Graphboard: A User Interface for Solving Graphical Problems by Drawing

François Schwarzentruber
ENS de Lyon, CNRS, University Claude Bernard Lyon 1, Inria, LIP, UMR 5668, 69342, Lyon Cedex 07, France

Vaishnavi Sundararajan
Department of Computer Science and Engineering, IIT Delhi, Hauz Khas, New Delhi, India

Abstract

We present a user interface called Graphboard that enables to draw graphs. The idea to solve exercises in logic whose answers are graphs or automata. The graphs are hand-drawn and the system recognizes the vertices, edges and labels. The formal graph corresponding to the picture is then evaluated against the question.

2012 ACM Subject Classification CCS → Applied computing Education → Interactive learning environments

Keywords and phrases Graph, Hand-written system, Exercices, Logic

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Vaishnavi Sundararajan*: Funded by the Scientific High Level Visiting Fellowships (SSHN) by the French Institute in India (IFI), Embassy of France in India.

1 Introduction

Computer Science and affiliated departments in universities across the world have seen a significant rise in student numbers over the last three decades, with each successive software-related boom contributing a sharper increase in students opting for such subjects. However, the number of teaching staff has not increased at the same rate, which makes for ever-worsening student-teacher ratios. This motivates the need for automated teaching support tools, which can help students learn concepts at their own convenience, and at their own pace.

In this work, we focus on tools which deal with problems of a graphical nature, in the domains of modal/FO/MSO logic and graph/automata theory. Many such tools already exist (e.g. [16], [15]). However, student adoption/repeated use of these tools is rare, if not entirely non-existent. We hypothesize that this is because of the following two reasons:

- These tools are desktop-first, and do not necessarily work well on mobile devices. Even if they do, the input required is either textual, or drag and drop – operations which are not natural/easy on a touchscreen mobile device – with complicated user interfaces.
- These tools do not provide personalized automated hints depending on the precise mistakes the student makes.

In this work, we choose to address these problems. We present Graphboard, a tool where users (students) are presented with problems to which the solutions are graphical. Users can draw solution diagrams completely freehand. The user interface is as minimal as possible, with the default mode being "draw", and various germane parts of the drawing being recognized automatically (i.e. without the user having to explicitly choose – for example – a "node drawer", then an "edge drawer" etc, out of a crowded menu). The user can erase by scribbling. In addition, in case of an incorrect solution, Graphboard provides feedback that

is specific to the error in the candidate solution, and leads the user to the correct solution via such helpful hints.

In the current software milieu, it is common to default to the assumption that one could use machine learning to do such image recognition. Training a new model for such drawing is time-consuming, energy-inefficient, and even the end result is not feasible to be run on {space and compute}-constrained mobile devices, and does not provide any guarantees. It is also a widely-held belief that large language models (LLMs) do relatively well at converting hand-drawn images into some manner of machine-readable format. However, interestingly, not all LLMs are equally good at this, and even the same LLM does not perform equally well with all sorts of input graphs. For example, Table 1 shows a breakdown of the performance of some LLMs (ChatGPT (GPT-4 model) [7], Perplexity [8], and Claude 3.7 Sonnet [2]) on generating TikZ code for a simple example graph drawn using Tableaunoir. The graph has four nodes laid out in a directed path from the first node through to the fourth. In addition, the first node has a self loop, and the second node has an edge back to the first, as well as an edge to the fourth node. This graph is fed in to the LLM both with and without labels on the nodes, in two separate experiments. The prompt given along with the image remains the same, namely "Convert this graph into TikZ code". Table 1 illustrates the results, with errors in the code highlighted in red. The errors encountered are semantic in nature, with most instances causing an issue due to the edge from the second node to the fourth being "misinterpreted".

Graphboard is designed to be an explainable tool without errors in image recognition, which uses less resources, and can run on "small" mobile devices.

