"Something that Happens Each Day" – Students' Explanations of What Algorithms Are

Martina Landman
TU Wien
Vienna, Austria
martina.landman@tuwien.ac.at

Tobias Kohn
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
tobias.kohn@kit.edu

ABSTRACT

Algorithm is a highly abstract concept that is difficult to define precisely, but has entered the public discourse. In addition to a procedural knowledge about algorithms, we also need to ensure pupils have a viable conceptual knowledge.

We have collected short explanations of what an algorithm is from 58 pupils in 6th grade and analysed their answers. While there is some understanding of algorithms as step-by-step procedures, we also found misconceptions such as a strong emphasis on repetition.

CCS CONCEPTS

• **Social and professional topics → K-12 education**.

KEYWORDS

algorithm understanding, K-12 education, student misconceptions

ACM Reference Format:
Martina Landman and Tobias Kohn. 2024. "Something that Happens Each Day" – Students' Explanations of What Algorithms Are. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649217.3653531

1 INTRODUCTION

The concept of *algorithms* is at the very core of computer science [12]. Yet, good definitions or explanations remain elusive to the extent where many courses introduce the concept by example and leave it at preliminary, often somewhat vague, working definitions. In the public discourse the term 'algorithm' also finds increasing popularity but with a somewhat different meaning [17]. This is reflected to a certain degree in our observations during outreach activities that many pupils answer that they know the term because of the '⟨insert-random-social-media-platform⟩ algorithm'.

Our K-12 outreach activities are based on CS Unplugged [3] and run with the aim of increasing the pupils' understanding of computational thinking and algorithmics. However, although various algorithms are at the core of the activities, the concept is usually not formally defined or introduced. Nonetheless, we often observe that some of our tutors choose to discuss the concept with the pupils. Still, particularly given the highly abstract nature of the algorithm-concept, it is not clear what notion of 'algorithm' pupils actually have in their minds.

During a two-day outreach activity we therefore ran a 'competition' asking the pupils to give their best shot at concisely explaining what an algorithm is. The students were allowed to both write or draw a picture on a small card. We then analysed these 58 explanations and definitions and present our findings in this paper. The study presents a first snapshot of what 6th grade pupils from a single school think what an 'algorithm' actually is. To the best of our knowledge it is one of the first studies to explicitly look at K-12 students' understanding of the algorithm-concept itself.

Our study was guided by the following two research questions:

RQ1 How do upper elementary school students describe and explain what an algorithm is?
RQ2 In what aspects do students' conceptions of what an algorithm is differ from established definitions?

Mostly in line with the canonical analogue of a 'cooking recipe' we found that students highlighted step-by-step execution as well as the fixed or predefined nature of an algorithm. Additionally, most students mentioned repetition as a key concept and showed some uncertainty concerning who would execute an algorithm. Even though many students also mentioned the problem-solving aspect of algorithms, few considered finiteness.

Given the prominent role of algorithms in computer science, we believe that our study contributes to a wider investigation of the relationship between conceptual and procedural knowledge of algorithms (i.e. being able to formulate what an algorithm is versus working with algorithms), as well as when and how the concept is best taught. After all, the recent rise of AI has led to a public awareness of and discourse about algorithms that clearly requires a thorough conceptual understanding of what an algorithm is, its possibilities and its limitations.

2 BACKGROUND

2.1 Related Work

The concepts of computation and algorithms clearly lie at the heart of computer science [12, 18]. In computer science education, there has recently been an increasingly strong emphasis on 'computational thinking' (CT) [24]—a term that has replaced the earlier 'algorithmic thinking' [9, 14] and is based on a broad view of computation that includes 'algorithms' as one of its key concepts [14, 21].

There has also been a growing awareness that the underlying model of computation is of paramount importance for a proper definition and conception of computation and algorithms [1, 13]. In the context of programming, this has been discussed for some time with the idea of the 'notional machine' [6, 7], although this

ITiCSE 2024, July 8–10, 2024, Milan, Italy

ACM ISBN 979-8-4007-0600-4/24/07.
https://doi.org/10.1145/3649217.3653531

concept has not been widely adopted or applied to computation and algorithms in general so far.

Correctly expressing or representing algorithms is a non-trivial task. Students generally seem to lack an understanding of algorithms [9]. With the difficulty of learning to program to begin with, it usually requires a lot of time and training before students are capable of implementing algorithms as computer programs. The CS Unplugged initiative therefore offers an alternative route without programming and where algorithms and computational thinking are at the centre of attention [3, 4, 23]. Alternatively, algorithm visualisation uses technological means to provide dynamic representations of algorithms so as to foster a better understanding [11, 22]. In contrast to these approaches, our study focuses on the abstract and general concept of algorithm rather than specific algorithms.

