ACTA DIDACTICA NAPOCENSIA

Volume 10, Number 1, 2017

FORMAL ABSTRACTION IN ENGINEERING EDUCATION — CHALLENGES AND TECHNOLOGY SUPPORT

Walther A. Neuper

Abstract. This is a position paper in the the field of Engineering Education, which is at the very beginning in Europe. It relates challenges in the new field to the emerging technology of (Computer) Theorem Proving (TP).

Experience shows, that *teaching* abstract models, for instance the wave equation in mechanical engineering and in electrical engineering, is difficult. This paper suggests novel technology to support *learning* in a novel way such that abstract models are better understood eventually.

Such support acknowledges learning and mastering abstraction as a long-term process, which requires revisiting physical and mathematical concepts again and again, which in turn continuously raises the level of abstraction at an individual pace and fosters students' familiarity with formal descriptions capturing essential properties of models and respective elements.

The paper discusses the potential of TP to support students' process of abstraction and expected impact on teaching mathematics at engineering faculties.

Key words: Engineering Education, Formal Models, Mathematical Abstraction, Interactive Worksheets, Computer Theorem Proving

. . . *''in recent years, many studies have highlighted an alarming decline in young people's interest for key science studies and mathematics. Despite the numerous projects and actions that are being implemented to reverse this trend, the signs of improvement are still modest. Unless more effective action is taken, Europe's longer term capacity to innovate, and the quality of its research will also decline.''* . . . these are the first lines of the Rocard-Report [46] written by the Directorate-General for Research on behalf of the EU Commission in 2007.

Another point are studies in STEM (*S*cience, *T*echnology, *E*ngineering and *M*athematics) education, which addict the ''alarming decline of interest'' in academic education to mathematics [49].

So there are challenges in education for mathematics and engineering.

1 Introduction: Challenges in engineering education

Since challenges in STEM education have been identified on a high administrative level by [46], considerable efforts have been invested. In the EU and in the US high-schools reacted quickly and started well-funded projects, for instance [9, 5, 6]. The keywords for these projects and respective recommendations are ''inquiry-based learning'' [10] and ''competence-oriented teaching'' [42]. The consequence for general practice of mathematics education appears to be less frustration of pupils, but also less formal operation and reduced knowledge --- which causes lots of sharp criticisms, e.g. [31], from academic STEM teachers.

At the academic level, which is addressed by this paper, reaction was slower. By now some universities of technology have appointed institutional resources to the topic, for instance at Darmstadt [1], at München [2] or at Hamburg [3]. There is initial work on theoretical foundations [16]. But still there seems no

[1] http://www.uni-stuttgart.de/bpt/

[2] http://www.prolehre.tum.de/

[3] https://cgi.tu-harburg.de/~zllwww/

literature to specifically address teaching applied mathematics at the academic level. So the paper starts more or less from scratch.

Reality and formality

Bruno Buchberger in [18] describes the róle of mathematics in science and generally in history of mankind (see Fig.1).

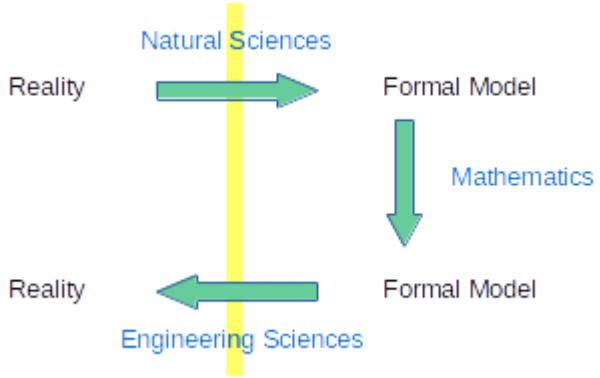

Figure 1: The róle of reality and formality, B. Buchberger [18].

There is a yellow line between the real world and the world of formal models in mathematics: the former is accessible by our senses, our actions in this world are governed by (hopefully) common sense --- where the latter is accessible by thought only and governed by formal logic.

One should not forget a major motivation for starting the great European development of mathematics in the age of enlightenment: Pascal, Descartes, Newton, Leibnitz and further hundreds of bright minds developed their thinking as a means against political and clerical power --- self-restriction to the pure laws of thinking should break traditions. So one hardly can overestimate the challenge posed to humans in learning this kind of thinking.

Nowadays mathematical reasoning in science and technology is considered more trustworthy than common sense, this kind of reasoning is what mathematics distinguishes from other disciplines. Mathematical reasoning has been consolidated into formal logics during the last fifteen decades. And during the last five decades mathematical reasoning has been mechanised in computers (the respective technology will be introduced below and it will be debated, if such mechanisation can help in learning).

So a first aspect of the challenge is: *mathematical reasoning is necessarily* very different *from common sense and students have to learn this different kind of reasoning.*

Formal models and levels of abstraction

Formal models in engineering are composed from simpler elements, some of which are basic physical concepts and some of which are mathematical objects.

Physical concepts are more abstract than frequently noticed. For instance, the notion of ‘‘velocity’’ seems clear for everyone, but ‘‘instantaneous velocity’’ already involves the mathematical notion of ‘‘limit’’. The level of abstraction increases with notions like ‘‘acceleration’’. A student learns

$\overrightarrow{F} = m\overrightarrow{a}$, the second law of motion. But what is really required in order to tackle engineering problems is understanding interdependencies like:

$$\begin{aligned} s(t) &= \frac{g}{2} \cdot t^2 \\ s'(t) &= g \cdot t \\ s''(t) &= g \end{aligned}$$

Thus, comprehension of essential properties of ''acceleration'' appears impossible without formal representation of such properties. Intuitive understanding of ''velocity'', ''acceleration'', etc is supported by experiments and concrete experience in early physics education. And on the other hand farmiliarity with abstract formal representations appears equally important in order to understand formal models in engineering.

Mathematical objects and respective abstractions are concern of wide-spread efforts in didactics of mathematics; so there are more details about levels of abstraction. The notion of ''function'' serves as an example:

1. The concept of ''function'' is introduced in early mathematics education at high-school. Even in academic settings intuitive and sensual introductions are successful [22]. But here we assume an introductory course in some engineering studies, where ''function'' is introduced as a specific relation and illustrated by several examples.

