Studying Steps

Steve Kieffer[1]

[1]Alpine Mathematics, https://alpinemath.org/

Abstract

Students of mathematical proofs need to understand proof steps, both individually, and as they constitute a whole proof. How can the steps be best presented, in order to facilitate study and comprehension? We examine different ways of presenting and studying proof steps in the PISE software, and consider usability questions, including: How should steps be arranged visually (list or layered layout), how should expansions be inserted (what we call unified or embedded mode), what are the most usable ways to link proof steps to enrichments (including links to original literature, narrative guides, and example exploration using a computer algebra system), and how should students be supported in actively recording and reviewing their own progress, questions, and notes as they study.

Keywords

studying proofs, argument maps, annotation software, multiple views, linked views, usability, integrated development environments, PISE

1. Introduction

Whether represented in formal logic, or in informal natural language, mathematical proofs can perhaps always be thought of as broken down into *steps*. If we can say, "From A follows B," or "From P and Q taken together follows R," then we have a logical argument in the form of steps.

The steps form a natural *directed acyclic graph* (DAG) in which the *directed edges* represent logical inference. "From A follows B" becomes

$$A \longrightarrow B$$

while "From P and Q taken together follows R" becomes

$$\begin{tikzcd} P \arrow[dr] & \\ & R \\ Q \arrow[ur] & \end{tikzcd}$$

Meanwhile, the *nodes* of the graph are where the actual mathematical content lies. Node P might say that "$n = u \cdot v$," while node Q says that "v is an integer," and node R concludes that "u divides n."

When we represent proofs this way, we are dealing with *argument maps*, and if these are to be displayed on a computer screen, then their use poses various usability challenges and questions. But argument maps are useful tools in many fields of study besides mathematics (such as law or philosophy [1]) and our goal in the present paper is to focus on a set of usability questions that are relevant when argument maps are applied in the study of mathematical proofs.

In particular, we will be looking at usability in the open-source PISE (Proofscape Integrated Study Environment) software [2], which is essentially an integrated development environment for the Proofscape argument mapping and annotation system [3] (but with an emphasis on *studying* proofs, hence "ISE" instead of "IDE"). Besides browsing argument maps, PISE also supports linking these maps to narrative guides, and to original proofs from the mathematical literature.

MathUI'24: Mathematical User Interaction, August 9, 2024, Montreal, QC, Canada
skieffer@alpinemath.org (S. Kieffer)
https://skieffer.info/ (S. Kieffer)

1.1. Questions

The specific usability issues we will consider are:

1. **Layout and the Order of Exploration:** Argument maps can be laid out in different ways. Which ways are best? Where order of exploration matters (e.g. viewing nodes where symbols are *introduced* into the proof before viewing nodes where those symbols are *used*), how is this affected by different layout methods?
2. **Expansions:** Extra steps can be spliced into argument maps, in order to clarify difficult inferential leaps. How should such "expansions" be drawn, and how do different drawing styles affect usability?
3. **Content Linking and Multiple Views:** Besides argument maps, students can be presented with narrative discussions of proofs, and archival documents showing proofs as they appear in the literature. All these content types can be linked, and viewed in multiple panels. What are the usability issues that arise?
4. **Example Explorers:** Can we provide tools to automatically illustrate proofs, by helping students to explore numerical examples of the mathematical objects and relations involved at any given step?
5. **Tracking Progress and Taking Notes:** What are the most helpful ways to support students in recording and reviewing their own notes, as they progress through their study?

1.2. Goals

A primary goal of this paper is to present the capabilities of PISE in helping students to study proof steps. Secondarily, it is hoped that the questions we articulate regarding each interaction technique may be viewed as illustrating essential issues for any such software, not just PISE. For, beneath the particular design decisions made in PISE there seem to lie some irreducible problems: If argument maps are to be drawn, there must be *some* layout; if maps are to be linked to enrichments, the links have to work *somehow*, and so on. Readers are therefore asked to view PISE as implementing one reasonable approach to such challenges, while viewing the usability questions raised in each part of Section 3 not just as questions about how PISE should work, but as potentially relevant to any software designed to serve similar purposes.

2. Background

2.1. Multiple Views, Linked Views, and Annotations

In any IDE, one of the main features is the simple ability to open a single piece of content in multiple, side-by-side views. (For example, this allows a coder to work on two different parts of a single source file at the same time.) This use of *multiple views* lies at the heart of many of our questions.

Indeed, while we do have questions involving alternative network layout methods, the layout methods themselves are not our focus here; instead, we are interested in what happens when the user displays the same argument map in multiple, side-by-side views, employing one layout method in one view, and another method in the other view (see Section 3.1).

In such situations, one of the main UI techniques that becomes of interest to us is what is called *brushing* [4]. This refers to any interaction in which the user selects or highlights an object in one view, and the system causes the same object to become similarly highlighted in another view. Once brushing techniques are involved, then we are talking not just about multiple views, but about *linked views*.

Linked views, meanwhile, are not always multiple views of the *same* content; at other times, two pieces of different but related content are viewed simultaneously, and the system will again set up some type of linking, this time between *corresponding* (not identical) objects in the two views. In our case, we are interested in linking argument maps to narrative discussions of proofs, and to original proofs from the literature (see Section 3.3). In so doing, we are concerned with a type of *annotation software* [5].

2.2. Stability of the Mental Map

Another central issue is *stability of the mental map* [6]. Many of our techniques, such as viewing a single proof in two different layouts, or expanding a proof to insert extra steps, involve altering an existing layout. We therefore always have to be conscious of the impact this may have on the inner mental map a student may have already formed. We must also ask whether multiple, linked views can help to counteract the problem, by helping the student to associate a new view with an old one.

