Creative Education
2013. Vol.4, No.9, 557-562
Published Online September 2013 in SciRes (http://www.scirp.org/journal/ce)
Scientific Research
http://dx.doi.org/10.4236/ce.2013.49081

Algorithmics for Preschoolers—A Contradiction?

Roland T. Mittermeir
Institute for Informatics-Didactics, Universität Klagenfurt, Klagenfurt, Austria
Email: roland@isys.uni-klu.ac.at

Received May 23rd, 2013; revised June 23rd, 2013; accepted July 1st, 2013

Developing an algorithm requires expressing it in some (formal) language. The respective language is usually understood to be textual (conventional programming language) or partly graphical (design languages, and languages in programming environments for children). As writing and reading are capabilities not to be presumed from preschoolers, many educators claim that confronting such young kids with algorithmic concepts is beyond their abstraction capability. This paper reports on an experiment with kindergarten-groups requiring them to discover simple algorithms without resorting to reading and writing. It clearly showed that limited capabilities of abstractions are not a hurdle at all, if the problems are posed in a way corresponding to the limited experience base of the children, and if solutions are small enough to be kept in memory and allow expressing themselves in other forms than writing.

Keywords: Algorithmics; Preschoolers; Capability for Expressing Algorithms; Capability for Abstracting

Motivation and Background

The experiments reported here took place within a funded project, *Informatik erLeben*, requiring that at least three age-groups are involved. These age groups could range from kindergarten via primary schools (Grades 1 to 4; age group: 6 to 10) to secondary schools I (Grades 5 to 8 or 9) or to secondary schools II (Grades 9 to 12 or 13).

The aim of the project has been to show pupils and school-students that *Informatics* is a technical subject resting on a scientific background of science, physics and mathematics and has also developed its own distinct contributions in theoretical and applied Computer Science. Thus, informatics is more than pushing buttons on a laptop or on a modern cell-phone; and the latter is being quite often the core of informatics instruction in school. Sometimes, fair enough, these topics are labeled as ICT-instruction. In some countries, Austria being one of them, unfortunately this instruction is performed under the label "Informatics instruction" with "Informatics" referred to as local synonym for Computer Science (which it is, when comparing university studies).

In a number of countries movements are against such semantic bending of the term "Informatics" and discussions as to what kind of knowledge should be introduced in the standard school curriculum emerged and led to a reversal of this trend. Respective arguments can be found in documents emanating from professional and/or learned societies (c.f. Wilson et al., 2010; Furber et al., 2012). Engraining algorithmic thinking is always a key element in such deliberations as well as in arguments raised by individual scientists, (c.f. Wing, 2006; Hromkovič, 2006; Mittermeir, 2010). This holds in spite of Dennings (2004) warning that programming does not circumscribe the complete wealth of CS and in spite of the difficulties school administrations are facing when demanding informatics-instruction beyond ICT-training, pointed at by Sentance, Dorling, & McNicol (2013) and Tort, & Drot-Delange (2013).

The project *Informatik erLeben* had a pilot-phase in the school-years 2008/09 followed by application phases in 2009/10 and 2010 till February 2012. Its overall aim is described in Bischof & Mittermeir (2008) and in Mittermeir, Bischof, & Hodnigg (2010). In the sequel, the paper focuses only on those parts that took place in kindergarten, i.e., with preschoolers.

In the remainder of the paper, first possible interpretations of the word "Algorithm" are discussed. On this basis, the details of the experiment conducted with preschoolers and the insights obtained are described. As this is definitely not the only approach to introduce true CS-concepts to preschoolers, and a section briefly discussing alternative proposals suitable for preschoolers depending on their relative maturity follows before drawing conclusions from the approaches described.

Description of the Experiment

Algorihms—Central Elements of Computing for All

It needs to be mentioned that preparing preschoolers for a professional career as software developer it is neither the author's intention nor, hopefully, an intention of parents or educational decision makers. But algorithmic thinking (or according to Wing (2006) computational thinking) is a capability important for all contemporary school graduates indeed. This is a position, though phrased differently, already taken by Harel (1987) when arguing for the importance of algorithmic concepts in many disciplines. Based on this claims, algorithmic concepts should be gradually introduced already early in a pupils education. However, for a substantial number of teachers describing and developing an algorithm requires such a high level of abstraction that students have to have passed at least the *n*-th grade (*n* being quite often quoted as a number between 8 and 10) before they can write an algorithm.

