Which Mathematics for the Information Society?

João F. Ferreira[1], Alexandra Mendes[1], Roland Backhouse[1] and L. S. Barbosa[2]

[1] School of Computer Science, University of Nottingham, Nottingham, England
{joao@joaoff.com, afm@cs.nott.ac.uk, rcb@cs.nott.ac.uk}
[2] CCTC & Dep. Informatics, Minho University, Braga, Portugal
lsb@di.uminho.pt

Abstract. MathIS is a new project that aims to reinvigorate secondary-school mathematics by exploiting insights of the dynamics of algorithmic problem solving. This paper describes the main ideas that underpin the project. In summary, we propose a central role for formal logic, the development of a calculational style of reasoning, the emphasis on the algorithmic nature of mathematics, and the promotion of self-discovery by the students. These ideas are discussed and the case is made, through a number of examples that show the teaching style that we want to introduce, for their relevance in shaping mathematics training for the years to come. In our opinion, the education of software engineers that work effectively with formal methods and mathematical abstractions should start before university and would benefit from the ideas discussed here.

We are all shaped by the tools we use,
in particular the formalisms we use shape our thinking habits,
for better or for worse, and that means we have to be very careful in
the choice of what we learn and teach, for unlearning is really not possible.
— E. W. DIJKSTRA in [10]

1 Introduction

Modern IT-driven societies demand highly skilled professionals who can successfully design complex systems at ever-increasing levels of reliability and security. Such a demand requires from these professionals a high degree of *mathematical fluency*, that is, the ability to resort to the mathematical language and method to build models of problems, and reason effectively within them.

However, there is little hope that such demands be met by current standards in school maths education. Students are not being adequately trained in formal reasoning and proof, and, as a result, they have difficulties in employing mathematics to solve new problems.

This paper describes our ideas on how to reinvigorate mathematics education. In summary, we propose a central role for formal logic, the development of a calculational style of reasoning, the emphasis on the algorithmic nature of mathematics, and the promotion of self-discovery by the students. This work is

being done in the context of the MathIS project [3], whose goal is to exploit the dynamics of algorithmic problem solving and calculational reasoning in both maths education and the practice of software engineering.

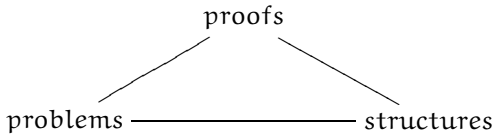
We start in section 2 by providing some background on the existing problems of the current standards in mathematics education and by explaining in detail how we think they can be improved.

We exemplify our ideas in section 3, where we present extracts of educational material that illustrate how we would rewrite and teach mathematics. The first example is a recreational problem, inspired by chess, that shows the effectiveness of a calculational formal logic. We often use recreational examples, since they can make serious concepts more palatable to students. The second example is on integer division. Elementary number theory is inherently algorithmic and we believe that exploiting this attribute can improve the way we teach it. We conclude the section by explaining how we think the material should be introduced. Our approach is based on teaching scenarios, which are detailed guidelines on how to solve specific problems. These scenarios are primarily written for teachers and they are designed to promote self-discovery.

The paper is concluded by a discussion on future work, assessment, and tool support.

2 Mathematics as *the art of effective reasoning*

The triangle



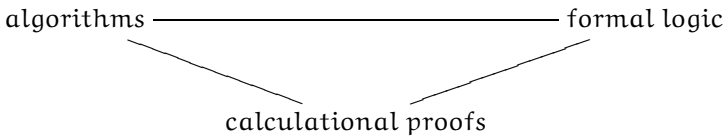
provides a perspective on the dynamics of mathematics as the interplay of its three vertices. Mathematics starts from *problems* which are modelled, abstracted, and generalised into precise (and hopefully, simple) *structures* upon which it becomes possible to reason formally about them, characterise possible solutions, and establish their properties. The adjective *formal* not only qualifies mathematical *proofs* as rigorous, but also suggests an underlying discipline for their production and communication.

Such a discipline is often absent, or disguised, or simply left implicit, in classical mathematical texts and sometimes even regarded with suspicion. However, the presence of a discipline would have tremendous benefit because it enables students to acquire mental tools which empower their reasoning skills and make them more rigorous and productive.