2. Later in the course the concept is extended by ''function composition''; for the purpose of consolidation ''composition'' is exemplified by chaining $f(x) = x^2$ and $f(x) = \frac{1}{x}$ with linear functions and observing translation, extension and compression of the typical graphs. A notion of ''function'' as ''pointwise application'' in Pt.1 is sufficient for this second level; but the formal notion of ''identity'' comes in.

3. An analysis course introduces ''differentiation of functions''. Here, in particular with respect to the ''chain rule'', a more abstract concept of ''function'' is helpful: if comprehended as an object with certain properties, this would help to see invariants with respect to ''composition'' and to better understand notion and notation of the chain rule.

4. Later the analysis course introduces differential equations; now a fairly abstract concept of ''function'' is required to comprehend solutions as specific classes of functions, for instance for the wave-equation.

5. Below we shall consider an advanced course in mechanics which introduces the two-mass-oscillator [47] based on the wave-equation.

6. Finally, at the high-end of abstraction, assume the notion of ''function'' as a prerequisite for teaching ''Formal Methods'' [13] (probably a not so hypothetic assumption, as soon as this might be requested by industry in the near future). At this level students must have a firm and confident concept of ''function'' in formal representation.

These points constitute increasing levels of abstraction during a student's learning process. Without anchoring notions in human intuition during early phases of education on the lower levels abstraction, such notions become useless.

Whether students have a useless notion of ''function'' cannot easily be assessed in an examination: they learn the wording of respective abstractions by heart, they also copy how to positively accomplish

certain tasks --- but is the notion really firm enough to comprehend Pt.5? [4] Apparently, the abstraction as ‘‘specific relation’’ from Pt.1 needs to be specifically related to all other points.

So a second aspect of the challenge is: *models consist of elements, which are comprehensible at different* levels of abstraction*, where lower levels should not be skipped.*

Understanding comprehensive models

Now let's give an example of how physical concepts and mathematical objects are composed to a comprehensive model. The example problem below involves the wave equation, applied to an engineering problem, the two-mass-oscillator [47].p.122 for masses m. Given the forces in Fig.2, the problem is to determine a spring constant for a spring between the two masses such, that the system shows no vibration to the outside (for further details see the running example in [39]):

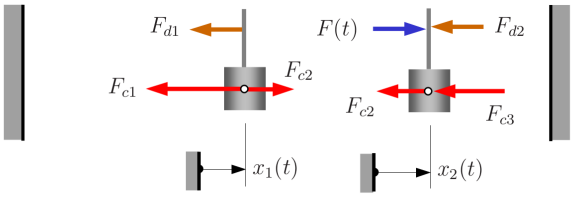

Figure 2: System with two oscillating masses ©W.Steiner 2015 [47]

A student solving this problem either takes a formula describing the spring constant from somewhere *or* (re-)constructs the model from scratch. In the latter case the forces exerted by springs, F_{c1} and F_{c2}, are determined by the space-time functions of the masses, $x_1(t)$ and $x_2(t)$, and by the spring contants c_1 and c_2:

$$F_{c1} = c_1 x_1,\ F_{c2} = c_2(x_2 - x_1),\ F_{c3} = c_1 x_2 \tag{1}$$

The forces exerted by the dampers (with damping factor d) involve a derivative:

$$F_{d1} = d\dot{x}_1,\ F_{d2} = d\dot{x}_2 \tag{2}$$

The law of dynamics, that mass times acceleration equals the sum of all forces, gives the following equations:

$$m\ddot{x}_1 = -F_{c1} + F_{c2} - F_{d1} \ \wedge\ m\ddot{x}_2 = -F_{c2} - F_{c3} - F_{d2} + F \tag{3}$$

And now a student has to operate formally (without any imagination for the involved elements !) and substitute (1) and (2) into (3) in order to get the equation (the further steps are shown later).

In these operations the two functions $x_i(t),\ i = 1..2$, and their respective derivatives $\dot{x}_i$ and $\ddot{x}_i$ are formal objects, with certain formal operations allowed upon them --- in order to do this, a student needs firm familiarity with formal representation of objects and operations --- while a notion of function as ‘‘pointwise application’’ (Pt.1 and Pt.2 on p.3) would interfere with understanding (and constructing) the model: A student needs a range in levels of abstraction wide enough to have the freedom to drop too concrete levels ob abstraction. Dropping details is **economy of thinking** required in dealing with complicated matter.

So a third aspect of the challenge is: *models can be complex such that understanding them requires to focus essential properties (most precisely represented formally) and to* drop details*.*

[4] If an abstract concept has only be learned by heart, this will turn out in later mathematics or engineering courses; however, in other studies such useless learning may remain undiscovered more often --- this might be a reason, why education in mathematics is critisised as unsuccessful more than in other subjects.

2 Technology for abstraction

The science of mathematics continuously develops higher levels of abstraction, and so do engineering sciences when applying mathematics. It took about two thousand years until mathematics was abstracted to formal logic, such that foundamental problems of mathematics, e.g. Russel's paradox, could be mastered reliably. However, it took not much more than a half of a century to mechanise formal logic in computers --- this is addressed in the subsequent section.

Systems that explain themselves

Theorem Proving (TP) on computers started with AutoMath [36] in 1967, i.e. as early as Computer Algebra. However, computation could be made usable easier than interactive proving. Nevertheless, currently TP begins to intrude into mathematics and specifically into Formal Methods [13] in engineering disciplines.

The subsequent discussion, how to understand TPs as ''systems that explain themselves'' uses the HOL family of TPs [27, 28, 11] as an example, in particular the TP Isabelle [40] as the conceptual and technological base of examples in this paper.

These TPs are **self-contained** in a very precise way: they comprise several layers of languages, where one layer justifies the next higher level on the firm grounds of formal logic. At the bottom there are the axioms of natural deduction [45].

