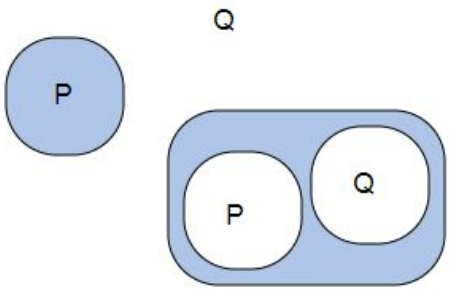

Figure 1. Graphs representing the statements “not P”, “Q”, and “P or Q”.

Quiche Final Project Report

Gavin Petilli and Peter Wood

1. Motivation and Audience

Created by logician Charles Sanders Peirce in the late 19th and early 20th century, existential graphs are a method of representing propositional logic statements graphically, as an alternative to the more traditional method of representing statements as strings of symbols [3]. Though existential graphs are often considered a niche visualization of formal logic, they provide an elegant and aesthetically pleasing alternative to traditional logical formulae. We believe that their relative anonymity is directly due to the lack of good software. One of our group members, Peter Wood, previously created a similar visualization software for existential graphs and, while it was technically usable and almost complete, the user interface was clunky and unintuitive. (It can be viewed currently at pdwood.github.io/Chrysaor.) Since we have yet to find an intuitive and sufficiently usable software to visualize existential graphs, we created Quiche, a simple and intuitive tool for existential graphs. Our target audience is students and enthusiasts of formal logic.

2. Background and related work

The representational formalism of existential graphs is extremely simple: the juxtaposition of two statements asserts their conjunction, and the drawing of a circle around one or more elements (known as a “cut”) represents the negation of the enclosed subgraph. Thus two or more statements enclosed in the same cut represents that at least one of them is false—that is to say, a generalized NAND, an operator which has been proven to be expressively complete by itself [4]. For the implementation of the existential graphs interface, we drew inspiration from many code samples on bl.ocks.org, a website for sharing examples of d3.js code. [1] [2]

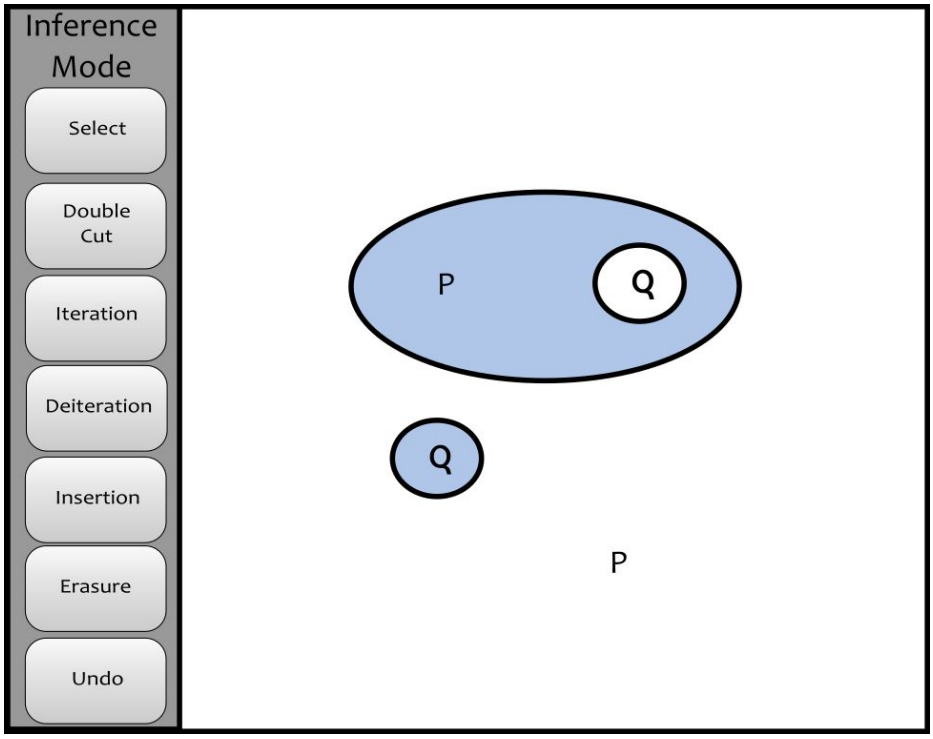

Figure 2: Mock-up screenshot of Inference Mode.

