13 Learning Tech
Tidsskrift for læremidler, didaktik og teknologi

LÆRE
MIDDEL
DK

Computationel Literacy in Schools

Cognitive, social and material aspects

By Roland Hachmann

Korrekt citering af denne artikel efter APA-systemet
(American Psychological Association System, 7th Edition):
Hachmann, R. (2023). Computationel Literacy. Cognitive, social and material aspects. *Learning Tech – Tidsskrift for læremidler, didaktik og teknologi*, (13), 1-27.
DOI: 10.7146/lt.v8i13.149529

Abstract

This conceptual article is situated in the cross-section between computer science and didactics. The article contributes to the research field by taking on questions regarding the role of materialities in relation to learning computational literacy in school. Computational literacy is here defined as patterned deployment of skills and dispositions that are applied in a technology-mediated and situated context with the aim of achieving a valued goal.

The article sets out with a brief outline of the emergence of computational literacy as a concept. Then moving on through a more in-depth presentation of the concept in which three intertwined aspects, the cognitive, the social, and the material, unfold. The article concludes with two examples of computational literacy in a school, and how materiality here plays an important role in relation to the subject-content and pedagogical approach.

Denne konceptuelle artikel indlejrer sig i snitfladen mellem datalogi og didaktik. Artiklen bidrager til det danske forskningsfelt ved at rejse en begyndende diskussion om materialiteters rolle i forhold til det at lære computationel literacy som et delvist grundlag for elevernes teknologiforståelser i skolen. Computationel literacy er i artiklen defineret som færdigheder og dispositioner, der bringes i anvendelse i en teknologimedieret og situeret kontekst med det sigte at opnå et værdsat mål.

Artiklen indledes med en kort skitsering af begrebets fremkomst og derefter en udfoldelse af computationel literacy som et begreb, hvor tre sammenflettede aspekter: det kognitive, det sociale og det materielle udfoldes. Artiklen afsluttes med to eksempler på computationel literacy i en skolekontekst, og hvordan materialitet her spiller en vigtig rolle i forhold til de stof- og indholdskriterier, der er afsættet for undervisningen.

Computationel Literacy in Schools

Cognitive, social and material aspects

By Roland Hachmann, UCSYD

Introduction

In this article, I will introduce and argue for the necessity of computational literacy in discussions about using computational technology in schools. My aim is to raise awareness that working computationally is more complex than the cognitive problem-solving strategies often associated with computational thinking (e.g., pattern recognition, problem decomposition, abstraction, and generalization).

The approach I am inspired by and want to develop in a Danish context originates from American educational researcher and IT learning theorist Andrea diSessa. In his book Changing Minds (2000), he defines the concept of computational literacy for the first time as a material intelligence (diSessa, 2000, p. 5), which is understood as:

> A socially widespread patterned deployment of skills and capabilities in a context of material support (...) to achieve a valued intellectual goal.
> (DiSessa, 2000, p. 19)

DiSessa emphasizes that, in addition to the cognitive aspect, computational literacy also includes social and material aspects. Together, these three aspects form a whole, and the computational relates not only to problem-solving strategies but also to expressing oneself creatively using the computer's capabilities, including coding and programming. In this contribution, I am particularly focused on the material aspects of computational literacy and how they are important for selecting content and learning materials in school. The reason I am particularly interested in this is that literature regarding these and related themes shows no significant focus on this.

Computational Empowerment as an experimental subject and a lack of focus on materiality

Computational empowerment is an emerging concept in Danish primary schools. It has been introduced through an experimental subject in the period 2018-2021. The experiment aimed to develop a subject focused on empowering students to live and prosper in a digital world (Tuhkala et al., 2019). The basic idea was that this could be done by working with and creating digital artifacts and technologies. In other words, the basic argument behind the subject is that digital empowerment is achieved through the ability to use specific technologies in specific contexts, computational thinking, digital design and fabrication. The subject draws mainly on the scientific disciplines of informatics (Caspersen et al., 2018), computer science (Caeli, 2021) and participatory design (Dindler, Smith & Iversen, 2019).

There is strong inspiration from object-based programming (Caspersen, 2022), which is characterized by modelling phenomena in the world through the design of computational models and digital artifacts (Madsen, Møller-Pedersen & Nygaard, 1993) in relation to problem-solving of complex problems. The Scottish professor of informatics and computer science Paul Dourish emphasises that the digital has a materiality that intervenes in the physical world (Dourish, 2017). This thinking is also reflected implicitly in the object-based approach to programming, where computational models precisely aim to be usable and useful for human activity in the world. Dourish also points out that the digital has specific properties that set the framework for what is possible to do:

> The materialities of information are those properties of representations and formats that constrain, enable, limit, and shape the ways in which those representations can be created, transmitted, stored, manipulated, and put to use – properties like their heft, size, fragility and transparency.
> (Dourish, 2017, p. 6)

In the Danish literature surrounding Computational Empowerment (see e.g. Unge Pædagoger, 2020 or LearningTech, 2021), there is a preponderance of focus on *why* and *how* students should, for example, design, model, and program. However, there is a lack of interest in *what* they do it with and which opportunities, and limitations the robot, the programming environment, or the design models have. This deficiency becomes even more pronounced when research studies focus on themes related to teaching subjects (didactics) and pedago-

gies. Here, my argument will be that it is not enough to examine *why* and *how* the students participate in teaching and learning processes, but also to consider the implications of *what* they engage with and through in the processes that unfold.

By presenting and discussing the concept of computational literacy in a Danish context it is this absence of focus on materiality that I am trying to highlight.

The article is structured in the following way. First, I will give a brief historical outline of the background that led to the emergence of the concept of computational literacy. I do this to give the reader a feel for the educational movements and attempts to implement the use of the computer in school – both internationally and in a Danish context.

Next, I will explain my understanding of the concept of computational literacy and along the way give examples of a speculative nature. These are mainly based on coding as an activity, and I have chosen to use Scratch as it is well- known in school.

In conclusion, two concrete examples are given from own empirical studies in schools, where computational literacy is in focus. Since this is a conceptual article with examples and not a report of empirical findings, the article does not have a method or analysis section per se. The examples only serve the purpose of concretizing my arguments on a more operational level.