So formulas justify formulas in TP and that is quite natural, it just reflects self-referentiality of human language as illustrated by an example, a calculation simplified in an obvious way:
$12345 \cdot 167 - 12345 \cdot 67 = 12345 \cdot (167 - 67) = 12345 \cdot 100 = 1234500$. These numbers can be considered as words and then the reader might discuss with someone else, why some specific usage of words is smart, or is correct or whatever --- in such a discussion the reader uses words to talk about other words (the numbers), so words are used to explain words.

TP does the same on a rigorous formal level, all the knowledge is *deductively complete*: each definition, lemma, theorem or proof is mechanically deduced from first principles (the axioms of logic).

For instance, in case the above calculation is done in TP, the numbers are represented as logical entities [5] and correctness is justified by applying theorems mechanically, for instance the theorem $a \cdot b - a \cdot c = a \cdot (b - c)$ for integers. And theorems like this one are proved correct mechanically in TP by using the laws of formal logics, which again are represented by formulas. So we have several levels of formulas: numbers, theorems, proofs and laws of logic, where one level formally justifies the next one.

In Isabelle mechanisation of mathematics [2] and of applied mathematics [1] is promoted by an increasing worldwide community at great speed; the coverage of mechanised mathematics exceeds the scope of introductory courses at engineering faculties already.

Some initiatives and prototypes

Since TP-based educational software still appears as an outsider, some initiatives and prototypes are briefly introduced here.

[5] In fact, in Isabelle clicking on ''$\cdot$'' leads to the definition of group, semigroup, etc. And clicking on ''*int*'' leads to the construction of integers from naturals by the equivalence relation $(a,b) \approx (c,d) \leftrightarrow a + d = b + c$, to the proofs that integers constitute a ring, etc.

IMPS [25] *''is an Interactive Mathematical Proof System intended to provide organizational and computational support for the traditional techniques of mathematical reasoning. In particular, the logic of IMPS allows functions to be partial and terms to be undefined. [. . .] One of the chief tools of IMPS is a facility for developing formal proofs. In contrast to the formal proofs described in logic textbooks, IMPS proofs are a blend of computation and high-level inference. Consequently, they resemble intelligible informal proofs, but unlike informal proofs, all details of an IMPS proof are machine checked.''*[24]

IMPS is definitely beyond prototype status, rather, it provided early contributions to include computation into TP, to adapt to the needs of mathematicians and to learning mathematics [23]. However, since IMPS emphasises formal proof, the system cannot address major parts in present STEM education.

E-Math builds upon a specific representation of applied mathematics [8] and in 2011 started an ambitious project in Baltic countries. *''A resource centre is set up that serves as an innovation and development platform for mathematics education and e-learning (the E-math center). The center provides support for teachers by distributing study material, organizing teachers education courses, and providing software tools. Experiences and best practices are shared and new methods and tools disseminated among the user community. The E-math center also provides a standardized platform for providing added value services, creating an attractive business market for mathematics, physics, computer science, and engineering education.''*[7]

The E-Math software uses the TP PVS [43] and plans to proceed to Isabelle. Since a major sponsor (Nokia) dropped out, the project's development slowed down.

Geometry Provers have surprisingly many implementations in prototypes for educational geometry systems[29, 35, 44] and in well established systems[30, 15], where the latter, GeoGebra, resides at University of Linz and is the software used worldwide most in mathematics education.

Geometry plays an important róle in many curricula: the intuitive representation of problems and the evidence of solutions is in contrast to formal mathematics. So geometry is considered most appropriate to introduce mathematical proof. In this line geometry provers, as tools relying on formulas, are hidden behind the scenes and try to prove conjectures stated by users.

Presently the situations is unsatisfactory, in particular, because the (automated) provers build upon axioms of geometry, which are not appropriate for students. Important work to provide axioms close to human intuition has been done by [48], respective implementation is pending.

The *Isac*-Prototype is a piece of software under construction, which implements all features addicted to TP-based software in this paper. The primary contribution to such systems is Lucas-Interpretation [37, 38]. Appropriateness to STEM education is discussed in [39] and the state of ongoing development can be retrieved from [3].

Abstraction as a process

§1 gave a list of levels in abstraction for the notion of ''function''. The respective points Pt.1..6 are not only a sequence of material presented one by one scattered over several courses, rather, the points involve a cumulative process of individual learning.

For instance, for ''function composition'' in Pt.2 a ''pointwise operation'' concept is still sufficient, where ''function'' is experienced as a statement of how to calculate *concrete* values (in the codomain) from *concrete* arguments. However, understanding the chain rule is difficult, if the ''pointwise operation'' concept is still predominant. Higher abstraction to a class of objects would be helpful; a

class, which contains a neutral element (the identity function), where not all operations are commutative, etc.

Theories of abstraction in mathematics education [34] confirm the example with ‘‘function’’: it begins with examples, with sensual experience by gestures --- and [33] says: *‘‘abstract[. . .] mathematical objects need to be linked to [. . .] empirical concepts if their learning is to be meaningful’’.*

Particularly instructive is the RBC-model [21], which identifies three observable epistemic actions constitutive for the emergence of a new abstract construct: *‘‘recognizing (R), building-with (B) and constructing (C). Recognizing refers to the learner realizing that a specific previous knowledge construct is relevant in the situation at hand. Building-with comprises the combination of recognized constructs, in order to achieve a localized goal such as the actualization of a strategy, a justification or the solution of a problem. The model suggests constructing as the central epistemic action of mathematical abstraction. Constructing consists of assembling and integrating previous constructs by vertical mathematization to produce a new construct. [. . .] the construct becoming freely and flexibly available to the learner [. . .] pertains to consolidation.’’.*

The long-term nature of increasing abstraction in the process of learning [26] is particularly stressed: *‘‘Consolidation is a never-ending process through which students become aware of their constructs.’’* [21].

Increasing abstraction is a crucial point of teaching. That assessment of students’ abstractions is difficult, that has been mentioned in §1: If teaching abstract objects and abstract properties in math (like in other disciplines) should not cause students to learn the teacher’s wording by heart, then there need to be a varity of abstractions: for instance adding to Pt.1..6 on p.3 permutations as ‘‘functions’’.

