The Center for Innovative Research in CyberLearning

CIRCL Primer - circlcenter.org

Computational Thinking

Contributors: Satabdi Basu, Eni Mustafara, and Katie Rich. Special thanks to the Cyberlearning 2016 Working Group for readings and resources (see Citation).

Printer-Friendly PDF | Google Doc for Comments | Questions? Contact CIRCL.

Back to Primers

Overview | Lessons | Issues | Projects | Resources | Readings | Citation

Overview

Computational thinking (CT), a term that experienced a surge of popularity in the 2000s, refers to a broad range of mental processes that help human beings find effective methods to solve problems, design systems, understand human behavior, and leverage the power of computing to automate a wide range of intellectual processes. Definitions vary, but there is general agreement that CT skills include the following:

- Formulating problems so that their solutions can be represented as computational steps and algorithms;
- Defining multiple layers of abstraction, understanding the relationships between the layers, and deciding which details need to be highlighted (and complementarily, which details can be ignored) in each layer when trying to understand, explain, and solve problems in different domains;
- Decomposing large complex tasks into manageable modular subtasks that supports parallel execution and multiple problem solvers;
- Iteratively developing solutions and systematically detecting and correcting errors;
- Analyzing the efficiency of various solutions;
- Reformulating seemingly difficult problems into solvable forms using reduction, transformation, recursion, and simulation.

While most existing definitions of CT describe it as a 'thought process', researchers in the field have increasingly realized the importance of focusing less on computational "thinking" and more on computational "doing". CT becomes evident only in particular forms of epistemic practices that involve the generation and use of external representations (i.e., representations that are external to the mind) by computational scientists. This pedagogical perspective is important since it means that engaging students in computational representational practices like the process of developing abstractions is required in order to support the development of their CT skills. This perspective also aligns with the 'learning-by-design pedagogy', which suggests that students learn best when they engage in the design and consequential use of external representations for modeling and reasoning.

Discussions of computational thinking emerged, in large part, out of desire by computer scientists to communicate the ways in which their discipline was more than programming. Coding or programming is one way to apply and practice aspects of computational thinking, but many aspects of computational

thinking — and therefore, computer science — can take place without engaging in coding. CT emphasizes conceptualization and developing ideas and algorithms for solving a problem rather than dealing with the rigid syntax of programming languages for producing artifacts that represent the solution to the problem. However, this does not mean that CT skills can be taught divorced from the use of computers. Though some CT concepts and principles can be introduced and explored through unplugged activities without the use of computers, prolonged use of such an approach deprives learners of crucial computational experiences. In other words, computers and other computational devices may not be synonymous with CT, but they are enablers of CT.

Also, it is noteworthy that though CT is often defined to draw on concepts fundamental to computer science, several CT skills are not exclusive to the field of computer science. For example, abstractions are used in all disciplines where modeling is a key enabler for conceptualization and problem solving, such as in science, engineering, mathematics, and economics. Similarly, logisticians and management scientists have studied scheduling extensively, and notions of tradeoff are central to the work of economists and engineers. Most disciplines involve problem solving, information retrieval and representation, modeling, debugging, testing, and efficiency considerations in some form or the other. Today, the wide spectrum of CT applications encompasses disciplines as diverse as science, mathematics, music, poetry, archaeology, and law.

Not surprisingly, CT is considered to represent a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use. With the proliferation of computers in our society, understanding the fundamentals of how computational solutions are designed is important for everyone. Jeannette Wing, who coined the term ‘Computational Thinking’ in 2006, argued that it should be included as a determinant of every child’s analytical ability along with reading, writing, and arithmetic by the middle of the 21st century. Just like young students initially learn to read so that they can later read to learn, they also need to learn to think computationally at an early age so they might later use it to learn complex concepts, represent solutions as computational steps, and solve problems using computational models and methods. It is no longer sufficient to wait until students are in college to introduce them to CT concepts. Students must begin to work with algorithmic problem solving and computational methods and tools during their K-12 years.

