A Study of Students' Learning of Computing through an LP-Based Integrated Curriculum for Middle Schools

Joshua Archer[1], Rory Eckel[1], Joshua Hawkins[1], Jianlan Wang[1], Darrel Musslewhite[2], Yuanlin Zhang[1],

[1]Texas Tech University,
[2]Laura Bush Middle School
josh.archer@ttu.edu, rory.eckel@gmail.com, joshhawkins44@gmail.com,
Jianlan.Wang@ttu.edu, D.mussle@yahoo.com, y.zhang@ttu.edu

Abstract

There has been a consensus on integrating computing into the teaching and learning of STEM (Science, Technology, Engineering and Math) subjects in K-12 (Kindergarten to 12th grade in the US education system). However, rigorous study on the impact of an integrated curriculum on students' learning in computing and/or the STEM subject(s) is still rare. In this paper, we report our research on how well an integrated curriculum helps middle school students learn computing (abstraction and programming) through the microgenetic analysis methods.

Introduction

Since the birth of the first electronic computers, computing has not only developed into a new and independent discipline, but it has also been integral to the practice of all STEM disciplines in today's society. As a result, Computing has been adopted as a key component of K-12 education (e.g., (K-12 Computer Science Framework Steering Committee 2016)), and there is also consensus on the need for integrating the teaching and learning of computing with that of STEM subjects in K-12 (e.g., (STEM education act 2015; NSF 2018b; Committee on STEM Education, National Science & Technology Council, the White House 2018)). There have been numerous projects integrating Computing and STEM learning (e.g., (NSF 2018a; Lee et al. 2020; Wang, Shen, and Chao 2021)).

However, it is still very challenging to study the impact of the integrated curriculum on students learning in computing and/or STEM subject(s). A critical source of the challenge is the operationalization of computing, also called *computational thinking* (CT) (Wing 2006) by the education community. As pointed in a recent review article by Wang, Shen and Chao (2021), there has been little consensus on how computing should be operationalized in education, and it is even harder to operationalize computing in the context of learning STEM subjects.

Recently, Zhang et al. (2019a) have proposed a Logic Programming (LP) based framework for integrating the teaching and learning of computing and STEM subjects, called *LPK12*. The base of the framework is the definition of *abstraction* using LP (Gelfond and Kahl 2014). Abstraction is well accepted as a foundation of Computing (K-12 Computer Science Framework Steering Committee 2016). This definition allows a deep and natural integration of Computing and STEM subjects. Salient features of LPK12 are: 1) LP, as a programming language, has a low floor and high ceiling (i.e., easy to learn and yet can be used to develop abstraction (or model) for "real" problems), desirable for K-12 classes (Papert 1980); 2) LP facilitates, through *Logic*, a unified treatment of some fundamental skills and topics in STEM and Computing; 3) Children at age 11 to 15 demonstrate substantial knowledge of natural language and the logical use of symbols related to abstract concepts by Piaget (Piaget 1972) and Vygotsky (Vygotsky and Vygotski 1987); and 4) It facilitates the alignment of the LPK12 curriculum to the standards of Computing and STEM learning (National Research Council 2012; NGSS Lead States 2013; Zhang et al. 2019a).

The operationalizable definition of abstraction by LPK12 and its explicit connection to other aspects of computing such as programming and science allow rigorous study of the impact of an LPK12 based integrated curriculum on students' learning of computing and science. LPK12 offers text based abstraction and programming which allows a natural connection to the authentic computing practices (Sengupta et al. 2015). The study of abstraction in the context of visual languages, such as (e.g., Scratch (Resnick et al. 2009)), has been very challenging (Guzdial 2004; Lye and Koh 2014; Aivaloglou and Hermans 2016) even in computing education only (not to mention in an integrated curriculum). In contrast, the text based approach by LPK12 provides rich opportunities for students to learn abstraction. However, the research by Zhang et al. (2019a) only shows a feasibility of the LPK12 approach.

In the research reported in this paper, we will study how well students can learn said text based abstraction and programming through an integrated curriculum. We note that abstraction and programming are considered as two core components of computing (K-12 Computer Science Framework Steering Committee 2016). In our study, we consider various levels of abstraction and programming. Our goal is to have a deeper understanding of how well students learn abstraction and programming. Since little is known about middle school students' learning of abstraction and programming through an integrated curriculum, we employ the

microgenetic method which is often used in the study of developmental psychology (Siegler and Crowley 1991) and recently in studying students' learning of science through games (Sengupta, Krinks, and Clark 2015).