3. Visualization design evolution

The visualization design remained largely constant over the duration of the project. Since existential graphs are themselves a well-defined visualization of statements in propositional logic, there was little room to change the form of the visualization, other than the decision to use rounded rectangles to represent cuts rather than ellipses (as shown in figure 2), which was changed for ease of implementation. However, the specifics of interactivity did change. Chrysaor, Peter's previous attempt at an interface for existential graphs, had a confusing set of key bindings for most actions, so that everything could be done using a few keys and a mouse without having a toolbar. For example, there was one inference rule that required the user to hold shift while dragging an object with the right-mouse button. We decided that it would be more user-friendly to have tools that could be selected from a toolbox. Early in the design, the mechanism for inserting cuts was going to be to physically draw a circle with the mouse, but this was deemed impractical, so instead clicking a point on the graph creates a small cut centered on the mouse cursor. At one point in the design of the project, it was suggested that dragging the edges of cuts could resize them; however, this feature was removed for reasons of complexity.

The overall design of the menu has not changed over the course of the project. The menu has always been a simple side menu, common on many websites, that can be collapsed by clicking the X in the corner of the menu, and reopened by clicking the "hamburger" symbol in the top-left corner of the screen. However, many small changes--partially based on user feedback--have been made. The colors of the buttons, which were originally arbitrary, were selected using colorbrewer2.org, with the red color being allocated to the Delete button, as was recommended by users in our study. Another feature added based on user recommendation was the the

change in button color when a mode is selected. This change greatly helps users remember what mode they are currently in. Additionally, various small changes were made to the layout to smooth transitions between modes and smooth edges on the menu to make it reflect the overall style of the program.

4. Feedback

Our feedback for this project was largely from people outside our target audience, who had some passing familiarity with formal logic but not a thorough knowledge in it. This in itself made us realize the necessity of including with the software an explanation of existential graphs and how they can be useful for visualizing formal logic. And despite this, they nevertheless had suggestions for changes to the interactivity of the project. For example, many people expressed a desire for the menu to be smaller or less obtrusive, and one respondent suggested replacing the text of the menu buttons with icons, a feature that may be added in a future version, with the text from the buttons still contained in tooltips. However, some users' suggestions, such as having the menu auto-hide when the mouse is not near the edge of the screen or using force-directed graphs to lay out the elements of the graph, were deemed too obtrusive or distracting.