Increasing access to CT instruction is now widely discussed as a social justice issue. The stereotypical image of a computer scientist is that of a young white male bent over a keyboard, working in a room by himself. Focus on CT as a much broader, collaborative problem-solving process has potential to break this stereotype and broaden participation in the computer science field. Beyond broadening participation within computer science, CT skills can prove to be beneficial for all career fields. However, in spite of recognizing the need to introduce all students to CT concepts and practices from an early age, several CT-based programs and learning environments are still primarily used in informal and extracurricular settings like summer camps and after-school computer clubs. Engaging students in CT through motivational extracurricular CT-based activities may be a good first step, but CT eventually needs to be integrated into the K-12 curricula, either as a stand-alone discipline or integrated with existing disciplines like science and mathematics. Curricular integration will help remove the variables of self-selection, confidence, and willingness to opt for elective and extracurricular programs from the equation. This will provide all students, irrespective of gender and ethnicity, equal access to CT concepts and practices.

Key Lessons

Key lessons on Computational Thinking (CT) definitions and frameworks:

- CT skills have been defined as a three-dimensional framework comprising computational concepts, practices, and perspectives.
    - Computational concepts refer to elements, such as sequences, loops, parallelism, events, conditionals, operators, and data structures that are present in many programming languages.
    - Computational practices refer to activities, such as being incremental, reusing and remixing, testing and debugging, and modularizing and abstracting that designers use to create programs.
    - Computational perspectives, such as expressing, connecting, questioning, potential study and career path in computing, and personal relevancy of computing refer to worldviews that designers develop as they engage with digital media, and how they see themselves within the field and the realm of future careers.
- CT practices in Science and Mathematics contexts have been defined in the form of a taxonomy (Weintrop at al., 2016) consisting of four main categories:
    - **Data practices** – Collecting data, Creating data, Manipulating data, Analyzing data, and Visualizing data
    - **Modeling and simulation practices** – Using computational models to understand a concept, Using computational models to find and test solutions, Assessing computational models, Designing computational models, and Constructing computational models
    - **Computational problem solving practices** – Preparing problems for computational solutions, Computer programming, Choosing effective computational tools, Assessing different approaches/solutions to a problem, Developing modular computational solutions, Creating computational abstractions, Troubleshooting and debugging
    - **Systems thinking practices** – Investigating a complex system as a whole, Understanding the relationships within a system, Thinking in levels, Communicating information about a system, Defining systems and managing complexity.

Key lessons from CT-based research:

- CT skills can be taught in concert with skills in other domains, and this can make learning both easier than learning each separately. In fact, the ACM K-12 taskforce recommends integrating programming and computational methods with curricular domains, such as science and mathematics, rather than teaching programming as a separate topic at the K-12 levels. Using CT has also been added as a recommended practice by the Next Generation Science Standards.
- Well-designed computational thinking activities can enhance learning of other embedded topics, such as mathematics.
- Children as young as kindergarteners can engage in simpler aspects of computational thinking, such as sequencing.

- The language students use to engage in programming activities — in particular, whether a block-based or text based language is used — can impact what students understand about computational thinking.
- Assessments for measuring students' CT skills should not be tied to any programming language in particular, if possible.
- Increasing engagement and interest in CT may not necessarily be synonymous with increased understanding and use of CT concepts and practices.
- Teacher professional development and teacher-friendly resources and examples of CT are required in order to introduce CT in the K-12 curricula.
- Besides introducing CS curricula for K-12 students, integrating CT with existing science or mathematics curriculum can be an effective means for teaching CT skills, especially at the middle school level.

Issues

While the importance of introducing all students to CT skills from an early age is widely acknowledged, a number of issues still plague the field. Some important issues are listed below.

Efforts have primarily focused on engaging students in CT concepts and practices through motivating contexts like game-design, storytelling, robotics, and app-design in after-school workshops, summer camps, or as part of other extra-curricular activities, making CT accessible only to a selected few. Such efforts have naturally prioritized an interesting and engaging experience for students using computational tools instead of deep learning of CT concepts.

Many research studies on computational thinking demonstrate what students are capable of achieving at different grade levels, but there have been fewer systematic studies of how computational thinking instruction can be feasibly addressed in a typical school year. More studies of how programs can be implemented within the constraints of schools are needed.

Lack of systematic CT curricula have also resulted in dearth of research studying students' learning and developmental processes while learning CT skills and using CT-based learning environments.