Our data and analysis show a profile of students' learning of abstraction and programming under LPK12 framework. Students do well at basic level of abstraction, such as variables (as an important abstraction element). For example, they do well in representing knowledge as *LP facts*, asking questions using *LP queries* (with or without variables). They also do well with reasoning, and writing programs based on said abstraction. This evidence is important for raising the attention of researchers and practitioners to the potential of text based abstraction and programming in an integrated curriculum (recall a significant weakness of the dominant visual language based approach is the lack of rigorous evidence on students' learning of abstraction). Thanks to the richness of abstraction in LPK12, we are also able to identify the limitation of students' understanding of abstraction under an intervention with limited time span.

The rest of the paper is organized as follows: the background and related work, the study design, and then a report on the experiment results. This will be followed by the discussion and conclusion sections.

Background

There is abundant work on CT and integration of Computing and STEM (Lee et al. 2020). A large majority of the work on the integration of Computing and STEM is based on visual programming environments such as Scratch and Alice (Kelleher and Pausch 2005). Little research is reported on students' learning of abstraction and text based programming at the middle-school level (Statter and Armoni 2020). A major contribution of this paper is to understand students' learning of abstraction and programming under LPK12 framework. This contribution is enabled by the explicit and systematic treatment of abstraction under LPK12.

The LPK12 framework for abstraction consists of a **problem**, **abstraction methodology** and representing the abstraction using **Logic Programming** (as a computer model solving the given problem).

Problem. The key component of the problem is explicit **questions**. Chemical elements from the periodic table are chosen as the problem's domain. For example, what is the *atomic number* of Hydrogen? what is the number of protons of a Hydrogen atom? Students are expected to answer these questions and understand *why* their answers are correct.

Abstraction. The students are asked to abstract needed knowledge for said questions. The questions keep our abstraction relevant, and it will follow the following methodology:

1. Identify *objects* and *relations* in the problem. For example, *objects* here are Hydrogen and its particle numbers. The *relations* will be the relation between an element and its respective atomic/proton number.
2. Identify *knowledge* about the problem and the relations. For example, there is knowledge between the proton number relation and the atomic number relation for an chemical element. (They are the same number for a given element)

Representing Abstraction Using LP. LP provides not only a precise language for representing the abstraction, but also automated reasoning with the abstraction (to solve intended problems). The resulting representation of an abstraction is called a *program* or a *computer model*. A LP program consists of three sections: *sorts* section declares the objects and their kinds in the problem, *predicates* section declares the relations among objects and *rules* section consists of knowledge about relations. The representation of the abstraction for chemical element above is as follows:

```
sorts
1.  #elements = {hydrogen}.
2.  #numbers = 1..200.
predicates
3.  atomicNum(#elements, #numbers).
4.  protonNum(#elements, #numbers).
rules
5.  atomicNum(hydrogen, 1).
6.  protonNum(E, X) :-
        atomicNum(E, X).
```

In line 1, the kind name *#elements* is given: and numbers for chemical elements. In Line 3, *atomicNum* is the relation name which is declared as a relation between a element and a number. Particularly, *atomicNum(X, Y)* means that X is the atomic number of Y. Statements such as those in line 5 and 6 are called *rules*. The statement in line 5 is a special case of a *rule* called a *fact*. Line 5 represents the relation that the Hydrogen element has an atomic number of 1, and depends on no other knowledge. Line 6 represents the relationship between atomic number and proton number of an element. The relation *protonNum(X, Y)* means that the proton number of element X is Y. The symbol `:-` in the rule is read as "if." The rule is read as for every E and X, the proton number of element E is X **if** the atomic number of element E is X. Note that E and X here are variables that refer to any element and any number. Following line 5 and 6, the syllogism tells us that the proton number of Hydrogen is 1.

Once the LP program is in place, the LP system can answer questions using the program. To answer "What is the atomic number of Hydrogen," a variable is introduced X meaning the atomic number of Hydrogen. Then a *query atomicNum(hydrogen, X)* can be written which means *find the value of variable X such that its the atomic number of Hydrogen*. Note this type of variable is different from the variables in a rule (see the previous paragraph).

There is an online LP programming environment *onlineSPARC* (Marcopoulos and Zhang 2019) which allows a user to easily type in the LP program. It also produces answers (automatically using the LP program) to queries posted by a user. More details can be found in (Zhang et al. 2019b).

