Using Micro Parsons Problems to Scaffold the Learning of Regular Expressions

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

Barbara J. Ericson
barbarer@umich.edu
School of Information,
University of Michigan
Ann Arbor, Michigan, United States

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

ABSTRACT

Regular expressions (regex) are a text processing method widely used in data analysis, web scraping, and input validation. However, students find regular expressions difficult to create since they use a terse language of characters. Parsons problems can be a more efficient way to practice programming than typing the equivalent code with similar learning gains. In traditional Parsons problems, learners place mixed-up fragments with one or more lines in each fragment in order to solve a problem. To investigate learning regex with Parsons problems, we introduce micro Parsons problems, in which learners assemble fragments in a single line. We conducted both a think-aloud study and a large-scale between-subjects field study to evaluate this new approach. The think-aloud study provided insights into learners' perceptions of the advantages and disadvantages of solving micro Parsons problems versus traditional text-entry problems, student preferences, and revealed design considerations for micro Parsons problems. The between-subjects field study of 3,752 participants compared micro Parsons problems with text-entry problems as an optional assignment in a MOOC. The dropout rate for the micro Parsons condition was significantly lower than the text-entry condition. No significant difference was found for the learning gain on questions testing comprehensive regex skills between the two conditions, but the micro Parsons group had a significantly higher learning gain on multiple choice questions which tested understanding of regex characters.

CCS CONCEPTS

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

KEYWORDS

regular expressions; regex; computer science education; Parsons problems; micro Parsons problems

ACM Reference Format:
Zihan Wu, Barbara J. Ericson, and Christopher Brooks. 2023. Using Micro Parsons Problems to Scaffold the Learning of Regular Expressions. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2023), July 8–12, 2023, Turku, Finland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3587102.3588853

ITiCSE 2023, July 8–12, 2023, Turku, Finland

ACM ISBN 979-8-4007-0138-2/23/07…$15.00
https://doi.org/10.1145/3587102.3588853

1 INTRODUCTION

Regular expressions (regex) are a text-processing method that uses characters to define search patterns for text capture [31]. They are widely used in data analysis, web scraping, and input validation, and are supported by all mainstream programming languages. Despite their power, both students [5] and even professional programmers [22] find writing regex difficult. Previous work has proposed using games to improve the learning of regex by stimulating students' interest [27] and providing failing test cases as feedback [5]. However, we found no prior research that attempts to scaffold the process of writing regex with Parsons problems [12].

Parsons problems require learners to rearrange mixed-up code lines in the correct order instead of writing code from scratch. Parsons problems were designed to maximize engagement, model good code, and provide immediate feedback [26]. In traditional Parsons problems, each block contains one or more lines of code. They have been found to be beneficial in terms of problem-solving efficiency [13], student engagement [14], and programming pattern acquisition [33]. Solving Parsons problems produces equivalent learning gains as writing code from scratch [13, 15, 33].

However, regex typically contain complicated information within a single line. To provide scaffolding for novices in regex, we introduce micro Parsons problems, a type of Parsons problems that provides code line fragments for learners to rearrange into the correct order in a single line. We shared preliminary results from our studies in a poster [37]. With the goal of understanding the effect of micro Parsons problems on learners, we proposed the following research questions:

RQ1 - What are learners' perceived advantages and disadvantages of micro Parsons problems compared with text-entry regex problems and which do they prefer?

RQ2 - Compared with text-entry regex problems, how do micro Parsons problems affect learners' dropout rate from an optional practice activity, problem completion time, cognitive load, and learning gain?

2 RELATED WORK

2.1 Cognitive Load Theory

Cognitive load theory (CLT) explains the relationship between humans' limited capacity of working memory and learning [29]. According to CLT, learning is the process of constructing and storing domain-specific knowledge in the form of schemas in long-term memory [30]. Schemas are representations of knowledge as well as

information-processing mechanisms [9]; thus, the construction of schemas is crucial for learning. For learners to construct schemas in long-term memory, they need to first process new knowledge in working memory. As working memory has a limited capacity, it can become the bottleneck for learning if there is too much information to process at once. There are two types of cognitive load: intrinsic load and extrinsic load. Intrinsic load refers to the cognitive load directly associated with the level of difficulty of the concept, and extraneous load is generated by the way information is presented to learners, and should be reduced in instructional activities to avoid cognitive overload [30]. According to Van Merrienboer et al. [32], replacing whole tasks, such as writing code from scratch, with *completion problems* where part of the solution is provided can reduce extraneous cognitive load. Parsons problems are a type of code completion problem that require learners to reconstruct the solution, and thus might reduce extraneous cognitive load for learners [13, 15].

