How young is too young?

Michal Armoni
Science Teaching Department
Weizmann Institute of Science

The downward evolution of computing education

Undergraduate programs ⇒

High schools ⇒

Middle schools ⇒

Primary schools ⇒

Kindergarten…

How young can one learn computing ?

Subsets of computing

Subsets that maintain the essence of computing

$\Updownarrow$

Computing is…?

Subsets of computing

computing
≠
programming

. . .

computing
≠
ICT

computing ≅
sequence
programming
?

computing ≅
computing –
programming
?

The purpose of K-6 computing education

- Increasing motivation to study computing

  $\Rightarrow$ Increasing enrollment in high school computing courses

  $\Rightarrow$ Increasing enrollment in college and university computing programs

1. #computing graduates << #computing jobs
2. A false image of computing $\Rightarrow$ negative attitudes
3. Career intentions are set early

The purpose of K-6 computing education

- Teaching computational thinking

1. The ideas, strategies, thinking patterns and heuristics employed in computing constitute a distinguished thinking paradigm
2. This thinking paradigm is a powerful tool in general contexts
3. Teaching early to affect one's toolbox

The purpose of K-6 computing education

Teaching computational thinking

Increasing motivation to study computing

knowledge acquisition

motivation

To-do list…

- Why?
- When? A reasonable starting point
- What? Any image-preserving subset of computing
- How? An effective teaching

…

Looking for a reasonable starting point that allows an effective teaching of an image-preserving subset of computing

Bruner (1960)

"Any subject can be taught effectively in some intellectually honest form to any child in any stage of development"

To-do list…

- Why?
- When? Any k-6 level
- What? Any image-preserving subset of computing
- How?

Looking for a reasonable starting point that allows an effective teaching of an image-preserving subset of computing

Any stage of development?

⬇

Child’s cognitive development

Piaget

Relevant stages of cognitive development

- Preoperational stage(1.5 to 7)

  Do not yet understand concrete logic, and cannot manipulate information mentally, only physically

- Concrete operational phase (7-12)

  Can only solve problems that apply to actual (concrete) objects or events, and not abstract concepts or hypothetical tasks. Develop an ability to think abstractly and rationally, but only about concrete or observable phenomena

Relevant stages of cognitive development

- Formal operational stage(12 and up)
  - ♦ Abstract thinking, reasoning that is independent of content
  - ♦ Meta thinking (reflection)
  - ♦ Thinking about what may hold, even if it does not hold now (scientific hypotheses)

Abstraction – the essence of computing

- "[T]his oscillation between “the representation” and “what it stands for” is an intrinsic part of the programmer’s game.” (Dijkstra, 1975)
- Thinking like a computer scientist […] requires thinking in multiple levels of abstraction.“ (Wing, 2006)
- "they [natural computer scientists] are individuals who can rapidly change levels of abstraction, simultaneously seeing things “in the large” and “in the small”" (Knuth, in Hartmanis, 1994)

If abstraction is a part of a computing primary curriculum

- Piaget – students must be in the formal operational phase → ~~primary school~~
- Bruner – any stage of development → primary school ✓

Piaget or Bruner?

If abstraction is not a part of a computing primary curriculum

not an image-preserving (honest) teaching

Importing from mathematics

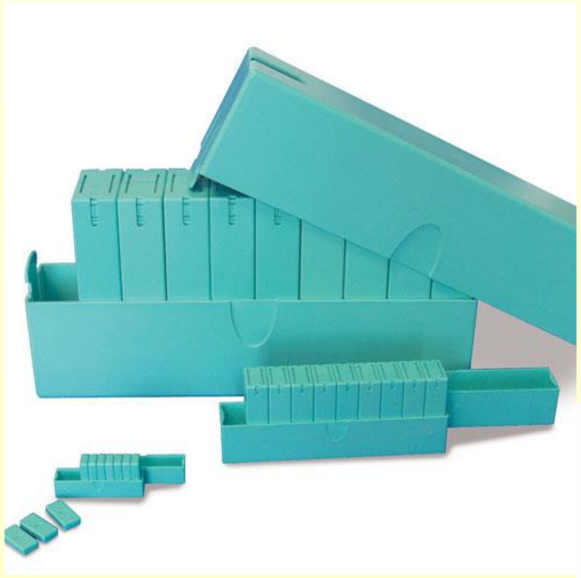
- Kindergarten: counting till 100, basic arithmetic till 20, without carry-over. Learning in context through spontaneous play.
- Primary school: Arithmetic (natural numbers, integers, fractions)