Related Work. Abstraction is taken more as an umbrella term in the literature, e.g., in K-12 Computer Science Framework (2017) and in CSTA (Computer Science Teachers Association) (2017) standards. For measuring the learning outcomes of abstraction, there is a lack of an "operational" definition of abstraction. LPK12 framework adopts an explicit

characterization of abstractions here. I.e., having questions first, then identifying objects, their relations and the relationship among the relations. Explicit and general arguments can be made for this understanding to support the standards for abstraction in (CSTA 2017). Firstly, objects and relations are at the core of (set theory of) discrete mathematics; which is universally recognized as a fundamental course for computer science. Secondly, function writing is listed as a standard in (CSTA 2017). Functions (also called procedures) are well recognized as a vehicle for representing abstraction (e.g., (Scott 2000)). It is well known that functions are a special form of relations. Finally, abstraction as procedure (or relations) is not commonly used or studied in the existing work (e.g., (Aivaloglou and Hermans 2016)). In fact, the rigorous evaluation of learning outcomes on abstraction and programming (text based) has been lacking in the Computing education research (e.g., (Guzdial 2004; Lye and Koh 2014; Wang, Shen, and Chao 2021)).

Study Design

The research question is: *how well do students learn abstraction and programming (i.e. Computing) with a LP based curriculum integrating Computing and Science for middle schools?*

Abstraction is not a simple process, and the development and learning of it may take a long time. Also the LP language and the integrated curriculum are new at the middle school level, and very little is known about the impact of these factors on student learning. Consequently, the *microgenetic analysis* method was employed (Siegler and Crowley 1991). We will elaborate on the integrated curriculum, implementation, the microgenetic method and the data analysis below.

LP Based Integrated Curriculum

In this research, we use a revised module for integrating Computing and Science (Chemistry) originally developed in (Zhang et al. 2019b). This module consists of eight lessons. Lesson 1 consists of a video motivating students on learning computer science and an introduction of a computer model for solve some familiar daily life problems.

Lesson 2 introduces *relations*, and *facts* using examples. The *family* domain was deliberately chosen so as to provide some familiarity for the students from Lesson 1 to Lesson 2. Later on, other domains are explored.

Lesson 3 stays with the *family* domain, and introduces the idea of *variables* and *queries* to the students. After showing students explicit examples of using variables and writing queries for one type of relations, we let students practice on different types of relations.

Lesson 4 continues to remain with the *familial* domain to introduce *rules*. We discuss the difference between variables used in queries and rules. A variable in a query refers to something unknown that we want to know. But a variable in a rule refers to any object.

Lesson 5 switches the problem domain to the *periodic table of elements* and *chemistry*. With that domain switch, all of the prior concepts (variables, relations, queries, facts) are tested under the entirely new relation between an element and its chemical symbol.

In lesson 6, we repeat the same concepts in lesson 5 with a new relation: the atomic number of an element.

Lesson 7 covers the relationship between the atomic number of an element and its proton number as discussed in the background section. rules are first introduced to represent that relationship. The abstraction level of rules are clearly higher than that of facts.

Finally, in Lesson 8; we introduce new chemistry concept *mass number* and its relationship with the number of protons and neutrons of an element. One piece of chemistry knowledge to get neutron number of an element from its mass and proton number in English is "N is the number of neutrons of an atom E if M is the mass number of the atom E, and P is the number of the protons of atom E, and N = M - P." It is represented by the rule

```
neutronNum(E, N) :- massNum(E, M),
        protonNum(E, P), N = M - P.
```

where "," between the relations to the right of `:-` means *conjunction/and*, $neutronNum(E, N)$ means that the number of neutrons of element E is N, and $massNum(E, M)$ means that the mass number of the element E is M. The students extend a given model with this rule, and then they will write a rule for obtaining the mass number of an element using its number of neutrons and protons. Prior topics of learning are flexed in lesson 8.

Implementation

The module is implemented in a STEM elective course in a middle school in USA. The participants were one teacher and their four sections of 6th-graders, five sections of 7th-graders, and four sections of 8th-graders. The total number of students was 317. The teacher went through a one-week PD in 2019 Summer. The module was planned for 8 weeks with two 50-minute sessions per week. The first week was in person while the rest was online due to the COVID-19 pandemic. The senior author (as a faculty) meets with the teacher every week to go through the materials for the week. The teacher presents the content and the way to teach the materials, and the author gives feedback. For the online parts, the teacher makes 15 videos (covering Lesson 2 to 8) each of which is of length from 3 minutes to 15 minutes. Each lesson has two videos except that Lesson 8 has three videos. Students are required to watch videos for one lesson per week from the 2nd to the 6th week and videos for Lesson 7 and 8 in the last week. The teacher sets up a two hour long office hour every week, and provides three quizzes. In addition, we conducted a post-test (in online form) of the computing content.

Microgenetic Method

Microgenetic method is employed in the study of developmental psychology. It has three key properties: a) Observations span the entire period from the beginning of the change to the time at which it reaches a relatively stable state. b) The density of observations is high relative to the rate of change of the phenomenon. c) Observed behavior is subjected to intensive trial-by-trial analysis, with the goal of inferring the

processes that give rise to both quantitative and qualitative aspects of change. (Siegler and Crowley 1991)

The key rationale for us to use microgenetic method is that the development of the abstraction and programming skills is a long process and it involves a large number of details. Here are the specifics about applying the microgenetic method in our research.

