

# Towards "middle school MPC"

J.N. Oliveira

(jno@di.uminho.pt)  
Dept. Informática,  
Universidade do Minho  
Braga, Portugal

IFIP WG2.1 meeting #64  
March-April 2009  
Weltenburg, Germany

# About the MathIS project

Name **MathIS** - *Reinvigorating Mathematics for the Information Society*

Project leader L.S. Barbosa (Formal Methods, Minho)

Requested funding 127K euros, FCT (Portuguese NSF) grant Nr. PTDC/EIA/73252/2006

Time scale 3 years, officially started Jan. 2009.

External consultants Roland Backhouse, Richard Bird, Raymond Boute, Tom Verhoeff.

## How to spell "MathIS"

Quoting Jeff Kramer [1]:

*Why is it that some software engineers and computer scientists are able to produce clear, **elegant** designs and programs, while others cannot? Is it possible to improve these skills through **education** and training? Critical to these questions is the notion of **abstraction**.*

## How to spell "MathIS"

Still Jeff Kramer [1]:

**Abstraction** is widely used in other disciplines such as **art** and **music**. For instance (...) Henri **Matisse** manages to clearly represent the **essence** of his subject, a naked woman, using only simple lines or cutouts. His representation **removes** all detail yet **conveys** much.



## From the project's abstract

- Modern **IT**-driven societies demand highly skilled professionals [able] to resort to **mathematical language** and method to build models of problems and situations and reasoning effectively about them.
- Such an ability is at the heart of what it means "**to understand**" and it may be considered a fundamental ingredient of democratic **citizenship**.
- There is little hope, however, that such expectations and demands be met by current standards in school maths education. [...] Worst of all, **mathphobia** (which seems to be spreading everywhere) has become a hot spot for the media.
- This situation calls for **emergency policies** capable of reinvigorating maths education and its effective application at all problem-solving levels.

## Scientific? Pre-scientific?

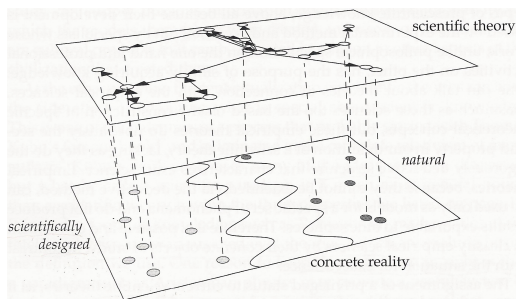
In an excellent book on the history of scientific technology ( *“How Science Was Born in 300BC and Why It Had to Be Reborn*), Lucio Russo [2] writes:

*The immense usefulness of **exact** science consists in providing **models** of the real world within which there is a guaranteed method for telling false statements from true. (...) Such models, of course, allow one to describe and **predict** natural phenomena, by translating them to the theoretical level via **correspondence rules**, then solving the “**exercises**” thus obtained and translating the solutions obtained back to the real world.*

Disciplines unable to build themselves around “*exercises*” are regarded as **pre-scientific**.

$$e = m + c$$

Also quoted from Russo's book :



Vertical lines mean **abstraction**, horizontal ones mean **calculation**:

*engineering = model first, then calculate ( $e = m + c$ )*

Emphasis on calculation — as acknowledged by many scientists and philosophers in the past:

## Ut faciant opus signa

[Symbolisms] *“have invariably been introduced to make things easy.*

*[...] by the aid of symbolism, we can make transitions in reasoning almost mechanically by the eye, which otherwise would call into play the higher faculties of the brain.*

*[...] **Civilisation** advances by extending the number of important operations which can be performed **without thinking** about them.”*

*(Alfred Whitehead, 1911)*



## Ut faciant opus signa

*“Certaines personnes ont [l'affectation] d'éviter en apparence toute espèce de **calcul**, en traduisant par des phrases fort longues ce qui s'exprime très brièvement par **l'algèbre**, et ajoutant ainsi à la longueur des opérations, les longueurs d'un langage qui n'est pas fait pour les exprimer.*

