The Education Column

BY

Juraj Hromkovič

Department of Computer Science
ETH Zürich
Universitätstrasse 6, 8092 Zürich, Switzerland
juraj.hromkovic@inf.ethz.ch

Interdisciplinary Education in Mathematics and Informatics at Swiss High Schools

Urs Hauser
Department of Computer Science, ETH Zurich, Switzerland
urs.hauser@inf.ethz.ch

Dennis Komm
Department of Computer Science, ETH Zurich, Switzerland
Pädagogische Hochschule Graubünden, Chur, Switzerland
dennis.komm@inf.ethz.ch

Abstract

Informatics will be introduced as mandatory subject at schools in Switzerland within the next years, and the corresponding curricula are under development. Mathematics and informatics have a lot of common ground, and therefore generate synergies that can allow both subjects to profit. To this end, we need a good coordination of mutual topics in form of an interdisciplinary spiral curriculum, which acts as a bridge between the two subjects. In this article, we supply some ideas by giving examples for Swiss high schools (grades 7 through 12).

1 Introduction

With finally making informatics a mandatory subject at Swiss high schools, resources are created that allow teaching the fundamental concepts in a sustainable way. Additionally, we now have the chance to coordinate mathematics and informatics such that both subjects profit from synergies. This article is organized as follows. In Section 2, we formulate general didactical reflections on both subjects as well as the underlying basic principles of our approach. In Section 3, we then give concrete examples that follow the idea of an *interdisciplinary spiral curriculum*. Section 4 contains concluding remarks.

2 Didactical Reflections

Mathematics and informatics are at the same time symbiotic and complementary. On the one hand, mathematics is an *axiomatic-deductive* discipline. It provides an exact language that allows us to explore and describe fundamental concepts of informatics. On the other hand, informatics is more of a *descriptive-constructive* discipline that strengthens problem solving and abstraction abilities, which are two of the core competencies of mathematics.

However, mathematics classes mostly facilitate constructive and algorithmic aspects of the subject; for instance, how to add two fractions, how to solve linear or quadradic equations, how to multiply 10-digit numbers, or how to deal with calculus of interest and compound interest. Obviously, such algorithmic calculations blend in very well with informatics, which has its focus on algorithms and their analysis. Instead of executing mathematical calculations (without comprehension), informatics provides an opportunity to design and implement algorithms, and consequently also to understand the calculation process in detail.

Both subjects are ideally taught in a problem-oriented way, and pupils are enabled and encouraged to develop, test, evaluate, implement, and optimize their own solution strategies. Through *discovery learning* in the context of the subject's history and genesis of specific notions, pupils are lead to a stepwise understanding of modern concepts. One possibility to reach this goal is an interdisciplinary spiral curriculum that captures notions of the respective other subject and gradually intensifies their study.

2.1 Basic Design Principles

We follow the basic design principles of mathematics classes as fomulated in "Homo Informaticus" [6]; they are also valid for informatics classes.

1. *Focus on the genesis of the fundamental notions (concepts).* The focus should not lie on conveying complex modern concepts which have been developed and optimized over a long period, since this prevents pupils from capturing the purpose of (or even the need for) these concepts.

2. *Concrete examples first, abstraction as a final discovery.* As a first step, a concrete problem, or even problem instance, should be introduced using discovery learning; then a strategy towards solving this task should be developed before a general method is implemented.

3. *Teach algorithmics instead of training calculation methods.* As mentioned above, pupils should have the opportunity to develop, test, evaluate, implement, and optimize their own problem solving methods.

3 Building Bridges

Mathematics and informatics are indeed strongly related and partly study the same notions. Our metaphor of "building bridges" refers to the design of both classes and curricula; and our goal is a coordinated spiral curriculum instead of two subjects held completely independently. Unfortunately, there is quite some risk to end up with such a curriculum as demonstrated with the subjects of mathematics and physics. It is a nontrivial challenge for curriculum designers to come up with curricula and topics that are well coordinated.