We conduct a one-on-one interview for each session in the following manner. By the schedule of the online teaching of the class (due to COVID-19 pandemic), every student is expected to spend two sessions (each of 50 minutes) to study the videos (made by the teacher) of the lessons. For each interviewee, we arrange two 50-minute sessions a week, coinciding with their study time. For each session, the student and the interviewer will meet using Zoom, and the student is expected to study by watching the video. During this process, the interviewer can ask probing questions to understand the students' understanding of the ongoing materials. Interviewer will also encourage the student to think aloud when solving a problem or answering a question from slides or the interviewer. The students can also ask questions, and interviewer can answer them in terms of the slides. The whole process is recorded.

Clearly we cannot interview all students. We sample the students using their performance as conceived by their teacher. Though some students dropped from the interview process, we have 9 students who completed the study. Among them, 3 are female, four 6th graders (one good, two average, one struggle), three 7th graders (all average) and three 8th graders (one good, one average and one struggle). The uneven of the number of students from different performance groups is because we have more average students and some students selected dropped out of the study.

We have three interviewers who are all undergraduate students. Before the implementation, they have studied all the teaching materials and carried out all related programming practices.

Before the interviews of each week, the senior author and the three reviewers meet to go through the lesson(s) to be studied this week. They set up written document on detailed learning outcomes and the corresponding questions (which are all short answer questions) inside the slides. We discuss possible ways students may answer the questions and possible probing questions (e.g., if student get the correct quickly, we may ask why or if student gets stuck, we may ask students to think aloud). During the interview, the interviewers will pay special attention when students work on those questions associated to the learning outcomes. Each interview is video recorded, in total there were 78 interviews in the experiment.

Data Analysis

The data collected include the post-test result and the recorded interviews. For the recorded interviews, we analyze them following the procedure below:

a) Transcribing each video. One author X leads the three interviewers to transcribe almost all video content (such as what was discussed in the teacher's video, what the student did and conversations between the student and the interviewer etc.) of the video. The author first gave a demo on how to transcribe a video. Then the whole team would independently transcribe a different video and then meet and discuss the differences in their respective transcriptions. After that each video will only be transcribed by one member.

b) Developing coding scheme. The scheme is essentially a grading rubric. For each question (planned or improvisational from interviewers), we label whether a student answer is correct, correct with guidance, or incorrect. Given improvisational questions, we also define a scheme for naming the questions so that there is no ambiguity between possible interviewers, questions, and students that could cause a conflict. The scheme is continuously revised as the team gaining more experience in coding the scripts. The coding scheme also asks each coder to label any interesting discoveries (e.g., reasoning process with LP based abstraction) which are not covered by the explicit learning outcomes.

c) Developing coding. The three interviewers complete the coding of questions. Both author X and the senior author are deeply involved in the coding process. At the very beginning, the whole team coded the same script and then meet together to review all the differences between members and getting consensus. The coding scheme will then be revised as needed. Team will work on more scripts until it feels that convergence of members' coding. Then, each video script will be coded at least by two team members who will then try to get consensus on the coding differences. Unresolved differences will be discussed in team meeting, which may lead to the revision of the coding scheme.

d) Aggregating the coding. The learning outcomes of all lessons form three groups: d.1) the **problems domain** including the daily life problem such as problems involving family in early lessons and chemistry problems in later ones. Students are expected to be able to answer questions either using their experience or what they learned from the class. d.2) The **abstraction**. *English based abstraction*: the students are expected to use English to describe questions and/or the knowledge needed to answer the questions. *Formal abstraction* using LP language: the students are expected to use the LP language to represent *variables*, questions as *queries* and knowledge as *facts* or *rules*. In this study, we always provide relation design and students' main tasks are to write facts and rules based on these relations. (In contrast, a more advanced activity is for students to design relations and then represent knowledge using them.) In designing the module, we intentionally add the following layers (across lessons) to formal abstraction using *facts*: from a daily problem domain on relationship among family members to the chemistry problem domain. We have two classes of questions in each domain: questions about *relations* and *facts* already discussed in the lecture, and questions about new relations and new facts (to check if students understand in a new "context"). d.3) *Programming*. Since abstraction is made explicit in our module, the *programming* section has a narrower scope than a general use of programming. It consists of three components: the *software use* (denoted by label PU) such as the onlineSPARC, creating file, save file, copy and paste etc., the *syntax* of the programs, and the *debugging*

	Par	FA	CS	AN	AN*	PN*
Correct	28	22	71	59	5	9
Guidance	2	0	6	1	2	12
Incorrect	0	0	2	0	2	5

Table 1: Student performance on *facts*

of the programs. Details are in the results section.