From computer literacy towards computational Literacy

Literacy is a debated term that both refers to general reading, writing and arithmetic skills and at the same time also refers to special ways of understanding what these skills are and how they are used in certain contexts (Gee, 2015).

Both in and outside of Denmark, since the 70s there has been great enthusiasm about the role of the computer and how it would fundamentally change the way teaching is done at school. There have been attempts to establish a subject around computer learning in school (Fisher, Frøkjær & Gedsø, 1972) at the same time as various attempts to develop programming languages and tools that could teach students to program a computer (Andersen, 2010). As Papert explains in his book Mindstorms, the assumption was that the use of computers would give students objects-to-think-with (Papert, 1980; Brennan, 2015), which could mediate a different and more creative thinking in the students in relation to understanding and solving the academic problems.

Historically, however, it has been shown that even if the computer remained in the classroom, the focus on teaching students to program disappeared (Caeli & Yadav, 2019; Caeli, 2021). Instead, the computer was increasingly seen as a tool that should support the existing practice through more efficient and appropriate ways of working. Rather than programming, the focus became that the students should have a basic competence and basic skills in relation to the use of the computer as a functional tool. The approach was what Molnar (1979) calls Computer Literacy. That is, a comprehensive mastery of, among other things, writing programs, database programs and spreadsheets as well as basic skills in relation to a functional use such as obtaining, storing and sharing information (Haigh, 1985).

In Denmark, it became compulsory in 1990 to teach students about Electronic Data Processing (computers) in all school subjects. In the following approx. 20 years, various initiatives were initiated to, among other things, ensure students' computer literacy. For example, a Junior Computer Driver License was launched in the mid-90s as the Danish answer to the European Computer Driving Licence.

There was also a focus on ICT in the subjects' management documents. A look into the various curricula and teaching guides (see https://www.digitalelaereplaner.dk/) shows that there was a focus on describing the place of information technologies in the subjects as useful tools that could support the training of professional skills. In some subjects, computing was also regarded as a sweeping impulse for change that raised new questions and demands for the subject. For example, in the teaching guide for the Danish subject at the time, you can read:

> In our time, computers have come to play a significant role in every person's linguistic reality, with significant consequences for both form and content. In connection with computers, new forms of communication have arisen, and the language and conceptual world of information technology raises new questions within ordinary language teaching. All this makes information technology a natural part of the Danish subject. The computer can also be a useful work tool at all grade levels, where it can open up new opportunities

> for training language skills, for process-oriented teaching and for collaboration and creativity in the writing process.
> (Ministry of Education, 1990)

The excerpt from the teaching guide above shows how computers created a renewed look at communicative situations and thus also opportunities for new ways of expression. In other words, they were considered a formatting force on the general linguistic reality of humans and thus also created a need in school to have an eye for new skills and understandings of the properties and possibilities of computers.

Although it is not the main focus of the article, it is worth noting that alongside the interest in computers, there was also a great focus on new media (electronic multimedia in particular) and their impact. Here, too, from the mid-90s, it became a focus in the subjects that, through analysis and production, the students had to learn to relate critically to the new media reality. When this is explicitly brought up here, it is because "IT and media", when you look chronologically at the publication of the curricula, came to be a dominant focus about the school subjects from 1995 until 2018 - rather than computing. The computer and programming as a subject field took a back seat in favor of a focus on the communicative properties of multimedia in and across school subjects (Christensen & Tufte, 2010).

In 2016, however, a breakthrough occurred when the government, with Strategy for Denmark's digital growth, focused on "that future generations should become skilled users of IT through understanding, developing and analyzing IT" (Ministry of Business, 2016). In this way, it would be ensured that the individual could not only participate in the digital society of the future, but also help create it. In continuation of this, several initiatives were launched, including a 4-year pilot program to test different models for strengthening technology understanding in Folkeskolen. One of the initiatives was the experimental subject to which I initially referred.

"Technology understanding in the Folkeskolen" (Tuhkala et al., 2019; Wagner et al., 2020) was set aside for experimenting with different academic approaches in several Danish primary schools through the testing of a total of 110 prototype courses (Slot et al., 2021). In various ways, these prototype courses were supposed to give students the opportunity to work with and understand technologies through, for example, technology analyses, digital fabrication and programming.

With the experimental subject as a starting point, robots, microcomputers and programming software have again received a special focus, and the students must now be able to go behind and create with and through technology. Rather than simply using, they must now be able to understand how the content is designed with specific intentions. The students must therefore not only possess a functional computer literacy, but also be able to see through and relate critically and constructively to the technology. At the same time, students must understand, think and express themselves in situated practices, mediated by the technologies they surround themselves with. This is thus a return to some of the intentions that were already advanced in the 60s in Denmark with, among other things, one of the pioneers of computer science, Peter Naur. The difference between the data science of the time and now is that the special focus is on data, data representation and data processes, which according to Naur should be the core of the subject (Naur, 1965; Caeli, 2021) has been expanded with, among other things, a special design expertise.

Computational Literacy

In his book 'Changing Minds', Andrea diSessa argues for a shift in focus from computer literacy to computational literacy. He distances himself from the whole idea of a predominantly skill-oriented use of the computer. At the same time, he also challenges Papert's more individualistic approach to learning, where the computer (an object to think with) is seen as a cognitive partner - a "thing in itself" (Papert, 1987).

Instead, diSessa proposes computational literacy based on three aspects: a cognitive, a social, and a material aspect. The cognitive aspect deals with how externalising representations can enable and support certain understandings and ideas. The social aspect highlights how computational literacy is socio-culturally and historically anchored, wherein specific norms and values emerge. Lastly, the material aspect pertains to the idea that the things (Kragelund & Otto, 2005) we surround ourselves with offer unique opportunities and constraints in expressing ourselves with and towards others.

The material aspect – things we have and things we do

Since the material turn at the beginning of the 21st century, research fields such as Science and Technology Studies (Danholt & Gad, 2021) and material culture studies (Kragelund & Otto, 2005; Hicks & Beaudry, 2010) have emphasized the meanings of materialities in relation to human life and being in the world. Materialities are seen in these

research environments as an important aspect, as they are part of a relational link between people and the environment. Materiality thus becomes a concept that covers both the materiality of physical things (things we have), but also as the practices (things we do) that unfold in relation to both physical and non-physical things such as computer software.