Such statements quite often equate the concept of algorithm with the concept of computer programming, ignoring that learning programming consists of two quite different capabilities:

1) Developing an algorithmic solution in order to transfer a statically given problem statement into a statically specified solution;

2) Translating the algorithmic solution, given precisely, but nevertheless in a (sub-) language the pupils are familiar with, into some formal language, eventually a programming language or a data manipulation language.

The description of how to reach some well-known target within the school house starting at the class-room as proposed by Kolczyk's (2008) spiral teaching model clearly distinguishes between these two tasks intermixed in traditional programming instruction.

It is hard to blame teachers falling into the trap of this misconception. When looking at certain dictionaries or handbooks of Computer Science, one finds definitions such as "Given both the problem and the device, an *algorithm* is the precise characterization of a method of solving the problem presented in a language comprehensible by the device. In particular…" (Korfhage, 1983). It is interesting that this definition is placed in the context of small FORTRAN programs apparently intended to explain the concept. A shorter definition following the same line of arguments is given by Maly (1984) in the Handbook of Computers and Computing. "An algorithm is a finite sequence of well defined instructions each of which can be carried out mechanically within a finite amount of time; furthermore, an algorithm always halts". It is noteworthy that the chapter on algorithms introduces also concepts of programming languages such as procedures or how to program recursion.

It has to be seen though, that the person who's name is honored by this term, Al Khowarizmi[1] focused in his books (originals destroyed when the "House of Wisdom" in Baghdad has been destroyed) in the early ninth century on algebra and arithmethic. He also introduced the Arabs to the number system used by Indian astronomers (Williams, 1997). The computations used for developing astronomical tables were basically to be executed on sand tablets, quite comparable to sheets of (erasable) paper (Berggren, 1986, 2011). So is the supposedly first algorithm ever published, Euclid's algorithm for finding the greatest common devisior between two integers, (presumebly Euclid, according to Shipley et al. (2006) between 5th and 3rd century) precisely defined, but independent from any particular device.

A definition commensurate with this traditional concept can be found in Marciniak's Encyclopedia of Software Engineering (1994). Here, one can read "1) A finite set of well-defined rules for the solution of a problem in a number of steps; for example a complete specification of a sequence of arithmetic operations for evaluating sine x to a given precision; 2) Any sequence of operations for performing a specific task (IEEE)."

The Experimental Group

The experiment reported here took place in a kindergarten in Klagenfurt, Austria. The group, following the full agenda of four interventions consisted of 10 pupils in the age from 3 years to 6 years. During the first intervention, the one dealing with search algorithms, only 6 pupils took part (4 girls and 2 boys). They fell into the age group of 4 to 6 years.

The intervention has been performed by Ernestine Bischof, a member of this department. The kindergarten teacher responsible for the children, Ms. Horn, was present during the full time. So was the director of this kindergarten, Ms. Krenn-Wache, during the initial unit.

The particular research question behind teaching algorithmic concepts to preschoolers has been how early, i.e. at how low an age group, one might start teaching informatics as a technical subject to children. The question came up during the piloting phase. There we determined that with most classes of primary school, more advanced topics could be addressed than originally anticipated.

Agenda of the Experiment

The intervention reported here lasted for one hour. It started with the algorithmic part, *find and describe a simple search algorithm*, and concluded by opening a PC, showing pupils the components and allowing them to disassemble the device.

The algorithmic part required pupils to identify within a bag of cotton (prohibiting visibility) the shortest among a set of colored pencils of different length just by sensing using one hand only. (The other hand was needed to hold the bag.)

Readers considering this to be too trivial a task for showing and discussing algorithmic concepts might pause here a little and define two different strategies to find a solution. As these doubtful readers are grown up, it seems fair to require from them also voicing arguments, why their strategy (their algorithm!) necessarily leads to the correct result.

The later requirement would obviously be totally inadequate for preschoolers. They were just required to remember how they solved the problem and not to mention their approach till everybody had her or his try. All of them had a chance to find the smallest one.

The sample is too small to generalize any gender differences out of the result. Nevertheless it is worth mentioning that all 4 girls presented a correct solution while both boys missed the target. Moreover, the girls had the result a little faster than the boys. However, all pupils worked concentrated and were able to describe their approach. Here again, two categories could be identified. One used rather an (almost) random approach, others worked according to a particular strategy.

