Computational Thinking, the Notional Machine, Pre-service Teachers, and Research Opportunities

Matt Bower
School of Education
Macquarie University
Australia
matt.bower@mq.edu.au

Katrina Falkner
School of Computer Science
University of Adelaide
Australia
katrina.falkner@adelaide.edu.au

Abstract

There is general consensus regarding the urgent and pressing need to develop school students' computational thinking abilities, and to help school teachers develop computational thinking pedagogies. One possible reason that teachers (and students) may struggle with computational thinking processes is because they have poorly developed mental models of how computers work, i.e., they have inadequate "notional machines". Based on a pilot survey of 44 pre-service teachers this paper explores (mis)conceptions of computational thinking, and proposes a research agenda for investigating the use of notional machine activities as a way of developing pre-service teacher computational thinking pedagogical capabilities.

Keywords: Computational thinking, notional machine, teacher education, K-12

1 Introduction

Recent changes in ICT curriculum have moved from a focus on the use of ICT, i.e. digital literacy, to the need for awareness of how to create and influence the creation of new technologies. Recognition has grown, that in addition to the need to increase awareness and interest in Computer Science (CS), the fundamental concepts and skills of CS are valuable for children to learn. This has provided a driver for CS curriculum to be introduced as early as the first year of schooling. Preparing students to engage in current technologies and participate as creators of future technologies requires more than is currently being provided. We need to ensure that our educational systems provide not only the fundamentals of digital literacy – familiarity with the tools and approaches to interact with technology – but also the computational thinking processes needed to understand the scientific practices that underpin technology.

In alignment with recent global trends, the Australian primary and secondary school system is undergoing a significant period of change, with the introduction of a National Curriculum from K-10, new learning areas, and the development of national assessment programs. This new curriculum, defined by the Australian Curriculum Assessment and Reporting Authority (ACARA), identifies that "rapid and continuing advances in ICT are changing the ways people share, use, develop and process information and technology, and young people need to be highly skilled in ICT. While schools already employ these technologies in learning, there is a need to increase their effectiveness significantly over the next decade" (ACARA, 2012). The ACARA documents include ICT awareness (i.e. digital literacy) as a key capability, embedded throughout the curriculum, and introduce a new learning area, Technologies, combining the "distinct but related" areas of Design and Technologies and Digital Technologies (DT) (ACARA, 2013a). DT explicitly addresses the development of computational thinking skills as core to the understanding of digital technologies.

The success with which the digital technologies curriculum is implemented will depend, to a large extent, on the quality of learning and teaching. Consultation with Industry, Community and Education within Australia (ACARA, 2013b) has identified significant concerns in relation to teacher development (particularly at K-7), appropriate pedagogy, and skills needed for integration of DT learning objectives with the teaching of other learning areas. Approximately 55% of respondents indicated concern with the manageability of the implementation of the proposed curriculum, while 45% of respondents did not think that the learning objectives were realistic. Support for the professional development of teachers, including the creation of community networks to share insights and pedagogical approaches and research, has been identified as crucial in expanding CS curricula (Gander, et al., 2013). Bell, Newton, Andreae, & Robins (2012) describe the New Zealand experience of the rapid introduction of a senior secondary CS curriculum, and the need for extensive teacher development that addresses both content knowledge and pedagogical knowledge. However, many of the teachers who will be responsible for teaching the DT curriculum have not completed any studies that encompassed computational thinking concepts or processes, let alone how to teach these.

A classic concept in the computing education literature relevant to computational thinking is that of the "Notional Machine" (du Boulay, O'Shea, & Monk, 1989). The Notional Machine is a mental model that enables its user to make predictions about how a machine will perform. Without an adequate notional machine it is not possible to perform computational thinking processes (as elaborated later in this paper). Based on a pilot study of 44 pre-service teachers, this paper analyses conceptions and misconceptions of computational thinking, and based on the survey results and literature review proposes a research agenda for developing computational thinking capabilities based on notional machine activities.

2 What is Computational Thinking?

Computational thinking, as defined by Wing (2006) is: "*solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science*". Computational thinking involves understanding the fundamental concepts and abstractions that underpin computer science, and then reformulating problems into a form that can be solved readily using what we already understand. ACARA defines computational thinking as "*a problem-solving method that involves various techniques and strategies, such as organising data logically, breaking down problems into components, and the design and use of algorithms, patterns and models*" (ACARA, 2012). Understanding computational thinking involves understanding core computer science concepts, and the ability to conceptualise and create abstractions that define solutions to problems. But why is it important that we understand computational thinking? Why do we need to develop these mental models as part of our education system?

In the US, a recent survey of CS education at High Schools identified that Schools are "failing to provide students with access to the key academic discipline of CS, despite the fact that it is intimately linked with current concerns regarding national competitiveness…" (Gal-Ezer & Stephenson, 2009). Furthermore, recent reports from the US and Europe have argued that it is essential that children be exposed to CS concepts and principles from the very start of their education so that "every child [may] have the opportunity to learn Computing at School" (Gander, et al., 2013; Wilson & Guzdial, 2010). If not, we face the risk of our youth being placed in the position of consumers of technology produced elsewhere, unable to actively participate as producers and leaders in this field (Gal-Ezer & Stephenson, 2009; Gander, et al., 2013; Wilson & Guzdial, 2010). As Alan Noble, Engineering Director for Australia and New Zealand notes, "there is a difference between using a smartphone and creating an app that reaches millions of people" (Noble, 2012).