	AL-father	AL-dad	AL-Query*	AL-Query*	AL-Query*	AL-Query*
Students	5	4	8	9	9	4
Correct	4	4	8	30	28	12
Guidance	0	0	0	0	3	0
Incorrect	2	1	5	0	2	0

Table 2: Student performance on *queries*

Results

Results from Microgenetic Analysis

Facts We will define the learning outcomes before we show the performance of the interviewees on them. In L2 (lesson 2 - family domain), we use **Par** to denote if the student can abstract and write facts about the *parent relation*; **Fam** to denote if the student can abstract in English, or write facts about new *familial* relations.

In L5 (chemistry domain + facts), AE-CS (apply) denote if the student can abstract in English about various chemical symbols; AL-CS (apply) denote if the student can represent chemical symbols using facts.

In L6 (chemistry domain + facts), AE-AN* denotes if students are able to articulate knowledge about the atomic number (AN) for an element in English; AL-AN* denote if students are able to write the fact on atomic number (AN) of other elements.

In L7 (chemistry and non-fact rules), AE-AN* denotes if students are able to articulate knowledge about the atomic number (AN) for an element; AE-PN* denotes if students are able to articulate knowledge about the proton number (PN) for an element (not avail); AL-AN* denotes if students are able to write the fact on atomic number (AN) of other elements; and AL-PN* denotes if students are able to write the fact on proton number (PN) of other elements.

For each label there could be several question instances. For example, for AE-CS (apply), there could be two question instances: *Can you write a fact for "the chemical symbol for carbon is C"?*; *Can you repeat what you did for oxygen?*. In Table **??**, the columns are named by the labels. Row 1 is the number of students with a recorded response. Row 2 is the number of correct responses for all instance questions corresponding to the labels. Row 3 is the number of responses that were correct with clarification from the interviewer. Row 4 is the number of incorrect responses.

From the table, we can see that students are doing well on representing knowledge in different domains when the given piece of knowledge can be represented by facts. But when the knowledge becomes more complex which involves more than one relations (see columns after AE-AN*), it is harder for students to represent each of these relations. For example, let us look at AE-PN* and AL-PN*, the questions that students seemed to struggle with most. For AE-PN*, the student would need to explain "N is the number of neutrons of an atom E if M is the mass number of the atom E, and P is the number of the protons of atom E, and N = M - P." This is no easy task. The intended answer to AL-PN* (the logical counterpart to the previous question) is `protonNum(P, E)`), which students struggle with because it is the first time they have to create such a construct completely from scratch.

Queries For queries, we have the following learning outcomes. In L3 (family), we have *AL-father* denoting if students are able to formulate a query to answer "Who is the father of Peter?; AL-dad denotes if students are able to formulate a query to answer "Who is the dad of Peter?"; In L4, AL-Query* denotes if students can formulate queries using variables for relations dad, father, mother, and mom. In L5, AL-Query* denotes if the students can formulate queries using variables for relation chemical symbol. In L7, AL-Query* denotes if students can formulate queries using variables for relations atomic number and proton number. In L8, AL-Query* denotes if students can formulate queries using variables for relation mass number.

Rules For rules, In L4, AE-mom denotes if students are able to articulate in English the knowledge defining *mom*, AE-parent denotes if students are able to articulate in English the knowledge defining *parent*, AL-mom denotes if students are able to write the *rule* for mom, AL-parent denotes if students are able to write the *rule* for parent, AE-dad denotes if students are able to *articulate in English* the knowledge defining *dad*, AL-dad denotes if students are able to write the *rule* for *dad*, AL-rule-R denotes if students are able to *read* a *rule*, and AL-Var-apply denotes if students are able understand variables in a rule correctly.

In L7, AL-rule-ANtoPN-R denotes if students are able to *read* the rule relating atomic number to proton number, AL-rule-PN denotes if students are able to *write* the rule for proton number, AL-rule-AN denotes if students are able to *write* the *rule* for atomic number.

In L8, AE-Neu denotes if students are able to articulate the knowledge for neutron number from mass number and proton number, AL-Neu denotes if students are able to write the rule for neutron number from mass number and proton number, AL-Neu-R denotes if students are able to to read the rule for neutron number, AE-Mass denotes if students are able to articulate the knowledge for mass mass number from proton number and neutron number, AL-Mass denotes if students are able to to read the rule for mass number.

