Three +1 Perspectives on Computational Thinking

Sebastian Mateos Nicolajsen, Magda Pischetola, Paweł Grabarczyk, & Claus Brabrand
Center for Computing Education Research (CCER), IT University of Copenhagen
Denmark

ACM Reference Format:
Sebastian Mateos Nicolajsen, Magda Pischetola, Paweł Grabarczyk, & Claus Brabrand. 2021. Three +1 Perspectives on Computational Thinking. In *21st Koli Calling International Conference on Computing Education Research (Koli Calling '21), November 18–21, 2021, Joensuu, Finland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3488042.3490024

ABSTRACT

Computational Thinking (CT) is a highly contentious subject with many diverging meanings and definitions. This study presents a preliminary literature review of 71 peer-reviewed articles on CT. The papers indicate the existence of five main *aspects* that have historically been used in association with its definition: Algorithm, Abstraction, Modelling, Simulation, and Implementation. Based on this preliminary literature study, semi-structured interviews with eight CT scholars are conducted, in order to evaluate these *aspects* and to identify qualitatively different *perspectives* on CT, which integrate the mentioned *aspects* in different ways. From the interviews, three different *perspectives* emerged, focusing on: **Reasoning**, **Simplification**, and **Automation**. Furthermore, the goal of having computationally educated citizens is extrapolated from the interviews, indicating an additional *perspective* (+1) titled **Empowerment**, which appears as embedded within all the previous three perspectives. This paper proposes to put these three (+1) perspectives in dialogue, in an effort to support researchers and practitioners working with CT across different fields.

1 INTRODUCTION

In recent years, Computational Thinking (CT) has drawn increasing attention, not only in Computer Science education but also across other disciplines and academic fields [80]. It has been used all around the globe to advocate computational knowledge for everyone and defined as a crucial competence to be developed across all educational levels [6, 85]. However, the definition of CT is still heavily debated and controversial.

First, it is still not yet clear how to separate CT from programming [10, 94] and whether a stronger focus should be given to a 'computational' part or to a 'thinking' part. On one hand, discipline-based approaches [61] emphasise the need to spread domain-specific knowledge from Computer Science that is considered useful for other fields [20]. On the other hand, psychology-based approaches [61] emphasise the cognitive aspects related to CT, such as problem-solving [38, 41] and cognitive information processing [58, 95]. The focus on cognitive theory is predominant in literature, probably due to the fact that the first wave of research on CT started in the 1980s when computers were less prolific [50].

Second, CT has been strongly related to teaching and learning. [61] gather these approaches in a category that they define as 'education-oriented.' These perspectives especially emphasise the abilities and skills acquired in relation to CT. In this view, CT-related constructs and tools are implemented to promote so-called CT skills [4, 37, 66, 71], considered among the necessary "twenty-first century skills" [60, 90]. A great amount of literature has been published with focus on practitioners' experiences about teaching CT [1, 22, 48], on how to conceive CT as part of a school subject [12, 13, 65, 83], on the effectiveness of some specific activities on the development of CT skills among students [16, 85, 90], and on the kinds of devices that should be used to learn CT [3, 19, 36]. Some studies offer a proposal about how to operationalise CT facets in educational activities that foster the development of different skills [71]. However, even among education-oriented approaches, there are great theoretical differences, which are often not explicit.

In some cases, CT has been considered as a synonym of 'CT education' *tout court* [5, 56]. In others, CT can be presented as intertwined with information and media literacy [23, 44] or associated with critical thinking [57]. Moreover, as there is still no consensus about which CT skills should be taught at school, there are several challenges regarding both positioning CT in the formal curriculum [76, 83] and assessing CT skills [8, 78], with consequences on teachers' preparation and self-efficacy [79].

From this brief introduction, it is evident that different theoretical perspectives on CT are directing actions towards different, but equally important, practices in learning and teaching. On these grounds, [50] suggest considering these perspectives in dialogue, rather than in opposition. The authors define three theoretical framings of CT - cognitive, situated, and critical - and propose that Computer Science researchers make room for multiple and interdisciplinary perspectives.

Drawing on these insights, this paper starts from the premise that some definitions of CT found in literature are not mutually exclusive, but rather co-constitutive and overlapping. However, despite the great body of literature that has been published on the topic in the last two decades, few studies have undertaken an analysis of common aspects that drive each theoretical perspective and related definition.

This study has the objective to address this gap in literature, by analysing common aspects and theoretical perspectives used in literature to define CT.

2 METHODOLOGY

This study is guided by the following research questions:

Koli Calling '21, November 18–21, 2021, Joensuu, Finland
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8488-9/21/11...$15.00
https://doi.org/10.1145/3488042.3490024

- **RQ1**: *What common **aspects** are used in literature to define CT?*
- **RQ2**: *How do these aspects contribute to different **perspectives** on CT?*

The research used an exploratory qualitative approach [33], in two steps. First, a preliminary literature review was conducted, investigating 71 articles on CT, and providing a historical and contemporary understanding on the concept and its main *aspects*. Education-oriented articles were excluded from the selection, as the focus of the study was not providing information for practitioners, but rather explore the theoretical groundings of CT. No limitations regarding the publication date were taken into account.

To triangulate the results, and further explore the topic, the preliminary study was used as the offset for the second iteration, in which qualitative semi-structured interviews were undertaken with scholars involved in CT studies. The interviewees were tasked to both validate and criticise the *aspects* identified during the preliminary literature review. Interviews were transcribed, coded, and triangulated with the results from the first iteration. This brought the amount of *aspects* down from 8 to 5. The qualitative codes assigned to the interviews were combined into different *perspectives*, which are composed by *aspects* that are considered of relevance by the interviewed scholars [77].

Figure 1 provides a visual summary of the iterations and overall research design:

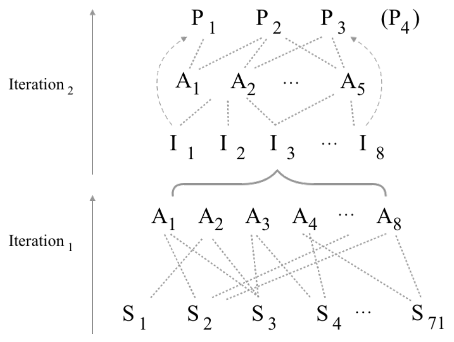

Figure 1: Two iterations (proceeding upwards) 71 (S)ource papers, eight initial (A)spects, eight (I)nterviews, which validated five of the initial (A)spects and formed three (+1) (P)erspectives.

2.1 Iteration 1: Preliminary literature Review

The goal of the preliminary literature review was to gather *aspects* that are related to a definition of CT. The gathering of articles were done in three steps. Firstly, database searches were conducted with the following keywords: "`CT AND (definition OR concept OR theory)`". Then, the references of identified literature reviews were investigated (e.g. [15, 87]). Lastly, the set of collected studies was compared to public lists of CT research papers.[1] The body of studies was filtered to not include education-oriented, as well as practice- and implementation-related studies, since these mostly function as studies *testing* certain theories. We do recognise that such a selection criteria may exclude some theoretical contributions. However, in most cases, these studies address specific sets of skills and educational contexts, without providing a clear definition of CT. Additionally, as the intention was to focus on the integration of different *aspects* that define CT, articles which only examined the depth of one single aspect of CT (e.g. Activities of Abstraction), were not investigated.

