Informal and Formal Representations in Mathematics

Manfred Kerber[i] and Martin Pollet[ii]

[i]School of Computer Science
University of Birmingham
Birmingham B15 2TT, England
www.cs.bham.ac.uk/~mmk

[ii]Fachbereich Informatik
Universität des Saarlandes
66041 Saarbrücken, Germany
www.ags.uni-sb.de/~pollet

Abstract. In this paper we discuss the importance of good representations in mathematics and relate them to general design issues. Good design makes life easy, bad design difficult. For this reason experienced mathematicians spend a significant amount of their time on the design of their concepts. While many formal systems try to support this by providing a high-level language, we argue that more should be learned from the mathematical practice in order to improve the applicability of formal systems.

It is not easy to say precisely what we learnt in the meantime, if we learnt anything at all. I believe that the most important results are:
1. *we need experiments with much more advanced mathematics than already done*
2. *a system for practical formalization of mathematics probably will not be a simple system based on small number of primitive notions*
3. *integration with a computer algebra systems may be necessary or at least a feasible system must have bigger computational power.*

Andrzej Trybulec [33]

1 Introduction

There is an old debate on the foundations of mathematics, which goes at least back to the time when Alfred North Whitehead and Bertrand Russell did their epochal work, which also resulted in a much narrower view on what "foundations" of mathematics should mean. Russell articulated the idea in 1903 in the *Principles of Mathematics* [29]: to reduce mathematics to formal logic. He and Whitehead carried it through in the famous *Principia Mathematica* [35]. Russell was at this time also in inspiring discussions with Ludwig Wittgenstein and strongly acknowledges Wittgenstein's contribution to his thoughts. He writes in [30] (quoted from the reprint in [31, p.178]):

As I have attempted to prove in The Principles of Mathematics*, when we analyse mathematics we bring it all back to logic. It all comes back to logic in the strictest and most formal sense.*

Although there is evidence that Wittgenstein shared Russell's view, he later took the opposite stance and attacked Russell's approach. In particular he discusses the important notion of proof (quoted from [36, p. 143]):

> *'A mathematical proof must be perspicuous.' ... I want to say: if you have a proof-pattern that cannot be taken in, and by a change in notation you turn it into one that can, then you are producing a proof, where there was none before.*

One of the reasons why the Principia is so difficult to read is that the main ideas of the proofs are no longer visible in very long and very detailed proofs. Wittgenstein continues (p. 176f) to question the idea to try to reduce everything to a very small number of primitives (had the resolution calculus already been invented at that time his attack might have been to try to reduce everything to one single rule):

> *Mathematics is a* MOTLEY *of techniques of proof. – And upon this is based its manifold applicability and its importance. ...*
> *Now it is possible to imagine some – or all – of the proof systems of present-day mathematics as having been co-ordinated in such a way with one system, say that of Russell. So that all proofs could be carried out in this system, even though in a roundabout way. So would there then be only the single system – no longer the many? – But then it must surely be possible to shew of the* one *system that it can be resolved into the many. –* One *part of the system will possess the properties of trigonometry, another those of algebra, and so on. Thus one can say that different techniques are used in these parts.*

He continues then to counter the argument that Russell and Whitehead have constructively shown the possibility to reduce everything to one single system (p. 185)[1]:

> *If someone tries to shew that mathematics is not logic, what is he trying to shew? He is surely trying to say something like: – If tables, chairs, cupboards, etc. are swathed in enough paper, certainly they will look spherical in the end.*
> *He is not trying to shew that it is impossible that, for every mathematical proof, a Russellian proof can be constructed which (somehow) 'corresponds' to it, but rather that the acceptance of such a correspondence does not lean on logic.*

Who is right? On the one hand, Whitehead and Russell could claim that in more than a hundred years no significant counterexample to their original claim has been found.[2] Wittgenstein, on the other hand, could claim that mathematical practice is far from the logicistic approach.

[1] By the way, Gödel's proof that formal systems like the Principia are necessarily incomplete is irrelevant for this argument, since not only the Principia but any other powerful system suffers from the same problems.

[2] Gödel's incompleteness theorem again could be considered as one, but seems for this discussion less relevant than it looks on first view.

As a good cabinet maker spends a lot of time on the design of tables, chairs, and cupboards, a good mathematician spends great care on the design of their concepts. We will discuss this in more detail in Section 3. Before we do that we want to clarify what we mean by "design", in Section 2. In Section 4 we discuss some of the most important approaches in formal reasoning systems to provide adequate design possibilities.

A lot of what we say in the following will look completely self evident to many. However, we feel that in the traditional theorem proving community, which we consider as our home community, the logicistic view has too strongly dominated ever since its start more than 50 years ago. A seemingly strong argument against new approaches in theorem proving has been of the type "Everything you can do in your system X, I can do in Y," where Y stands typically for standard first-order logic. This kind of thought is so strong that proponents of the conventional approach to theorem proving seem to fail to even understand why this argument – albeit it may be true – is unhelpful and misleading.

For instance, Pat Hayes said in 1974 ([12] quoted from [5, p.18]):

> *A more recent attack on conventional theorem-proving ... is that it is too concerned with "machine-oriented" logic, and not enough with "human oriented" logic. I confess to being quite unable to understand what this could possibly mean.*

In this paper we try to clarify why the argument, although it may be technically correct, is pragmatically flawed. The argument is pragmatically wrong, since it is meant to say "Your system X is redundant and uninteresting, since we have system Y already, which suffices for everything you want to do." Since this argument was widely accepted the focus in the field was set much too narrow on the study of foundational systems. For other communities, our observations in this paper may be trivial.

Of course there are exceptions and we claim in no way that we are the first to have a look at this relationship of mathematical practice and fundamental systems. We cannot give a comprehensive overview of this type of work here but we want to mention some work in this direction, which we think provides very important starting points to the support of the design of mathematical concepts. In AUTOMATH, N.G. de Bruijn developed the idea of a mathematical vernacular [6], which should allow to write everything mathematicians do in informal reasoning, in a computer assisted system as well. In this tradition, Hugo Elbers looked in [10] at aspects of connecting informal and formal reasoning, in particular the integration of computations in formal proofs. Francis Jeffry Pelletier [24] as well as Henk Barendregt and Arjeh Cohen [2] discuss related philosophical questions on the nature of proof. Proof planning [7] in general can be viewed as an attempt to simulate informal reasoning (and integrate it with formal reasoning). Following this paradigm, Alan Bundy made first steps towards a Science of Reasoning [8], which goes beyond a narrow focus on a particular calculus. Ursula Martin took in [19] a close look at the mathematical practice and its relationship to computer algebra and computer-assisted reasoning. Michael Beeson presented in [3] a system that combines deductive and computational reasoning steps. He proposed to use the mathematical standard for checking the correctness of proofs generated by the system, namely peer reviews. Claus Zinn looked at the understanding of informal mathematical discourse [37].