2.3. Proofscape and PISE

Proofscape is both an *argument mapping* system, and an *annotation* system. It has been reviewed elsewhere [3] in the capacity of argument mapping system, with a focus on its *node types* (e.g. *intro* nodes for introducing symbols into proofs, versus *assertion* nodes for making assertions), and its *edge types* (*deduction* edges for logical inference, and *flow* edges for directing the reader – see Section 3.1). Coverage of Proofscape as an annotation system, along with documentation for PISE, can be found online at https://docs.proofscape.org. In particular, the documentation explains how proofs become represented and browsable within the software.

3. Usability Questions

3.1. Layout and the Order of Exploration

When it is time to draw an argument map, we are engaged in the problem of *network layout* [7], and PISE offers two basic layout options: *layered layout* (Figure 1a) provided by `elkjs`[1] [8], and a simple *ordered list layout* (Figure 1c) in which the steps appear in the same order as in the original prose proof.[2]

Layered layout [9] can provide insight into the *structure* of the proof: the use of premises becomes clear, nodes arrange themselves into lines of reasoning, subgoals may naturally emerge. All this can help the student to organize the proof, and understand the role played by each step.

However, what may be lost in layered layout is *order*, namely, the order imposed on the steps when the proof is written in prose form. This is both good and bad. The prose order is usually in some ways artificial, and giving this up may be a good thing; on the other hand, the prose order serves purposes like ensuring that symbols are introduced before they are used, and losing this can be a disadvantage.

To achieve a layered layout in which some important aspects of order are preserved, Proofscape features *flow edges*. Besides the *deduction edges* (drawn solid) that indicate logical inference, the flow edges (drawn dashed) tell the student which node to consider next (see Figure 1a).

Users may choose to *suppress* flow edges (Figure 1b), which can improve the layout, by simplifying the underlying graph. With fewer connections to draw, it may become possible to produce a more compact layout (and so make better use of screen space), and in some cases a non-planar graph may even become planar (so that edge crossings may disappear). Compare Figures 1a and 1b.

Using multiple, linked views, users may display the same proof in different layouts, with PISE's built-in *brushing* ensuring that the same nodes are always selected in each view. One view can be in ordered list layout, to guide the study, while another can be in layered layout, to show structure.

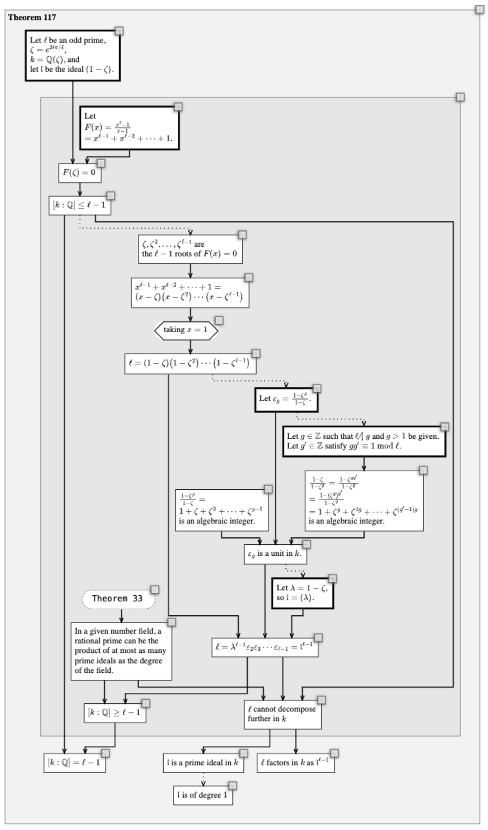
3.1.1. Questions

Do flow edges provide an important aid in exploring a proof in a sensible order? Or, with practice, can users learn to use other cues instead, such as the bold boundaries marking *intro* nodes?

Consider for example the flow edge leaving the assertion node stating the inequality $[k : \mathbb{Q}] \leq \ell - 1$, fourth down from the top in Figure 1a. The same node has two deduction edges leaving it, showing

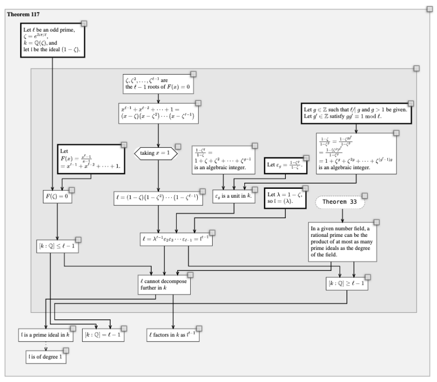
[1] https://github.com/kieler/elkjs/

[2] Figures in this paper are meant to give basic impressions, and, where they may contain illegible text, readers are invited to explore the software online (follow links through https://proofscape.org) in order to see things more clearly. Some proofs and other enrichments shown in these figures are from the Toeplitz Project, https://toeplitzproject.org/.

