Scratch Analysis Tool(SAT): A Modern Scratch Project Analysis Tool based on ANTLR to Assess Computational Thinking Skills

Zhong Chang, Yan Sun
Beijing Key Laboratory
of Intelligent Telecommunications
Software and Multimedia
Beijing University of
Posts and Telecommunications
Beijing 100876, China
Email: viviansnow@bupt.edu.cn,
sunyan@bupt.edu.cn

Tin-Yu Wu
Department of Computer
Science and Information
Engineering
National Ilan University, Taiwan
E-mail: tyw@niu.edu.tw

Mohsen Guizani
Fellow of IEEE,
Department of Electrical &
Computer Engineering,
University of Idaho, U.S.A
E-mail: mguizani@uidaho.edu

Abstract—With the introduction of computer programming in schools around the world, Scratch has risen in prominence for its thinkable, meaningful and social. Aiming to assessing the Computational Thinking skills of a Scratch program, we design and implement a new Scratch program analysis tool based on ANTLR. To solve some flaws (e.g. high failure rate and low efficiency) in Dr. Scratch which is the most relevant tool to assess Computational Thinking skills of Scratch programs, we choose the recognition tool ANTLR to design the system module and the assessing flow. And then, we customize more than 200 lexical and syntax parser rules in ANTLR. Furthermore, we expand the grading standard of assessing Computational Thinking skills in Dr. Scratch. Some fundamental concepts in Computer Science, such as stack, queue and recursion method, are involved in our grading standard. Experiment results show the performance (e.g. success rate, execution time) of SAT is superior to that of Dr. Scratch.

Keywords—Analysis Tool, Scratch, Computational Thinking, ANTLR.

I. INTRODUCTION

Scratch\(^1\), which is a block-based visual programming environment from MIT, has risen in prominence for its thinkable, meaningful and social aspects\(^1\). Since the flexibility of block-based visual programming, many studies have demonstrated the applicability of Scratch as a tool for education\(^2\)\(^3\). For example, it has been proved to be feasible to teach some other disciplines by using Scratch. Particularly, an important research focus is to assess Computational Thinking (CT for short) skill of a Scratch program.

Computational Thinking put forward by Wing is a skill, which can “take an approach to solving problems, designing systems and understanding human behavior. It draws on concepts fundamental to computing”\(^4\), and makes a great influence in the field of computer science and education. In the past few years, computational-thinking-oriented course has been wildly adopted in not only Chinese colleges and universities\(^5\)\(^6\) but also the K-12 education systems worldwide\(^7\).

However, it is not easy to assess the CT skills of a Scratch program, mainly because Scratch programs are developed and run within a graphical user interface(GUI). The researchers have made great efforts in this aspect. One of the most relevant tools is Dr. Scratch, which is a web tool to automatically assess CT skills of Scratch programs\(^8\). Dr. Scratch is powered by hairball which uses python package Kurt. The latter is a Python module used to read and write Scratch program files, and is based on Python built-in module Cpickle which can serialize Scratch JSON files to Python objects.

In practical application, Dr. Scratchs obsolete infrastructure based on Cpickle results in high failure rate and low efficiency. To solve these flaws in Dr. Scratch, we design a kind of Scratch Analysis Tool(SAT) based on language recognition tool ANTLR, which is a static analysis tool to assess CT skills of a Scratch program. Owing to the flexibility of LL(*) parsing strategy existed in ANTLR\(^9\), we choose the recognition tool ANTLR to implement this SAT in order to achieve the assignment of CT score robustly and effectively. And then, aiming to the scratch programs, we customize more than 200 lexical and syntax parser rules in ANTLR. Furthermore, some fundamental concepts in Computer Science, such as stack, queue and recursion method, are not involved in Dr. Scratch. Therefore, we expand the grading standard of assessing CT skills in Dr. Scratch. Experiment results show the performance (e.g. success rate, execution time) of SAT is superior to that of Dr. Scratch.

The remainder of this paper is organized as follows. Section II presents background and related work. Section III overviews the whole system. Section IV proposes the analysis methodol-
ogy. Experiments and comparison results are given in Section V. The paper concludes with section VI.

II. BACKGROUND AND RELATED WORK

Scratch, created by the Lifelong Kindergarten Group at the MIT Media Laboratory, is a media-rich software development environment for novices. Programs in Scratch are composed of scripts that are built by dragging and dropping blocks that represent program components, such as expressions, conditions, statements, and variables. This mode of interaction prevents the trivial syntax errors that can cause students’ frustration. Scratch programs consist of 2-dimensional interactive animations which are named sprites. They move on the screen as a result of user input or scripts in the program. Audio and video from a webcam can also be integrated into Scratch programs.