*Ces personnes-là sont en arrière de cent ans.”*

*(Evariste Galois, 1831)*

# Ut faciant opus signa

*(...) De manera, que  
quien sabe por Algebra,  
sabe científicamente.*

*... in this way, who knows by Algebra knows  
**scientifically***

(Pedro Nunes, 1567)

## Ut faciant opus signa

*I feel that controversies can never be finished . . . unless we give up complicated reasonings in favour of simple calculations, words of vague and uncertain meaning in favour of fixed **symbols** . . . every argument is nothing but an error of calculation. [With symbols] when controversies arise, there will be no more necessity for disputation between two philosophers than between two accountants. Nothing will be needed but that they should take pen and paper, sit down with their calculators, and say '**Let us calculate**'.*

Gottfried Wilhelm Leibniz (1646-1716)

## What should we do?

- Despite all this good advice, **calculational** techniques and **constructive** proofs are not taught at school, in general.
- What's the strategy to follow to "**re-factor**" the whole thing from beginning to end?
- **When** should it start? higher education is **too late!**
- **Why not** from the actual beginning of **symbolism** in middle-school maths textbooks?
- This will lead us to the  $\pm 7$ th year of maths education (12-13 year old kids).

## Precautions

In dealing with middle school teaching we've been warned to bear in mind the following guiding principles:

- Don't disturb the normal flow of things.
- (Re)use existing problems in the textbooks to "smuggle" the good thinking habits.
- Be careful not to generate "antibodies".
- Teachers likely to be less receptive than the students themselves.

My personal feeling is that the earlier in the syllabus the more "MPC" should mean: *programs (and proofs) are already (hidden) in your maths textbook — shall we calculate them?*

## Difficulties

I've started looking into maths textbooks as preparation for the work of a research assistant to join the project soon. Enough to realize that:

- **Logic** reasoning is missing as a proper topic (though it can be found in the philosophy course, 10-11th year).
- **Proofs** virtually absent from middle school curricula.
- Geometry still an exception.

Challenge:

- Train students to do (alternative) **constructive** proofs (eg. in geometry)

What do we mean by “*constructive*” in this setting ? Let us see an example.

# From a textbook (vector calculus)

## The problem

### Uma demonstração

$[OACB]$  é um paralelogramo.

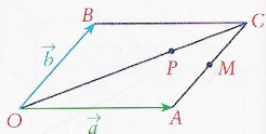
$$\overrightarrow{OA} = \vec{a} \text{ e } \overrightarrow{OB} = \vec{b}$$

$$\overrightarrow{OP} = \frac{2}{3}\overrightarrow{OC}$$

$M$  é o ponto médio de  $[AC]$ .

Prove que  $B$ ,  $P$  e  $M$  pertencem à mesma recta.

Procure acompanhar os passos seguidos na seguinte demonstração.



## From a textbook (vector calculus)

The given proof:

### Demonstração:

Tem-se sucessivamente:

$$\overrightarrow{OC} = \overrightarrow{OA} + \overrightarrow{AC} = \vec{a} + \vec{b}$$

$$\overrightarrow{OP} = \frac{2}{3}\overrightarrow{OC} = \frac{2}{3}(\vec{a} + \vec{b})$$

$$\overrightarrow{OM} = \vec{a} + \frac{1}{2}\vec{b}$$

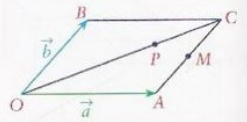
$$\overrightarrow{BP} = \overrightarrow{BO} + \overrightarrow{OP} = -\vec{b} + \frac{2}{3}(\vec{a} + \vec{b}) = \frac{2}{3}\vec{a} - \frac{1}{3}\vec{b}$$

$$\overrightarrow{BM} = \overrightarrow{BO} + \overrightarrow{OM} = -\vec{b} + \vec{a} + \frac{1}{2}\vec{b} = \vec{a} - \frac{1}{2}\vec{b} = \frac{3}{2}\left(\frac{2}{3}\vec{a} - \frac{1}{3}\vec{b}\right)$$

Portanto,  $\overrightarrow{BM}$  é colinear com  $\overrightarrow{BP}$  porque  $\overrightarrow{BM} = \frac{3}{2}\overrightarrow{BP}$ .