Another need is suggested by the psychological theories cited above, active operation done by students; this is discussed in the subsequent section.

Interactive operation fostering abstraction

In order to prepare students for independently solving problems in engineering, the crucial question is, how to create learning situations, where they actively manipulate mathematical objects, where they can work as indepentently as possible in order to ask their individual questions, where they get answered their questions, where trial&error learning is possible in an efficient way, which make students prepared to tackle comprehensive models as introduced in §1 eventually.

The above cited theory [21] emphasises *‘‘constructing as the central epistemic action of mathematical abstraction’’.*

This paper’s main point is, that ‘‘constructing’’ is most appropriately supported by a system, that guides students through stepwise problem solving, that suggests a next step if the student gets stuck.

And if such a system is a ‘‘system that explains itself’’, then all explanations are available, which can be gained from mechanical justification: each object is rigorously deduced from another one (finally from the axioms of formal logic), different language layers justify each other (mechanised problem specifications motivate problem solutions, theorems justify formula transformations, etc).

The latter kind of justification is illustrated by a `Solution` of the problem stated in §1:

```
21  Problem [determine, 2-mass-oscillator, DiffEq]:
211   Specification:
212   Solution:
2121                                                    forces of springs
2122     [F_c1 = c_1 x_1, F_c2 = c_2 (x_2 - x_1), F_c3 = c_1 x_2]
2123                                                    forces of dampers
```

2124	$[F_{d1} = d\dot{x}_1,\ F_{d2} = d\dot{x}_2]$
2125	mass times acceleration equals sum of all forces
2126	$[m\ddot{x}_1 = -F_{c1} + F_{c2} - F_{d1},\ m\ddot{x}_2 = -F_{c2} - F_{c3} - F_{d2} + F_2]$
2127	`Substitute` $[F_{c1}, F_{c2}, F_{c3}, F_{d1}, F_{d2}]$
2128	$[m\ddot{x}_1 = -c_1x_1 + c_2(c_2 - x_1) - d\dot{x}_1,\ m\ddot{x}_2 = -c_2(c_2 - x_1) - c_1x_2 - d\dot{x}_2 + F_2]$
2129	`Rewrite_Set` normalise
212a	$[m\ddot{x}_1 + d\dot{x}_1 + c_1x_1 - c_2(x_2 - x_1) = 0,\ m\ddot{x}_2 + d\dot{x}_2 + c_2(x_2 - x_1) + c_1x_1 = F_2]$
212b	switch to vector representation
212c	$\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ F \end{pmatrix}$
22	$\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \ddot{x} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \dot{x} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} x = \begin{pmatrix} 0 \\ F \end{pmatrix}$

The justifications in the above `Solution` (for the `Specification` see [39]) are shifted to the right margin. Some of them are concerned with geometric intuition, e.g. `2121..2122` or with laws of physics, e.g. `2123..2126`; [39] discusses how *Isac* deals with these.

Here we focus steps which are concerned with abstract operations, in particular the steps `2127 Substitute` and `2129 Rewrite_Set`: these steps operate on functions $x_1, x_2 \ldots \ddot{x}_1, \ddot{x}_2$ in a completely abstract way, while sensually based experience with the notion of ''function'' cannot help at all --- abstraction to objects with properties of ''algebraically closed field'' appears more helpful at this point [6].

So: How prepare students for such abstraction? How make them familiar with abstract notions? How foster students' confidence in operating on an abstract formal level?

This papers' answer is: Use TP-based ''systems that explains themselves'' from the very beginning, from calculations like the one introduced in §2, $12345 \cdot 167 - 12345 \cdot 67 = 12345 \cdot (167 - 67) = 12345 \cdot 100 = 1234500$. Firstly students will experience, that each formula transformation is justified by a theorem, e.g. the first step above by $a \cdot b - a \cdot c = a \cdot (b - c)$.

Later, in some other course, students will hear about groups; such abstraction might overwhelm a student initially, but if experiencing the same kind of formal justification again and again (always connected with the notion of ''groups'') students will get familiar with abstract formal operation. And an alert teacher won't miss opportunities to point at certain abstraction (all of which are available under the surface of a TP-based system) during his classes.

The reader also might note an example for ''economy of thinking'' in step `212b..212c`, the switch to vector notation in order to simplify the appearance of the differential equation. [47] goes even further and writes $M\ddot{x} + D\dot{x}_2 + Cx = F(t)$.

So: The long-term process of abstraction discussed in §2 and confirmed by theory, e.g. [26, 21], is most appropriately supported by self-contained systems, by ''systems that explains themselves''.

3 Expected impact of self-containedness

TP-based systems are self-contained as mentioned above --- is respective interpretation as ''systems that explains themselves'' justified? What can be expected for educational practice?

[6] The ultimate level of abstraction for a mathematical object is: their meaning, the semantics, is given by the formal operations applicable to the object and by the properties of these operations.

New standards for educational software

This first answer to the above questions clarifies, what one can expect from educational systems, which exploit the conceptual and technological strengths of TP. In particular coverage of essential parts in doing mathematics (see §2 on p.5) and ability to foster abstraction by interaction (see §2 on p.7) allow to postulate novel standards in software for STEM.

A student can rely on the system that wrong steps in construction of a problem solution are rejected [7]. This is a big step forward, since Computer Algebra Systems just deliver a final result without showing intermediate steps. And they are designed such, that they deliver some result anyway. If not possible, for instance, if there is no closed solution to an integral, it just returns the input; or it drops some solutions of an equation.

Such behaviour is appropriate for experts, who know what they do and what the limitations of their software tool is --- but it is inappropriate for educational software: a student wants to learn applied mathematics, doesn't know the system and wants to get reliable feed-back from the system, whether she or he is correct or not.