[3] MathIS is a 3-year project that has started in January 2009. More details are available from the project portal at `www.di.uminho.pt/mathis`

Also, the notion of *proof* is almost swept under the carpet in most school manuals and in teaching practice. Teachers usually associate proofs with intricate, heavily semantic arguments, often presented in a non-systematic, pseudo-intuitive way, which they (rightly) suppose an average student is not able to grasp, let alone to master, reproduce or adapt to new contexts. If proofs, as usually taught, are inadequate, a mathematical discipline relegating proofs to a minority of highly crafted students is misleading and useless. Actually, not only proving skills, i.e. the ability of formally explaining and justifying an argument, lie at the very heart of what mathematics is about, but also in modern IT societies, proofs (of functional correctness, of security compliance, etc.) have achieved the status of a core business. Proofs pay V.A.T. nowadays. This is why training software engineers in formal development methods has become so important as to justify holding international conferences on the subject. We believe, however, the issue goes far deeper into the educational system and should also be addressed at such levels.

But how can the contributions of computing science improve current standards in mathematical education? As a contribution to a wider debate, we would like to single out in this paper three main topics: the rediscovery of the central role of *formal logic*, the development of a *calculational* style of reasoning, and the emphasis on the *algorithmic character* common to a great number of mathematical problems. In a sense, they form the vertices of another triangle mirroring the one depicted above:



(1)

A central role for formal logic. The rediscovery of the central role of *formal logic*, not only in foundations, but also in applied mathematics and as part of a number of standard engineering curricula, is perhaps the main consequence to mathematics of computing science development. An indicator of this move is the almost universal presence of a course on formal logic in every computing science or mathematics undergraduate curriculum.

Proficiency in mathematics, however, would benefit from an earlier introduction and explicit use of logic in high school. Note this is usually not the case in most European countries; the justification for such an omission is that *logic is implicit in mathematics and therefore does not need to be taught as an independent issue*. Such an argument was used in Portugal to eliminate logic from the high-school curriculum in the nineties. The damage it caused is still to be assessed, but it is certainly not alien to the appalling indicators and statistics in what concerns the country overall ranking in mathematics education [20].

We believe that logic should be introduced using simple problems that emphasize formalization and calculation. Recreational problems like the one we show in section 3.2, for example, are easy to understand and usually grab students' attention. Also, logic puzzles, where the goal is to solve simultaneous

equations on booleans, can be introduced by analogy with simultaneous equations on numbers. High-school students already learn how to solve simultaneous equations on numbers; going from the reals to the simpler boolean domain, where each variable is either true or false, should be no problem.

A calculational reasoning style. Two decades of research on *correct-by-construction* program design have created a new discipline of algorithmic problem solving and shed light on the underlying mathematical structures, modelling, and reasoning principles. Starting with the pioneering work of Dijkstra and Gries [12,15], and in particular, through the development of the so-called *algebra of programming* [8,4], a *calculational style* [2,22,11] emerged, emphasising the use of systematic mathematical calculation in the design of algorithms. This was not new, but routinely done in algebra and analysis, albeit subconsciously and not always in a systematic fashion. The realization that such a style is equally applicable to logical arguments [12,15] and that it can greatly improve on traditional verbose proofs in natural language has led to a systematization that can, in return, also improve exposition in the more classical branches of mathematics. In particular, lengthy and verbose proofs (full of *dot-dot* notation, case analyses, and natural language explanations for "obvious" steps) are replaced by easy-to-follow calculations presented in a standard layout which replaces classical implication-first logic by variable-free algebraic reasoning [22,14].

The systematization of a calculational style of reasoning, proceeding in a formal, essentially syntactic way, can greatly improve on the way proofs are presented. In particular it may help to overcome the typical justification for omitting proofs in school mathematics: that they are difficult to follow but for exceptional students. Moreover, in school mathematics, there are many examples which show how the formalization of topics arising in different contexts results in formulae with the same *flavour*, which can be manipulated thereafter by the same rules of the predicate calculus, without reference to a 'domain specific' interpretation of such formulae in their original area of discourse. This is the essence of formal manipulation, and yields proofs that are shorter, explicit, independent of hidden assumptions, easy to re-construct, check and generalise.