2.2 Parsons Problems

Parsons problems require learners to rearrange mixed-up code into the correct order. Extra code blocks that are not part of the correct answer, called *distractors*, are often included in Parsons problems to highlight syntax errors and common misconceptions [17, 21, 26]. Parsons problems have been found beneficial in terms of problem-solving efficiency [13], student engagement [14], and programming pattern acquisition [33].

Researchers have also investigated different types of Parsons problems [10, 12]. Some have investigated adaptive Parsons problems, which dynamically adjust the difficulty of Parsons problems based on a learner's previous performance and provide help on demand [11]. Others explored different types of feedback provided by Parsons problems, including line-based feedback and execution-based feedback [19]. Another variant proposed more recently is faded Parsons problems, in which the code blocks contain incomplete code with blanks that learners need to fill in [34]. Although prior work has explored many different features of Parsons problems, we are not aware of any research on Parsons problems that ask learners to assemble fragments in a single line [12].

2.3 Systems for Learning Regular Expressions

Most prior work on systems that help learners to learn regex are educational games that aim to encourage practice. *Regex Parser II* [27] asks learners to provide test cases for a given regex, and *Learn Regex* [28] situates regex practice in a jewelry shop and asks learners to create patterns of beads with regex syntax. Some systems explore different types of feedback provided to the students, such as failed examples [5, 6], descriptive hints [6], and correct/incorrect binary feedback [6]. Many websites also provide practice that introduces regex syntax and special symbols to learners through different levels, such as Regex Learn [1], RegexOne [2], and RegExr [3]. Existing types of regex practice problems include writing regex to match given strings [1, 2, 4], writing strings that match or do not match a given regex [4, 27], or determining if the provided string and regex matches [4]. We did not find any prior work that scaffolds the process of constructing a regex with Parsons problems.

3 REGEX MICRO PARSONS SYSTEM

We designed a web-based tool that supports students solving regex problems with either a micro Parsons problem or traditional free text entry. The tool also includes other features specifically designed for regex practice, such as real-time pattern matching. Fig. 1 displays the interface for a micro Parsons problem.

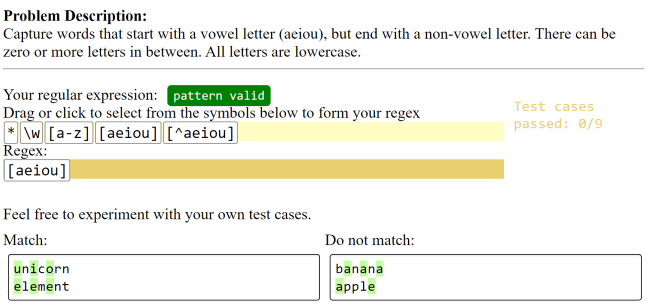


Figure 1: An example problem in the tool with a problem description, micro Parsons input, and supporting features including highlighted test strings, the validity of the regex, and the score on hidden test cases.

Similar to traditional Parsons problems, our micro Parsons interface provides blocks and asks learners to rearrange them in the correct order. We iterated the design according to the feedback from the think-aloud study described in section 4. The final design (shown in fig. 1) provides a list of the possible input blocks in the top row, and learners can then drag these blocks into position on the second row and see the execution results on the bottom.

Several important features of traditional Parsons problems were adopted in this new context. First, the input includes distractors (extra blocks), which require learners to compare blocks that are easily confused. For example, a `*` (an asterisk which repeats the preceding regex zero or more times) can be included in a problem with a `+` (a plus which repeats the preceding regex one or more times). Second, we adopted execution-based feedback similar to [19, 33] that compiles the regex, provides test cases that match and do not match, and highlights the matches in the test cases.

There are also new features that are unique to micro Parsons problems. The most important design consideration for creating micro Parsons problems is to maintain the atomicity of blocks. In traditional Parsons problems, each line of code is an individual unit, and the arrangement of blocks will not influence the interpretation of the code. However, in micro Parsons problems the order of operations can influence the result. For example, if the blocks provided in the micro Parsons problem include `abc` and `+`, when learners arrange them into `abc+`, the repetition only applies to the last character, which will confuse learners as they would expect the content inside a block to be indivisible. Thus, we use parentheses to enclose the content `(abc)` to maintain the intended meaning of `(abc)+`. This applies to contexts outside of regex, such as providing `a + b`, `*`, and `c` in other programming languages that support arithmetic. In this case, we could provide `(a + b)` to make it clear that the addition should happen before the multiplication. Micro Parsons problems can contain blocks that have more than one character

(e.g. block `[A-Z]`, which means any upper case characters between A and Z inclusive), just like traditional Parsons problems can have more than one line of code in a block.