In what follows, we provide a few examples of topics that achieve this goal using a spiral approach; of course, these suggestions cannot always be implemented in a one-to-one fashion, but have to be fine-tuned according to the given curricula and other circumstances. More detailed examples are found in the textbooks "Einfach Informatik" [7, 8, 10].

3.1 Building Bridges with a Turtle

The concept of *turtle graphics* is in many respects very well suited to implement the basic design principles stated in Section 2.1 within a spiral curriculum throughout high school (Figure 1).

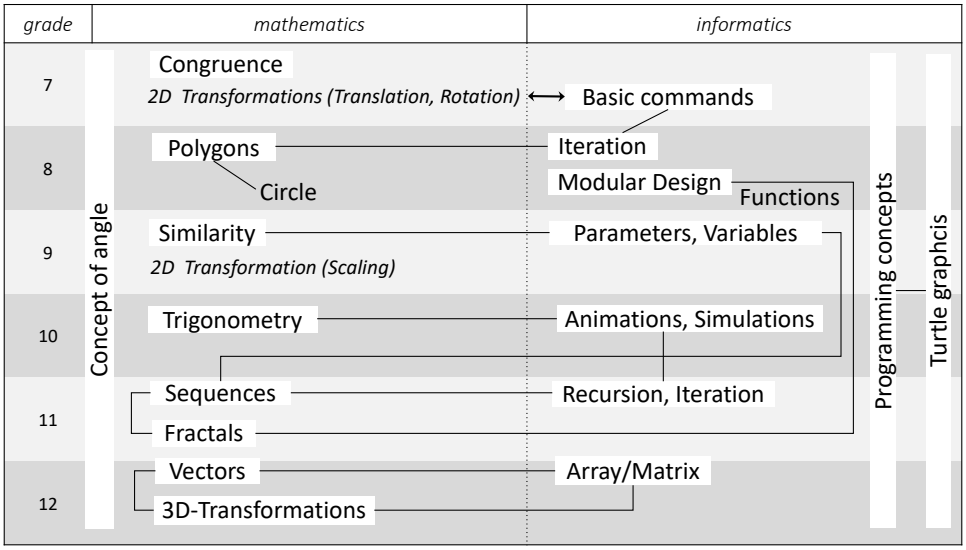
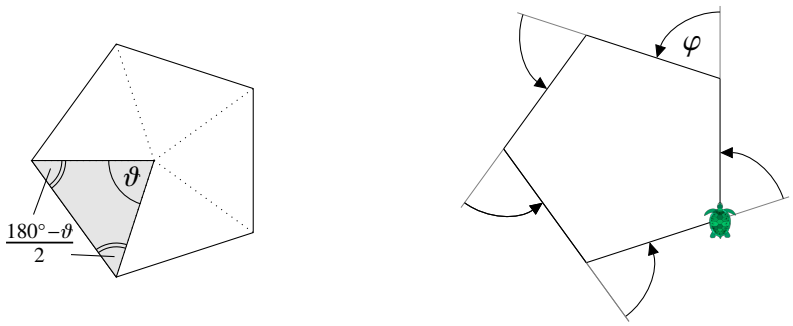


Figure 1: Turtle graphics from 7th to 12th grade

The immediate visual feedback and constructivistic approach of turtle graphics make it a perfect fit for the introduction of *programming concepts* such as a modular design, iteration, selection, parameters and variables etc.; note however, that we

have to be careful with the concept of variables in mathematics and variables in informatics [11]. Moreover, turtle graphics allows for a very vivid approach towards viewing mathematical concepts (mainly from geometry) from a different angle. Geometry classes of 7th and 8th grade are suited very well to be taught in combination with turtle graphics.

