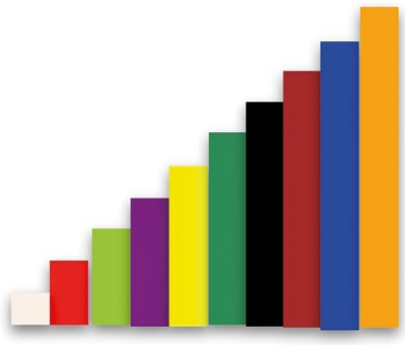
USING HASKELL WITH 5- TO 7-YEAR-OLDS

Ian Benson and Jenny Cane show how Haskell helps learners to reason symbolically with Cuisenaire-Gattegno Mathematics: a proven approach to teaching algebra before arithmetic

STORY BY Ian Benson and Jenny Cane

At Bursted Wood Primary School, children in key stage 1 had already embarked on a secure mathematical journey, putting algebra at the heart of the curriculum. The demands of the 2014 National Curriculum meant that in Year 1 children were expected to be taught to use all four arithmetic operations, and fractions, as operators. As a result, Bursted Wood adopted a series of textbooks written by Caleb Gattegno called *Numbers in Colours*. These textbooks use Cuisenaire rods to encourage the learner to explore patterns, and then to give letter code names to the patterns, and record the relationships between them with equations. It is only after the letter code exploration that they finally use a rod as a unit of measure (e.g. white representing one), and transform their equations into arithmetic. Very early on in Year 1, children are able to read and write a variety of complex algebraic equations using plus, minus, multiply and fractions as operators, and translate these into arithmetic. The results had been impressive throughout key stage 1. In Year 2 we wanted to take it a step further by running computing lessons using the same use of natural language and level of sophistication.



■ The ten Cuisenaire rods arranged in order as a staircase with step size White.

Why Haskell?

Haskell is closely aligned to conceptual mathematics and number theory, the inspiration for Gattegno Mathematics. It has four attributes that make it particularly appropriate for mathematics education:

- Its syntax is very close to traditional mathematical notation (but beware it can be too terse).
- Variables are statically typed, which means that programmers are protected by the compiler from passing invalid information as an argument (input) or output value.
- The Haskell interpreter evaluates functions 'lazily', only when required. This makes it possible to program infinite data structures in finite memory.
- Information is passed from the arguments to functions by pattern matching. This is an intuitive way for programmers to envisage and reason about the behaviour of their programs.

19. Are the following true

$o=10w=5\ r=2y$
$B=9w=3g$
$t=4\ r=2p=8w$
$b=7w$
$d=6w=3r=2g$
$y=5w$

$p=4w=2r$
$g=3w$
$r=2w$

■ Equivalent expressions using Gattegno's rod codes to represent the relationship between a rod and equivalent trains of one colour

Starting with a staircase

To start, we went back to a familiar pattern that Gattegno introduces in the textbooks, called a staircase. This is built from one rod of each colour arranged in size order (figure 1). The children were able to physically build the staircase. However, because they were so familiar with the pattern, they could also access it virtually, which would help with debugging and logical reasoning later.

A role-play game to establish computational thinking was introduced before any actual programming began. This allowed the children to familiarise themselves with the function calls `successor` and `predecessor` (`succ`, `pred`) while traversing the staircase. The children would take turns to ask each other questions such as `succ White` (answer Red) or `pred Orange` (answer Blue). To help the children access the new programming vocabulary, one child

```
data Colour = NaC | Red | Purple| DarkGreen |Brown |
Orange deriving (Show, Ord, Eq, Enum)


w=2
g = 3*w
p = 2*r
r=2*w
y = 5*w
d = 6*w
k = 7*w
t = 8*w
n =9*w
o = 10*w
```

■ A rod arithmetic calculator built after studying the staircase step size White, and its conjugate with every other rod removed

would take the role of the 'engine' and the other 'interpreter'. This was extended to `succ $ succ` (successor of successor) which uses function composition `$`. Some children could creatively enjoy inventing functions such as `succ $ succ $ succ White` to test the technique. The children experimented by writing down their calls before building themselves a rod calculator in Haskell. The language soon became a natural part of lessons and meant that it was easy to progress to the next step.

run, and that this would require carefully debugging and retyping. The children could spot bugs in the engine because they had already worked out the answer. The staircase was a familiar pattern, so they could predict what the output should be.

Children were also able to edit the `Colour` data type to model odd or even staircases and test their functions to see what the effect would be on the derived `successor` function. A transcript (see insert) shows a pupil working with the

> SOME CHILDREN COULD CREATIVELY ENJOY INVENTING FUNCTIONS

Programming

Very soon the children were able to construct their own rod calculator in Haskell, by first constructing the engine using the `Colour` data type. Then the children asked the same role-play questions, this time by typing the language into the interpreter window. It soon became apparent that errors in the engine meant that the interpreter would not

number names for the rods, measured with White rather than the colour names. He becomes aware as he is speaking that the version of the `succ` function that works for a staircase with a step size White is distinct from the version of `succ` in an even staircase, and he corrects himself. It is clear to see that he has constructed an image of the staircase and is virtually traversing it.

■ Transcript of a conversation about program behaviour

```
13:08 Q: So what are you doing? Are you doing |succ| or the other one?
19:08 A: I am doing |succ|
21:11 Q: What does that mean?
24:03 A: It means that if I do |succ|, you add one number on ... If I had six
and I did |succ| it equals (pause for reflection) 7
37:00 Q: You're right. So you do it ....
40:07 A: (speaks over Q) Actually 8 because there are no odd.
46:03 Q: Ah. You are just doing the evens.
48:06 Q: Right, Brilliant
```

WHERE TO GO NEXT

Many Cuisenaire and Gattegno resources can be found online:

- New developments in arithmetic teaching in Britain: introducing the concept of 'set' **helloworld.cc/2pYFgOw**
- The common sense of teaching mathematics: **helloworld.cc/2p9dyKO**
- Gattegno Mathematics, Book 1: **helloworld.cc/2qz7xZe**
- Virtual Rods: **helloworld.cc/2pI8ctc**
- Sample code for these exercises: **helloworld.cc/2pFyx9j**

Calculating

To build the rod arithmetic calculator, the pupil reuses the equations that they studied in Year 1. Gattegno's definitions lie in the algebraic domain. Programming illustrates the distinction between algebra and arithmetic. Making this clear is crucial to investigating the level of abstraction that the learner has reached. The figure shows a rod arithmetic calculator constructed after four terms of Gattegno Mathematics. The values associated with the rod code letters are modelled as constant functions in Haskell. Nine equations are derived from Gattegno's equivalent expressions. The tenth, which binds an integer value to the code for White, shows the choice of measuring unit, which fixes the values of the other codes.

By introducing Haskell to the children we have been able to take the mathematics that the children are familiar with and successfully and creatively introduce programming functions. We have been able to provide both the teacher and child with a rich situation, allowing them to think both mathematically and computationally. It is exciting to think where our investigations may lead us to next! (HW)