# Identifying the Big Ideas of K-12 Computer Science Education

Tim Bell, University of Canterbury, NZ
Paul Tymann, Rochester Institute of Technology, New York, USA
Amiram Yehudai, Tel Aviv University, Israel

סמינר מורים מובילים 11/7/2018

# The big ideas in CS

- When teaching cs, we may focus on details and lose sight of the bigger picture.
  - Especially in K-12
  - Why teach “coding”?, binary numbers?, quicksort?
- Goal: present a list of the 10 “big ideas” of CS
  - based on input from curriculum designers and computer science education experts around the world.
- Presented in a way a teacher can engage with
  - use to relate topics taugt to context of a bigger picture.
- Project started by Tim Bell, U. of Canterbury, NZ
  - Author of *CS Unplugged* , and CS Field Guide

## My goal of this lecture

- Ongoing research, to get feedback of active CS teachers
  - at different age groups
- Should we drop some of the 10 items?
- Are we missing ideas?
- Are these ideas touched upon?
- Is the level of discourse suitable?

## Possible take away

- It may useful for a teacher to think – why do I teach this? What is the context?

## Special challenge

- In some countries (eg. NZ), CS teachers, especially for early ages, are not CS graduates.
- Traing non CS teachers to teach CS in, eg. 1st grade is challenging
- (Deciding what/how to teach is equally challenging)
- What is the situation in Israel?

## Big ideas of science education

- [Harlen *et al.*, 2015] had the goal of identifying: *"the key ideas that students should encounter in their science education to enable them to understand, enjoy and marvel at the natural world."*
- Identified 10 big ideas, such as
  - "All matter in the Universe is made of very small particles",
  - "Organisms are organised on a cellular basis, and have a finite life span."
- It also identified four ideas *about* science, relating to scientific method and the impact of science on the world [Harlen *et al.*, 2015].

## The big ideas in CS

Borrowing from “Big ideas of science education” our goal is:

*The key ideas that students should encounter in their computing education to enable them to understand, enjoy and marvel at the digital world.*

## The goal

- The big ideas presented here are meant to identify the concepts that practicing computer scientists see as being key for newcomers (especially students) to understand the discipline.
- For example, a common misconception outside the discipline is that computer science is mainly about programming, whereas it can be articulated as part of a bigger picture.
- Our goal is to paint the bigger picture of what computer science is about in terms that make sense to a curious parent, school administrator or educationalist.

## (Misconceptions)

## Big ideas are/are not

- Not meant as general principles, discipline areas, or curriculum topics
  - but rather ideas that capture the culture of the subject.
- Not intended to cover *every* idea in the study of CS
  - But does have broad coverage
- Big ideas for computing education should be stable
  - if major changes are needed, the program is too trendy and technological or core not well selected [Armoni 2013].
- The big ideas are intended to focus on core topics.
- The ideas do not reflect the weight given to a topic.
  - For example, programming

## The big ideas may help

- To ensure that important concepts are included in new curricula as they are designed.
- To influence the allocation of teaching resources
- To enable lay people to see how CS is different from other disciplines.
- Teachers and others will see how a topic fits into the big picture
  - – cover the entire discipline as a whole and not as a disparate collection of unrelated ideas.

## Audience and Terminology

- To inform people outside the discipline, the text of the big ideas tries to avoid technical terms that an informed lay person might not be aware of.
- The list of ideas has been developed by soliciting input from computer scientists involved in education, and tested with teachers who do not have formal education in computer science.
- Aim also at schools that teach Digital Technologies, Computing, and Computational Thinking (not just CS)
- Use “digital devices” instead of “computer”
  - broader range of computing machines eg. smartphones

## Related work

- What to cover in CS courses is debated for many years
- The ACM CS curriculum (for universities and colleges) covers topics chosen by a panel of experts to define a "body of knowledge" [Sahami *et al*., 2014].
- Recent AP course "CS Principles" [College Board, 2016, Kick and Trees, 2015] focuses on overview of the subject, rather than specific skill of programming.
  - **A**dvanced **P**lacement courses are advanced US high school courses that get credit in colleges