Como  $[BP]$  e  $[BM]$  são paralelos e têm em comum o ponto  $B$ , os pontos  $B$ ,  $P$  e  $M$  pertencem à mesma recta.

É capaz de reproduzir esta demonstração?





## From a textbook (vector calculus)

Comments:

- **verification**, not calculation
- not **goal** oriented
- not “constructive” enough
- tricky? cf. final question “*Are you able to replay this proof?*” ...

In a constructive proof, the **starting** point will be the **goal** itself:

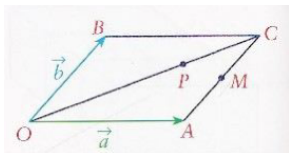
**Goal:**  $B$ ,  $P$  and  $M$  on the same line

equivalent to

$$\vec{BP} = k\vec{BM} \text{ for some } k$$

Proof amounts to calculating such  $k$ , if any

# “Constructive proof” (modeling)



$$\vec{BP} = k\vec{BM}$$

$$\Leftrightarrow \{ \text{vector sums} \}$$

$$\vec{BO} + \vec{OP} = k(\vec{BO} + \vec{OA} + \vec{AM})$$

$$\Leftrightarrow \{ \text{let } \vec{OP} = k_1\vec{OC}, \vec{AM} = k_2\vec{AC} \}$$

$$\vec{BO} + k_1\vec{OC} = k(\vec{BO} + \vec{OA} + k_2\vec{AC})$$

$$\Leftrightarrow \{ \text{let } \vec{OA} = \vec{a}, \vec{OB} = \vec{b} \}$$

$$-\vec{b} + k_1(\vec{a} + \vec{b}) = k(-\vec{b} + \vec{a} + k_2\vec{b})$$

## “Constructive proof” (calculation)

$$\begin{aligned}
 & -\vec{b} + k_1(\vec{a} + \vec{b}) = k(-\vec{b} + \vec{a} + k_2\vec{b}) \\
 \Leftrightarrow & \quad \{ \text{“al-muqâbala” (1,2,3)} \} \\
 & k_1\vec{a} + (k_1 - 1)\vec{b} = k\vec{a} + k(k_2 - 1)\vec{b} \\
 \Leftrightarrow & \quad \{ \text{equality rule (4)} \} \\
 & k_1 = k \wedge k_1 - 1 = k(k_2 - 1) \\
 \Leftrightarrow & \quad \{ \text{“calculus of al-gabr and al-muqâbala”} \} \\
 & k = k_1 = \frac{1}{2 - k_2}
 \end{aligned}$$

Given problem corresponds to  $k_2 = \frac{1}{2}$  and  $k_1 = \frac{2}{3}$ .

Other cases:  $k_2 = k_1 = 1$  ( $P = M = C$ ),  $k_2 = 0 \wedge k_1 = \frac{1}{2}$  ( $M = A$ ), etc

# Thanks to

Vector calculus:

$$k(\vec{a} + \vec{b}) = k\vec{a} + k\vec{b} \quad (1)$$

$$(k + j)\vec{a} = k\vec{a} + j\vec{a} \quad (2)$$

$$k(j\vec{a}) = (kj)\vec{a} \quad (3)$$

and, for non co-linear  $\vec{a}, \vec{b}$ , the equality rule:

$$k\vec{a} + j\vec{b} = m\vec{a} + n\vec{b} \Leftrightarrow k = m \wedge j = n \quad (4)$$

## Summary

Constructive proof means **calculating** a (often necessary and) sufficient condition for the **goal** to hold.