That such a syntax-driven approach is extremely effective in practice cannot be understated. For example, it was the formal manipulation of Maxwell's equations that led to conjecturing the existence of electromagnetic waves, confirmed experimentally shortly afterwards. As noted by Dijkstra, presenting calculation, i.e. the manipulation of uninterpreted formulae, as an alternative to traditional, informal mathematical reasoning, accomplishes Leibniz's dream: *Traditional mathematics did not provide the most hospitable environment for its realization; this, therefore, had to take place in a separate discipline, which is now known as Computing Science* [9]. Several authors [16,21] point out that in mathematics, formal calculation is both a convenience that people, contrary to popular opinion, naturally adopt, and an asset for discovery and development. The only obstacle that keeps it from universal and systematic use throughout science (and thereby feeds prejudice) is the calculationally deficient notation that still prevails in many disciplines.

Making explicit the algorithmic contents of mathematics. Another contribution of computing science to mathematical education is in the systematic identification of the algorithmic content of a large part of mathematics. Recall that, algorithmic problems are the ones where the solution involves, possibly implicitly, the design of an algorithm, i.e., a sequence of instructions that can be mechanically executed to solve it.

Algorithms have been studied since the beginning of civilization. However, the advent of the digital age has revolutionized the nature, the pace and the importance of algorithm development. The unprecedented demands on precision and concision that this entails have brought about massive improvements in our problem-solving skills [3]. Their potential for reshaping mathematical teaching, introducing powerful behaviour abstractions (such as *invariants* or *contracts*), problem decomposition techniques or goal-directed derivations, is just beginning to loom.

In the next section, we show two examples that can be used to introduce algorithmic skills. First, in subsection 3.3, we show how the emphasis on a calculational approach can also lead to a constructive and precise derivation of the integer division algorithm. In subsection 3.4, we present an algorithmic problem whose goal-oriented solution is based on problem decomposition and invariants.

3 An educational programme

How can such an 'inheritance' of computing science development be carried back to high-school mathematics and effectively improve current teaching standards?

As mentioned in the introduction, the authors are currently involved in planning and implementing a pilot educational programme, targeting high-school students, to address this question.

Our first observation is that the simple reasoning style introduced in the first year of high-school algebra is calculational. A deeper analysis of the structure of these calculations (formerly taught by examples) establishes a basis for consolidation and for extension to other, more advanced mathematical subjects. On the other hand, exploring the dynamics of algorithmic problem solving, working from concrete problems through goal-directed constructions, will strengthen their logical skills and ability to reason in an efficient way. We believe that exploring the vertices of triangle (1) has a potential to make students proficient in structuring formal arguments and aware of the central role of *proofs* in the mathematical practice. This programme will help to validate such hypothesis.

3.1 The programme

The main component of this programme is the development of specific educational material supporting the use of a calculational approach and algorithmic problem solving strategies in the *practice of mathematics*. This material, in the

form of example-driven *teaching scenarios* whose structure is detailed in subsection 3.4, is designed for use with teams of up to 20 volunteer high school students in the context of extra-curricular "Maths' Clubs". Our focus is placed on two domains which are both understood as strategic by the secondary school teachers collaborating with the project (mainly the first one) and attractive to students (mainly the second):

- the *refactoring* of specific areas of the high-school mathematics curriculum, to build an alternative to their usual presentation in the classroom,
- *recreational mathematics*, in the form of logic puzzles and combinatorial games, which, lying outside standard mathematics curricula, are an attractive source of non trivial examples for the envisaged techniques.

In both cases the emphasis is placed on the judicious use of formal logic in mathematical reasoning and the intertwined development of calculational proofs and algorithms, making the, often hidden, algorithmic contents explicit. Both domains are illustrated in detail in the next two sub-sections. Finally, subsection 3.4 discusses the structure of a *teaching scenario* by means of a concrete example.

3.2 Recreational mathematics

Les hommes ne sont jamais plus ingénieux que dans l'invention des jeux.
(Men are never more ingenious than in inventing games.)
—GOTTFRIED W. LEIBNIZ to De Montmort
(29 July 1715)

Recreational mathematics is a type of mathematics that usually appeals to students and inspires them to study further. Recreational problems are often based on real-life situations, and thus, are easily understood and do not generally require an advanced knowledge of mathematics to be solved. Here we present an example inspired by chess, whose solution can be easily obtained by calculation.