A study conducted with undergraduate computer science students in the Netherlands looked into the students' understanding of the concept of algorithm [19]. Their focus was on levels of abstraction used by the students with four proposed levels: at the *execution level* an algorithm is essentially the execution of a program, at the *program level* the algorithm is the process described by the program, at the *object level* the algorithm can be viewed as an object in its own right and at the *problem level* different algorithms can be used to describe problems and their intrinsic complexity. The study found an increase in the abstraction levels with years of study, but noted that the mean of the answers given settled around the program and object levels. During their study, the undergraduate students were also asked to give their definition of 'algorithm'. However, the study does not go into details of the students' answers.

The previously mentioned paper on students' understanding of algorithms [19] also notes that many students have difficulties clearly expressing their thoughts, limiting the reliability of answers collected from students. On the other hand, we also have to be aware of the limitations of explanations given by instructors and what students understand thereof. Explanations are often based on metaphors and analogies—even more so for pupils in elementary/secondary schools—, which provide a powerful means of teaching, but may also easily lead to misconceptions [8]. In the answers students gave in our study we can very clearly recognise misconceptions that arise from the examples and metaphors provided by the instructors.

From a cognitive point of view we are more interested in the students' *conceptual* (and *factual*) *knowledge* than their *procedural knowledge* [2, 20]. Procedural knowledge is explicitly defined as "how to do something [...] and criteria for using skills, algorithms, techniques and methods" [2] whereas conceptual knowledge is about "the interrelationships among the basic elements within a larger structure that enable them to function together" and "knowledge of principles, generalizations, theories, models, and structures" [2]. Our focus on the conceptual understanding is therefore in contrast to most studies that look into specific algorithms and the students' abilities to apply them.

2.2 The Term 'Algorithm'

According to the Merriam-Webster dictionary an *algorithm* is "a step-by-step procedure for solving a problem or accomplishing some end" [16]. This description is quite general and can be interpreted and understood differently. In computer science, the notion of 'computation' plays an important role even in informal definitions, such as, e.g., an algorithm is "a series of elementary computation steps which, if carried out, will produce the desired output" [18] or "an algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output" [5]. The difference between the public perception and more mathematical definitions has been pointed out before [10, 17]. Given that the term is increasingly picked up by the media, particularly in the context of AI and marketing, it would not be surprising to find that students' notion of algorithms are strongly informed by media and public discourse rather than education.

For our study we follow Knuth's definition of what an algorithm is, since he managed to give a rather precise definition without referring to computational models such as Turing machines [12]: an algorithm is a sequence of operations for solving a specific type of problem that can be compared to terms like 'recipe' or 'process', but needs to exhibit five major 'features' [12]:

- *Finiteness.* The algorithm must always terminate after a finite number of steps.
- *Definiteness.* Each step of an algorithm must be precisely and unambiguously defined.
- *Input.* Algorithms have zero or more inputs provided before or during execution.
- *Output.* An Algorithm has one or more outputs that are related to the input in a specific way.
- *Effectiveness.* An algorithm's operations should be basic enough for manual execution in a finite time.

The point of 'effectiveness' is that each step in an algorithm must either be an algorithm itself or a simple computational step. The two items 'definiteness' and 'effectiveness' put hard limits to what an individual step of an algorithm might entail, which, of course, boils down the model of computation or notional machine.

3 METHODOLOGY

3.1 Data Collection

The faculty of computer science at our institution runs an outreach programme aimed at local schools. This includes a two-day event that is attended by all 6th grade students of an elementary school (i.e. approx. 150 pupils aged 11–12 years). The two days include various 'unplugged' activities on, e.g., artificial intelligence, sorting algorithms and data encoding. The activities are mostly run by undergraduate students who have received training in working with pupils through the materials, but are given the freedom of teaching according to their own style and preferences. Graduate students additionally oversee these workshops. The first author of this paper is responsible for the training of the tutors as well as the organisation of the workshops.

According to the national school curriculum, the pupils participating in this study should have a basic understanding of the concept of algorithms. This means they should be able to comprehend, execute and formulate independently clear instructions.

During the two days we run a competition among the participating pupils, asking for their definition or explanation of what

an 'algorithm' is. Each entry to the competition was to be written on a paper card (about half a page). The pupils were permitted to write or draw pictures, but were asked to come up with their own explanations. The involved graduate students then chose the 'best' answer(s) and handed out a prize to the respective students.

We received an anonymous copy of the students' answers after the winners had been chosen. That is, the authors of this paper set up the competition and received the compiled list of answers, but were not involved in running the competition itself. Our institution's review board approved the study.

There were a total of five drawings, which all illustrated parts of the written text (e.g., one drawing showed a number of playing cards where the text mentioned that algorithms are about sorting cards). We therefore ignored the drawings for our analysis.