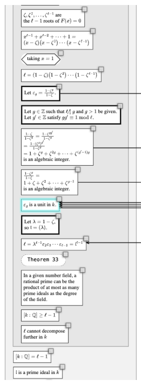


Theorem 117
Let ℓ be an odd prime, $\zeta = e^{2\pi i/\ell}$, $k = \mathbb{Q}(\zeta)$, and let $\mathfrak{l}$ be the ideal $(1-\zeta)$.
Let $F(x) = \frac{x^\ell - 1}{x-1} = x^{\ell-1} + x^{\ell-2} + \cdots + 1$.
$F(\zeta) = 0$
$[k:\mathbb{Q}] \leq \ell - 1$
$\zeta, \zeta^2, \ldots, \zeta^{\ell-1}$ are the $\ell - 1$ roots of $F(x) = 0$
$x^{\ell-1} + x^{\ell-2} + \cdots + 1 = (x-\zeta)(x-\zeta^2)\cdots(x-\zeta^{\ell-1})$
taking $x = 1$
$\ell = (1-\zeta)(1-\zeta^2)\cdots(1-\zeta^{\ell-1})$
Let $\varepsilon_g = \frac{1-\zeta^g}{1-\zeta}$.
Let $g \in \mathbb{Z}$ such that $\ell \nmid g$ and $g > 1$ be given. Let $g' \in \mathbb{Z}$ satisfy $gg' \equiv 1 \bmod \ell$.
$\frac{1-\zeta^g}{1-\zeta} = 1 + \zeta + \zeta^2 + \cdots + \zeta^{g-1}$ is an algebraic integer.
$\frac{1-\zeta}{1-\zeta^g} = \frac{1-\zeta^{gg'}}{1-\zeta^g} = \frac{1-(\zeta^g)^{g'}}{1-\zeta^g} = 1 + \zeta^g + \zeta^{2g} + \cdots + \zeta^{(g'-1)g}$ is an algebraic integer.
ε_g is a unit in k.
Let $\lambda = 1 - \zeta$, so $\mathfrak{l} = (\lambda)$.
Theorem 33
In a given number field, a rational prime can be the product of at most as many prime ideals as the degree of the field.
$\ell = \lambda^{\ell-1}\varepsilon_2\varepsilon_3\cdots\varepsilon_{\ell-1} = \mathfrak{l}^{\ell-1}$
$[k:\mathbb{Q}] \geq \ell - 1$
ℓ cannot decompose further in k
$[k:\mathbb{Q}] = \ell - 1$
$\mathfrak{l}$ is a prime ideal in k
ℓ factors in k as $\mathfrak{l}^{\ell-1}$
$\mathfrak{l}$ is of degree 1

(a) Hilbert's Theorem 117, with flow edges. Several *intro* nodes (bold boundary) have a dashed flow edge leading into them, as does one *assertion* node (fifth node from the top).



(b) The same proof after suppressing flow edges.



(c) An excerpt of the proof in list layout. All edges are suppressed except those incident to the currently selected node.

Figure 1: Layout methods

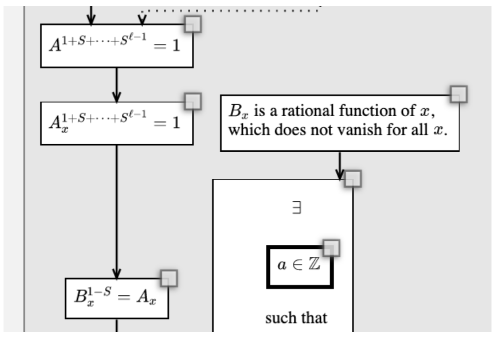
that this node is not further *used* until much later in the argument; the flow edge (pointing to the next node down and to the right) says, "Go and consider this sub-argument first."

While this is helpful, perhaps the user can figure out as much unaided. Using Figure 1b instead, where the flow edges have been suppressed, the user reaches the same node (again, fourth down from the top, on the left), and then, moving forward along one of its outgoing deduction edges (say the one on the right), discovers that they must work backwards from the node so reached. In this way, the same sub-argument can be discovered.

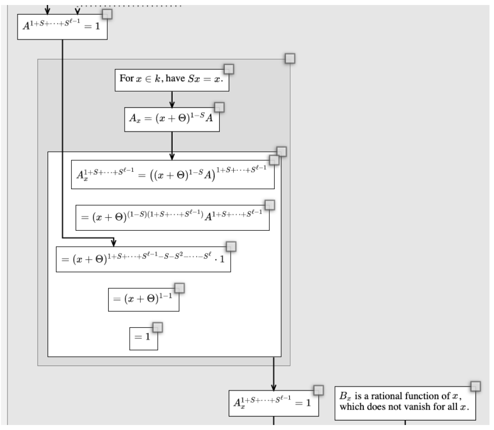
As for introduced symbols, the trained user will remember to seek out the *intro* nodes (bold boundary) first. Still, in cases where one intro node may use symbols introduced by another, flow edges can again be helpful.

What about using multiple, linked views of a proof, with one view in layered layout, and the other in list layout, with the latter guiding the exploration? Is this more or less effective than using layered layout with flow edges?

Finally, what about disruption of the mental map? Supposing the user first familiarizes themselves with a proof *with* flow edges (aiming to benefit from the additional guidance), and then decides to suppress them (aiming to benefit from more compact layout), how significant is the cost of mentally adjusting to the new layout? Do users find the cost prohibitive, or do they find this a beneficial way to study?


$A^{1+S+\cdots+S^{\ell-1}} = 1$
$A_x^{1+S+\cdots+S^{\ell-1}} = 1$
B_x is a rational function of x, which does not vanish for all x.
$\exists$
$a \in \mathbb{Z}$
such that
$B_x^{1-S} = A_x$

(a) A section of an argument map representing the proof of Theorem 90 from Hilbert's *Zahlbericht*, before any expansions have been inserted.