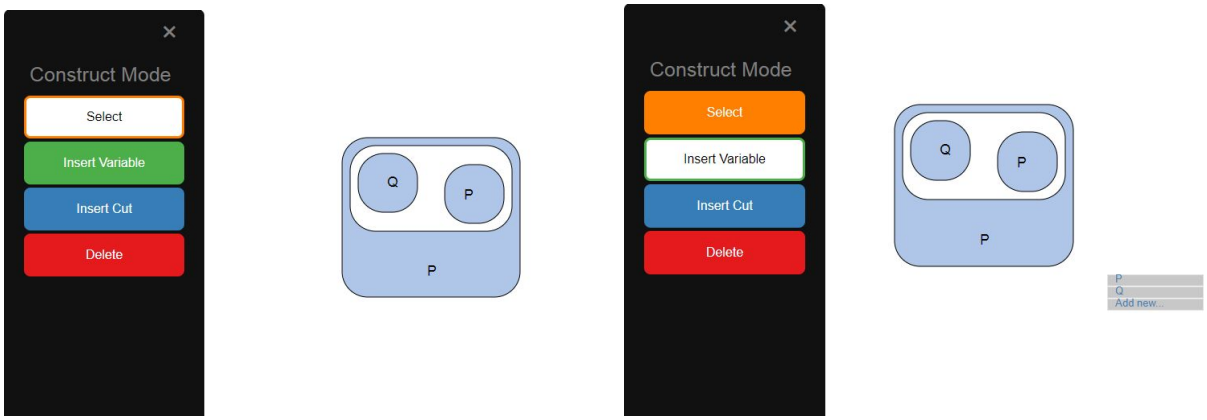

Figure 3: Examples of construct mode. In the left image, the Select tool is active. In the right image, the Insert Variable tool is selected and the variable selection menu has been opened.

5. Core features

Quiche consists of two modes: Construct Mode, in which graphs can be manipulated in arbitrary ways, and Inference Mode, in which graphs can be manipulated only according to valid rules of inference. Due to time constraints, Inference Mode is largely unimplemented. Clicking the name of the mode in the menu will change the mode.

In Construct Mode, the user can interact with the graph using four tools, shown on buttons in the menu.

- The Select tool (shown in figure 3, left) allows the user to move any variable or cut in the graph by clicking and dragging. Moving a cut will move all of its children

with it. Moving a cut will cause any parent cuts to expand or contract as is necessary to accommodate the movement.

- The Insert Variable tool (shown in figure 3, right) allows the user to add variables to the graph. Clicking anywhere on the graph prompts the user to select a variable to add using a context menu. Variables already existing on the graph are listed for convenience, however, there is also an option to add a new variable.
- The Insert Cut tool allows the user to insert cuts into the graph. Clicking anywhere on the graph will insert an empty cut as a child of the highest-level parent clicked on. The parent node is expanded as necessary to accommodate the new cut.
- The Delete tool allows the user to remove any cuts or variables. Clicking on a cut or variable will remove it and any children it has from the graph.