New curricula introduced in England (British Department for Education, 2013), Australia (ACARA, 2012), New Zealand and the new ACM CS standards (Seehorn, et al., 2011) have identified the need to educate for both digital literacy and CS, and the need to promote both learning areas from the commencement of schooling to support youth in participating in an increasingly digital society. Students who are exposed early generally have deeper interactions with computers, focused on exploring computers and related concepts rather than just utilising the computer for set tasks (Schulte & Knobelsdorf, 2007). Early exposure increases interest in computing by increasing computing self-efficacy (Akbulut & Looney, 2009).

However, it is also stressed that students would benefit from education in CS as an independent scientific subject on par with learning areas such as Mathematics or English (Gander, et al., 2013). It is essential that our education systems evolve, requiring the clear articulation of CS as a distinct discipline, including the integration of CS as a fundamental learning area across the curriculum and the exploration of the societal and cultural impacts of technology. Computational Thinking should be seen as an enabling subject (such as literacy or numeracy) whereas computing should be seen a separate discipline equivalent to Mathematics or Physics (BCS, 2010).

Developing capacity for computational thinking goes beyond building individual understanding and capabilities, however, but helps address a significant concern over the shrinking pool of qualified ICT professionals available to meet the demands of a rapidly growing industry. In a recent report by PWC (2013) on strategies and challenges in accelerating Australian innovation, they identify that "*Even if all international students were to stay in Australia post graduation, the supply of computer science and engineering graduates would still fall short of the numbers needed to accelerate growth*", while the Bureau of Labor Statistics (Lockard and Wolf, 2010), identifies that within computer and mathematical occupations, there is a 22.0% increase in employment projected from 2010-2020 (14.3% for all occupations).

3 Notional Machine

For many decades before developing computational thinking capabilities emerged as an important social agenda, Computer Science education researchers have been searching for the reasons why students find computing difficult. A foundational theory in computer science education that explains why students struggle to master computing concepts and processes is that of the "notional machine". The notional machine is an abstract version of the computer, "an idealised, conceptual computer whose properties are implied by the constructs in the programming language employed" (du Boulay, et al., 1989, p. 431).

The notional machine has been used in numerous studies (refer to Robins, Rountree, & Rountree, 2003, p. 149) and provides a theoretical orientation for examining how people think about computing and the misconceptions that may arise. That the notional machine assists learning is not a hypothetical proposition. For instance Mayer (1989) showed that students supplied with a notional machine model were better at solving some kinds of problems than students without the model.

In order for students to progress towards expert behaviour as efficiently as possible it is important to have an understanding of the difficulties they experience. This allows educators to provide scaffolding that helps learners to surmount these difficulties and allows the students themselves to pre-empt impediments to their learning by being aware of their potential before they arise. Du Boulay (1989) describes five inextricably linked potential sources of difficulty when learning computer programming:

1. general orientation (what programs are for and what can be done with them)
2. the notional machine (a model of the computer as it relates to executing programs)
3. notation (the syntax and semantics of a particular programming language)
4. structures (schemas and plans)

5. pragmatics (the skills of planning, developing, testing, debugging and so on).

Du Boulay et al. (1989) note that much of the early difficulty in learning computing arises from the student's attempt to deal with these different kinds of difficulties all at once. 'Misapplication of analogy', 'interaction of parts' and 'overgeneralisation' errors result. In the early stages teachers can assist the learning process by trying to address these domains separately (as far as possible) so as to reduce interference between them.

Du Boulay et al. (1989) suggest that in order for novice programmers to overcome comprehension problems caused by the hidden, unmarked actions and side effects of visually unmarked processes the notional machine needs to be simple and supported with some kind of concrete tool which allows the model to be observed. They suggest that the visibility component of such models be supported through 'commentary' – a teacher delivered or automated expose of the state of the machine. On the other hand the simplicity component of the machine can be supported through:

1. *functional simplicity* (operations require minimal instructions to specify)
2. *logical simplicity* (problems posed to students are of contained scale)
3. *syntactic simplicity* (the rules for writing instructions are accessible and uniform).

Du Boulay et al. (1989) conclude that matching visibility and simplicity components of notional machines to different populations of novice learners leads to improved educational outcomes. One would also suspect that without notional machine cognitive models, students' computational thinking progress would be severely restricted in the long term, and that the more sophisticated a student's notional machine the more developed their problem solving abilities. Both of these conjectures represent potential areas for further research.

As mentioned, the Notional Machine is a discipline specific mental model, and the literature on mental models also sheds light on how learning and teaching computational thinking may be enhanced. Norman (1983) distinguishes between the target system (the system that the person is learning or using), the conceptual model of the target system (an accurate and appropriate representation of the target system), the user's mental model of the target system (which may or may not be accurate and suffice), the researcher's conceptualization of the learner's model (a model of a model). Often teachers attempt to provide students with a conceptual model of a system to support the formation of students' mental models. Effective representations are those that capture the essential elements of the system leaving out the rest, with the critical point being which aspects to include and which to omit (Norman, 1993). Successfully selecting and describing the poignant features of a system allows students to concentrate upon the critical aspects of the system without being distracted by irrelevancies. When acquired, such conceptual models enhance students' capacity to reason and think. However if critical features are omitted or represented in a way that students misunderstand, then students may not comprehend crucial aspects of the system and may subsequently form misguided conclusions (Norman, 1993).

