Looking for Busy Beavers. A socio-philosophical study of a computer-assisted proof

Liesbeth de Mol

Looking for Busy Beavers. A socio-philosophical study of a computer-assisted proof.

Liesbeth De Mol*
elizabeth.demol@ugent.be

Abstract

"*Young man, in mathematics you don't understand things, you just get used to them*"
John von Neumann

1 Introduction

What exactly is the impact of the computer on mathematics? If one were to believe some of the advocates of computer-assisted mathematics "*computers [are] changing the way we do mathematics*" [8]. This alleged change concerns a shift in perspective on mathematical knowledge and the way it is attained, a change brought about and made explicit through the computer. Mathematicians like Borwein and Bailey [9, 10], Seiden [41] and Zeilberger [46] have emphasized on several occasions that the increasing significance of computer-assisted mathematics makes it more and more clear that quasi-empirical or experimental methods must be included and be taken more seriously within the mathematical discourse, that mathematics has more in common with the empirical sciences than is usually believed. They question the traditional ideas on mathematical certainty, proofs, rigor and understanding. Also, within the philosophy of mathematics, (examples of) computer-assisted mathematics (are) is mainly discussed in the context of work that can be placed in the tradition of Polya and Lakatos,[1] work that emphasizes the significance of the practice of the mathematician,[2] the fallibility of mathematics and the significance of (quasi-)heuristics opposing the more traditional idea that mathematics is without history, without change, that mathematics is no more than a body of absolute and certain knowledge.[3]

*This research was supported by the Fund for Scientific Research, Flanders, Belgium.
[1]See for example [24] and [37].
[2]See for example [21]
[3]See for example [45]

The idea that computer-assisted mathematics has this kind of impact on mathematics and its philosophy has of course also been opposed. Indeed, several philosophers and mathematicians have argued for several different reasons that the computer does not have this kind of epistemological impact on mathematical methods. In the end, theoretically, it is not capable of anything we were not already capable of before.[4]

The aim of this paper is to contribute to the question concerning the impact of the computer on mathematics, a question which, in our opinion, will only gain in importance in the future. Here, I will focus on computer-*assisted* proofs. This work is to be situated in a larger research project that aims to develop a more systematic and complete approach towards computer-assisted or, as we will identify it in the remainder of this paper, mechanized mathematics.[5] Such approach is still lacking in the literature.

One important part of such a systematic approach towards mechanized mathematics is the micro-analysis of (well-known and less well-known) examples throughout the history of mechanized mathematics, starting from (but not necessarily ending with) the accounts of the mathematicians themselves. There are already some examples in the literature of relatively detailed case studies like MacKenzie's socio-history of the four-color theorem [33], probably the most famous example of a computer-assisted proof, and Van Bendegem's account of the Collatz problem [6], which is a typical example of a problem studied with the help of the computer. In this paper we will look at a relatively unknown example of a *computer-assisted proof*, i.e., the solution of the Busy Beaver problem for the class of Turing machines with 2 symbols and 3 resp. 4 states. The Busy Beaver game (or competition) for a certain class of Turing machines (with m states and n symbols) is to find the Turing machine which prints out the maximum number of 1s before halting when started from a blank tape (See Sec. 2.1 for the technical details).

As is stated in the title of this paper, the case analysis should be regarded as a socio-philosophical analysis. This means here that I will start from a relatively detailed (micro)-analysis of a specific example of a computer-assisted proof, tracing the immediate consequences and problems as they are interpreted by the mathematician(s) her/himself during the process of proving and in the communication of the proof. I will then discuss the most important philosophical problems related to computer-assisted proofs in the context of the case analysis.

1.1 What are computer-assisted proofs?

Within the literature one can easily determine different kinds of computer "proofs". There is, for example, an important difference between a (1) mechanized probabilistic proof that shows that a certain very large number x is prime, using the Miller-Rabin primality test, (2) visual proofs as they for example occur

[4] See for example, [5, 13, 16, 30, 43, 44] Note that this does not necessarily mean that they oppose the idea of mathematics being not that absolute body of truths.