$A^{1+S+\cdots+S^{\ell-1}} = 1$
For $x \in k$, have $Sx = x$.
$A_x = (x+\Theta)^{1-S}A$
$A_x^{1+S+\cdots+S^{\ell-1}} = ((x+\Theta)^{1-S}A)^{1+S+\cdots+S^{\ell-1}}$
$= (x+\Theta)^{(1-S)(1+S+\cdots+S^{\ell-1})}A^{1+S+\cdots+S^{\ell-1}}$
$= (x+\Theta)^{1+S+\cdots+S^{\ell-1}-S-S^2-\cdots-S^\ell} \cdot 1$
$= (x+\Theta)^{1-1}$
$= 1$
$A_x^{1+S+\cdots+S^{\ell-1}} = 1$
B_x is a rational function of x, which does not vanish for all x.

(b) An expansion has been inserted in unified mode, showing how to reach the second node from the top-left in Figure 2a. An edge shows where the previous proof node is used in the expansion.

Figure 2: Expansion in unified mode

3.2. Expansions

When proofs are represented in the form of argument maps, it is easy to *splice in* extra steps, and thereby provide *expansions*, to clarify difficult inferential leaps. Users can be made aware of the availability of an expansion for any given step in a proof, and can elect to display it, if they desire further clarification.

Graphically, there can be different ways of drawing expansions, when the user requests them. PISE offers two expansion modes, called *unified* and *embedded*. Briefly, the meanings of the two modes are as follows:

- Unified mode: expansions are drawn at the same scale as the proof.
- Embedded mode: expansions are scaled down to a smaller zoom level.

3.2.1. Unified Mode

The idea of unified mode is that all proofs and expansions co-exist in a single, unified space, and therefore (a) they are all drawn at the same scale, and (b) edges will be drawn connecting proof nodes and expansion nodes.

Figure 2a shows a segment of a proof diagram. Starting from the top-left of the figure, we have a node which displays an equation $A^{1+S+\cdots+S^{\ell-1}} = 1$, and then below it a node displaying a second equation $A_x^{1+S+\cdots+S^{\ell-1}} = 1$, which the proof claims follows from the first (the definition of A_x is present in the larger diagram, but not visible in Figure 2a).

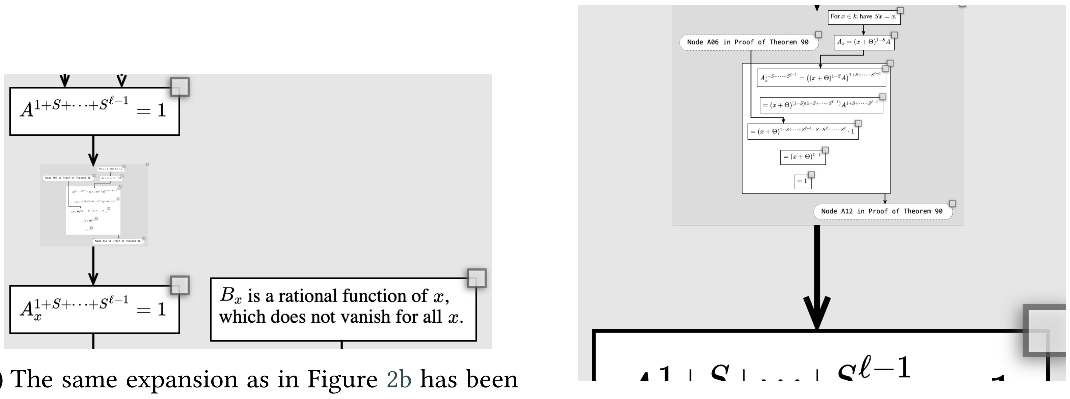
Supposing the user did not see how this follows, they could right-click the second node, and discover that an expansion is available. When the expansion is inserted in unified mode, space is made between the two nodes of the proof, and additional steps are inserted, as in Figure 2b.

3.2.2. Embedded Mode

The idea of embedded mode is that expansions are scaled down and embedded into the proofs on which they expand. If expansion E expands on proof D, then E is drawn at a smaller scale, so that the whole of E is about as big as a single node of average size in D.

Drawing expansions this way makes it so that you have to zoom in to read them. This can be thought of as a visual metaphor, representing expansions as "details."

When the same expansion from Figure 2b is inserted in embedded mode, the result is as in Figure 3a. Much less space is required; however, the user must zoom in, as in Figure 3b, in order to read the expansion.


$A^{1+S+\cdots+S^{\ell-1}} = 1$
$A_x^{1+S+\cdots+S^{\ell-1}} = 1$
B_x is a rational function of x, which does not vanish for all x.
Node A06 in Proof of Theorem 90
Node A12 in Proof of Theorem 90

(a) The same expansion as in Figure 2b has been inserted, this time in embedded mode.

(b) Zooming in, the expansion becomes legible.

Figure 3: Expansion in embedded mode

Furthermore, whereas in Figure 2b, unified mode allowed PISE to draw edges linking nodes of the proof to nodes of the expansion, Figure 3b shows how embedded mode requires PISE to instead draw "ghost nodes" in the expansion, which refer to the corresponding nodes of the surrounding proof.

3.2.3. Comparison

Each expansion mode has its advantages and disadvantages.

Unified mode keeps all nodes simultaneously legible, and allows them to connect to each other across expansion boundaries; however, more space is required, and the visual distinction between expansions and their surroundings is subtle, being marked only by the expansion's outer boundary box.