# From a 7th year maths textbook

- Let's now go to the point where symbolic notation turns up for the first time.
- First page of chapter on multiplying and dividing rational numbers:

**Vais aprender**

- Calcular o produto de dois números racionais relativos.
- Calcular o quociente de dois números racionais relativos.
- Calcular o valor numérico de expressões envolvendo números racionais relativos.
- Justificar que o produto de dois números negativos é um número positivo.

## 4. Multiplicação e divisão de números racionais



Na Matemática como no desporto é fundamental compreender e cumprir as regras.

### 4.1. Multiplicação de números racionais relativos

A multiplicação de números racionais positivos já é tua conhecida, bem como as suas propriedades.

**Propriedade comutativa**

$$\frac{1}{2} \times \frac{3}{5} = \frac{3}{5} \times \frac{1}{2} = \frac{3}{10}$$

**Propriedade associativa**

$$\left(\frac{1}{2} \times 3\right) \times \frac{5}{7} = \frac{1}{2} \times \left(3 \times \frac{5}{7}\right)$$

$$\frac{3}{2} \times \frac{5}{7} = \frac{1}{2} \times \frac{15}{7}$$

$$\frac{15}{14} = \frac{15}{14}$$

**Propriedade distributiva da multiplicação relativamente à:**

**• adição**

$$3 \times \left(\frac{1}{5} + 5\right) = 3 \times \frac{1}{5} + 3 \times 5$$

$$= \frac{3}{5} + 15 =$$

$$= \frac{3}{5} + \frac{75}{5} = \frac{78}{5}$$

**• subtração**

$$3 \times \left(\frac{1}{5} - 5\right) = 3 \times \frac{1}{5} - 3 \times 5$$

$$= \frac{3}{5} - 15 =$$

$$= \frac{3}{5} - \frac{75}{5} = -\frac{72}{5}$$

**O elemento absorvente da multiplicação é (0) zero.**

$$0 \times \frac{1}{3} = \frac{1}{3} \times 0 = 0$$

**OBSERVAÇÃO**

**Propriedades da multiplicação**

- $a \times b = b \times a$
- $a \times (b \times c) = (a \times b) \times c$
- $a \times (b + c) = a \times b + a \times c$
- $a \times (b - c) = a \times b - a \times c$
- $a \times 1 = 1 \times a = a$
- $a \times 0 = 0 \times a = 0$

116

## From a 7th year maths textbook

Draw your attention to the text-box at the bottom, on the left:

The image shows a page from a math textbook. On the left, a vertical orange bar contains the page number '156' in a red box. The main content is a white box with an orange header 'OBSERVAÇÃO'. Below the header, the title 'Propriedades da multiplicação' is followed by a list of six properties of multiplication. To the right of this box, there is a light blue box containing the text '• subtração' and '0 elemento absorv'.

**156**

**OBSERVAÇÃO**

**Propriedades da multiplicação**

- $a \times b = b \times a$
- $a \times (b \times c) = (a \times b) \times c$
- $a \times (b + c) = a \times b + a \times c$
- $a \times (b - c) = a \times b - a \times c$
- $a \times 1 = 1 \times a = a$
- $a \times 0 = 0 \times a = 0$

• subtração

**0 elemento absorv**

## First program: multiplication

The **properties** of multiplication are enough for letting students to derive programs involving addition and multiplication, eg.

$$a \times 0 = 0 \quad (5)$$

$$a \times 1 = a \quad (6)$$

$$a \times (b + c) = (a \times b) + (a \times c) \quad (7)$$

For  $c = 1$  one has

$$a \times 0 = 0$$

$$a \times 1 = a$$

$$a \times (b + 1) = (a \times b) + (a \times 1)$$

(Haskell 1st year students do this all the time without noticing)

## First program: multiplication

To make sure it's the student program which is will be running and not the native operation ( $*$ ), invite her/him to prefix " $*$ " by a "."

$$a .* 0 = 0$$

$$a .* (b+1) = (a .* b) + a$$

while doing obvious simplifications ( $a \times 1$  replaced by  $a$  according to the second clause, which can then be omitted.)