[5] I follow Derrick Henry Lehmer [27] here, a computer pioneer and number theorist, who is one of the main inspirators of the present work.

in fractal geometry (see for example [36]), (3) McCune's proof of the Robbins algebra conjecture which relies on an automated theorem prover [34] and (4) Hales' proof of the sphere packing problem [19].
In this paper, unless indicated otherwise, we will use the term computer-assisted proof (CaP for short) in the sense of Derrick H. Lehmer (see e.g. [26]), who was involved with one of the first true computer-assisted proofs [29]. A CaP is a proof that proves a theorem that *practically* could not have been (or, thus far, has not been) proven, re-proven or verified by human mathematical reasoning alone. I.e., (certain parts of) both the process that results in the proof as well as the proof itself must be *humanly impractical*. As a consequence, these proofs are, practically speaking, not surveyable by humans. Furthermore, the proof is also *machine impractical* in that, besides the programming, certain parts of the proof could not have been done by the computer. In this sense, we use the term computer-*assisted* proof rather than computer proof. These proofs typically involve the verification of a large number of cases by the computer, although the work of the computer is not restricted to this verification. A well-known (and probably the most famous) example is the proof of the four-color theorem (4CT for short) by Appel and Haken [1, 3].[6]
This definition is not intended as a once-and-for-all-given definition. It should be understood as an instrument to evaluate and demarcate certain computer applications which, when analyzed, can in their turn change the semantic content of CaP's.

1.2 Why Busy Beavers?

There are two main reasons that motivate my choice for this specific case analysis.
The first and least important one is that in the present case study the proofs and the problem itself can be relatively easy explained. Unlike, for example, the computer-assisted proof of the sphere packing problem, the "proofs" are simple enough to be explained up to a relatively high level of detail. The case-study is thus ideal for the intended micro-analysis.
The main reason for selecting this case is that it is not well known. In the context of CaPs, and, more generally, mechanized mathematics one focuses mainly on the more famous examples and neglects the more "normal" ones. Within the literature on CaP's the discussions are usually restricted to one of the following three examples, in order of increasing popularity:

(a) The non-existence of a finite projective plane of order 10 by Lam, Thiel and Swiercz [15]

[6]Note that several examples from the literature that are quite frequently considered as examples of computer proofs are excluded by this definition. For example, if a computer finds a counter-example to a certain conjecture, this does not count as an example of what is here understood under computer-assisted proof as, once the counterexample has been found, the human can easily check that it is a counterexample, and thus disprove the conjecture.

(b) The sphere packing problem by Thomas Hales [19]

(c) The 4CT by Appel and Haken [1, 3] and its alternative proofs by Robertson et al [40] and Gonthier [18].

(a) is mostly only mentioned without any real discussion, it is yet another example of a CaP, while (b) and especially (c) have given rise to several different heated debates, going from the question whether such proofs are really proofs to the problem of the refereeing of such proofs.
Of course, one cannot deny that e.g. the 4CT *is* more interesting than the Busy Beaver example to be studied here since, on the one hand, the result is a proof of an old mathematical problem with a long history, and, on the other hand, it is not situated within the context of theoretical computer science (as is the case for the present case study). The same goes for the other two. However, even if (a), (b) and (c) have already been studied in the literature and are more interesting, this is no reason not to be interested in less well-known and interesting examples of CaPs. First of all and generally speaking, if one restricts the attention to only the "famous" examples, this might lead to an all too restrictive view on CaPs. Secondly, by drawing the attention to the fact that there are more than three CaPs, one draws the attention to the fact CaPs are not as abnormal as one might believe and one thus counters the argument that, as there are only a few CaP's, they cannot be that important. And abnormal they are not. A very simple search on the following terms: *Computer proof* (C.Pr.), *Machine proof* (Mach. Pr.), *Mechanical proof* (Mech.Pr.), *Computer-assisted proof* (CaP) and *Automated proof* (Aut.Pr.) in MathSciNet and DBLP, two on-line databases,[7] resulted in Table 1:
Although these numbers are not overwhelming and it is furthermore not the

	C. Pr.	Mach. Pr.	Mech. Pr.	CaP	Aut.Pr.	Tot.
DBLP	196	42	24	22	72	356
MathSciNet	73	16	53	161	79	382

Table 1: Overview of the number of papers in MathSciNet and DBLP that mention computer proofs.

case that all of these proofs are CaPs in the sense understood here, they show that one cannot simply discard the significance of CaPs on the basis of numbers. Although they have not yet become a "normal" method of mathematics, they are more important than is usually believed.
Thirdly, by focusing on the less well-known examples, it becomes possible to study the impact of the computer not only on the level of the great innovations of mathematics – the famous examples – but also on the level of the "everyday" practice of the mathematician, which is not redundant *if* one accepts the view that mathematics cannot be reduced to its great achievements.

[7] MathSciNet is a database for mathematics in general (http://www.ams.org/mathscinet/search.html), whereas DBLP is a computer science database (http://www.informatik.uni-trier.de/ ley/db/).

Finally, it is not because one has seen one very important CaP that one has seen all. By studying more and more examples of CaPs it becomes possible to tackle certain more general questions more exactly. What kind of problems can be solved with CaPs? What kind of methods are used and in what way are they different from other methods? How are CaPs communicated and perceived? What kind of techniques are used to convince the fellow mathematicians of the proof? Is there an evolution in the way CaPs are made and formalized?...These kind of questions allow to get a more concrete view on what CaPs are and in what sense they really differ from other proofs. These general questions cannot be answered properly if one restricts ones attention to only three CaPs.