The material aspect of computational literacy focuses primarily on externalized representations, including symbols and signs, through which we as humans can maintain aspects of our thinking and social practices in a way that makes them reproducible, transportable and manipulable. In other words, it is about retaining parts of our thinking and experiences, so that through their concrete and material form they can be transformed, shared and negotiated within the communicative practices we participate in at different times.

The things we do something with consist of certain representations and have certain possibilities of use, interpretive frameworks, conventions, and rules. This means that when talking about computational literacy, there are outer limits to what it means. The concept is not all-encompassing in a school context and is not just about being able to read, write, or calculate on a computer in general. On the other hand, we are discussing specific professional practices and activities where different types of hardware or software are included.

Coding and programming are things we do in a program on a computer using a specific language and specific actions. From a material perspective, a focus would be on the properties and conditions the program and the computer embed in the activity.

Depending on which programming language you code and program in, there are special grammatical and syntactic rules that you must follow. There is thus a lexical system and an alphabet, where certain signs, words and terms mean something completely specific and are used in a way that differs from other writing contexts in the school's practice. Inserting an incorrect character or giving an incorrect command has direct consequences for the execution of the program on the computer or in the robot.

In addition, different coding genres and aesthetic modes of expression are also instrumental in challenging, expanding or maintaining the student's opportunities to express themselves or make objects do something specific.

The visual and block-based programming tool Scratch is based on several specific ideas about what programming and coding for children is and how this should be learned (Maloney et al., 2010). From a learning theory perspective, Scratch is based on the same basic

constructionist idea as, for example, LOGO (Papert, 1980). The student must construct their knowledge through concrete experiences and activities – often in a learning environment with others who can test and give feedback. Although Scratch can be used for relatively complex tasks, the purpose is to introduce programming and coding to students with no previous experience or prerequisites for this. This has had an impact on the design and how various functions are concretised the user interface (Kafai & Resnick, 1996, p. 3). The basic idea is that it should be easy for the student to express creative ideas. This is attempted to be achieved by the students becoming familiar with coding through visual representations and simple functions that can be gradually expanded when the students experience a need for this in line with their mastery level.

An example of the complexity reduction is the way Scratch works with variables. Variables are often difficult to understand for beginners because they covertly perform underlying tasks in a program. It requires a relatively high level of abstraction on the part of the student to understand how their content (values) is retrieved at specific times in the program. Therefore, a design feature in Scratch is that the visual surface makes visible what the variable contains (numbers or text) and what it links to when the program is executed.

From a more technical perspective, newer versions of Scratch build on JavaScript, a procedural and object-oriented programming language that uses dynamic typing as its default rule set. This means that there are very specific rules associated with how to make the program execute commands on the computer, but the program still executes despite less use of variables and inconsistent coding. In connection with coding, characters such as: =, #, ", / or {} have specific grammatical meanings, just as certain words and phenomena such as script, loop, div, function, value and variables have a very specific linguistic and syntactic meaning.

In Scratch, students do not directly encounter this text-based code. Instead, the coding elements are represented through different blocks (movement, appearance, events, control, registration, etc.) that can be put together as building blocks that can carry out commands either visually on the screen (e.g. make a cat move) or through physical extensions such as Micro:bit, Lego Spike Prime, Makey Makey and others. The various blocks in Scratch have different colors and shapes that partly support the students' orientation and coding process, but at the same time are also subject to very specific syntactic rules for how the blocks must be put together to get the program to run as intended. Even if students don't see it on the surface, Scratch is still coding and strings of code. Students must learn to follow certain rules, logics, and

conventions embedded in the programming environment.

The material perspective of computational literacy becomes interesting here because it is precisely a pattern-recognizing use of skills and dispositions in a materially supported context. By seeing the characteristics and conditions of the coding language and building or restoring their coding skills, the students are enabled to express themselves in a meaningful way when they must solve professional challenges that require coding and programming.

It is pattern recognition understood in the way that students across situations and activities must decode and resituate their knowledge and skills in relation to the requirements of the coding activity and the tools they have available.

Scratch is a piece of software and is thus not physically tangible. Nevertheless, it has material properties that consist in the possibilities and limitations (some of which are outlined above) the program offers in relation to, for example, creating, sharing, storing or manipulating representations of information (Dourish, 2017). In this way, Scratch becomes both something we have at hand and can be used to interact with our physical world (e.g. get a robot to draw), but also something that gives birth to certain practices, where the student does something specific or expresses himself in specific ways in relation to academic content.

The social aspect and the school's subject cultures

As indicated at the outset, computational literacy also has a social aspect. The social aspect implies the notion that literacy is always aimed at something, i.e. embedded in a social and situated context.

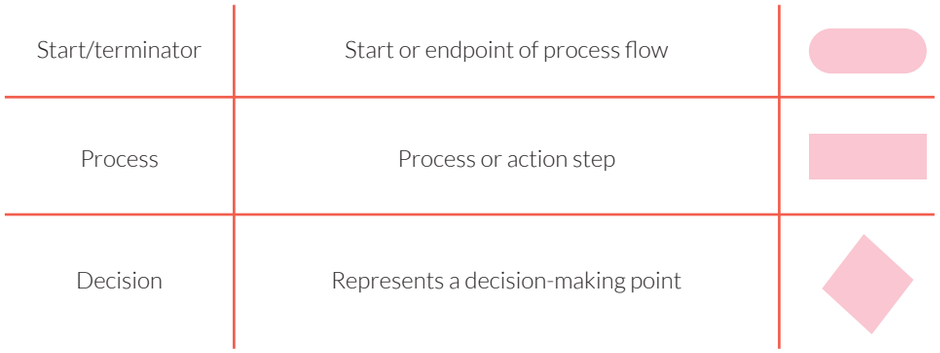
This makes computational literacy a dynamic concept that always reflects the socio-cultural context in which one finds oneself. Patterned deployment in diSessa's definition, which I translate in this article to pattern-recognizing use, emphasises that it is precisely a boundary that separates computational literacy from other forms of literacy. There is a difference between coding in Scratch or writing texts in Word, even though both take place on a computer program. Scratch and Word have (with the material aspect in mind) different properties that support different modes of expression. Although in both cases it is a writing practice based on writing literacy, the two activities must be understood and approached differently, and based on different professional views of what writing is and can contribute to in certain communicative situations.