Problem (chess moves) In chess, a bishop moves along the diagonal. That is, starting from a position $(\mathtt{i},\mathtt{j})$, a bishop can move a (positive or negative) distance $\mathtt{k}$ to the position $(\mathtt{i}{+}\mathtt{k}\,,\,\mathtt{j}{+}\mathtt{k})$ or to the position $(\mathtt{i}{+}\mathtt{k}\,,\,\mathtt{j}{-}\mathtt{k})$. (This is provided, of course, that the bishop stays within the boundary of the board.)

Show that a move from the position $(\mathtt{i},\mathtt{j})$ to the position $(\mathtt{i}{+}\mathtt{k}\,,\,\mathtt{j}{+}\mathtt{k})$ does not change the colour of the square. **Hint:** The definition

$$[\ \mathit{white}.(\mathtt{i},\mathtt{j})\ \equiv\ \mathit{even}.\mathtt{i}\ \equiv\ \mathit{even}.\mathtt{j}\ ]\,,$$

can be useful. (The square so-called "everywhere" brackets denote universal quantification over all the free variables.)

A calculational solution The goal is to prove the following two equalities:

$$[\ \mathit{white}.(\mathtt{i},\mathtt{j}) \ \equiv \ \mathit{white}.(\mathtt{i}{+}\mathtt{k}\ ,\ \mathtt{j}{+}\mathtt{k})\ ]\text{, and} \tag{2}$$

$$[\ \mathit{white}.(\mathtt{i},\mathtt{j}) \ \equiv \ \mathit{white}.(\mathtt{i}{+}\mathtt{k}\ ,\ \mathtt{j}{-}\mathtt{k})\ ]\ . \tag{3}$$

A calculational and annotated proof of (2) is as follows:

$$\begin{array}{ll}
 & (\mathit{white}.(\mathtt{i},\mathtt{j}))[\mathtt{i}\ ,\mathtt{j}\ :=\ \mathtt{i}{+}\mathtt{k}\ ,\mathtt{j}{+}\mathtt{k}] \\
= & \quad \{ \quad \text{substitution (corresponds to the move)} \quad \} \\
 & \mathit{white}.(\mathtt{i}{+}\mathtt{k}\ ,\ \mathtt{j}{+}\mathtt{k}) \\
= & \quad \{ \quad \text{definition of } \mathit{white} \quad \} \\
 & \mathit{even}.(\mathtt{i}{+}\mathtt{k}) \equiv \mathit{even}.(\mathtt{j}{+}\mathtt{k}) \\
= & \quad \{ \quad \mathit{even} \text{ distributes over addition, i.e.,} \\
 & \qquad [\ \mathit{even}.(\mathtt{a}{+}\mathtt{b}) \ \equiv \ \mathit{even}.\mathtt{a} \ \equiv \ \mathit{even}.\mathtt{b}\ ] \quad \} \\
 & \mathit{even}.\mathtt{i} \equiv \mathit{even}.\mathtt{k} \equiv \mathit{even}.\mathtt{j} \equiv \mathit{even}.\mathtt{k} \\
= & \quad \{ \quad \text{associativity and symmetry of } \equiv \quad \} \\
 & \mathit{even}.\mathtt{i} \equiv \mathit{even}.\mathtt{j} \equiv \mathit{even}.\mathtt{k} \equiv \mathit{even}.\mathtt{k} \\
= & \quad \{ \quad \text{associativity and reflexivity of } \equiv \quad \} \\
 & \mathit{even}.\mathtt{i} \equiv \mathit{even}.\mathtt{j} \\
= & \quad \{ \quad \text{definition of } \mathit{white} \quad \} \\
 & \mathit{white}.(\mathtt{i},\mathtt{j})\ .
\end{array}$$

The proof of (3) is similar and left to the reader. Note that the solution constitutes a unified interface for reasoning about how the colour of the squares change regardless of the chess piece. For example, we can use the same proof structure to prove that the move of a knight always changes the colour of the square (in that case, the key property is that the numbers 1 and 2 have different parities).