## Existing “general principles”

- “Great principles of computing” by Peter J. Denning and Craig H. Martell (2015),
  - six principles (communication, computation, recollection, coordination, evaluation and design).
  - complementary to the approach presented here
  - a useful tool for validating the big ideas developed here.
  - expanded into 11 in their book, overlap with our big ideas
- “Fundamental Ideas of CS” [Schwill 1994]
  - Focused on software development - three main areas (algorithmization, structured dissection, and language)

## Existing “general principles”

- The principles of Computational Thinking (CT) :
  - six facets of computing [Wing, 2006, Tedre and Denning, 2016].
  - Logic, Evaluation, Algorithms, Pattern, Decomposition and Abstraction
  - intended to identify the thinking needed to engage with the ideas of computing

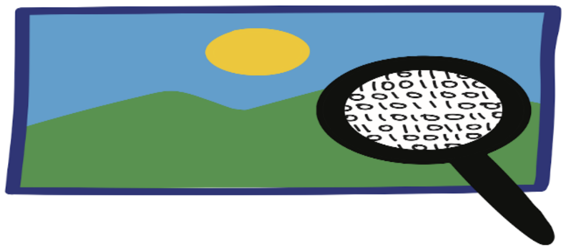
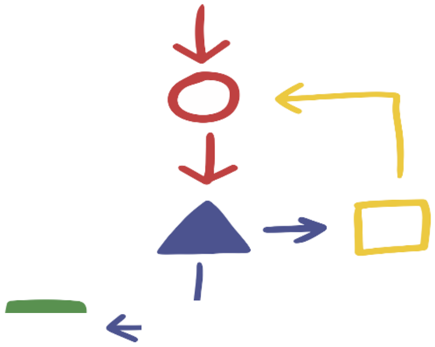
## Developing the big ideas

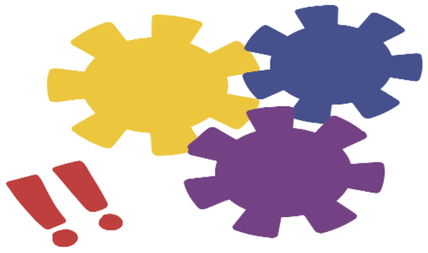
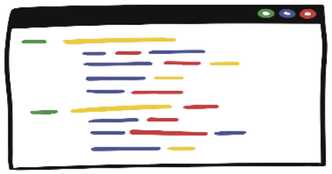
- Started by Tim Bell
  - Author of *CS Unplugged* , and CS Field Guide
- Input from community
- Mostly in CS education conferences
  - People were asked to write their ideas
  - Circulated to participants
  - New ideas added, some merged
- Project joined by Paul Tymann, RIT, USA and A.Y.
  - Experience in CS K-12 curriculum (NZ, USA, Israel)

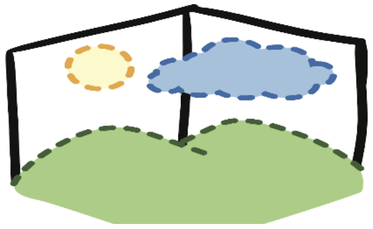
## About the big ideas

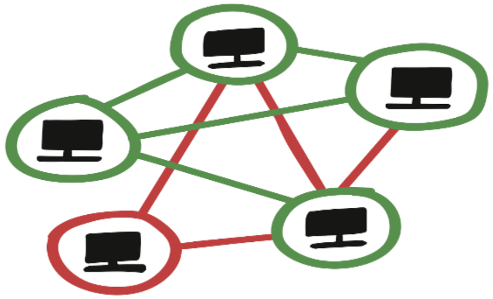
- Order of ideas reflect logical development
- Start from data and algorithms, and then how they interact
- The idea of “human centric” in the middle
  – Could be made first, but terminology is needed
- Focus on common models of computing
  – Include mobile, supercomputers, embedded systems
  – But eg. quantum computing not covered
- Abstraction usually mentioned as important
  – Applies to almost each of the ideas
  – not as a separate idea