The social aspect also considers that not all academic challenges and their solutions are legitimate focal points within the framework of the subjects at school. Relevant challenges and solutions are determined by professionals in the school's context and evaluated through the subjects and the associated subject disciplines. It is in didactic contexts that content areas are recognized as important and legitimate or rejected as irrelevant. Computational literacy in school will, therefore, always be associated with special academic practices and phenomena in teaching that are recognised. This makes the work on computational literacy at once diverse and narrow. Narrowly because these are not skills that can simply and seamlessly be transferred between contexts and professional practices. These are certain logics and methods that must be transformed by the students across professional understandings, professional views and integrated into situated practices.

At the same time, it is a question of professionalism and just as it is not only in the Danish subject that you read and write or in mathematics that you calculate, but a computational literacy is also based on some skills that can be used across different subjects. For example, the programming of a robot can both have a scientific aim, but also be based on more artistic and creative processes in a subject such as visual arts. Again, the student's computational literacy will be brought into play in different ways and with different purposes.

On a more superficial level, computational literacy, like other literacies, also has a situated and dynamic relationship with a changing society. School and subjects are in constant development and thus also different understandings of what subjects and professionalism are. An example of this can be found in the current trial subject for technology understanding in primary school, which is also referred to earlier in the article. One of the test models was technology understanding as integrated professionalism in the existing subjects, where the students were taught through the formulated prototype courses in various topics on the border between on the one hand the existing subjects and on the other hand skills within design and informatics. Specifically, the students work in one of the prototype courses "Have we caught a real monster?" (Kiær, Godtliebsen, Lorenzen, Nielsen & Nissen, 2020) with algorithmic processes and automation through flowcharts in the Danish subject (1st class). Students are presented with selected symbolic representations within the flowchart genre: Terminal point, process and decision (see Figure 1 below). These are used to describe different types of processes. At the end of the course, the students must create flowcharts themselves as part of an instructional text that describes how to catch a monster.

Figure 1.
Excerpt from the prototype.



Flowcharts originally originated in engineering and from the late 1940s became part of computer science as a way of describing and designing computer programs based on (Goldstine & von Neumann, 1947). Without developing an in-depth discussion of flowcharts as a form of representation, it is, however, an example of a genre niche (diSessa, 2001, p.24) in the subject that requires students to read and write in a new way and recognize the rules and logics (patterns) that are nested. The students can partly draw on experiences from other reading and writing practices, but at the same time must also learn the flowchart genre's own grammar.

If the gaze is raised above a level of activity, does the prototype process open a discussion of whether concepts such as automation, algorithmic thinking and flowcharts belong to the content and genres of the Danish subject? The prototype is of course part of an experimental program - a reinterpretation of the Danish subject that intends to add a different professional dimension. Nevertheless, the point here is that the social aspect in relation to computational literacy includes a discussion about whether the professional community around the didactics of the Danish subject recognizes the forms of representation and expression, genres and practices that come with it.

The cognitive aspect and computational thinking

The cognitive aspect of computational literacy draws on the notion that the things we use and do something with are connected to the way we think and perceive the world around us. This should not be understood as deterministic, but to a greater extent as that things and thinking are not separate, but rather dialectically connected in that things and thinking in different ways create and are created by each other.

The Danish computer scientist Peter Naur already emphasized in the 60s the importance of understanding the tools (things) that are at hand and how these things in specific situations will frame people's thinking, their perception of problems and how they are solved. From a computational literacy perspective, it is clearly traced here how the material and the cognitive are intertwined. A central component of Naur's theory is a symmetrical relationship between tools, people and problems (Naur, 1965; Caeli, 2021). He emphasizes that people's solution to a problem will involve the use of those tools (physical or non-physical) that are available and considered appropriate for the task. An important point is that if the tool is changed, the problem will be approached differently or even perceived in new or different ways. The framework of tools here means both enabling thinking and action, but also that they can limit us by keeping our focus on specific possibilities and thus prevent us from seeing others. At the same time, according to Naur, the choice of a preferred tool depends on how a problem is understood by the people involved and what a desirable solution might be. Stated to the point, Naur's thinking in this context contributes to the fact that problems exist in people's minds and tools designed to solve problems that are not recognized by anyone are meaningless.

A more recent concept that has gained a central place in relation to technology understanding in school is computational thinking (Denning & Tedre, 2019; Yadav & Bertelsen, 2022). The term relates specifically to cognitive aspects of problem solving and was revitalized when Jeanette Wing published her essay Computational Thinking (Wing, 2006). Although a formal definition of computational thinking is still debated in various research fields around education and school, there is some agreement in the research literature that computational thinking covers some basic cognitive skills, including abstraction, analysis, problem decomposition, pattern recognition, modeling and algorithmic thinking (Grover & Pea, 2013; Jacob & Wagenshauer, 2018; Kafai & Proctor, 2021).

Wing's recent definition of computational thinking as: "the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried

out by an information-processing agent" (Wing, 2011, p .20) points out the desire for a set of general cognitive skills that are transferable across different disciplines. Wing's arguments that students in school should learn to think like a computer scientist revived the discussions previously raised by both Papert, diSessa, Naur and others about computational thinking in relation to problem solving across different professional domains in school and human life (Tucker, 2014; Jacob & Wagenshauer, 2018).

At Wing, the primary focus is on ways of thinking that lead towards an automation process rather than carrying out the automation process yourself with various tools at hand. In other words, Wing differs from, for example, Papert by focusing on "learning to compute" rather than "compute to learn" (Caspersen et al., 2018; Dohn, 2021). Wing (and many others after her) are mainly focused on cognitive strategies in relation to problem solving.

From a computational literacy perspective, however, there is more to the cognitive aspect than problem-solving strategies. Expressing oneself is associated with much more than problems and their solution. It can equally be about expressing imagination, creativity, hope or taking a critical stance. Here it is about sharing one's ideas and thoughts with others through the computational things, physical and non-physical, that are available. Problem formulations and solutions can of course also be a sub-element here, but not a goal in itself. In line with Bers (2021), the relationship between being able to formulate and problem-free on the one hand and expressing oneself on the other must be seen as a continuum. From a computational literacy perspective, the student should move between these extremes when, for example, they code on the computer. Scratch can of course be used to teach students problem-solving strategies such as debugging, problem decomposition, sequencing, pattern recognition, automation, etc. But Scratch is also a programming environment where students can express themselves creatively and create content that reflects understanding, wonder, emotion, identity and imagination (Kafai & Proctor, 2021; Bers, 2021; Kafai, Peppler & Chapman, 2009). This can be done both through the programming of small games, but also other forms of expression such as animated films, music or interactive stories.