Some sub-domains of computer science have lead to specialised mental models of how students learn computing being developed. For instance, without a viable mental model of recursion that correctly represents active flow (when control is passed forward to new instantiations) and passive flow (when control flows back from the terminated instantiations) students cannot reliably construct recursive algorithm traces (Gotschi, Sanders, & Galpin, 2003).

There are several advantages to such domain specific models. Firstly, they can inform educators' decisions about the required approach to learning – in the case of recursion a constructivist approach is required in order for students to create a viable mental model adequate to apply design concepts and solve problems. Secondly, domain specific models assist lecturers by providing accurate mental models, such as Kayney's 'copies' model of recursion, that have been demonstrated as successful at promoting understanding. Thirdly, such research explicitly exposes non-viable mental models that students may form (such as the looping, magic, and step models), allowing lecturers and pupils to pre-empt student errors (Gotschi, et al., 2003).

4 Developing Computational Thinking

There are a variety of broad recommendations about how to develop computational thinking generally, most of which emanate from the Computer Science education literature. Pedagogues recommend connecting Computational Thinking to young people's interests (Resnick, et al., 2009), for instance, through computer games (Carter, 2006; Lenox, Jesse, & Woratschek, 2012) or multimedia based learning tasks (Blank, et al., 2003). A games based approach to introducing programming in the middle years has been shown to help develop computational thinking concepts (events, alternation, iteration, parallelism, additional methods, parameters, local and global variables) at the same time as enhancing students enjoyment of learning computing (Repenning, Webb, & Ioannidou, 2010).

Providing students with a low floor (easy to learn), high ceiling (hard to master, many opportunities to learn), wide walls (flexible and adaptable to a wide range of applications) enables students of different ability levels to remain engaged (Resnick, et al., 2009). Stephenson et al. (2005) recommend designing course materials that incorporate meaningful learning through the use of problem-solving approaches, appealing experimental environments, and an explicit emphasis on design and a real-world focus. Supporting skills beyond programming has been shown to increase student satisfaction with computing and may broaden further participation (Repenning & Ioannidou, 2008).

Creating a conducive learning environment has also been proposed as a way to enhance computational thinking. For instance, Stephenson et al. (2005) recommend establishing a welcoming environment that models life-long learning. Barr & Stephenson (2011) suggest increased use of computational vocabulary by teachers and students where appropriate, acceptance of failed solution attempts by teachers and students, and tasks involving team work by students. Yet, there is little research to substantiate these claims.

5 The challenge of teaching Computational Thinking

One of the main problems faced by the domain is that many students perceive computing to be essentially the same as technology training, which can be seen as repetitive and teaching skills that the students already know such as how to use standard Office tools (BCS, 2010). It is also possible that teachers (including pre-service teachers) may not always have a firm understanding of what computational thinking involves (as will be explored later in this paper).

Studies have identified increased anxiety and concern over preparation time when dealing with unfamiliar content (Curzon, McOwan, Cutts, & Bell, 2009). Even in cases where teachers are experienced with computing fundamentals, the integration of new tools can create anxiety that causes them to deviate from their planned lessons (Meerbaum-Salant, Armoni, & Ben-Ari, 2013).

Training teachers to teach computational thinking is an essential piece of the puzzle (BCS, 2010; Black, et al., 2013). Poor lessons demotivate learners, creating negative attitudes towards the subjects, and this can create a vicious cycle of demotivating teachers who in turn create poorer lessons (BCS, 2010). Professional development is critical in order for teachers to effectively develop computational thinking pedagogies, (Barr & Stephenson, 2011). This is not only about offering training courses, but also establishing effective communities of practice to provide ongoing support and sharing of resources (Black, et al., 2013).

It is also critical to provide resources to help teachers effectively teach computational thinking concepts and processes (Barr & Stephenson, 2011). Settle et al (2012) identify specific difficulties for educators in translating materials into existing curriculum, with an emphasis on the increased difficulty in adopting and integrating new tools. It is challenging is to provide teachers with material which effectively conveys the most important aspects of computing without reducing it to tool use or programming, both of which are misconceptions of computing (Battig, 2008). Tinapple, Sadauskas, & Olson (2013) further comment on the challenge of implementing lessons where expected software and/or hardware are not easily available.

Another issue is that teachers often utilise fun activities with a focus on impressive technology, physical computing and programming using constructionist environments rather than providing opportunities for deep learning of computational thinking (Black, et al., 2013). These results are indicative on a focus on tool usage for engagement, rather than a deep understanding of computational thinking processes and concepts.

6 A pilot survey of pre-service teachers

In order to gauge pre-service teachers' perceptions of computational thinking learning and teaching in light of the upcoming Australian Digital Technologies Curriculum a pilot survey was run in April of 2014. The anonymous online survey was issued to 84 pre-service teachers who were completing the 300 level subject "EDUC362 – Digital Creativity and Learning" at Macquarie University. The survey was conducted during Week 3 of Semester 1 (March 2014). A total of 44 pre-service teachers volunteered to respond. Apart from demographic questions relating to age, gender and the program of study in which the student was enrolled, the survey asked about pre-service teachers' awareness of the upcoming Australian Digital Technologies curriculum, their conceptions of the term 'computational thinking', their understanding of pedagogies and technologies that can be used to develop computational thinking, and their confidence to teach computational thinking.