After an initial review of the collected data, lines were coded if they centralised any specific *component* or *idea* in their description of CT. Afterwards, lines were compared across articles. If two lines utilised the same term, or described the same idea, they were merged. The ideas which were central to more than single articles, became the initial eight *aspects*. The analysis refers to the method in the Grounded Theory [42], where coding starts from open categories and leads to a systematic relation or integration between the emerging categories.

2.2 Iteration 2: Semi-Structured Interviews

To evaluate the *aspects* that emerged from the preliminary literature review, a semi-structured interview was constructed. The aim of this second iteration was to explore which *aspects* would be mentioned by the scholars as relevant for a conceptual definition of CT, or *perspective*, as we called it. Specifically, the interview aimed to identify (1) the interviewee's *definition* of CT, based on some of the identified *aspects*; (2) any *objections* to other definitions; and (3) what the interviewee believes is the *intent* of CT. Beyond the semi-structured interview, the interviewees were also introduced to the initial eight *aspects*. They were then tasked to evaluate these; which do they find relevant, which could be discarded, and which were lacking. Consequently, three of the initial *aspects* were discarded. An example of this, was the Pragmatic aspect, which closely resembled the Implementation aspect.

The interviewees were eight prominent scholars within CT. They were selected based on their publications concerning CT, as well as involvement in related (CT) research centres. Four of these researchers were, at the time of the interview (2020), employed at Danish universities:

CT Researcher	Affiliation
Andrea DiSessa	University of California, Berkeley
Dor Abrahamson	University of California, Berkeley
Jeannette Wing	Avanessians Director of the Data Science Institute, Columbia University
Matti Tedre	University of Eastern Finland
Michael E. Caspersen	Director, It-vest (3 collaborating universities)
Morten Misfeldt	Head of Center for Digital Education, University of Copenhagen
Nina Bonderup Dohn	Head of Center for Learning Computational Thinking, University of Southern Denmark
Tom Nyvang	Co-Head of Center for Computational Thinking, Aalborg University

Table 1: Overview of interviewed researchers.

The interview (both the semi-structured interview and the follow up exercise) was subsequently transcribed following [86] and coded in two rounds. *Firstly*, the transcripts were coded using *data-driven*

[1] https://csedresearch.wordpress.com/computational-thinking/

qualitative coding and organised in six *categories* to steer the analysis further towards the area under investigation:

Definitions:	*"CT is..."*
Applications:	*"CT can be used for..."*
Implications:	*"CT will impact..."*
Considerations:	*"CT is too..."*
Motivations:	*"CT is important because..."*
Practitioners:	*"CT teachers must be able to..."*

Secondly, the resulting descriptions were coded once more, in the same approach as the articles from the first iteration. The result of this, is the quotes presented in the paper.

Interviews were conducted online; in Danish, for the four Danish-speaking researchers; and in English, for the remaining four, who were all native or fluent in English. For the purposes of this paper, selected quotes have been translated from Danish. All scholars have afterwards received transcripts, translations, and specific quotes for the opportunity to correct them.

3 ASPECTS OF COMPUTATIONAL THINKING

Data analysis from the preliminary literature review underlines the presence of five main *aspects* related to different definitions of CT: Algorithm, Abstraction, Modelling, Simulation, and Implementation. A visualisation of these aspects and how they have appeared to relate to the physical computer, is given in Figure 2. It is here important to notice that this visualisation is a rough approximation of the trends found during Iteration 1. In what follows, we will provide a brief historical overview of the appearance in literature of each *aspect*.

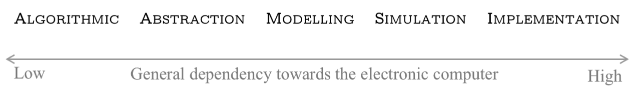

Figure 2: Aspects and their general appearing dependency towards the physical computer.

3.1 The Algorithm Aspect

This *aspect* emerged in the 1960s with researchers such as Snow and Perlis, who argued for the need to introduce new language forms and communication descriptions, in education, to improve algorithmic knowledge [15, 43, 53]. During the 1970s, definitions began to describe what algorithms, and the associated thinking, was [54]. This introduced the need of being able to transform something into an appropriate (algorithmic) representation, as described by Dijkstra [29]. Simultaneously, other researchers began to describe the benefits of 'algorithmic thinking'. Knuth put many of the other aspects investigated, inside of algorithmic thinking, introducing algorithmic thinking as a superset of the others [54]. During the 1990s and early 2000s the idea of studying algorithms, remained central. In 2011, Aho described the ability to formulate problems in an algorithmic manner as CT [2].

3.2 The Abstraction Aspect

Abstraction is often referred to as a central concept of Computer Science [91]. The term is very common in computer science literature around 1965 and 1990. In the 1960s, researchers were arguing for abstractions as fundamental to the benefits of Computer Science [39]. Soon hereafter, and for the next 15 years, researchers began describing abstractions as common concepts to be navigating, dealing with, and developing—when working within Computer Science [29]. Researchers continued to stress the opportunities, and importance, of abstractions [55]. Additionally, researchers—through out the years—suggested a close relation between this aspect, the model, and automation [93]. And while researchers of today continue to underline this aspect of CT, they also stress a need to understand the implications, and consequences, of abstraction.

3.3 The Modelling Aspect

A model is "a simplified description, especially a mathematical one, of a system or process, to assist calculations and predictions [70]. The development of a model is known as modelling. In this aspect we are including both terms, *model* and *modelling*, as they often appear in literature as synonyms. Since the 1950s, researchers such as Fein and Perlis began arguing for the importance of models that are used to design and develop computer systems [35, 53]. Knuth (1985) described the important ability of "representing reality," which today is understood as part of modelling [55]. Papert (1980) continued discussing the properties of the physical computer, focusing on its ability to give us a grounded reference [72]. When Wing (2006) provided her definition of CT, she described a computer/model, which can be both an electronic computer and a human being [92]. This view on models goes against arguments of prior researchers, such as Knuth describing the physical computer as central in exploring the richness of the discipline [54]. Later, researchers also presented an interesting hierarchy between the model, and the algorithm—arguing for representations (models) being the essence, and algorithms being secondary [26]. Aho summarised this aspect by mentioning how models are central to CT and computing in general [2].

3.4 The Simulation Aspect

When simulation emerged as a scientifically-valid tool, definitions did not explicit a major interest in the concept of CT as part of more sciences [24]. However, the literature shows the widespread use of computational simulations in other disciplinary fields after 1960. Even mathematicians found tools to aid them in theorem proving [32]. This was when interest in subdividing teaching of different Computer Science aspects within other disciplinary areas, was acknowledged, albeit for various other reasons; e.g. economy [47]. In the 1960s, Nygaard & Dahl set out to create a generic "simulation system," but ended up inventing the object-oriented programming paradigm wherein classes and objects represent (simulate) concepts and phenomena from reality, "[Simula 67] is [..] intended mainly as a general purpose programming language, but with simulation capabilities" [21]. Document analysis shows that a new growth in interest occurred after 1980, which is when computational science became viable. After the 1990s, the interest appears to decline. This may relate to the fact that sciences had developed independent computational branches - and the separate notion of computational science, was no longer needed. This evolution was truly recognised by the year 2000, when computational literacy was introduced, arguing for CT being able to become a literacy, not only being part of other sciences, but intertwined within all of society [25, 30]. Simulations became even more beneficial, with the new found profileration of machine learning [93].