2 Looking for Busy Beavers

In a paper titled *On non-computable functions* [38] Tibor Radó proposed an example of "*the phenomenon of non-computability in its simplest form*", an example which is now known as the Busy Beaver function $\Sigma(m), m \in \mathbb{N}$. He provided the following motivation for formulating and studying the problem [39]:

> Let us note that our main objective is to observe the phenomenon of non-computability in its simplest form, so that we can use the insight we achieve to see better what tasks we can delegate to computers. Actually, the comments to be presented here originated with the writers studies relating to the optimal design of automatic systems, and specifically with efforts to use computers to the limit of their capabilities for this purpose.

In other words, the computer not only plays a fundamental role in the solution of specific cases of the problem, but also led to the formulation of the problem. Furthermore, as an example of an uncomputable problem, it is situated in the context of the theory of computing and is thus, on the theoretical level, closely related to the computer.
Recall that, given Turing's thesis or any other logically equivalent thesis, a problem is considered non-computable (or recursively unsolvable) iff. there is no Turing machine that is able to compute it. One of the more famous examples is the halting problem, i.e., the problem to decide (compute) for any Turing machine whether or not that machine will halt.
The fact that a problem is non-computable in general, does not mean that every instance of the problem is also non-computable. I.e., it is not because there is a Turing machine with a non-computable halting problem that every Turing machine has a non-computable halting problem. One can thus search for strategies that allow to decide a certain generally undecidable problem for specific classes of "decidable" Turing machines. This is the goal Radó, and, after him, several other researchers, set himself: to compute the Busy Beaver function $\Sigma(m)$ for specific ms. It was soon understood that the computer would be an indispensable helper.

After a preliminary section (Sec. 2.1), defining some of the basic notions used here, we will give a (relatively) detailed account of computer-assisted proofs that $\Sigma(3) = 6, \Sigma(4) = 13$ (Sec. 2.2). This will be followed by a discussion of three of the typical features of these proofs (Sec. 2.3). In the next section (Sec. 3), we will confront these proofs with some of the fundamental epistemological problems related to computer-assisted proofs.

2.1 Some Preliminaries

A (standard) Turing machine T consists of a read-write head and a two-way infinite tape. A blank is denoted by the symbol 0. To start with, the tape contains a finite initial configuration, possibly empty, on an otherwise blank tape. In its initial state (state 1), the head reads the leftmost symbol of the initial configuration.[8] The machine is said to halt when it reaches the halting state H. T is formally defined by the finite set of states Q plus the halting state H, a finite set of symbols $\Sigma = 0, 1, ...$ and a transition function $f : Q \text{ x } \Sigma \rightarrow (\Sigma \text{ x } \{L, R\} \text{ x } Q)$. The transition function f determines for any state $q_i \in Q$ and any symbol $s_j \in \Sigma$ what the machine should do when in state q_i, reading the symbol s_j. I.e., if $f(q_i, s_j) = (s_{i,j}, D_{i,j}, q_{i,j})$ then, if T is in state q_i, reading symbol s_j, T replaces s_j by $s_{i,j}$, moves in direction $D_{i,j} = \{L, R\}$ (L stands for left, R stands for right) and goes to state $q_{i,j}$. In what follows, an instruction of a Turing machine will be represented by the quintuple $(q_i, s_i, s_{i,j}, D, q_{i,j})$.
Let $HT(m, 2)$ be the class of Turing machines with m states and 2 symbols that halt when started from a blank tape. Then, for $T \in HT(m, 2)$ let $\sigma(T)$ resp. $s(T)$ denote the number of symbols different from 0 left on the tape resp. the number of computation steps before T halts. Let $\Sigma(m)$ be the maximum $\sigma(T)$ and $S(m)$ the maximum $s(T)$ with $T \in HT(m, 2)$.

Definition 1 *The Busy Beaver problem is the problem to determine* $\Sigma(m)$ *for any* $m \in \mathbb{N}$

Definition 2 *The maximum shift number problem is the problem to determine* $S(m)$ *for any* $m \in \mathbb{N}$

Both problems were proven to be uncomputable by Radó [38].[9] Note that computing specific values $\Sigma(m), S(m)$ for specific n comes down to solving a special case of the halting problem for the class of machines with n states as one needs to be able to determine the subclass $HT(m, 2)$. Table 2 gives an overview of the known values in the Busy Beaver competition.

[8] Of course, if the initial configuration is empty, the head starts at some arbitrary square, reading 0.

