Regex Parsons: Using Horizontal Parsons Problems to Scaffold Learning Regex

Zihan Wu
ziwu@umich.edu
University of Michigan, School of Information
Ann Arbor, Michigan, USA

Barbara Ericson
barbarer@umich.edu
University of Michigan, School of Information
Ann Arbor, Michigan, USA

Christopher Brooks
brooksch@umich.edu
University of Michigan, School of Information
Ann Arbor, Michigan, USA

ABSTRACT

Regular expressions (regex) are a popular text processing method used in programming. They are widely used in data analysis, web scraping, and input validation, and are supported by all mainstream programming languages. However, both students and even professional programmers perceive writing regex as difficult. Meanwhile, Parsons problems are a type of code completion problem used for introductory level programming education, and they can be a more efficient way to practice programming. Since regex is also a formal language, we created a horizontal version of Parsons problems to support teaching regex. We present results from an evaluation study in a MOOC comparing solving Parsons regex problems versus traditional regex problems. A higher percentage of students in the Parsons condition completed the practice and posttest. There was no significant difference in the learning gain from pretest to posttest between the conditions.

CCS CONCEPTS

• **Social and professional topics → Computing education**; • **Applied computing → Interactive learning environments**.

KEYWORDS

Regular expression, Parsons problems, MOOC dropout, learning gain

ACM Reference Format:
Zihan Wu, Barbara Ericson, and Christopher Brooks. 2021. Regex Parsons: Using Horizontal Parsons Problems to Scaffold Learning Regex. In *21st Koli Calling International Conference on Computing Education Research (Koli Calling '21), November 18–21, 2021, Joensuu, Finland.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3488042.3489968

1 INTRODUCTION

Regular expressions (Regex) are a text processing method used in programming. Regex uses characters and symbols to define search patterns and captures text patterns identified by the expression [5]. It is widely used in data analysis, web scraping, and input validation, and is supported by all mainstream programming languages. Despite its power, writing a regex is perceived as difficult by both students [1] and professional programmers [3]. Previous work has proposed using games to improve the teaching of regex by stimulating students' interests [4] and providing failing test cases as feedback [1]. However, we have not found prior research that attempts to scaffold regex practice.

Prior work on teaching introductory level programming has studied the effect of using Parsons problems, a type of programming practice question which requires students to rearrange mixed-up code in the correct vertical order. Compared with writing code, solving Parsons problems takes less time but has no significant difference in students' learning performance and retention [2, 6]. As regex is also a formal language, we are interested in adopting techniques similar to Parsons problems in teaching regex. More specifically, instead of asking students to write a regex from scratch, a regex Parsons problem asks students to choose from characters or blocks of characters and rearrange them horizontally. In this paper, we designed a system for horizontal Parsons problems as a new type of practice for regex.

2 SYSTEM DESIGN

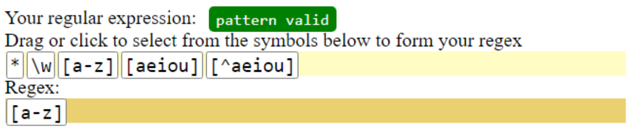
We designed a web-based tool for horizontal Parsons problems with supportive features such as real-time regular expression matching.

Traditional Parsons problems are designed for code solutions with multiple lines of code and provide the code blocks in a mixed-up order for learners to fix. Regular expression problems have solutions that fit on a single line, whereas code problems typically are several lines of code. In addition, the symbols in the solutions are often repeatedly used. Thus, instructors would need to divide the answer for regex problems into code blocks, which also need to be reusable.

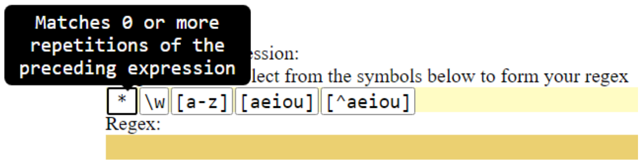
Fig. 1 demonstrates the interface for constructing a Parsons regex solution. It provides a list of blocks for solving the problem, and the learners can drag and drop the blocks to the input area below or click on the blocks to add them to the end of the input area. Tooltips explain the meaning of the blocks. When users hover over a symbol in the block, a tooltip will show them the explanation for that symbol.

Several important features of Parsons problems have been applied in this new context. First, the input includes distractors, which require learners to compare symbols and blocks that are easily confused. Second, combined blocks (e.g. block "[A-Z]" in Fig. 1(a)) are provided as scaffolds for learners to reduce the complexity of the problem. For a combined block, the tooltip explains the meaning of the whole block.

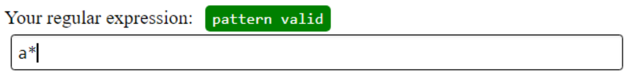
The tool also includes supportive features that gives timely feedback on the current solution, including immediate feedback on the



(a) A Parsons-style input for regex.


Matches 0 or more repetitions of the preceding expression
ssion:
lect from the symbols below to form your regex
*
\w
[a-z]
[aeiou]
[^aeiou]
Regex:

(b) A tooltip appears when hovering on a block in a Parsons-style input.



(c) A traditional text-entry-style input for regex.

Figure 1: A demonstration of Parsons-style input and traditional text-entry-style input for regex.

correctness of the regex, the current matches in the provided test string, and the results from the test cases. These features are available in both the Parsons and traditional text-entry input versions.