Our tool is available here:
`https://perso.ens-lyon.fr/francois.schwarzentruber/teaching/graphboard/`

2 Related work

2.1 Graph editors

There are many graph editors, including Quiver [9], GraphViz [6] etc. These allow one to draw graphs, but do not have any pedagogical element to their operation. Quiver requires the user to interact with a grid-based canvas by clicking and dragging to create a graph, while GraphViz requires the input to be provided using the DOT language. There are also websites which allow users to create graphs and then visualize the result of various preprogrammed algorithms on these graphs, with a view to understanding how these procedures work. One such website is Graph Online [4], where the user can create a graph using clicks and drags (or by inputing an adjacency list, and assign weights/labels to edges, and then run one of many preset algorithms to see the result. The graph cannot be input freehand, nor are the intermediate steps of the various algorithms made visible to the user. Another similar tool is VisuAlgo [11]. The user can input a graph as an edge/adjacency list or as an adjacency matrix, and then generate the graph and see various properties about it (whether it is a tree, or a DAG, or complete, or bipartite, etc). There is an e-lecture mode which teaches many concepts about graph theory as well. The website also includes a short quiz, which requires drawing graphs via a click-and-drag-based interface, and the output of the quiz is tailored towards helping the user (by explaining why an incorrect answer did not work). However, the questions for the quiz appear to be randomly generated from a set pool, and the feedback about correctness appears at the very end of the quiz; a user does not get a second chance to fix their solution based on the "hint" provided as part of the results.

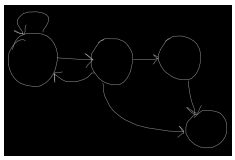
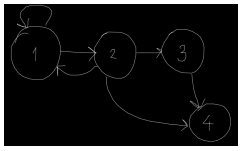
	Unlabelled Graph	Labelled Graph
Drawing		
ChatGPT (GPT-4)	latex Copy Edit \documentclass{article} \usepackage{tikz} \usetikzlibrary{arrows.meta} \begin{document} \begin{tikzpicture}[node distance=2.5cm, thick, ->, >=Stealth] % Nodes \node (A) [circle, draw] {}; \node (B) [circle, draw, right of=A] {}; \node (C) [circle, draw, right of=B] {}; \node (D) [circle, draw, below of=C] {}; % Edges \draw (A) edge[bend left] (B); \draw (B) edge (C); \draw (C) edge (D); \draw (D) edge[bend left] (A); \draw (A) edge[loop above] (A); \draw (D) edge[bend left] (B); \draw (B) edge[bend left] (A); \end{tikzpicture} \end{document}	latex Copy Edit \documentclass{article} \usepackage{tikz} \begin{document} \begin{tikzpicture}[node distance=2cm, ->, >=stealth, thick] % Nodes \node[circle, draw] (n1) {1}; \node[circle, draw, right of=n1] (n2) {2}; \node[circle, draw, right of=n2] (n3) {3}; \node[circle, draw, below of=n3] (n4) {4}; % Edges \path (n1) edge[loop above] (n1); \path (n1) edge (n2); \path (n2) edge[loop left] (n2); \path (n2) edge (n3); \path (n2) edge[bend left] (n4); \path (n3) edge (n4); \path (n4) edge[loop right] (n4); \path (n4) edge[bend left] (n1); \end{tikzpicture} \end{document}
Perplexity	tex \documentclass[tikz,border=5mm]{standalone} \usetikzlibrary{arrows.meta, positioning} \begin{document} \begin{tikzpicture}[every node/.style={circle, draw, minimum size=1cm}, arrow/.style={-Stealth, shorten >= 2pt, shorten <= 2pt},] \node (q1) at (0,0) {}; \node (q2) [right=of q1] {}; \node (q3) [right=of q2] {}; \node (q4) [below right=of q3] {}; \draw[arrow] (q1) to [loop above] (q1); \draw[arrow] (q1) -- (q2); \draw[arrow] (q2) -- (q3); \draw[arrow] (q3) -- (q4); \draw[arrow] (q2) to[bend right] (q1); \draw[arrow] (q2) to[bend right] (q4); \end{tikzpicture} \end{document}	tex \documentclass[tikz,border=2mm]{standalone} \usetikzlibrary{arrows} \begin{document} \begin{tikzpicture}[->, >=stealth', node distance=2cm] % Define the nodes \node[circle, draw, thick] (1) {1}; \node[circle, draw, thick, right of=1] (2) {2}; \node[circle, draw, thick, right of=2] (3) {3}; \node[circle, draw, thick, below of=3] (4) {4}; % Draw the edges \draw[thick] (1) edge[loop above] (1); % Self-loop on 1 \draw[thick] (1) -- (2); % 1 -> 2 \draw[thick] (2) edge[bend left] (1); % 2 -> 1 \draw[thick] (2) -- (3); % 2 -> 3 \draw[thick] (3) -- (4); % 3 -> 4 \draw[thick] (1) edge[bend right] (4); % 1 -> 4 \end{tikzpicture} \end{document}
Claude 3.7 Sonnet	TikZ Code for Graph \documentclass{article} \usepackage{tikz} \usetikzlibrary{arrows.meta} \begin{document} \begin{tikzpicture}[node distance=3cm, every node/.style={draw, circle, minimum size=1.5cm}, arrow/.style={->, >=Stealth, thick}] % Define nodes \node (A) at (0,0) {}; \node (B) at (3,0) {}; \node (C) at (6,0) {}; \node (D) at (6,-3) {}; % Define edges \draw[arrow] (A) -- (B); \draw[arrow] (B) -- (C); \draw[arrow] (A) to[bend right=30] (B); \draw[arrow] (C) to[bend right=45] (D); \draw[arrow] (A) to[bend right=60] (D); \end{tikzpicture} \end{document}	TikZ Code for Graph \documentclass{article} \usepackage{tikz} \usetikzlibrary{arrows.meta} \begin{document} \begin{tikzpicture} % Define the nodes \node[circle, draw, minimum size=1cm] (1) at (0,0) {1}; \node[circle, draw, minimum size=1cm] (2) at (3,0) {2}; \node[circle, draw, minimum size=1cm] (3) at (6,0) {3}; \node[circle, draw, minimum size=1cm] (4) at (5,-3) {4}; % Define the edges with arrows \draw[->, >=Stealth] (1) -- (2); \draw[->, >=Stealth] (2) -- (3); \draw[->, >=Stealth] (2) -- (4); \draw[->, >=Stealth] (3) -- (4); \draw[->, >=Stealth] (1) to[bend right=40] (2); % Optional: to match the black background in the original image % Uncomment the next line if you want a black background with white eleme % \begin{scope}[on background layer] \fill[black] (-1,-4) rectangle (7,1); \end{tikzpicture} \end{document}