[9] Radó only considered 2-symbolic Turing machines. The reason for this is that any n-symbolic Turing machine can be simulated or reduced to a 2-symbolic Turing machine. This was proven by Shannon. In current research on the Busy Beaver problem, one also considers Busy Beaver functions for classes of Turing machines with the number of symbols $m > 2$. See for example [35]. We will not consider these generalized Busy Beaver problems here.

m	$S(m)$	$\Sigma(m)$	Source
1	1	1	Trivial
2	4	2	Mentioned by Radó 1962
3	21	6	Radó and Lin (1965), Brady (1983), Kopp (1981)
4	107	13	Brady (1983), Kopp (1981)
5	$\geq$ 47 176 870	$\geq$ 4098	Marxen and Buntrock (1990)
6	$> 2.5 \times 10^{2879}$	$> 4.6 \times 10^{1439}$	Terry and Shawn Ligocki (2007)

Table 2: Overview of the current result in the Busy Beaver competition

2.2 Determining $\Sigma(m)$...

[...] when the writer wanted to find a certain highway on an automobile trip, he received the following directions [...]: "Drive straight ahead on this road; you will cross some steel bridges; and after you cross the last steel bridge, make a left turn at the next intersection." Luckily, the unsolvable problem implied by this advice was resolved by a member of the construction crew who volunteered the information that "after you cross the last steel bridge, there isn't another steel bridge until you reach Richmond, 130 miles away."

Tibor Radó, 1962

In [38], Radó pointed out that the case with $m = 1$ trivial and that the case $m = 2$ was computed during a seminar. For any $\Sigma(m), S(m)$ with $m > 2$ laborious and lengthy proofs, including long computations seemed unavoidable. The reason for this is that the number of Turing machines with 2 symbols and m states grows exponentially fast for increasing m. Indeed, the size of the class of 2-symbol Turing machines with m states is equal to $(4m + 1)^{2m}$.
Radó and Lin proved with the help of the computer that $\Sigma(3) = 6, S(3) = 21$ [31]. Brady [12] and Kopp [22] (reported in [32])[10] proved that $\Sigma(4) = 13, S(4) = 107$ and also confirmed the results by Radó and Lin. In what follows, we will give a relatively detailed account of the proofs of these results.
Before doing so, I must point at a difference between the Turing machine representation used by, on the one hand, Brady and, on the other, Kopp, Radó and Lin. Contrary to Brady, they treat a halt as a separate branch to a state 0 within a normal entry of a Turing machine. This has an effect on the total number of 2-symolic Turing machines with m states. Instead of $(4m + 1)^{2m}$, there are now $[4(m + 1)]^{2m}$ distinct 2-symbol, m-state machines. Table 3 gives an overview of the total number of machines with 2, 3 and 4 states for both approaches. As is clear from table 3 the approach by Brady results in a smaller number of cases to start from.
The three proofs all make use of a series of computer-assisted reductions of the total number of cases for each of the classes of 2-symbolic Turing machines with 3 and 4 states until finally no so-called *holdouts* remain. I.e., it was determined for each of the machines individually, whether or not they halt when started

[10] Kopp is the girl name of Machlin.

m	$(4m+1)^{2m}$	$[4(m+1)]^{2m}$
2	6561	20736
3	4 826809	16 777 216
4	6 975 757 441	25 600 000 000

Table 3: Number of 2-symbol Turing machines with m states.

from a blank tape, and, if they halt, which values $s(T)$ and $\sigma(T)$ they have. For each of the proofs found, there was a conjectured value for $\Sigma(m)$ and $S(m)$. These were proven lower bounds, found by making use of certain heuristic and explorative methods.[11] In all cases, the conjectured values turned out to be the correct values.

The proofs consist of two main stages of reduction, a more theoretical and a more heuristic stage. In the first stage, certain theoretical considerations are used that result in the immediate (computer-assisted) elimination of a large number of machines. Radó and Lin came to the conclusion that all machines of which the first instruction is not (1, 0, 1, R, 2) could be eliminated.[12] These methods were also used by Brady and Kopp. They extended the argument by Radó and Lin by what they call a *tree generation*, generating instructions as they are needed.[13] This method could be easily automated and was used to eliminate a large number of machines. Table 4 gives an overview of the number of remaining machines, called the *holdouts*, after the application of the several elimination methods used in stage one for each of the proofs.[14] The differences between the number of remaining machines after application of the tree normalization between Brady and Kopp can be explained by slight differences in their respective approaches. As is clear from Table 4 certain theoretical considerations allowed for a serious reduction in the number of cases to be considered. However, the number of remaining cases is still too large to be humanly practical