Open-ended responses were analysed using qualitative coding techniques. First classified using an open-coding phase to determine preliminary analytic categories. Next, axial coding was carried out to determine emergent themes and refine categorisations. Lastly, a selective-coding phase supported representation of the conceptual coding categories for reporting purposes. (See Neuman, 2006, for further details of the approach.) If responses addressed multiple issues they were coded in more than one category, meaning that it was possible to have a greater tally of responses across the items than the number of respondents.

Quantitative data was interpreted and reported using standard descriptive statistics techniques.

6.1 Results

Of the 44 students who chose to respond, 38 were intending to be primary school teachers and 5 were planning to be secondary school teachers (2 science, 2 languages, and 1 english/history). On respondent did not indicate their intended teaching level. The large majority of respondents were in their third or fourth year of their program (42 out of 44). The age distribution was right skewed with 29 participants indicating that they were in the 18-24 age range. A total of 33 females and 11 males participated.

6.1.1 Awareness of Computational Thinking

Pre-service teachers' awareness of the upcoming Australian Digital Technologies Curriculum (ADTC) and whether they had heard of the term 'computational thinking' is shown in Table 1.

	Heard of 'Computational Thinking'	Not heard of 'Computational Thinking'
Aware of ADTC	15	11
Unaware of ADTC	11	7

Table 1: Awareness of the upcoming Australian Digital Technologies Curriculum (ADTC) and the term 'computational thinking'

The table demonstrates that an awareness of the upcoming ADTC did not necessarily imply an awareness of 'computational thinking', even though computational thinking was highlighted by the Australian Curriculum Assessment and Reporting Authority (ACARA) as a distinguishing core feature of the ADTC. Similarly, awareness of computational thinking did not necessarily derive from the ADTC, with a quarter of students indicating that they had heard of computational thinking but did not know about the impending ADTC.

6.1.2 Conceptions of 'computational thinking'

There were 32 pre-service teachers who chose to respond to the question "what does computational thinking mean to you?". Table 2 summarises their responses into the categories that emerged from the coding process. Note once again that some responses are tallied under two or more categories if the response incorporated multiple elements. 'Problem solving using technology' has been included as a separate category to 'problem solving' or 'using technology' as it demonstrates a deeper understanding of computational thinking than either of the latter two categories.

Computational thinking construct	fn
problem solving using technology	11
using technology	10
technological thinking	5
logical thinking	5
gathering/organising/processing information	3
analytical thinking	3
critical thinking	2
creative thinking	2
mathematical thinking	2
problem solving	2
thinking like computer	2
scientific thinking	1
structured thinking	1
strategic thinking	1
testing	1
efficiency	1
non-descript	1

Table 2: Summary of pre-service teacher conceptions of 'computational thinking'

Over one third of respondents described computational thinking as involving "problem solving using technology", though descriptions varied widely in sophistication. For example:

Problem solving using technology; using technology in a variety of ways to approach a problem; analysing and logically organising data, generating problems that require computers assistance; identifying, testing, and implementing possible solutions

Using computer technology to solve a problem.

Having heard of the term computational thinking did not necessarily result in more sophisticated responses being provided. For instance, the first response above is from someone who had not heard of computational thinking and the second response is from someone who had.

Nearly one third of the pre-service teachers considered computational thinking to merely be using technology, for instance "awareness of how to operate software, ability to 'self help'". Two students described it as problem solving without associating it with technology, and one student had a blurred conception of computational thinking as both digital literacies and problem solving using technology: "Digital Literacy, the ability to use technology to solve problems and assist learning, create digital artefacts".

Pre-service teachers were able to identify types of thinking associated with computational thinking, namely logical thinking, analytical thinking, critical thinking, creative thinking, mathematical thinking, scientific thinking, structured thinking, and strategic thinking. Some were able to identify activities and concepts associated with computational thinking, such as testing, efficiency, gathering information and organising data. Only two students were able to associate computational thinking with more than three of any of the above elements.

Two pre-service teachers erroneously thought computational thinking was thinking like a computer, for instance "Thinking or memorising in a way that computer works". One pre-service teacher gave the non-descript response "a process or a way of thinking to understand topics". There were five students who used the term "technological thinking" or synonymic phrases, which has no clear meaning,

6.1.3 Associated Pedagogies

There were 30 pre-service teachers who chose to respond to the question "What pedagogical strategies do you have (or can you think of) for developing school students' computational thinking capabilities?" Their responses are summarised in Table 3.

pedagogical strategies	fn
using technology	13
group work	6
problem based tasks	6
active learning	4
direct instruction / modelling	3
inquiry based approach	3
games/play	2
none / non-descript	2
provide scaffolding	2
teacher familiarity with technology	2
authentic problems	1
brainstorming	1
establish purpose	1
provide process for thinking	1
safe environment	1
writing code	1

Table 3: Summary of pre-service teacher pedagogical strategies to develop computational thinking

The most popular pedagogical strategy represented in students' responses (n=13) was to simply use technology, for instance: "Continuous practice, engagement and exposure to different computer technology". Four of these responses also mentioned problem solving in association with the use of technology. Six students made general mention of how group work strategies could be used (for instance, collaborative learning, cooperative learning, paired learning). There were sixteen instances where responses discussed the nature of the learning process (problem-based learning, active learning, inquiry learning, games based learning, brainstorming, writing code). It is interesting to note that only one pre-service teacher mentioned writing code. There were another ten cases where responses discussed the role and responsibilities of the teacher (direct instruction / modelling, provide scaffolding, be familiar with technology, establish purpose, provide processes for thinking, creating a safe environment).

