Embedding Computational and Other Specialized Thinking in Knowledge Building

Carl Bereiter
IKIT, University of Toronto

Abstract: "Computational thinking," as it is currently being promoted, includes not only craft knowledge of software coding but also a kind of thinking developed by computer programmers that is believed to have more general value in many kinds of design and problem solving tasks. Virtually every discipline and skilled occupation has also been credited with distinctive ways of thinking worth learning for more general purposes. "Thinking like a lawyer," for instance, is an explicit learning objective in many law schools but may also have more general application in moral reasoning—not as a preemptive model but as an additional tool. This paper examines "thinking like" various kinds of specialists as an educational objective in terms of the close association between specialist thinking and substantive knowledge and the problems of extracting generalizable knowledge and skills from activities that highlight the exercise of craft skills. Embedding craft skills and "thinking like" a specialist within a larger Knowledge Building framework is proposed as a way of keeping larger objectives alive while obtaining the motivational and skill-learning value of concrete productive activity.

Introduction

Although developing knowledge and skills related to digital media has been an educational objective since the advent of personal computers and appears on every list of 21st century skills, the emergence of computational thinking as an educational objective (Denning & Tedre, 2019) represents a "rise above" the familiar goals. It aims at both a higher level of craft skills (ability to design and code software rather than only use it) and a higher level of cognition (internalizing the kind of thinking programmers do and applying it to design thinking and problem solving more generally). Computational thinking thus becomes something that should not only have a role in Knowledge Building but something Knowledge Building should be about—in the same sense that Knowledge Building is about ecology, evolution, complexity, knowledge creation, human rights, and other topics worthy of study.

Thinking like a programmer is said to differ from ordinary thinking in its emphasis on breaking a complex problem down into smaller parts that can be addressed separately. It is also said to involve thinking of problems in a way that a computer can help solve them. The two are quite different. The first is a heuristic, a strategic move that may or may not be helpful with a particular problem. A heuristic may be thought of as a conceptual tool that can be added to a person's thinking resources without any fundamental change to the person's other resources or way of thinking. Thinking of ways that computation could help in achieving some elusive objective, however, is neither a conceptual tool nor a skill. It is a design task calling for some level of invention and/or some new insight. It is, thus, a Knowledge Building/knowledge creation challenge. It means becoming, at least in some situations, a different kind of thinker.

Thinking like a programmer belongs to a fairly large and unorganized problem space that may be called "thinking like a specialist." If you enter the phrase "learning to think like" into a web search engine along with the name of some discipline or occupation you will discover discussions about the nature and value of "thinking like" in an impressive variety of fields. All the major academic disciplines will be represented, along with learned professions such as law and medicine. But "thinking like" applies to less conspicuously intellectual fields. In order to become more alert to dangers on the street, we are encouraged to "think like a cop." To safeguard our home against theft we are encouraged to "think like a burglar." There is also thinking like the practitioner of a particular trade or craft: e.g., thinking like a mechanic (Downtown Autobody, 2013) or like a weaver (Seitemaa-Hakarainen, Viilo, & Hakkarainen, 2010).

The idea running through all these discussions is that practitioners in a demanding field develop specialized knowledge skills attuned to the nature of their work but potentially of benefit outside it. Deliberate efforts to teach these specialized kinds of thinking are rare, however. Law is the most notable exception. Learning to think like a lawyer appears to be an objective that runs through entire law degree programs. In *Thinking Like a Lawyer: A New Introduction to Legal Reasoning,* Schauer *(2009)* discusses such distinctive characteristics of legal reasoning as the weight given to rules and precedents. He is clear that this does not always lead to the fairest judgements in particular cases, that the law is more concerned with fairness over a wide range of cases. However, this wider view of fairness could well have a place alongside reasoning from general moral principles, case-based reasoning, and reasoning by analogy in any sort of knowledge building dealing with human policy issues.