Of the 70 answers we received, three were empty, one was obviously copied from the internet and eight were verbatim copies of other answers. Removing these duplicates and empty answers left us with 58 distinct answers (see Table 1).

3.2 Analysis

The collected data was analysed qualitatively based on Mayring's inductive category formation [15]. In this method, the codes and categories emerge during the sifting and analysis of the material.

The two authors reviewed the material individually over several rounds and discussed any differences until the two authors reached agreement on the coding categories and then classification/tagging of the data. These agreements included re-coding with each other's categories and finding matches and overlaps. Redundant codes were removed, highly overlapping codes were combined and (sub)categories were created. This also included a translation of the students' answers to English, followed by a fresh coding of the English translations and a comparison whether the coding remained stable. The final coding system is clustered into four groups, each with several sub-codes and/or subcategories (Figure 1):

- *Keywords.* Characterisations that use synonym-like descriptions or mention specific examples to explain the term algorithm, such as, e.g., 'a sequence of instructions'.
- *Properties.* Properties attributed to the algorithm, such as, e.g., 'fixed/predefined' or referring to a 'program'.
- *Executing Actor.* The entity that is executing the algorithm or acting it out, respectively, usually inferred, such as, e.g., 'an activity *you* do' tagged as 'human actor'.
- *Structure elements.* Description of structure such as, e.g., 'mandatory repetition' or 'stepwise execution'.

Note that some codes may appear in more than one category (e.g., 'sequence'), since they both act as a keyword and a structural description, say.

After coding the data material in a descriptive way, we evaluated and interpreted the data by counting occurrences, comparing potential semantic dependencies and looked at keywords.

4 RESULTS

We provide an overview of selected answers in Table 1, which are representative of the 58 different answers we received in total. Additionally, we highlight some of the results according to themes that emerged from the coding process in this section.

$\mathcal{A}_1$	Something is called an algorithm if it is a step by step sequence to, e.g., bake something, to solve a problem…
$\mathcal{A}_2$	An algorithm is a sequence of several steps and instructions, which are commonly aimed at a specific problem so as to solve it. Some commands may be repeated.
$\mathcal{A}_3$	An activity that you do time and again each day
$\mathcal{A}_8$	If a certain process regularly repeats itself
$\mathcal{A}_9$	An algorithm is a procedure to solve a problem. Ex.: sorting cards
$\mathcal{A}_{10}$	Executing commands (can be repeated)
$\mathcal{A}_{11}$	An algorithm is also used to solve problems and an algorithm is something that continuously repeats.
$\mathcal{A}_{14}$	A way to, e.g., solve problems, such as shuffling cards by colour or value
$\mathcal{A}_{15}$	A frequently recurring process
$\mathcal{A}_{17}$	Something recurring that repeats time and again. You always do it the same way or a procedure, respectively
$\mathcal{A}_{18}$	A sequence of instructions that are repeated
$\mathcal{A}_{21}$	Predefined procedure that solves problems or tasks one step at a time. Program code that continuously repeats itself
$\mathcal{A}_{22}$	An algorithm is a task that repeats itself
$\mathcal{A}_{26}$	An algorithm explains to a program what it should be doing and how it can do it.
$\mathcal{A}_{27}$	An algorithm is a procedure that solves problems or tasks in a stepwise manner.
$\mathcal{A}_{28}$	An algorithm is a fixed programmed system that is capable of solving specific problems and tasks.
$\mathcal{A}_{29}$	Something that is repeated time and again.
$\mathcal{A}_{30}$	Algorithm is a string of instructions that are executed to solve a task/problem.
$\mathcal{A}_{31}$	If something repeats, it will slowly turn into an algorithm. For instance traffic lights. Another great example is the lift in a tower block. For instance the lift waits at 7am on the ground floor close to the entrance (each day from Monday to Friday) and because this is happening an algorithm will form automatically.
$\mathcal{A}_{33}$	When something repeats at a specific time of the day. For instance if a lift is always used at 7am on the 4th floor, it will learn to wait at 7am on the 4th floor. A specific order, time and again.
$\mathcal{A}_{34}$	A sequence of commands and planned things.
$\mathcal{A}_{35}$	An algorithm is understood to be something that is programmed, a program that repeats
$\mathcal{A}_{36}$	For instance you go to bed every day, this is an algorithm (a repetition). An algorithm solves problems
$\mathcal{A}_{37}$	A recurrence of things in life
$\mathcal{A}_{39}$	A (repeated) sequence of instructions, e.g. go there, do that
$\mathcal{A}_{40}$	Recurring things that we do time and again such as, before going to bed, brushing our teeth and changing clothes. Also when sorting cards you have to figure out your own algorithm to sort the cards as fast as possible
$\mathcal{A}_{42}$	An exactly defined sequence
$\mathcal{A}_{45}$	A continuously recurring event
$\mathcal{A}_{47}$	If on a computer something repeats time and again
$\mathcal{A}_{50}$	A string of different commands that are repeated time and again. Clearly defined tasks or a specific process of tasks, respectively. For instance eating or recipe for cooking
$\mathcal{A}_{53}$	An algorithm is a continously recurring process in daily life, e.g., getting ready, brushing teeth, putting on pyjamas, going to bed
$\mathcal{A}_{55}$	A specific procedure to solve a problem/task
$\mathcal{A}_{56}$	a string of subsequent actions such as in a recipe for cooking. The actions are commonly done time and again and are never changed
$\mathcal{A}_{58}$	Algorithm is if something has an exact order that needs to be followed