Overall responses were lacking in detail so in most cases it was difficult to tell whether pre-service teachers had a concrete understanding of how the pedagogy could be applied to develop computational thinking. Responses also revealed more about pre-service teacher conceptions of computational thinking. For instance, one respondent's pedagogical strategies were:

> *Using group work (heterogeneous groups) for students to engage in negotiation, reasoning and student discussion. I would also use apps for students to engage in thinking abstractly and outside of the square such as Comic life, I-movie.*

It is unclear how this respondent would use group work to develop computational thinking, and it appears that while the student did associate abstraction with computational thinking, they did not appear have a clear understanding of how technology could be used to develop computational thinking.

6.1.4 Supportive Technologies

Asking pre-service teachers the question "*How can technologies be used to help develop school students' computational thinking capabilities? (Provide specific examples if you can.)*" offered further insight into their conceptions of computational thinking (see Table 4). Of the 26 students who responded to this question, 10 provided only unspecific suggestions about how technology could be used to develop computational thinking, for instance "organise and help the logical thinking". Six students talked generally about how technology could be used to increase engagement, for of which were from the unspecific respondents. For example: "technological resources can be more engaging/exciting to students".

Technologies to support computational thinking	fn
unspecific	10
engagement	6
conduct research (e.g. searching Internet)	5
presentation tools	4
software/apps - general	4
comic/story creation tools	3
mindmapping	3
create 3D objects	2
data analysis (e.g. spreadsheet)	2
practice - general	2
brainstorming software	1
none	1
program creation	1
publishing tools	1
websites	1

Table 4: Summary of pre-service teacher identified technologies for developing computational thinking

Some pre-service teachers provided more specific suggestions about how technology could be used to develop computational thinking, but for many of these it was unclear how it actually would develop computational thinking. For instance, using comic/story creation tools, mindmapping tools, brainstorming and presentation tools are not obviously and usually related to developing compuational thinking. The specific examples of technologies that pre-service teachers identified were Mindmeister, Comic Life, Toontastic. Prezi, iBooks, and Google Sketchup. Five students mentioned using the Internet for research purposes, and only one identified a technology that was specifically related to computational thinking (the code.org website).

6.1.5 Pre-service teacher confidence

There were 32 pre-service teachers who chose to respond to the questions relating to how confident they felt to develop their students' computational thinking capabilities (see Figure 1). From the graph it can be seen that 18 of the 32 pre-service teachers (56%) indicated that they were to some degree unconfident rather than confident about teaching computational thinking.

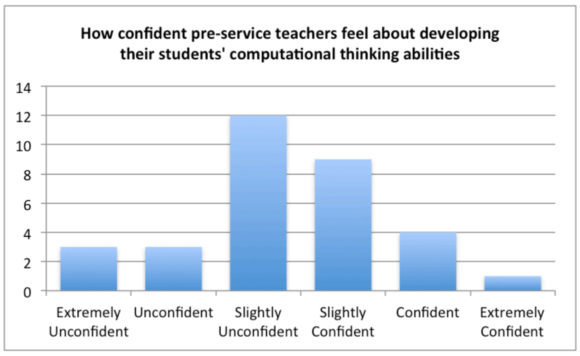

Figure 1: Pre-service teachers' confidence about developing their students' computational thinking abilities

It is important to note that responses on the confident side of the scale did not mean that pre-service teacher confidence was warranted. For instance, some pre-service teachers indicated that they were 'slightly confident' about developing their students'' computational thinking abilities, but had not heard of the term computational thinking and had poor conceptions of computational thinking such as:

> *Computational thinking is ones ability to navigate and problem solve using the medium of technology such as ipad, macbooks and IWB's.*

> *teaching and learning using technology*

More concerning, there were some teachers who had heard of computational thinking and indicated that they were 'confident' about developing their students' computational thinking abilities yet had erroneous conceptions of computational thinking, for instance:

using the computer to help with forming ideas and opinions / - how technology can help your thinking

6.1.6 Lack of Confidence

When pre-service teachers were asked "what prevents you from feeling confident about developing your students' computational thinking capabilities?" responses related to pedagogical issues, technology issues, general issues, circumstantial and affective issues.

Nine pre-service teachers felt unconfident about developing their students' computational thinking because of pedagogical issues, including unfamiliarity with the curriculum (5), lack of pedagogical strategies (3), lack of lesson ideas (1), and uncertainty how to apply computational thinking to real world situations (1). There were eight pre-service teachers who felt that they did not have the technological knowledge and experience to feel confident about teaching computational thinking, though many of these appeared to be confusing computational thinking with general technology usage (for instance "I lack ICT knowledge"). One of these pre-service teachers felt they did not have the required computer science and programming knowledge.