3.5 The Implementation Aspect

This *aspect* has its roots in Engineering. "Engineers work to apply scientific principles in order to build structures and machines" [28]. Despite their differences in purpose and procedures, Computer Science and Engineering share the process of designing "intended applications" [73]. They both use mathematics and technology to find the best solution to a problem, and implement it [96]. It is worth mentioning that the concept of 'software engineering' is not introduced until 1968, during a NATO meeting in which experts described the current period as a 'software crisis' [64]. In 1967, researchers discussed to whom the computer belonged, describing it as belonging to both science and engineering [69]. Hamming even argued that computing was more related to engineering, than mathematics, which was opposing the predominant argument at the time [47]. When Wing discussed CT in 2006, she—and the frameworks which followed her definition [18]—drew parallels to both mathematics and engineering [92]. However, compared to Hamming, her focus lies on the interaction with the real world, rather than 'balancing of conflicting aims' [47].

4 PERSPECTIVES ON COMPUTATIONAL THINKING

The analysis of the existing definitions of CT helped us identify key aspects that appear in these definitions. While these *aspects* remain to be central elements of CT, it becomes clear through both Iteration 1 and 2 that these do not themselves constitute viewpoints on CT. Rather, they are a *means* (to and *end*) and act building blocks for such viewpoints. Consequently, we will now switch our attention from *aspects* to *perspectives*. *Perspectives* can be seen as reconstructed definitions built from certain subsets of aspects. They are viewpoints which gather and present opinions of different intents and suggestions of *what* CT aims to do, and *how*.

Iteration 2 addressed **RQ2** by organising these results through a qualitative analysis. Semi-structured interviews with scholars supported such an organisation. The results indicate the existence of (at least) three different *perspectives* on CT, with an additional *perspective* (+1) that appears to be embedded in the previous ones. These *perspectives* are defined by a central aim of CT, emerging from the interviews: **Reasoning**, **Simplification**, **Automation**, and **Empowerment**. In Figure 3, they are visualised in a spectrum that represents their *General dependency towards the electronic computer*, whose ends are defined as *Low* and *High* dependency. The **Reasoning** and **Automation** *perspective* are, respectively, positioned at either extreme of the spectrum. **Simplification**, which is a *perspective* in its own right, is located in the middle, but is also utilised by and within the former two *perspectives*. This organisational structure will be motivated and explained in the following along with the combination, relevance, and integration of *aspects* within each *perspective*. Furthermore, Table 2 provides an overview of which articles express the different *perspectives*.

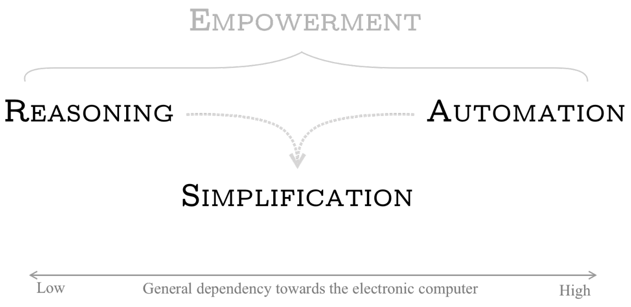

Figure 3: Overview of the three (+1) *perspectives* along with their inter-dependencies.

Perspective	Articles
Reasoning:	[2, 6, 7, 11, 15, 25, 26, 29, 34, 35, 39, 40, 43, 45, 53–55, 59, 61, 62, 67, 71, 72, 74, 81, 82, 87, 92, 93]
Simplification:	[2, 6, 15, 21, 24–26, 29, 30, 32, 35, 39, 43, 47, 53–55, 72, 84, 87, 92, 93]
Automation:	[2, 11, 15, 17, 18, 21, 24–26, 29, 30, 32, 34, 35, 39, 43, 45, 47, 53–55, 63, 69, 72, 74, 89, 92, 93]

Table 2: Overview of which articles express the different *perspectives* (or their *aspects*).

4.1 The Reasoning Perspective

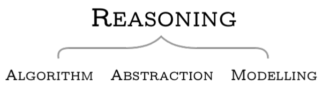

Figure 4: The Reasoning Perspective and its related aspects.

Information technology and Computer Science is ubiquitous. Once trained, its effects and applications become both comprehensible and understandable. We learn to utilise such reasoning, to understand and solve preexisting problems in new and different ways. Some may go as far, as comparing it to the discovery of *gravity*. Even today, this idea is a fundamental part of reasoning about our world and its physical properties - helping us understand when the apple will hit the ground, and how faraway galaxies are orbiting each other. Exactly this construction of new formalizations, and their use in different domains, is central to this *perspective*, which concerns itself with, *applying concepts of Computer Science as a way of processing information and solving general problems.*

Proponents of this *perspective* concern themselves with formalising and synthesizing processes and concepts that enable novel approaches to solving problems, but also understanding problems. Abstraction is at the core of these processes, based on Algorithm—and algorithmic thinking—and Modelling, or "models of computation" [2]. Knuth (1985) noticed that this kind of reasoning differed from mathematics, as it included—besides abstraction—a reduction to simpler problems and information structures, i.e.

decomposition [55]. Tedre and Denning (2016) refer to this understanding of CT as a sort of "mental zoom lens" and underline that such a view was already present in Dijkstra's work in 1974 [87]. Aho (2011) defined such a thought process as one made by "computational steps" [2]. Once constructs are formalised, we can work with them by reducing similar problems to these formal descriptions, abstracting away the details which are insignificant to the construct. We can then apply any *computer* to interpret these constructs according to our formalisation. This last step is what Wing (2008) describes as "mechanizing our abstractions and their relationships," so that a computer will interpret the abstractions [93]. In the interviews, these aspects are mentioned with a focus on how CT reflects the way that a computer scientist thinks, that is, utilising specific cognitive procedures to solve problems:

> People in computing, or whoever works in any way with programming, I think they are computational thinkers, by their profession. So, everybody who looks into the world through a computational lens, in one way or another is somehow a computational thinker.
>
> - Matti Tedre, Int.

> CT is thinking like a computer scientist, and thus, what are the ways, in which computer scientists are trained to solve problems? And then you go through this list that you have, which is basically ways a computer scientist attacks problems. [...] We have a lot to offer to the world. The ways in which we think can help everyone. Regardless of whether you are a Computer Science major, or not, and regardless of what your profession is.
>
> - Jeannette Wing, Int.

A few interviewees have also underlined how this way of understanding CT is related to language; a focus that appears in literature since the 1960s [40, 53, 72]. The languages applicable for constructing such formalisations are prolific and range from "wordy" descriptions and graphical representations (such as data structures or UML diagrams) to domain-specific languages to full-blown general-purpose programming languages. Central to all of them, is their ability to allow us to *formalise* a given process, approach, or problem. According to the following quotes from the interviews, the computer and its feature of programming languages is central to this process, as follows:

> CT is about how you train the mind, and we should ask ourselves, what should people be doing to train their minds better. Should they be studying Latin? Or maybe they should be studying Logo.
>
> - Dor Abrahamson, Int.