Coding in Scratch requires that students partly understand the program's syntax (the material aspect as argued earlier) and partly be able to translate their thoughts and ideas computationally. It is thus a matter of duplicity. A student may well be familiar with and skilled at coding in Scratch, but this is not necessarily the same as being able to model and transform creative ideas into something that can be expressed through coding. The same applies the opposite way, where the student can have really good ideas, but fails to use the program to express them. Put another way: A student who is good at writing is not necessarily a good writer, just as a student who is good at sudoku is not necessarily good at mathematics - and vice versa.

From a cognitive aspect, conditions are set for students in relation to decoding and modeling phenomena so that they can be expressed or interpreted through the use of computational things.

Computational literacy in school

Based on the three aspects, computational literacy must be defined as certain ways of thinking (the cognitive) and acting in a situated context (the social), supported by things (the material). Teaching in and through computational literacy in school should therefore be based on considerations that reflect these in relation to the content and subject selection.

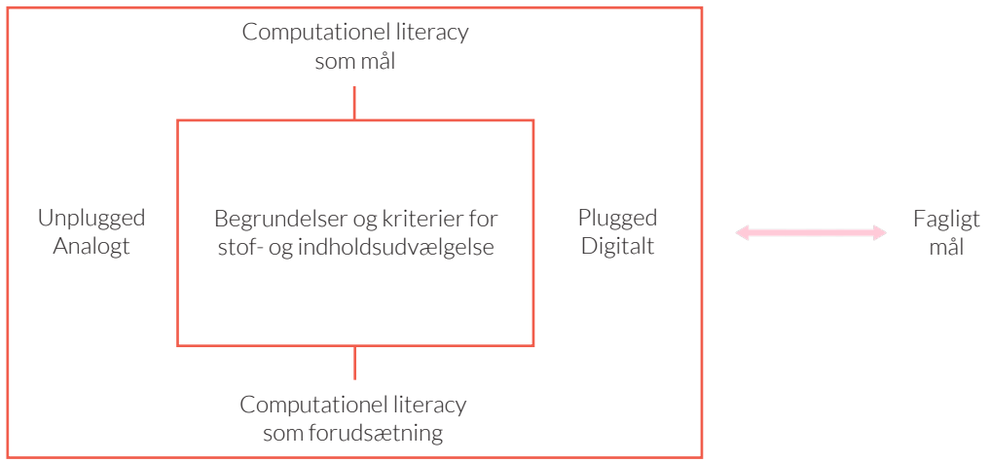
Programming is cognitively abstract and requires a broad foundation and understanding of symbol manipulation and coding and can in this light be compared to proof in mathematics or advanced analyzes of literature in the Danish subject (Lu and Fletcher, 2009). In both cases, it requires that the students have a well-developed literacy. To achieve this, computational literacy in school should be developed through many different approaches (Kafai & Proctor, 2021), which focus on students' prerequisites and dispositions in relation to a gradual progression between wholes and parts. This could mean, for example, that the teaching was based on students both directly and indirectly encountering the concepts and areas of competence linked to computational literacy in and across the subjects. Coding in Scratch, which has been the overall example in the article, could be one of many approaches to teaching students basic programming skills and the ability to create and recreate symbolic representations through experiments and computational methods.

The computational in school can be both a primary academic focus, but also somewhat secondary (Tannert, Lorenzen & Berthelsen, 2022). As a primary focus, learning computational literacy becomes a goal, and the material and content the students encounter supports this very purpose. As a secondary focus, it becomes to a greater extent a prerequisite for being able to solve other professional challenges through the inclusion of different computational strategies. Furthermore, the prominent research literature emphasizes the need to work through both plugged and unplugged approaches in teaching so that

cognitive and bodily perspectives are used through the use of both analogue and digital things (Brennan & Resninck, 2012; Grover & Pea, 2013, Mikkonen, 2021; Caeli & Yadav, 2019; Dohn, 2021).

In Figure 2, I have tried to create a simple graphic representation that illustrates how justifications for subject and content selection (what and how) can be related to considerations about the role of computational literacy and which approach to work with computational methods is emphasized.

Figure 2.
Justifications for subject and content selection from a computational literacy perspective.



This simplistic illustration is only intended to draw attention to the connection between content perceptions, forms of activity and computational literacy in teaching. The illustration covers far from the complexity of the teaching. My purpose is simply to point out that the teacher's framing of activities with, for example, coding and programming games in Scratch reflects a subject and content selection that mainly emphasizes the use of computational literacy in a primarily plugged/digital programming environment. Whereas the previous example from the prototyping course with the use of flowcharts reflects an acquisition of computational skills through an unplugged/analog approach.

Examples from school

In the following, I will give two examples of computational literacy in school. This is a selection of our own empirical studies, and the examples serve in this article as an elaboration of the concept as well as Figure 2. That is to exemplify computational literacy as resp. goal in itself in relation to problem solving (example 1) and as a prerequisite for participating in other professional activities that are based on more creative expressions (example 2). At the same time, the material aspect is central in both examples, both in the form of manipulations of physical things, but also using digital tools.

Example 1: Escape Puzzles as a way to learn about Micro:bit

An example of a more physical analogue approach to problem solving and programming with computational literacy as a goal has previously been described and developed in Forfatter (2022a; 2022b). Here, as part of a longer course on Micro:bit, students participate in a scenario-based Escape Puzzle (see figure 4). The activity has a purpose to give students insight into the various sensor options embedded in a Micro:bits and their application logics. Through the activity, the students must physically manipulate various wooden boxes, in which a Micro:bit with a specific programmed sensor property is stored. During the process, the students get hints from one of the boxes in the form of words on a display such as: "Loves metal", "Light" or "Speed", which should help them with the decoding. Only by activating all sensors at once can the students complete the task.

As tangible and physical things, the wooden boxes have the property that the sensor properties of the Micro:bit require physical, bodily actions by the student.