From what we can see in the table, we have fewer students to respond with fewer correct answers, which means most students are not doing well. But we do observe their understanding of variables and ability to read rules correctly. In future, more exercises and instruction are needed for stu-

	AE-mom	AE-parent	AL-mom	AL-parent	AE-dad	AL-dad	AL-rule-R	AL-var*	AL-rule*	AE-Neu	AL-Neu	AL-Neu-R	AE-Mass	AL-Mass
Students	4	2	4	2	2	1	4	9	3	1	2	2	5	8
Correct	2	2	4	1	2	0	7	2	3	1	0	3	0	2
Guidance	2	0	0	1	0	1	1	1	0	0	2	1	5	8
Incorrect	0	0	0	0	0	1	2	2	2	0	0	0	0	1

Table 3: Student performance on *rules*

	L2	L4	L5
PU	2/2/0/0	4/5/0/0	9/15/0/0
PS-comment	2/5/1/0	5/11/3/0	9/40/2/0
PS-syntax	1/0/1/0	n/a	4/6/0/0
PS-rule	2/5/1/0	5/9/5/0	9/36/6/0
PD-test	1/3/0/0	3/4/2/0	9/12/14/0
PD-error	n/a	n/a	1/3/0/0

	L6	L7	L8
PU	8/5/3/1	6/10/0/0	n/a
PS-comment	9/31/1/0	n/a	7/23/3/1
PS-syntax	1/2/0/0	2/1/3/0	1/1/0/0
PS-rule	9/30/3/0	7/14/4/0	7/20/6/2
PD-test	9/16/14/0	6/2/7/0	8/5/28/0
PD-error	n/a	3/2/2/0	1/2/0/0
PD-debug	2/2/3/0	6/4/11/0	8/3/10/0

Table 4: Results on programming

dents to write rules. Problem decomposition methods may be used to help students to identify the components in the English description of the knowledge. The schedule is a bit rushed by assigning only two sessions for L7 and L8.

Programming For programming, the *syntax* includes the correct use of comments (denoted by PS-comment) and correct usage of syntax (PS-syntax) for facts and rules. *Debugging* includes the awareness of the use of queries to *test* a program (denoted by PD-test), the ability to note an error resulted from the discrepancy between the LP answer and the expected (predicted) answer (PD-error), and the ability to locate errors if there is any (denoted by PD-debug). Note we do not include L3 in the table because the nature of this lesson is about queries, and there is no programming.

In Table 4, for a given question asked (the row title) during a given lesson (the column title) the entry consists of 4 numbers separated by a slash *S/C/G/I*. *S* is the number of students who was asked this question, *C* is the number of students able to correctly answer this question, *G* is the number of students able to answer correctly after receiving guidance, and *I* is the number of students unable to correctly answer

For instance; *(3/2/0/1)* would indicate that *out of 3 students, two correctly answered and one incorrectly answered the question*. Again, note that it is possible for there to be more than one instance of a particular label per student. From Table 4, we can see that students are doing well on using the software, on the syntax and testing their rules. But we do not observe much about students' awareness of errors and debugging opportunities.

Case Study - *Reasoning* Finally, from the transcripts, we discover some interesting observations. Here we give a few examples. A less emphasized element in both the lesson design and the interview protocol design is the lack of *reasoning*. A surprise discovery is the support of LP based approach on students' reasoning capability. Specifically, students do well knowing when to predict the answer to a query using an LP program (in contrast to their own prior knowledge). Lesson 3 focuses on queries and enough questions were asked by interviewers for interviewees to predict answers to the queries. 6 students were asked to answer 20 questions. They are correct on 18 of them and correct under guidance on the rest.

Case Study - *Variables* We pay special attention to variables because they are important concepts in both mathematics and computer science. In our teaching, we introduce one use of variables as follows: "A variable is used to refer to something: Sometimes, you use it to refer something you don't know but *you want to know*, e.g., variables in a query; Sometimes, you just use it to refer to something more general than a specific object, e.g., *a person* or *another person* in a piece of knowledge in English." The following excerpts is related to the second use of variable. The student is referred to as *student* and interviewer is referred to as *interviewer* . The transcript is italicized, and the comments are in regular font following the excerpt.

Interviewer instructs student to add the comment and rule relating [the relations of] *father and dad.*

The intended comment is: "% For every X and Y, X is the dad of Y if X is the father of Y." and the intended rule is:

```
dad(X, Y) :- father(X, Y).
```

Student [typing]:

```
% John is the dad of Peter.
dad(john, peter).
```

Note that *student* here only gives an instance which is resulted from their reasoning (because John is known to be the father of Peter, John is the dad of Peter). This representation is good but it is not as general as we would like.