There were thirteen pre-service teachers who indicated more general reasons for their lack of confidence including a poor understanding of what computational thinking means (4), a general lack of knowledge (6) and a general lack of experience (3). Two pre-service teachers did not feel confident about teaching computational thinking due to circumstantial factors relating to becoming a teacher:

Still learning about being a teacher so not yet confident in any particular area

I wasn't not taught like this at school, content and the use of technology

One pre-service teacher spoke directly about the fear of the unfamiliar affecting their confidence:

Because it is something new to me and to teach something i am just coming to terms with slightly scares me and i lose confidence because of that

6.1.7 Building confidence

When pre-service teachers were asked "*What could help you to feel more confident about developing your students' computational thinking capabilities?*" the most common response related to explicit professional development (11 respondents). Other items identified by students provide insight into the form that such professional development might take. There were 6 pre-service teachers who indicated they would like a better understanding of pedagogical strategies, 7 who wanted greater exposure to and experience with technology, and 7 who felt that a better understanding of computational thinking would improve their confidence to teach computational thinking. There were seven students who indicated that greater understanding and practice generally would be beneficial.

Pre-service teachers identified other factors that could improve their confidence in developing computational thinking including more resources and information, learning more about computer programming, learning more about the research relating to computational thinking, and having well planned lessons.

6.2 Limitations of this study

A limitation of this study is that it was only issued to a small sample of pre-service teachers from one university, and results may vary widely depending on the institution. As well, students were not asked about their previous studies of computing, which one would expect would have a large influence on their responses. Any future iterations of the survey will ask students about their previous exposure to computing.

The survey was conducted before pre-service teachers completed a topic on computational thinking in the third year unit they were studying. This was done so that responses were more representative of the general pre-service teacher population of the university, most of whom do not complete the unit which was offered for the first time in 2014. After completing the unit student responses may have been quite different. However, it is conjectured that many universities do not yet have any courses that cover computational thinking as an explicit topic, and as such the responses may be more representative of the broader pre-service teacher population both nationally and internationally.

As this was an online survey students may not have been motivated to provide elaborate responses that accurately represented the full extent of their perceptions and conceptions. Semi-structured interview techniques may be necessary to probe more deeply into pre-service teacher thoughts surrounding computational thinking.

7 Discussion of results

Generally speaking pre-service teachers had a weak understanding of computational thinking. There are a large proportion of pre-service teachers who confuse computational thinking with using technology generally (for instance word processing or searching the internet). Pre-service teachers correctly associated computational thinking with problem solving using technology, logical thinking, gathering/organising/processing information, analytical thinking, critical thinking, creative thinking, mathematical thinking, scientific thinking, structured thinking, strategic thinking, testing and efficiency, though only two students were able to associate it with more than three of these points. This indicates that there is extensive potential to improve pre-service teachers' conceptions of computational thinking. The data also implied we should not assume that because pre-service teachers are aware of the upcoming Digital Technologies Curriculum they understand computational thinking, or visa versa – half of respondents were aware of one but not the other.

For many of the pre-service teachers the extent of their pedagogical strategies for developing computational thinking was simply to have students use technology. Collectively pre-service teachers were able to identify generally appropriate pedagogical strategies such as types of group work and student centred learning. Several teachers identified the role of the teacher in providing instruction and creating a conducive learning environment. Yet because responses were almost invariably lacking in detail there was no evidence to indicate that the pre-service teachers had specific and clear ideas about how to develop their students' computational thinking capabilities.

The technologies they identified to support the learning of computational thinking provided further verification that many students did not understand what was meant by computational thinking - the specific tools that were suggested (such as Comic Life and iBooks) bore no specific relation to computational thinking and only one student mentioned a purpose built platform (the code.org website).

Pre-service teachers were of varying confidence about teaching computational thinking, and some were overconfident based on their evidenced understanding. Not only were the majority on the unconfident side of the response spectrum, but several of those who indicated confidence had poorly formed or incorrect conceptions of what computational thinking actually meant. There were classic examples of third order ignorance (Waite et al 2003) where pre-service teachers were unaware that they did not know.

Responses from pre-service teachers indicated that they would value professional learning opportunities that focused on:

- Developing their computational thinking pedagogical capabilities - understanding of the curriculum, lesson ideas, strategies for implementation, links to real world examples
- Technological understanding - exposure to and practice with the sorts of technologies that can be used to develop computational thinking, and even elementary programming instruction
- Content knowledge - a better understanding of what computational thinking is and means.

This accords with the well renowned Technology Pedagogy and Content Knowledge (TPACK) model of teacher learning and practice (Mishra & Koehler, 2006). Responses also highlighted the need for both knowledge and practice. Some pre-service teacher responses highlighted the importance of affective considerations when designing professional learning - this is unfamiliar territory for many teachers who have never been taught or learnt computational thinking so it is important to sensitively scaffold their confidence.

8 Computational thinking research agenda

The notional machine has been an important and useful construct in computer science education (Robins et al, 2003) but there has been little if any work investigating how it can be used to understand and enhance computational thinking learning and teaching. There is urgent and pressing need to develop school students' computational thinking capabilities and teachers' computational thinking pedagogies (as established through the literature review and also by the data collected in this study). Thus there are several research opportunities to investigate how notional machines can inform our understanding of computational thinking and improve how it is learnt. Phrased as research questions, these are:

1. *How do notional machine constructs map to different computational thinking environments?* For instance, how do we define notional machines for computational thinking systems that may vary from Eclipse, to Scratch, to Beebots?
2. *How can 'visability' (du Boulay et al., 1989) be used to support computational thinking within computational thinking environments?* There may be several pedagogical strategies along the lines of including visual debugging-style output within programs to make the operations of the machine visible to students, thus enhancing their notional machine, but their effectiveness has not been investigated specifically from a computational thinking frame of reference.
3. *How can 'functional simplicity'(du Boulay et al., 1989) be best instantiated through easy to understand instructional sets?* This relates to the quality of introduction and explanation of how the machine works, and success may reside in illuminating exemplars, economical explanation, and powerful analogies). As Norman (1993) points out in order to help students form accurate mental models it is just as critical to decide what should be left out as what should be included.
4. *How can 'syntactic simplicity' (du Boulay et al., 1989) be fostered through accessible and uniform programming grammars?* This has been applied in some of the computational thinking tasks available through Code.Org, Scratch, Alice, and the like that use visual interfaces to write programs. Ideally teachers would utilise and even create non-computer based tasks that develop computational thinking abilities, in which case an understanding of syntactic simplicity is critical.
5. *How do we incrementally graduate the 'logical simplicity'(du Boulay et al., 1989) of the problems to be solved in line with the developing conceptions of the novice computational thinker?* (Scope and sequencing and timing issues are crucial so that students are neither bored nor overwhelmed - low floor, high ceilings, wide walls. Bower's Taxonomy of Task Types provides a one possible hierarchy for incrementing task complexity). The idea is to attempt to avoid problems relating to trying to learn about what computational thinking means, developing notional machines, learning languages, learning computing structures, and developing computational thinking process skills all at once (the 5 sources of difficulty identified by DuBoulay). Teachers need to know how to deconstruct computational thinking to avoid possible student cognitive overload.
6. *Where do 'misapplication of analogy', 'overgeneralisation' and 'interaction of parts' and potentially other types of errors commonly occur in the curriculum?* An understanding of these errors and where they occur helps teachers to better support the learning of computational thinking constructs. More importantly, how can we use these instances to create threshold learning experiences.

7. *How do researchers and educators accurately gauge novice mental models of target systems so that we can understand how to effectively guide learners towards correct conceptual models?* As Norman (1983) distinguishes between the correct conceptual model of the target system, the user's mental model of the target system, and the researcher's conceptualisation of the learner's model, understanding how to gauge and contrast these may be the key to understanding computational thinking learning and teaching, As Gotschi, Sanders and Galpin (2003) point out, domain specific models not only provide a point of reference to help identify non-viable mental models but also provide teachers with a resource to help develop their students' mental models.
8. *How do we best structure teacher professional learning in order to most effectively develop their computational thinking pedagogical capabilities?* This not only relates to the execution of professional learning courses, but also the development of an appropriate learning community around computational thinking pedagogy comprised of pre-service teachers, in-service teachers, researchers and developers. The pre-service teachers provide some general ideas, as does the literature, yet the devil will be in the detail.

Universities should be playing a key role in the development of teachers, methods and curriculum (Tucker, et al., 2003). A key element for a successful curriculum in schools is founding the resources and teaching practices on research into computer science education (Hazzan, Gal-Ezer, & Blum, 2008). In order to develop high quality computing curriculum is to have the course part of the research process, whereby teaching and learning data is used to iteratively refine the educational process (Hazzan, et al., 2008). Teachers can then become active participants in the research process. In Israel the teacher preparation process includes some research components, so that teachers can learn how to iteratively refine their teaching practices. In this way, research projects can contribute to the education of students, teachers and the educational community at large.

9 Concluding remarks

Accurate notional machines underpin successful performance in computational thinking. A structured rather than haphazard approach to examining notional machine understanding is required if we are to help students (and teachers) identify their misconceptions and take appropriate remedial action. Notional machine understanding is a prerequisite for effective teaching of computing, but not a guarantee. Teachers also need to have an appropriate repertoire of computational thinking pedagogies and technological knowledge in order to successfully teach computational thinking concepts and create a conducive learning environment for students.

This paper calls for further research into how the notional machine can be used to better understand and develop the computational thinking abilities of students as well as the computational thinking pedagogical capabilities of teachers. Results from this study suggest that pre-service teachers are ill prepared to teaching computational thinking, and need pedagogical strategies, experience with relevant technologies, and a better understanding of what computational thinking means. The computer science and education fields more generally need a greater understanding of how computational thinking is effectively learnt and taught in order to better support students and teachers.

The literature has identified visibility, functional simplicity, syntactic simplicity, logical simplicity and graduation as critical pedagogical issues, but how these relate to specific aspects of computational thinking learning is an open question. As yet there is no clear understanding of how to best describe and gauge notional machines, nor key places where novice misconceptions appear in the computational thinking curriculum. This paper is a call to action and an invitation to researchers interested in working on understanding the computational thinking research questions identified in this paper.

10 References

ACARA (2012). The shape of the Australian curriculum: technologies. Retrieved 17 August, 2014, from http://www.acara.edu.au/curriculum_1/learning_areas/technologies.html

ACARA (2013a). The Australian curriculum: Technologies information sheet. Retrieved 17 August, 2014, from http://www.acara.edu.au/curriculum_1/learning_areas/technologies.html

ACARA (2013b). Draft Australian Curriculum: Technologies Foundation to Year 10 Consultation Report. Retrieved 17 August, 2014, from http://www.acara.edu.au/curriculum_1/learning_areas/technologies.html

Akbulut, A. Y., & Looney, C. A. (2009). Improving IS student enrollments: Understanding the effects of IT sophistication in introductory IS courses. *Journal of Information Technology Education, 8*, 87-100.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48-54.

Battig, M. (2008). *Piltdown man or inconvenient truth? A two-year study of student perceptions about computing.* In Proceedings of ISECON.