> [...] it is related to general cognitive skills, which have been trained in mathematics courses, but which can also be trained in many other different problem spaces.
>
> - Morten Misfeldt, Int.

However, exactly this idea strongly enforces a dependency towards general-purpose programming languages, and thereby, indirectly, on the computer, as part of training your mind. Other scholars argue that the computer is rather an intermediary obstacle which heightens the barrier to entry:

> If you want humanists to think about problem-solving in this way, then it can be a really good idea to introduce a set of problems which can be solved analoguely, so that the computer-part doesn't get in the way.
>
> - Nina Bonderup Dohn, Int.

And exactly this *low dependency towards the physical computer* is most common within this *perspective*. But no matter which stance one agrees with, they both argue that CT is a way to apply Computer Science's ways of reasoning in a more general context. Once applied in this way, it is one specific way of looking at, and solving, problems:

> In the same way, we can say that in economics we use mathematics. Is that then economics or mathematics? Well, it is the mathematical language. You also use Danish and English when writing articles. And writing an article isn't only writing; it's also a realisation process. You know that yourself, when you have written something; if you are really thorough with writing and structuring your thoughts, what you express, then it's also a realisation process. So the computational is also a language.
>
> - Michael E. Caspersen, Int.

4.2 The Simplification Perspective

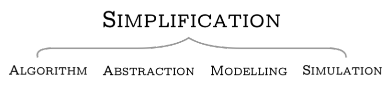

Figure 5: The Simplification Perspective and its related aspects.

Complexity is a barrier to entry for working with many concepts in different domains. However, CT allows us to create objects which simplify our understanding of concepts which are otherwise difficult to comprehend. Objects such as these could be *mas*, which simplify an otherwise incomprehensible amount of data, highlighting the parts relevant to us, such as drawing paths feasible for a hiking trip in an otherwise steep and uncertain environment. This is done mainly through abstractions and representations, defined by models, which in turn allow us - with the computer - to *simulate* certain paths and how they will affect our hike. And exactly this simplification is central to this *perspective*, which is concerned with *reducing the complexity of a problem so that it becomes more approachable.*

Simplified descriptions of objects have increasing relevance to practice, and this is why **Simplification** is both related to **Reasoning** and **Automation**. In this *perspective*, while formalisation - and algorithms - are part of realising simplifications, it is rather the process of Abstraction and simplifying which is central. In 1967, Forsythe explained that one of the reasons the computer is a valuable tool is this capacity to "store the abstractness of the information", be it represented by numbers, letters, punctuation, or in any other way [39]. We can represent anything in a computer, which in turn allows us to comprehend and reflect about the problem or domain that we are addressing:

> Take modelling of dynamic systems, such as waves. It is very difficult mathematics, but modelling it computationally is so absurdly simple that a child in fifth-sixth grade can understand the model, enter it, and manipulate it; gaining ownership of it. [...] Reality is always more rich and nuanced than what we represent, both mathematically and computationally. It's always a map, not a landscape. There is uncertainty in our representation. Then, we can transform as crazy as we want the representation we have, and at some point, we interpret the result.
>
> - Michael E. Caspersen, Int.

Two reflections can be drawn from this centrality of representation. First, CT appears to be connected with understanding the domain that we are aiming to describe computationally. This reminds us of what Perlis highlighted, that is, understanding the computer structure (*model*) is what allows individuals to transform existing structures and create new ones (*modelling*) [53]. The computer also has a physical nature which is grounding any abstract work, as Papert pointed out [72]. The computer model can simulate reality without a mathematisation, and this is, according to Forsythe the great potential of computer models [39]. Second, the uncertainty in our representations has consequences, as the following excerpts clarify:

> [...] a lot of mathematicians and physicists object to computational representations, because that's not the real stuff - the real stuff is calculus, we live in a continuous world. I have gotten this a lot, and I think that's one view, but I think computation is a legitimate representational system, and understood appropriately it's not everything, but most of the understanding I want to convey to students.
>
> - Andrea DiSessa, Int.

> This realisation that it is not simply objective that we have modelled in a certain way, and that our decision has resulted in a specific algorithmic solution of our problem. But, the model has chosen something instead of something else - the algorithm does what we ask, and humans are only subjects.
>
> - Tom Nyvang, Int.

4.3 The Automation Perspective

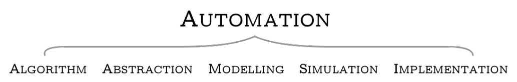

Figure 6: The Automation Perspective and its related aspects.

Technological advancements have through generations improved the every day life of people across the world with Automation, that is, by substituting human effort with mechanical, electrical, or computerised elements [70]. Technology has both simplified and completely removed manual workflows in all domains. Sometimes these automations are invisible to the user and sometimes they are actively used concepts. However, these automations are not necessarily available to all, as they may require training. An example of this could be utilising technology such as a bike. Once you are able to utilise this tool, you will greatly reduce travel time from A to B. Yet, you are still required to control and actively turn the wheels using your legs. As you become more confident and faster on your bike, the benefit of the bike increases. The idea of CT being such a tool for **Automation** is the essence of this *perspective* as it is concerned with *enabling individuals with the capabilities of computational technology.*

In this *perspective*, we see a *High computer dependency* which bases itself on all *aspects.* **Automation** answers the question: how can we use technology to improve the efficiency and/or reliability of something we do or want to do? Alternatively, how can processes be automated by using technology? As in the **Reasoning** *perspective*, here algorithms are central again, not only as processes that "describe and transform information" [63], but as an object of study, aiming at pragmatic applications of mathematics to problems [39]:

> CT is not about the fact that we, as people, should think like computers, but rather how we should think to formalize it for computers.
>
> - Tom Nyvang, Int.

Here, the computer is seen as a *tool* that can provide useful automations across all disciplines:

> A computational approach to science is simply the way the world is now. Computational biology, computational approaches in physics, [...]
>
> - Andrea DiSessa, Int.

> For me, it's a set of competencies involved in developing it-artefacts, which are those we typically mention. But, there are a lot of these, until the point in which we code on a computer that might as well, and most often, happen analoguely - optionally digitally supported. But here, they are simply cognitive processes which are being supported, akin to when I am being assisted, when I write in Word: It simply happens faster. The cognitive processes, until we code, happen just as well, and sometimes better when done analoguely.
>
> - Nina Bonderup Dohn, Int.

> In general, there are a lot of ways of automating data, which haven't got anything to do with programming as such.
>
> - Matti Tedre, Int.

As these last two excerpts show, a large part of **Automation** is not dependant on the computer. However, while the computer is not necessary for **Automation**, it is central in *modern* **Automation**. As with the bike, when reaching a certain level of awareness, we can utilize information technology as an extension of our mind and use it to manipulate the world, which is also part of the "beauty" to some of the interviewees:

> [...] to my personal experiences, when first programming, there's a type of experience of awe and magic; and delight of constructing things, as Papert says: "Rather

> than a computer programming me, I am programming it; the computer becomes my construction material, my sandbox."
>
> - Dor Abrahamson, Int.