Exploring Signature Pedagogies: Approaches to Teaching Disciplinary Habits of Mind (Gurung, Chick, & Haynie, 2009) has chapters on teaching discipline-specific thinking in all the major subjects at the post-secondary level. At the school level, the only serious effort appears to be thinking like a historian: e.g., the "Reading Like a Historian" program (https://sheg.stanford.edu/history-lessons). In contrast, the typical school approach to teaching "scientific method" appears designed to teach students *not* to think like a creative scientist (Kirschner, 1992). School mathematics, with its emphasis on executing algorithms and solving specific assigned problems, gives little inkling that there is such a thing as mathematical thinking, an exception being work that engages students in discovering patterns and formulating rules to describe them (Moss & Beatty, 2006). Literature courses could be greatly enhanced by getting students to "think like a literary critic"—which is not the same as having an opinion about a literary work and defending it but involves going more deeply inside the work.

The Role of Substantive Knowledge in "Thinking Like"

Thinking skills approaches, such as those going by the name of "21st century skills" (Binkley, Erstadt, et al., 2012), typically focus entirely on process. Substantive knowledge is treated as something on a different dimension (Anderson & Krathwohl, 2001), something for thinking skills to act upon. However, in thinking like a programmer, a zoologist, a weaver, or other of the many specialized kinds of thinking, substantive knowledge of the domain is inseparable from thinking processes. This is evident in what is perhaps the most highly developed set of heuristics for thinking like a specialist, TRIZ (Orloff, 2013). TRIZ consists of 40 principles for producing inventive solutions to engineering problems. One principle is "Intermediary" and another is "Cheap, short-lived objects." These and a growing number of sub-principles may have the general effect of helping a person look at a design problem in different ways. This is the process aspect. But trying to apply the principles to an actual design problem will be futile unless one has abundant knowledge not only of the particular problem but other more remotely related knowledge enabling one to discover, for instance, a cheap, temporary object or material that could be placed between moving parts to mediate and not disrupt their interaction. The same is true of other domain-specific problem-solving heuristics—for instance, the geometric and physical knowledge needed to make effective rather than time-wasting use of Polya's (1945) suggestion to draw a picture to aid in solving a mathematics problem or the syntactical knowledge and skill required to make use of Christensen's (1963) "generative rhetoric" in producing more readable and effective sentences. In summary, learning to think like a specialist in a variety of disciplines and lines of work has potential to add greatly to a person's cognitive resources for productive thought; heuristics and principles can provide a way into different kinds of specialized thinking, but they need to be intimately linked to substantive knowledge and skills. Commercial instructional materials as well as theoretically-grounded "scripts" (Kollar, Fischer, & Slotta, 2007) seldom come close to the level of detailed heuristics and substantive content required to help the learner begin to "think like" a practitioner of the subject being taught. The theme of this year's Summer Institute—save lives, save the planet—brings in an incredible variety of specialties and specialized knowledge. Work on this theme will expose students to many different kinds of specialized thinking but it will be important to keep the ways of thinking attached to substantive knowledge.

Embedding Practical Skills in Knowledge Building

Many kinds of specialized thinking have a practical skills component, often including manual skills. Craft skills, obviously important in "thinking like" an expert craftsperson, are also important in more academic disciplines. There is the chemist's craft skill in diagramming molecules and also skill in handling laboratory glassware, the electronic engineer's skill in producing and using wiring diagrams and also neatly soldering connections, the biologist's skill in preparing slides and using a microscope. To a some extent craft skills can be learned separately from the conceptual part of specialist thinking, but for the student, the craft skills enable the kinds of experimentation and exploration that develop the conceptual part. Therefore, they commonly take precedence. This can be detrimental, however, if the craft activity gains so much attention that the conceptual part gets no attention. This seems to be a common phenomenon in school learning activities, where, for instance, producing a poster to display what has been learned leads to almost all the attention being devoted to the craft of poster-making and little to the content. Elementary school students will talk about "making" an experiment, with little sense of a knowledge building purpose or of scientific thinking (Carey, et al, 1989).