# The 10 big ideas

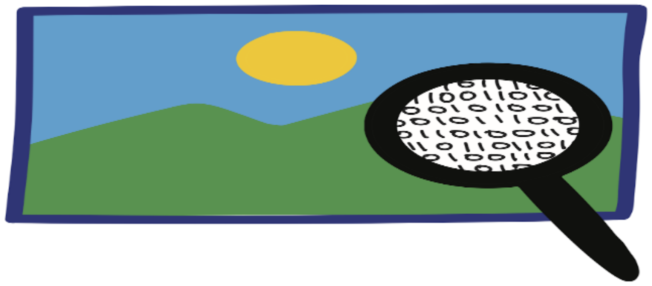







## 1. Information is represented in digital form.



## 1. Digital Representation

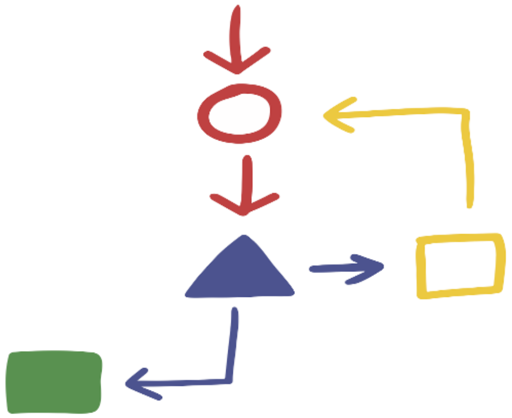
- Huge variety of information is stored and shared
- Simple
  - eg. number of steps on a fitness tracker
- or complex
  - details of every transaction through a big bank
- Includes text, images, video, sound and scientific readings.

All of this information is reduced to binary digits (bits) and makes digital devices useful.

## 1. Digital Representation

- Lead to versatile devices: same hardware (eg. smartphone) for different purposes:
  - music, photos, email, videos,
  - because all represented as bits,
  - easy to store, copy, transmit on same hardware.
- Non-digital (analog) devices are specialized
- Digital data can be shared without loss of quality
  - analog devices reduce quality at copy or re-transmit.

## 2. Algorithms interact with data to solve computational problems.



## 2. Algorithm

An algorithm is a well defined process that acts on data to solve some problem,

for example

- finding the shortest route on a map,
- matching two strands of DNA,
- changing the brightness of a photo.

## 2. Algorithm

- Can only include steps that a computer could do
- Full power of a digital device is realised by an algorithm using three structures for program flow:
  - sequencing
  - selection (based on some values)
  - iteration
- a computer can also
  - read in information (input)
  - give out information (output)
  - and store data to use later on

## 3. The performance of algorithms can be modelled and evaluated

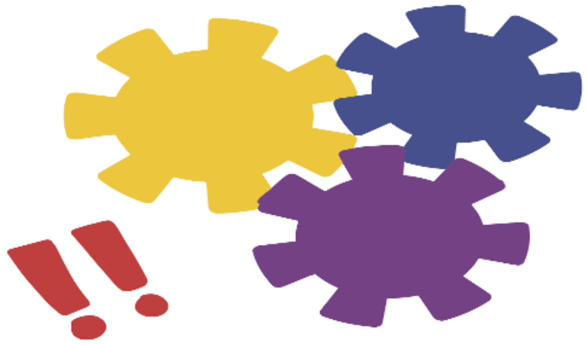
## 3. Performance

- Main resources used by algorithm: time and space (memory).
- Time is a key factor:
  - slow programs are annoying to users,
  - if a program takes decades, should know it in advance
  - An inefficient algorithm wastes power; have an environmental impact; battery may run out.
- Some algorithms need a lot of spare memory
  - May make the algorithm infeasible in some cases,
  - but may be tradeoff if the algorithm is faster.