All phases of problem solving are covered as shown by the example with the wave equation --- this is a consequence of TP's coverage of mathematics' essentials; the following phases can be identified:

- The **modelling phase** translates a problem into formulas. The problem might be stated by a text, a figure (e.g. Fig.2) or other representations. The result of this phase are elements filled into `Given` and to `Find`; this involves understanding the problem (probably without knowing a solution, so far --- an important separation of concerns); so a TP-based system can help using data hidden behind the scenes. [39] postulates a graphical input tool for this kind of examples.

- The **specification phase** relates the data input so far to models (called `Problems` in the example) already present in a TP-based system. There are three dimensions of knowledge:
  - *Theories* define the notions involved in problem descriptions and in problem solutions; Isabelle's theories can be viewed at [1, 2] together with all proofs.
  - *Problems* capture the formal `Specifications`; one can expect respective domain-specific libraries in the future [14], which assist engineers in addition to libraries of methods..
  - *Methods* describe how to solve a problem. [38] describes a programming language, which works on Isabelle theories and problems ins the *Isac*-prototype..

  The specification phase deals with arrangement of the elements of the three knowledge dimensions and aligns sub-problems in a sequence [4].

- The **solve phase** finally creates the steps towards a solution of the problem; the steps include respective justifications, can be formulated as free as possible and can even be suggested by the TP-based system [38].

The phases are highly intertwined: modelling with specification, and the solve phase recursively enters modelling and specification for sub-problems.

All levels of abstraction are mechanised in a TP-based system. For instance, the wave equation as an abstract model for the two-mass-oscillator [47] allows to identify levels of abstraction as follows:

[7] The opposite cannot be guaranteed: that a correct step towards a solution is acknowledged by the system --- the system's knowledge might by limited such that a specific step cannot be derived from the logical context.

(1) An appropriate TP-based implementation can animate a dynamic model --- this is a concrete representation of the model (see for instance [32]): The student sees two masses oscillating, and probably gets an idea of the two modes of oscillation (oscillating in the sam direction -- system vibrates / oscillating in opposite directions -- non vibrating system) superimposing each other.
(2) At a first level of abstraction the student can consider numerical input and output of subproblems and experiment with different values.
(3) The next level of abstraction is given by the symbolic representation of the subproblems, e.g. the differential equation with coefficients reflecting attenuation according to spring constants.

These three levels of abstraction can still be covered by use of an Algebra System for certain subproblems. [39] shows how these levels are accessible in a TP-based system, in particular, how a student can switch between (2) and (3). However, the subsequently following levels are only accessible by TP-based systems.

(4) Open subproblems in order to see intermediate steps including justifications.
(5) Review the justifications given by laws of physics. The justifications are referenced mechanically, the reference can optionally contain multi-media explanations.
(6) Review the justifications given by theorems of mathematics. Theorems are accompanied with their proofs, so even these could be investigated.
(7) Herewith interests of engineers might by satisfied; a ‘‘pure’’ mathematician, however, might go further until the axioms of the underling logic are reached --- this is indeed possible with so-called LCF-style provers.

The major opportunity to learn efficiently is here, that students are free to switch from one level of abstraction to the other at any subproblem and at any time.

Decomposition and construction are available in parallel, another feature which fosters abstraction. According to §2 on p.6, abstraction involves: *recognising* details of knowledge, *building-with* identifies problems, methods and theories and *constructing* arranges them to a sequence of sub-problems solving the main problem eventually.

So, switching the perspective from details to survey and vice versa, is essential --- and this is best supported by a system, which represents a problem `Solution` as a tree (see p.7), where each branch can be collapsed or expanded.

Decomposition and construction are also available in a more literal sense: The system can calculate the problem solution automatically and the student can study the details (decomposition) or the student constructs all steps with the feed-back and help from the system (construction).

Concrete examples and abstract concepts are connected --- in a straight forward way: For instance, the calculation on p.7 comprises differentiation as a sub-problem. For an exercise in an introductory course the calculation can be done automatically by the system until differentiation begins; then the system enforces step-wise interactive solution of the sub-problem and upon successful completion of the sub-problem, the system continues automatic calculation until the final result.

Students can be satisfied, that they have solved the sub-problem (and accomplished the exercise); but they are also free to look into the róle of differentiation within the highly abstract concepts involved --- an early offer to consolidate abstraction (and an answer to the question: ‘‘What for do I learn differentiation?’’).

And when teaching the wave-equation, teachers need not repeat differentiation and other basics: the TP-based system is ready to support repetition --- as it is ready to consolidate abstraction at any occurrence of a concept.

Advances for teaching and learning

Novelty of software features induced by TP-technology will establish advantages for both, for teaching as well as for learning. One can expect, that details will become clearer the more of the user requirements [8] are implemented and the more educators use TP-based software in practice. But general trends are visible already now:

Teaching can take profit from the wider coverage of software support in conjunction with respective authoring tools (the latter not discussed here).

- *Balance between teaching and active learning:* Traditionally mathematics education emphasises practical exercises in company with lectures. This emphasis comes with a new flavour now: the wider coverage of software support (for all phases of problem solving etc.) allows lecturers to focus aspects, which never can be taught by machines.
- *Closer connection of introductory math with engineering application:* The flexibility in students' access to worked examples allows to design collections of advanced examples such that it can be used by both, by introductory courses in mathematics as well as in the advanced courses themselves.

Learning can take profit from the same software features with additional effects seen from students' side. Most helpful appears, that TP-based software can solve predefined problems in a way which supports both, survey as well as indepth study.

- *It is easy to repeat old stuff.* Exercises and worked examples are easier retrieved in an electronic system than on paper; and if revisited, old stuff can be investigated from different points of view: TP-based systems are self-contained.
- *It is easy to look forward to future stuff.* Students' questions ''What for do I learn that?'' can be answered any time by a mechanised search on the worked examples. Each abstract concept in mathematics can immediately be connected with relevant applications in engineering.
- *There is no need to understand all at once.* Models in engineering tend to be comprehensive such, that not all can be understood by students in one go, that context is lost when explanations are scattered over several lectures. TP-based software can include the essential explanations such, that these can be recalled by students any time.

Objections from STEM educators

Finally the most challenging objections are documented, which have been raised by educators when encountering *Isac* first time --- and answered in the form of ''freuqently asked questions''.

