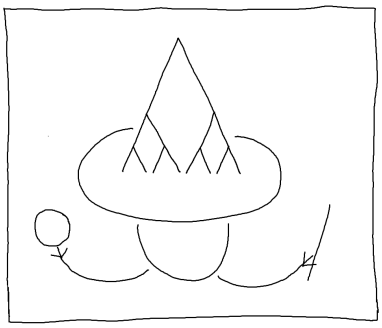

Treehehe

An interactive visualization of natural deduction proof trees

Author: Chelsea Battell
Kind of Project: Design/Development

1 Reading Review

Data in this visualization project are trees, so exploring visual formalisms for trees is an important first task. Herman et al. [4] discuss a number of techniques for visualizing trees. Several space-saving representations are presented, such as H-tree layout, radial view, and balloon view. In Treehehe, much of the utility of a proof tree comes from being able to read a flow of inferences, seeing the leaves as axioms or assumptions and the root as the conclusion of the proof. Each node contains a propositional formula, so to be readable all nodes need to have the same orientation. The space-saving layouts could be useful if it is clear which node is the root and some other form of iconography could be used for constructing formulas. For the current project, we will use the Reingold and Tilford layout for trees, which could be described as visually rooted, since we have a convention that tells us which node we see as the root. We will invert the standard tree constructed with this algorithm so that the root is at centre bottom. Another benefit of the Reingold and Tilford algorithm for this application is the vertical alignment of nodes at the same level (distance from the root) in a proof.

Since proof trees can easily escape the bounds of a display, navigation is another topic explored by Herman et al [4] that is relevant to this project. Zooming is not necessary because there is no refinement of information to be seen. The user will want to be able to scroll around the graph while exploring. The technique of incremental exploration and navigation will be used to display only a desired subtree of the proof.

In [7] we learn many best practices from Ware for using colour in a visualization. In Treehehe, colour will be used to highlight a focused node in a proof tree. One mode of interaction in this project will be a walk-through of a proof. In a walk-through, low saturated colours and small luminance contrast will be used to make the past and future inferences less salient and more difficult to read. Then the focus of the user will be on the current inference of interest.

Interaction is an essential component of this project because it elevates a proof tree from a static graphic to a visualization that can lead to further understanding and insight along a variety of paths. Duval claims that proof trees are disappointing and visualizations in general do not lead to immediate comprehension in mathematics [2]. It appears that what is meant by *visualization* in this article only includes static graphics representing mathematical objects. He also does not fully explore other advantages to explicit articulation of proof trees, such as cognitive offloading and observation of patterns in large proofs, which are particularly useful in proofs where the goal to be proven has a complicated logical structure. Here we propose that adding interaction to a proof tree will aid in understanding of both the logic used to build it and the proposition being proven, although this is stated without evidence and without a study to verify this hypothesis. There is more to discuss relating Duval's work to what is being attempted here, and this will be expanded on in the final report.

Figueiras presents a taxonomy of interaction consisting of eleven categories of techniques [3]. The categories of interaction that will be used in this presentation are: abstract/elaborate, select, and overview/explore.

The interaction technique *abstract/elaborate* is used to adjust the level of abstraction of the data. Data in Treehehe are trees, where each node contains a proposition and transitions are labeled with a rule name and possibly side conditions. Two alternative views will be available for a proof tree. One will display the tree with all detail, and the other will only show the rules that are used, allowing a more abstract and structural perspective of the tree.

A user may want to *select* data to learn more about it or track how it changes in response to other interaction. In Treehehe, a node can be selected to see more detail on its role in the proof tree. Its related parent and child inferences can be highlighted, the rules used displayed, and the substitutions used in applying the rules explicitly stated.

Techniques in the *overview/explore* category first display an overview of the data, then allow exploration through zooming, filtering, and the display of details on demand. Large proof trees cannot be easily displayed on a monitor. In this situation, it may not be possible to have a full proof tree with all details, thus no proper "overview", but there will be icons to allow the minimizing and expansion of subtrees and some form of panning to allow exploration of subtrees of the proof. The abstract, structural view mentioned above should help the user to see a condensed view of the tree.

Lin and Yang [9] present five facets of proof comprehension: basic knowledge, logical status, summary, generality, and application. These facets will be used to guide the design of this project.

Yi et al. observe that sensemaking is one path to insight [10]. Sensemaking, in the context of interactive visualizations, is an intentional process in which a person continually reframes their understanding of a concept. The authors also propose four processes for gaining insight: provide overview, adjust, detect pattern, and match mental model.