Embedded mode visually emphasizes the distinction between an expansion and its surroundings, by scaling it down to make it look like "details." This arrangement lets us zoom in if we are interested in those details, or stay zoomed out and let the expansion simply appear as an (illegible) "reminder" that further details are available. Embedded mode packs a lot into a small space, but nodes at different levels are not simultaneously legible, and we cannot draw edges connecting nodes across different levels.

3.2.4. Questions

Is one expansion mode better? Is there a more natural one, which should be the default mode?

Is it true that in embedded mode edges *cannot* be drawn across the expansion boundary, or could there be a technique for drawing them, using ports [8]? More importantly, would drawing such connections be *useful* to the student, or just add visual noise?

What about disruption of the mental map? Can we expect embedded mode to be less disruptive? It seems reasonable, due to (a) fewer interconnecting edges between the nodes of the expansion and the nodes of the proof, and (b) a smaller overall amount of new space required in the original layout.

Should expansions perhaps not be inserted into existing layouts at all, but instead be brought up in pop-up windows? Or would this make it harder for students to relate the expansions to the original proof, especially in the case of multiple expansions viewed simultaneously?

3.3. Content Linking and Multiple Views

In certain kinds of software, various types of "content" may be displayed simultaneously, and *linked* to each other, so that clicking in one view automatically *navigates* another view, in order to show something to the user. Some familiar examples include:

- Google Maps: Clicking steps in the detailed driving directions navigates the map.
- Lessons at chess.com: Clicking steps in a lesson advances through moves on the game board.

- IDEs: Selecting a stack frame in a debugging panel navigates to the corresponding source code in a code panel.

In this kind of software, there can be two usability issues related to content linking, when *multiple views* of the navigated content are open:

1. **Telegraphing links:** How can users *be informed* of where navigation will take place? The problem may be familiar from debugging in IDEs: Supposing a source file is open in two panels, which of the two will auto-navigate, when a stack frame is clicked in a debugging panel?
2. **Choosing links:** Can users *have control* over where navigation will take place (or even turn it off completely)? For example, when debugging in many IDEs, the answer is that the *most recently active* panel will navigate. Therefore the user can control linking by clicking the desired panel to make it active, before navigating.

Content linking with multiple views is among the major usability questions in PISE, due to the multiple related content types that can be viewed there.

3.3.1. Content Types

PISE enables users to browse three different *content types*. The *chart panels* in which proof argument maps are displayed are the first type. Besides these are *notes panels* and *doc panels*, which display, respectively, narrative guides to proofs (we may refer to these as “notes”, “annotations”, or “discussions”), and archival documents containing proofs from the literature (“docs”).

3.3.2. Link Types

The three content types can be linked. For example, each node in a chart panel can be linked to the corresponding step in a doc panel. A discussion in a notes panel can have clickable links that navigate a chart panel and/or a doc panel to show the steps being discussed.

Of the six conceivable link types, PISE currently supports the five described in Table 1.

Outgoing links from notes panels ($\mathsf{N} \to \mathsf{C}$ and $\mathsf{N} \to \mathsf{D}$) are a simple matter of a discussion pointing to, highlighting, and navigating a chart or document.

Links back and forth between charts and docs ($\mathsf{C} \to \mathsf{D}$ and $\mathsf{D} \to \mathsf{C}$) are easy because there will be a one-to-one correspondence between steps in a proof diagram, and steps in the original prose form of the proof.

Finally, links pointing into notes panels ($\mathsf{D} \to \mathsf{N}$ and $\mathsf{C} \to \mathsf{N}$) are the trickiest case, because a discussion may refer to a given point within a chart or doc any number of times. Therefore, when clicking a node in a chart, say, we cannot simply navigate to “the” point in a linked notes panel where this node is mentioned; it may be mentioned in multiple places (or not at all).

In the case of outgoing links from doc panels, PISE solves this problem by bringing up a context menu when the user clicks any point within a document that has been referenced multiple times by currently open, linked panels (of either kind, N or C). The menu lists each of the linked reference points, and the user can click any of them to carry out the navigation (and hover triggers the same tab highlighting described in Section 3.3.3).

At present, the same solution has *not* been implemented for outgoing links from chart panels, and this is due to an assumption about usage patterns. The assumption is that, when a user clicks a point in a document, their only reason for doing so could be to discover linked content; therefore the linking context menu is appropriate.

On the other hand, it is believed that clicking a node in a chart is less often about attempting to discover a list of linked points in a discussion. The simplest and most common reason for clicking a node is simply as an aid in studying that step of the proof. When a node is clicked, the node and its incident edges are highlighted. Optionally, the neighboring nodes may be highlighted as well. In addition, the neighboring nodes become visible in the "neighborhood view" (an optional inset view in a

Table 1
Link types currently supported in PISE. The letters C, D, N stand for the content types of charts, docs, and notes, respectively.

Link Type	Description
$\mathsf{N} \to \mathsf{C}$	A discussion navigates a proof diagram.
$\mathsf{N} \to \mathsf{D}$	A discussion points to and highlights text in a document.
$\mathsf{C} \to \mathsf{D}$	Clicking a node in a proof navigates to and highlights the corresponding point in that proof as it appears in a document.
$\mathsf{D} \to \mathsf{C}$	The label for a node is generated from a document, and/or clicking that point in the document navigates to that node in a chart.
$\mathsf{D} \to \mathsf{N}$	Clicking words that have been highlighted by a discussion navigates to that point in the discussion (or loads a context menu in the case of multiple references).

chart panel). Clicking a node will also navigate a linked doc panel, due to the $\mathsf{C} \to \mathsf{D}$ links which *are* currently supported; however, it seems potentially too disruptive to ordinary usage patterns to also bring up a context menu to resolve multiple $\mathsf{C} \to \mathsf{N}$ links.