This initial experiment has been conceptually repeated by asking the kids to identify the longest pencil. This task was a bit more difficult, as the difference in length among the longest and second-longest pencil was minor. Considering this differences, several rounds of trials were made. Nevertheless, some succeeded right away. In order to further help those, who did not find the solution on their own; director Krenn-Wache proposed to use a wooden brick to adjust the pencils to a common bottom-line and another one to measure their height. This eventually led to full success for all participants.

Evaluation of the Experiment

Obviously, the experiment suffers from a small sample size. Consequently, one has to be extremely careful with interpretations, especially those concerning the gender differences observed. But, the results are clear enough to state the following.

- Children from an age group ranging from 4 to 6 years are capable of pondering about a good strategy, i.e., of devel-

[1]There exist several transliterations for the shortened name of Mohammed ibn Musa Al-Khowarizmi, depending on how the Arab letter "ف" is transcribed. Hence, Khwarizmi, is another transliteration often found.

oping some algorithm.
- Children of this age group were able to verbalize the algorithm they devised.
- Children of this age group could reflect on different solutions and apply hints to improve or revise their algorithm productively.

One has to mention that a set of various evaluative measures has been applied in experiments targeted for pupils of primary or secondary school. In the case of preschoolers we were limited to the reports from the kindergarten teacher. Fortunately, she was highly cooperative and provided us with very informative reports that were helpful in re-interpreting the experimenter's observations during the intervention.

In this context it is worth mentioning, that she returned to the issues dealt with during the experiment the day after. This was certainly instrumental to keeping some aspects and terminology for a long time, as noted in her report after the end of the project, more than half a year after this experiment.

Related Experiments

Before concluding, the paper wants to shed some light on approaches similar to the one reported. Most of them are not targeted specifically for preschoolers. However, notably the approach reported by Futschek & Moschitz (2011) seems to fit perfectly the topic of this paper, even if the authors have developed it for primary school.

Tim the Train

The approach to familiarize very young children with algorithmic concepts proposed by Futschek & Moschitz (2011) uses a short wooden train (2 to 3 wagons) to be loaded by differently shaped and differently sized blocks of wood. The target is that no carriage is overloaded and no block remains.

The suitability of the approach for preschoolers is due to the fact that the algorithm is not to be given in writing. It is rather to be specified by a set of symbols burned into wooden pieces (like domino stones). The set of symbols available encompass: empty the full train (reset), image of a wagon with errors pointing left or right (move train forward or backward), and errors putting up or down (load or unload the piece placed after this command stone).

The trick with this conceptually already rather complex approach is the limited language. The narrow vocabulary reduces the complexity sufficiently such that pupils in primary school, and supposedly also preschoolers, are able to solve such tasks.

Other Units Performed with Preschoolers

During the project *Informatik erLeben* further units presenting informatics as technical subject were made with the children. They also had to pay a visit to the university as well as to a company working in informatics. As these units were not specifically related to algorithmic thinking, they are mentioned here only briefly.

- A hardware unit. The children could open and disassemble a PC no longer in use in the department. This was probably the unit most exciting to them. Gender roles were switched in so far as boys took a leading role. But girls nevertheless were very interested and had their part in this unit. The dominance of boys in the hardware unit has been observed also in primary and secondary schools. Since there the groups were larger, we insisted that disassembling hardware was done in at least two single gender groups.
- Synthesis of color. The unit has been almost a direct copy of the unit for primary school classes. Children got glasses filled with diluted acrylic colors of yellow, magenta, and cyan. Posed with the question, how a printer can produce arbitrary colors if it is only loaded by these three, the prepared classes sufficed to generate the answer "by mixing". Then, children could experiment on their own, intermittent by further questions and input from the teacher.
- During a visit to the usability lab of the HCI-group of the university the children could explore various alternatives to remotely control a user interface or to control the movement of a swarm of butterflies.

Analysis of Algorithms

Part of the repertoire of the *Informatik erLeben* units are various sorting algorithms. One or the other of $O(N^2)$-algorithms will have been intuitively known already by some of the children (usually a specific one playfully explored with sorting some toy-bricks or similar material, but a given child usually strictly follows the approach once identified). Quite expectedly, $O(N \log N)$-algorithms are not known. Hence, they are introduced under the motivation "think before sweating".