We are not aware, however, of work in which the design process in mathematics is compared to the design possibilities of computer based mathematical support systems. In this paper, we look at design problems, which current automated reasoning systems suffer from.

We try to exemplify in the rest of the paper why these matters are crucial for automated theorem proving systems. One important issue is that of acceptance among mathematicians. Current automated theorem provers do not find wider acceptance among mathematicians, while computer algebra systems do. The initial investment that a mathematician has to do before he/she gets any benefit from a theorem prover is still quite high. This is different for computer algebra systems, partly since in many computer algebra systems things are *as they should be, as an inexperienced but mathematically educated user would expect them to be.* That is, these systems are typically *well-designed.* In theorem proving systems they are too often *not as they should be, not as an inexperienced user would them expect to be.* We will argue that this is *not* just a deficiency of the user interface, but that the problem with automated theorem provers is much deeper, it goes to the core of these systems, namely to the formal representation of mathematical knowledge and the reasoning that can be performed with this knowledge.

2 Good and Bad Design

Before we look at representation issues in mathematics we want to widen our point of view and have a more general look at design issues. Donald Norman gives a fascinating introduction into "The Design of Everyday Things" [22]. As the title of his book says, it is mainly on everyday things such as light switches or door handles. Norman gives impressive insights why when we face difficulties in everyday situations this is often due to bad design; and conversely when we do not face such difficulties this is often due to good design. His insights are of a very general nature and we will discuss in the next section that the principles for good design hold in mathematics as well and form another facet of mathematics which go in some aspects beyond logic.

Let us look at a simple example how bad design can make life difficult. Very often light switches are carelessly installed so that in a particular situation it is not possible to intuitively know which one to use. For instance, in the office of one of the authors there are two light switches next to each other, the left one for the upward directed light and the right one for the downward directed light, or is it the other way around? Since nobody can guess or remember this, the switches are labelled "up" and "down". Unfortunately these labels cannot be read in the darkness before you switch the light on. This is bad design. A good design for the problem would be to locate the two switches not next to each other but on top of each other, the upper one for the up-light, the lower one for the down-light. A similarly bad design is used in almost all cookers with four hot plates arranged in a 2×2 form. The four switches belonging to the four hot plates are arranged in a straight line so that labels are necessary to remember which of the two left (or right) hot plates are controlled by which of the two left (or right) switches. If the switches for the back plates were only slightly moved up out of the straight line, the problem would go.

Although Norman does not relate design issues to mathematics, most of his observations can be translated to a mathematical context. For instance, design principles allow us to answer questions like "Why and how do we find a proof without major search effort, although it is a difficult one and we do not know it?" Norman states four principles of good design which help us to get certain things right, although we do not know precisely what to do [22, p.55]: *"Precise behavior can emerge from imprecise knowledge for four reasons.*

1. *Information in the world. Much of the information a person needs to do a task can reside in the world. Behavior is determined by combining the information in memory (in the head) with that in the world.*
2. *Great precision is not required. Precision, accuracy, and completeness of knowledge are seldom required. Perfect behavior will result if the knowledge describes the information or behavior sufficiently to distinguish the correct choice from all others.*
3. *Natural constraints are present. The world restricts the allowed behavior. The physical properties of objects constrain possible operations: the order in which parts can go together and the ways in which an object can be moved, picked up, or otherwise manipulated. Each object has physical features – projections, depressions, screwthreads, appendages – that limit its relationship to other objects, operations that can be performed to it, what can be attached to it, and so on.*
4. *Cultural constraints are present. In addition to natural, physical constraints, society has evolved numerous artificial conventions that govern acceptable social behavior. These cultural conventions have to be learned, but once learned they apply to a wide variety of circumstances."*

Norman [22, p.188f] develops in the sequel seven principles of (good) design for transforming difficult tasks into simple ones:

1. *Use both knowledge in the world and knowledge in the head.*
2. *Simplify the structure of tasks.*
3. *Make things visible: bridge the gulfs of Execution and Evaluation.*
4. *Get the mappings right.*
5. *Exploit the power of constraints, both natural and artificial.*
6. *Design for error.*
7. *When all else fails, standardize.*

Following such principles it will be possible to design light switches and door handles well. Without doubt good designers follow such principles, but not only in their relationship to everyday objects, but also in their relationship to mathematical objects. In the next section we will have a closer look at one example where careful design of a mathematical entity helps avoiding errors, so-called multiplication tables. We will also briefly mention further examples.

3 Design and Representation in Mathematical Practice

We will take a closer look at multiplication tables, a concept that seems on a first view easy, and on a second difficult to model in existing theorem proving systems. Multiplication tables are part of rigorous mathematics in the sense that they appear not only as comments or illustrations in mathematical textbooks, but are usually introduced in definitions and their properties are stated as theorems.[3] We believe that the features of the concept "multiplication table" as well as those we present in the sequel are not only a matter of presentation but that they are used to encode and retrieve information about mathematical concepts in an efficient way and that they ease the actual process of finding and presenting proofs. To represent a mathematical object such as a multiplication table requires careful design.

3.1 Multiplication Tables

Multiplication tables were introduced by Cayley to represent the operation of finite abstract groups. The information encoded in a table is that the operation is a binary operation, defined on $\{d_1, \ldots, d_n\} \times \{d_1, \ldots, d_n\}$ and with range $\{c_{11}, \ldots, c_{nn}\}$, the operation is discrete and has a finite domain and codomain.

$$\begin{array}{c|ccc} \circ & d_1 & \cdots & d_n \\ \hline d_1 & c_{11} & \cdots & c_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ d_n & c_{n1} & \cdots & c_{nn} \end{array}$$

The table has its own notion of well-formedness, that is, all d_i have to occur and have to be different, the table must be fully filled. Here you find Norman's principles 1, 3, and 4. Multiplication tables are designed in a way that their structure puts "information in the world" that makes it difficult to violate well-formedness. It is hard to imagine when you have got the task to define a specific operation starting with an empty multiplication table that you forget a case, since that would leave a hole in the structure. The table itself is of the form that it constrains the possibilities. For instance, it is impossible to enter more than one entry per field. This prevents any over-specification of $\circ$. Furthermore, although the order of the d_i in the columns and rows could in principle be different, cultural conventions prevent that.