The next step in all the proofs is to turn to, what Brady calls, more heuristic proof techniques. In the next stage, Radó and Lin first used the conjectured value $S(3) = 21$ in order to eliminate some further machines. Clearly, all those machines that halted before this respective number of steps was reached could be eliminated. Kopp used the same technique, although she did not use the conjectured value for $S(4)$ but decided to run each of the remaining machines for some hundred steps $n > 107$. In case of Kopp, this technique led to the elimination of 1364 resp. 182,604 machines for the 3-state resp. the 4-state

[11] For example, in [11] Brady mentions that he used certain heuristic methods to conjecture that $\Sigma(4) = 13, S(4) = 107$. Note that in the ongoing research on the Busy Beaver competition one still makes use of several heuristic methods to determine lower bounds, methods which are also used in proofs of Busy Beaver winners.

[12] For an explanation why this can be done the reader is referred to [31].

[13] For more details the reader is referred to [32].

[14] The method of (tree) normalization was not the only method used. However, it is the most important one.

n	Radó and Lin	Brady	Kopp
3	82,944	± 4,000	3,936
4	\	± 550,000	603,712

Table 4: Number of Turing machines remaining after the first series of reductions.

case. Brady first reduced the ± 4,000 resp. ± 550,000 remaining machines to 27 resp. 5,820 *holdouts* by coupling the tree generation program to "*a heuristic solution to the halting problem.*". No exact numbers are known in the case of Radó and Lin.
So how to proceed from here? In the next steps, Kopp, Brady, Lin and Radó made use of more explorative and heuristic methods. These were used in order to:

- identify or "discover" different types of "patterns", called *infinite loops* by Kopp, in the behavior of the holdouts
- automate the detection of these patterns in the holdouts

Kopp, Brady, Lin and Radó were able to identify different types of infinite loops with the help of the computer. For each of these types of loops it can be proven that, if they occur in a given Turing machine, then that machine will never halt and thus its halting problem is decided. Now, if it could be proven for each of the holdouts that its ultimate behavior is an infinite loop than it is proven that these holdouts will never halt and one can thus prove the result (since the values $s(T)$ and $\sigma(T)$ are known for each of the halting machines).
What Kopp, Brado, Radó and Lin basically did was first to print-out the behavior of some of the holdouts, study it and try to detect certain patterns that could then be generalized and be proven to be cases of infinite loops. Programs were then written that allowed for the automated detection of infinite loops which could then result in the elimination of machines whose ultimate behavior was one of the infinite loops found and formalized in a program. In the end, several types of infinite loops were detected. The most important ones are simple loops, Christmas trees, shadow Christmas trees and counters.[15]
Identifying and then detecting different types of infinite loops was the hardest part of the Busy Beaver proofs. Machlin (Kopp) and Stout describe it as follows ([32], pp. 91–92):

> The major effort in calculating busy beaver numbers [...] lies in proving that large numbers of machines are in infinite loops [i.e., never halt]. The approach taken [by Brady, Radó and Lin and Kopp] is to examine some of these machines by hand, elicit a common behaviour which insures that a machine is in an infinite loop, and then write a program which examines candidate machines and proves that some

[15] Note that not all of these types were found by Radó and Lin.

> of them do indeed have that behavior. This process tends to iterate, with the researcher constantly trying to reduce the number of unclassified machines by either generalizing types of behavior earlier searched for, or by discovering new types of behavior

Brady gives the following description of the structure of the automated detection of infinite loops:

> BBFILT was used to separate heuristically the 5,820 holdouts into "Xmas Trees", "Counters," and "Unknown," while BBFXX, a modification of BBFILT separated the "Alternating Xmas trees" from the "Unknown" set. BBX2 was the Xmas Tree prover [...]. BBSHAD was a modification to handle "Trees with Shadow", BBALTX was an extension of BBX2 to handle "Alternating Xmas Trees", while BBALTX1 was a minor modification of BBALTX to handle double sweeps in which the extremum was reached on alternate sweeps only. BBC was the counter prover, while BBCM was a modification of BBC to handle "two-shot" carries and some cases of cell interdependence. More than 18 other programs were written for various housekeeping purposes, simulating and displaying machine behavior, exploring other reduction and filtering possibilities, etc. In all, at least 53 files were created and maintained for the project. Keeping track of what resembled a large scientific experiment became a major task in itself.

After the infinite loop detection program was applied, the small number of remaining hold-outs were then examined "by hand" and all eliminated as other cases of infinite loops. Hence, the results that $\Sigma(4) = 13, S(4) = 107$ (in case of Brady and Kopp) and that $\Sigma(3) = 6, \Sigma(4) = 13$ (in case of Lin and Radó, Brady and Kopp). Even this last stage was partly computer-assisted. As Brady explains:

> All of the remaining holdouts were examined by means of voluminous printouts of their histories along with some program extracted features.