Standard solutions to parity problems are usually done within the familiar domain of numbers. In this particular example, a standard solution would claim that the parities of $\mathtt{i}{+}\mathtt{k}{+}\mathtt{j}{+}\mathtt{k}$ and $\mathtt{i}{+}\mathtt{j}$ are the same. However, reasoning within the boolean domain can be more effective: the algebraic manipulations may be less familiar than ordinary arithmetic, but they are easier because the domain is much simpler.

3.3 Refactoring school mathematics

The elementary theory of numbers should be one of the very best subjects for early mathematical instruction. It demands very little previous knowledge, its subject matter is tangible and familiar; the processes of

reasoning which it employs are simple, general and few; and it is unique among the mathematical sciences in its appeal to natural human curiosity.
—G. H. HARDY in the sixth Josiah Willard Gibbs Lecture
(New York, 1928)

We now present an example taken from elementary number theory, an area in which many important concepts are of algorithmic nature [6] [7]. In particular, we show how the specification of integer division as a Galois connection can lead to effective proofs and how to use it as a specification of the division algorithm.

Integer Division as a Galois Connection The integer division of P by Q, here denoted by $P \div Q$, is introduced early in school as the integer x such that

$$P = x \times Q + r \quad \wedge \quad 0 \leq r < Q \quad .$$

This formulation is usually accompanied by many examples that convey the concept of division, dividend, and remainder. However, we believe that the students do not learn how to reason effectively about division. Properties like the following

$$[\ (a \div b) \div c = a \div (c \times b)\ ] \tag{4}$$

are rarely discussed, and even when they are, their justification is typically informal and imprecise. To improve the situation, we propose the introduction of the integer division as the Galois connection:

$$[\ k \times Q \leq P \ \equiv\ k \leq P \div Q\ ] \quad . \tag{5}$$

We can use this definition to effectively prove properties of integer division. For instance, replacing k by $P \div Q$, we establish the property:

$$[\ (P \div Q) \times Q \leq P\ ] \quad .$$

We can also conclude that $0 \leq P$ is equivalent to $0 \leq P \div Q$, by replacing k by 0. Also, using indirect equality, definition (5) can used to prove property (4) in just three steps:

$$\begin{array}{ll} & k \leq (a \div b) \div c \\ = & \quad \{ \quad \text{definition (5)} \quad \} \\ & k \times c \leq a \div b \\ = & \quad \{ \quad \text{definition (5) and associativity} \quad \} \\ & k \times (c \times b) \leq a \\ = & \quad \{ \quad \text{definition (5)} \quad \} \\ & k \leq a \div (c \times b) \quad . \end{array}$$

Moreover, definition (5) is a suitable specification for an algorithm that computes $P \div Q$. Note that the goal of such an algorithm is to compute a solution to the equation

$$x :: \ [\ k \times Q \leq P \ \equiv \ k \leq x\ ] \quad .$$

If a solution to this equation exists, then it is unique (because the relation $\leq$ is reflexive and anti-symmetric). Furthermore, an important property of the solution x is that it is the largest integer that satisfies

$$x \times Q \leq P \quad . \tag{6}$$

As a consequence, x also satisfies

$$\neg((x+1) \times Q \leq P) \quad . \tag{7}$$

In other words, the goal is to compute a value x that satisfies properties (6) and (7). For brevity, we do not show the full derivation. Instead, we would like to stress that the derivation of the division algorithm is an educational example that can be used to teach algorithmic techniques such as loop formation, using the invariant to calculate assignments, and proving progress. These standard techniques in algorithm development can be used to obtain the following intermediate algorithm:

$\{\ 0 < Q \ \wedge \ 0 \leq P\ \}$

$x \ := \ 0\,;$

$\{$ **Invariant:** $\ x \times Q \leq P\ \}$

do $\ (x+1) \times Q \leq P \ \rightarrow \ x := x+1$

od

$\{\ x \times Q \leq P \ \wedge \ \neg((x+1) \times Q \leq P)\ \}$.

Another technique that can be taught is the introduction of extra variables and computations to produce more efficient versions. In this example, the algorithm above can be further optimized by introducing the computation of the remainder.

Also, the calculational approach allows us to be more constructive because the requirements emerge from the calculations. As an example, we do not need to assume that the divisor Q is positive; it emerges as a necessary condition in the proof that the bound-function is bounded below.