We also introduced other features based on the need for learning regex. First, as regex sometimes includes repeated patterns, the blocks in this system are reusable. When learners add a block to their regex, the block in the top row does not disappear. Another feature is to include explanations for the blocks. Regex uses many symbols to indicate specific meanings, such as repetition (e.g. `*`, `+`), character sets (e.g. `[abcde]`), and character classes (e.g. `\d`). Novices often need cheat sheets during practice. Thus, we embedded explanations for the blocks in the tool in the form of a tooltip which displays when the user pauses the cursor on a block. The interface also shows what matches the regex and the score on hidden test cases.

4 THINK ALOUD STUDY

We first conducted a within-subjects think-aloud study with eight participants to better understand student reactions to both micro Parsons and text-entry problems. Participants were enrolled in a course where regex was being taught for the first time (either at the undergraduate level or professional Masters level). During the interview, we asked participants if they were confident about their ability to write regex, and only two participants reported that they were confident. Each participant spent approximately 45 minutes on the study and received a $20 gift card for completing the study.

The study was conducted through videoconferencing software, and we recorded the audio and participants' screens after obtaining consent. The participants first completed a brief tutorial on the think-aloud process and how to use the tool, and then moved on to work on a set of four problems from easy to difficult.

Each question had two different versions: one was a micro Parsons problem (MP), and the other was a traditional text-entry problem (TE). The two versions only differ in their input mechanism and share all other features. Participants were randomly assigned to one of the two groups. One group solved the four problems as TE, MP, TE, MP while the other solved the exact same problems as MP, TE, MP, TE. The goal was to reduce any possible order effects. All participants were provided with a cheatsheet that contained explanations of regex characters and combinations of characters. They were asked to verbalize their thought processes while solving the problems. For each question, when they passed all test cases or spent at least seven minutes on a problem (which was enough time to complete the problem independently if not stuck), they were prompted to move on to the next problem. Finally, a semi-structured interview was conducted to gather advantages, disadvantages, preferences, and feedback on the tool interface.

4.1 Results

4.1.1 Perceived advantages and disadvantages of micro Parsons problems. When asked their opinions about the two different types of regex problems, all participants reported that micro Parsons problems were easier than text-entry problems. Some commented that micro Parsons problems provided more scaffolding and were steering them in a certain direction, and they were more willing to experiment with characters and combinations they were not familiar with to form possible solutions.

Some participants expressed that micro Parsons felt more constrained. For example, while solving a micro Parsons problem, P7 asked *"Can I just type it in myself?"* They explained that it was because they had a different solution in mind. However, this type of constraint can also be beneficial. Some participants expressed that they appreciated that the micro Parsons problem "forced" them to use a limited set of characters. Learners' behaviors in the text-entry condition also suggested that this restriction might be necessary. When trying to match a non-vowel lowercase letter in the text-entry condition, P2 attempted to use square brackets with all 21 letters to match a non-vowel letter, while the only option available in the micro Parsons condition was the character set that used negation instead: `[^aeiou]`.

4.1.2 Learners' preferences for micro Parsons problems versus text-entry problems. At the end of the interview, we asked each participant about their preference between the micro Parsons problems and text-entry problems. All participants found micro Parsons problems easier than text-entry problems. Four participants (50%) preferred text-entry problems, mostly because they felt less constrained, and the text-entry problems felt closer to the authentic task. Both of the participants who were confident in their ability to create regex before the study preferred text-entry problems. Three participants (38%) preferred micro Parsons problems, and expressed that the blocks gave them something to start with by trying different things. P7 expressed that they were curious about the micro Parsons problems. This suggested that the novelty of micro Parsons problems could be more interesting and thus more motivating for some learners. One participant (13%) said that if their goal was to complete the problem, they would prefer a Parsons problem, but if their goal was to learn more, they would prefer a text-entry problem. This is consistent with previous findings [18], in which the researchers found that learners found Parsons problems easier than writing textual code, but felt they would learn more while solving text-entry problems if they were not stuck.

5 BETWEEN-SUBJECTS FIELD STUDY

We conducted a between-subjects study in a MOOC with one condition solving micro Parsons problems and the other solving text-entry problems in order to investigate differences in learners' dropout rate from an optional practice activity, learning gains, problem completion time, and cognitive load.