Interviewer explains that they want student to write a rule and reminds them to say that [knowledge on the relationship between father and dad] *in English.*

Student typed "If you are" but then changed to "% if John is the dad of Peter he is the father of Peter." Interviewer: "we don't want to have to write this for every single child right? we want it to work for everything." Student: "so instead of writing Peter you would write Y which would be the variable for the child?"

Here, *Student* is able to use variable (Y) to refer to a general person (child).

Student [typing]: "If John is the dad of Y he is the father of Y." Interviewer: "also, if you can imagine we had more dads, we would also want this to work for every father and

Familiar			New		
fact	query	rule	fact	query	rule
52%	34 %	N.A.	50%	22%	16%

Table 5: Post test results

child combination, we don't only want it to work for John. Student [typing]: "if X is the dad of Y he is the father of Y."

Now *Student* is able to produce an English description where variables are explicit and the logic connective "if" is explicit. The translation of the English to a rule would be straightforward according the examples given in class.

This example demonstrates that students can achieve a high level of understanding of variables and rules.

Results from Post-test

The post-test consists of questions on facts, queries and rules. For each of them, we ask students to work on domain knowledge covered in class and in a newer, less familiar domain. The percentage in each entry (Table 5) is the average (across lessons) percentage of the students who give the correct answer for each of a set of questions. The post-test is used for qualitative analysis. In post-test, students have done well in writing facts both in familiar and new domains. Their performance is reasonable on queries on both familiar and new domains (see discussion section). Rules in a new domain is challenging for students. The distribution of the post test results agree with what we would expect from microgenetic analysis. Students in microgenetic analysis are doing better possibly because of two reasons. 1) The questions were done in the lecture video during an interview. 2) Thanks to interviews, interviewees won't skip videos and could pay more attention to each session, and get help from interviewers. In contrast, it may be challenging for students to study online by themselves without a rigorous time schedule.

Discussion

This research was significantly affected by the COVID-19 pandemic. The teacher had to make videos for online teaching, and it was the first time for the teacher to carry out online teaching. Similarly, it was the first time for the students to have an online learning asynchronously. There is not a mechanism to force students to study the videos. As a result, students in this experiment lack the opportunities for class discussions and hands-on experiences on programming. Even worse, due to time constraints, lessons 7 and 8 are rushed in one week. They focus mainly on complex rules (a higher level of abstraction). However, our post-test was designed to fully measure students' understanding of rules. These may be important factors affecting students' performance in both interviews and in the post-test. As we can see, the absolute performance of students is low in the post-test.

We did not get as much information from the interview as we planned due to the following reasons. The one on one interviews are both time-intensive and challenging. Although we have detailed discussions and documentation on what to do during the interview before each week, it is still challenging for the interviewers to face logistics; various situations during interview and the load of knowing the planned questions and asking improvisational questions. We have cases that some sessions have to be canceled due to schedule changes from students, interviewers forgot to record a session, planned questions are not fully covered and etc. We have also weekly team meeting for interviewers to report and discuss issues they came across. One weakness of the team meeting is that students might not be able to be aware of their own major problems. For example, only at the later stage of the study, did we find that one interviewer focused only on a narrow set of reasoning questions across his/her interviews while missing most planned questions.

Although we did not get the data as we planned, thanks to the large number of interviews and recordings, we are still able to obtain data that is reasonable for a preliminary study. However, for future larger scale studies, we will leverage on our experience obtained in this experiment to improve data quality. For example, for future interviews, we will set up more strict procedures, produce a handy checklist for a set of questions an interviewer has to cover, and conduct timely review of the recordings for each interviewer.

Conclusion

There are few rigorous empirical evidences on students learning of computing, particularly abstraction, through a computing curriculum. They are even less in the context of text based abstraction and programming and integrated curriculum. Thanks to the characterization of abstraction in the LPK12 framework by Zhang et al. (2019a), we are able to carry out an intensive study of the impact of an LPK12 based integrated curriculum on students' learning of abstraction and programming – two core components of computing. Our data shows strong evidence that students are doing very well in abstraction at the level of *facts*, *variables*, *queries*, and *reasoning*. It indicates that middle school students are likely developmentally ready for learning text based and higher level abstraction. As for programming, students show fluency in using the software (agreeing with the facts that a lot of students are using software for many tasks including writing documents and playing games), in syntax of LP programs and in using queries to test their programs. In both our interviews and post-tests, we did find that complex rules, higher level of abstraction, pose challenges to students. We expect more content and pedagogical support, practices, and longer time may be needed for students to master higher level of abstraction. More rigorous research is expected to understand how best to support students' development of higher level of abstraction.