Recall that a *bound function* is a natural-number-valued function of the program variables that measures the size of the problem to be solved. A guarantee that the value of such a bound function is always decreased at each iteration is a guarantee that the number of times the loop body is executed is at most the initial value of the bound function. In this example, a possible bound function is $P - x$, and the proof that it is bounded below is as follows:

$0 \leq P - x$

$$
\begin{array}{lll}
= & \{ & \text{cancellation} \ \} \\
& x \leq P & \\
= & \{ & \text{we know from the invariant that } x \times Q \leq P; \\
& & \textbf{assuming that } 0 < \mathbf{Q}, \text{ we have } x \leq x \times Q; \\
& & \text{because } \leq \text{ is transitive, we also have } x \leq P \quad \} \\
& \mathsf{true} \ . &
\end{array}
$$

Note that the assumption $0 < Q$, highlighted in bold in the calculation, emerges naturally from the shape of the invariant.

3.4 Teaching scenarios

What is teaching?
In my opinion, teaching is giving opportunity
to the students to discover things by themselves.
— GEORGE PÓLYA in TEACHING US A LESSON
(MAA Video Classics, Number 1)

The success of teaching depends on the amount of discovery that is left for the students: if the teacher discloses all the information needed to solve a problem, students act only as spectators and become discouraged; if the teacher leaves all the work to the students, they may find the problem too difficult and become discouraged too. It is thus important to find a balance between these two extremes.

We propose the introduction of educational material in the form of teaching scenarios, which are fully worked out solutions to algorithmic problems together with "method sheets"— detailed guidelines on the principles captured by the problem, how the problem is tackled, and how it is solved. Although they can be used directly by the student, they are primarily written for the teacher and they are designed to maintain a balance between the two extremes mentioned above. In other words, they are designed to promote self-discovery. In general, each scenario is divided into the following sections:

- **Brief description and goals** This section provides a summary of the scenario, allowing the teacher to determine if it is adequate for the students.
- **Problem statement** This section states the problem (or problems) discussed in the scenario.
- **Students should know** This section lists pre-requisites that should be met by the students. The teacher can use it to determine if the scenario is adequate for the students.
- **Resolution** This section presents a possible solution for the problem in the style advocated here.
- **Notes for the teacher** In this section, the solution presented above is decomposed into its main parts and each part is discussed in detail. To maintain the balance mentioned in the first paragraph, we also recommend how

the teacher should present the material, including questions that the teacher should or should not ask and important concepts that should be introduced.
- **Extensions and exercises** This section can be used for homework or project assignments. All the exercises are accompanied by their solutions.
- **Further reading** Recommended reading for the teacher and the students. It may include discussions and comparisons between conventional solutions and the one presented in the scenario.

We now present some extracts from a scenario that generalizes the problem "The Chameleons of Camelot", found in [17, p. 140]. Its goal is to help students recognizing, modelling, and solving algorithmic problems. The solution is goal-oriented and explores an invariant of the underlying non-deterministic algorithm. It is also an example of problem decomposition and it can be used to convey the notions of loop, guard, postcondition, and non-determinism. To obtain the full version, please visit the website `http://joaoff.com/aps/scenarios`.

The Chameleons of Camelot On the island of Camelot there are three different types of chameleons: grey chameleons, brown chameleons, and crimson chameleons. Whenever two chameleons of different colours meet, they both change colour to the third colour.

For which numbers of grey, brown, and crimson chameleons is it possible to arrange a succession of meetings that results in all the chameleons displaying the same colour?

For example, if the number of the three different types of chameleons is 4, 7, and 19 (irrespective of the colour), we can arrange a succession of meetings that results in all the chameleons displaying the same colour. An example is:

$$(4\,,7\,,19)\rightarrow(6\,,6\,,18)\rightarrow(0\,,0\,,30)\quad .$$

On the other hand, if the number of chameleons is 1, 2, and 3, it is impossible to make them all display the same colour.