Table 1 Performance of various LLMs on generating TikZ code for an example graph.

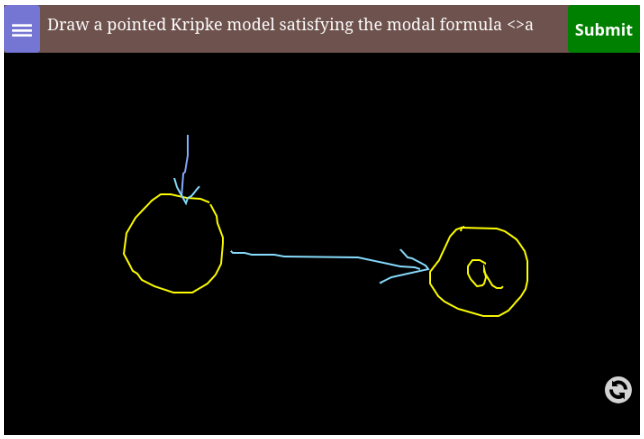


Figure 1 Graphical user interface of our proposed tool Graphboard: a simple question and a big black surface on which the student draws its answer. The top-left button is to switch to another exercise. The top-right button is to submit the solution. the bottom-right button is erase the whole board.

There are many graphical user interfaces for teaching graph theory [5], geometry [3], model theory [10], automata theory [1]. However, some of these require a computer to be installed on (and do not support a mobile device), some only permit limited interaction by the student, and all of them rely on a drag-and-drop system which is not natural on smartphones, and require some learning curve just to use the tool.

On the contrary, we want to develop a tool for learning logic and graph theory where the input of the user is free. And the pedagogical tool should be able to recognize what the user has drawn.

2.2 Handwriting systems

There are techniques to improve visual feedback in drawing software for teaching [12]. But they do not tackle teaching logic. Our recognition algorithm is close to the use of constraint multiset grammars like in [17] (all is based on the seminal paper [18] in which a rule rewrites ("recognizes") a multiset β of elements that satisfy some constraint into another multiset α).