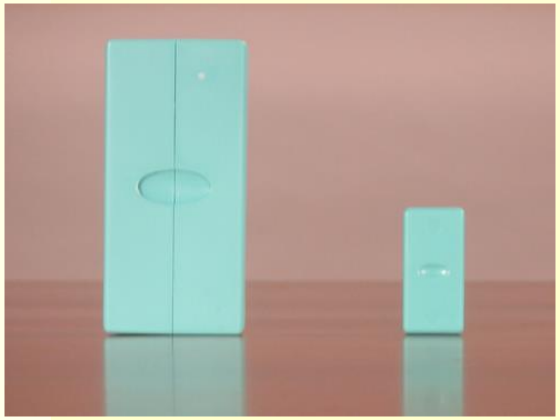
- Middle school: Algebra, Geometry (inc. proof design)
- High school: Calculus, Trigonometry…

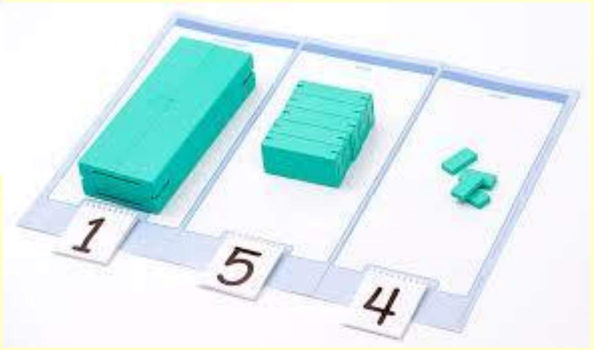
But some concepts are taught in a spiral way (function, multiple representations…)

Importing from mathematics

DiGi blocks as a scaffolding for learning decimal representation of natural numbers







Importing from mathematics

Decimal representation of numbers is a relatively abstract concept, assumed to be too difficult for kindergarten students.

Kindergarten students could effectively learn decimal representation of numbers, when using DiGi blocks. They generalized the concept and transferred to other contexts. (Sela, 2003).

Importing from mathematics

Schoenfeld (1986)

Teaching abstract concepts using concrete reference systems (e.g. concrete colored block systems for teaching numbers and number operations).

Schoenfeld points at a few factors and obstacles that complicate such didactic learning processes.

Importing from mathematics

- The more natural a representation is, the harder it might be to abstract the underlying ideas.
- In most cases, the concrete world does not completely map into the abstract world, which might cause confusion, difficulties, and even misconceptions. Honest; image preserving
- Children fail to connect the concrete world and the "real" abstract world. In a sense, for them the two worlds can live side by side, with no connection between them (concrete geometrical construction vs. geometry theory).

The arching theme of

fundamental ideas

(Bruner 1960; Schwill, 1994)

Abstraction: A fundamental idea of computing

fundamental ideas of computing

- define the essence of the discipline
- are a teaching and learning challenge (research indicates that even high-achieving undergraduate students have some difficulties with black boxes (Armoni, Gal-Ezer, & Hazzan, 2006)
- are a research challenge

And many more: Concurrency, decomposition (top-down), composition (bottom-up)

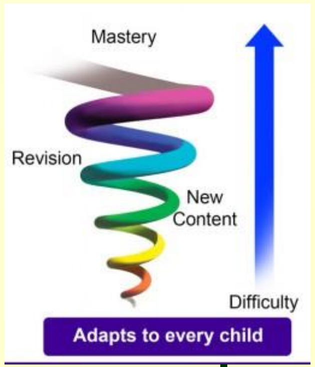
Fundamental ideas

- The horizontal criterion (wide applicability, in multiple ways and various contexts
- The vertical criterion (different intellectual levels)
- The criterion of time (historical development)
- The criterion of sense (related to everyday language and thinking)

Bruner’s framework for teaching: along fundamental ideas

- Fundamental ideas should be taught early
- In an age-appropriate way
- Revisit again and again in higher developmental levels
- Reflect backwards, to previous acquaintances and make explicit connections

Spiral teaching



- constructivism
- Very difficult to implement (coordination and collaboration along the spiral)

To-do list…

- Why?
- When? Any k-6 level
- What? Any image-preserving subset of computing
- How? Spiral teaching along the axes of fundamental ideas

$$\text{A }\overbrace{\text{reasonable starting point}}^{\text{Any k-6 level}}\text{ that allows an }\overbrace{\text{effective teaching}}^{\text{Spiral teaching}}\text{ of an }\underbrace{\text{image-preserving subset}}_{\text{Fundamental ideas}}\text{ of computing}$$

The case of abstraction

Teaching abstraction for primary school students:

- Finding an age-appropriate (non-abstract, concrete) way to teach abstraction
- While achieving effective teaching
- And not causing any damage (stable misconceptions, incorrect image)

? research !