Provide overview means a user can gain a higher-level understanding of a data set. In Treehehe, this is realized in the viewing of a complete proof tree. In the case of a large proof tree, this may have to be restricted to the "structure view" mentioned earlier, where only the rule names are visible to show the structure of a proof but no other detail.

Through *adjust*, a user is able to explore a data set. This can come from a variety of interaction techniques, such as adjustments to the level of abstraction, or selecting a range of values. In Treehehe, the alternation between detail and structure view and the hiding of subtrees can help the user develop insight about the logic and formula proven.

When new structure is observed in data and possibly new discoveries made, the user has been able to *detect* a *pattern*. In Treehehe, examples of how this is realized include observing repeated arguments in a proof and being able to detect loops.

Match mental model means one has a bridge between the data and their mental model of it. Having an external visual version of the mental model allows for cognitive offloading. In this project, we will see the value of having a visualization of a proof to interact with rather than trying to hold the entire picture in one's head. It is then possible to allocate more mental resources to reasoning and gaining insight through the other processes.

2 Detailed Description

2.1 Problem Domain

Proofs are a naturally visual exercise and an essential part of comprehension and advancement in logic. Proof trees are a common representation used to illustrate both the structure and details of a logical argument. There are many advantages to be found using proof trees: they are useful for learning about a logic through experimentation with writing proofs, they reveal structure in the proof that a linear proof presentation may obscure, they allow cognitive offloading when working through a challenging argument, and they can serve as a form of documentation. Standard presentations of proof trees are static. Adding interaction to such visualizations makes it easier to realize the advantages mentioned above.

2.2 Use of Visualization

There are critical limitations in the use of proof trees in certain mediums. Even a straightforward, "small" proof tree can easily escape the bounds of a sheet of paper. Creating a digital version eliminates bounds on the size of the tree, but it is still desirable to have all pertinent parts visible on a screen at the same time. A proof tree in the visualization created in this project will be an interactive, digital version where subtrees can be hidden. This will be done by selecting an icon on the node that will cause the subtree rooted at that node to be minimized. It will be ideal for the transitions when minimizing or expanding subtrees be as smooth as possible to avoid disrupting the mental model of the data.

A proof tree is a static and declarative object, which requires some level of expertise to be parsed by the reader. In [2], one of Duval's criticisms of proof trees and mathematics visualizations is that they don't aid in operational or discursive comprehension. To overcome this limitation of static proof trees, the user will be able to interact with the tree in two distinct ways.

The first will be in the form of a proof walk-through that will support operational comprehension by using colour to focus attention on the current inference in a proof and distinguish between previously seen inferences and future inferences. Future and past inferences will be coloured to be less visible. Another panel on the page will have some text to provide discourse.

The second form of interaction will allow more free-form exploration of the proof tree. The user can click on any node to focus on it. At this point, the rule used to infer this formula will be visible. Side conditions may be shown, as well as the substitutions for the rule schema variables.

Weber and Mejia-Ramos [8] and Alcock and Wilkinson [1] have argued that too much detail visible in a proof obscures high-level structural information which is important for proof compre-

hension. To address this concern, there will be two views for this proof visualization: detail view and structure view. Detail view will show a proof tree, possibly with subtrees hidden as discussed above, with full formulas, rule names, and rule side conditions. Structure view will show the shape of the tree and which rules are used.

2.3 Design Outline

2.3.1 Approach

The goal of this project is to build a tool for visualizing and interacting with proof tree representations of natural deduction proofs. Before such a proof can be built, a set of rules of the logic must be determined. These rules have the following form:

$$\frac{premise_1 \quad \dots \quad premise_n}{conclusion}\ \text{rule name}$$

The meaning of this rule is, if $premise_1, \dots, premise_n$ are all true, then *conclusion* is true. The rule name is to the right of the horizontal line separating premises from conclusion. A concrete rule example is the rule $\wedge_I$ for building a proposition that contains the symbol $\wedge$ which means "and" and its use is called a conjunction:

$$\frac{P \quad Q}{P \wedge Q}\ \wedge_I$$

This rule says "if P and Q, then $P \wedge Q$", consistent with our understanding of the meaning of the symbol $\wedge$. The variables P and Q are called schema variables (sometimes called metavariables or syntactic categories). To use this rule requires some pattern matching. If the desired substitutions for P and Q are known, then the proposition in the conclusion is $P \wedge Q$ where P and Q are substituted as required. The rule $\wedge_I$ for conjunction introduction is part of the set of rules for natural deduction, the logic that will be used in Treehehe. Our presentation of natural deduction will follow the lecture notes of Frank Pfenning [6] with labels on assumptions in hypothetical premises. See Appendix A for the natural deduction rules that are used in this report.