In computational thinking— learning to think like a software developer—the related craft skill is, of course, skill in producing code that actually works. In some cases, this is the whole point of instruction, but education for computational thinking has the larger purpose of using this craft skill for more general educational purposes. Thus, there is having students create computer games to teach oceanography (Yarnall & Kafair, 1996) or using robot programming in mathematics or social studies (Khanlari & Scardamalia, 2019). The difficulty that commonly arises, however, is that the programming activity is so motivating and absorbing that it is difficult to get students to give attention to the academic subject matter, which typically lacks the sensational appeal and challenge of the programming activity. The study by Yarnall and Kafai (1996) illustrates the problem dramatically. Grade 5 students were asked to produce computer games to teach younger students about the ocean environment. Although the activity proved highly motivating, the games students produced were simplistic question-answer games (thus a step backward from the intended constructivist learning approach) and the online discussion dealt with programming and avoided scientific issues. Similar results have been reported with the very popular programming environment, Scratch. In summary, programming activities by school students develop craft knowledge of programming but little disciplinary knowledge, knowledge of gaming or whatever art is involved, or higher-level computational thinking. Developing craft knowledge may be sufficient justification for giving programming a place in the curriculum, but it should be possible to go beyond this, to make programming a more significant contributor to general educational development.

Knowledge Building is the obvious way to broaden the scope of intellectual engagement, but it is still likely to suffer in a head-to-head competition with coding activities, such as those that have made Scratch not just a coding language but the basis for a vast community of students doing interesting things with it. A potential solution is to start Knowledge Building *before* the programming activity, which then needs to be carried out within a context of collaborative knowledge building—sustained creative work with knowledge and ideas in the domain of study.

As a hypothetical case, let us assume that the students have previously learned the basics of Scratch and have used it to produce simple graphics. The topic of study now is plant nutrition. If Scratch is brought into an initial stage of work on this topic, the result is likely to be pictures of plants, in the best cases animations showing the development from seed to leafy plant and then to flower or fruit. In the process little or nothing will have been learned beyond simple facts of plant development. If, however, Knowledge Building is done from the beginning before there is anything to do with computation, there will be questions, consulting of authoritative sources, experiments, more and deeper questions, and much presenting and building on students' own ideas about plant nutrition. Questions will arise about how plants can get nutrients from the earth whereas animals cannot and students will encounter the idea that plants produce their own food through something called "photosynthesis," which is an idea that calls for considerable explanation building. At some point the question is introduced (hopefully by a student): Is there any way we could use Scratch to help build our understanding of plant nutrition? This then becomes a question to pursue as part of this particular knowledge-building project. Ideas brought up will be considered and tried out in Scratch. Animations may now show x-ray views of what is hypothesized to go on in leaves, roots, stems or trunks. These may now be discussed not only from the standpoint of how the program was written but more crucially from the standpoint of how well the animation fits with known facts. Improvements will not be limited to improvements in style and program operation but will be progress toward better explanations of the phenomena represented. What students carry away with them from the experience should include increments in computational thinking and also a substantial advance in understanding of the plant world surrounding them.

Computational Thinking in a World of Neural Nets

The discussion so far has ignored the elephant in the room of computational thinking—artificial neural nets or ANNs. These are behind most of the news-making innovations in digital technology—driverless cars, handheld speech translators, face recognition, championship players of board games—and promise to play a rapidly increasing role in our lives. The computational thinking involved is of a fundamentally different kind from the kind involved in older software and that students are learning in school (Gurney, 2001). This is not to say that what students learn in working with Scratch or programming a mobile robot is useless, but it is definitely 20th, not 21st century computational thinking. Schools ought to be addressing the revolutionary implications of ANNs not only through modernized work on computational thinking but through a more general overhaul of disciplinary education. ANNs are part of a closely related family of developments in contemporary thought that include complex systems theory (Wilensky & Jacobson, 2014), dual process theory in cognitive science (Stanovich, 2004, pp. 31-80)), "hot cognition" (Thagard, 2006) in which emotions are not just influences on thought but are an integral part of it, a neuroscience in which real neural nets are featured players, and an approach to learning that gives the intentional development of intuitive knowledge a vital role (Brown, 2017). How to address all of these in a way that recognizes their interconnectedness and yet is adapted to the capabilities of students at different stages of development is a challenge far beyond the scope of this paper. While no educational approach has an off-the-shelf response to this challenge, Knowledge Building is arguably the approach best equipped to develop one.