## 3. Performance

- running time of an algorithm is usually estimated based on the size of the input
  - Eg. number of items being searched through, the number of streets in a map, the number of pixels in an image.
- The time may grow with size of the input (linear)
  - But may be better than that, and maybe worse.
- It is important to estimate the time before implementing the algorithm
  - May be very sensitive to the size of the input
  - a program may work well in tests, but take much longer with a larger input

## 4. Some computational problems cannot be solved by algorithms.



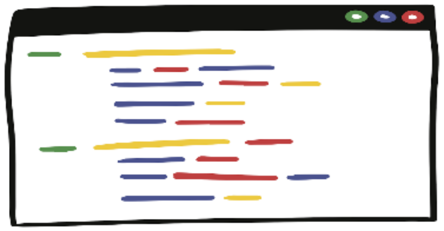
## 4. Unsolvable problems

- Some computational problems will never have programs written to solve them (these problems are not computable).
- For example, it can be proved that no one will ever be able to write a general app that can determine whether or not another app will freeze your smartphone (this is more formally known as the halting problem).

## 4. Unsolvable problems

- In addition, exist many practical problems for which all *known* algorithms for optimal solution are “intractable,”
  - no machine has the resources to execute the algorithm when size of input is fairly large.
- For these problems, need heuristics to find an approximate solution instead of optimal solutions
  - Some problems have proofs that they are intractable
  - but for many problems an algorithm that runs in a reasonable amount of time was not found, despite decades of research, yet we also have not proved that an algorithm cannot exist
- Resolving this issue is widely regarded as one of the biggest questions in computer science.

## 5. Programs express algorithms and data in a form that can be implemented on a computer



## 5. Programs

- Programming involves taking algorithms (which might exist only in the programmer's head, or may have been designed by a team of people) and turning them into program instructions that can be executed by a computer.
- Program instructions are written in a programming language which is precisely defined.
- These instructions manipulate data on the computer, so the form and meaning of the data is dictated by the program.

## 5. Programs

- To fully control a general purpose computer (most digital devices) need six properties:
  - 3 control structures (sequence, selection, iteration),
  - and 3 ways to deal with data (input, output and storage)
- These are also key elements of writing programs.
- So any programming language that has all of these can be used to write any computation that any other full programming language could be used for
- The differences between languages is mostly how well they suit a particular situation
  - e.g. for processing files, running on a smartphone, teaching programming, or running an enterprise system.

## 6. Digital systems are designed by humans to serve human needs.

## 6. Humans

- This is the driver for all the ideas above
- Digital devices must be fast, reliable and match a need appropriately if people are to use them.
- And because they are designed by people, the process for designing them needs to enable the developers to efficiently turn creative ideas into working products.

https://youtu.be/Uw0PISu2pog

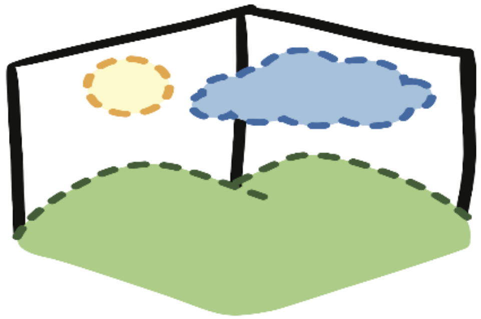
## 6. Humans

- There are 3 broad areas concerned with human factors:

1. Creating interfaces that are easy to use in the situation they are intended for
2. Developing software on a large scale, making sure the product meets user's needs, is reliable, does what intended, and completed in a timely fashion
3. The impact of technology on humans, the responsibilities of those who work on it, and even whether or not we should construct it.

- Require understanding human behavior (psychology), interaction (sociology), and capability (physiology).