Note that there are particular reasoning methods connected to the representation. To check the basic property of closedness, one has to go through the elements of the table and check whether for all elements holds $c_{ij} \in \{d_1, \ldots, d_n\}$. The commutativity of $\circ$ is checked by verifying that the table is symmetric with respect to the diagonal. This intuitive form of reasoning depends on the cultural convention to use the same order for rows and columns. Another cultural convention is to write a (potential) unit element as first element (or second element in the presence of a zero, which typically goes in the first place). With this convention, it is checked that d_1 is a unit element by establishing the equality of the columns under $\circ$ and d_1 and the rows right to $\circ$ and d_1. Inverse elements can be checked by establishing that each column and each row contain the neutral element exactly once. These reasoning patterns follow partly the design principle number 2, since they are nat-

[3] We use here the word "rigorous" and not "formal" in order to distinguish it from "formal logic" and mechanical systems. Mathematicians would probably use the word "formal," since they are happy with these concepts as a level of formalization that clarifies concepts unambiguously.

ural and easy to reconstruct. From the group properties only associativity requires a logic level proof.[4]

Of course, it is possible to define the same operation in a logic, possible formalizations are for example:

- First order: extend the signature by a function constant $\circ$ and add the assumptions $d_1 \circ d_1 = c_{11} \wedge d_1 \circ d_2 = c_{12} \wedge \ldots \wedge d_n \circ d_n = c_{nn}$.
- Higher order: use the description operator[5] to define the operation as $\circ \equiv \lambda x. \iota y. (x = (d_1, d_1) \wedge y = c_{11}) \vee \ldots \vee (x = (d_n, d_n) \wedge y = c_{nn})$.

While the special representation of multiplication tables can be translated into these general logical formalisms, parts of the information stored in the mathematical representation are lost. In the first case the compoundness of the table representation is hard to reconstruct. We are speaking about a set of equations suitable for equational reasoning steps, but to recognize that the set of equations is suitable for an abstract method for closedness or commutativity as mentioned before is not so obvious. Also the special reasoning methods for proving commutativity, inverse elements, and neutral element are not so obvious, but require search in a set of formulae. From a human interface point of view, the lack of structure in the formulae puts the burden of guaranteeing well-definedness on the human. He or she has to be careful not to forget the definition of one element or not to over-define one expression by inserting two formulae like, for instance, $d_1 \circ d_1 = a$ and at a different place $d_1 \circ d_1 = b$.

Although, the higher order formalization of the operation seems preferable over the first order one since it encodes $\circ$ as one compound object, here as well it is hard to recognize what kind of function is encoded. Actually, there is a proof obligation to be shown in an application of the function to arguments, namely that there really is a unique element with these properties.

In the rest of this section we look at further examples for mathematical concepts and procedures which are difficult to represent directly in a foundational system. Although we do not relate them in detail to the design principles discussed in the previous section, it would not be hard to establish similar relationships here as well.

3.2 Representations Adapted to Objects

Similarly to multiplication tables there are other objects which typically have specialized representations. Quite related are matrices. A matrix is a two dimensional array that may contain numbers or more complex objects. It has similarities with multiplication tables and a possible formal definition as lists of lists is the same as the one of multiplication tables. However its usage is very different, and human

[4] Surely, for people experienced with this type of proof, there is not anything to prove anymore, but it boils all down to trivial computations, which according to the Poincaré principle [2, 15] can be considered as not being a part of the proof. This is different for beginners, whose perspective we have taken here.

[5] The description operator ι returns the element of a singleton. $\iota y.P[y]$ denotes the unique element c such that $P[c]$ holds, if such a unique element exists.

mathematicians have different methods attached to these concepts. Matrices are typically used to represent linear mappings and other transformations in vector spaces. The equation[6]

$$\left(\begin{array}{c|ccc} \alpha & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & T^{-1} & \\ 0 & & & \end{array}\right) \cdot \left(\begin{array}{c|c} 1 & uT \\ \hline 0 & \\ \vdots & BT \\ 0 & \end{array}\right) = \left(\begin{array}{c|c} \alpha & \alpha uT \\ \hline 0 & \\ \vdots & T^{-1}BT \\ 0 & \end{array}\right)$$

is taken from a proof about the tridiagonalization of matrices.

In principle matrices over a field F can be represented in logic by functions from an index set into F. However, this representation does not lend itself to the definition of multiplication of matrices in which product elements are calculated by traversing the left matrix left-right and the right matrix of the product top-down. The generalization of this principle makes it easy to establish the relationship accounted for above. Note that this form of abstraction is not just a matter of analogy which helps to find a proof, since it would be quite difficult to state – let alone prove – the property above without abstraction. Furthermore the explicit matrix representation exhibits advantages mentioned above for multiplication tables. We have presented some work on this in [25], which was extended in [32].

Note that matrices are available in Computer Algebra Systems (CASs) as primitives and that direct manipulations are possible. This, however, does not make it obsolete to introduce them directly into automated theorem proving systems as well. For instance, the matrices used in the equation above cannot be easily defined in a CAS, since they represent more than one instance, they are generalized matrices representing any matrix that has the same 'form.' Establishing the equation itself cannot be done by computation, but requires reasoning. Once established it can become a further computation rule.

3.3 Dynamic Representations

A feature of mathematical representations is that they are dynamic in the sense that as new knowledge is available the basic representation of objects may change. For instance, the existence of inverse elements of a *group* G can be formalized by $\forall_{x\in G}\exists_{y\in G}\ x \circ y = e \wedge y \circ x = e$ with the unit element e. Since for each group element there exists exactly one inverse element, the inverse of an element x is usually denoted with the help of a function as $\mathit{inv}(x)$. Whereas the introduction of a function denoting the inverse elements is possible in most of the interactive theorem proving systems, the question whether the concept group should be introduced as $\mathit{group}(G, \circ)$, $\mathit{group}(G, \circ, e)$, $\mathit{group}(G, \circ, e, \mathit{inv})$ seems to be more subtle.