2.3 Some features of the proofs

In what follows I will discuss three important features of the Busy Beaver proofs in more depth.

2.3.1 Experimental and heuristic methods.

As was shown in Sec. 2.2, the second stage of each of the proofs is more involved. It is also this stage which can be called the more explorative and heuristic stage of (the process of finding) the proof. As Brady describes it [12]:

> In this final stage of the $k = 4$ case, one appears to move into a heuristic level of higher order where it is necessary to treat *each machine* as representing a *distinct theorem*. [...] The proof techniques, embodied in programs, are entirely heuristic, while the inductive proofs, once established by the computer, are completely rigorous and become the key to the proof of the new and original mathematical results: $\Sigma(4) = 13$ and $S(4) = 107$.

The "heuristic" character of the second stage of the proof needs to be situated on two different levels: on the one hand, the *identification* of different kinds of infinite loops (simple loops, christmas trees, shadowy christmas trees, counters) and their variants, on the other, the actual *detection* of these loops in the class of holdouts, ultimately reducing the number of holdouts to 0.

The identification of new types of infinite loops can be considered as an experimental process in the following sense: the behavior of some (randomly selected) holdouts was printed out, then examined by hand and a new type of loop or some variant of an already known type was possibly identified. Brady also mentions that the computer was not only used to merely print out the behavior to assist in the identification of infinite loops, but also to extract certain features that might indicate an infinite loop. It was not known in advance whether the holdout studied would show some new pattern of an infinite loop. Maybe more steps would be needed in order for such pattern to show itself or maybe it was a halting machine. It was also unknown in advance how many different types and which types one could expect. To paraphrase Lehmer, this process of identification of infinite loops is a process of *exploring* the universe of mathematics, assisted by the computer.

The heuristic character of the actual *detection* of infinite loops in the number of holdouts, concerns the use of what Brady has called *heuristic programs*. These are identified as such, because there is no guarantee that the decision made by the program is the correct one. Given for example one of the christmas tree detection programs, then the uncomputability of the halting problem combined with the practical fact of finite time implies that it is not guaranteed that this christmas tree detection program will detect every case (or a variant) of a christmas tree. I.e., there are cases of holdouts which might actually be cases of christmas trees, but which will not be detected as such by the detection program. It might for example be the case that the holdout is a yet undiscovered variant of a christmas tree, or that the typical behavior of a (variant of a) christmas tree, as described in the christmas tree detection program only "shows" itself after say billions and billions of computation steps. Since one does not know in advance if a given holdout is a case of a christmas tree, nor, if it is, *when* the typical behavior of a christmas tree will be observed, one needs to make certain choices, in order to assure that the christmas tree detection program will halt for every case. Machlin and Stout explain that if their christmas tree detection program ran too many steps without finding the desired behavior then the machine remained a holdout, even if it might in fact be a case of a christmas tree. This problem is described as follows in the case of a program called the

backtracking program [32]:

> While backtracking can be useful, it cannot be guaranteed to always stop since otherwise it would suply a solution to the halting problem. As with all the heuristics we discuss, one must make some decision as to how long to run this technique before abandoning it.

Another example of a kind of heuristic program (used by Brady and Kopp) is the "tentative" classification of the holdouts as cases of christmas trees or counters. This classification was made on the basis of the rate at which new tape squares were visited. On the basis of the decision "made" by this program, either a counter or christmas tree detection program was run. Brady calls this program (BBFILT) a heuristic filter, "*a heuristic technique based upon experimental observation.*"
It has been argued that CaPs like the 4CT show that there are certain parts of mathematics that are (quasi-)heuristic in nature. Tymoczko's paper [45] is the most well-known paper in this respect. His main reason however for considering the 4CT heuristic in nature, is the fact of its human unsurveyability. This argument has been countered in the literature on many occfasions.[16] Now, the authors of the Busy Beaver proofs very clearly do not shy away from identifying certain aspects of their proofs as experimental, explorative or heuristic in nature. However, their reasons for doing so has nothing to do with the unsurveyability of their proofs, but rather with the inherent and practical unpredictability of the different Turing machines to be considered. It is this unpredictability that forces them to explore the behavior of the different machines and to use so-called heuristic programs.