## 7. Digital systems create virtual representations of natural and artificial phenomena.



## 7. Virtual Representations

- Computer simulations and virtual systems are used to create virtual versions of *processes* in the physical world, and also to create imagined scenarios.
- Simulations can be used to reduce cost
  - e.g. simulating a structure to fine tune it before building it, or simulating different financial scenarios to choose the most effective strategy
- Simulations can also be used to reduce risk
  - e.g. simulating dangerous situations to give aircraft pilots experience, or simulating spread of a disease to determine how best to prepare for an epidemic.

## 7. Virtual Representations

- Virtual *worlds* can be created by generating images and sounds artificially, to make the user feel that they are in a world imagined by the designer
  - Eg. computer games, virtual reality and augmented reality.
- Virtual *machines* provide a simulation of a computer, and this approach is widely used because it protects the physical hardware and the software from each other, which can provide a safer and more flexible environment.
- Artificial Intelligence models human intelligence, attempts to replicate human decision making and reasoning.

## 7. Virtual Representations

- These virtual representations take advantage of the unique properties of digital devices that enable a system to work on vast amounts of data
- if something goes wrong, they can be re-started in full working order by simply restoring some data, which is considerably simpler than reinstating physical objects that have been damaged or placing humans in a dangerous situation.

## 8. Protecting data and system resources is critical in digital systems.

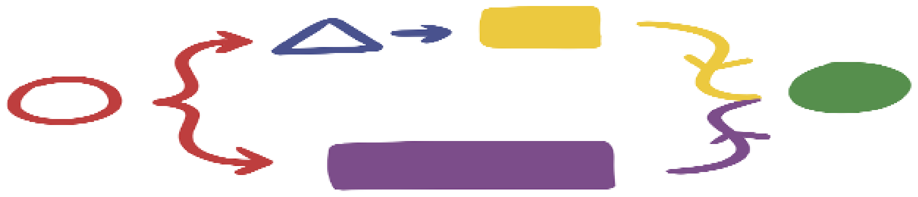
## 8. Protection

- Modern computing systems provide access to data and resources that if used inappropriately, could breach privacy, provide unauthorised access to financial data or other resources, or even bring about physical harm.
- Security professionals often say that the weakest link in the security of a computer system is the user
  - so it is essential that all computer users understand basic computer security principles.
- These principles include
  - confidentiality (prevent unauthorised access to information),
  - availability (legitimate users can access their information),
  - and integrity (the information is accurate).

## 8. Protection

- Computer users must be aware of and understand the tools and techniques that they can use to make their computing environment more secure.
- These tools include
  - encryption methods,
  - detecting and blocking attacks,
  - authenticating who is accessing a system,
  - and allowing users to recover from damage, whether malicious or accidental.

## 9. Time dependent operations in digital systems must be coordinated.



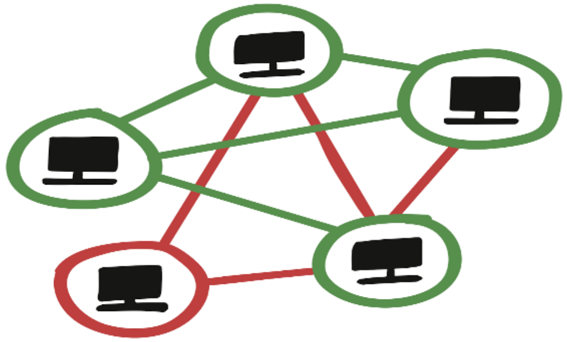
## 9. Coordination

- Digital systems have many components that can run independently
  - these components can be working in parallel, and on independent schedules.
- Parallelism occurs at many levels in digital systems:
  - instruction execution on a CPU,
  - multi-core systems in a laptop,
  - data being transmitted over a network through multiple routes,
  - large “big data” systems process huge data in small chunks and combine the results.

## 9. Coordination

- Some systems (eg. embedded systems, that obtain their input from multiple sensors, and produce output to multiple actuators), must cope with this inherent concurrency even if they have only one processing unit.
- When a computational task is spread over several independent parts of the system, need to make the most of the ability to spread the work over multiple devices.
  - Problems need to be broken up into as many parts as possible that can be processed independently and recombined,
  - The dependency between these operations can restrict how easily can break a problem into parts.