The notion of an *angle* constitutes a very challenging topic as it has a number of different manifestations [4, 14]. The angle as the result of two rays with a common starting point from elementary geometry is joined by the angle as the rotation of a ray around its starting point. It has been demonstrated that pupils develop flawed conceptions and thinking patterns, which is why the notion of an angle should be investigated both systematically and holistically in school [15]; and turtle graphics lends itself as an extremely fruitful investigation tool. A classical example from 8th-grade mathematics is the *interior angle theorem* and its generalization to (convex) polygons. The preparation for approximating circumferences is typically done by studying the *regular n-gon*. Drawing such a polygon introduces the concept of iteration and also gives the pupils the opportunity to understand outer angles as rotations. Furthermore, the pupils make the discovery that the Turtle rotates n times by the same angle φ (using the command `left(`φ`)`) and ends up at its starting position; see Figure 2b. This leads to the observation that, in this case, the sum of



(a) Approach using basic triangles (b) The Turtle on its way via exterior angles

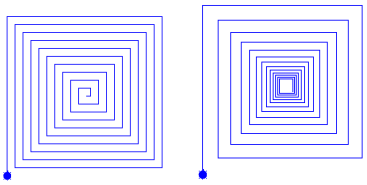
Figure 2: Drawing n-gons with the Turtle

all outer angles amounts to $360°(= n \cdot \varphi)$, and the sum of the inner angles can be formulated as $n(180° - \varphi) = n \cdot 180° - n \cdot \varphi = n \cdot 180° - 360° = 180°(n - 2)$. The classical approach arguing by using the angles of the (isosceles) "basic triangles" (as shown in Figure 2a) provides a less intuitive access; of course, it is desirable to have the pupils discover it as an alternative.

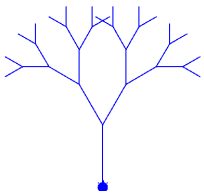
Pupils who want to draw a circle using the Turtle will use the same technique that is used for a regular n-gon; using a number of corner points that is as large as

possible. This leads to discovering the principle of *limit processes* and the relation between the circumference and the perimeter of regular n-gons (with a large number of corner points). There are many other common concepts in the curricula of both subjects, such as similarity transformations (mappings that preserve distances and angles, for instance *scaling*) and animating a growing figure using parameters and variables.

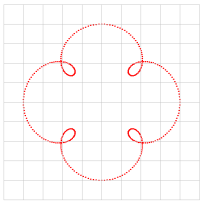
Turtle graphics can also be used for advanced classes and topics. *Sequences and series* and *fractals* can be very nicely explained together with (new) programming concepts such as *iteration* and *recursion*, and again give pupils the opportunity to discover and study these notions on their own; see Figures 3a, 3b and 3d. The Turtle recursively drawing a tree on the screen can be observed and studied step by step. On top of that, graphics that are almost impossible to describe algebraically can now be implemented; for instance, *cycloids*; see Figure 3c. Usually, pupils have fewer problems with visualizing recursive processes than with their formal definition.



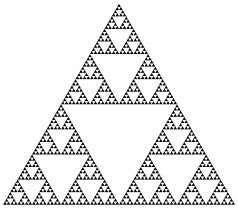
(a) Arithmetic and Geometric Sequences



(b) Recursion



(c) Epizycloid



(d) Sierpinski Triangle

Figure 3

For higher grades, turtle graphics offers a great alternative to using coordinate systems (as is done in algebraic geometry), enabling pupils to describe geometric objects using simple commands like `forward()` and `right()`. Designing a figure using turtle graphics gives a local description of an object, as the Turtle always acts on its current position, instead of a description within a global coordinate system. A geometrical object is described using a dynamic process instead of an equation

or a system of equations; for instance, instead of the formula $x^2 + y^2 = r^2$, we obtain an algorithm that uses the Turtle to draw a circle. As a result, in higher grades, turtle graphics allows to create vivid *animations and simulations* (independent of a coordinate system) that allow making areas like *trigonometry* and *vector geometry* more comprehensible. Excellent inputs can be found in the textbook "Turtle Geometry" [1].