''TP-based systems will replace human teachers.''

Answer: No.

A ''system that explains itself'' justifies each step *mechanically*, so it can*not explain* the relevance (and not the beauty) of mathematics for humans in common and in person.

But TP-based system can be a medium to connect work of different teachers; see ''concrete examples and abstract concepts are connected'' on p.10. And they are a patient source for inquiry by students across courses and studies, and hopefully trigger challenging questions, which only can be answered by humans.

[8] *Isac*'s user requirements under construction at `https://hg.risc.uni-linz.ac.at/wneuper/isac-doc2`.

''TP-based systems are not needed in Mathematics for SE.''

Answer: Right, not yet.

This objection addresses an introductory course in Software Engineering (SE) which continues a series of courses [17, 19, 20] well established at RISC Linz/Hagenberg. And the answer given contains ''not yet'' for the following reason:

This introductory course aims to provide prerequisites for formal problem solving in a most general way. Formal aspects of ''thinking, speaking and writing'' in mathematics are taught, deepened in exercises and students' achievements are examined finally --- all tried and tested with good results since decades. So why change a successful course by introduction of a TP-based system? Thus ''no, *not* needed''!

But ''not *yet* needed'' as long as there are no advanced courses building upon the introductory course --- let's take the abstract concept of ''type'' as an example: This concept would be indispensable in an advanced course on Formal Methods [12], which would include practical exercises on abstract models compararable to those of §1 and §2. In such exercises students would encounter error messages like this one:

```
Type unification failed: Clash of types "_ list" and "_ -> _"
Type error in application: incompatible operand type
Operator:  foldr :: (int -> ??'a -> ??'a) -> int list -> ??'a -> ??'a
Operand:   ins :: int -> int list
```

Independent from what students have been taught in the introductory course, what exercises they did, some semesters later in the advanced course they might be unable to apply what they have learned, for instance unable to comprehend the above error message.

The issue with teaching and examining abstract concepts like ''type'' (comparable with ''function'' from §1; with consolidation as a long-term activity, see §2, supported by continuous re-visiting §2 under various aspects from §3) is, that students learn word by word what they have been taught, they repeat even the application in exercises and the examiner is satisfied. But that does not mean, that students are able to apply the abstract concept in another context (see [41] for this phenomenon with functions).

The concise support, which TP-based systems offer in this case, is connection of introductory courses with advanced courses, see ''concrete examples and abstract concepts are connected'' in §3.

''My explanations cannot be done better by a machine: *I hand out exercises together with (final) results to my students, they do the exercises and compare their results with my results, and if they cannot figure out how to get one of my results, they come to me and I explain.''*

A machine, i.e. TP-based software, cannot explain better than a human, right. But besides many possibilities to fail with explanations (students asks a question inappropriately, students don't admit that they still have not understood, explanations are given just for a specific case (bypassing generalisation), explanations do not match the level of abstraction the student has asked for, etc) there is a principal limitation on humans' explanations:

Explanations from human to human appeal to common sense, advocate examples and analogies which call for human imagination based on sensual perception --- however there are abstract concepts (e.g. ring, field, imaginary number, . . .), which cannot be related to sensual perception, which gain meaning only from formal operation on them --- and these operations are presented *better by a machine*. And if the machine lures into trial & error learning, experiences are gathered which foster abstraction eventually.

''A machine just justifying steps cannot teach abstraction.'' Right, there is no machine able to *teach* abstraction --- rather, TP-based systems support to *learn* abstraction:

The learning opportunities made available by a TP-based system is to do *many* steps and to re-do them, to switch between detail and survey, to switch between ‘‘decomposition and construction’’ and to change levels of abstraction according to §3 on p.9.

An analogy to learning chess appears might clarify th situation: chess software just does mechanically step by step --- but the learner’s brain works on a much higher level: it compares, combines, concludes from the steps observed and builds up individual experience, which prepares for better playing chess eventually.

‘‘A machine induces senseless drill & practice’’. This objection usually is caused by two kinds of confusion:

The first one confuses old-fashioned software for drill & practice with TP-based systems; so this paper should have clarified this kind of confusion

The second one confuses interactive operation on mechanised mathematics with reminiscences of mathematics education critisised by [46], when students copied problem solutions from blackboard to their notebook, learned the algorithms by drill & practice and all that learning happened without understanding --- and now operating on mathematics mechanised in computers appears as bad as it was in these former times.

However, TP-based systems can request students to actively do steps in problem solving by adaptive user-guidance [38], such a system has more knowledge under the surface, than an average student can investigate, and a student can ask for the underlying knowledge at any step. So TP-based systems are tools appropriate for projects like [5, 6, 9] and meet the recommendations given by [46].

4 Conclusions

The paper is summarised as follows: Models in engineering science, for instance the wave equation, are composed from physical and mathematical concepts. The concepts and their composition are comprehensive sucht that many details need to be dropped in order to comprehend them (‘‘economy of thinking’’ §1).

In particular, details concerning sensual experience and concrete representation need to be dropped in order to get the essential properties into focus. While the concrete details are emphasised in early phases of education and at lower levels of abstraction, essential properties are most clearly captured by formal expressions --- and these are hard to comprehend for students (‘‘abstraction as a process’’ §2).

This paper proposes a novel approch to foster better understanding of comprehensive models in engineering science: Provide a new generation of educational software (‘‘systems that explain themselves’’ §2), which represents problem solving close to traditional paper&pencil work, but underneath has all knowledge available in high abstraction such that all mathematical objects are deduced from first principles (i.e. the axioms of natural deduction).

Given complete deductive justification, such systems explain themselves. If used from the beginning in problem solving, these systems continuously offer students to experience abstraction at a high level, which can be inspected on demand.

And these systems provide guided interaction in problem solving such that students can gain familiarity with formal abstraction on their individual ways and at their own pace --- as a preparation for better understanding comprehensive models in engineering sciences eventually (‘‘interactive operation fostering abstraction’’ §2).

Conclusions for immediate action are: Stakeholders in engineering education on the one side, and stakeholders in (Computer) Theorem Proving (TP) on the other side, are invited to joint projects

realising the promises addressed in this paper.