Table 1: A selection of the answers given by the students

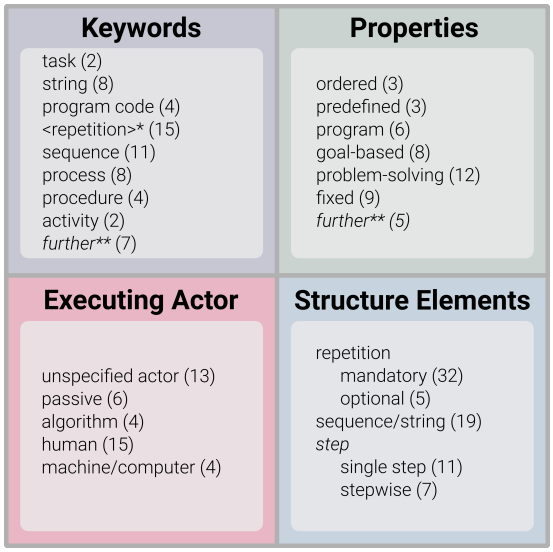

Figure 1: Overview of all code occurrences, clustered in four main categories.
*** paraphrased / ** codes that occurred only once**

4.1 Descriptions and Properties of Algorithms

Most answers specify the algorithm through a categorising keyword such as, e.g., $\mathcal{A}_2$ "An algorithm is a *sequence of several steps and instructions*, [...]" or $\mathcal{A}_{42}$ "an exactly defined *sequence*" (highlights added). As shown in Figure 1, we found 15 answers characterising algorithms as some repeating entity such as in, e.g., $\mathcal{A}_{11}$ "an algorithm [...] is *something that repeats*" (similarly $\mathcal{A}_{29}$), $\mathcal{A}_{36}$ "*a repetition*" and $\mathcal{A}_{37}$ "a *recurrence* of things in life". Another 11 answers characterised algorithms as sequences and eight as strings of something. Some characterisations, however, were more surprising, such as $\mathcal{A}_{22}$ "a *task* that repeats itself" or $\mathcal{A}_{45}$ "a recurring *event*".

Repetition. Repetition is perceived as a key characteristic of algorithms with almost two thirds of students explicitly mentioning repetition (37 out of 58). A majority of 32 students said that some form of repetition is a mandatory part of an algorithm whereas five students saw it as an optional possibility.

While the theme of repetition occurs in a majority of the answers given, there is less agreement on what exactly is repeated and in what manner. Compare, for instance, $\mathcal{A}_3$ "an activity that you do time and again each day", $\mathcal{A}_{18}$ "a sequence of instructions that are repeated", and $\mathcal{A}_{37}$ "a recurrence of things in life". In $\mathcal{A}_3$ we find a nested repetition, i.e. an activity that is repeated both during a single day and each day, bringing together the idea of *repetition* (as in $\mathcal{A}_{18}$) and *recurrence* (as in $\mathcal{A}_{37}$). Overall, three answers mention both as in $\mathcal{A}_3$, 23 answers mention repetition and 11 answers mention a recurrence.

With regards to *what* is repeated we find ten answers saying that "something" is repeated, six answers saying that "a process" is repeated and five answers each for "program code" and "commands" or "instructions", respectively. Other answers are even less specific such as, e.g., $\mathcal{A}_{37}$ speaking of "things". An answer that stands out is $\mathcal{A}_2$, which mentions "[...] Some commands may be repeated" as the only example where the repetition does not encompass the whole process or activity. In other words, repetition refers to a *loop* in only a single answer, although two other answers might also have loops in mind ($\mathcal{A}_{10}$ and $\mathcal{A}_{39}$).