3.2 Building Bridges with Euclid

In primary school, pupils learn how to calculate with fractions, which is then resumed in grades 7 and 8 when the notion of *divisibility* is introduced; see Figure 4. At this point, mathematics classes include calculating the *greatest common divisor* (gcd) of two natural numbers, and, of course, *prime numbers*. However, while most

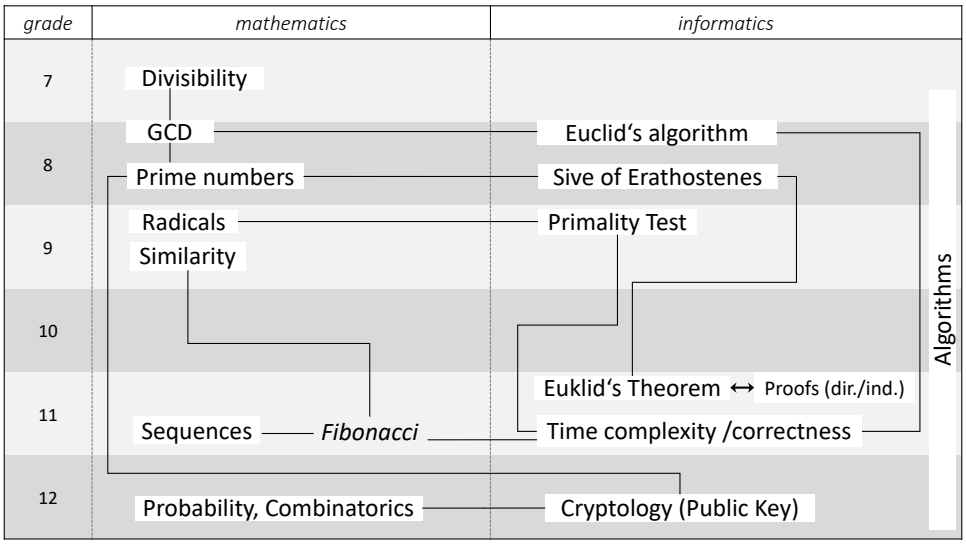


Figure 4: Euclid, prime numbers, and algorithms

pupils still know the definition of prime numbers years after having finished school, they are usually not aware of why these numbers are so important or of how to check whether a given (large) number is prime. Prime numbers are perceived as bookish and to be not much more than a tool to compute, for instance, the gcd. A well-coordinated curriculum is able to fix this.

Euclid's algorithm plays an important role in the genesis of both subjects. The exact formalization of a "method to compute the gcd" builds a bridge to the notion *algorithm*. After all, this systematic method is about 2 000 years old and is therefore considered to be one of the oldest nontrivial algorithms we know. Due to time constraints, in mathematics classes, the focus is usually put on the simple

calculation of the gcd. Conversely, in parallel informatics classes, the emphasis can be placed on a primality testing algorithm, which can then be implemented as a computer program.

The mathematical concept of *divisibility* automatically leads to the notion of prime numbers. The *sieve of Eratosthenes* allows for a very intuitive approach towards prime numbers; here, the pupils can do mathematics in an independent and empirical way. In primary school, pupils have been dealing with multiplication tables, and at this point they learn an algorithm to find prime numbers. The sieve of Eratosthenes, as an algorithm, gives a starting point for interesting questions for advanced classes.

According to Wagenstein, Euclid's proof that there are infinitely many prime numbers is an "indispensable piece of any mathematics curriculum" [16]. Studying it, pupils can learn about an example of direct and indirect proofs, which they again discover themselves. Euclid himself used a direct approach, whereas today the proof is mostly given using contradiction.