The unit consists of analyzing two sorting algorithms of different complexity. It is advisable, though not necessary, to work on this unit after the children or students are already familiar with some basics of computer technology such as the unit "executing a simplified 'human' computer" mentioned below. Knowing basic hardware aspects helps them to better appreciate why comparisons are only possible between two elements. Thus, at the elementary level, the computer cannot do some pattern matching considering several pieces at a time, as a human would do. In order to stress this, students/ children play various roles. Among them, one child serves as index pointing to the element (child in the arrangement to be sorted), one serves as intermediate memory (register) to keep the value against which the currently considered element is to be compared, and another child remembers the position of the element serving for the comparison. One child might serve only to ask for the values involved, compare them and initiate the next move. Different algorithms might necessitate different roles from those just mentioned. All children not playing one of the special roles are lined up as elements (data) to be sorted.

With students attending higher grades, a given student can play various roles simultaneously (e.g., remembering value and position). With pupils of lower grades in primary school, it is important though, not to overload them with too many duties. With primary schools, the classes are usually large enough, that even a fair spread of the special roles leaves enough persons to be (re)sorted. In kindergarten the groups are smaller. Consequently, it might be advisable to perform the unit with children from two or even three groups at a time. About 15 children appear to form a well sized group.

After each of the two sorting experiments, a discussion about the activities that took place is held with the class in order to identify characteristics of the algorithm. In this discussion, all children should be involved. But one could assign to some of them the special role of observer, if the group tends to become too large. Otherwise, especially those children serving as items to be sorted are particularly invited to voice their observations.

It needs to be mentioned that sorting units have not been applied to preschoolers so far. They were applied to children attending primary school, even from lowest grades onwards, up to students attending upper grades in secondary school. For all these groups, reading capability can be presupposed. Hence motivating the quest for different algorithms, notably for searching and sorting, can easily be obtained by asking, how learners would look up the phone number of their school in a phonebook or search for some particular information in a dictionary, given their preferred internet-source has broken down temporarily.

For children knowing the alphabet and the collating sequence of letters, defining the sorting task is easy in so far as they are asked to form a line sorted on tallness. Since the size of a person can be assumed to be statistically not related to the name of this very person, applying different algorithms to re-sort them by given name seems to be a fair task. Analyzing assumes data currently in random order with respect to the new criterion. With higher grades, one might discuss the effect of pre-sorted data. With lower grades, we were happy if the children saw that different algorithms, though producing the same result, had different resource (here: time) consumption. Below, the modifications needed for preschoolers are described.

- Sorting for preschoolers. As alphabetization cannot be presupposed with preschoolers, the concept of sorting by name is no option. Instead of name, domino-like (one sided) stones can be used. Children would first pick randomly a stone that has been placed face-down on a table. Then they line up according to their tallness like in a gym-class. The student controlling the algorithm by asking each child for the value of its sorting criterion asks now for the number of dots on the stone instead of asking for the given name.
- Number systems. The domino-like stones just introduced could also be used to familiarize students with different number systems, including the binary system.

Normally one would think about a domino like stone as an arrangement as shown in **Figure 1**. However, one can also dissolve the plain number (5 in this case) into a sum of powers of two as shown in **Figure 2**.

The crossed off empty field might be disturbing for the expert. However, it is introduced to remain compatible with the empty domino field. It seems advisable to keep it initially and let the children discover the effect of crossing off the field with only one dot. Thus, the empty field is not needed. Crossing off everything will lead to the same number "0".

From having those stripes with some fields present, some crossed off; it will be an easy step to write underneath a crossed-off field "0" and underneath a still existing field "1". For each "1" the number of dots are counted and added. The number of dots in "0"-fields are ignored during this summation. (See also discussion of the unit related to why computers could perform calculation, detailed below.)

More Advanced Units

Though results from the literature and our experience of vastly differing capabilities among age groups results rather from school classes, it seems fair to assume that this holds at least to some extent also for groups of preschoolers.

Another experience we made was that with 5 to 6 years old

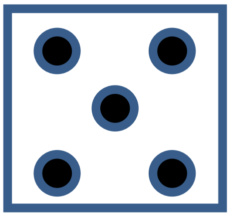

Figure 1.
Example of a one-sided domino-like stone.