References

Anderson, L. W., and D. R. Krathwohl, Eds. 2001. "*A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives: Complete Edition*." New York : Longman.

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., Rumble, M., (2012). Defining Twenty-First Century Skills. In B. McGaw & E. Care (Eds), *Assessment and Teaching of 21st Century Skills* (pp 17-.66). New York, NY: Springer.

Brown, J.S. (2017). *Sensemaking in our post-AlphaGo world.* Retrieved from http://johnseelybrown.com/SensemakingStanford.pdf

Carey, S., Evans, R., Honda, M., Unger, C., & Jay, E. (1989). An experiment is when you try it and see if it works**:** a study of grade 7 students' understanding of the construction of scientific knowledge. *International Journal of Science Education, 11,* 514-529.

Christensen, F. (1963). A generative rhetoric of the sentence. *College Composition and Communication, 14(3), 155-161.* DOI: 10.2307/355051

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

Downtown Autobody. (2013). There's no substitute for thinking like a mechanic. https://downtownautobody.com/substitute-thinking-like-mechanic/

Gurney, K. (2001). Computers and symbols versus nets and neurons. Published online at https://www.semanticscholar.org/paper/1-%3A-Computers-and-Symbols-versus-Nets-and-Neurons-Gurney/4975b2ac01a680c3b9e526cb11e975031863948d

Gurung, R. A. R., Chick, N. L., & Haynie, A. (2009). *Exploring signature pedagogies: Approaches to teaching disciplinary habits of mind.* Sterling, VA: Stylus Publishing.

Khanlari, A. & Scardamalia, M. (2019). Knowledge Building, robotics, and math education. In proceedings of the 13th International Conference on Computer Supported Collaborative Learning (CSCL) 2019, Volume 2 (pp. 963-964). Lyon, France: International Society of the Learning Sciences.

Kirschner, P. (1992). Epistemology, practical work and academic skills in science education. *Science and Education, 1,* 273–299.

Kollar, I., Fischer, F., & Slotta, J. D. (2007). Internal and external scripts in computer-supported collaborative inquiry learning. *Learning and Instruction*, *17*(6), 708–721.

Moss, J., & Beatty, R. (2006). Knowledge building in mathematics: Supporting collaborative learning in pattern problems. *International Journal of Computer-Supported Collaborative Learning, 1*(4), 441-465.

Orloff, M. A. (2013). *Inventive thinking through TRIZ: A practical guide.* New York, NY: Springer.

Polya, G. (1945). *How to solve it.* Princeton, NJ: Princeton University Press.

Schauer, F. (2009). *Thinking like a lawyer: A new introduction to legal reasoning*. Cambridge, MA: Harvard University Press.

Seitamaa-Hakkarainen, P., Viilo, M., & Hakkarainen, K. (2010). Learning by collaborative designing: Technology-enhanced knowledge practices. *International Journal of Technology and Design Education, 20* (2), 109-136.

Stanovich, K. E. (2004). *The robot's rebellion: Finding meaning in the age of Darwin*. Chicago: University of Chicago Press.

Thagard, P. (2006). *Hot thought: Mechanisms and applications of emotional cognition.* Cambridge, MA: MIT Press.

Wilensky, U., & Jacobson, M. J. (2014). Complex systems and the learning sciences. In K. Sawyer (Ed.), *Cambridge handbook of the learning sciences* (pp. 319-338). New York: Cambridge University Press.

Yarnall, L., & Kafai, Y. (1996, April). *Issues in project-based science activities: Children's constructions of ocean software games*. Paper presented at the annual meeting of the American Educational Research Association, . New York: http://www.gse.ucla.edu/kafai/Paper_Kafai%2FYarnall.html.