From grade 9 on, the concept behind a simple primality algorithm is comprehensible, and even first steps towards optimization (for instance, only testing up to the square root of the input number) can be understood. Doing so, the pupils not only revisit the notion of the square root, but while implementing the algorithm also learn about the *modulo operation* as an extension of the already known *division with remainder*. One of the benefits is to discover that even the computer struggles with testing larger prime numbers. With this, the students get a first idea of what "efficiency" refers to, and that even small changes in an algorithm, such as only testing until $\lfloor\sqrt{N}\rfloor$ instead of $N-1$, for an input number N, makes a difference [13]. One of the most important points is how important prime numbers are for our digital society. Every teacher of mathematics will remark that prime numbers are critical in data encryption, and thus online banking etc.; but this is just an abstract note so far. Thanks to informatics, prime numbers and their applications and importance become a lot more tangible and their study constitutes an exciting challenge.

From grade 11 on, Euclid's algorithm can be revisited; this time, we can formally prove why the algorithm indeed solves the task of computing the gcd of two given numbers; that is, we give a *correctness proof*. When dealing with complexity, we can again follow the genesis of the question of efficient algorithms. In 1841, the French mathematician Jacques Binet for the first time investigated the effort of executing Euclid's algorithm. It turned out that the worst case is met when two consecutive *Fibonacci numbers* are given as input; this can be made to fit well into the content of both grade 9 and 11. As mentioned above, in 12th-grade cryptography classes, prime numbers can be studied once again when discussing, for instance, public-key cryptography.

Algorithms and Complexity. Instead of starting with Euclid's algorithm, an alternative entry to *complexity theory* can start with *fast exponentiation* and the *square-and-multiply* algorithm, which builds a bridge to the mathematics topic of exponentiation. Here, the pupils discover that a certain number of multiplications seems necessary in order to compute a^n.

From grade 11 on, pupils know about logarithms and should be able to discover the bound $\lceil \log_2 n \rceil$ themselves. Then again, logarithms and computing them intimidates quite a number of pupils. On the one hand, this is due to an abstract introduction to logarithms in mathematics classes, which defines logarithms as the operation inverse to exponentiation (which was only developed by Euler). On the other hand, computing logarithms is usually done using exercises that involve exponentiation within growth or decay functions; here, commonly at an early stage, the natural logarithm (with base e ≈ 2.7181) is considered, which is even more of a mystery. In such situations, pupils tend to memorize ways to solve such problems without really understanding what is behind.

A more tangible introduction to logarithms can be achieved using, for instance, *binary search*, where an initially given problem of size n is reduced by a factor of 2 in each iteration. Another approach is to use a binary tree modelling the rounds of a game, where n players compete in a knockout tournament. The relation between the rounds and the number of players is then given as the binary logarithm $\log_2 n$. Moreover, the binary logarithm is used frequently in the analysis of algorithms. This is not simply the result of computing with binary numbers, but also, for instance, due to analyzing strategies that use some sort of 2-way branching. If some problem allows for n possible outcomes (solutions), then each iteration again (as in binary search) divides this number by a factor of 2; see Figure 5. Therefore, the number of iterations until a solution is found can be computed by

$$\frac{n}{2^i} = 1 \iff i = \log_2 n \, .$$

For pupils of grade 12, it is already possible to understand the basic principle of the *big-O notation* if it is explained carefully and using examples; furthermore, the quantifiers have to be used in a nonformal way. The benefit is the repetition of the graphical representation of different functions such as $O(\log n)$, $O(\sqrt{n})$, $O(n \log n)$, $O(n)$, $O(n^2)$, and $O(2^n)$ and their behavior, as shown in Figure 6. It gets particularly interesting when comparing two examples in $O(n)$ and $O(n^2)$, respectively, where the quadratic function is smaller up to a specific number of operations. In general, *sorting algorithms* (selection sort and insertion sort for grade 9, mergesort or quicksort for grade 11) are very good examples to study concepts such as complexity or recursion.

In the context of both primality testing and sorting algorithms, turtle graphics can also be used to investigate algorithmic complexity [12].