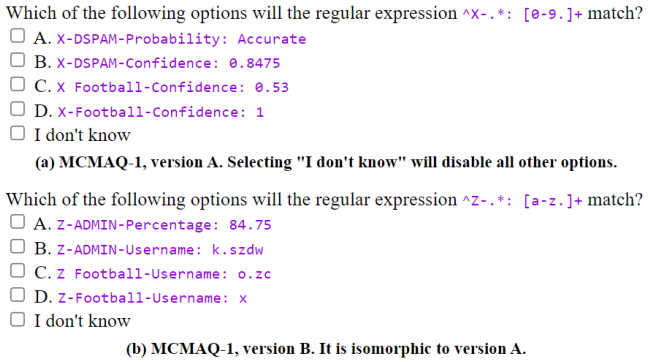
5.1 Participants and Procedure

With IRB approval, we conducted the study in an introductory-level data science MOOC opened to the public. The course included regex as a method for text manipulation. The study material was deployed as an optional ungraded assignment for the learners to practice regular expressions. We used MD5 to hash all students' IDs to deidentify the data.

Upon starting the ungraded assignment, learners were randomly assigned to one of two conditions: micro Parsons condition or text-entry condition. As the tasks were delivered through the MOOC. Learners could choose to exit the study or come back to where they

left off at any time. Learners completed a pretest, a practice session, and a posttest in that order.

The pretest contained two multiple-choice questions (MCQs) that evaluated learners' knowledge of the meaning of regex characters, two multiple-choice-multiple-answer questions (MCMAQs) asking learners to choose all the strings that would be matched by a given regex, and one text-entry question asking learners to write a regex according to a problem description. To encourage learners to answer the questions without getting outside help, we added prompts to remind learners that the test was not graded, and provided an "I don't know" for all choice-based questions. The posttest contained the same numbers and types of questions as the pretest. To minimize the effect of learning from answering the pretest questions, we created isomorphic questions that were slightly different but did not change what regex symbols were being tested by the problem (see fig. 2) for the second MCQ and the two MCMAQs. The first MCQ and the text-entry regex problem were kept the same as they mostly consisted of written descriptions, and it was difficult to create isomorphic questions that did not change the meaning of the problems. For the questions that had isomorphic variants, learners randomly received one version of the isomorphic problem in the pretest and the other version in the posttest. Learners did not receive any feedback for their test answers until they completed the posttest, after which they were shown the correct answers to all the test questions. The test questions were developed based on the practice problems such that all the characters and combinations of characters that were tested were covered in the practice problems. We tested the problems with three volunteers with introductory knowledge of regex to ensure the wording of the problems was clear prior to starting the study.



(a) MCMAQ-1, version A. Selecting "I don't know" will disable all other options.

(b) MCMAQ-1, version B. It is isomorphic to version A.

Figure 2: An example of a pair of isomorphic problems used in the test questions.

After each participant completed a brief tutorial on the type of entry used in their condition (micro Parsons or text-entry), they were redirected to the practice problems. The set of practice problems contained five problems, four of which were the same problems as in the think-aloud study. For each practice problem, learners could choose to spend any time on it until they passed all test cases or could skip the problem. When learners successfully passed all of the test cases for a practice problem, they were asked to rate their perceived cognitive load on the Paas scale [24], a 9-point Likert scale with evidence for reliability and validity [25]. This scale has been widely used to measure working memory load [23].

5.2 Results

The tool was deployed on a MOOC platform for 14 weeks, and 5,683 students clicked into the webpage for the study. 3,752 (66.02%) learners completed the pretest, and were considered participants. 1,899 students were assigned to the Parsons group, and 1,853 students were assigned to the text-entry group. All analysis was done in Python with Scipy and Statsmodels packages. To verify the two groups had a similar level of prior knowledge in regex, we used Welch's t-test [1] to compare the pretest score of two groups, and no difference was found on any type of pretest questions at a 0.01 level of significance.

5.2.1 Dropout Rate From The Optional Practice Activity. Table 1 shows the number of students who dropped out (did not complete the activities) by the end of each part of the study, and the dropout rate of the two groups. We used a two proportion z-test to test for a signficant difference between conditions (see table 1). Two proportion z-tests are used to compare the proportions of two independent samples, and produce the same p-value as the Chi-Squared test for homogeneity (2x2).