The executing actor. As an 'executing actor' we refer to an instance that executes the algorithm's steps or instructions, if mentioned by the participants. Three quarters of the answers imply that there is an executing entity (42 out of the 56). Of these 42 responses that provide some evidence that the algorithm is 'executed' in some form or performed by something/someone, there were 15 answers that imply human actors, e.g., $\mathcal{A}_3$, $\mathcal{A}_{17}$ and $\mathcal{A}_{40}$. Four answers refer to machines or computers as actors of the algorithms (e.g., $\mathcal{A}_{47}$) and six indicate through passive voice that an algorithm is executed but do not indicate any actor (e.g., $\mathcal{A}_{30}$).

Four answers mention the algorithm or procedure itself as an executing actor (e.g., $\mathcal{A}_{21}$, $\mathcal{A}_{27}$ and $\mathcal{A}_{28}$). Answer $\mathcal{A}_{26}$ stands out in that it attributes a remarkable 'cognitive' ability to an algorithm as an entity that "explains to a program what it should be doing and how it can do it".

Finally, 13 answers imply that an algorithm requires some execution but do not mention this explicitly. $\mathcal{A}_2$ is an example for a required action (mentioning 'instructions' and 'problem-solving'), but not explicitly mentioning an actor.

Algorithmic attributes. With regards to RQ2 we especially looked out for attributes that are also found in formal definitions of algorithms, such as finiteness, stepwise execution, etc. Since none of the answers explicitly mentioned finiteness (or determinism), we used the tag 'goal-based' as a proxy to indicate that the algorithm would terminate and come to a conclusion.

Seven answers refer to a *stepwise* behaviour of the algorithm.

Eight answers were tagged as *goal-oriented*, as shown for example in answer $\mathcal{A}_{55}$ by the clear determination that a problem must be solved. In contrast, $\mathcal{A}_{14}$ mentions problem solving, but does not presuppose it as a mandatory criterion, since it only speaks of "e.g., solve problems". We therefore did not tag it as goal-oriented, but considered this problem-solving aspect more of an option than the purpose of the algorithm.

Seven answers mentioned the *fixed* nature of the algorithm. For instance, $\mathcal{A}_{55}$ talks of a "specific procedure" whereas $\mathcal{A}_{56}$ says that actions "are never changed". In contrast, $\mathcal{A}_7$ states "an algorithm is a program code than can be repeated forever. They continuously improve", explicitly contradicting the idea of a fixed procedure.

4.2 Origins and Purpose of Algorithms

Relatively few answers gave hints as to where algorithms come from or their specific purpose. However, we consider the views expressed by the pupils interesting enough to point them out in this section.

The aspect of solving problems and tasks. Twelve answers specify the purpose of algorithms as problem-solving. One answer $\mathcal{A}_2$ says that an algorithm is commonly used to solve a "specific problem", whereas six more answers say that an algorithm solves "a problem" and five say that an algorithm solves "problems". For instance, $\mathcal{A}_{55}$ says "a specific procedure to solve a problem/task". In contrast, $\mathcal{A}_{11}$ says "an algorithm is used to solve problems [...]" and $\mathcal{A}_{27}$ says "an

algorithm is a procedure that solves problems or tasks in a stepwise manner".

Of those five answers referring to algorithms as solving "problems", three also mentioned the requirement for repetition. Answer $\mathcal{A}_2$ mentions that some instructions might be repeated, whereas all other answers that mention the problem-solving aspect do not mention repetition at all.

Formation of an algorithm. $\mathcal{A}_{31}$ is remarkable in that this answer describes the genesis of how an algorithm forms: "if something repeats, it will slowly turn into an algorithm. [...] an algorithm will form automatically." A similar notion is expressed by eight other answers, including $\mathcal{A}_8$, $\mathcal{A}_{47}$ and $\mathcal{A}_{20}$ "if a specific process repeats itself regularly" or $\mathcal{A}_{15}$ "a frequently recurring process". An algorithm is therefore not something that is actively designed, but either forms like a tradition because of repetition or is a synonym for a repeated process, event or tradition.

Compare this with $\mathcal{A}_{21}$ and $\mathcal{A}_{42}$ whose mentioning of '(pre) defined' indicate an intentional design. $\mathcal{A}_{50}$ seems at first to also lean towards a formation by repetition, but then speaks of "clearly defined tasks". Given the two examples "eating" and "recipe" it is not clear whether this answer can be clearly coded as either one or the other; in fact, we assume that the student accepts both possibilities as legitimate process of how algorithms form.

5 DISCUSSION

5.1 The Role of Repetition

The prominence of repetition in the students' answers is remarkable. Even more so considering that, except for one case, all answers referred to a (necessary) repetition or recurrence of the algorithm itself rather than a looping structure within the algorithm. So, where does this come from?