Acknowledgments

We thank Pratim Sengupta for suggesting the microgenetic analysis method to us, Arthur Jones for conducting interviews, Jikui Zhao for coding, and Edna Parr and Jeremy Wagner for their support in our implementation. This research is partially supported by National Science Foundation (NSF) grant DRL-1901704.

References

Aivaloglou, E.; and Hermans, F. 2016. How kids code and how we know: An exploratory study on the Scratch repository. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*, 53–61.

Committee on STEM Education, National Science & Technology Council, the White House. 2018. Charting a course for success: America's strategy for STEM education. https://www.whitehouse.gov/wp-content/uploads/2018/12/STEM-Education-Strategic-Plan-2018.pdf. Accessed: 2022-12-06.

CSTA. 2017. CSTA K-12 computer science standards. https://csteachers.org/page/standards. Accessed: 2022-12-06.

Gelfond, M.; and Kahl, Y. 2014. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents*. Cambridge University Press.

Guzdial, M. 2004. Programming environments for novices. *Computer science education research*, 2004: 127–154.

K-12 Computer Science Framework Steering Committee. 2016. K-12 Computer Science Framework. http://www.k12cs.org. Accessed: 2022-12-06.

Kelleher, C.; and Pausch, R. 2005. Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Comput. Surv.*, 37(2): 83–137.

Lee, I.; Grover, S.; Martin, F.; Pillai, S.; and Malyn-Smith, J. 2020. Computational thinking from a disciplinary perspective: Integrating computational thinking in K-12 science, technology, engineering, and mathematics education. *Journal of Science Education and Technology*, 29(1): 1–8.

Lye, S. Y.; and Koh, J. H. L. 2014. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41: 51–61.

Marcopoulos, E.; and Zhang, Y. 2019. onlineSPARC: A Programming Environment for Answer Set Programming. *Theory Pract. Log. Program.*, 19(2): 262–289.

National Research Council. 2012. *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. National Academies Press.

NGSS Lead States. 2013. Next generation science standards: for states, by states. https://www.nextgenscience.org/get-to-know. Accessed: 2022-12-06.

NSF. 2018a. STEM+C PI Summit. http://stemcsummit.edc.org/. Accessed: 2022-12-06.

NSF. 2018b. STEM+C Program. https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=505006. Accessed: 2022-12-06.

Papert, S. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Piaget, J. 1972. Intellectual evolution from adolescence to adulthood. *Human development*, 15(1): 1–12.

Resnick, M.; Maloney, J.; Monroy-Hernández, A.; Rusk, N.; Eastmond, E.; Brennan, K.; Millner, A.; Rosenbaum, E.; Silver, J.; Silverman, B.; et al. 2009. Scratch: programming for all. *Communications of the ACM*, 52(11): 60–67.

Scott, M. L. 2000. *Programming language pragmatics*. Morgan Kaufmann.

Sengupta, P.; Dickes, A.; Farris, A. V.; Karan, A.; Martin, D.; and Wright, M. 2015. Programming in K-12 science classrooms. *Communications of the ACM*, 58(11): 33–35.

Sengupta, P.; Krinks, K. D.; and Clark, D. B. 2015. Learning to deflect: Conceptual change in physics during digital game play. *Journal of the Learning Sciences*, 24(4): 638–674.

Siegler, R. S.; and Crowley, K. 1991. The microgenetic method: A direct means for studying cognitive development. *American psychologist*, 46(6): 606.

Statter, D.; and Armoni, M. 2020. Teaching abstraction in computer science to 7th grade students. *ACM Transactions on Computing Education (TOCE)*, 20(1): 1–37.

STEM education act. 2015. Public Law No: 114-59.

Vygotsky, L. S.; and Vygotski, L. S. 1987. *The collected works of LS Vygotsky: Volume 1: Problems of general psychology, including the volume Thinking and Speech*, volume 1. Springer Science & Business Media.

Wang, C.; Shen, J.; and Chao, J. 2021. Integrating computational thinking in stem education: A literature review. *International Journal of Science and Mathematics Education*, 1–24.

Wing, J. M. 2006. Computational thinking. *Communications of the ACM*, 49(3): 33–35.

Zhang, Y.; Wang, J.; Bolduc, F.; and Murray, W. G. 2019a. LP based integration of computing and science education in middle schools. In *Proceedings of the ACM Conference on Global Computing Education*, 44–50.

Zhang, Y.; Wang, J.; Bolduc, F.; and Murray, W. G. 2019b. LP Based Integration of Computing and Science Education in Middle Schools. In Zhang, M.; Yang, B.; Cooper, S.; and Luxton-Reilly, A., eds., *Proceedings of the ACM Conference on Global Computing Education, CompEd 2019, Chengdu,Sichuan, China, May 17-19, 2019*, 44–50. ACM.