Table 1: Two Proportion Z-Test of Dropout Rate

	Parsons ($n = 1,899$)		Text-Entry ($n = 1,853$)		z	Cohen's h
	n	%	n	%		
Prac.	896	47.18	1008	54.40	-4.42***	0.14
Post.	1248	65.72	1311	70.75	-3.31***	0.11

$^{*}p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$.

There was a significant difference in the dropout rate during the practice problems by condition (p <.001). During the practice problem session, around 47% (896 of 1,899) of the micro Parsons group dropped out (did not complete the activities), and around 54% (1,008 of 1,853) of the text-entry group dropped out. There was also a significant difference between the dropout rates by the end of the posttest (p <.001). By the end of posttest, around 66% (1,248 of 1,899) of the micro Parsons group dropped out, and around 71% (1,311 of 1,853) of the text-entry group dropped out.

5.2.2 Problem Completion Time. The problem completion time was computed as the elapsed time between the point when the learner navigated to the page for each practice problem and when the learner passed all the test cases for a problem. After observing that the median time spent on each problem was under 12 minutes, we removed outliers who spent more than an hour on one problem, as they probably took breaks while solving a single problem.

We also performed a Welch's t-test according to [8, 35] and calculated Hedge's g as the effect size [7] for the time spent for both groups on each practice problem (see table 2). For the first three practice problems, there was a significant difference between the

[1]According to [8, 35], Welch's t-test is preferred over a Student's t-test in the social sciences, especially when the two groups have unequal sample sizes.

Table 2: T-Test for Problem Completion Time

#	Parsons			Text-Entry			t	p	g
	n	$\overline{x}$	σ	n	$\overline{x}$	σ			
1	765	5.0	35.6	390	12.3	126.4	-12.0***	<.001	0.90
2	781	4.5	38.9	437	9.7	111.5	-9.3***	<.001	0.64
3	784	4.4	32.8	475	5.9	53.5	-3.9***	<.001	0.24
4	471	11.9	118.6	370	12.1	120.9	-0.2	0.807	0.02
5	378	14.1	120.6	274	14.2	124.3	-0.1	0.909	0.01

$^{*}p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$.

average time taken to complete a micro Parsons problem and the average time taken to complete the equivalent text-entry problem (p <.001 for all three problems), and the average time taken for micro Parsons problems was lower than the text-entry condition. However, there was no statistically significant difference by condition for the last two practice problems that were designed to be more difficult.

5.2.3 Cognitive Load. We also used a Welch's t-test to understand if there was a difference between conditions in terms of self-reported cognitive load (see table. 3). For the first two practice problems, learners' self-reported cognitive load from the Parsons group was significantly lower than the text-entry group (p <.001). The average reported cognitive load by the micro Parsons group for the first two problems was 4.81 and 4.45, and the average reported cognitive load for the text-entry group was 5.33 and 5.03. For the fourth problem, learners from the Parsons group reported a significantly higher cognitive load (p <.001).

Table 3: T-Test for Self-Reported Cognitive Load

#	Parsons			Text-Entry			t	p	g
	n	$\overline{x}$	σ	n	$\overline{x}$	σ			
1	801	4.8	2.8	456	5.3	2.5	-5.5***	<.001	0.32
2	804	4.5	2.9	479	5.0	2.9	-5.9***	<.001	0.34
3	811	4.3	3.0	501	4.2	2.9	1.4	0.168	0.08
4	519	6.0	2.4	399	5.3	2.8	6.9***	<.001	0.46
5	392	6.2	2.6	294	6.0	2.7	2.1*	0.040	0.16

$^{*}p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$.

5.2.4 Learning Gain. We calculated the participants' scores including the two multiple-choice questions (MCQs) that were testing learners' abilities to recall the meaning of regex characters (2 points, one point for each question), two multiple-choice-multiple-answer questions (MCMAQs) that were testing learners' abilities to read regex and identify corresponding strings (8 points, one point for each option) (see fig. 2), and one text-entry question (10 points, one point for each unit test) to test learners' abilities to write a regex. The learning gain was calculated by subtracting an individual's pretest score from their posttest score. Scores for the different types of problems were calculated separately as they were intended to test different skills, and it is difficult to assign appropriate weights on each type of problem to compute a full mark.