Teaching framework

Teaching computing for primary school students:

Unplugged programming
"Turing machine" (de Oliviera et al., 2014)
Nursery rhymes (DiVano & Mirolo, 2011)
Code Monkeys Island

Robots
Beebot

Game design
Kodu

Block-based programming
Scratch
Alice

Unplugged activities
CS unplugged
CS4fn
Physical computational objects

Text Programming
Logo
Code Monkeys

combinations

Experience reports and empirical research

- 4th grade students studied computing (45 min. × 10 weeks) using a variant of Logo and Basic. Positive impressions (Frost, 2007)
- 3rd grade and up – computational thinking (without programming), using formal notation for reasoning. Spiral planning (recursion, non-determinism, abstraction). (Lu & Fletcher, 2009)

Experience reports and empirical research

- 5-day introductory parallel programming course to 3-4-graders. “As computer hardware becomes increasingly parallel, there is a greater need for software engineers who are proficient in designing parallel programs, and not just by “parallelizing" sequential designs. Teaching parallelism first is an important step towards educating tomorrow's programmers.” (Gregg et al., 2012)
- “None of our students had any difficulty with parallelism”

Experience reports and empirical research

Middle school students taking a course on computer science concepts and ideas through Scratch had significant difficulties with concurrency (Meerbaum-Salant, Armoni, & Ben-Ari, 2013)

Age-appropriate way?

Examine the tools to identify any abstract and challenging ideas. For example, concurrency and Cartesian coordinate system are inherent in Scratch.

Experience reports and empirical research

- Using Storytelling Alice to teach elementary (grades 1–5) computer science concepts. (Gardner & Feng, 2012)
- International survey among the CS community regarding their attitudes and opinions towards teaching computing in primary schools.
  - A majority perceived that anybody could learn to code, with effort and motivation, however, more advanced levels of programming require mathematical logic, a desire and ability for problem-solving and abstract thinking.

Experience reports and empirical research

- Assessing computational thinking (in the context of Scratch) of 4th-graders (Seiter, 2015)
  - ♦ There was much difficulty with a question that did not require conditional execution or repeated execution. It required synchronization of a conversation between 2 or 3 sprites and costume changes.
  - ♦ Skills in reading and arithmetic not age-corresponding ⇒ very low achievements on the assessment tasks

Experience reports and empirical research

- Analyzing Scratch projects of 1st-6th graders (Seiter & Foreman, 2013)
  - ♦ Initialization – 3rd-4th grade (and getting better)
  - ♦ Conversation – 3rd-4th grade
  - ♦ Variables – 5th-6th grade
  - ♦ User interaction – 5th-6th grade
- Teaching abstraction using Scratch in 7th grade, an on-going research project (Statter & Armoni)

Experience reports and empirical research

- 3rd-graders could not solve reading / understanding tasks with blocks of BYOB/snap!, but could solve similar tasks with the same level of difficulty that were using everyday language (Sabitzer, Antonitsch, & Pasterk, 2014)
- Reading code in visual block-based environments is complex. 4th-grade students can interpret such programs, but not all (Dwyer et al., 2015)

Experience reports and empirical research

- 4th-graders: verbal algorithmic task (Dwyer, 2014)
  - ♦ Could recognize problems in other students' sequences
  - ♦ Could come up with algorithms for "small" problems. Failed when the input got larger.

Experience reports and empirical research

- Teaching concepts of graph algorithms age 5 and up (Gibson, 2012)
  - Concrete problem-based learning
  - Graph homomorphism (equivalence class)
  - Subgraphs (specific)
  - How should I describe this graph so I can solve the problem easier/better (the connection between data structures and algorithms)

Experience reports and empirical research

- Duncan, Bell & Tanimoto (2014)
  - Surveying computing curricula
  - Surveying research
    - 4th-5th grade – a good age too start learning computing
  - Splitting programming : 1st-2nd grade – sequencing
    - Incorrect image?
    - motivation

Experience reports and empirical research

"learning to program is like learning to read"

Logical thinking

A program = a general existence proof

Abstraction

Even if the proof is easy, the difficulty lies in understanding the concept of such a proof

"Coding is easy and can be learnt in a very short time"

?

Empirical research

Thank you
Enjoy WiPCSE 2015