The first formalization eases the possibility to inherit properties of subsumed concepts with $\mathit{group}(G, \circ) \Rightarrow \mathit{monoid}(G, \circ)$. The latter formalization makes the unit elements and inverse elements directly accessible but already contains the

[6] Mathematicians store information even in the letters they choose for their objects. Even without further information it is relatively easy to reconstruct that T^{-1}, B, and T should represent submatrices since they make use of upper-case letters, while the Greek α stands for a scalar, and u denotes a vector.

uniqueness of the inverse element of an element in form of the inverse function. The process of the actual exploration of basic principles of group theory that starts with the tuple $(G, \circ)$ and later introduces the neutral and inverse elements as useful parameters of the concept can hardly be modelled in current automated reasoning systems. Rather it is necessary to choose one formalization and to stick to it. This way it is not only the case that the introduction of a concept cannot be modelled adequately, but perhaps more seriously re-representations of concepts are not supported. However, while one representation may be best suited for one task, it may turn out to be unsuitable for a different one. In the latter situation mathematicians change representations. Different formalizations model different views on something that is one single mathematical concept to a mathematician. If a mathematician had to use one of the standard theorem proving systems, he or she would need to know in advance which choice of representation to make, since the choice of a good formalization is crucial for the success. But how do you know which formalization is best, when you start to explore something?

Another example of dynamic re-representation, which is very simple, but for which the different reasoning complexities are striking, is when associativity holds for an operation $+$. Once associativity is established, no mathematician would still use brackets, but the notation for a term like $((1 + x) + y)$ would change to $1 + x + y$. The property is encoded into the notation and can be retrieved from the given term. On a reasoning level this simple shift in representation can make a dramatic difference. For a term with $n + 1$ summands there are $\frac{1}{n+1}\binom{2n}{n}$ different ways to put brackets in. That means for a medium sized expression with just 10 summands there are already 4862 ways to represent it. If all these representations are part of the search process it is no surprise that automated theorem provers find such expressions difficult. The design of these systems does not allow for a change in representation, a user must write down unnecessary brackets in order to be syntactically correct, the brackets, however, do not help but unnecessarily confuse the reasoner. These all are signs of bad design. In the next sub-section we will look more closer at the change of representation.

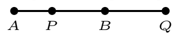
3.4 Change of Representations

Sometimes an appropriate reformulation of a problem into another representation is already the key step to find a proof. Different representations allow to apply knowledge from different sources to a problem. The importance of re-representation was pointed out by George Pólya [27, vol.2, p.80]:

> *When you are handling material things (for instance, when you are about to saw a limb off a tree) you automatically put yourself in the most convenient position. You should act similarly when you are facing any kind of problem; you should try to put yourself in such a position that you can tackle the problem from the most accessible side. You turn the problem over and over in your mind; try to turn it so that it appears simpler. The aspect of the problem that you are facing at this moment may not be the most favourable: Is the problem as simply, as clearly, as suggestively expressed as possible? Could you restate the problem?*

Of course you want to restate the problem (transform it into an equivalent problem) so that it becomes more familiar, more attractive, more accessible, more promising.

We exemplify this importance by different forms of re-representations:

– *functions:* Given an Euclidean space $\mathcal{R}$ with a metric function $|\ | : \mathcal{R} \times \mathcal{R} \to \mathbb{R}$ then for each pair $A, B \in \mathcal{R}$ of disjoint points there exists exactly one distance preserving function $g_A^B : \mathbb{R} \to \mathcal{R}$ with $g(0) = A$, $g(|AB|) = B$. With the help of this function the lines in Euclidean space can be interpreted as images of the real numbers. Notions of the real numbers, as intervals, correspond to notions in the abstract Euclidean space, namely line segments.
– *representation theorems:* Poincaré's model for the hyperbolic plane, where a line has infinitely many parallel lines through one point, is a unit disk, where circle segments correspond to hyperbolic lines. With this representation it becomes possible to re-represent constructions in the hyperbolic plane as constructions in the Euclidean plane.
– *theory change:* Geometric constructions can be represented as field extensions. The possibility to construct a square equal in area to a circle using compass and ruler can be re-represented to the question whether π belongs to a particular class of algebraic numbers (which it does not since it is transcendental).
– *inheritance:* The properties of monoids and groups are inherited by the multiplicative and additive substructures of rings and fields.
– *diagrams:* $P \in [AB], B \in [AQ] \Rightarrow P \in [AQ], B \in [PQ]$ which is obvious when this situation is expressed in a diagram: 

And of course there exist re-representations between different theories. Some of them changed the structure of mathematics itself:

– *Cartesian geometry:* arithmetic representation for geometry. This makes it possible to reduce geometrical problems to arithmetic problems and solve them arithmetically. This is actually the starting point of Descartes' idea to take arithmetic as a foundational system so that all problems should be translated to arithmetic and then solved by equation solving.
– *set theory:* mathematical concepts are representable as sets. Set theory is another foundational system on which most of mathematics can be based *in principle.*
– *group theory:* the group concept can be used to represent geometric transformations, permutations, and the solvability of polynomials.

Certain forms of representations are very important to form new concepts. The theorem that every permutation can be decomposed into transpositions, that is, can be represented as a product of transpositions, makes the definition of even and odd permutations suggestive. It is hard to imagine how to define the concept without this particular representational form. Once these concepts have been formed they become the starting point for the introduction of other concepts like the alternating group, which is defined as permutation group containing only permutations with an even number of transpositions.

Let us look at a different example, the concept of real numbers. Real numbers can be defined as Dedekind cuts or Cauchy sequences. However, Cantor's second diagonalization proof that the reals are uncountable is difficult to imagine without having the representation in the decimal (dual, or another) number system.

Often the opposite of a change in the representation happens in mathematics, that is, so-called overloading is used. The use of the same notation for different concepts, e.g. $|\,|$ in the example above for the concept of distance, $+$ for operations that behave like the well-known addition on natural numbers, etc. Even formally incorrect notation, such as equality for isomorphisms, is used to enable the transfer of knowledge from a known concept to a new concept. Certain *sort* and *type* systems allow for some kind of overloading, but they do not offer the kind of knowledge transfer that humans achieve this way in using overloaded symbols in an analogical way.