Figure 3.
Photo from empirical study before and during the activity (Hachmann, 2022a)

As Mikkonen (2019; 2021) has shown through studies of 'Bodygramming', where university students through body-based and collaborative activities simulate the computer's algorithmic processes, bodily experiences with such abstract phenomena are an important prerequisite for understanding the computational perspective in relation to, for example, programming a microcomputer. Mikkonen emphasizes that bodily play and social activities illustrate the differences between perspectives and forms of abstraction for a computer and its user respectively.

Understanding these differences makes it easier for students to learn programming because they understand the perspectives and forms of abstraction that programming addresses. In the same way, the Escape Puzzle activity is based on the students discovering certain functions and possibilities of the Micro:bit as a material thing through a bodily experience. The three aspects of computational literacy are expressed by the fact that they 1) must decode the material properties of the boxes and physically manipulate them rather than coding in e.g. Scratch, 2) they use strategies based on collaboration and common understandings for the approach to the problem, 3) problem decomposition, pattern recognition and algorithmic thinking embedded as part of the cognitive aspects.

The course also reflects a subject view (the social aspect) that emphasizes that students learn basic concepts and skills through a playful and experimental approach.

The material nature of the wooden boxes and the name 'MysteryBox' make the activities both analyses, interpretations and joint reflections in relation to the solution of the task. The long-term goal is the students' acquisition of a computational literacy, but this is done here based on the idea of a sequential problem solving and progression, where the students gradually and in a movement from concrete to abstract understanding learn basic concepts, methods and strategies that are basic academic and fundamental in order to could use the Micro:bit as a tool on a more abstract level later in the course.

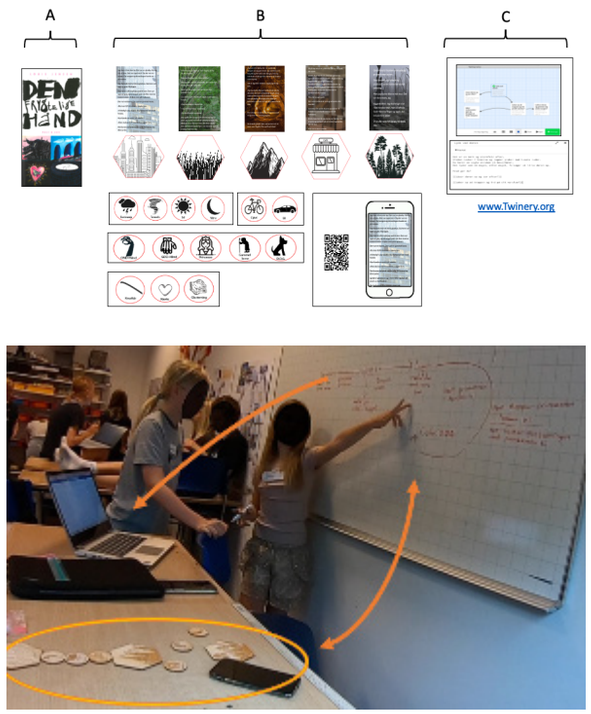
Example 2: The horrible hand – teaching writing through coding

The second example I will use here focuses on computational literacy as a prerequisite that supports other professional learning. The example is based on the teaching of writing in Danish at intermediate level, where students in 5th-6th grade. class participates in process-oriented writing and co-authorship of a fantastic narrative. For this, a didactic design was developed (see Figure 5 below) that includes both plugged/analogue and unplugged/digital perspectives.

The students are first read an extract from The Horrible Hand (Jensen, 2001), which is a novel within the genre fantastic story (A). After this, the students must use several physical wooden pieces (B) to generate ideas and structure their story, which will be a continuation and conclusion of the read text. On the back of the five large wooden pieces is a QR code that gives students the opportunity to read selected parts of The Horrible Hand as inspiration. The smaller round pieces represent different elements, people or objects from the story. At the end of the process, students must create interactive narratives in the online application Twine (C). Here, the students must create an interactive hypertext story, where the reader is the main character and along the way, through hyperlinks, chooses what the main character in the story does.

The computational part consists in the students having to make use of a mixture between the written language they normally know and html coding, as a form of expression. Twine uses an html encoding and students must be able to understand this programming format to create their narratives. For example, to create new passages, they must use square brackets around a text or use html code to embed images, change the text color, etc.

Figure 4.
Graphic illustration of the didactic design (left) and its realization in a 5th grade (right)



From a computational literacy perspective, the materiality of the wooden blocks constitutes a limitation in relation to the students' idea generation. At the same time, the pieces, through their material nature, enable the students to use sequential and algorithmic thinking, which gives them the opportunity to decompose their narrative into smaller parts and arrange the content in different ways in relation to the patterns they recognize in the excerpts from the original story. This preparatory work trains the students in computational strategies and methods while at the same time enabling a certain degree of corporeality and reification in relation to the structuring of narratives

in their narrative. The movement from B→C in Figure 4, which is also marked with orange arrows, shows that the students are remaking sign systems. It is described by Selander & Kress (2012) as a transduction process where from a computational literacy perspective, the materiality of the wooden blocks acts as a constraint on students' idea generation. At the same time, the blocks enable students to employ a sequential and algorithmic thinking through their material nature, allowing them to decompose their narrative into smaller parts and arrange the content in various ways based on the patterns they recognize in excerpts from the original story. This preliminary work trains students in computational strategies and methods, while also enabling a degree of embodiment and objectification in structuring narratives in their storytelling. The transition from B to C in Figure 5, also marked by orange arrows, shows how students transform sign systems. This process, described by Selander & Kress (2012) as a transduction process, involves students transforming one representational form into another through various forms of symbol manipulation.

Concluding remarks

As was pointed out at the outset, my mission with this article is to introduce the concept of computational literacy and through this to establish a basis for forward-looking discussions of the concept in a school context. There are areas in the article that are touched only superficially and that require more in-depth exposition. The article must be seen as a start to this. It has been important for me to make room for concrete examples, because in the forward-looking discussion there is a need for empirical examples to link more theoretical perspectives to.

Through the article, I have tried to substantiate theoretically and through the examples to show that the computational should not only be understood as cognitive problem-solving strategies, but also contains social and material aspects that are equally important to consider - and which I believe are overlooked. As I have been around, there is a focus on what the students must learn or create, how and why, but there is very little focus on what the things this is done with have properties and implications for process and product. Programming and coding are what I would call a computational design practices that

imply a knowledge of the materiality of the material with which one designs. This applies both in relation to, for example, software and hardware.