> [...] which to me means programming, in some form or another, in order to think and do things. It requires a certain level of technical competence. It should be easier than current environments, that's why I am still busy working on our Boxer project. But then it should centrally, kind of, mediate thinking, and intellectual doing for whatever purposes that you have. And it is really important for me the "whatever purposes" you have.
>
> - Andrea DiSessa, Int.

While it is apparent that the computer may be human, the excerpt above provides us with an example of pragmatic capabilities and its influence on our ability of automating *something*. Properly using computation, will allow us to:

> [...] harness computations for doing all kinds of jobs and tasks.
>
> - Matti Tedre, Int.

The question then becomes if we can automate everything. Is complete **Automation** possible? Does **Automation**—using, e.g., AI—remove jobs?

4.4 The Empowerment Perspective

Common to the three *perspectives* explored, is the goal of empowering society; may this be through new ways of reasoning, simplifying complex ideas, or automating tasks. While such ideas are laudable, the *perspective* of empowering society stretches further than this. It critically examines what it means to be dis-powered, commenting on the consequence of not being able to understand and harness computations. Already towards the end of the 1950s, Snow argued that *"Those who don't understand algorithms, can't understand how decisions are made"* [15]. And this *perspective* of CT emphasises the governing role which digitisation is assuming:

> [...] I think, when we live in a world, or anybody lives in a world governed by algorithms and information flows that are processed by a computer, we will probably benefit from knowing how that data, those information flows, are being automated, and how they are being processed.
>
> - Matti Tedre, Int.

It is also a *perspective* which emphasises the need for action and decision; contemplating about our future:

> How far can automation take us? Can everything be automated? Is there always something important left over that cannot be automated? Will AI displace more jobs through automation than it generates?
>
> - P. J. Denning (2018) [27].

Consequently, CT becomes *a tool to relate critically to the world.* However, while this *perspective* assumes a critical position, it is also a *perspective* of dreams and ideas of the future. What can CT unlock?

> You multiply the quotient by the dividend, and you add the remainder? That's all algorithms! But our fourth grade teacher never uses that word. And there are so many opportunities for children to learn that what they are doing, is following an algorithm. With that concept, one can liberate the thinking, and I could imagine, even a nine year old's way of thinking of different algorithms, to solve the same problem.
>
> - Jeannette Wing, Int.

> I would say that [CT] is, ultimately, really some kind of word for being able to be a productive participant within the collective practice of doing Computer Science.
>
> - Dor Abrahamson, Int.

The very idea that this may be a new literacy is central to this *perspective*: A computational literacy. In this sense, CT is strongly education-oriented and it can and should be integrated in the curriculum at all levels of education. The challenge is to establish which skills are related to each level of education.

> The reason why it may make sense to take this discussion, is to try to figure out what we really know about this complex of goals for teaching. From my perspective, it would be teaching in primary school, high school, but it could also be the university. It's about getting people ready for a digital society. [...] The discussion is after all how much and how. So, should we be able—if we point to concrete skills—to program 40 lines of code in Python that are connected? Then there are some which should be able to, and others who shouldn't.
>
> - Morten Misfeldt, Int.

5 DISCUSSION

Based on the presented results, this section proposes a qualitative discussion focusing on two main takeaways from this research.

First, the study revealed that a spectrum of *perspectives* has always been present in history. In the following, we present a short summary of how the different *aspects* and *perspectives* are intertwined and have changed historically.

In the 1950s and through the 1970s, we see the prevalence of a focus on Abstraction, which is considered at the basis of a particular way of thinking, sometimes related to cognitive procedures, sometimes to the specific scientific approach of Computer Science. Kahn (2017) noticed that this way of thinking has been named 'algorithmic thinking' in the late 1950s and early 1960s, and translated into CT by Wing's essay in 2006 [51]. In our analysis, it is evident that Abstraction—with support of Algorithm and Modelling—is at the core of the perspective we named **Reasoning**. It is also worth noticing that scholars generally associate this *perspective* with a *Low(er)* dependency on an electronic computer.

In the 1980s, Papert raised the interest of education around programming languages for children [72]. This historical moment, through the 1990s, can be related to a prevalence in literature of two aspects: Abstraction and Modelling. The idea of CT is grounded here in the possibility of learning about complex systems by experimenting and simulating reality. The computer has a central role for CT, as it allows for both designing models and applying modelling through IMPLEMENTATION. In our study, the perspective that emerges from the combination of the referred aspects - ABSTRACTION, MODELLING, and IMPLEMENTATION - is **SIMPLIFICATION**. The specificity of this *perspective* is that it does not suggest a barrier to entry, as its goal is "simplifying" the concepts down to a comprehensible level of the end user. The emergence of a focus on simplifying problems reminds us of the fact that these *perspectives* on CT are not mutually exclusive. In fact, as we mentioned throughout the paper, **SIMPLIFICATION** can be considered as part of the other perspectives.

In the 2000s and the 2010s, a new focus on ALGORITHM is given by the debate around machine learning and the possibility to substitute human decisions with computerised processes, such as speech recognition, image recognition, and 'intelligent' robots [51]. It is evident that the possibility to work with the internet and large data-sets is influencing our way of thinking about CT gradually. Informed by a focus on ALGORITHM, ABSTRACTION and IMPLEMENTATION, CT is increasingly related to **AUTOMATION**. This notion of **AUTOMATION** has its roots in the Greek word *automatos* - "*acting of itself*".[2] The name itself was coined by the automobile industry in the 1940s and used to describe their automatic devices in production lines. This *perspective* underlines the need for training, as a fundamental barrier to the use of an electronic computer.

These results can be related to previous research which has underlined the existence of a stronger focus on *computational* aspects on one side, versus *cognitive* aspects on the other [50, 61]. The **REASONING** perspective would be closer to the former category, as it is connected with work procedures from Computer Science, while **SIMPLIFICATION** is closer to the latter category with Psychology-based approaches that give attention to abstraction and problem-solving as composing 'one way of thinking' [7]. However, as this study showed, a general procedure to reduce complexity supports both concepts formalisation in **REASONING** and processes of **AUTOMATION** at the other pole of the spectrum. This in-between-ness of **SIMPLIFICATION** makes it compelling to ask a few questions: do we always need to simplify problems in order to solve them? Can all problems be simplified? As few authors underline in their research, a critical standpoint to CT is helpful to answer these questions and to evaluate *when* and *with what purpose* the electronic computer is needed. This critical thinking, in connection to CT, could help clarify what could be the greatest contributions of Computer Science in other domains.

A second element of discussion emerging from this study is the overall interest expressed by interviewees for CT as a form of **EMPOWERMENT**. The meaning that is given to empowerment has different nuances in each *perspective*, as follows.

From the point of view of **REASONING**, there is the need to develop a procedure that is helpful in formalising constructs and mechanising computational processes. This entails being exposed to training about the steps that are used in Computer Science to solve problems. It can be inferred that the meaning of CT is thinking logically about computer processes, which is a necessary preparation for life [68]. In this view, empowerment is the development of such a *specific mind-set* and the focus is given to *computing as a central field* that influences all other fields.