3.5 Structured Knowledge

A mathematical textbook is usually highly structured with *definitions*, *theorems* and *proofs* as main categories. This categorization is reflected in the formalizations for deduction systems. A closer look reveals that there exists a finer classification. For instance, definitions can be simple, inductive, or implicit. Some theorems are explicitly introduced as tools which can be applied in other situations or contain equivalence statements that are useful for re-representation. A proof can contain subproofs that will be repeatedly used for other theorems and that are emphasized to be important by the author, and a proof may contain an algorithm for the construction of mathematical objects.

A typical schema for the introduction of mathematical concepts is that a definition is followed by theorems giving simple properties and typical examples for this concept. One could argue that this structure has no significance for deduction systems because it is solely beneficial for the human process of learning and understanding. We try to show that the structure can become important as a basis for the reasoning process.

Take as an example the continuity of functions. The basic properties that are usually provided after this definition are that continuity is preserved for the sum, product, and composition of continuous functions. Usually as an example the identity on the reals will be given. Now suppose it is required to show that any polynomial is continuous. Instead of going back to the definition itself, the basic properties of the concept and the example of the identity function are used to prove the continuity of polynomials. The basic properties can be seen as attached to the definition and preferably used to prove simple proof obligations.

Of course it is possible to argue that the attached properties can be retrieved from a database that consists only of the plain structure of definitions and theorems. However, it is then difficult to query for a property like continuity of $+$. Should the query consist of all theorems containing the symbol 'continuous' or '$+$', or both? The structure that is lost in the plain logic representation would have to be reconstructed through elaborated query techniques that give useful results.

Note that the standard mathematical practice to build hierarchies of mathematical concepts is a very important means to reduce complexity in theorem proving. For instance in set theory, it is possible to define symbols like $\subset$, $\cup$, $\cap$ using the symbol $\in$. When you build a topology on top, using functions like $\circ$ for inner and $^{-}$ for closure, you can carry through most proofs on a level where you make use of abstract properties of $\subset$, $\cup$ and $\cap$, and can avoid totally to go down to the level of elements that involves $\in$.

Another category of mathematical knowledge form examples. They are crucial for the understanding of concepts. This is not only important to give semantics to syntactically defined concepts, but it can also be very relevant on the level of syntactic proof construction. We exemplify this for the theorem that "Groups G of order greater than two have non-trivial automorphisms." It seems to be hard to synthesize automorphisms directly only from the given assumption that G is a group. In this case, the standard human heuristic to try one of the typical examples, namely $x \mapsto gxg^{-1}$ and $x \mapsto x^{-1}$, provides already the crucial idea to prove this theorem.

3.6 Limited Expressiveness of Mathematical Tasks in Logic

When we take mathematical textbooks as basis for what is part of mathematics and what not, we can find statements that are not expressible in formal languages at all. Naturally comments or diagrams, and a number of models are not represented. Historic statements, statements about the relevance of properties and concepts and so on are part of mathematics, but not part of logic, although they are often important for a deeper understanding and an informed proof search. As Robinson cites himself in [28]: "Logic deals with what follows from what. ... The correctness of a piece ... does not depend on what the reasoning is about." That is a strength and a weakness of logic at the same time.

But even at the level of problem formulation, it is not always possible to apply a formal system in a straightforward way. Let us look for instance at the mathematical task (provided in some context): "Determine the maximum of $f(x)$ in the interval $[a, b]$." Does the obvious formalization "$\exists_{x\in[a,b]}\ \max(f, x)$" reflect the meaning of the sentence? Not necessarily, assume somebody comes up with the argument "Since f is a continuous function on a compact interval it has a maximum." This might be a correct argument to prove the logical formulation but it would not solve the original task. On a closer look adequacy of the logical statements depends on the logic used. If interpreted in a constructive way the logical formulation is adequate; if interpreted classically it is not. While there are constructive and classical systems around, it seems to be inappropriate that one has to decide for a constructive system once and for all, in order to be able to formulate a standard task like the one above.

While one can hardly cover all aspects of mathematics in a computer-based system, it seems for many applications – like the recently emerging applications in education – important to find a coverage which is as broad as possible.

3.7 A 'Natural' Calculus

The suggestions for a 'Mathematical Vernacular' [6] seem not to question that all concepts of mathematics are *sufficiently* expressible in a language consisting of functions and relations, potentially enriched by types or sorts. The design of the Vernacular seems not to be focused on the objects mathematicians are interested in, but on the reasoning framework.

The strength of a formal system can be measured by the de Bruijn factor, that is, the ratio of the length of the proof in the formal system compared to its version in a mathematical textbook.[7] A reassuring observation is that in the experiments with Automath and Mizar the de Bruijn factor remained constant for the proofs of a wide range of differently complex theorems.

Even if we agree with the conclusion that the proofs constructed in formal calculi are already in principle an approximation of standard mathematical proofs, it is important to note that such a comparison is based on the *output* of mathematical work, the language used by mathematicians to communicate proofs in textbooks and articles. While a comparison of such completed proofs, proofs after all search is finished may be interesting, they often do not resemble the proof construction. As an example look at standard ϵ-δ-proofs. In the proof construction you compute a sufficient criterion on δ, choose δ as a function of ϵ and then you prove that with this δ the difference of the function values is indeed smaller than ϵ. In a finished proof a crucial part of the proof construction – the construction of δ – is redundant and hence not presented. For this reason it is impossible to understand prima facie why δ has been selected as it is.

Traditionally, the formulations of final proofs are minimalist and mathematicians repress their original ideas, even the order of steps can be different, in favour of an objective rigorous style. As Pólya [26, p. vi], pointed out:

> *We secure our mathematical knowledge by demonstrative reasoning, but we support our conjectures by plausible reasoning ... Demonstrative reasoning is safe, beyond controversy, and final. Plausible reasoning is hazardous, controversial, and provisional. ... In strict reasoning the principal thing is to distinguish a proof from a guess, a valid demonstration from an invalid attempt. In plausible reasoning the principal thing is to distinguish a guess from a guess, a more reasonable guess form a less reasonable guess. ... [plausible reasoning] is the kind of reasoning on which his [a mathematician's] creative work will depend.*

The 'demonstrative reasoning' corresponds to a formulation in reasoning steps that were investigated by logicians. In this sense current deduction systems are suitable as proof checkers for existing and well-understood parts of mathematics but lack to act as proof assistants for the exploration and construction of new mathematical knowledge. How can 'plausible reasoning' be modelled? Proof planning follows the paradigm of proof search on an abstract level and can be seen as

[7] The notion is not without problems, since mathematical proofs are not standardized with respect to their detailedness. Furthermore there are other aspects for the quality of a proof than its length.

an important step into this direction. But as described in [4] even proof planning depends on structural restrictions of the underlying calculus and uses a formal language as the only representation for mathematical concepts.

