Our goal is to define an educational tool for teaching Computer Science to young children (8 to 12 years old) and students without a solid mathematical background.

**Characteristics**

- **No mathematical knowledge needed to understand and use it**
  
  in CS courses, computing machines are always introduced using mathematical definitions, even classical exercises are usually based on algorithms taken from number theory (eg factorial, Fibonacci); this do not scale to children. We propose to use pattern matching to express examples and exercises.

- **A physical, mechanical metaphor for reasoning about computation**
  
  so to enforce the idea that computation is a property shared by many different system, some of which made up with a few simple pieces. We think it is possible to hide formal semantics inside real world objects, and animate their behavior by game-like simulations.

- **Direct manipulation and learning-by-doing**
  
  with our tool students can define their own computational elements, extending the basic set; it should be easy to explore problem-solving and design methodologies (top-down and bottom-up project development).

- **Cheap support (paper), wide availability (web-based simulator), individual learning.**
  
  The tool is both web and paper-based, so every school can print the basic kit; so even when it is not possible to have a PC for each student, our tool can be used on individual basis.

Andrea Valente

av@cs.aue.auc.dk
In the basic c-cards set, there are seven card types:

- the **marker-source** card and the **marker-pit** card (first row);
- the **cross** card, the **east** and **west turn** cards and the **confluence** card (second row)
- the **switch** card (last row in the figure) is the only one double-faced: after cutting it, it will be folded along the thick vertical line.

Together with the seven card types, there are **markers** (the small rectangle at the far right on the first row).
Cards can be connected to build circuits.
A card is mainly a set of ports (plus some arrows connecting them), so we can consider a card circuit as a graph of ports, each one possibly containing markers.

Graph-rewriting rules are simple to explain to young students and support our visual and physical metaphor for computation. **The figure shows the rules for source and pit cards** (first row), **cross and switch cards** (second and third row respectively).

All rules work up-to card rotations and are deterministic: they are triggered by the presence of markers, and depend upon the current state of cards.

Rules for turn and confluence cards are omitted.

**Andrea Valente**

**av@cs.aue.auc.dk**
Editor and simulator

C-cards can be printed and cutted. But we also provide an editor and simulator called E-SI, implemented in dynamic html and JavaScript (can be used via Internet, on every platform, or offline).

The image shows the editing of a card circuit.

It is possible to save up to two circuits (using cookies).
Editor and simulator

Here we see the animation of the execution.

A marker is traversing the card-circuit, from top to bottom of the image.
What kind of exercises?

To show the behavior of c-cards to a class, we can start by composing the circuit in the figure; after building it, we can animate it, placing two markers on the \textit{In1}-labeled source card, and one on the other, labeled \textit{In2}.

After a number of steps, the two markers will arrive at the \textit{Out2} pit, while the other one will stop at the \textit{Out1} pit.

The action performed by this circuit is to \textit{swap} its inputs: if markers starting from the \textit{In1} source card are in state TRUE, and markers starting from \textit{In2} are FALSE markers, then the circuit is behaving as a \textbf{boolean not-gate}, swapping TRUE and FALSE values.

In the figure all unreachable ports are grayed, to simplify spotting the two main paths markers can follow in the circuit.
An example of exercise for the student

Let’s consider the shape in the figure, where the *In* source card is defined by the pattern *(white.black)*

a white marker, then a black and so on. The desired circuit have to separate white markers from black ones.

Note for teachers

If we label markers with a progressive counter, as they exit from the source card, here white markers will be odds, and the black ones even: a circuit implementing these specifications correctly must make white (odd) markers stop at the *Out1* pit, and black (even) ones at the *Out2* pit card. *The implementation is given in the right side of the figure.*

This means we are counting module 2, but there is no need to describe the exercise in this *numerical* way!

Andrea Valente

av@cs.aue.auc.dk
Abstraction and reuse
macro-cards

A **modular** exercise

Suppose now we want to make a more complex exercise, working with a source defined by

\[(\text{white.slashed.black})\]

The resulting circuit will have three pits, in which makers of the same color will accumulate: the first (western) pit will receive all white markers, the second all the slashed ones and the black markers will stop at the third.

**The (white.black) circuit could be a starting point for the definition of this one!**

**Macros**

A standard feature of programming languages is the possibility of defining macro-instruction, and use them to write more readable and compact programs.

**The corresponding concept in c-cards is the one of macro-card.**

Once a circuit is defined and we can decide to use it repetitively to build more complex circuits. Therefore we can map a whole circuit it into a single *large* card, preserving the overall behavior, or even define a new basic card type, implementing that behavior.

**Andrea Valente**

**av@cs.aue.auc.dk**
A *random* extension frequencies and probability

An extension to the basic deck is the *random* card: a marker entering this card will be sent out from the left or the right, with \{1/2,1/2\} probability distribution. *This is the non-deterministic version of the switch card.*

Using two or more of these cards *in cascade*, we can have a marker-source implementing almost every probability distribution needed.

**Exercise with the random card**

Build a circuit that divides white markers from black ones, with 50% probability of making an error.

Is it possible to have only 25% probability of making an error?

Andrea Valente

avn@cs.au.dk
A random extension
noisy channels and transmission theory

Using the random card it is possible to discuss of data transmission, errors and related problems.

If the channel is implemented like below, it will transmit without errors.

If the channel is implemented using random cards, it can generate transmission errors: in this case there is 1/2 probability of swapping the transmitted symbol.

Andrea Valente

av@cs.aue.auc.dk
Experiments and collaborations

We are currently looking for collaborations to set up experiments with c-cards. Such experiments should be conducted with high-school students with humanistic background, or with young students from 8 to 12.

Work in progress

Some extensions are being investigated, to enhance the expressive power of the basic deck: we have defined a new card that can provide Turing-completeness to c-cards (we aim at prove this and then study new kind of exercises based on the extended deck).

Related work and bibliography


Andrea Valente av@cs.aue.auc.dk