There are studies to measure the effectiveness of using tablets in a learning context. For instance, [13] is about learning how to write (which is a different task that our tool) while [14] is about learning geometry which is somehow closer. Interestingly, the last paper is about a tool that provides feedback where our tool does not (yet!).

3 Our tool

3.1 Graphical user interface

Figure 1 shows a screenshot of our tool in action. The user (typically a student) has to solve a problem. Here: "Draw a pointed Kripke model satisfying the modal formula $\diamond a$. The user draws her/his solution on the virtual blackboard. The tool automatically recognizes the vertices, arrows pointing initial worlds (here only one), arrows that are edges (here only one), letters that saying which propositions are true in a given world. The system converts the image as a symbolic graph (here a pointed Kripke model), and check that it answers the question (here that the pointed graph satisfies formula $\diamond a$).

3.2 Technicalities

In our case, the image is drawn by the user on the tablet/smartphone/computer. So the image is not a bitmap, but a finite set of polylines. Letters are recognized via kNN (k-nearest neighbors method) by comparing polylines with the polylines in the dataset. Then we rely on the idea of constraint multiset grammars [18] with the following (informally presented) rules:

a polyline that represents a letter a	$\rightarrow$	a letter a
a polyline which is almost a circle	$\rightarrow$	a vertex
a polyline surrounded by an arrow head	$\rightarrow$	an arrow
an arrow pointing to a vertex but without an incoming vertex	$\rightarrow$	a pointed vertex
an arrow from vertex u to vertex v	$\rightarrow$	a directed edge from u to v
a polyline which is not an arrow from vertex u to vertex v	$\rightarrow$	an undirected edge from u to v
a letter a inside a vertex u	$\rightarrow$	a labels vertex u
a letter a outside a vertex, near an edge e, e being the nearest	$\rightarrow$	a labels edge e

We are aware that these rules are not perfect and the approach seems naive. They is room left for improvements. But the system seems to work.

4 Description of the demo

The demonstration is about the tool at the url `https://perso.ens-lyon.fr/francois.schwarzentruber/teaching/graphboard/`. We will show some exercises, and solve them in front of the audience. Then we will invite the audience to play with the tool and answer the exercises on their own.

Here are the type of exercises we propose for now:

- Draw a pointed Kripke model satisfying a modal formula, e.g. $\Diamond p$
- Draw a graph satisfying a FO formula, e.g. $\forall x, \exists y E(x, y)$
- Draw an deterministic/non-deterministic automaton for a given regular expression/language. Example: draw a deterministic finite-state automaton with two states for recognizing the language over $\{a, b\}$ with an even number of 'a's.

5 Structuring research studies

Table 2 summarizes some research questions and a proposed methodology to answer them. Many of our research questions concern the relevance of our tool and the efficacy of learning by handwriting solutions with tailored hints along the way.

6 Future work

6.1 Take a photo!

An additional facility we might want in our tool is to process photos of graphs drawn on the board. For the moment, this functionality is still work in progress. Figure 2 shows how the process works. Starting with a photo, we transform it into a skeleton which is a binary bitmap. Then we apply Zhang-Suen algorithm [19] to get a thin skeleton (line of 1-pixel thickness). After that we extract the graph of polylines via a depth-first search performed on

Research questions	Participants	Sampling and instruments	Analysis
Does the use of a free graph editor lead to fewer "drop outs" by learners (vs another type of interface)?	CS/Math Bachelor students	We ask students to use our tool vs a tool with another interface	Compare the time spent on the tools
Does drawing solutions by hand and personalized hints lead to more "perseverance"? (Do people persist even when they get answers wrong?)	CS/Math Bachelor students	We ask students to use our tool vs a tool with another interface	Compare the time spent by students correcting wrong answers vs exiting the tool
Does the tool feel more accessible to students with motor disabilities?	CS/Math Bachelor students	We ask students to use our tool vs a tool with another interface.	Compare the time spent by students + questionnaire
Is it better to take a photo from a real board vs drawing directly on the tablet?	CS/Math Bachelor students	We ask students to take a photo vs use the tool directly.	Compare the time spent by students + questionnaire

Table 2 Structuring research studies.

the grid of pixels. We get a list of polylines that we can then use for recognizing the drawn graph.