A concept such as computational literacy with the three underlying aspects can frame a language around the integration of computational perspectives in both new and existing literacy practices. DiSessa's approach has been, for me, a viable way to unfold these aspects. As I have explained, the term arose as a reaction against an instrumental use of the computer in school and as an attempt to emphasize the computer as an opportunity for children and students to express themselves creatively and in new ways that could not be done before. I see a danger that the same instrumental approach to technology is repeated in the operationalization of technology understanding in school. Students learn about complex problems, problem solving, design methods and technological agency, while the creative and imaginative seem to be downplayed. In the Danish school, there is not a long tradition of focusing on computational, coding and programming. With the experimental subject technology understanding as a way of establishing new skills in school, a look at the computational seems inevitable. In recent Danish research literature in the area, however, there seems to be a tendency, with Wing's definition of computational thinking as a point of departure, to reduce the computational to the students' ability to solve problems cognitively. This has consequences for the knowledge aspects that are worked with. In example 1, there is a high degree of corporeality involved in the students' (de)coding, just as in the example I refer to bodygramming as a computational method. These approaches go beyond cognitive problem-solving strategies: *"the thought process involved in formulating problems and their solutions"* (Wing, 2011) or computational thinking as it is called in the experimental subject.

One area that the social aspect of computational literacy opens is whether coding and programming have legitimacy in relation to the school's purpose. One argument for computational literacy would be that digitalisation, data and technology are integral parts of students' lives in and out of school. Although in the last 50 years there has been a look at IT in school, in the last 5-10 years there has been an acceleration in digitization and a focus on the significance of algorithms, automation and data for our lives. This gives rise to rethinking parts of the educational task at school. In this context, the school's task is not to train the students to become computer scientists or engineers, but to give the students a basic understanding, literacy and a democratic competence in relation to the society and the education system they encounter outside the school. This requires, among other things, that they get a basic understanding of what, for example, algorithms and automation are and how they are created with different purposes that ultimately influence their lives. This can be done both directly and indirectly through a focus on problem solving and design processes, but it is equally important to focus on students' opportunities to develop imagination and give them the opportunity to be creatively expressive in relation to their critical thinking.

As with other literacies, it takes time to develop a computational literacy. Just like reading and writing, it requires that you do it in many different contexts and in many ways. The field is new and there are many questions of both an empirical and theoretical nature that remain unanswered and require research. Nevertheless, there is a history and preliminary work that enables a discussion of computational literacy as an object field for further research in school and educational contexts. As I have argued, it is necessary to look more broadly at the computational in school, especially if it is to continue to have a justification. Different literacy concepts are already part of the school's practice and subjects. It must be investigated and discussed whether computational literacy as a concept can impart something valuable in relation to the focus on technology understanding.

References

Andersen, T. (2010). Den digitale revolution: Fortællinger fra datalogiens verden. Datalogisk Institut, Københavns Universitet.

Brennan, K. (2015). Objects To Think With. Constructivist Foundations, 10(3), 313-314.

Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. 25.

Bers, M. U. (2021). Coding as a playground: Programming and computational thinking in the early childhood classroom (Second edition). Routledge.

Caeli, E. N., & Yadav, A. (2019). Unplugged Approaches to Computational Thinking: A Historical Perspective. *TechTrends*. https://doi.org/10.1007/s11528-019-00410-5

Caeli, E. N. (2021). Computational Thinking in Compulsory Education: Why, What, and How? A Societal and Democratic Perspective. PhD Dissertation. Graduate School Arts, Aarhus University.

Caspersen, M. E. (2022). Informatics as a Fundamental Discipline in General Education: The Danish Perspective. I H. Werthner, E. Prem, E. A. Lee, & C. Ghezzi (Red.), *Perspectives on Digital Humanism* (s. 191-200). Springer International Publishing. https://doi.org/10.1007/978-3-030-86144-5_26

Caspersen, M. E., Iversen, O. S., Nielsen, M., Hjorth, A.,& Musaeus, L. H. (2018). *Computational Thinking – Hvorfor, hvad og hvordan?* It-vest – samarbejdende universiteter.

Danholt, P., & Gad, C. (2021). Videnskab, teknologi og samfund: En introduktion til STS / Peter Danholt og Christiopher Gad. Hans Reitzel.

Denning, P. J. & Tedre, M. (2019). *Computational thinking.* The MIT Press.

DiSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy* (1. paperback ed). The MIT Press.

Dindler, C., Smith, R. C. & Sejer Iversen, O. (2019). *En designtilgang til teknologiforståelse*. Dafolo

Dohn, N. B., Hansen, J. J., & Hansen, S. B. (Red.). (2020). *Designing for situated knowledge transformation*. Routledge.

Dohn, N. B., Mitchell, R., & Chongtay, R. (Red). (2021). *Computational thinking: Teoretiske, empiriske og didaktiske perspektiver*. Samfundslitteratur.

Dohn, N.B. (2021). Computattional Thinking – indplacering i et landskab af it-begreber. I Dohn, N. B., Mitchell, R., & Chongtay, R. (Red). *Computational thinking: Teoretiske, empiriske og didaktiske perspektiver*. Samfundslitteratur.

Dourish, P. (2017). The Stuff of Bits: An Essay on the Materialities of Information. MIT Press.

Fischer, C., Frøkjær, E. & Gedsø, L. (1972). *Datalære i skolen—Om data og edb i samfundet*. G.E.C. Gads Forlag.

Hachmann, R. (2020). *Didactic Design for Transformations of Subject-content Knowledge: An investigation of student teachers' transformations of subject-content knowledge between professional education and practice* [PhD Dissertation]. University of Southern Denmark.