3 EVALUATION STUDY AND FIRST RESULTS

To evaluate Parsons-regex as a scaffolding technique, we conducted a between-subjects study to compare the effect of Parsons regex problems and traditional regex problems. We next report the first results from this study.

3.1 Participants and Methods

A field study in an introductory level data science MOOC was conducted in Coursera. The study material was deployed as an optional ungraded assignment. It contained a pretest, five practice problems, and a posttest. The tests used isomorphic problems. It included different types of problems to evaluate learners' ability to memorize regex symbols, read regex patterns, and write a regex based on a problem description. To encourage students to complete the assignment and give honest answers, students could choose "I don't know" for each test question. The students were randomly assigned to two groups, the Parsons group and the text-entry group. The only difference between the two groups was the format of the five practice questions they received: the Parsons group used the drag-and-drop input, while the text-entry group used plain text-entry input.

3.2 Results

After deploying the tool in the MOOC for 35 days, 2,103 students started the optional assignment, 1,573 completed the pretest, 734 students completed the practice problems, and 462 students completed the entire study.

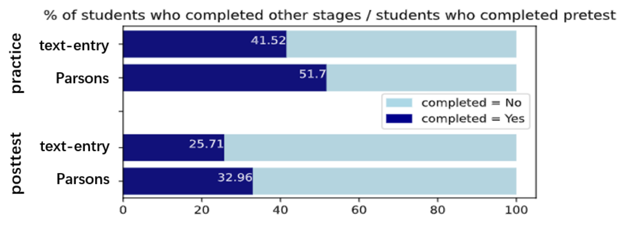


Figure 2: Percentage of students who completed the practice problems/posttest out of students who completed the pretest in each group.

Table 1: Pre-test and Learning gain

Stage	Text-Entry (n = 128) $\bar{x}$	σ	Parsons (n = 162) $\bar{x}$	σ	t-test
Memorize Symbol: multiple choice questions, full mark is 2					
pretest	0.77	0.46	1.07	0.59	-3.48**
learning gain	0.42	0.47	0.54	0.49	-1.49
Read Regex: multiple answer questions, full mark is 4					
pretest	2.30	1.46	2.57	1.39	-1.91
learning gain	0.25	1.64	0.07	2.24	1.10
Write regex: a text-input question, full mark is 10					
pretest	5.34	6.59	5.51	4.81	-0.60
learning gain	1.14	9.38	1.30	4.96	-0.52

$^{*}p < .05$. $^{**}p < .001$.

3.2.1 *Dropout Rate.*

Out of the 2,103 students who started the optional assignment pages, only 1,573 of them completed the pretest. 795 students were randomly assigned to Parsons group, and 778 students were assigned to text-entry group. Fig. 2 shows the percentage of students in each group that completed practice problems and the posttest. A significantly lower percentage of students from the text-entry group completed the activities compared to the Parsons group.

3.2.2 *Learning Gain.*

The average pretest score for the three types of questions before practice problem did not show a significant difference, which indicates the population for two conditions are at the same level in terms of regex. We calculated the learning gain for students who answered both pretest and post test questions, and dropped the data where students only completed part of the optional assignment or selected the "I don't know" option. Table 1 shows the pretest score and learning gain for three types of test questions.

4 SUMMARY

In this poster, we introduced a horizontal Parsons problem tool to scaffold learners while practicing regex problems and presented the results from a between-subjects experiment. Like traditional vertical Parsons problems, horizontal Parsons problems reduce the problem space for learners. Initial results provided evidence that students retention rates were higher for Parsons regex problems versus traditional free text entry problems with no significant difference in the learning gain.

REFERENCES

[1] Christopher W. Brown and Eric A. Hardisty. 2007. RegeXeX: an interactive system providing regular expression exercises. *ACM SIGCSE Bulletin* 39, 1 (March 2007), 445–449. https://doi.org/10.1145/1227504.1227462

[2] Barbara J. Ericson, Lauren E. Margulieux, and Jochen Rick. 2017. Solving Parsons Problems versus Fixing and Writing Code. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '17)*. Association for Computing Machinery, New York, NY, USA, 20–29. https://doi.org/10.1145/3141880.3141895

[3] Louis G. Michael, James Donohue, James C. Davis, Dongyoon Lee, and Francisco Servant. 2019. Regexes are Hard: Decision-Making, Difficulties, and Risks in Programming Regular Expressions. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, New York, U.S., 415–426. https://doi.org/10.1109/ASE.2019.00047

[4] Ariel Rosenfeld, Abejide Ade-Ibijola, and Sigrid Ewert. 2017. Regex Parser II: Teaching Regular Expression Fundamentals via Educational Gaming. In *ICT Education (Communications in Computer and Information Science)*, Janet Liebenberg and Stefan Gruner (Eds.). Springer International Publishing, Cham, 99–112. https://doi.org/10.1007/978-3-319-69670-6_7

[5] Ken Thompson. 1968. Programming techniques: Regular expression search algorithm. *Commun. ACM* 11, 6 (1968), 419–422.

[6] Rui Zhi, Min Chi, Tiffany Barnes, and Thomas W. Price. 2019. Evaluating the Effectiveness of Parsons Problems for Block-Based Programming. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 51–59. https://doi.org/10.1145/3291279.3339419