There might be the view that we can come up with a more powerful logic which provides the best possible representation. Marvin Minsky gave a strong argument why this would not be the case in artificial intelligence in general, but why multiple representations are necessary. He recommends [21]:

> *Everywhere I go I find people arguing about which representation to use. One person says, "It is best to use Logic." The next person says, "No, logic is too inflexible. Use Neural Networks." The third person says, "No, Neural Nets are even less flexible, because you have to reduce everything to mere numbers. Instead, you should use Semantic Networks. Then, the different kinds of things can be linked by concepts instead of mere numbers!" But then the first person might complain, "No, Semantic Nets are too arbitrary and undefined. If you use Formal Logic, that will remove those ambiguities." What is the answer? My opinion is that we can make versatile AI machines only by using several different kinds of representations in the same system! This is because no single method works well for all problems; each is good for certain tasks but not for others. Also different kinds of problems need different kinds of reasoning."*

As we have seen for multiplication tables different types of representations allow for specialised and efficient reasoning methods connected to them, opposed to search on the logic level. Mathematicians carefully design their concepts to keep the search spaces small. We need to understand this aspect of mathematical reasoning much better in order to understand the versatility of the reasoning capabilities of human mathematicians.

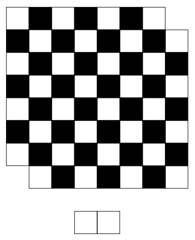
Historically the development of many concepts went the way that certain meta expressions were introduced, which were later reified and became object expression. The development of number systems might illustrate this. Having natural numbers and fractions, irrational numbers and negative numbers as well were considered as odd ones out, which only later became first class citizens. Then imaginary numbers were the odd ones and negative square roots were considered as strange entities which were used only for convenience. Likewise, functions were first concrete in nature, and only much later it was possible to speak about them.

In the next section we will contrast mathematical representations, which are very flexible and extendible, to formal computer systems, which still lack the required flexibility to design concepts to the level of sophistication which is achieved in informal mathematics.

4 Formal Representations of Mathematics

Up to here we have strongly argued how important a broad range of specialized constructs is for the adequate representation of mathematical knowledge and for proof construction. Representations find their analogues in data structures used in existing implementations of mathematical software systems, a range of general purpose and specialized theorem proving systems, computer algebra systems, and educational software. These data structures provide functionality and the ability to implement mechanisms working on them. In this section we want to take a brief look at some important features of systems from which ideas can be borrowed to realize a flexible system of the kind we envisage.

In a case study [16], we looked at different formalizations of the so-called mutilated checkerboard problem in different systems.



The challenge is to prove that *"It is impossible to cover the mutilated checkerboard shown in the figure with dominoes like the one in the figure. Namely, a domino covers a square of each color, but there are 30 black squares and 32 white squares to be covered."*

In the past a multitude of formalizations have been developed, for each system (be it an abstract system or an actual running system) at least one that is particularly well suited for that system. McCarthy states already in his original challenge paper [20] two different formalizations, one for first order logic without equality and one for first order with equality. At the second QED workshop organized by the Mizar group in Warsaw in 1995, McCarthy himself came back to the problem and presented a different formalization which makes use of set-theoretical notions and basic integer arithmetic. He gave this as a challenge to the community, namely to build a system to which one can give the essential hint to colour the mutilated checkerboard and to recognize that it has more white than black tiles but that any covering would cover equally many white and black tiles. The Mizar group took up the challenge and built directly a proof of the problem in Mizar using a formalization which is close to McCarthy's challenge [1]. There are also formalizations in Isabelle [23] and Coq [13].

While these systems show quite a flexibility in formalizing this problem and proving it, there are a number of mathematical concepts which need further structures. In particular the choice of representation is outside the formal system and the search for a good representation is not supported by the systems. We discuss some important ones in the following, in which special constructs have been made available to allow for a good representation.

4.1 Sorted Extensions to Logic

Sorted logic is a good example to exemplify the importance of careful design. It is an old insight that sorted logic (more carefully put, some classes of sorted logics) can have significant advantages over unsorted logic. Let us just consider the case of standard first-order logic with and without its order sorted extension. Sorts can be considered as special unary predicate symbols. Typical formulations in sorted logic are $\forall x_{\mathsf{Human}}\text{.}\mathsf{Mortal}(x)$, $\mathsf{socrates} \lessdot \mathsf{Human}$, $\neg\mathsf{Mortal}(\mathsf{socrates})$. The equivalent in unsorted logic is $\forall x\text{.}\mathsf{Human}(x) \Rightarrow \mathsf{Mortal}(x)$, $\mathsf{Human}(\mathsf{socrates})$, $\neg\mathsf{Mortal}(\mathsf{socrates})$.

The translation from the sorted to the unsorted logic is called relativization. The possibility to relativize any sorted problem formulation can be and has been used as an argument *against* the use of sorted logic in line with the standard argument "Everything you can do in your system of *sorted logic*, I can do in *unsorted logic*." This argument is – although correct – unhelpful because of the counterargument "Everything you can do in your *unsorted logic*, I can do in *sorted logic*, but in a smaller search space!"

The reason for the smaller search space is due to a better design realized by the sorted system. Although the formalism of sorted logic is equivalent in strength to the unsorted one, a concrete formulation (which makes use of sorts in a non-trivial way, that is, whose relativization is not equal to itself) is *not*. It is actually *weaker*, since certain things can *not* be derived in sorted logic which can be derived in unsorted logic (concretely, formulae like $\mathsf{Human}(1) \Rightarrow \mathsf{Mortal}(1)$ are syntactically possible in unsorted logic, but the equivalent is rejected in sorted logic). To generalise this observation: A particular design is *better* than an alternative one if certain redundant, or heuristically uninteresting derivations cannot be made.