Suppose we wish to prove that from the n assumptions $P_1, \dots, P_n$, we can derive the conclusion C. This goal can be written as $P_1, \dots, P_n \vdash C$, called a sequent. If, by applying rules from our logic and making appropriate substitutions, we are able construct a tree with all of the P_1 to P_n as leaves and C as the root, then this proof tree is evidence for the validity of the sequent $P_1, \dots, P_n \vdash C$. See Figure 1 for an example proof of a sequent.

The medium for this project will be a web page. The page will have the following regions: example selection panel, main tree view panel, header, information panel, and rules panel. A sketch of the web page can be seen in Figure 2.

From the example selection panel, the user can select an example proposition. Examples will be drawn from Logic in Computer Science by Huth and Ryan [5] to ensure a breadth of examples from a textbook presenting natural deduction at an introductory level. Once a proposition is chosen, its proof tree will be displayed in the main tree view panel.

The proof tree for the selected proposition will be displayed using a variant of the Reingold and Tilford algorithm, inverted so that the root of the proof tree, the formula derived, is at bottom center. See Figure 1 for an example proof tree. In this tree layout, the root of the tree is visually

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p \wedge r}\,u}{p}\wedge_{E_1} \qquad p \supset q}{q}\supset_E \qquad \dfrac{\dfrac{}{p \wedge r}\,u}{r}\wedge_{E_2}}{q \wedge r}\wedge_I}{p \wedge r \supset q \wedge r}\supset_{I^u}$$

Figure 1: An example proof of $p \supset q \vdash p \wedge r \supset q \wedge r$, formatted in LaTeX

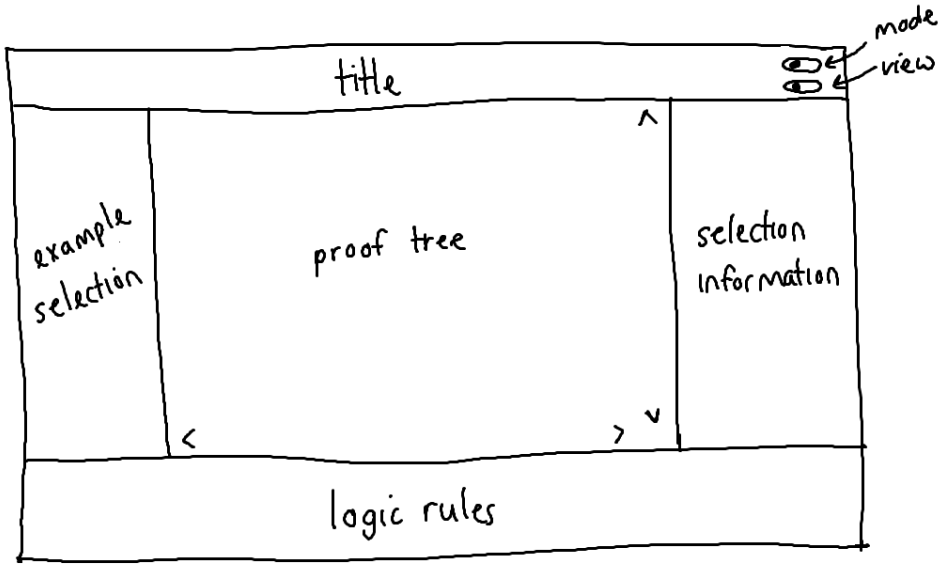

Figure 2: A sketch of the layout of Treehehe

salient; it is isolated at the bottom point of the tree, and we can always depend on the conclusion of the proof tree appearing in that location. The example proof is laid out by LaTeX and is an example of how such a tree is typically presented. To allow interaction, we need each node of the tree to be an accessible object. From this style of proof presentation, we can see that the proof resembles a tree, but it is not immediately clear how each of the components should be represented in a tree abstraction.

There is a subtlety to how we view proofs as trees. To ease implementation and future discussion of proof tree structures, additional formalisms are proposed in interpreting proofs as trees. Each node contains a proposition, which is something that is either true or false. Propositions that serve as assumptions in hypothetical premises of rules such as $\supset_{I^u}$ require a label to reference when these assumptions are discharged, so a node may also contain a label. An edge between two nodes is directed from the proposition used as a premise to the proposition being inferred. Each edge is labeled with a rule name and side conditions that must hold to use the rule, where applicable. See Figure 3 for an illustration of the proof presented in Figure 1 converted to this tree abstraction.