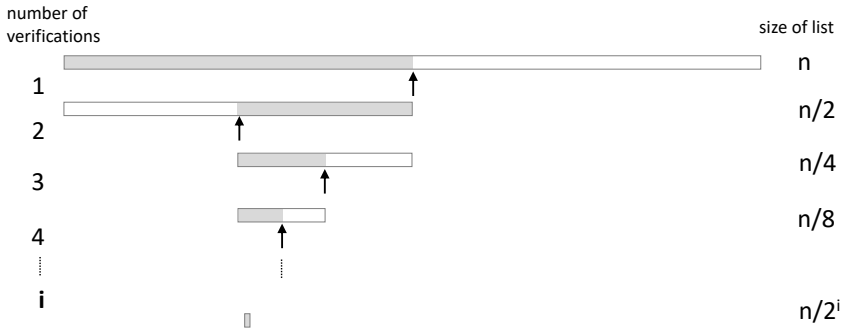


Figure 5: Binary search

$f(n)$ \ n	10	100	1000
$\log_2 n$	3.32	6.64	9.97
n	10	100	1000
$0.5n^2$	50	5000	$5 \cdot 10^6$
2^n	1024	$1.3 \cdot 10^{30}$	$1.1 \cdot 10^{301}$

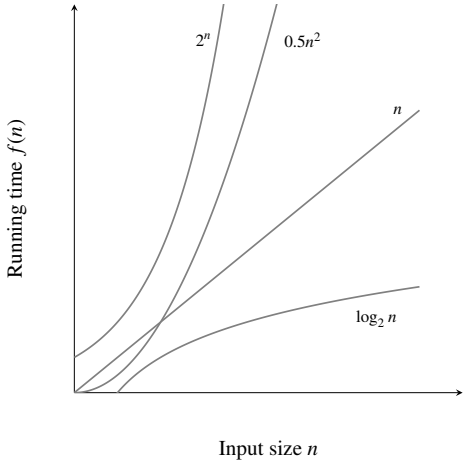


Figure 6

Numerical Methods. Numerics is a branch of mathematics that is inconceivable without computer aid. However, it is unfortunately not a fixed part of mathematics curricula, although it is commonly used in specialization (higher-level) subjects (*Schwerpunktfach*) and physics (error analysis); and it would thus be desirable to strengthen the pupils' basic knowledge of numerical methods. In informatics, the most important algorithms can be treated also in their historical context. As an example, in 11th-grade calculus, their concrete implementation can provide a very vivid approach towards limits and approximations. These topics are very complex for pupils as they have to imagine limits in an abstract fashion or witnessing them using trivial examples, but without truly experiencing the effects. Instead of memorizing derivation and integration rules, laws of symbolic differentiation and integration can be discovered experimentally. Other important numerical methods are approximating square roots by *Heron's method*, approximating roots of functions using the *regula falsi* method, *Newton's method*, the *Gaussian elimina-*

tion method, or *Horner's method*, up to computing *Taylor series* and differential equations. It is very desirable to see some of these important topics that connect mathematics and physics being introduced to all pupils; not only those with specialization in higher-level subjects (*Schwerpunktfach*). More excellent examples are provided by the textbook "Computer-Mathematik" [3].

Cryptology. Studying ciphers and how to break them offers an incredibly large variety of synergies to mathematics, which we will only outline in what follows. Following our basic design principle of mimicking the genesis of the topic (see Section 2.1), we can start with simple cryptographic protocols in grades 7 and 8, such as CAESAR, SKYTALE, and POLYBUS; as an application of percentage calculation, an easy cryptanalysis (more specifically, a *frequency analysis*) can be conducted [7]. If someone wants to improve the CAESAR cryptosystem so that letters are mapped to arbitrary letters, valid keys correspond to *permutations* of the 26 letters, given that each letter may only appear once. With this, it is therefore possible to introduce basic notions of *combinatorics*. When encrypting through masking (e.g. One-time pad), the pupils can also discover the binary representation of numbers and eventually even the (secure) cryptosystem *one-time pad*. Modular arithmetic and frequency analysis are ideal applications of mathematics classes.