Table 4: T-Test for Learning Gain

Question Type	Parsons (n = 651)		Text-Entry (n = 542)		t	p	g
	$\overline{x}$	σ	$\overline{x}$	σ			
MCQs	0.61	1.40	0.36	2.18	5.25***	<.001	0.30
MCMAQs	0.56	4.04	0.55	4.25	0.11	0.913	0.01
Text-Entry	0.84	3.74	0.89	3.74	-0.26	0.796	0.02

$^{*}p < 0.05$, $^{**}p < 0.01$., $^{***}p < 0.001$

A Welch's t-test was performed on the pretest score of participants in two groups, and no significant difference was found between the two groups for any type of question. We also used a Welch's t-test to compare the learning gain of the two independent groups as suggested by [8, 35] (see table 4). For MCQs that were testing the understanding of regex symbols (full points: 2), the average learning gain for learners in the micro Parsons problem group was 0.61, which was significantly higher (p <.001) from the text-entry group, whose average learning gain was 0.36. There was no significant difference by condition for the MCMAQs and the text-entry regex problem that were testing the more comprehensive abilities to identify strings that can be matched by a given regex (p = 0.913) and write a regex according to a problem description (p = 0.796). We also performed a Welch's t-test on the pretest score of participants who completed both the pretest and the posttest in the two groups, and no difference was found on any type of test questions using a 0.01 level of significance.

6 DISCUSSION

RQ1: Perceived advantages and disadvantages of micro Parsons problems and learners' preferences. In our first study, we observed several potential benefits of micro Parsons problems over traditional text-entry problems for some learners. First, micro Parsons problems can be more motivating for some learners since they are puzzle-like. Second, micro Parsons can provide more scaffolding by limiting the problem space. For novices who are not familiar with regex, micro Parsons problems provide a limited set of characters or groups of characters for them to start with, instead of going through a cheat sheet or other resources to locate characters that might be useful. In the think-aloud study, some participants expressed that the scaffolding made them feel more comfortable exploring their own solutions and experimenting with unfamiliar combinations of characters. Third, the limit in solution space guarantees learners' exposure to certain characters or groups of characters that they might not attempt to use otherwise, which can be particularly useful in an instructional setting. However, certain features of micro Parsons problems are also considered disadvantages for some learners. The limited solution space, for example, takes away some of the learners' freedom to create solutions that are not supported by the given blocks in a micro Parsons problem. This is an example of the expertise reversal effect [20, 29], an effect predicted by cognitive load theory, where the information intended to provide more scaffolding for novices becomes redundant for advanced learners who have already constructed internal schemas. In addition, micro Parsons problems are not perceived as ideal for learners who want to practice in an authentic environment.

Regarding learners' preferences, we found that learners have mixed preferences towards micro Parsons problems and text-entry problems, which is consistent with prior findings [18].

In general, learners perceive micro Parsons problems as easier than text-entry problems regardless of their prior experience. Regex contains many characters and one regex problem can often be solved by many different regex patterns. One of the major features of micro Parsons problems is that they limit the solution space, which leads to both advantages and disadvantages as shown in our think-aloud study.

RQ2: Micro Parsons reduced the dropout rate for optional regex practice activities. Our study showed that using micro Parsons problems can reduce the dropout rate for optional practice activities with regex in a MOOC compared with traditional text-entry problems (see 5.2.1). Several potential reasons might have contributed to this result. First, micro Parsons problems provided more scaffolding than plain text-entry regex problems, and guaranteed that learners always had something to experiment with when they felt stuck. Learners' subjective feedback was also supportive of micro Parsons problems' ability to provide scaffolding. Second, as suggested by participants' reasons for preferring micro Parsons problems, they might be more enjoyable to solve and less daunting. As discovered by [16], both enjoyment and perceived difficulty of the content have effects on learners' dropout behavior in MOOC. According to the expectancy-value theory of motivation [36], intrinsic value (enjoyment) also correlates with learners' choice to continue learning activities.

RQ2: Micro Parsons problems' effect on problem-solving time and self-reported cognitive load varied by problem. A significant difference in problem completion time between conditions was found for the first three practice problems, which were designed to be easier. The decrease in problem completion time in Parsons conditions might be because micro Parsons problems provided more scaffolding, or the drag-and-drop interactions were easier to perform than text entry. With respect to cognitive load, a significant difference between the two groups was found for the first two problems (the two easiest problems) and the fourth problem.

The different effects on easy and hard problems could be related to the differences in the dropout rate by condition, which are further discussed in section 7. It is also possible that although micro Parsons problems were able to provide scaffolding for simple regex problems, they did not provide enough scaffolding for learners for more complex regex problems.