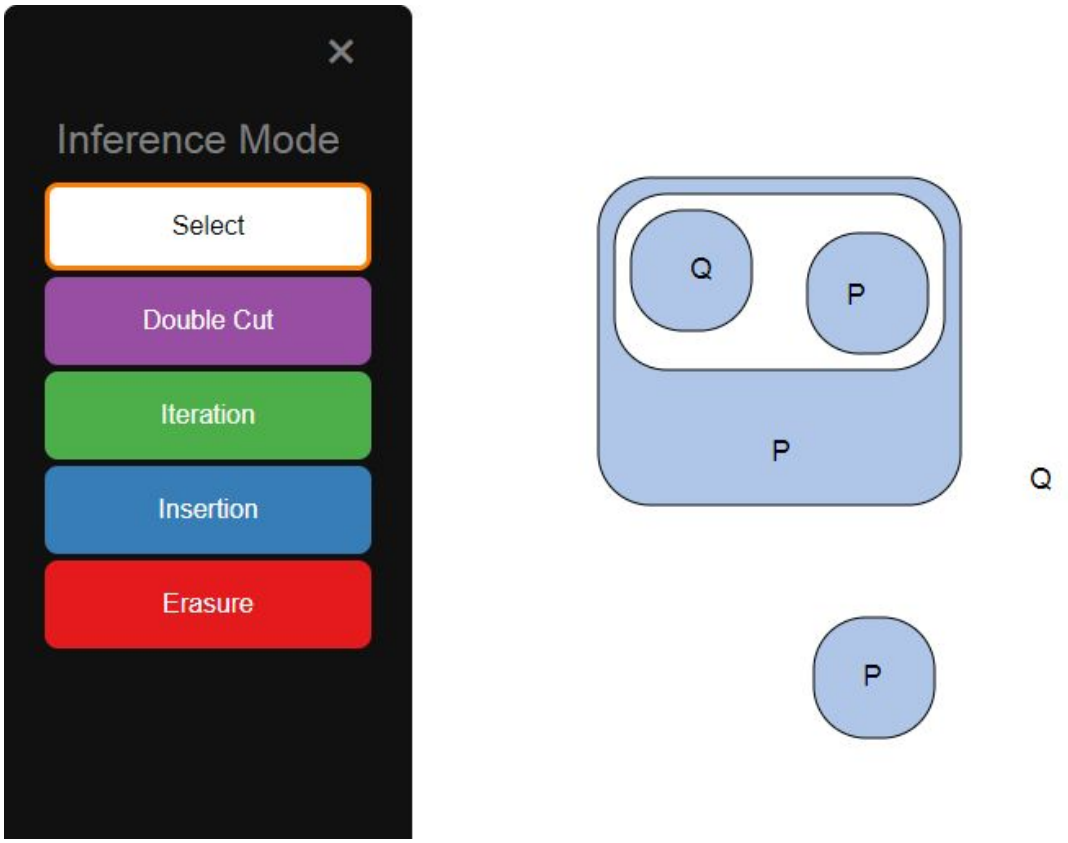

Figure 4: The inference mode menu, with the Select tool currently active.

In Inference Mode, shown in figure 4, the user can still move the graph at will, but can only change the structure of the graph using the inference rules defined in John F. Sowa's paper "Peirce's Tutorial on Existential Graphs". There are four rules of inference: Insertion, Erasure, Double Cut, and Iteration/Reiteration. These inference rules have been grouped into five tools that, along with the Select tool, are the six tools shown on the menu in this mode.

- The Select tool functions equivalently to the Select tool in Construct Mode.

- The Double Cut tool allows users to add or remove double cuts: that is, cuts that contain exactly one cut and nothing else. Since this structure represents a double negation, it can be added and removed from the graph without changing its meaning. Clicking between two such cuts will remove both of them, and [it's like a rectangular selection tool idk how to explain it]
- The Iteration tool allows users to copy portions of the graph into deeper levels (that is, inside more cuts). The user does this by clicking a cut or atomic variable to select it, which then highlights in green all areas it can be iterated into. Clicking one of these areas will then create a copy of the selection.
- The Deiteration tool is the opposite of the Iteration tool: as before, the user clicks a cut or atomic variable to select it. All copies of the selection at deeper levels will be highlighted in red, and the user can click one of them to remove it from the graph.
- The Insertion tool allows the user to add arbitrary elements to the graph on odd (blue) levels (that is, levels enclosed by an odd number of cuts, with the lowest level being zero). To use this, the user clicks anywhere on an odd level, and a popup window will appear, containing a new instance of Quiche in Construct Mode. The user can then construct an arbitrary graph, which will then be inserted into the parent graph at the location they clicked.
- The erasure tool allows the user to remove arbitrary elements from the graph on even (white) levels (that is, levels enclosed by an even number of cuts, including zero). To use this, the user clicks on a cut or atomic variable, and it is removed from the graph; in the case of a cut, all contents of the cut are removed as well.

6. Distribution of labor

We divided the project labor between the implementation of the existential graph itself and the implementation of menus and design of the overall layout. Peter used D3.js to create a SVG representation of an existential graph, implementing additional functionality to enable interaction with the graph. Gavin created the overall HTML and CSS layout, created the side menu and enabled its interaction with the graph.

Works Cited

1. Bostock, Mike. “Circle Dragging I.” *Bl.ocks.* http://bl.ocks.org/mbostock/c206c20294258c18832ff80d8fd395c3. 4 July 2016
2. Jakosz. “Context menu for SVG elements.” *Bl.ocks.* http://bl.ocks.org/jakosz/ce1e63d5149f64ac7ee9. 12 April 2018.
3. Sowa, John F. Peirce’s tutorial on existential graphs, *Semiotica* **186:1-4**, Special issue on diagrammatic reasoning and Peircean logic representations, 2011, pp. 345-394. http://www.jfsowa.com/pubs/egtut.pdf.
4. van Heuveln, Bram. Existential Graphs: Computability and Logic. 13 January 2015. http://www.cogsci.rpi.edu/public_html/heuveb/teaching/Logic/CompLogic/Web/Presentations/EG.pdf.