6.2 Additional exercises

We can also add more problems of a graph theoretic nature – tree decomposition, walks, max-flow-min-cuts etc. Some of these might also be helped by adding some visual animations to highlight the solution or hints towards it.

6.3 Incorporating ML and GenAI techniques

We could use techniques from machine learning to improve the image detection algorithms, especially if we can store data for each individual user (whether in the form of cookies or by making them log in and logging data). In addition, we can use generative AI techniques to generate hints that are more relevant and useful for the user, based on their past history and preferences.

Note that our method of relying on a grammar is very strict compared to general machine learning techniques. But one advantage is that our model is explainable. So we could also use such symbolic information about the graphical input to help the student to improve her answer (e.g. 'please draw your arrow head a bit better'). Also, in the long term, it might be possible learn (or fine tune) the grammar rules from personalized interaction data and feedback collection.

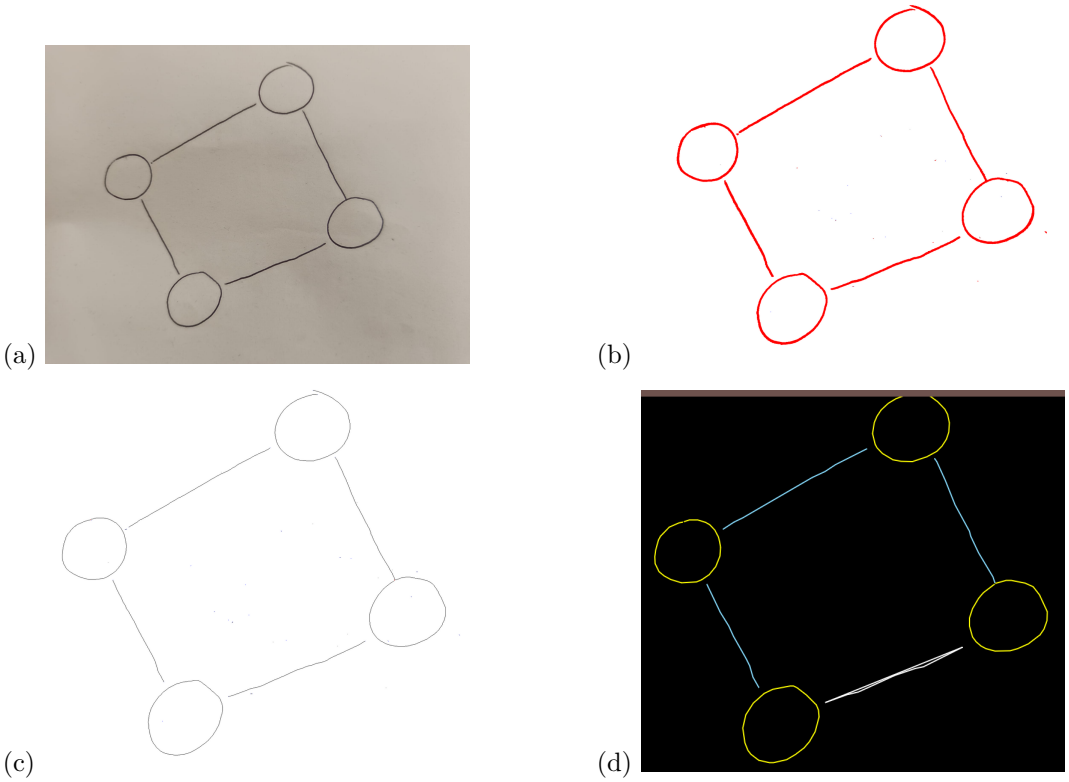


Figure 2 Intermediate results of the algorithm for recognizing a graph from a photo: (a) input photo. (b) skeleton. (c) Thin skeleton. (d) Polylines and recognition.

6.4 Others

Other questions of interest we might explore are the following:

- How do we provide automated feedback tailored to the expertise level of the user?
- Can we perform automated visual feedback? How?
- How do we provide an interface for "visual discussion"? Should we go from chatbot to graphbot (discuss by drawing)?