[2]https://www.lexico.com/definition/automaton

From the point of view of **SIMPLIFICATION**, it is important to understand how to transform reality through representation, and what the consequences of this process are. Subjectivity comes into play in the CT process, and **EMPOWERMENT** is related to *thinking critically about the outcomes of a subjective representation of reality.* In this sense, CT empowers citizens for complex problem solving [49], but also towards the understanding of values and cultures embedded in technology [14].

Finally, from the point of view of **AUTOMATION**, **EMPOWERMENT** means *knowing how to work with electronic computers both efficiently and ethically.* The emphasis here is not so much on specific computational processes, but rather on human-computer interaction and societal reflections [88], participatory design [9, 52], and prototyping as a tool to sustain and scale automated processes [46]. In recent years, it has become increasingly clear that *creating* and *deploying* digital technology is not as innocuous as once thought. Digital artifacts always impact the real world in intended, but also *unintended* ways. People need an understanding of this; and society needs rules for this. People need competences in order to partake in the democratic process of negotiating the rules for our future common digital infrastructure and technology.

Despite the different takes on the conceptualisation of CT, this additional *perspective* that we named **EMPOWERMENT** seems to be a common purpose embedded in all the other three. In many ways, literacy could be used as a synonymous of **EMPOWERMENT**, which can be problematic in terms of finding a clear and generalised definition of CT:

> But, there is always this risk that when you try to market CT to broader audiences, you reach a bit too far and promising a bit too much. That is the risk; but every description, I know, runs either on risk of being too niche, for computing only; or, being unmarketable.
>
> - Matti Tedre, Int.

[31] argue that what distinguishes CT from literacy is the understanding of 'thinking' as merely abstract thinking, from 'thinking-with' objects and artefacts. This focus on the materiality of CT can be a further step in the development of theories that consider the interaction with specific computational 'objects' to develop specific skills for the twenty-first century.

6 CONCLUSION

Through a preliminary literature review, which was used as the input for interviews with eight prominent scholars, this study has revealed that theoretical *perspectives* on CT can be based on very different purposes and procedures, which inform both scientific research and educational policies implementation. Nevertheless, most of the retrieved publications on CT do not refer to an *explicit perspective* or conceptualisation. Such a lack of theoretical grounding might lead to a confusion of goals and objectives of proposals based on the achievement of CT, and to a variety of practices that are mainly informed by practitioners' subjective understandings.

On the other hand, when a theoretical perspective is mentioned in the examined literature on CT, it is often based on an argument to prove that some emphasis should be given *either* to the 'computational' element *or* to the 'thinking' element. The attempt of this study was to focus on these *perspectives* as complementary ways to understand the concept of CT. In fact, a third one has emerged from the document analysis and the interviews, one that has critical thinking at its core, in order to evaluate when and for what purpose it is necessary to automate human processes. Building on these results, future research might find other *perspectives* that are related to the ones presented in this paper.

Furthermore, this study has shown that an attention on skills and competences related to CT is generalised by most of the examined authors, even when defending different *perspectives*. In some cases, **Empowerment** was also expressed as literacy [30]. Further research is needed in this respect, to clarify the theoretical and practical differences of working with CT and/or/as computational literacy [75].

In conclusion, we advocate for an open and interdisciplinary dialogue between researchers championing different *perspectives* on CT. We suggest that this dialogue could start from the (overlapping) *aspects* that compose the concept of CT and define it in different ways. In fact, far from being mutually exclusive, these common *aspects* balance CT goals and practices in a complementary way. Placed on a spectrum that shows their dependency towards the electronic computer, they offer *perspectives* that can lead to different activities, learning styles, teaching methods, and policies.

It is worth further exploring what skills are related to each *perspective*, how they can be developed by youth in the twenty-first century, and what is the specific purpose of each proposed activity. It is our hope that this would provide a starting point for:

> [...] a deep conversation, to the point where we could make some decisions about ways to go.
>
> - Andrea DiSessa, Int.

ACKNOWLEDGMENTS

The authors would like to thank the eight interviewees for their time, availability, and interest in sharing their *perspectives* on CT.

REFERENCES

[1] Friday Joseph Agbo, Solomon Sunday Oyelere, Jarkko Suhonen, and Sunday Adewumi. 2019. A systematic review of computational thinking approach for programming education in higher education institutions. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. 1–10.

[2] Alfred V Aho. 2011. Ubiquity symposium: Computation and computational thinking. *Ubiquity* 2011, January (2011).

[3] Juan-Francisco Alvarez-Herrero et al. 2020. Pensamiento computacional en Educación Infantil, más allá de los robots de suelo. (2020).

[4] Charoula Angeli and Michail Giannakos. 2020. Computational thinking education: Issues and challenges. *Computers in Human Behavior* 105 (2020), 106185.

[5] Charoula Angeli and Nicos Valanides. 2020. Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior* 105 (2020), 105954.

[6] Ashley Ater-Kranov, Robert Bryant, Genevieve Orr, Scott Wallace, and Mo Zhang. 2010. Developing a community definition and teaching modules for computational thinking: accomplishments and challenges. In *Proceedings of the 2010 ACM conference on Information technology education*. 143–148.

[7] Alan F Blackwell, Luke Church, and Thomas RG Green. 2008. The Abstract is an Enemy: Alternative Perspectives to Computational Thinking.. In *PPIG*. 5.

[8] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, Vol. 1. 25.

[9] Loren Britton, Goda Klumbyte, and Claude Draude. 2019. Doing thinking: revisiting computing with artistic research and technofeminism. *Digital Creativity* 30, 4 (2019), 313–328.

[10] Francisco Buitrago Flórez, Rubby Casallas, Marcela Hernández, Alejandro Reyes, Silvia Restrepo, and Giovanna Danies. 2017. Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research* 87, 4 (2017), 834–860.

[11] Alan Bundy. 2007. Computational thinking is pervasive. *Journal of Scientific and Practical Computing* 1, 2 (2007), 67–69.

[12] Elisa Nadire Caeli and Jeppe Bundsgaard. 2019. Datalogisk tænkning og teknologiforståelse i folkeskolen tur-retur. *Tidsskriftet Læring og Medier (LOM)* 11, 19 (2019), 30–30.

[13] Elisa Nadire Caeli and Martin Dybdal. 2020. Teknologiforståelse i skolens praksis. *Tidsskriftet Læring og Medier (LOM)* 12, 22 (2020).

[14] Michael E Caspersen, Judith Gal-Ezer, Andrew McGettrick, and Enrico Nardelli. 2019. Informatics as a fundamental discipline for the 21st century. *Commun. ACM* 62, 4 (2019), 58–58.

[15] Michael E Caspersen, Ole Sejer Iversen, Mogens Nielsen, Hermes Arthur Hjorth, and Line Have Musaeus. 2018. Computational Thinking — hvorfor, hvad og hvordan?: Efter opdrag fra Villum Fondens bestyrelse. (2018).

[16] Morgane Chevalier, Christian Giang, Alberto Piatti, and Francesco Mondada. 2020. Fostering computational thinking through educational robotics: A model for creative computational problem solving. *International Journal of STEM Education* 7, 1 (2020), 1–18.

[17] Douglas E Comer, David Gries, Michael C Mulder, Allen Tucker, A Joe Turner, and Paul R Young. 1989. Computing as a discipline. *Commun. ACM* 32, 1 (1989), 9–23.