With the continuous development of Scratch, more and more researcher have begun to pay attention to analyze the Scratch program. However, some features of Scratch make it difficult to be analyzed. First, Scratch is designed to allow students to learn computer science programs. Meanwhile, it is added great creativity in their work. This creative freedom is one of the reasons that Scratch programs are challenging to be analyzed[10]. Second, block-based visual and auditory programming languages, in which the programs developed and run within a GUI, make evaluation and analysis more difficult than traditional text-based programming language. Third, unlike traditional text-based programming, its entrance is multiple, and twelve Hat blocks are designed to start a script. This makes it more difficult to search through the program.

To the best of our knowledge, more than four projects have been contributed to analysis of Scratch program. Burke and Kafai developed Scrape, which was a tool that can help humans to understand patterns across Scratch files including level of nesting, sum of loops, etc[11].

Boe et al. [10] developed Hairball, a plugin-able framework was useful for static analysis of Scratch programs based on python language. Hairball inspired by Lint, performed static analysis of a program to detect different kinds of problems, such as dead code, wrong attribute initialization or incorrect message synchronization. Open source Python package Kurt, which was used by hairball that provided simple access to all the elements contained within a Scratch program, i.e., the images, sounds, stages (backgrounds), sprites and most importantly the scripts.

Ota et al. [12] presented Ninja Code Village, which was a comprehensive learning-support environment for the Scratch. It provided more than 60 sample functions that were commonly used in Scratch programs. In addition, these functions were used in a program in order to foster students competencies in abstraction, modeling, and decomposition. It also provided automatic assessment of CT concepts such as conditional statements, loops, data, and parallelism in order to develop students programming skills.

Moreno-Len et al. [8] presented Dr. Scratch, a web tool to automatically assess CT skills of Scratch programs. Dr. Scratch assess CT skills of a Scratch program in seven dimensions: abstraction and problem decomposition, logical thinking, synchronization, parallelism, flow control, user interactivity and data representation. The scores in these seven dimensions are aggregated to get the final total scores (mastery scores). The score ranges from 0 to 21 points. On the basis of the hairball, Moreno-Len et al. developed the mastery plugin according to the grading standard.

III. SYSTEM OVERVIEW

A. System Module

ANTLR, ANother Tool for Language Recognition for short, is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It is widely used to build languages, tools, and frameworks[13]. ANTLR automatically generates a language recognizer according to a formal grammar file, which includes customized lexical rules and syntax parser rules representing how a grammar matches the input. ANTLR is able to generate Lexer and Parser code for several target languages including Java, C++, Python, Javascript, among others. Python is selected as our tool target language for its flexibility, convenience and wide-usage.

The Scratch File Format is the file format, which is used to encode Scratch programs when they are saved or downloaded. The Scratch File Format comprises a ZIP archive containing program information, which is encoded in a JSON file named project.json and program media in separate files. The program metadata, sprites, scripts, and the information of the program media are all stored in the project.json. Thus, parsing and recognizing information in project.json is the key to analyze a Scratch program.

In order to analyze Scratch programs and assess the programs CT skills by ANTLR properly, we custom and expand official JSON grammar file according to Scratch JSON code. After more than 200 lexical rules and syntax parser rules being customized by us, a one-to-one mapping from most visual blocks in Scratch to ANTLR parser rules is created. Unknown blocks or extended blocks won’t intercept the recognition, they will be recognized as a normal JSON grammar. Table I shows a part of visual blocks in Scratch and their syntax parser rules, which we customize. It should be pointed out that, in the ANTLR grammar, uppercase variables represent a lexical rule while lowercase variables represent a syntax parser rule.

We have developed a web tool, which allows users to analyze Scratch programs by uploading them. Figure 1 shows the system module of assess procedure. A Scratch program, which has been finished by dragging and dropping the visual blocks on the screen, can be uploaded to server on a web page. Then, the program would be input into the SAT, which has been generated according to the customized ANTLR grammar. Finally, after the analysis of Scratch program, the assessing score will be shown on a specific web page.