Curricular integration of CT requires development of systematic CT assessments, an area that is under-investigated despite its importance being well recognized. Thorough assessments of computational thinking have yet to be developed, in part because the field lacks a shared understanding and vocabulary for what computational thinking entails.Such assessments can provide a thorough understanding of students' difficulties in using computational methods and tools, which can then lead to the development of systematic scaffolds to support students' development and use of CT skills.

While computational thinking is proclaimed to be a literacy appropriate for all students, it remains unclear where instruction for all should end and when instruction for only students with further interest in computer science should begin.

Projects

Examples of NSF Cyberlearning projects that overlap with topics discussed in this primer.

- Track 2: CS10K: BJC-STARS: Scaling CS Principles through STARS community & leadership development
- EXP: Learning Parallel Programming Concepts Through an Adaptive Game
- EXP: Understanding Computational Thinking Process and Practices in Open-Ended Programming Environments
- CAREER: Constructing Modern and Inclusive Trajectories for Computer Science Learning
- DIP: Extending CTSiM: An Adaptive Computational Thinking Environment for Learning Science through Modeling and Simulation in Middle School Classrooms

Principled Assessment of Computational Thinking – Applying the ECD approach to create assessments that support valid inferences about computational thinking practices, and is using the assessments and other measures to investigate how CS curriculum implementation impacts students' computational thinking practices.

Learning Trajectories for Everyday Computing (LTEC) – Developing learning trajectories for computational thinking (CT) in K-5 and addressing which aspects of CT might be integrated with math instruction in elementary school.

CTSiM: Computational Thinking using Simulation and Modeling – Leverages the synergy between CT and STEM in middle school contexts. Students use an agent-based visual programming language to build computational models of different science phenomena, which they can then simulate and compare against the results generated by expert simulations. Student learning is guided by online resources and an adaptive scaffolding framework.

CTSTEM – Promoting CT in high school science and math to empower all students to participate in a computational future.

Resources

Conferences & Organizations:

- ICLS – International Conference of the Learning Sciences
- SIGCSE – Special Interest Group on Computer Science Education
- SPLASH-E – Systems, Programming, Languages and Applications: Software for Humanity – Education Track
- AERA SIG/ATL and SIG/LS (Special Interest Groups in Advanced Technologies for
- Learning and Learning Sciences)
- Computer Science for All NYC

CIRCL Primer - circlcenter.org

- RESPECT – Research on Equity and Sustained Participation in Engineering, Computing, and Technology

Videos:

- Computational Thinking by Grady Booch (an ACM webinar)
- NSF 2016 Video Showcase videos on computational thinking
- Code.org Video Library

Social Media/Web:

- CS Education DIscussion Forum on Facebook
- Computing Education Blog by Mark Guzdial
- CIRCL Project Spotlight: Principled Assessment of Computational Thinking
- CIRCL Perspective on Matthias Hauswirth and his blog Learning to program? Or programming to learn?
- How to teach computational thinking, by Stephen Wolfram and response to the article: The keys to a well-rounded computer science education (by Hadi Partovi of code.org)

Curriculum:

- Exploring Computer Science (ECS) for high school students, with a focus on creating equal access to computing for students who are traditionally underrepresented in the computing workforce. A lot of schools are using this.
- Computer Science Principles (CSP) advanced placement course
- Mobile CSP – focuses on mobile computing
- Code.org curriculum for elementary, middle school, and high school
- Computational Thinking in Simulation and Model-Building (CTSiM) from Vanderbilt has Science + CT Curricular Units you can request access to
- Computational Thinking in Science and Math (CT-STEM) at Northwestern University has lessons and assessments you can request access to
- Exploring Computational Thinking (ECT) by Google for Education
- Zoombinis computational thinking game by TERC
- CS Unplugged learning activities

Standards and Assessment:

- K-12 Computer Science framework (release date September 2016)
- Advanced Placement Computer Science Principles
- CSTA K–12 Computer Science Standards
- Assessments for ECS Units 1-4 developed by PACT (need a CS10K account; free)

Teacher Resources:

- CS10K Community – where teachers of ECS and CSP can come to connect with each other and with the resources and expertise they need
- Computational Thinking for Educators – A free online course helping educators integrate computational thinking into their curriculum
- ScratchEd online community – where Scratch educators share stories and resources
- CSTA – Computer Science Teachers Association

Tools:

- Scratch, Dr. Scratch (assessment), and ScratchX (extensions)
- Alice
- Modkit
- Arduino
- ARIS
- MIT App Inventor
- VENVI
- Blocky Talky
- Ready, Steady, Code
- SiMSAM
- Sketching with Electronics
- App Lab
- Ear Sketch
- CodeCombat
- Tree House
- CodeAcademy
- StarLogo Nova
- Snap!
- Minecraft Modeling

Readings

References and key readings documenting the thinking behind the concept, important milestones in the work, foundational examples to build from, and summaries along the way.

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

Aho, A. V. (2012). Computation and computational thinking. The Computer Journal, 55(7), 832- 835.

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.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proceedings of the 2012 annual meeting ofthe American Educational Research Association, Vancouver, Canada.

National Research Council (2010). Report of a Workshop on The Scope and Nature of Computational Thinking. Washington, D.C.: National Academies Press.

National Research Council (2011). Report of a Workshop on the Pedagogical Aspects of Computational Thinking. Washington, D.C.: National Academies Press.

National Research Council (2004). Computer Science: Reflections on the Field, Reflections from the Field. Washington, D.C.: National Academies Press.

Margolis, J. (2010). Stuck in the Shallow End: Education, Race, and Computing. Cambridge, MA: MIT Press.

Margolis, J. & Fisher, A. (2003). Unlocking the Clubhouse: Women in Computing. Cambridge, MA: MIT Press.
Hemmendinger, D. (2010). A plea for modesty. Acm Inroads, 1(2), 4-7.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. Educational Researcher, 42(1), 38-43.

Basu, S., Biswas, G. & Kinnebrew, J.S. (2016). Using multiple representations to simultaneously learn computational thinking and middle school science. In Thirtieth AAAI conference on Artificial Intelligence. Phoenix, Arizona, USA.

Basu, S., Biswas, G., Sengupta, P., Dickes, A., Kinnebrew, J. S., & Clark, D. (2016). Identifying middle school students' challenges in computational thinking-based science learning. Research and Practice in Technology Enhanced Learning, 11(1), 1-35.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. Education and Information Technologies, 18(2), 351-380.

Basawapatna, A. R., Repenning, A., & Koh, K. H. (2015, February). Closing The Cyberlearning Loop: Enabling Teachers To Formatively Assess Student Programming Projects. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (pp. 12-17). ACM.

Bienkowski, M., Snow, E., Rutstein, D. W., & Grover, S. (2015). Assessment design patterns for computational thinking practices in secondary computer science: A first look (SRI technical report). Menlo Park, CA: SRI International.

Clements, D. H., & Sarama, J. (1997). Research on Logo: A decade of Progress. In C. D. Maddux & D. L. Johnson (Eds.), Logo: A retrospective (pp. 9-46). New York, NY: The Haworth Press.

Peters-Burton, E. E., Cleary, T. J., & Kitsantas, A. (2015). The Development of Computational Thinking in the Context of Science and Engineering Practices: A Self-Regulated Learning Approach. International Association for Development of the Information Society.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127-147.

Citation

Suggested citation:

Basu, S., Mustafaraj, E., & Rich, K. (2016). CIRCL Primer: Computational Thinking. In *CIRCL Primer Series*. Retrieved from http://circlcenter.org/computational-thinking

Special thanks to the Cyberlearning 2016 Working Group for contributing helpful resources and references. Members of that group, in alphabetical order:

- Gautam Biswas – Vanderbilt University
- Doug Clark – Vanderbilt University
- Jody Clarke-Midura – Utah State University
- Andre Denham – University of Alabama
- Brian Dorn – University of Nebraska, Omaha
- Aditya Johri – George Mason University
- Breanne Litts – Utah State University
- Caitlin Martin – Digital Youth Network
- Eni Mustafaraj – Wellesley College
- Keith Ostfeld – Children’s Museum of Houston
- Katie Rich – University of Chicago
- Patricia Schank – SRI International
- Michelle Wilkerson – University of California, Berkeley