References

[1] Archive of Formal Proofs. http://afp.sourceforge.net. URL looked up Sep.2016.

[2] Generic proof assistant ‘‘Isabelle’’. http://isabelle.in.tum.de/. Knowledge built from Higher-Order Logic (HOL). URL looked up Sep.2016.

[3] Isac-project. http://www.ist.tugraz.at/isac/History. URL looked up Sep.2016.

[4] Slide movie on specifying sub-problems and determining a sequence. http://www.ist.tugraz.at/projects/isac/publ/movie-sub-problems.pdf. URL looked up Sep.2016.

[5] (2002). Pathway, the pathway to enquiry based science teaching. http://pathway.ea.gr/, URL looked up Jul.2016.

[6] (2014). Creative little scientists. http://www.creative-little-scientists.eu/, URL looked up Jul.2016.

[7] Back, R.-J. E-math, improving competence in mathematics using new teaching methods and ICT. http://www.imped.fi/e-math/. URL looked up Jul.2016.

[8] Back, R.-J. and Wright, J. v. (2006). *Mathematics with a Little Bit of Logic: Structured Derivations in High-School Mathematics*. Manuscript.

[9] Baptist, P. SINUS - a trademark for improving mathematics and science education in Germany. http://www.sinus-transfer.de/. URL looked up Jul.2016.

[10] Baptist, P. and Raab, D. (2007). SINUS Transfer. Steigerung der Effizienz des mathematisch-naturwissenschaftlichen Unterrichts. Tech. rep., Zentrum zur Förderung des mathematisch-naturwissenschaftlichen Unterrichts, Universität Bayreuth. http://www.sinus-transfer.de/fileadmin/MaterialienBT/sinus-transfer.pdf.

[11] Bertot, Y. and Casteran, P. (2004). *Coq’Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Series. Springer Verlag. ISBN: 978-3-540-20854-9.

[12] Bjørner, D. (2006). *Software Engineering*, vol. 1,2,3 of *Texts in Theoretical Computer Science*. Springer, Berlin, Heidelberg.

[13] Bjørner, D. (2009). *Domain Engineering. Technology Management, Research and Engineering*, vol. 4 of *COE Research Monograph Series*. JAIST Press, Nomi, Japan.

[14] Bjørner, D. (2013). *Domain Science and Engineering as a Foundation for Computation for Humanity*, chap. 7, 159--177. Francis & Taylor. In: J.Zander & P.J.Mosterman (eds.), Computational Analysis, Synthesis, and Design of Dynamic Systems.

[15] Botana, F., Hohenwarter, M., Janičić, P., Kovács, Z., Petrović, I., Recio, T. and Weitzhofer, S. (2015). Automated theorem proving in GeoGebra: Current achievements. *Journal of Automated Reasoning*, 55(1):39--59. doi:http://dx.doi.org/10.1007/s10817-015-9326-4.

[16] Breitschuh, J., Sonnenschein, E., Fuchs, J. and Albers, A. (2016). Fachliches Problemlösen in der Maschinenkonstruktion --- Untersuchung von Struktur und Erlernbarkeit mittels multimodaler Technikmodelle. *Journal of Technology Education (JOTED)*, 4(2):212--232. http://www.journal-of-technical-education.de/index.php/joted/article/view/86/100.

[17] Buchberger, B. Thinking, Speaking, Writing. Lecture Notes. RISC-Linz, Austria.

[18] Buchberger, B. (2005). Mathematik: Die Kunst des effektiven Handelns. Invited talk at MathSpace, Wien.

[19] Buchberger, B. and Lichtenberger, F. (1981). *Mathematics for Computer Science I - The Method of Mathematics (German)*. Springer, Berlin, Heidelberg, New York. Second Edition.

[20] Buchberger, B. and Lichtenberger, F. (1981). *Mathematik für Informatiker I.* Springer-Verlag Berlin Heidelberg New York.

[21] Dreyfus, T. (2012). Constructing abstract mathematical knowledge in context. In: *12th nternational Congress on Mathematical Education*. Seoul, Korea. http://www.icme12.org/upload/submission/1953_f.pdf.

[22] El-Demerdash, M., Lealdino, P. and Mercat, C. (2016). The effectiveness of kinesthetic approach in developing mathematical function graphs recognition and understanding at university level. Presentation at CADGME'16, http://mc2-project.eu/.

[23] Farmer, W. (2000). A proposal for the development of an interactive mathematics laboratory for mathematics eduction. In: *CADE-17 Workshop on Deduction Systems for Mathematics Education*, 20--25.

[24] Farmer, W. M. Imps, interactive mathematical proof system. http://imps.mcmaster.ca/. URL looked up Jul.2016.

[25] Farmer, W. M., Guttman, J. D. and Fábrega, F. J. T. (1996). **IMPS**: An updated system description. In: McRobbie, M. and Slaney, J. (Eds.), *Automated Deduction---CADE-13*, vol. 1104 of *Lecture Notes in Computer Science*, 298--302. Springer-Verlag.

[26] Ferrari, P. (2003). Abstraction in mathematics. *Phil. Trans. R. Soc. Lond.*, 358(1435):1225--1230.

[27] Gordon, M. and Melham, T. (1993). *Introduction to HOL: a theorem proving environment for higher order logic*. Cambridge University Press.

[28] Harrison, J. (2009). Hol light: An overview. In: *Proceedings of TPHOLs*, vol. 5674 of *LNCS*, 60--66. Springer.

[29] Janičić, P. (2006). GCLC --- a tool for constructive euclidean geometry and more than that. In: *Mathematical Software -- ICMS 2006*, 4151, 58--73.

[30] Kortenkamp, U. and Richter-Gebert, J. (2007). *The Interactive Geometry Software Cinderella.2*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[31] Kühnel, W., Bandelt, H.-J., Jahnke, T., Klein, H.-P., Remus, D., Schweighofer, M., Sonar, T., Spindler, M. and Walcher, S. (2015). Zur neuen Schulmathematik im Abitur --- Die Bildungsstandards der MKM von 2012. *Mitteilungen der DMV*, 23:106--109.