References

1. Automata tutor. `https://automata-tutor.live-lab.fi.muni.cz`. Last accessed: April 24, 2026.
2. Claude 3.7 Sonnet. `https://claude.ai`. Last accessed: March 3, 2025.
3. Geogebra. `https://www.geogebra.org`. Last accessed: April 24, 2026.
4. Graph online. `https://graphonline.top`. Last accessed: April 24, 2026.
5. Graphtea. `https://graphtea.github.io`. Last accessed: April 24, 2026.
6. Graphviz. `https://graphviz.org`. Last accessed: April 24, 2026.
7. OpenAI ChatGPT. `https://chatgpt.com`. Last accessed: March 3, 2025.
8. Perplexity: Pro Version. `https://www.perplexity.ai`. Last accessed: March 3, 2025.
9. Quiver. `https://q.uiver.app`. Last accessed: April 24, 2026.
10. Tarski's world. `https://www.gradegrinder.net/Products/tw-index.html`. Last accessed: April 24, 2026.
11. Visualgo. `https://visualgo.net/en/graphds`. Last accessed: April 24, 2026.

181 **12** Islam Barchouch, Nathalie Girard, Éric Anquetil, Laura Leconte, and Eric Jamet. Improving
182 feedback generation in a drawing-based ITS. In Sabine Graf and Angelos I. Markos, editors,
183 *Generative Systems and Intelligent Tutoring Systems - 21st International Conference, ITS 2025,*
184 *Alexandroupolis, Greece, June 2-6, 2025, Proceedings, Part I*, volume 15723 of *Lecture Notes*
185 *in Computer Science*, pages 259–269. Springer, 2025. doi:10.1007/978-3-031-98281-1_21.
186 **13** Nathalie Bonneton-Botté, Sylvain Fleury, Nathalie Girard, Maëlys Le Magadou, Anthony
187 Cherbonnier, Mickaël Renault, Éric Anquetil, and Eric Jamet. Can tablet apps support the
188 learning of handwriting? an investigation of learning outcomes in kindergarten classroom.
189 *Comput. Educ.*, 151:103831, 2020. URL: https://doi.org/10.1016/j.compedu.2020.103831,
190 doi:10.1016/J.COMPEDU.2020.103831.
191 **14** Tiphaine Colliot, Omar Krichen, Nathalie Girard, Éric Anquetil, and Éric Jamet. What
192 makes tablet-based learning effective? A study of the role of real-time adaptive feedback.
193 *Br. J. Educ. Technol.*, 55(5):2278–2295, 2024. URL: https://doi.org/10.1111/bjet.13439,
194 doi:10.1111/BJET.13439.
195 **15** Loris D'Antoni, Martin Helfrich, Jan Kretínský, Emanuel Ramneantu, and Maximilian Wein-
196 inger. Automata tutor v3. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided*
197 *Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24,*
198 *2020, Proceedings, Part II*, Lecture Notes in Computer Science, pages 3–14. Springer, 2020.
199 doi:10.1007/978-3-030-53291-8_1.
200 **16** Olivier Gasquet, Andreas Herzig, Bilal Said, and François Schwarzentruber. *Kripke's Worlds -*
201 *An Introduction to Modal Logics via Tableaux*. Studies in Universal Logic. Birkhäuser, 2014.
202 doi:10.1007/978-3-7643-8504-0.
203 **17** Omar Krichen, Nathalie Girard, and Éric Anquetil. Extension of a bi-dimensional grammar
204 for online interpretation of structured documents: application on architecture plans and
205 geometry domains. In *17th International Conference on Frontiers in Handwriting Recognition,*
206 *ICFHR 2020, Dortmund, Germany, September 8-10, 2020*, pages 325–330. IEEE, 2020. doi:
207 10.1109/ICFHR2020.2020.00066.
208 **18** Kim Marriott. Constraint multiset grammars. In Allen L. Ambler and Takayuki Dan Kimura,
209 editors, *Proceedings IEEE Symposium on Visual Languages, St. Louis, Missouri, USA, October*
210 *4-7, 1994*, pages 118–125. IEEE Computer Society, 1994. doi:10.1109/VL.1994.363633.
211 **19** T. Y. Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns.
212 *Commun. ACM*, 27(3):236–239, 1984. doi:10.1145/357994.358023.