### Main learning outcome

Students see programs **arising** from their own maths books, not **invented** or coming out of the blue.



## Repeating the experiment

Later in the book, the following properties of exponentials turn up:

$$a^0 = 1$$

$$a^1 = a$$

$$a^{b+c} = a^b \times a^c$$

Students will easily repeat the previous experiment, thus writing an Haskell program which computes  $a^x$  — where they can re-use their own `(.*)`.

**NB:** pattern scales up to lists, cf. for instance

$$\text{mul } [] = 1$$

$$\text{mul } [a] = a$$

$$\text{mul } (b ++ c) = (\text{mul } b) \times (\text{mul } c)$$

## For loops — a natural evolution

- Programs like  $(a^*)$  provide a nice way to introduce students to the derivation (later in the course) of **for-loops**.
- Why? Folds over natural numbers “are” for-loops. Think eg. of

$$n = succ^n 0$$

— you calculate  $n$  by looping over  $succ$  with initial value  $0$ .

- In general,  $(foldN f i)n = f^n i$ . In other words,  $foldN$  “is” the **for-loop** combinator:

```
for b i 0 = i
for b i (n+1) = b (for b i n)
```

(This is quite often ignored in the teaching.)

## For-loops for free

In other words, students' calculations above already deploy a CbC (correct by construction) "for-loop" implementation of multiplication:

```
a .* n = for (a+) 0 n
```

something to be encoded (much later!) imperatively, eg. in C:

```
int mul(int a, int n)
{
  int s=0; int i;
  for (i=1;i<n+1;i++) {s += a;}
  return s;
};
```

## Not so immediate for-loops

Now consider the challenge of encoding the square function,  $sq\ n = n^2$ . Following the same approach, let students first recall known facts about squares, including Newton's binomial formula:

$$\begin{aligned} 0^2 &= 0 \\ 1^2 &= 1 \\ (a + b)^2 &= a^2 + 2ab + b^2 \end{aligned}$$

Playing the same game, the following will be obtained:

$$\begin{aligned} sq\ 0 &= 0 \\ sq\ (n + 1) &= sq\ n + \underbrace{2n + 1}_{\text{odd } n} \end{aligned}$$

**By the way:** students aware that  $n^2$  is the sum of the first  $n$  odd numbers.

## Not so immediate for-loops

- However, *sq* is not a for-loop because each additive contribution *odd*  $n = 2n + 1$  is dependent on  $n$ .
- What about *odd* itself? Ask the students to try and exploit “its maths”,

$$\text{odd } 0 = 1$$

$$\text{odd}(n + 1) = 2 + \text{odd } n$$

which lead immediately to for-loop *for* (2+) 1.

- Still, students don't know what to do with *sq*. What can we do about this?

## Two-variable for-loops

By putting *sq* and *odd* side by side,

$$\text{sq } 0 = 0$$

$$\text{sq } (n + 1) = \text{sq } n + \text{odd } n$$

$$\text{odd } 0 = 1$$

$$\text{odd } (n+1) = 2 + \text{odd } n$$

observe that both functions share the same input pattern and can thus run “together”, co-operating with each other. Thus proceed to **tupling**,

$$(\text{sq}, \text{odd})x = (\text{sq } x, \text{odd } x)$$

only to exploit “the maths” of this pair of functions:

$$(\text{sq}, \text{odd}) 0 = (0, 1)$$

$$(\text{sq}, \text{odd}) (i+1) = \text{let } (s, o) = (\text{sq}, \text{odd}) i \text{ in } (s+o, 2+o)$$

Clearly, this is for-loop  $\text{for}(0, 1)((s, o) \mapsto (s + o, 2 + o))$  which computes  $i^2$  on variable  $s$  and  $\text{odd } i$  on variable  $o$ . Thus the code which follows:

## Two-variable for-loops

C code for *sq* (and *odd*, implicitly):

```
int sq(int n)
{
  int s=0; int i; int o=1;
  for (i=1;i<n+1;i++) {s+=o; o+=2;}
  return s;
};
```

### Learning outcome

The number of **variables** required by a for-loop implementation of a given function over the natural numbers is the number of **mutually recursive functions** which such given function “unfolds” into once “their maths” are inspected.