3.3.3. Telegraphing Links

In PISE, telegraphing links is simple: When the user hovers their mouse pointer over any active element in Panel A, which, if clicked, will cause navigation to take place in linked Panel B, then the *tab* at the top of Panel B glows slightly. This lets users know where navigation will happen, upon click.

3.3.4. Choosing Links

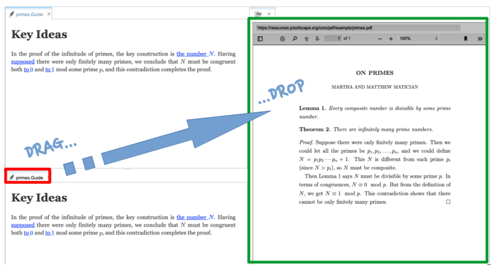
Each time a content panel is loaded, PISE starts by applying a heuristic, to establish links. This heuristic uses among other things the *most recently active panel* rule mentioned above. For example, if two copies of a document were open (in doc panels), and a notes panel was then loaded which referred to that document, the $\mathsf{N} \to \mathsf{D}$ link automatically established by PISE would be to the doc panel that had been more recently active.

No heuristic, however, can always choose what the user wants, in all situations. The user should therefore have the ability to control the linking of content panels. In PISE this is achieved via a drag-and-drop interaction. For example, in Figure 4a, there are two notes panels N_1 and N_2 open on the left (say N_1 above N_2), showing two copies of the same discussion. The discussion is about the proof that is open in the doc panel D on the right. PISE's heuristic automatically establishes *both* of the links $N_1 \to D$ and $N_2 \to D$, since there is only one copy of the document, so both discussions should navigate it. In the other direction, the heuristic might establish *only* the link $D \to N_1$, since (we may imagine) N_1 and D were present first, and then N_2 was opened as a second copy.

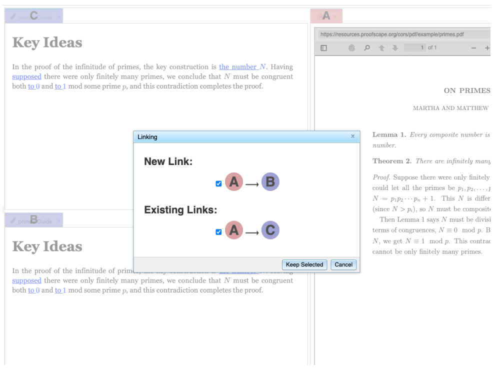
In this scenario, if the user wants D to navigate N_2 instead of N_1, they can begin by dragging the *tab* of the N_2 panel, and dropping it onto the *content area* of the D panel (Figure 4a). The suggested mnemonic here is, "TAb = TArget, CONTent = CONTroller." This brings up the *linking dialog* (Figure 4b), which shows both the new, proposed link, and the existing link(s). In the background, letters have been overlaid on all the panel tabs, so that the dialog can meaningfully describe the possible links. The user can select any subset of the proposed and existing link(s) to keep (including none, if they want to turn off all linking).

3.3.5. Questions

How might $\mathsf{C} \to \mathsf{N}$ links be supported? Nodes already do have a right-click context menu (for things like loading linked notes not yet open). Should the necessary menu for disambiguating multiple $\mathsf{C} \to \mathsf{N}$ links become a submenu thereof?



(a) Drag tab A onto content area B to make B navigate A.



(b) After drag and drop, linking dialog appears.

Figure 4: Drag-and-drop linking

Is tab glow for the telegraphing of links always helpful? Or can it become annoying, once the user is aware of the link? Should the glow appear only after a certain delay (a long hover indicating the user's uncertainty)?

As for manual link control, is the drag-and-drop interface intuitive? Can users remember which tab to drag onto which content area? (Is the mnemonic, "TAb = TArget, CONTent = CONTroller" helpful?)

3.4. Example Explorers

The notes panels in PISE (see Section 3.3.1) are pages of text enriched with various types of interactive "widgets." Among these widgets are the bits of clickable text that cause linked chart and doc panels to navigate.

Other widget types allow page authors to build *example explorers*: collections of *parameter widgets*, where students select numerical input values, and *display widgets*, which display the results of calculations using the SymPy [10] computer algebra system.[3]

An example explorer page can be *attached to* a node in a proof, so that when users right-click the node, they see an option to load the explorer in a new notes panel. The explorer can then help to illustrate the mathematics taking place at that step in the proof.

For example, the explorer page in Figure 5 is linked to the second node down from the top-right in Figure 1b. The explorer starts with some introductory text, then offers two parameter choosers, and finally shows a display, illustrating the algebra taking place on that node of the proof. Here, the user has loaded the explorer twice, in side-by-side panels, in order to compare the final displays after dialing in different parameters.