2.3.2 The process of the proof

Normally, when a proof is published in a paper, no (conscious) mention is made of the process of finding that proof.[17] The actual proof and the process of finding that proof are considered to be strictly separated. What counts is that there is a proof of some theorem. The proof is that which needs to be communicated, what needs to be published, what remains. The dirty details of the process that resulted in the proof are considered irrelevant.
In the published accounts of the Busy Beaver proofs, on the other hand, one finds that a lot of information is given about the process of finding the proof. In a certain way, this should not come as a surprise: as is clear from the respective papers, as far as Busy Beaver proofs are concerned, the process of finding the proof and the proof itself are very much intertwined. It is during the process of reducing the number of holdouts (the process of the proof itself) that new (variants of) types of infinite loops are discovered, that computer programs need to be refined or that new programs need to be written,...
Even though the initial intertwinement between the process of finding the proof

[16] See Sec. 3 for more details.

[17] Of course, to say that no traces at all can be found in published mathematical papers of the process of finding and the practice underlying a proof, is caricatural.

and the proof itself is in a certain way trivial,[18] the fact that the Busy Beaver proofs are represented and communicated in terms of their discovery processes is an important (but of course not exclusive) feature of these proofs. So why chose this strategy? Brady gives the answer [12]:

> While not all the exploratory activities are reproducible, the runs shown [...] can be reproduced, so that by utilizing the techniques described in this paper the proof can be corroborated

In other words, information on the process of finding the proof is provided in order for the reader-mahematician to be able to verify the proof and see if it does not contain errors. This is a very important feature as it is not only a local strategy against the problem of hardware and software errors in CaP's (See Sec. 3), but also a way to "convince" other mathematicians of the result: even though they do not have all the details they have enough information to convince themselves of the proof.

2.3.3 Man-computer interaction and the machine's responsibility

A last feature that needs to be mentioned here is the fact that the respective proofs result from a complex process of man-computer-interactions.
In the first stages of the Busy Beaver proofs the process of man-computer interaction is relatively simple. The human work is strictly separated from the computer work. First, there is the theoretical and human idea of tree generation which is translated into executable computer code. The computer then generates the reduced class of Turing machines. There is only one moment of interaction: when the human translates the theoretical ideas to the computer and asks the computer to execute them. Here, one could say that the machine's role is a very passive one. It is a mere calculation, following an order. It is not very involved, it is hardly responsible for the actual result. This is probably also one of the main reasons why the computer is hardly mentioned by Radó, Lin, Brady and Machlin and Stout in describing this first stage!
In the second stage however, which is the more heuristic stage, the interactive process is far more complicated. It is through a constant process of back-and-forth interaction, using a programming language, the display and print-outs as the means of communication, the "common languages" (interfaces) through which man and computer communicate with each other, that the proof is finally found. During this process, new types of infinite loops are discovered, new programs are written or old ones are extended,...In this second stage, the computer is more actively involved in the process of (finding) the proof. Here, its contribution is also more explicitly mentioned by Radó, Lin, Brady and Machlin and

[18] From a certain point of view, it is indeed almost trivial to say that if one is "in" the process of searching for a proof, and this process ultimately results in a proof, then searching for that proof is also always "making" that proof, hence the intertwinement. However, once the proof is "found" the proof is all that needs to be represented, the proof which might be very different from the proof "as it was found". In the end, it is the proof that counts not the process that resulted in the proof

Stout. Although, during this process of interaction, it is relatively clear which kind of things are done by the computer and which are done by the human, one cannot say that both sides are strictly separated from each other, they are involved with each other and it is this involvement that results in a proof.
This aspect of mechanized mathematics, the way man and computer interact with each other and the machine's involvement in this process, is mostly neglected in the literature on this topic. One of the exceptions is Derrick H. Lehmer. He used the idea of the amount of machine involvement in and responsibility for a result to order the different reasons why a mathematician would want to add pulse circuitry to the more usual pencil-and-paper method.[19] For Lehmer, the computer's responsibility or involvement is at its highest in case of CaPs.[20] In fact, for Lehmer, computer-assisted proofs result from a true man-computer *collaboration*. He described this process as follows [27]:

> We are dealing here with a man-machine cooperative. The man furnishes to the machine the best information that he has about the proposed theorem and the sort of proof that he thinks is likely to succeed. From this you will infer correctly that the actual proof is unknown to the man. In fact he doesn't know whether the theorem is false, or, if true, whether the machine can prove it. The machine is asked to carry out the logical steps of the proof, if indeed it can, in the allotted time. You will infer from this that there are a great many steps and that they cannot be carried out by hand. Usually the steps are not only numerous but are connected in some complicated combinatorial way. Here we are exploiting not only the speed of the computer but also its logical circuitry that allows it to keep track of and to modify its own complicated program to a degree well beyond human capability. Theorems of this kind are not easy to find in those drab branches of mathematics where elaborate proofs are not the rule. However, there are infinitely many such theorems in number theory alone.