## Three-variable for-loops

Later on, arithmetic/geometric **progressions** turn up. For instance, consider the computation of the number of squares one can draw on a  $n \times n$ -tiled wall, given by

$$ns \ n \stackrel{\text{def}}{=} \sum_{i=1}^n i^2$$

Besides  $i^2$ , it involves a summation. Once  $i^2$  is already done, students should check “the maths” of what’s new — summation:

$$\underbrace{\sum_{i=1}^0 i^2}_{ns \ 0} = 0 \quad \text{and} \quad \underbrace{\sum_{i=1}^{n+1} i^2}_{ns(n+1)} = \underbrace{\sum_{i=1}^n i^2}_{ns \ n} + (n+1)^2$$



## Three-variable for-loops

Proceeding as in the previous example will lead to

$$ns\ 0 = 0$$

$$ns(n+1) = ns\ n + \underbrace{(n+1)^2}_{sq1\ n}$$

$$sq1\ 0 = 1$$

$$sq1(n+1) = sq1\ n + \underbrace{2n+3}_{lin\ n}$$

$$lin\ 0 = 3$$

$$lin(n+1) = 2 + lin\ n$$

Let them check the  $(n+1)$  steps:

$$((n+1)+1)^2 = (n+1)^2 + 2(n+1) + 1$$

$$2(n+1) + 3 = (2n+3) + 2$$

## Three-variable for-loops

Thus the triple function which follows,

```
(ns,sq1,lin) 0 = (0,1,3)
(ns,sq1,lin)(i+1) =
    let (n,s,l) = (ns,sq1,lin) i
    in (n+s, s+1, l+2)
```

leading straight into (if required) the following code, in C:

```
int ns(int x)
{
int n=0;int s=1;int l=3;int i;
for (i=1;i<x+1;i++) {n+=s;s+=1;l+=2;}
return n;
};
```

## Further up the ladder

Eventually, **Taylor series** and the like will be part of our fictional students maths education. They will then be able to derive the following code

```
float exp(float x, int n)
{
    float h=x; float e=1; int s=2; int i;
    for (i=0;i<n+1;i++) {e=e+h;h=(x/s)*h;s++;}
    return e;
};
```

from the  $n$ -term approximation to Taylor (in fact: Maclaurin) series for the exponential function

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} \quad (8)$$

## Not so immediate calculations

Let us finally consider the challenge of calculating **whole division**, that is, the operation  $\div$  such that eg.

$$\begin{array}{r|l} 7 & 2 \\ 1 & 3 \end{array} \quad 2 \times 3 + 1 = 7 \quad , \text{ "ie."} \quad 7 \div 2 = 3$$

that is,

$$\begin{array}{r|l} n & d \\ r & q \end{array} \quad q = n \div d \Leftrightarrow d \times q + r = n \wedge$$

$q$  is largest ( $r$  smallest)

Is this a “good” specification of  $n \div d$ ?

## “Al-gabr rules

- Perhaps not: it involves a supremum, which is hard to reason about. A more subtle and more interesting specification is that of the following “al-gabr” rule,

$$q \times d \leq n \Leftrightarrow q \leq n \div d \quad (d > 0) \quad (9)$$

where universal quantification over the natural numbers is left implicit.