[32] Lichtblau, S. Motion of two masses connected by springs. http://demonstrations.wolfram.com/MotionOfTwoMassesConnectedBySprings/. Wolfram Demonstrations Project. URL looked up Sep.2016.

[33] Mitchelmore, M. and White, P. (2004). Abstraction in mathematics and mathematics learning. In: *Proceedings of the 28th Conference of the International Group for the Psychology of Mathematics Education*, vol. 3, 329 -- 336. https://www.emis.de/proceedings/PME28/RR/RR031_Mitchelmore.pdf.

[34] Mitchelmore, P., M.C.and White (2012). Abstraction in mathematics learning. In: *Encyclopedia of the sciences of learning*, 31--33. Springer, New York. doi:10.1007/978-1-4419-1428-6_516. http://www.merga.net.au/documents/MERJ_19_2_Editorial.pdf.

[35] Narboux, J. (2007). GeoProof: A user interface for formal proofs in geometry. In: *MathUI 07*. Linz, Austria. URL https://hal.inria.fr/inria-00495958.

[36] Nederpelt, R. P., Geuvers, J. H. and de Vrijer, R. C. (1994). *Selected Papers on Automath*. Vol. 133 of Studies Logic. Elsevier, Amsterdam. ISBN 0-444-89822-0.

[37] Neuper, W. (2012). Automated generation of user guidance by combining computation and deduction. In: Quaresma, P. and Back, R.-J. (Eds.), *Electronic Proceedings in Theoretical Computer Science*, vol. 79, 82--101. Open Publishing Association. doi:10.4204/EPTCS.79.5. http://eptcs.web.cse.unsw.edu.au/paper.cgi?THedu11.5.

[38] Neuper, W. (2016). Lucas-interpretation from users' perspective. In: *Joint Proceedings of the FM4M, MathUI, and ThEdu Workshops, Doctoral Program, and Work in Progress at the Conference on Intelligent Computer Mathematics*, 83--89. Bialystok, Poland. http://cicm-conference.org/2016/ceur-ws/CICM2016-WIP.pdf.

[39] Neuper, W. (2016). Rigor of TP in educational engineering software. In: *Joint Proceedings of the FM4M, MathUI, and ThEdu Workshops, Doctoral Program, and Work in Progress at the Conference on Intelligent Computer Mathematics*, 90--95. Bialystok, Poland. http://cicm-conference.org/2016/ceur-ws/CICM2016-WIP.pdf.

[40] Nipkow, T., Paulson, L. and Wenzel, M. (2002). *Isabelle/HOL --- A Proof Assistant for Higher-Order Logic*, vol. 2283 of *LNCS*. Springer. doi:10.1007/3-540-45949-9.

[41] Nitsch, R., Fredebohm, A., Bruder, R. et al. (2015). With functions in secondary mathematics education --- empirical examination of a competence structure model. *Int. J. of Sci. and Math Educ.*, 13(657). doi:10.1007/s10763-013-9496-7.

[42] OECD (2013). PISA 2015, draft mathematics framework. Draft subject to possible revision after the field trial, OECD, 2, rue Andre Pascal 75775, Paris, Cedex 16, France. https://www.oecd.org/pisa/pisaproducts/Draft%20PISA%202015%20Mathematics%20Framework%20.pdf [Retrieved 2.Jul.2016].

[43] Owre, S., Rajan, S., Rushby, J., Shankar, N. and Srivas, M. (1996). PVS: Combining specification, proof checking, and model checking. In: Alur, R. and Henzinger, T. (Eds.), *Computer-Aided Verification*, 411--414. CAV'96.

[44] Quaresma, P. and Santos, V. (2014). Visual geometry proofs in a learning context. In: *Proceedings of the 4th International Workshop on Theorem Proving Components for Educational Software*, 1--7.

[45] Riser, J. (1970). Gentzen Gerhard. Investigations into logical deduction. english translation of 4422 by szabo m. e.. american philosophical quarterly, vol. 1 (1964), pp. 288-306, and vol. 2 (1965), pp. 204-218.bernays paul. introduction. therein, vol. 1, p. 288. *The Journal of Symbolic Logic*, 35:144--145. doi:10.1017/S0022481200092604. URL http://journals.cambridge.org/article_S0022481200092604.

[46] Rocard, M. and al. (2007). Science education NOW: A renewed pedagogy for the future of europe. Tech. rep., European Communities, Directorate-General for Research, http://ec.europa.eu/research/science-society/document_library/pdf_06/report-rocard-on-science-education_en.pdf.

[47] Steiner, W. (2015). Vorlesungsskriptum Technische Mechanik III. FH OÖ, Fakultät für Technik und Umweltwissensschaften.

[48] Stojanovic-Durdevic, S., Narboux, J. and Janičić, P. (2015). Automated generation of machine verifiable and readable proofs: A case study of tarski's geometry. *Ann. Math. Artif. Intell.*, 74(3-4):249--269. doi:10.1007/s10472-014-9443-5. URL `http://dx.doi.org/10.1007/s10472-014-9443-5`.

[49] Zimmermann, K., Heusgen, K., Möller, C. and Zupanic, M. (2007). Studiengangsbezogene Dropoutanalysen Konzeption, Ergebnisse und Empfehlungen für die Technische Universität Dortmund. Tech. rep., Technische Universität Dortmund, Hochschuldidaktisches Zentrum, Vogelpothsweg 78, 44227 Dortmund.

Author

Walther A. Neuper, Technische Universität Graz, Austria, `wneuper@ist.tugraz.at`

Acknowledgement

The following experts from the Universities of Applied Sciences in Wels, Hagenberg and Salzburg are involved in current requirements engineering: Stephan Dreiseitl, Günther Eibl, Klaus Schiefermayr, Wolfgang Steiner and Stefan Sunzenauer. The author owes these persons a great debt of gratitude for their precious time spent, the novel ideas contributed to the design of *Isac* and the discussions which deepened insight into general features of TP-based systems as presented in this paper.