We hypothesise that the instructors' original messages were emphasising that algorithms are fixed and static entities. That is, each time you execute an algorithm, you follow the same instructions in the same order. $\mathcal{A}_{50}$, for instance, starts with the concept of repetition, followed by a clarification "clearly defined tasks or a specific process of tasks" and the canonical example of a recipe for cooking. Likewise, $\mathcal{A}_{33}$ says "a specific order, time and again" and $\mathcal{A}_{17}$ clarifies the repetition by "you always do it the same way".

It is interesting to observe that the 13 answers directly mentioning the fixed or predefined nature of algorithms are dwarfed in number by the 37 answers referring to repetition. We could probably classify at least 31 of these answers as misconceptions in that they focused on the wrong aspect of the explanation, story or examples given. The explanations given by the tutors, and the analogues and metaphors employed in particular, clearly need to be revised as they seem unfit to solicit a correct understanding.

5.2 The Executing Actor

Concerning the 'executing actor' of the algorithm, there is some disagreement and insecurity among the students. For instance, $\mathcal{A}_1$ refers to problem-solving and baking, but the answer does not provide any information about *who* actually does the problem solving or the baking and executes all the steps mentioned. In contrast, $\mathcal{A}_{40}$ describes a human actor using the term "we" and $\mathcal{A}_{56}$ uses passive voice for explaining an execution without mentioning an actor.

Strongly related to the question of the actor is the nature of the algorithm itself. $\mathcal{A}_{21}$, $\mathcal{A}_{27}$ and $\mathcal{A}_{28}$ speak of the algorithm as an entity that (actively) solves problems. $\mathcal{A}_{26}$ even goes so far to state that an algorithm "explains to a program what it should be doing and how it can do it", very clearly expressing not only actorship, but also higher cognitive functioning. In stark contrast, $\mathcal{A}_{58}$ sees an algorithm much more as a recipe to be followed, i.e. where the algorithm itself has no active role at all. The bulk of answers, however, are much more vague in whether algorithms are seen as entities that themselves perform actions or as passive instructions to be followed and executed by a distinct actor.

Interestingly, merely four answers explicitly spoke of a computer or machine being directly involved (e.g., $\mathcal{A}_{47}$) and six indicated a 'program' or 'program code' (e.g., $\mathcal{A}_{21}$). This is somewhat surprising but hints at some success in teaching computational thinking not as necessarily machine-based.

5.3 Defining Properties of Algorithms

Comparing with formal definitions of algorithms, we find a mixed bag. While a considerable part of the pupils described algorithms as sequences, processes, procedures or program code of some kind, other aspects such as finiteness seem to have been entirely neglected. Almost a third of the students explicitly mentioned the stepwise nature of algorithms with others implicitly indicating it through examples or keywords such as "a *string* of commands". Hence, the notion of a step-by-step procedure seems to have been fairly well understood by a majority.

Finiteness. Nothing about 'finiteness' can be found in the given answers of the students. In none of the answers is it ever explicitly mentioned that an algorithm has to terminate.

What we can find are eight students who attribute the goal of solving a problem to algorithms. One interpretation is that the algorithm has completed its task—and therefore terminates—after this goal has been reached, i.e. the problem has been solved. The mentioning of endless repetition, as for example in $\mathcal{A}_{29}$ "something that is repeated time and again" speaks rather of an opposite understanding. Here the repetition was put so strongly into the foreground that the most important criterion, the termination of the algorithm, got completely lost.

One possible explanation could be a confusion between computation and algorithm with the distinction between the two lying in the termination of the latter. While it is unlikely that the pupils are aware of these two concepts, we would argue that most modern computer applications have a never-ending character with a continuous query-reply-cycle. This is particularly true for applications such as chat and social media applications as well as internet search engines. At the same time, the term 'algorithm' is almost inflationary used in the media and public discourse.

Definiteness and effectiveness. Some answers explicitly mention definiteness, such as $\mathcal{A}_{42}$ and $\mathcal{A}_{50}$. Much more common, however, was the theme of a fixed order or fixed sequence.

The notion of a (computational) step that is *not* precisely defined might be rather alien to pupils of that age. If we assume that the

individual steps of an algorithm are 'obviously clear' then any uncertainty as to the execution of the algorithm would necessarily come from a change of the sequence of steps (either a rearrangement or the addition/removal of certain steps). The pupils' emphasis that the sequence itself is fixed could either indicate that an algorithm as such cannot be modified, or it means that no steps can be skipped or executed out-of-order. We argue that the latter interpretation would be closely related to definiteness.

As mentioned above, definiteness and effectiveness strongly relate to the idea of a computational model, which is hardly ever taught explicitly in the context of algorithms. However, as elaborated in our discussion of the executing actor above, there are some vague and implicit ideas about an underlying computational model expressed through the executing actor. Hence, even though no answer actually speaks of effectiveness as such, we would argue that some of it is still encoded in the answers given by the pupils.