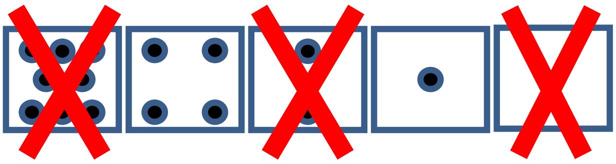

Figure 2.
A stripe consisting of images showing domino-like stones of increasing value from right to left with invalid elements crossed off to prepare for binary numbers.

children the capability to read and write numbers is present to a larger extent than the capability to write characters or words (smaller set of symbols, simple construction rule). Further, simple arithmetic (addition, subtraction) over a small range of numbers (1 to about 20) might be presumed. Thus, future experiments might include also units demanding some arithmetic.

Candidates are the following.

- Executing a simplified "human" computer. With this unit defined for primary school pupils, a program containing I/O, load and store operations and operations to be performed by the arithmetic logical unit (ALU) are performed. The role of the devices involved (input and output channel, bus, storage, ALU, program counter, and in a more involved version also registers) are assumed by pupils. The current version of this unit requires reading the assembly program to be executed. A variant for preschoolers might use graphic symbols instead.
- Another unit even more banking on calculation capabilities of the children would be to give them an answer why the computer could do calculations even if it works only on a bi-state, i.e. binary, basis. The tasks can be done either individually or in small groups. To have enough pre-knowledge in order to motivate this example, children should be familiar at least with the numeric range from 0 to 20; 0 to 30 would be preferable. The approach would use single field domino stones having no, 1, 2, 4, 8, and if possible 16 large marks. Below, each field contains 4 bars (in case of 16 marks 5 bars), containing either no mark, one mark on the rightmost bar, only one on the bar left to the rightmost, only one on the bar right to the leftmost, and finally only one on the leftmost. Apparently, the marks on the bars indicate the position of a "1" in the binary code representing the number of large marks in the field above them. Given this equipment and pencil and paper, students could build a decimal number by noting the number of large marks on the stones available as decimal numbers and summing those afterwards. Then each one might place the stones that would add up to this number in a column and put below marks in those positions that had marks in the bottom line of the stones now forming a column. Thus, two versions of the number

sought were obtained: a decadic one and a binary one. That both systems rest upon positions indicating the respective power of the base, in one case 10, in the other case 2 should be mentioned only by persons possessing an adequate background in mathematics and mathematics didactic. But one does not need to go so far, it suffices to mention the similarity of the system and to convince the children that an algorithm ensuring the proper conversion between those systems does exist.

It needs to be mentioned though that both proposals rest on our experience in primary school and in kindergarten. But so far they have not been put into practice.

Discussion

The various approaches described in the previous subsections seem simple enough that teachers or kindergarten teachers can easily adapt them to the particular situation in their class or kindergarten-group. Such modifications are foreseen and indeed intended by the designers of the *Informatik erLeben* project (Mittermeir, Bischof, & Hodnigg, 2010). Though not explicitly mentioned, the continuation of the approach by having pupils write Scratch or BYOB programs is apparently also part of the approach presented by Futschek & Moschitz (2011). Doing so, however, requires utmost care.

With *Tim the Train* it seems almost natural to define the problem in such a way that not either the train or the blocks are moved but that only a crane is serving as robot controlled by the pupil's algorithm. The suggestion is obvious. However the simplification has the price that the length of the algorithm would markedly increase and, therefore, its perceived complexity would increase too.

Similar considerations might be raised for the searching problem of *Informatik erLeben* detailed the previous section. E.g. the choice of algorithms to be selected depends largely on whether the bag is placed on a table or is freely held by one hand of the experimenting child.

The example solutions given below should show that irrespective of the details the teacher finally decides upon, she or he should first establish a list of expected solutions, both wrong or correct, before deciding on the particular variant of the approach chosen. The list should be rich enough to contain several legitimate solutions allowing discussing merits and weaknesses of competing approaches. It should also contain some anticipated mistakes in order to explain not only the child who fell into such a trap but make the whole group aware why such errors do occur and what should be considered to avoid them. Especially with very young children it is important to not blame them for errors but to make them learn and grow from understanding why a particular error occurred.