Hachmann, R. (2022a). The Cyber Weapon: Decomposing Puzzles in Unplugged Computational Thinking Practices with Computational Objects. *KI – Künstliche Intelligenz*, *36,* 59-68. DOI:10.1007/s13218-022-00756-8

Hachmann, R. (2022b). Cybervåbnet: computational praksis i 8. klasse. I: S. S. Fougt, J. Bundsgaard, T. Hanghøj & M. Misfeldt (red.), Håndbog i Scenariedidaktik (p. 489-501). Didaktiske studier Nr. 8. Aarhus Universitetsforlag. DOI:10.2307/j.ctv33jb48k.37

Gee, J. P. (2015). The new literacy studies. I J. Rowsell & K. Pahl (Red.), The Routledge Handbook of Literacy Studies (1. udg., s. 35-48). Routledge. https://doi.org/10.4324/9781315717647

Goldstien, H. H. & von Neumann, J. (1947) *Planning and coding of problems for an electronic computiong instrument*. The Institute for Advanced Study Princeton, New Jersey

Grover, S. & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Haigh, R. W. (1985). Planning for Computer Literacy. The Journal of Higher Education, 56(2), 161–171. https://doi.org/10.2307/1981664

Hansen, J. J. (Red.). (2018). *Digital skrivedidaktik*. Akademisk Forlag.

Hicks, D. & Beaudry, M. C. (Red.). (2010). *The Oxford handbook of material culture studies*. Oxford University Press.

Jacob, S. R., & Warschauer, M. (2018). Computational Thinking and Literacy. *Journal of Computer Science Integration*, *1*(1). https://doi.org/10.26716/jcsi.2018.01.1.1

Jensen, L. (2001). *Den frygtelige hånd*. Høst & Søn.

Kafai, Y. & Resnick, M. (1996). Constructionism in Practice: Designing, Thinking, and Learning in a Digital World. Lawrence Erlbaum Associates, Inc.

Kafai, Y. B., Peppler, K. A. & Chapman, R. N. (Red.). (2009). *The Computer Clubhouse: Constructionism and creativity in youth communities*. Teachers College Press.

Kafai, Y. B., & Proctor, C. (2021). A Revaluation of Computational Thinking in K-2 Education: Moving Toward Computational Literacies. *Educational Researcher*. https://doi.org/10.3102/0013189X211057904

Kiær, K., Godtliebsen, A., Lorenzen, R.F., Nielsen L. & Nissen, A. (2020). *Har vi fanget et monster?* 09.06.22: https://xn--tekforsget-6cb.dk/wp-content/uploads/2020/09/fanget-et-monster-1.kl_.-dansk-15.09.20.pdf

Kragelund, M. & Otto, L. (Red.). (2005). *Materialitet og dannelse: En studiebog* (1. udgave). Danmarks Pædagogiske Universitets Forlag.

Lu, J. J. & Fletcher, G. H. L. (2009). Thinking about computational thinking. *ACM SIGCSE Bulletin*, *41*(1), 260-264. https://doi.org/10.1145/1539024.1508959

Madsen, O.L., Møller-Pedersen, B. & Nygaard, K. (1993). Object-Oriented Programming in the BETA Programming Language. Addison-Wesley.

Maloney, J., Resnick, M., Rusk, N., Silverman, B. & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education*, *10*(4), 1-5. https://doi.org/10.1145/1868358.1868363

Mikkonen, J. (2019). Bodygramming. Embodying the computational behaviour as a collective effort. *The Design Journal,* 22, 1423-1437.

Mikkonen, J. (2021). Kropsbaseret Computational Thinking. I N. B. Dohn, R. Mitchell, & R. Chongtay (Red.), Computational Thinking – Teoretiske, empiriske og didaktiske perspektiver. Samfundslitteratur.

Molnar, A.R. (1979). The Next Great Crisis in American Education: Computer Literacy. *Journal of Educational Technology Systems*, 7, 275-285.

Naur, P. (1965). The Place of Programming in a World of Problem, Tools, and People. *Information processing 1965*, 195-199.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York: Basic Books

Papert, S. (1987). Information technology and education: Computer criticism vs. technocentric thinking. Educational Researcher, 16(1), 22-30

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Selander, S. & Kress, G. (2012). *Læringsdesign – I et multimodalt perspektiv*. Frydenlund.

Slot, M. F., Hachmann, R., Hjorth, M., & Von Sehested, M. (2021). Teknologiforståelse - en sammenhængende faglighed: En beskrivende analyse af 110 undervisningsforløb. *Learning Tech*, *10*, 296-322. https://doi.org/10.7146/lt.v6i10.125600

Soon, W. & Cox, G. (2020). Aesthetic programming: A handbook of software studies. Open Humanities Press.

Tannert, M., Lorentzen, R. F. & Berthelsen, U. D. (2021). Computational Thinking as Subject Matter. I A. Yadav & U. D. Berthelsen. (Red). *Computational Thinking in Education*. Routledge. https://doi.org/10.4324/9781003102991-5

Tucker, D., McCowan, F., Deek, C., Stephenson, J., Jones, J. & Verno, A. (2006). A model curriculum for k-12 computer science: Report of the acm k-12 task force computer science curriculum committee. Technical report, Association for Computing Machinery, New York, NY.

Christensen, O. & Tufte, B. (2010) Pædagogik, didaktik og levende billeder – En introduktion. I Martin Brandt-Pedersen, Ole Christensen, & Henrik Poulsen. (2010). *Læring med levende billeder*. Samfundslitteratur.

Tuhkala, A., Wagner, M.-L., Iversen, O. S. & Kärkkäinen, T. (2019). Technology Comprehension – Combining computing, design, and societal reflection as a national subject. *International Journal of Child-Computer Interaction*, *20*, 54-63. https://doi.org/10.1016/j.ijcci.2019.03.004

Undervisningsministeriet. (2018). *Læseplan for forsøgsfaget teknologiforståelse*. https://www.emu.dk/sites/default/files/2019-02/GSK.%20L%C3%A6seplan.Tilg%C3%A6ngelig.%20Teknologiforst%C3%A5else.%20pdf.pdf

Undervisningsministeriet. (1990). EDB i folkeskolens fag: Dansk og EDB *Undervisningsvejledning for folkeskolen.* Undervisningsministeriet.

Wagner, M.-L., Iversen, O. S. & Caspersen, M. E. (2020). Teknologiforståelses rationale: På vej mod coputational empowerment i den danske grundskole. *Unge Pædagoger*, *81*(1), 6-14.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2010). *Computational Thinking: What and Why?* Link Magazine, 6. https://www.cs.cmu.edu/~CompThink/papers/TheLinkWing.pdf

Yadav, A. & Berthelsen, U. D. (Red.). (2022). *Computational thinking in compulsory education: A pedagogical perspective* (First Edition). Routledge.