The integration of sorts in a system also demonstrates how subtle design issues can be. We cannot discuss this in detail here, but we will give some indication. It should be noted that sorts are not necessarily exclusively beneficial.[8] If we take standard order-sorted logic we have for each relation like Human and Mortal to decide whether we want to formalize it by a sort symbol or by a predicate symbol. If we want to prove some statement like $\neg\mathsf{Human}(\mathsf{pegasus})$, we cannot formalize Human as a sort symbol. This is a serious flaw in standard sort systems, also it is not possible to model the genesis of concepts in an adequate way. In order to perform the reasoning above we need to know $\mathsf{socrates}{<}\mathsf{Human}$ a priori. It is not possible to infer that Socrates is a human being later in the reasoning process. This unduly limits the flexibility of the system. Ideally you would want to have the advantages of a sorted formulation whenever possible, but also benefit from the flexibility of the unsorted formulation when necessary. To our knowledge only Christoph Weidenbach's system [34] offers these possibilities.

4.2 Data Structures in Computer Algebra Systems

CASs offer many more primitive data structures such as matrices than standard theorem proving systems. With these structures it is possible to cover large parts of the 'computational' part of mathematics. As we tried to show, there is a grey area in which deduction and computation go hand in hand, since any structure in a CAS is concrete, while mathematical expressions often make use of ellipses, for instance, expressions which contain dots like $x_1, x_2, \dots, x_n$, or the multiplication table and matrix in Sections 2 and 3.2. A promising first approach in the direction of formalizing ellipses in reasoning can be found in [9].

[8] For a detailed discussion why sorts/types may be harmful see [18].

4.3 Data Structures for Reasoning

Koedinger and Anderson [17] introduce a representation different from a purely logical formalization called diagram configuration model (DC) for diagrammatic reasoning in geometry. The representation is based on DC schemas that encode typical geometric situations that were identified through observations on the problem solving behavior of experts in this domain. The schemas contain the main property of the situation, the subsumed properties and the different ways under which the schema can be established. The level of abstraction allows for an efficient inference algorithm that introduce the DCs as inference steps.

Mateja Jamnik describes in [14] a diagrammatic representation for theorems and inference steps based on this data structure that allow to infer theorems in arithmetic. The proofs constructed in the diagrammatic representation can be translated to proof planning and then formally verified with a theorem prover.

These examples are important in our context since they show that at least for particular domains it is possible to build special reasoners for diagrammatic reasoning which are distinct from a purely logic based system, but which can be formally linked to such a system. An adequate data structure for diagrams seems to be the key point for the success of both approaches. Only this data structure allows the implementation of an efficient inference mechanism.

4.4 Little Theories

Another important logic-based approach which approximates mathematical reasoning well is the little theory approach of IMPS [11]. As we can see in textbooks from different areas of mathematics, there is no fixed hierarchy of theories. Each textbook presumes knowledge from different areas of mathematics and by this induces a partial order of theories. That there exists a total order of theories is rather a theoretical result than used in everyday mathematics. The attempt to realize this total order prohibits the flexibility to use theorems constructed for one area in another area. The little theory approach also allows to relate different formulations with each other without necessitating a hierarchy.

5 Conclusion

In this paper, we presented aspects of mathematical representations that we think are highly relevant for mathematical theorem proving and that can – if at all – be implemented in traditional theorem proving systems only with great difficulty. While systems which make use of a language which is comparatively close to mathematical practice are less of a problem than systems whose input languages are close to mathematical logic, we think that the lack of acceptance of theorem proving systems (compared to the success of computer algebra systems) is at least partly due to unsolved problems of mathematical design.

We have summarized some approaches to good design in Section 4. More work in this direction is necessary to offer well-designed tools to mathematicians and other people interested in computer assisted mathematics. While certain parts might turn out to be straightforward other aspects will require a much deeper understanding. To give one example for the size of the task, if we wanted to integrate multiplication tables as a primitive in automated reasoning systems, it should be relatively easy to offer concrete multiplication tables with all the advantages mentioned in sections 2 and 3 (as it is relatively easy to offer order-sorted sorts). However, it will be difficult to offer general type multiplication tables which contain ellipses, and flexible ways to reason about them. Human beings have an amazing capability to reify structures in their space of discourse. We seem not to have yet a deep understanding of these capabilities.

What is way forward? Let us go back to the quote from the start of this paper, since it clearly shows such a way forward. Andrzej Trybulec says that firstly, *"we need experiments with much more advanced mathematics than already done"*. Only complex examples show the complex problems and show ways to their solution. Addressing examples such as reasoning with matrices presents new challenges and when we solve those we can improve the corresponding systems.

Secondly, *"a system for practical formalization of mathematics probably will not be a simple system based on small number of primitive notions"*. We could not agree more. Good design is not done once and for all, it remains a continued task that mathematicians have to master. While many systems are carefully designed and makes life easy for many problems, we as a community have not yet solved the problem to give mathematicians the tools at hand to provide and adapt this design for themselves. This will be a big challenge for the future.

Thirdly, *"integration with a computer algebra systems may be necessary or at least a feasible system must have bigger computational power"*. The field of reasoning systems can certainly learn a lot from that of computer algebra systems, but also an integration of different activities is very important. Up to now strong systems often are quite specialized. To integrate them in a flexible way and to still keep them maintainable is unsolved.

Acknowledgements We thank Mateja Jamnik for fruitful discussions. Martin Pollet thanks Peter Pröhle for the insights he gave him into the process of mathematical problem solving.

References

1. Grzegorz Bancerek. The mutilated chessboard problem – checked by Mizar. In Roman Matuszewski, editor, *The QED Workshop II*, pages 37–38, 1995. Available from `http://www.mcs.anl.gov/qed/index.html`.
2. H. Barendregt and A.M. Cohen. Electronic communication of mathematics and the interaction of computer algebra systems and proof assistants. *Journal of Symbolic Computation*, **32**(5):3–22, 2001.
3. M. Beeson. Automatic generation of epsilon-delta proofs of continuity. In J. Calment and J. Plaza, editors, *Artificial intelligence and symbolic computation*, pages 67–83. Springer Verlag, Berlin, Germany, LNCS 1476, 1998.
4. C. Benzmüller, A. Meier, E. Melis, M. Pollet, J. Siekmann, and V. Sorge. Proof planning: A fresh start? In M. Kerber, editor, *IJCAR-Workshop: Future Directions in Automated Reasoning*, pages 30–37, Siena, Italy, 2001.