Notes for the teacher As said before, teaching scenarios are primarily written for the teacher and are designed to promote self-discovery. The following extract illustrates the general tone of the section *Notes for the teacher*; in particular, it suggests a formalization of the goal and how to decompose the problem:

- **Determine the postcondition and decompose the problem** Now that we have modelled the underlying algorithm, we have to express our goal. The answer comes directly from the problem statement: "For which numbers of grey, brown, and crimson chameleons is it possible to arrange a succession of meetings that results in all the chameleons displaying the same colour?". So we need to express formally that all the chameleons display the same colour. One alternative is:

$$g = b = 0 \vee b = c = 0 \vee c = g = 0\quad . \tag{8}$$

> An informal description can be useful (e.g. "Provided that there is at least one chameleon, the first disjunct means that there are only crimson chameleons, the second means that there are only green chameleons, and the third means that there are only brown chameleons. Their disjunction means that at least one of these statements is true."). At this stage, the teacher should note that instead of working directly with the final goal (8), we can think of how to get to intermediate states that simplify the problem. The teacher should lead the students to the observation that if any two types of chameleons are equally numbered, we can arrange a meeting between all the chameleons of these two types. We suggest the teacher start with some concrete examples until the students get there (e.g. $(0\,,0\,,0)$, $(5\,,3\,,5)$, and $(10\,,171\,,10)$). Formally, we can express these states as:
>
> $$g = b \ \vee \ b = c \ \vee \ c = g \quad . \tag{9}$$
>
> If the algorithm reaches a state that satisfies this expression, it remains to arrange a meeting between all the chameleons of two equally numbered classes.
> (...)

The section also includes questions that we recommend the teacher to ask, together with a justification for its importance:

- *What is our goal? What do we want to prove? How do we express it formally?*

  Every time we are working in a goal-oriented fashion, this question should be asked explicitly. The teacher may need to help the students formalizing the states where there are chameleons of only one colour; if that is the case, we suggest they help with the first disjunct and let the students do the other two.

To emphasize the self-discovery nature of the scenarios, we also include obtrusive questions that the teacher should not ask:

- *Can you see that if two different types of chameleons are equally numbered, the problem is easy to solve?*

  The teacher should start by asking the students which states make the problem easy to solve. With the help of some examples, we believe that most students will get to the fact that if two different types of chameleons are equally numbered, the problem is easy to solve.

Self-discovery is also promoted by the sections *Extensions and exercises* and *Further reading*, which are both designed to encourage further work by the students.

4 Conclusions and future work

Our own experience in teaching formal methods at the university tells us (and employers of our students confirm so) that good thinking habits rooted in sound principles add to overall effectiveness and aptness to face adversity and unexpected challenges. If explicit examples need to be found on the advantages of a sound mathematical basis in software engineering, the 2004 collapse of the Portuguese school teacher allocation system serves as a typical illustration: the problem was later solved by a small software house which claims to use formal methods in their normal practice. This situation drove the country's attention to the need for better trained software engineers.

Our claims in this paper are that there is a need to act at lower levels of the educational system and that a number of issues arising from computing science research may have a decisive impact on reinvigorating mathematics education to meet the challenges of modern IT-driven societies. Stressing the algorithmic content of mathematics, for example, already led us to novel results in number theory [6] [7].

It is important to note that there is more educational material than the examples shown in this paper. In fact, we believe that the project can only succeed if there is an abundance of material and guides ready for the teachers to use. In our opinion, providing resources and assistance to the teachers is the best way to overcome the challenge of convincing them to use the approach we propose.

One problem we foresee is the difficulty in assessing the impact of our project. The use of test and control groups, randomized trials, and assessment based on lectures to students in Math's Clubs have serious flaws. (Some of the difficulties involved in the assessment are pointed out by Herbert Wilf in his essay [24].) Nevertheless, the novel results mentioned above and preliminary results on the didactical suitability of the calculation format obtained within the group (see [13]) encourage us to continue our efforts. Also, the success claimed by related work like [1] and [19], makes us believe that we can have a positive impact.

A topic we have omitted in this paper but which is central to the MATHIS project concerns the development of tool support. Actually, this capitalizes on recent developments and increased flexibility in human-computer interaction technology, which we believe is mature enough to provide an infra-structure for the envisaged methodological shift. A collection of tools to enable on-screen calculation with mathematical formulae in a blackboard-like style, resorting to tablet PCs and e-learning platforms, is under development [18]. A direct inspiration for such tools is MATHSPAD [5,23], a general-purpose structure editor for on-screen algebraic calculation.