The header will contain the project title and controls to toggle the interaction mode and view

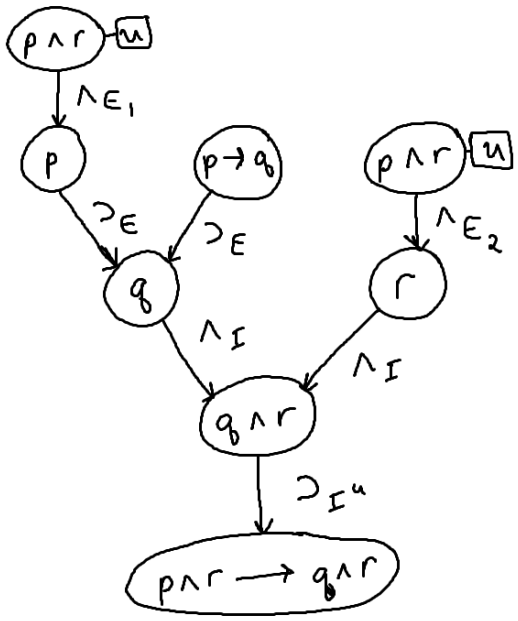

Figure 3: Comparison of the standard proof tree display to its abstract tree representation

type. The interaction mode is either *explore* or *walk-through*. The default mode is *explore*. In this mode, the user can click on any node of the proof tree, causing it to be focused visually and information specific to that node to be displayed on the information panel. This information includes, as applicable, the rule used to infer the current proposition, the rule in which it is used to infer the next proposition in the proof, and substitutions made to the schema variables in the rules to match the focused proposition.

The alternative mode is *walk-through*, which allows the user to step through the proof. At each step, a node in the tree will be visually focused through a use of colour as discussed in the reading review. There will be two buttons to move forward or backward in the proof tree. The information panel will be used in the same way for the walk-through mode as it was for explore mode.

The walk-through will take the user through a forward reading of the proof; that is, from assumptions to conclusion. This is in contrast to goal-directed proofs which work from the conclusion to assumptions using "backward" reasoning. There are benefits to goal-directed proofs, namely in automated proving and proof search, but their reading can feel unnatural without practice. Our decision to progress from assumptions at leaves to the conclusion at the root brings us to an issue requiring further contemplation: how best to traverse the tree.

A priority is comprehension of the operational semantics of the proof, and since we have decided on a forward presentation of the reasoning, we suggest a variant of a depth-first search for traversing a proof tree. Traversal will begin at the left-most leaf and travel down that branch to the root. Then searching back through unvisited children, the next leaf from the left will be found and the algorithm will repeat. Other tree traversals will be considered if this option does not seem to ease proof comprehension.

The final area on the web page is the rules panel. It will be minimized by default, but can be

opened to show the user the rules of the logic in their most general form, without any substitutions for schema variables.

In the reading review the issue of displaying a large proof on a too-small display was discussed. To allow a more condensed display of the proof that is easier to navigate, two further features will be added. The first uses the technique of incremental exploration and navigation [4] and is an option to minimize subtrees rooted at any node. Each node will have a small button to allow the subtree it roots to be expanded or minimized. The second feature is an alternative view in the *explore* mode. The default view is as discussed above and we will call it *detail view*. The alternative is called *abstract view* and shows only the rules used with no other detail, leading to insight on the structure of the proof. This view will be accessed through a control, possibly a toggle, that will only be visible while in *explore* mode.

2.3.2 Technology

This project will be developed as a web page using HTML, CSS, and Javascript. The page will be hosted using GitHub Pages. These technologies have been chosen because they are relatively easy to quickly develop in and share with a wide audience.

The data are proof trees, and they will be written directly in HTML. It would be desirable to have a visual interface for building proof trees, but this addition is out of scope for this project.

CSS will be used to style the page and for data layout. A few libraries exist for laying out trees but they may not meet the needs of proof tree layouts. Most examples found for laying out trees are to build treeview controls for horizontal systems like those used for visualizing a file system. Flexbox and Grid are CSS layouts used for building responsive layouts and are being explored for building proof trees but without success so far. The alternative horizontal tree layout will be considered, with benefits being that it is simpler to lay out and more compact.