[18] Computer-Science-Teachers-Association. [n. d.]. *Computational Thinking operational definition.* https://k12cs.org/computational-thinking/

[19] Miguel Á Conde, Francisco J Rodríguez-Sedano, Camino Fernández-Llamas, José Gonçalves, José Lima, and Francisco J García-Peñalvo. 2021. Fostering STEAM through challenge-based learning, robotics, and physical devices: A systematic mapping literature review. *Computer Applications in Engineering Education* 29, 1 (2021), 46–65.

[20] Valentina Dagienė, Tatjana Jevsikova, Gabrielė Stupurienė, and Anita Juškevičienė. 2021. Teaching computational thinking in primary schools: Worldwide trends and teachers' attitudes. *Computer Science and Information Systems* 00 (2021), 33–33.

[21] Ole-Johan Dahl. 2004. The Birth of Object Orientation: the Simula Languages. In *From Object-Orientation to Formal Methods: Essays in Memory of Ole-Johan Dahl*, Olaf Owe, Stein Krogdahl, and Tom Lyche (Eds.). Springer, Berlin, Germany, 15–25. https://doi.org/10.1007/978-3-540-39993-3_3

[22] Imke de Jong and Johan Jeuring. 2020. Computational Thinking Interventions in Higher Education: A Scoping Literature Review of Interventions Used to Teach Computational Thinking. In *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*. 1–10.

[23] Andreas Dengel and Ute Heuer. 2018. A curriculum of computational thinking as a central idea of information & media literacy. In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education*. 1–6.

[24] Peter J Denning. 2007. Computing is a natural science. *Commun. ACM* 50, 7 (2007), 13–18.

[25] Peter J. Denning. 2009. The Profession of IT Beyond Computational Thinking. *Commun. ACM* 52, 6 (June 2009), 28–30. https://doi.org/10.1145/1516046.1516054

[26] Peter J Denning. 2010. Ubiquity symposium'What is computation?' Opening statement. *Ubiquity* 2010, November (2010).

[27] Peter J Denning. 2018. Where to from here? *ACM Inroads* 9, 4 (2018), 17–21.

[28] Cambridge Dictionary. [n. d.]. *Engineering.* https://dictionary.cambridge.org/dictionary/english/engineering?q=Engineering

[29] Edsger W Dijkstra. 1974. Programming as a discipline of mathematical nature. *The American Mathematical Monthly* 81, 6 (1974), 608–612.

[30] Andrea A DiSessa. 2001. *Changing minds: Computers, learning, and literacy.* Mit Press.

[31] Nina Bonderup Dohn, Stig Børsen Hansen, and Jens Jørgen Hansen. 2019. *Designing for situated knowledge transformation.* Routledge.

[32] Thomas A Easton. 2006. Beyond the algorithmization of the sciences. *Commun. ACM* 49, 5 (2006), 31–33.

[33] W Edmonds and T Kennedy. 2017. Mixed methods. *An applied guide to research designs* (2017), 177–180.

[34] AP Ershov. 1981. Programming, the second literacy. *Microprocessing and Microprogramming* 8, 1 (1981), 1–9.

[35] Louis Fein. 1959. The Role of the University in Computers, Data Processing, and Related Fields. In *Papers Presented at the the March 3-5, 1959, Western Joint Computer Conference* (San Francisco, California) *(IRE-AIEE-ACM '59 (Western))*. Association for Computing Machinery, New York, NY, USA, 119–126. https://doi.org/10.1145/1457838.1457859

[36] Cristian Ferrada, Javier Carrillo-Rosúa, Danilo A Díaz-Levicoy, and Francisco Silva-Díaz. 2020. La robótica desde las áreas STEM en Educación Primaria: una revisión sistemática. (2020).

[37] José Figueiredo and Francisco J García-Peñalvo. 2017. Improving computational thinking using follow and give instructions. In *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality*. 1–7.

[38] Andreas Fischer, Samuel Greiff, and Joachim Funke. 2017. The history of complex problem solving. (2017).

[39] George E Forsythe. 1967. A university's educational program in computer science. *Commun. ACM* 10, 1 (1967), 3–11.

[40] George E Forsythe. 1968. What to do till the computer scientist comes. *The American Mathematical Monthly* 75, 5 (1968), 454–462.

[41] Julian Fraillon, John Ainley, Wolfram Schulz, Daniel Duckworth, and Tim Friedman. 2019. *IEA international computer and information literacy study 2018 assessment framework.* Springer Nature.

[42] Barney G Glaser and Anselm L Strauss. 1967. The discovery of grounded theory: strategies for qualitative research: Aldine Transaction. *New Brunswick, NJ* (1967).

[43] Saul Gorn. 1963. The computer and information sciences: a new basic discipline. *SIAM Rev.* 5, 2 (1963), 150–155.

[44] Sarah Gretter and Aman Yadav. 2016. Computational thinking and media & information literacy: An integrated approach to teaching twenty-first century skills. *TechTrends* 60, 5 (2016), 510–516.

[45] Mark Guzdial. 2008. Education Paving the way for computational thinking. *Commun. ACM* 51, 8 (2008), 25–27.

[46] Joachim Halse, Eva Brandt, Brendon Clark, and Thomas Binder. 2010. *Rehearsing the future.* The Danish Design School Press.

[47] Richard Wesley Hamming. 1969. One man's view of computer science. *Journal of the ACM (JACM)* 16, 1 (1969), 3–12.

[48] Ting-Chia Hsu, Shao-Chen Chang, and Yu-Ting Hung. 2018. How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education* 126 (2018), 296–310.

[49] Ole Sejer Iversen, Rachel Charlotte Smith, and Christian Dindler. 2018. From computational thinking to computational empowerment: a 21st century PD agenda. In *Proceedings of the 15th Participatory Design Conference: Full Papers-Volume 1.* 1–11.

[50] Yasmin Kafai, Chris Proctor, and Debora Lui. 2020. From theory bias to theory dialogue: embracing cognitive, situated, and critical framings of computational thinking in K-12 CS education. *ACM Inroads* 11, 1 (2020), 44–53.

[51] Ken Kahn. 2017. A half-century perspective on Computational Thinking. *Tecnologias, sociedade e conhecimento* 4, 1 (2017), 23–42.

[52] Zachary Kaiser. 2019. Creativity as Computation: Teaching Design in the Age of Automation. *Design and Culture* 11, 2 (2019), 173–192.

[53] Donald L Katz. 1960. Conference report on the use of computers in engineering classroom instruction. *Commun. ACM* 3, 10 (1960), 522–527.

[54] Donald E Knuth. 1974. Computer science and its relation to mathematics. *The American Mathematical Monthly* 81, 4 (1974), 323–343.

[55] Donald E Knuth. 1985. Algorithmic thinking and mathematical thinking. *The American Mathematical Monthly* 92, 3 (1985), 170–181.

[56] Siu-Cheung Kong and Harold Abelson. 2019. *Computational thinking education.* Springer Nature.

[57] Bill Kules. 2016. Computational thinking is critical thinking: Connecting to university discourse, goals, and learning outcomes. *Proceedings of the association for information science and technology* 53, 1 (2016), 1–6.