## 10. Digital systems communicate with each other using protocols.



# 10. Communication protocols

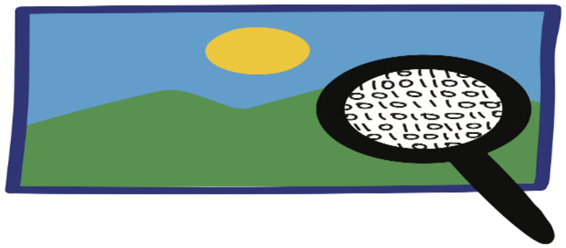
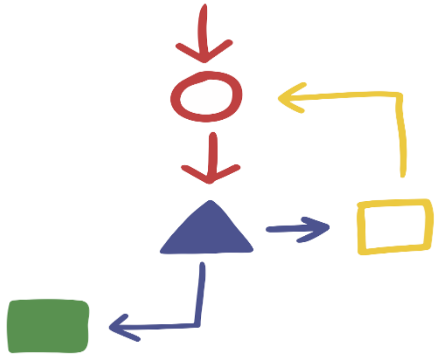
- Very few digital devices are an “island” - most are connected by networks.
- The goal is to get data through the networks as quickly as possible.
- Networks are prone to errors from faulty components or transmission interference, and are also vulnerable to attack from people wanting to eavesdrop on the data or prevent it getting through.

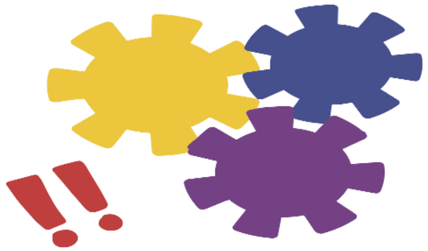
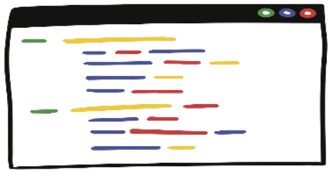
## 10. Communication protocols

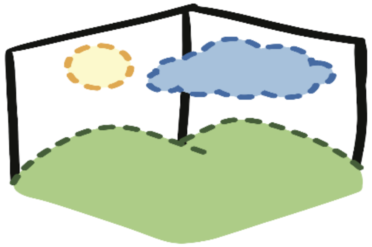
- Techniques are available to minimise these issues, to the extent that people use smartphones and the Internet for sending important and private information without being concerned about reliability and security.
- Protocols that ensure that the data has arrived safely and efficiently are essential for almost any situation: personal communications, commercial transactions, or military control all need to be sure that the data gets through reliably.

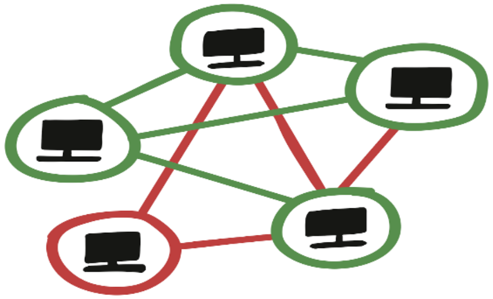
## Summary

- The big ideas presented here provide a framework to inform those who are teaching computer science curricula so that the wide range of individual topics can be seen as part of a bigger picture.
- Managed to converge on list that incorporates what many colleagues regard as the key ideas in CS.
- An online list with more detail may be found in
  - http://www.cosc.canterbury.ac.nz/research/RG/CSE/big-ideas/
  - http://www.canterbury.ac.nz/media/documents/oexp-engineering/BigIdeas-webdocument.pdf





## Evaluation

- As part of our research, we would like to evaluate our choice by surveying the opinions, mostley of people who have been teaching CS to K-12 pupils.
- In particular, we ask you to participate
- (Some of you already responded, thanks)
- The survey is anonymous