B. Assessing Flow

Figure 2 illustrates the inner assessing flow of the Scratch Analysis Tool using a small snippet of code, which is shown on the upper of the figure. First, character streams of JSON
TABLE I
A PART OF VISUAL BLOCK IN SCRATCH AND THEIR RULES IN ANTLR

<table>
<thead>
<tr>
<th>visual block</th>
<th>syntax parser rule in ANTLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ DOREPEAT_TOKEN , NUMBER , value</td>
<td>[ DOREPEAT_TOKEN , value , value , value ]</td>
</tr>
<tr>
<td>[ DOIF_TOKEN , value , value , value ]</td>
<td>[ [ FORWARD_TOKEN , NUMBER ]</td>
</tr>
</tbody>
</table>

are input into the Lexer in sequence. After that, the Lexer will break up the character streams into tokens by lexical rules. Then, the syntax parser feeds off the token streams and begins to recognize the sentence structure according to the syntax parser rules. After completing lexical analyzing and syntax parsing, an abstract syntax tree (AST), which is a highly processed and condensed tree structure, is created. Finally, the result will be generated after analyzing the AST.

IV. METHODOLOGY
A. Computational Thinking score assignment

Although researchers have proposed different grading standards to assess the CT skills, they seldom have paid attention to the combination of blocks and their meanings in the concept of computer science. As mentioned in the Dr. Scratch papers, the researchers were also uncertain about whether it was enough to confirm student fluency in certain CT concepts by using particular blocks or sum of some specific blocks merely. However, grading standard in other tools is insufficient. Thus, complementary score assignment is needed to be introduced into the grading standard.

Inspired by the work of Aman Yadav et al.[14], who presented stack and queue, which were the primary conceptions in data structure, is a key factor in CT education. On the basis of grading standard in Dr. Scratch, in our work, an extended grading standard is created. It mainly discusses the
TABLE II
Grading standard of assessing CT level in SAT

<table>
<thead>
<tr>
<th>CT dimension</th>
<th>Basic (1pt)</th>
<th>Developing (2pt)</th>
<th>Proficient (3pts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction and problem decomposition (A&amp;PD)</td>
<td>more than one script and more than one sprite</td>
<td>procedures (definition of own blocks)</td>
<td>use of clones, use of stack or queue</td>
</tr>
<tr>
<td>Parallelism</td>
<td>two scripts on green flag</td>
<td>two scripts on key pressed or on the same sprite clicked</td>
<td>two scripts on when I receive message, or video or input audio, or when backdrop changes to logic operations, recursion wait until, when backdrop changes to, broadcast and wait</td>
</tr>
<tr>
<td>Logical thinking (LT)</td>
<td>if</td>
<td>if else message broadcast, stop all, stop program</td>
<td>wait until, when backdrop changes to, broadcast and wait</td>
</tr>
<tr>
<td>Synchronization</td>
<td>wait</td>
<td>wait</td>
<td>repeat until, repeat until</td>
</tr>
<tr>
<td>Flow control (FC)</td>
<td>sequence of blocks</td>
<td>repeat forever, repeat until</td>
<td>wait</td>
</tr>
<tr>
<td>User interactivity (UI)</td>
<td>green flag</td>
<td>broadcast, stop all, stop program</td>
<td>wait</td>
</tr>
<tr>
<td>Data representation (DR)</td>
<td>modifiers of object properties</td>
<td>Variables</td>
<td>Lists</td>
</tr>
</tbody>
</table>

In order to elaborate the actions clearly, the algorithm judging whether having a recursion in a Scratch program is illustrated in Algorithm 1. Let \texttt{procedef}, \texttt{call}, and \texttt{root} denote the syntax parser rules of defining a procedure, calling a procedure and root of AST, respectively. Let variables \texttt{if\_use\_recursion}, \texttt{procname} and \texttt{callname} denote judging result, procedure name and procedure name of a call function, respectively.

Algorithm 1 Algorithm judging whether having a recursion in a Scratch program

**Input:** AST

**Output:** the result of whether use recursion in program

\begin{verbatim}
if \_use\_recursion
1: if enter a root rule then
2: if \_use\_recursion ← false
3: end if
4: if enter a procedef rule then
5: set procname according to value of children
6: end if
7: if enter a call rule then
8: set callname according to value of children
9: if callname = procname then
10: if \_use\_recursion ← true
11: end if
12: end if
13: if exit a procedef rule then
14: clean procname
15: end if
16: if exit a root rule then
17: return if \_use\_recursion
18: end if
\end{verbatim}

Figure 3 shows the blocks of a Scratch program, which accepts two numbers (e.g. “Please input first number”, “Please input second number”) and produces their greatest common divisor using Euclidean algorithm (e.g. “the greatest common divisor is”). In our new grading standard, the program can get 3 points in logical thinking line. However, the program only gets 1 point in Dr. Scratchs logical thinking line due to the use of \texttt{if} block in GCD procedure. It is obvious that logical thinking skills of this program have been underestimated in traditional Dr. Scratch.