In the analysis of self-reported cognitive load, problem four demonstrated a reverse pattern as the micro Parsons condition group reported a significantly higher cognitive load. We identified one potential reason. Our micro Parsons tool included a UI component "expandable blocks", which automatically expanded a pair of parentheses into two blocks. In problem four, when learners clicked the `(?:)` block, it automatically expanded to `(?:` and `)`, which might have been an unexpected result. Learners were only briefly exposed to this type of block in the tutorial. Thus, it is possible that this feature added extraneous cognitive load for learners in the Parsons condition.

RQ2: The micro Parsons group had a significantly better learning gain on the two MCQs testing the recall of regex characters or characters combinations, but no significant difference was found for the more complex type of test problems. The higher learning gain on the recall MCQs in the Parsons group could be from direct exposure to regex character combinations and the tooltips that explained the meaning of the block. The second MCQ tested the understanding of a non-capturing group `(?:)`, which was included as a block in the fourth practice problem in the micro Parsons group. Learners had to use a non-capturing group to solve the problem and the explanation of the block was also included in the tool which ensured the exposure. Although the text-entry group received the same problem description, they can bypass the use of the non-capturing group by using additional parentheses around the whole regex. This can serve as an example of using micro Parsons problems to ensure exposure to specific characters and character combinations instead of only using what the learner already knows. However, we did not find a significant difference between conditions for the MCMAQs and text-entry questions that required more comprehensive skills in regex.

7 LIMITATIONS AND FUTURE WORK

One limitation of the think-aloud study is the limited number of participants. As the field study was unsupervised in nature, there are several limitations, one of which is related to the selection effect. During the study, learners could decide to drop out at any time, and as observed in section 5.2.1, there was a significant difference between the dropout rate between the two groups. This might result in a different selection effect for the two groups in the study. The micro Parsons condition might have kept learners who are less competent in regex, and who would have otherwise dropped out in the text-entry condition due to frustration. Thus, the learning gain, cognitive load, and problem-solving time could be effected by the significant difference in dropout rate. Future work should investigate the effect in other contexts where differential dropout is less likely, such as in a classroom study. As the field study was deployed as a completely optional assignment, we only included a limited number of questions in the pretest and the posttest. This limits the generalizability of the learning gain in this study, and also suggests that future work can further explore micro Parsons problems' effect on learning gain with more questions in the pretest and posttest. In addition, as the study was an optional assignment in a MOOC, people who entered the study could be motivated to learn and practice, which limits the generalizability of the result to a broader population. As participants were not supervised during the study, it was also possible that they navigated away from the page or even searched for answers.

This research was the first to explore the use of the micro Parsons problems to scaffold learning regex. It found that a significantly higher percentage of students completed the practice problems in the micro Parsons condition than in the text-entry condition, which provides evidence that they successfully scaffolded or motivated some students during regex practice. It also provided evidence that micro Parsons practice resulted in significantly better learning gains with respect to particular characters/character combinations than the text-entry condition. We plan to explore the potential benefits of micro Parsons problems in different contexts, such as for creating SQL statements or when creating a single line of Java code. Future work can also explore the transition from using micro Parsons problems to the authentic task of text-entry problems.

REFERENCES

[1] 2023. Regex Learn - Regex Interactive Course. https://regexlearn.com/learn

[2] 2023. RegexOne - Learn Regular Expressions. https://regexone.com/

[3] 2023. RegExr: Learn, Build, & Test RegEx. https://regexr.com/

[4] Pavel Azalov, Michelle Cullen, and Robert Rinish. 2004. ReExpress: a tutor for regular expressions mentoring with technology. In *Proceedings of the 5th conference on Information technology education*. 268–268.

[5] 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.

[6] Loris D'antoni, Dileep Kini, Rajeev Alur, Sumit Gulwani, Mahesh Viswanathan, and Björn Hartmann. 2015. How can automatic feedback help students construct automata? *ACM Transactions on Computer-Human Interaction (TOCHI)* 22, 2 (2015), 1–24.

[7] Marie Delacre, Daniel Lakens, Christophe Ley, Limin Liu, and Christophe Leys. 2021. Why Hedges'g* s based on the non-pooled standard deviation should be reported with Welch's t-test. (2021).

[8] Marie Delacre, Daniël Lakens, and Christophe Leys. 2017. Why psychologists should by default use Welch's t-test instead of Student's t-test. *International Review of Social Psychology* 30, 1 (2017).

[9] Paul DiMaggio. 1997. Culture and cognition. *Annual review of sociology* 23 (1997).

