, H. Horacek, M. Kohlhase, K. Konrad, A. Meier, E. Melis, M. Pollet, V. Sorge, 'L Ω UI: Lovely Omega User Interface,' *Formal Aspects of Computing* 11(3), 1999, p1-17
22. A. Strauss, J. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing a Grounded Theory*, Sage Publications, 1998