B. Analysis of AST

Analyzing the AST is a key step to produce the final results. Since it is hard to manipulate and utilize tree structures directly, ANTLR can generate a tree walker iterating over the nodes in the AST via a depth-first walk named Listener. For ease of invocation, ANTLR also automatically creates an enter and exit method for each rule in the syntax parser grammar, ANTLR Listener will execute the related method during depth-first walk. Thus, we define actions in enter and exit methods of related syntax parser rules according to the grading standards. Based on the above, the analysis result will be concluded after ANTLR Listener is executed.

V. EXPERIMENT

In this section, the experimental environment is first described. Then, a case study for calculating the greatest common divisor of two numbers using Euclidean algorithm in Scratch is introduced in detail. At last, two groups of comparison experiments show the superiority of the designed SAT.

A. Experimental environment

Our experiments run on a computer with an Intel Sandy Bridge i7 CPU and 4GB memory. Legacy Python2.7 is se-

Fig. 3. Source code of a Scratch program aim to calculate the greatest common divisor of two numbers
lected as the Python interpreter for Python3 is not supported by Hairball.

To compare the SAT with Hairball Mastery plugin, we randomly downloaded 10,000 Scratch programs from a website named haohaodada\(^2\), which is a website widely used for studying, sharing, and exhibiting Scratch programs in primary school in Zhejiang Province and Beijing in China. All the categories of Scratch programs are included in these 10,000 programs, such as animations, art, music, games, stories, Arduino extension and so on.

B. Case study

In order to validate the correctness of SAT, we assess the program shown in Figure 3 by using SAT. As shown in Figure 4, the assessment results of SAT are as follows: the program doesn’t score for Parallelism and Synchronization for the reason that no relevant concepts are used. The program gets 1 point for FC dimension since only sequence structure is used here. Furthermore, the program gets 2 points for A&PD dimension because there is a procedure named GCD; UI dimension, as two numbers are asked to type; DR dimension, because of variables x and y. Finally, the program obtains 3 points for LT dimension in view of recursive procedure GCD. Therefore, the total score of the program is 10 points. SAT returns the correct result as expected.

C. Efficiency Comparison and Performance Analysis

First, we collect 10,000 programs, and compare the assessing efficiency of SAT and Hairball Mastery plugin. Obviously, SAT can assess all programs successfully while the hairball mastery plugin can assess 9173 programs of them. As shown in Figure 5, the success rate of the Hairball Mastery plugin is lower than our assessing tool SAT. Hence, compared with Hairball, SAT increases the success rate so as to improve the efficiency of program evaluation.

There are two main reasons of Hairball Mastery plugin’s failure. The first one is that system usually meets out of memory when assess a large Scratch program. The second one is that the Hairball Mastery plugin unable to dispose of extension blocks from third-party.

Figure 6 gives the comparison results on the median, interquartile range of the execution time for two different samples by using two kinds of tools. The outliers, which usually represent some extremely large and complex Scratch programs (e.g. super mario game, minecraft game), can be plotted as discrete points.

As shown in Figure 6, Q1 (the first quartile), median, and Q3 (the third quartile) of Hairball Mastery plugin for 9173 samples are 366ms, 429ms and 495ms respectively. Similarly, Q1, median and Q3 of SAT for 10,000 samples are less, and they are 60ms, 105ms and 233ms, respectively. As far as SAT for 9173 samples is concerned, we can find that its corresponding Q1, median and Q3 are 58ms, 95ms and 188ms, which are as same as those of hairball mastery plugin.

\(^2\)http://www.haohaodada.com/

Fig. 4. Screenshots and the analysis result of project which shown in Figure 3

Obviously, the performance of SAT is much better than that of hairball mastery plugin.

We further compare the mean execution time for two different samples by using two kinds of tools, which are 512ms, 502ms and 352ms, respectively. The results show that SAT has a large performance advantage comparing with hairball mastery plugin.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we design and implement a new Scratch Project analysis tool to assess CT skills based on ANTLR. The tool, for the first time, takes basic concepts of data structure and algorithm into account when assessing the CT skills of a Scratch program. Benefit from robustness and performance of ANTLR, experimental results show that this tool is superior to other tools in terms of success rate and other performance metrics.
The biggest limitation of the tool is that some key CT competences cannot be measured by analyzing the static codes of a program, such as the designing, debugging or remixing skills. Therefore, in the near future, we plan to carry out new research and approach to record and receive feedback of student action by customizing Scratch program.

ACKNOWLEDGMENT

This work is partly supported by the National Natural Science Foundation of China under Grant 61672109, 61772085, 61532012, 61370196.

REFERENCES