Acknowledgements. Long-term collaboration with J. N. Oliveira on calculational approaches to mathematics is deeply acknowledged. This research was supported by FCT (the Portuguese Foundation for Science and Technology), in the context of the MATHIS Project under contract `PTDC/EIA/73252/2006`.

The work of João F. Ferreira and Alexandra Mendes was further supported by FCT grants SFRH/BD/24269/2005 and SFRH/BD/29553/2006, respectively.

References

1. Ralph-Johan Back, Linda Mannila, Mia Peltomaki, and Patrick Sibelius. Structured derivations: A logic based approach to teaching mathematics. In *FORMED 2008: Formal Methods in Computer Science Education, Budapest*, 2008.
2. R. C. Backhouse. Mathematics and programming. A revolution in the art of effective reasoning. Inaugural Lecture, School of Computer Science and IT, University of Nottingham, 2001.
3. R. C. Backhouse. *Program Construction*. John Wiley and Sons, Inc., 2003.
4. R. C. Backhouse and P. F. Hoogendijk. Elements of a relational theory of datatypes. In B. Möller, H. Partsch, and S. Schuman, editors, *Formal Program Development*, pages 7–42. Springer Lect. Notes Comp. Sci. (755), 1993.
5. R. C. Backhouse and R. Verhoeven. Mathspad: A system for on-line preparation of mathematical documents. *Software - Concepts and Tools*, 18:80–89, 1997.
6. Roland Backhouse and João F. Ferreira. Recounting the rationals: Twice! In *Mathematics of Program Construction*, volume 5133 of *LNCS*, pages 79–91, 2008.
7. Roland Backhouse and João F. Ferreira. On Euclid's algorithm and elementary number theory. Submitted for publication. Available from http://joaoff.com/publications/2009/euclid-alg/, 2009.
8. R. Bird and O. Moor. *The Algebra of Programming*. Series in Computer Science. Prentice-Hall International, 1997.
9. E. W. Dijkstra. A new science, from birth to maturity. note EWD1024, 1988.
10. E. W. Dijkstra. On the cruelty of really teaching computing science. note EWD1036, 1988.
11. E. W. Dijkstra. On the economy of doing mathematics. note EWD1130, 1992.
12. E. W. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics*. Springer Verlag, NY, 1990.
13. João F. Ferreira and Alexandra Mendes. Student's feedback on teaching mathematics through the calculational method. In *39th ASEE/IEEE Frontiers in Education Conference*. IEEE, 2009.
14. D. Gries, W. H. J. Feijen, A. J. M. van Gasteren, and J. Misra. *Beauty is our Business*. Springer Verlag, 1990.
15. D. Gries and F. Schneider. *A Logical Approach to Discrete Mathematics*. Springer Verlag, NY, 1993.
16. David Gries. Improving the curriculum through the teaching of calculation and discrimination. *Communications of the ACM*, 34(3):45–55, 1991.
17. Ross Honsberger. *In Polya's Footsteps: Miscellaneous Problems and Essays (Dolciani Mathematical Expositions)*. The Mathematical Association of America, October 1997.
18. Alexandra Mendes. Work in progress: Structure editing of handwritten mathematics. In *38th ASEE/IEEE Frontiers in Education Conference*. IEEE, 2008.
19. Z. Michalewicz and M. Michalewicz. *Puzzle-based Learning: Introduction to Critical Thinking, Mathematics, and Problem Solving*. Hybrid Publishers, 1st edition, 2008.
20. OCDE Report. *Education at a clance: OCDE indicators 2006*. Paris: OCDE Publishing, 2006.
21. DIMACS Symposium. Teaching logic and reasoning in an illogical world. Technical report, Rutgers University, 1996.

22. A. J. M. van Gasteren. *On the Shape of Mathematical Arguments*. Springer Lect. Notes Comp. Sci. (445), 1990.
23. R. Verhoeven and R. C. Backhouse. Towards tool support for program verification and construction. In Jim Woodcock Jeanette Wing and Jim Davies, editors, *FM'99 - Int. Formal Methods Symposium*, pages 1128–1146. Springer Lect. Notes Comp. Sci. (1709), 1999.
24. Herbert S. Wilf. Can there be "research in mathematical education"? Available from `http://www.math.upenn.edu/~wilf/website/PSUTalk.pdf`.