3.4.1. Questions

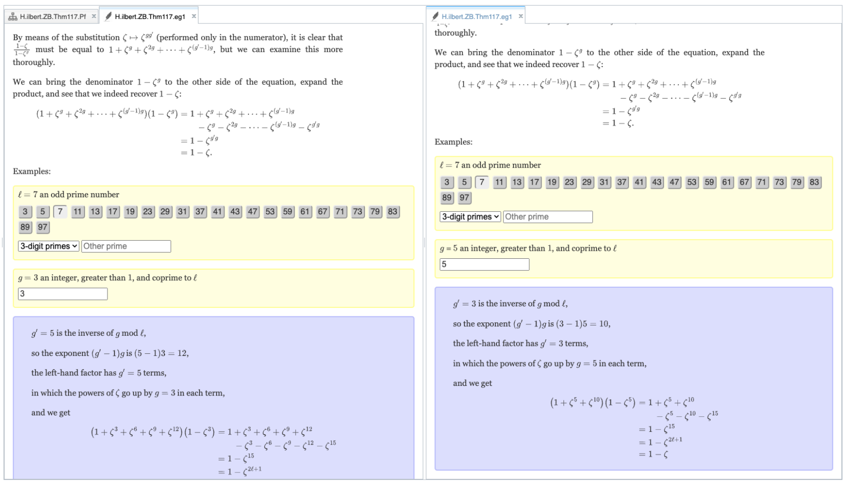
Currently, example explorers like the one in Figure 5 are designed by hand. A human author chooses parameters, and designs displays. This means that explorers are sometimes available (on those steps where authors have had time to prepare them), and sometimes not.

Could example explorers instead (or in addition) be generated automatically, based on a formal encoding of the mathematical objects in play at any given step in the proof, and thereby become something that students can count on as being always available?

3.5. Tracking Progress and Taking Notes

As a student works their way through a proof, PISE provides ways for them to keep track of their progress, and record notes.

[3]https://www.sympy.org



By means of the substitution $\zeta \mapsto \zeta^{gg'}$ (performed only in the numerator), it is clear that $\frac{1-\zeta}{1-\zeta^g}$ must be equal to $1+\zeta^g+\zeta^{2g}+\cdots+\zeta^{(g'-1)g}$, but we can examine this more thoroughly.

We can bring the denominator $1-\zeta^g$ to the other side of the equation, expand the product, and see that we indeed recover $1-\zeta$:

$$\begin{aligned}(1+\zeta^g+\zeta^{2g}+\cdots+\zeta^{(g'-1)g})(1-\zeta^g) &= 1+\zeta^g+\zeta^{2g}+\cdots+\zeta^{(g'-1)g} \\ &\quad -\zeta^g-\zeta^{2g}-\cdots-\zeta^{(g'-1)g}-\zeta^{g'g} \\ &= 1-\zeta^{g'g} \\ &= 1-\zeta.\end{aligned}$$

Examples:

$\ell = 7$ an odd prime number

$g = 3$ an integer, greater than 1, and coprime to ℓ

3

$g' = 5$ is the inverse of g mod ℓ,

so the exponent $(g'-1)g$ is $(5-1)3 = 12$,

the left-hand factor has $g' = 5$ terms,

in which the powers of ζ go up by $g = 3$ in each term,

and we get

$$\begin{aligned}(1+\zeta^3+\zeta^6+\zeta^9+\zeta^{12})(1-\zeta^3) &= 1+\zeta^3+\zeta^6+\zeta^9+\zeta^{12} \\ &\quad -\zeta^3-\zeta^6-\zeta^9-\zeta^{12}-\zeta^{15} \\ &= 1-\zeta^{15} \\ &= 1-\zeta^{2\ell+1}\end{aligned}$$

thoroughly.

We can bring the denominator $1-\zeta^g$ to the other side of the equation, expand the product, and see that we indeed recover $1-\zeta$:

$$\begin{aligned}(1+\zeta^g+\zeta^{2g}+\cdots+\zeta^{(g'-1)g})(1-\zeta^g) &= 1+\zeta^g+\zeta^{2g}+\cdots+\zeta^{(g'-1)g} \\ &\quad -\zeta^g-\zeta^{2g}-\cdots-\zeta^{(g'-1)g}-\zeta^{g'g} \\ &= 1-\zeta^{g'g} \\ &= 1-\zeta.\end{aligned}$$

Examples:

$\ell = 7$ an odd prime number

$g = 5$ an integer, greater than 1, and coprime to ℓ

5

$g' = 3$ is the inverse of g mod ℓ,

so the exponent $(g'-1)g$ is $(3-1)5 = 10$,

the left-hand factor has $g' = 3$ terms,

in which the powers of ζ go up by $g = 5$ in each term,

and we get

$$\begin{aligned}(1+\zeta^5+\zeta^{10})(1-\zeta^5) &= 1+\zeta^5+\zeta^{10} \\ &\quad -\zeta^5-\zeta^{10}-\zeta^{15} \\ &= 1-\zeta^{15} \\ &= 1-\zeta^{2\ell+1} \\ &= 1-\zeta\end{aligned}$$

Figure 5: An example explorer in side-by-side panels.

3.5.1. Goal Boxes

Each node in a proof diagram has a *goal box* in its upper-right corner (visible for example in Figure 2a). The student can left-click this to place a check mark. In the case of *intro* nodes, this should mean, "I have read and understood the definition(s)," while for *assertion* nodes it should mean, "I have understood how this step follows."

Meanwhile, the student can right-click the goal box to open a dialog where they can record free-form notes, perhaps noting a question to return to later, or an explanation as to how the step follows.

3.5.2. Study Pages

The accumulated checkmarks and free-form notes recorded on goal boxes for a given proof can be assembled and reviewed at any time. The user can right-click the proof in its chart panel, and request the proof's *study page*.

The study page is an automatically generated notes panel which lists all the nodes in the proof by name (Figure 6a), and for each one shows both the state of the goal box (checked or unchecked) and displays any notes the student may have recorded. For example, Figure 6b shows a node the student has checked, and on which they have recorded a clarification as to how that step follows.