Input and Output. Nothing about I/O is mentioned by any of the answers. Moreover, there is virtually no indication of algorithms as entities that process or work on data. Algorithms seem to either be processes embedded in (and interacting with) 'daily life' or procedures to solve a problem with no explicit interaction mentioned.

6 RESEARCH QUESTIONS

6.1 RQ1: How do upper elementary school students describe and explain what an algorithm is?

The student responses are dominated by somewhat unspecific and nebulous 'salient properties', but also very concrete examples. There are a number of variations on the theme of repetition as a defining property, but whether that repetition pertains to a continuous loop, the scheduled execution at specific times or just means that an algorithm can be 'reused' is not quite as clear.

We perceive a potential threat to the students' comprehension stemming from the examples mentioned by the students. While 'brushing your teeth' is surely meant to illustrate the idea of following a specific routine, the pupils seem to rather pick up the idea of doing something over and over again. In other cases, the pupils even deduced that algorithms (automatically) emerge out of a specific habit or recurrence. We would therefore caution against leaning too heavily on 'real-life' examples when teaching the concept.

When considering responses from novices, we have to be mindful that they might lack the vocabulary and understanding needed to even formulate precise questions. However, the answers we collected contain a number of keywords and properties which suggest that the students were able to express their ideas well enough to take the responses as actual reflections of their comprehension.

6.2 RQ2: In what aspects do students' conceptions of what an algorithm is differ from established definitions?

In general, students seem to have understood that algorithms are step-by-step procedures to be executed or followed. With some indication of a computational model as expressed through the executing actor, the explanations partly match the definiteness/effectiveness aspect of algorithms. However, the students' answers do not mention the need for an unambiguous and precise language or instructions, although the notion of a fixed sequence carries some of that characteristic. Moreover, instead of a guaranteed termination (aspect of finiteness), we often find infinite repetition to be brought forward as a key feature.

To arrive at better explanations or definitions we see two main attributes missing that should be more emphasised: the purpose of an algorithm as computing a 'result' (and thus necessarily terminating) as well as the idea of an underlying computational model.

7 LIMITATIONS AND FUTURE WORK

Our study clearly has an explorative character with a relatively small sample size. Moreover, since all pupils attended the same school, there is a high interdependence of the collected answers, further exacerbated by the shared tutors, some of whom might have discussion the idea of what an algorithm is in some detail and with various examples. We should also expect that students had some discussions among each other, further putting the independence of the collected answers into question. The provided results are therefore not necessarily representative of a larger student population.

Due to the voluntary and competitive nature of the survey, students who already had some initial idea of what an algorithm is likely participated with greater enthusiasm. Thus, the data set is probably missing answers from those pupils who had no working notion about the term algorithm at all. Receiving responses from all children holds potential for a future follow-up study. However, considering the limitations of children in expressing their comprehension of the concept of the term algorithm in a short written statement, a test or questionnaire to quantify their occurrence of the identified misconception could give us even more insights. Exploring and comparing algorithmic understanding across different educational levels is also a topic for future work.

Probably the greatest issue is that the pupils may not have the language to correctly express salient features of the algorithm-concept. As indicated above, the 'fixed sequence'-property of algorithms might actually refer to the idea of determiniteness. Follow-up studies will have to seek to better differentiate what the pupils mean.

8 CONCLUSION

It is imperative that we not only teach procedural knowledge and an intuitive understanding of (specific) algorithms, but also discuss the general concept of algorithms, their possibilities and limitations. As a first step towards establishing such a discussion of algorithms in general education, we have looked at how 6^{th} grade pupils describe the concept of an algorithm.

Our data shows a somewhat hazy notion that seems to be primarily based on the concept of repetition. However, we also found frequent mentioning of properties such as step-wise execution or the aim of solving problems. This indicates that the pupils have indeed developed a notion of the concept of algorithms, but that we need to improve the instruction and teaching towards working out the key properties more clearly.

REFERENCES

[1] Alfred V Aho. 2012. Computation and computational thinking. *The computer journal* 55, 7 (2012), 832–835.

[2] Lorin W. Anderson and David Krathwohl (Eds.). 2001. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives* (complete ed. ed.). Longman, New York.

[3] Tim Bell, Jason Alexander, Isaac Freeman, and Mick Grimley. 2009. Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology* 13, 1 (2009), 20–29.

[4] Tim Bell and Jan Vahrenhold. 2018. CS Unplugged—How Is It Used, and Does It Work? In *Adventures between lower bounds and higher altitudes: essays dedicated to Juraj Hromkovič on the occasion of his 60th birthday*, Hans-Joachim Böckenhauer, Dennis Komm, and Walter Unger (Eds.). Lecture notes in computer science, Vol. 11011. Springer, Cham, 497–521. https://doi.org/10.1007/978-3-319-98355-4_29

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2022. *Introduction to algorithms* (fourth edition ed.). The MIT Press, Cambridge, Massachusetts and London.