[10] Yuemeng Du, Andrew Luxton-Reilly, and Paul Denny. 2020. A Review of Research on Parsons Problems. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*. 195–202.

[11] Barbara Ericson, Austin McCall, and Kathryn Cunningham. 2019. Investigating the Affect and Effect of Adaptive Parsons Problems. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. 1–10.

[12] Barbara J Ericson, Paul Denny, James Prather, Rodrigo Duran, Arto Hellas, Juho Leinonen, Craig S Miller, Briana B Morrison, Janice L Pearce, and Susan H Rodger. 2022. Parsons Problems and Beyond: Systematic Literature Review and Empirical Study Designs. *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education* (2022), 191–234.

[13] Barbara J. Ericson, James D. Foley, and Jochen Rick. 2018. Evaluating the efficiency and effectiveness of adaptive parsons problems. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. 60–68.

[14] Barbara J Ericson, Mark J Guzdial, and Briana B Morrison. 2015. Analysis of interactive features designed to enhance learning in an ebook. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*. 169–178.

[15] 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*. 20–29.

[16] Thommy Eriksson, Tom Adawi, and Christian Stöhr. 2017. "Time is the bottleneck": a qualitative study exploring why learners drop out of MOOCs. *Journal of Computing in Higher Education* 29, 1 (2017), 133–146.

[17] Kyle James Harms, Jason Chen, and Caitlin L Kelleher. 2016. Distractors in Parsons problems decrease learning efficiency for young novice programmers. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. 241–250.

[18] Carl C. Haynes and Barbara J. Ericson. 2021. Problem-Solving Efficiency and Cognitive Load for Adaptive Parsons Problems vs. Writing the Equivalent Code. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[19] Juha Helminen, Petri Ihantola, Ville Karavirta, and Satu Alaoutinen. 2013. How do students solve parsons programming problems?–execution-based vs. line-based feedback. In *2013 Learning and Teaching in Computing and Engineering*. IEEE, 55–61.

[20] Slava Kalyuga. 2009. The expertise reversal effect. In *Managing cognitive load in adaptive multimedia learning*. IGI Global, 58–80.

[21] Ville Karavirta, Juha Helminen, and Petri Ihantola. 2012. A mobile learning application for parsons problems with automatic feedback. In *Proceedings of the 12th koli calling international conference on computing education research*. 11–18.

[22] 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, 415–426.

[23] Fred Paas, Juhani E Tuovinen, Huib Tabbers, and Pascal WM Van Gerven. 2016. Cognitive load measurement as a means to advance cognitive load theory. In *Educational psychologist*. Routledge, 63–71.

[24] Fred G Paas. 1992. Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive-load approach. *Journal of educational psychology* 84, 4 (1992), 429.

[25] Fred GWC Paas, Jeroen JG Van Merriënboer, and Jos J Adam. 1994. Measurement of cognitive load in instructional research. *Perceptual and motor skills* 79, 1 (1994), 419–430.

[26] Dale Parsons and Patricia Haden. 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*. 157–163.

[27] 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.

[28] Julie M Smith. 2020. Learn Regex: A Novel Tool for Learning Regular Expressions. In *Proceedings of the 21st Annual Conference on Information Technology Education*. 293–293.

[29] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science* 12, 2 (1988), 257–285.

[30] John Sweller. 1994. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction* 4, 4 (1994), 295–312.

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

[32] Jeroen JG Van Merrienboer and John Sweller. 2005. Cognitive load theory and complex learning: Recent developments and future directions. *Educational psychology review* 17, 2 (2005), 147–177.

[33] Nathaniel Weinman, Armando Fox, and Marti A Hearst. 2021. Improving instruction of programming patterns with faded parsons problems. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–4.

[34] Nathaniel Weinman, Armando Fox, and Marti A. Hearst. 2021. Improving Instruction of Programming Patterns with Faded Parsons Problems. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–4.

[35] Robert M West. 2021. Best practice in statistics: Use the Welch t-test when testing the difference between two groups. *Annals of Clinical Biochemistry* 58, 4 (2021), 267–269.

[36] Allan Wigfield and Jacquelynne S Eccles. 2000. Expectancy–value theory of achievement motivation. *Contemporary educational psychology* 25, 1 (2000), 68–81.

[37] Zihan Wu, Barbara Ericson, and Christopher Brooks. 2021. Regex Parsons: Using Horizontal Parsons Problems to Scaffold Learning Regex. In *Proceedings of the 21st Koli Calling International Conference on Computing Education Research*. 1–3.