[58] A Labusch, B Eickelmann, and M Vennemann. 2019. Computational Thinking Processes and Their Congruence with Problem-Solving and Information Processing. In *Computational Thinking Education.* Springer, Singapore, 65–78.

[59] P Larsson, Mikko-Ville Apiola, and Mikko-Jussi Laakso. 2019. The uniqueness of computational thinking. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).* IEEE, 687–692.

[60] Irene Lee, Shuchi Grover, Fred Martin, Sarita Pillai, and Joyce Malyn-Smith. 2020. Computational thinking from a disciplinary perspective: Integrating computational thinking in K-12 science, technology, engineering, and mathematics education. *Journal of Science Education and Technology* 29, 1 (2020), 1–8.

[61] Yeping Li, Alan H Schoenfeld, Andrea A diSessa, Arthur C Graesser, Lisa C Benson, Lyn D English, and Richard A Duschl. 2020. Computational thinking is more about thinking than computing. , 18 pages.

[62] Richard E Mayer, Jennifer L Dyck, and William Vilberg. 1986. Learning to program and learning to think: what's the connection? *Commun. ACM* 29, 7 (1986), 605–610.

[63] Patrick Mendelsohn, TRG Green, and Paul Brna. 1990. Programming languages in education: The search for an easy start. In *Psychology of programming.* Elsevier, 175–200.

[64] Microsoft. 2011. *The Software Crisis: A Brief Look at How Rework Shaped the Evolution of Software Methodolgies.* https://docs.microsoft.com/en-gb/archive/blogs/karchworld_identity/the-software-crisis-a-brief-look-at-how-rework-shaped-the-evolution-of-software-methodolgies

[65] Thilde Emilie Møller, Vibeke Schrøder, and Mads Middelboe Rehder. 2019. Lærerfaglig teknologiforståelse. *Studier i læreruddannelse og-profession* 4, 1 (2019), 125–143.

[66] Chrystalla Mouza, Yi-Cheng Pan, Hui Yang, and Lori Pollock. 2020. A multiyear investigation of student computational thinking concepts, practices, and perspectives in an after-school computing program. *Journal of Educational Computing Research* 58, 5 (2020), 1029–1056.

[67] Enrico Nardelli. 2019. Do we really need computational thinking? *Commun. ACM* 62, 2 (2019), 32–35.

[68] Peter Naur. 1966. *Plan for et kursus i datalogi og datamatik.* A/S Regnecentralen.

[69] Allen Newell, Alan J Perlis, and Herbert A Simon. 1967. Computer science. *Science* 157, 3795 (1967), 1373–1374.

[70] Lexico Oxford dictionary. [n. d.]. *Automation.* https://www.lexico.com/definition/automaton

[71] Tauno Palts and Margus Pedaste. 2020. A model for developing computational thinking skills. *Informatics in Education* 19, 1 (2020), 113–128.

[72] Seymour A Papert. 1980. *Mindstorms: Children, computers, and powerful ideas.* Basic books.

[73] David Lorge Parnas. 1999. Software engineering programs are not computer science programs. *IEEE software* 16, 6 (1999), 19–30.

[74] Arnold Pears. 2019. Developing Computational Thinking,"Fad" or "Fundamental"? *Constructivist Foundations* 14, 3 (2019), 410–412.

[75] Magda Pischetola. 2021. Teacher professional development in Higher Education and the Teknosofikum project. *LearningTech* 10, 1 (2021), 46–75.

[76] Jake A Qualls and Linda B Sherrell. 2010. Why computational thinking should be integrated into the curriculum. *Journal of Computing Sciences in Colleges* 25, 5 (2010), 66–71.

[77] Sharon M Ravitch and Nicole Mittenfelner Carl. 2019. *Qualitative research: Bridging the conceptual, theoretical, and methodological.* Sage Publications.

[78] Emily Relkin, Laura de Ruiter, and Marina Umaschi Bers. 2020. TechCheck: Development and validation of an unplugged assessment of computational thinking in early childhood education. *Journal of Science Education and Technology* 29 (2020), 482–498.

[79] Peter J Rich, Stacie L Mason, and Jared O'Leary. 2021. Measuring the effect of continuous professional development on elementary teachers' self-efficacy to teach coding and computational thinking. *Computers & Education* 168 (2021), 104196.

[80] Barbara Sabitzer, Heike Demarle-Meusel, and Maria Jarnig. 2018. Computational thinking through modeling in language lessons. In *2018 IEEE Global Engineering Education Conference (EDUCON).* IEEE, 1913–1919.

[81] Cynthia Selby and John Woollard. 2013. Computational thinking: the developing definition. (2013).

[82] Valerie J Shute, Chen Sun, and Jodi Asbell-Clarke. 2017. Demystifying computational thinking. *Educational Research Review* 22 (2017), 142–158.

[83] Hyo-Jeong So, Morris Siu-Yung Jong, and Chen-Chung Liu. 2020. Computational thinking education in the Asian Pacific region.

[84] Thomas Hvid Spangsberg and Martin Brynskov. 2017. Towards a dialectic relationship between the implicit and explicit nature of computational thinking: a computer semiotics perspective. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research.* 197–198.

[85] Xiaodan Tang, Yue Yin, Qiao Lin, Roxana Hadad, and Xiaoming Zhai. 2020. Assessing computational thinking: A systematic review of empirical studies. *Computers & Education* 148 (2020), 103798.

[86] L Tanggaard and Svend Brinkmann. 2015. Interviewet: Samtalen som forskningsmetode, I Kvalitative Metoder. *Hans Reitzels Forlag, København* (2015), 29–53.

[87] Matti Tedre and Peter J. Denning. 2016. The Long Quest for Computational Thinking. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '16).* Association for Computing Machinery, New York, NY, USA, 120–129. https://doi.org/10.1145/2999541.2999542

[88] Ari Tuhkala, Marie-Louise Wagner, Ole Sejer Iversen, and Tommi Kärkkäinen. 2019. Technology Comprehension—Combining computing, design, and societal reflection as a national subject. *International Journal of Child-Computer Interaction* 20 (2019), 54–63.

[89] Annette Vee. 2013. Understanding computer programming as a literacy. *Literacy in Composition Studies* 1, 2 (2013), 42–64.

[90] Xuefeng Wei, Lin Lin, Nanxi Meng, Wei Tan, Siu-Cheung Kong, et al. 2021. The effectiveness of partial pair programming on elementary school students' computational thinking skills and self-efficacy. *Computers & Education* 160 (2021), 104023.

[91] Computer Science Wiki. [n. d.]. *Abstraction.* https://computersciencewiki.org/index.php/Abstraction

[92] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[93] Jeannette M Wing. 2008. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366, 1881 (2008), 3717–3725.

[94] Gary Ka-Wai Wong and Ho-Yin Cheung. 2020. Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming. *Interactive Learning Environments* 28, 4 (2020), 438–450.

[95] Osman Yaşar. 2018. A new perspective on computational thinking. *Commun. ACM* 61, 7 (2018), 33–39.

[96] Osman Yasar and Rubin H Landau. 2003. Elements of computational science and engineering education. *SIAM review* 45, 4 (2003), 787–805.