[6] Benedict Du Boulay. 1986. Some difficulties of learning to program. *Journal of Educational Computing Research* 2, 1 (1986), 57–73.

[7] Sally Fincher, Johan Jeuring, Craig S. Miller, Peter Donaldson, Benedict du Boulay, Matthias Hauswirth, Arto Hellas, Felienne Hermans, Colleen Lewis, Andreas Mühling, Janice L. Pearce, and Andrew Petersen. 2020. Capturing and Characterising Notional Machines. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. Association for Computing Machinery, New York, NY, USA, 502–503. https://doi.org/10.1145/3341525.3394988

[8] Michal Forišek and Monika Steinová. 2012. Metaphors and analogies for teaching algorithms. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 15–20.

[9] Gerald Futschek. 2006. Algorithmic Thinking: The Key for Understanding Computer Science. In *Informatics education - the bridge between using and understanding computers (Lecture notes in computer science)*, Roland T. Mittermeir (Ed.). Springer, Berlin and Heidelberg, 159–168. https://doi.org/10.1007/11915355_15

[10] Robin K Hill. 2016. What an algorithm is. *Philosophy & Technology* 29 (2016), 35–59.

[11] Christopher D. Hundhausen, Sarah A. Douglas, and John T. Stasko. 2002. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing* 13, 3 (2002), 259–290.

[12] Donald Ervin Knuth. 1997. *The art of computer programming. Volume 1, Fundamental algorithms* (3rd ed. ed.). Addison Wesley, Place of publication not identified. https://permalink.obvsg.at/

[13] Tobias Kohn and Dennis Komm. 2018. Teaching Programming and Algorithmic Complexity with Tangible Machines. In *Informatics in Schools. Fundamentals of Computer Science and Software Engineering*, Sergei N. Pozdniakov and Valentina Dagienė (Eds.). Springer International Publishing, Cham, 68–83.

[14] Linda Mannila, Valentina Dagiene, Barbara Demo, Natasa Grgurina, Claudio Mirolo, Lennart Rolandsson, and Amber Settle. 2014. Computational Thinking in K-9 Education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference* (Uppsala, Sweden) *(ITiCSE-WGR '14)*. Association for Computing Machinery, New York, NY, USA, 1–29. https://doi.org/10.1145/2713609.2713610

[15] Philipp Mayring. 2021. *Qualitative content analysis: A step-by-step guide* (1st edition ed.). SAGE Publications Ltd, Thousand Oaks.

[16] Merriam-Webster. 2023. Algorithm. www.merriam-webster.com/dictionary/algorith

[17] Brent Daniel Mittelstadt, Patrick Allo, Mariarosaria Taddeo, Sandra Wachter, and Luciano Floridi. 2016. The ethics of algorithms: Mapping the debate. *Big Data & Society* 3, 2 (2016), 1–21.

[18] Cristopher Moore and Stephan Mertens. 2011. *The nature of computation*. Oxford University Press, Oxford.

[19] Jacob Perrenet, Jan Friso Groote, and Eric Kaasenbrood. 2005. Exploring students' understanding of the concept of algorithm: levels of abstraction. *ACM SIGCSE Bulletin* 37, 3 (2005), 64–68.

[20] Bethany Rittle-Johnson. 2019. Iterative development of conceptual and procedural knowledge in mathematics learning and instruction. In *The Cambridge handbook of cognition and education*, J. Dunlosky and K. A. Rawson (Eds.). Cambridge University Press, Cambridge, 124–147. https://doi.org/10.1017/9781108235631.007

[21] Cynthia Selby and John Woollard. 2013. *Computational thinking: the developing definition*. University of Southampton (E-prints). https://eprints.soton.ac.uk/356481

[22] Clifford A. Shaffer, Matthew L. Cooper, Alexander Joel D. Alon, Monika Akbar, Michael Stewart, Sean Ponce, and Stephen H. Edwards. 2010. Algorithm Visualization: The State of the Field. *ACM Trans. Comput. Educ.* 10, 3, Article 9 (aug 2010), 22 pages. https://doi.org/10.1145/1821996.1821997

[23] Rivka Taub, Mordechai Ben-Ari, and Michal Armoni. 2009. The Effect of CS Unplugged on Middle-School Students' Views of CS. *SIGCSE Bull.* 41, 3 (jul 2009), 99–103. https://doi.org/10.1145/1595496.1562912

[24] Jeannette M. Wing. 2006. Computational Thinking. *Commun. ACM* 49, 3 (mar 2006), 33–35. https://doi.org/10.1145/1118178.1118215