It is not expected that this tool will be very useful on small-display devices such as mobile phones, but the page will still be designed to be responsive to changes in window size. Flexbox will likely be used for this purpose.

Javascript and jQuery will be used for the interaction logic and possibly also for drawing some elements of proof trees if other libraries and CSS tools don't meet the needs of the project.

The MathJax library will be used so that LaTeX can be embedded in HTML. It will be used for the layout of propositions and not full proof trees since each node of the tree needs to accept interaction.

I have used these technologies for a few small to medium sized projects in the last four years. In 2014 I completed a project in a compilers class where I wrote a compiler for a version of a knitting language. I also made a small GUI that allowed a user to push buttons to build a knitting pattern and then check that the pattern is correct (has no errors according to the standard rules of knitting). The pattern was then compiled to the expanded form of knitting pattern that a knitter can work from. My most recent project using web technologies was a website to use for a presentation and future reference on the theory behind various logic programming languages, culminating in programming with higher-order logic). I also have a small static personal web page.

2.4 Plan

Week of	Tasks
October 14	• planning: review literature on visualization in math, proof visualization, proof comprehension
October 21	• proposal: draw designs and finish writing
October 28	• implementation: choose CSS or Javascript frameworks to help with the layout of trees • implementation: enumerate examples to include in the project
November 4	• implementation: initial layout of page, including supplementary content for selected node • report: document structure and abstract
November 11	• implementation: logic and styling for proof walk-through • report: notes for content of each section
November 18	• implementation: final decisions for colours and fonts • presentation: plan content of talk and begin slides • report: notes for content of each section
November 25	• implementation: brief logic tutorial, time permitting • presentation: finish preparation for talk • report: writing
December 2	• report: writing
December 9	• report: writing
December 16	• report: writing

A Natural Deduction Rules

Below are the natural deduction rules that are used in the report. This is not the full set of natural deduction rules.

$$\frac{P \quad Q}{P \wedge Q}\ \wedge_I \qquad \frac{P \wedge Q}{P}\ \wedge_{E_1} \qquad \frac{P \wedge Q}{Q}\ \wedge_{E_2}$$

$$\frac{\begin{array}{c}\overline{P}^{\,u} \\ \vdots \\ Q\end{array}}{P \supset Q}\ \supset_{I^u} \qquad \frac{P \quad P \supset Q}{Q}\ \supset_E$$

- The rule $\wedge_I$ is used to construct a proposition that is a conjunction of two propositions P and Q, given P and Q can both be assumed or derived
- The rules $\wedge_{E_1}$ and $\wedge_{E_2}$ are used to derive a proposition with a conjunction eliminated given a proposition that contains a proposition as the top-level logical symbol
- The rule $\supset_{I^u}$ is used to construct a proposition that is an implication between two propositions P and Q given a derivation that allows us to derive Q if P is assumed; the assumption P is given a label u so that it is clear which assumption is being discharged in any use of this rule
- The rule $\supset_E$ is used to derive a proposition with an implication eliminated; given the premise P and a premise with P as the antecedent, $P \supset Q$, we can conclude Q

References

[1] Laura Alcock and Nicola Wilkinson. e-proofs: Design of a resource to support proof comprehension in mathematics. *Education Designer*, 1(3), 2011.

[2] Raymond Duval. Representation, vision and visualization: Cognitive functions in mathematical thinking. basic issues for learning. In *Proceedings of the Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*, Cuernavaca, Morelos, Mexico, October 1999.

[3] Ana Figueiras. Towards the understanding of interaction in information visualization. In *2015 19th International Conference on Information Visualization*, July 2015.

[4] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, January 2000.

[5] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, The Edinburgh Building, Cambridge CB2 8RU, UK, second edition, 2004. Chapter 1.

[6] Frank Pfenning. Lecture notes on natural deduction. For Constructive Logic course, August 2009.

[7] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufman, third edition, 2013.

[8] Keith Weber and Juan Pablo Mejia-Ramos. Mathematics majors' beliefs about proof reading. *International Journal of Mathematical Education in Science and Technology*, 45(1):89–103, 2014.

[9] Kai-Lin Yang and Fou-Lai Lin. A model of reading comprehension of geometry proof. *Educational Studies in Mathematics*, 67(1):59–76, January 2008.

[10] Ji Soo Yi, Youn ah Kang, John Stasko, and Julie A. Jacko. Understanding and characterizing insights: How do people gain insights using information visualization. In *Proceedings of the 2008 Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*. ACM, April 2008.