3.5.3. Questions

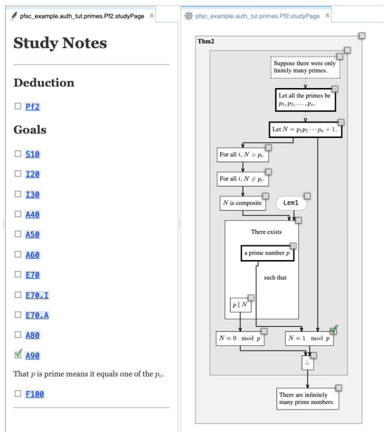
Are the node names in the study page useful? Or do users need to see graphics, reproducing an image of each node within the page, so that they can have an immediate sense of which node is being discussed? Would it be enough if instead hovering over a node name immediately caused the corresponding node to glow in the linked chart panel?

Should "bubbling up" of checkmarks be automated? At present, even if the user checks all the nodes in a proof, the goal box for the proof itself does not then become automatically checked. Should it? Or is it best to leave the student to do their own "recap" first?

"Gamification" has been a popular theme in recent didactics research [11]. Can or should the study process in PISE be further gamified, by building in some type of special rewards for students as they accumulate checkmarks, and/or challenges before checkmarks will be accepted?

4. Conclusion

We have reviewed various techniques for studying the steps of mathematical proofs, and raised questions about each. It is hoped that these questions might provide a basis for future usability studies, and


pfsc_example.auth_tut.primes.Pf2.studyPage

Study Notes

Deduction

☐ Pf2

Goals

☐ S10

☐ I20

☐ I30

☐ A40

☐ A50

☐ A60

☐ E70

☐ E70.I

☐ E70.A

☐ A80

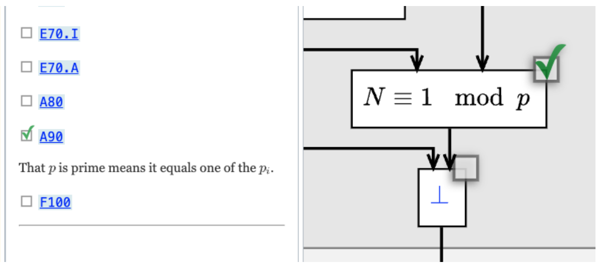
☑ A90

That p is prime means it equals one of the p_i.

☐ F100

pfsc_example.auth_tut.primes.Pf2.studyPage
Thm2
Suppose there were only finitely many primes.
Let all the primes be $p_1, p_2, \ldots, p_n$.
Let $N = p_1 p_2 \cdots p_n + 1$.
For all i, $N > p_i$.
For all i, $N \neq p_i$.
N is composite
Lem1
There exists
a prime number p
such that
$p \mid N$
$N \equiv 0 \mod p$
$N \equiv 1 \mod p$
$\perp$
There are infinitely many prime numbers.

(a) The auto-generated study page (left) lists all nodes of the proof by name, and shows their checkboxes and notes.


☐ E70.I

☐ E70.A

☐ A80

☑ A90

That p is prime means it equals one of the p_i.

☐ F100

(b) Clicking a node name in the study page navigates the chart panel to show that node. The goal boxes are linked, so that they show the same checkmarks, and can be clicked in either panel. The user has recorded a clarification on node A90.

Figure 6: Auto-generated study page for a proof

eLearning studies in the field of mathematics, as well as encouraging community engagement in further development of the open-source PISE software[4] discussed here.

References

[1] T. Van Gelder, Argument mapping with Reason!Able, The American philosophical association newsletter on philosophy and computers 2 (2002) 85–90.

[2] S. Kieffer, PISE–Proofscape Integrated Study Environment (CICM'22 system entry), in: Work-in-progress papers presented at the 15th Conference on Intelligent Computer Mathematics (CICM 2022) Informal Proceedings, 2022, p. 28.

[3] S. Kieffer, Argument mapping for mathematics in Proofscape, in: Diagrammatic Representation and Inference: 8th International Conference, Diagrams 2014, Melbourne, VIC, Australia, July 28–August 1, 2014. Proceedings 8, Springer, 2014, pp. 57–63.

[4] M. O. Ward, Xmdvtool: Integrating multiple methods for visualizing multivariate data, in: Proceedings Visualization'94, IEEE, 1994, pp. 326–333.

[5] J. Wolfe, Annotation technologies: A software and research review, Computers and Composition 19 (2002) 471–497.

[6] K. Misue, P. Eades, W. Lai, K. Sugiyama, Layout adjustment and the mental map, Journal of Visual Languages & Computing 6 (1995) 183–210.

[7] G. D. Battista, P. Eades, R. Tamassia, I. G. Tollis, Graph drawing: algorithms for the visualization of graphs, Prentice Hall PTR, 1998.

[8] C. D. Schulze, M. Spönemann, R. Von Hanxleden, Drawing layered graphs with port constraints, Journal of Visual Languages & Computing 25 (2014) 89–106.

[9] K. Sugiyama, S. Tagawa, M. Toda, Methods for visual understanding of hierarchical system structures, IEEE Transactions on Systems, Man, and Cybernetics 11 (1981) 109–125.

[10] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, et al., SymPy: symbolic computing in Python, PeerJ Computer Science 3 (2017) e103.

[11] L. da Rocha Seixas, A. S. Gomes, I. J. de Melo Filho, Effectiveness of gamification in the engagement of students, Computers in Human Behavior 58 (2016) 48–63.

[4] https://github.com/proofscape/pise