- Pick an arbitrary pencil.
  This is basically ruled out since it would demonstrate lack of cooperation and motivation by this child.
- Pick a pencil; compare its length to another one. If it is smaller than keep it, if the other one is smaller take this for further comparison. Stop after you made as many comparisons as there are pencils in the bag.
  The approach does not guarantee a correct solution, since the pencil to be chosen might be used several times while others are not compared at all. The fact that equality is not considered in this solution should be discussed, but is of minor importance. That the stopping criterion will be reached after n-1 instead of n comparisons might be considered irrelevant for preschoolers. It is important, though, that some stopping criterion is mentioned. In several cases, it is mentioned only after questioning for it.
- Same as before, but pencils still to be used for comparison are placed on one side of the bag, after having been used, the pencil not serving as new smallest element is placed on the other side.
  This algorithm will, if correctly performed yield the correct result. The stopping criterion discussed above becomes obvious (no further pencil left for comparison) and has a better chance of being voiced by the respective child itself. It is to be noted that it would be very difficult to come to this solution, if the bag with pencils is not placed at least partly on the table.
- Sorting all pencils and finally picking the first one on the end containing the small pencils.
  This again yields a correct solution. The investment seems high. During the discussion phase of the experiment one should discuss in which situation this investment will pay off. Apparently, this solution will hardly appear, if the bag cannot be placed on a table.
- "Sorting" with the aid of rulers or wooden blocks as supporting device.
  This comes close to the aid proposed by the head-master of the kindergarten where the experiment detailed in the previous section took place. She proposed two wooden blocks for the second version of the experiment, looking for the longest pencil. Here, we first consider only one such block for marking the common base-line for all pencils. This would not change the algorithm but it would at least speed up finding a solution. Above, "sorting" is put under quotes to indicate that sorting need not necessarily be done by a conventional sorting algorithm. Physical objects, pencils in particular, can be moved quasi-parallel in a synchronous manner defeating the algorithmic complexity of computerized searching and sorting algorithms.
- Using one wooden block as base and one as tally.
  This will not work well for finding the shortest pencil, but it works well for finding the longest pencil. As one block ensuring a common base, the other one can be gliding down slowly, remaining parallel to the block defining the base. The first pencil blocking this movement will be longer than the longest found so far. One might note that two analogous formulations from the computer scientist's point of view (find shortest, find longest) give way to quite different solutions with different algorithmic complexity.
- Using the flexibility of the bag.
  The previous operations can all be performed inside the bag. However, at least the final one might suggest that the gliding block is moved outside the bag. If this is permitted, one might as well align the pencils as discussed above and then visually inspect how the bag shows the profile of its content defined by the length of the individual pencils. Some educators might consider this cheating and prohibit this approaches or variations thereof. However, it in no way violates rules of the problem specification and it still requires an algorithmic solution in the preparatory phase before using the pattern recognition capability of the human visual system. It is just an innovative solution not foreseen by the problem's designer (nor seen during or experiments).

Conclusion

The main conclusion of the experiment reported here is that preschoolers are capable of designing algorithms, expressing them, and understanding them if presented properly. Thus, arguments such as "children below some already advanced age are incapable of abstracting to the extent of designing and working with algorithms" need to be revised by adding the suffix "if not properly introduced in a way conforming to their age and general non-algorithmic capabilities".

This conclusion should not be overly surprising, since one can give preschoolers already a small set of tasks they have to (and they can) organize themselves. Likewise, one might ask them to follow a sequence of tasks in unspecified or moderately defined (open alternatives) order and they will perform them in an efficient, non-redundant way.

Identifying loop structures is certainly easier if some written representation as in *Tim the Train* exits, since repetition will become more obvious if visible. But even in the strictly verbal approach followed by having kindergarten-children sorting pencils, those who had a strategy did not report an overly long story but mentioned that they repeated a certain subsequence of steps until a stopping criterion had been reached.

In comparison with experiments or interventions held in school, it needs to be mentioned that the relative freedom kindergarten teachers have in comparison to school teachers proved positive. Deepening an intervention on the day thereafter is much easier in kindergarten than in school with an already crowded schedule.

Acknowledgements

The contribution of Karin Hodnigg to the overall design of the project during the piloting phase and of Ernestine Bischof who conducted the interventions in the Kindergarten-group, refined many units and held many interventions is gratefully acknowledged. So is the excellent cooperation of Ms. Horn-Hohenegg, the person responsible for the Kindergarten-group we were working with. The indirect contribution of Peter Holub, who challenged me to "downgrade" some already existing units adapting them for preschoolers, needs special mention.