5. R. Brachman and H. Levesque, editors. *Readings in Knowledge Representation.* Morgan Kaufmann Publishers, Los Altos, California, 1985.
6. N.G. de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. In R. Nederpelt, J. Geuvers, and R. de Vrijer, editors, *Selected Papers on Automath*, pages 865–935. Elsevier, North-Holland, Amsterdam, The Netherlands, 1994. Studies in Logic, Volume 133.
7. A. Bundy. The use of explicit plans to guide inductive proofs. In E. Lusk and R. Overbeek, editors, *Proc. of the 9th CADE*, pages 111–120, Argonne, Illinois, USA, 1988. Springer Verlag, Berlin, Germany, LNCS 310.
8. A. Bundy. A science of reasoning. In *Computational Logic: Essays in Honor of Alan Robinson.* MIT Press, 1991.
9. A. Bundy and J. Richardson. Proofs about lists using ellipsis. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proc. of the 6th LPAR*, pages 1–12. Springer Verlag, Berlin, Germany, LNAI 1705, 1999.
10. H. Elbers. *Connecting Informal and Formal Mathematics.* PhD thesis, Eindhoven University of Technology, 1998.
11. W. Farmer, J. Guttman, and F. Thayer. Little theories. In D. Kapur, editor, *Proc. of the 11th CADE*, pages 567–581, Saratoga Springs, New York, USA, June 1992. Springer Verlag, Berlin, Germany, LNAI 607.
12. P. Hayes. Some problems and non-problems in representation theory. In *Proc. of the AISB Summer Conference*, pages 63–79, Sussex, 1974.
13. Gérard Huet. The mutilated checkerboard (Coq library), 1996. `http://coq.inria.fr/contribs/checker.html`.
14. M. Jamnik. *Mathematical Reasoning with Diagrams: From Intuition to Automation.* CSLI Press, 2001.
15. Manfred Kerber. A dynamic Poincaré principle. In Jonathan M. Borwein and William M. Farmer, editors, *Mathematical Knowledge Management – 5th International Conference, MKM 2006*, pages 44–53, Wokingham, UK, 2006. Springer, LNAI 4108.
16. Manfred Kerber and Martin Pollet. A tough nut for mathematical knowledge management. In Michael Kohlhase, editor, *Mathematical Knowledge Management – 4th International Conference, MKM 2005*, pages 81–95, Bremen, Germany, 2006. Springer, LNAI 3863.
17. K. Koedinger and J. Anderson. Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science*, 14:511–550, 1990.
18. L. Lamport. Types are not harmless. Technical report, 1995.
19. U. Martin. Computers, reasoning and mathematical practice. In U. Berger and H. Schwichtenberg, editors, *Proceedings of the NATO Advanced Study Institute on Computational Logic, Marktoberdorf, Germany, July 29 - August 10, 1997.* Springer Verlag, 1999.
20. John McCarthy. A tough nut for proof procedures, 1964. Stanford Artificial Intelligence Project Memo No. 16, 1964. Available from `http://www-formal.stanford.edu/jmc/`.
21. M. Minsky. Future of AI technology. *Toshiba Review*, 1992. `http://web.media.mit.edu/~minsky/papers/CausalDiversity.txt`.
22. D. Norman. *The Design of Everyday Things.* The MIT Press, London, 1998.
23. Lawrence C. Paulson. A simple formalization and proof for the mutilated chess board. *Logic Journal of the IGPL*, 9(3):475–485, 2001. Also published as Technical Report Computer Laboratory, University of Cambridge, 394, May 1996.
24. F.J. Pelletier. The philosophy of automated theorem proving. In John Mylopoulos and Ray Reiter, editors, *Proc. of the 12th IJCAI*, pages 538–543, Sydney, 1991. Morgan Kaufmann, San Mateo, California, USA.

25. Martin Pollet, Volker Sorge, and Manfred Kerber. Intuitive and formal representations: The case of matrices. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *Mathematical Knowledge Management. Third International Conference, MKM*, Białowieża, Poland, September 19-21, 2004. Springer, LNCS 3119.
26. G. Pólya. *Mathematics and Plausible Reasoning.* Princeton University Press, New Jersey, USA, 1954.
27. G. Pólya. *Mathematical Discovery – On Understanding, Learning, and Teaching Problem Solving.* Princeton University Press, New Jersey, USA, 1962/1965.
28. A. Robinson. Proof = Guarantee + Explanation. In S. Hölldobler, editor, *Intellectics and Computational Logic*, pages 277–294. Kluwer Academic Publishers, 2000.
29. B. Russell. *The Principles of Mathematics.* George Allen & Unwin Ltd, London, UK, 2nd edition, 1937 edition, 1903.
30. B. Russell. The philosophy of logical atomism. *The Monist*, 28/29:495–527/32–63,190–222,345–380, 1918/1919. Republished in [31, p.177-281].
31. B. Russell. *Logic and Knowledge.* Allen & Unwin, London, 1956.
32. Alan P. Sexton and Volker Sorge. Processing textbook-style matrices. In Michael Kohlhase, editor, *Mathematical Knowledge Management, 4th International Conference, MKM 2005, Bremen, Germany, July 15-17, 2005, Revised Selected Papers*, volume 3863 of *Lecture Notes in Computer Science*, pages 111–125, 2006.
33. Andrzej Trybulec. Towards practical formalization of mathematics. In Fairouz Kamareddine, editor, *Abstracts of the invited talks at the Workshop on 35 years of Automath*, 2002. `http://www.cee.hw.ac.uk/~fairouz/automath2002/abstracts/abstracts.html`.
34. C. Weidenbach. Extending the resolution method with sorts. In *Proc. of the 13th IJCAI*, pages 60–65, Chambéry, France, 1993.
35. A.N. Whitehead and B. Russell. *Principia Mathematica.* Cambridge University Press, Cambridge, UK, 1910.
36. L. Wittgenstein. *Remarks on the Foundations of Mathematics.* Basil Blackwell, Oxford, England, third edition edition, 1956.
37. Claus Zinn. *Understanding Informal Mathematical Discourse.* PhD thesis, Universität Erlangen Nürnberg, 2004.