What happens if a fault (for instance, accidentally inverting a single bit) occurs while encrypting using One-time pad? This and related questions can be discussed addressing error-correcting codes [9]. In grade 12, public-key cryptosystems can provide exciting insights into both algorithmic and mathematical considerations that are crucial for our everyday life [2].

4 Conclusion

With the elaboration of the new curricula, we should take the chance of building bridges and using synergies between mathematics and informatics. The content should be interdisciplinary and implemented into classes using a spiral curriculum. Mathematics is both a research tool and a precise language that allows us to formalize and verify fundamental informatics concepts; concepts that are not simply presented as complete methods one has to memorize, but that are developed and discussed with respect to their genesis. There are numerous other topics we could have described here. Of course, it is impossible to coordinate everything, but it surely is a step in the right direction to implement a selected number of examples. If we succeed to coordinate and link mathematics and informatics in a spiral approach, both subjects will greatly benefit, and the topics covered will become an exciting journey of discovery.

References

[1] H. Abelson and A. diSessa: *Turtle geometry: The computer as a medium for explaining mathematics*. Boston MIT Press, 1986.

[2] K. Freiermuth, J. Hromkovič, L. Keller, and B. Steffen: *Einführung in die Kryptologie*. Vieweg+Teubner Verlag | Springer 2011.

[3] W. Gander: *Computer-Mathematik*. Birkhäuser Verlag, 1986.

[4] H. Hadas, R. Hershkovitz, and B. Schwarz. The role of contradiction and uncertainty in promoting the need to prove in dynamic geometry environments. *Educational Studies in Mathematics* 44:127–150, 2000.

[5] J. Hromkovič: *Berechenbarkeit*. Vieweg+Teubner Verlag | Springer 2011.

[6] J. Hromkovič: Homo Informaticus. *EATCS Bulletin* 115, 2015.

[7] J. Hromkovič: *Einfach Informatik – Strategien entwickeln*. Klett & Balmer Verlag, Baar 2018.

[8] J. Hromkovič: *Einfach Informatik – Daten darstellen, verschlüsseln, komprimieren*. Klett & Balmer Verlag, Baar 2018.

[9] J. Hromkovič, L. Keller, D. Komm, G. Serafini, and B. Steffen: Entdeckendes Lernen am Beispiel fehlerkorrigierender Codes. *Log-in* 168:50–55, 2011.

[10] J. Hromkovič and T. Kohn: *Einfach Informatik – Programmieren*. Klett & Balmer Verlag, Baar 2018.

[11] T. Kohn: Variable Evaluation: an Exploration of Novice Programmers' Understanding and Common Misconceptions. In *Proc. of SIGCSE 2017*, pages 345–350.

[12] T. Kohn and D. Komm: Teaching programming and algorithmic complexity with tangible machines. In *Proc. of ISSEP 2018*, to appear in Lecture Notes in Computer Science, Springer-Verlag 2018.

[13] D. Komm and T. Kohn: An introduction to running time analysis for an SOI workshop. *Olympiads in Informatics* 11:77–86, 2017.

[14] I. Kontorovich and R. Zazkis: Turn vs. shape: teachers cope with incompatible perspectives on angle. *Educational Studies in Mathematics* 93:223–243, 2016.

[15] K. Krainer: *Lebendige Geometrie. Überlegungen zu einem integrativen Verständnis von Geometrieunterricht anhand des Winkelbegriffs*. Doctoral Thesis, Universität Klagenfurt 1989.

[16] M. Wagenschein: *Naturphänomene sehen und verstehen. Genetische Lehrgänge. Das Wagenschein Studienbuch.* H. Chr. Berg (editor), pages 220–227, 2009.