Funding of the project by "generation innovation", a venture of the Austrian Ministry of Traffic, Innovation, and Technology is gratefully acknowledged. The piloting phase of the project has been supported by KWF, the Kärntner Wirtschaftsförderungs Fonds.

REFERENCES

Berggren, J. L. (2011). *Mathematik im mittelalterlichen Islam*. New York: Springer, Heidelberg, Springer-Verlag.

Bischof, E., & Mittermeir R. (2008). *Informatik-erLeben*. Institut f. Informatiksysteme. Klagenfurt: Universität Klagenfurt. http://informatik-erleben.aau.at

Denning, P. J. (2004). The field of programmers myth. *Communications of the ACM, 47,* 15-20. doi:10.1145/1005817.1005836

Furber, S. et al. (2012). Shut down or restart? The way forward for computing in UK Schools. London: The Royal Society. http://royalsociety.org/uploadedFiles/Royal_Society_Content/education/policy/computing-in-schools/2012-01-12-Computing-in-Schools.pdf

Futschek, G., & Moschitz, J. (2011). Learning algorithmic thinking with tangible objects eases transition to computer programming. In *Proceedings of the 5th ISSEP, informartics in schools—Contributing to 21st century education* (pp. 155-164). Berlin: Heidelberg, Springer.

Harel, D. (1987). *Algorithmics: The spirit of computing*. Boston: Addison Wesley Publishing.

Hromkovič, J. (2006). Contributing to general education by teaching informatics. In *Proceedings of the 2nd ISSEP, informatics education—The bridge between using and understanding computers* (pp. 25-27). Berlin: Heidelberg, Springer.

Kolczyk, E. (2008). Algorithm—Fundamental concept in preparing informatics teachers. In *Proceedings of the 3rd ISSEP, informatics education—Supporting computational thinking* (pp. 265-271). Berlin: Heidelberg, Springer.

Korfhage, R. R. (1983). Algorithm. In A. Ralston, & E. D. Reilly Jr. (Eds.,), *Encyclopedia of Computer Science and Engineering* (2nd ed., pp. 51-54). New York: Van Nostrand Reinhold Company Inc.

Maly, K. (1984). Algorithms. In A. H. Seidman, & I. Flores (Eds.), *The handbook of computers and computing* (pp. 805-814). New York: Van Nostrand Reinhold Company Inc.

Marciniak, J. J. (1994). *Encyclopedia of software engineering.* New Jersey: John Wiley & Sons.

Mittermeir, R. (2010). Informatikunterricht zur Vermittlung allgemeiner Bildungswerte. In *Proc. 25 Jahre Schulinformatik in Österreich*: *Zukunft mit Herkunft* (pp. 54-73). Wien: OCG Schriftenreihe.

Mittermeir, R. T., Bischof, E., & Hodnigg K. (2010). Showing core-concepts of informatics to kids and their teachers. In *Proceedings of the 4th ISSEP, teaching fundamental concepts of informatics* (pp. 143-154). Berlin: Heidelberg, Springer.

Sentance, S., Dorling, M., & McNicol, A. (2013). Computer science in secondary schools in the UK: Ways to empower teachers. *Proceedings of the 6th ISSEP, informatics in schools: Sustainable informatics education for pupils of all ages* (pp. 15-30). Berlin: Heidelberg, Springer.

Shipley, G., Vanderspoel, J., Mattingly, D., & Foxhall L. (2006). *The Cambridge dictionary of classical civilization*. Cambridge: Cambridge University Press.

Tort, F., & Drot-Delange, B. (2013). Informatics in the French secondary curricula: Recent moves and perspectives. In *Proceedings of the 6th ISSEP, informatics in schools: Sustainable informatics education for pupils of all ages* (pp. 31-42). Berlin: Heidelberg, Springer.

Williams, M. R. (1997). *A history of computing technology* (2nd ed.). Los Alamitos, CA: IEEE Press.

Wilson, C., Sudol, L. A., Stephenson, Ch., & Stehlik, M. (2010). Running on empty: The failure to teach K-12 computer science in the digital age, ACM and csta. http://www.acm.org/Runningonempty http://csta.acm.org/Runningonempty

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49,* 33-35. doi:10.1145/1118178.1118215