- Note how the  $\Leftrightarrow$  side of (9) ensures  $n \div d$  as the largest  $d$  such that  $q \times d \leq n$ .
- Rule (9), however, calls for an effective way of calculating with binary **relation**  $n \leq m$ . Can this replace equality, as in the previous CbC examples?

## Indirect equality principle

Yes, in an indirect way:

$$n = m \quad \Leftrightarrow \quad \langle \forall x :: x \leq n \Leftrightarrow x \leq m \rangle \quad (10)$$

This resembles a similar device in set theory to define set equality,

$$A = B \quad \Leftrightarrow \quad \langle \forall x :: x \in A \Leftrightarrow x \in B \rangle$$

of which another variant is:

$$A = B \quad \Leftrightarrow \quad \langle \forall X :: X \subseteq A \Leftrightarrow X \subseteq B \rangle$$

Let us see how indirection (10) leads to the algorithm:

# CbC using “al-gabr” rules and indirect equality

$$q \leq n \div d$$

$$\Leftrightarrow \quad \{ \text{rule (9) assuming } d > 0 \}$$

$$q \times d \leq n$$

$$\Leftrightarrow \quad \{ \text{cancellation} \}$$

$$q \times d - d \leq n - d$$

$$\Leftrightarrow \quad \{ \text{distribution law} \}$$

$$(q - 1) \times d \leq n - d$$

$$\Leftrightarrow \quad \{ (9) \text{ again, assuming } n \geq d \}$$

$$q - 1 \leq (n - d) \div d$$

$$\Leftrightarrow \quad \{ \text{trading } -1 \text{ to the right} \}$$

$$q \leq (n - d) \div d + 1$$

## CbC using “al-gabr” rules and indirect equality

That is, every natural number  $q$  which is at most  $n \div d$  (for  $n \geq d$ ) is also at most  $(n - d) \div d + 1$  and vice versa. We conclude that the two expressions are the same

$$n \div d = (n - d) \div d + 1 \quad (11)$$

for  $n \geq d$ . For  $n < d$ , we reason in the same style:

$$q \leq n \div d$$

$$\Leftrightarrow \{ \text{(9) and transitivity, since } n < d \}$$

$$q \times d \leq n \wedge q \times d < d$$

$$\Leftrightarrow \{ \text{since } d \neq 0 \}$$

$$q \times d \leq n \wedge q \leq 0$$

$$\Leftrightarrow \{ q \leq 0 \text{ entails } q \times d \leq n, \text{ since } 0 \leq n \}$$

$$q \leq 0$$



## If-then-else's — eventually!

Thus students have calculated the **then** and **else**-parts of the algorithm:

$$n \div d = \text{if } n < d \text{ then } 0 \text{ else } (n - d) \div d + 1$$

Side comment: **if-then-else** combinator considered harmful if introduced too early in the training — it enables students to start taking **arbitrary** decisions.

## Basic skills so far

“Design patterns” to master in getting programs out of textbooks, besides **indirection** and basic arithmetics:

- **distribute**

$$\frac{a}{c|d} = \frac{a}{c} | \frac{a}{d}$$

- **assoc**

$$a|(b|c) = (a|b)|c$$

- **abide**

$$\frac{a|b}{c|d} = \frac{a}{c} | \frac{b}{d}$$

(cf. matrices)

## Wrapping up

- *Matisse* much wider than shown in this talk (just a task of the project)
- Emphasis on **algorithmic problem solving** — cf. work by Roland's student João Ferreira
- Includes support technology development — cf. work by Alexandra Mendes, also at Nottingham
- Added after Eric's talk: lots of **probability** stuff in textbooks — an opportunity for new calculation approaches !
- **Matisse** is just starting — everybody welcome to collaborate and report on their own experience.



J. Kramer.

Is abstraction the key to computing?

*Commun. ACM*, 50(4):37–42, April 2007.



L. Russo.

*The Forgotten Revolution: How Science Was Born in 300BC  
and Why It Had to Be Reborn.*

Springer-Verlag, September 2003.