BCS, T. C. I. f. I. (2010). *Consultation response to Royal Society's Call for Evidence – Computing in Schools*. The Royal Society: T. C. I. f. I. BCS.

Bell, T., Newton, H., Andreae, P., & Robins, A. (2012). *The introduction of computer science to NZ high schools: an analysis of student work.* In Proceedings of the 7th Workshop in Primary and Secondary Computing Education, (pp. 5-15): ACM.

Black, J., Brodie, J., Curzon, P., Myketiak, C., McOwan, P. W., & Meagher, L. R. (2013). *Making computing interesting to school students: teachers' perspectives.* In Proceedings of the 18th ACM conference on Innovation and technology in computer science education, (pp. 255-260): ACM.

Blank, G. D., Pottenger, W. M., Sahasrabudhe, S., Li, S., Wei, F., & Odi, H. (2003). Multimedia for computer science: from CS0 to grades 7-12. *EdMedia, Honolulu, HI*.

British Department for Education (2013). *The national curriculum in England*. Cheshire, UK: Crown.

Carter, L. (2006). *Why students with an apparent aptitude for computer science don't choose to major in computer science.* In ACM SIGCSE Bulletin, (pp. 27-31): ACM.

Curzon, P., McOwan, P. W., Cutts, Q. I., & Bell, T. (2009). *Enthusing & inspiring with reusable kinaesthetic activities.* In ACM SIGCSE Bulletin, (pp. 94-98): ACM.

du Boulay, B., O'Shea, T., & Monk, J. (1989). The black box inside the glass box: presenting computing concepts to novices. In E. Soloway & J. C. Spoher (Eds.), *Studying the Novice Programmer* (pp. 431-446). Hillsdale, NJ: Lawrence Erlbaum.

Gal-Ezer, J., & Stephenson, C. (2009). The current state of computer science in US high schools: A report from two national surveys. *Journal for Computing Teachers*, 1-5.

Gander, W., Petit, A., Berry, G. r., Demo, B., Vahrenhold, J., McGettrick, A., et al. (2013). *Informatics education: Europe cannot afford to miss the boat*.

Gotschi, T., Sanders, I., & Galpin, V. (2003). Mental models of recursion *Proceedings of the 34th SIGCSE technical symposium on Computer science education* (pp. 346-350): ACM Press.

Hazzan, O., Gal-Ezer, J., & Blum, L. (2008). *A model for high school computer science education: the four key elements that make it!* In ACM SIGCSE Bulletin, (pp. 281-285): ACM.

Lenox, T., Jesse, G., & Woratschek, C. R. (2012). Factors influencing students decisions to major in a computer-related discipline. *Information Systems Education Journal, 10*(6), 63.

Lockard, C.B and Wolf, M. (2012). Occupational employment projections to 2020. *Monthly Labor Review* , January 2012, 84-108.

Mayer, R. E. (1989). The psychology of how novices learn computer programming. In E. Soloway & J. C. Spoher (Eds.), *Studying the Novice Programmer* (pp. 129-159). Hillsdale, NJ: Lawrence Erlbaum.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education, 23*(3), 239-264.

Neuman, W. L. (2006). *Social research methods – Qualitative and quantitative approaches (6th Edition)*. Boston: Pearson Education.

Noble, A. (2012). Science the key to seize control of the future (26th December). *Sydney Morning Herald.* Retrieved from http://www.smh.com.au/opinion/politics/science-the-key-to-seize-control-of-the-future-20121225-2bv55.html

Norman, D. A. (1983). Some observations on mental models. In D. Gentner & A. L. Stevens (Eds.), *Mental Models*. Hillsdale, NJ: Erlbaum.

Norman, D. A. (1993). *Things That Make Us Smart*: Perseus Books.

PWC (2013). *The startup economy: How to support tech startups and accelerate Australian innovation*.

Repenning, A., & Ioannidou, A. (2008). Broadening participation through scalable game design. *ACM SIGCSE Bulletin, 40*(1), 305-309.

Repenning, A., Webb, D., & Ioannidou, A. (2010). *Scalable game design and the development of a checklist for getting computational thinking into public schools.* In Proceedings of the 41st ACM technical symposium on Computer science education, (pp. 265-269): ACM.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60-67.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172.

Schulte, C., & Knobelsdorf, M. (2007). *Attitudes towards computer science-computing experiences as a starting point and barrier to computer science.* In Proceedings of the third international workshop on Computing education research, (pp. 27-38): ACM.

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., et al. (2011). CSTA K-12 Computer Science Standards: Revised 2011.

Settle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., et al. (2012). *Infusing computational thinking into the middle-and high-school curriculum.* In Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education, (pp. 22-27): ACM.

Stephenson, C., Gal-Ezer, J., Haberman, B., & Verno, A. (2005). The new educational imperative: Improving high school computer science education. *Computer Science Teachers Association (CSTA), New York, New York.*

Tinapple, D., Sadauskas, J., & Olson, L. (2013). *Digital culture creative classrooms (DC3): teaching 21st century proficiencies in high schools by engaging students in creative digital projects.* In Proceedings of the 12th International Conference on Interaction Design and Children, (pp. 380-383): ACM.

Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2003). A Model Curriculum for K–12 Computer Science. *Final report of the ACM K-12 task force curriculum committee.*

Wilson, C., & Guzdial, M. (2010). How to make progress in computing education. *Communications of the ACM, 53*(5), 35-37.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.