As is reflected by this quote, the idea of a man-computer collaboration does not mean that the computer is assigned some kind of "artificial intelligence", the idea that the computer is capable to simulate or really be as intelligent as a human mathematician. On the contrary, it is made explicit by Lehmer that the computer's contribution lies in doing those things we are really bad at, while the human mathematician takes care of those things the computer is bad at. The computer is involved here because it "thinks" differently than we humans do. It is an active partner in the process of finding a proof, however, a partner that

[19] See [27], pp. 745–749 and [28], pp. 118–119

[20] Another similar parameter Lehmer uses to order different usages of the computer in mathematics is the question whether the mathematician, who publishes the result that was established in some or the other way with the help of the computer, will mention the computer or not and if yes, how much responsibility he will assign to the computer. A rather extreme example in this respect are some of the papers (co-)authored by Shalosh B. Ekhad. See for instance [17]

is not human and should not be or behave like a human. In fact, it is because the computer is thinking *not* like a human mathematician that it is so good at what it does! In a way, this is the perfect collaboration: getting new results by combining different talents.[21]

3 Managing Unsurveyability, Mathematical Understanding and errors

Since the 4CT there has been a growing number of philosophical papers on CaPs, both by philosophers and mathematicians. The main question is whether or not computer-assisted proofs like that of the 4CT change our understanding of what a proof is. I.e., do CaPs have any fundamental impact on the epistemology of mathematical proofs and thus, ultimately, mathematical knowledge? There are three main problems in this context.

A. The problem of unsurveyability. One of the most frequently discussed problems with respect to CaPs (basically, the 4CT) is the problem of unsurveyability. The problem comes down to what was once called the falling-tree conundrum [14]: Has a tree fallen if no one can hear it? Has a theorem been proved if no one can read its proof, surveying every one of its details? Besides the *possibility* of a lengthy theoretical part (as is the case for the 4CT and the sphere packing problem) and thousands of lines of code that need to be surveyed and reviewed in order to convince oneself of the proof, CaPs involve millions of computations that, practically, *cannot* be surveyed by a human. This problem has several consequences. First of all, as we do not know all the details of the proof, as we have not "followed" the proof in all its details, one must ask how one can still understand the proof (see problem **B.**). Secondly, as CaPs are unsurveyable one must ask how one can be sure that they are error-free, how can one rely on a proof for which one does not have all the details (see problem **C.**)?
On the basis of the inherent unsurveyability of the 4CT, Tymoczko argued that this is a new kind of proof. Because of its unsurveyability, the proof of the 4CT shows that there are a posteriori mathematical truths, truths which rely on empirical evidence. For Tymoczko, the proof of the 4CT shows that mathematics is fallible and empirical. The epistemological role Tymoczko assigns to the 4CT because of its unsurveyability has been opposed in the literature. Some have argued that the unsurveyability of CaPs does not show that mathematics is fallible and empirical (See for example [30, 43, 44]). One of the main arguments here is that Tymoczko places too much weight on the human factor of proof.

[21] This point was also made by Appel in an interview on the four-color theorem: "*The computer [...] was not thinking like a mathematician [...] The computer was using [...] these bits of knowledge it had in every conceivable way, and any mathematician would say, 'No, no, no, you have got to organize yourself, you have got to do it that way', but the computer wasn't doing that. And it was more successful, because it was not like a mathematician.*" (quoted in [33])

It is not because a proof is humanly unsurveyable that it is for the computer (See e.g. [4, 30, 23]). Another argument is that surveyability is actually not an essential property of proofs (see e.g. [16, 44]). Others have argued that although they agree with Tymoczko that empirical evidence is used in mathematics, this is not something novel introduced by the 4CT (See for example [16]). A different perspective on the unsurveyability of proof is offered by Shanker [42]: he claims that a proof is only a proof when it is humanly surveyable, hence, CaPs like the 4CT cannot be considered as proofs.

B. The problem of mathematical understanding. This problem is very closely related to problem **A.**. It basically comes down to the following question: how can we achieve mathematical understanding if part of the argument is hidden in a box? According to Bonsall [7], the fact that we do not have all the details of the proof implies that CaPs like the 4CT are not "real" mathematics. He says that: "*It is no better to accept without verification the word of a computer than the word of another mathematician.[...]We cannot possible achieve what I regard as the essential element of a proof – our own understanding – if part of the argument is hidden away in a box*" Halmos shares this opinion [20]. For him, CaPs like the 4CT are like oracles. All you know is that the 4CT is true but you do not know why it is true. He goes on "*I feel that we, humanity, learned mighty little from the proof; I am almost tempted to say that as mathematicians we learned nothing at all. Oracles are not helpful mathematical tools.*" Related to this is the idea that theorems like the 4CT seem to have a certain arbitrariness, it is not clear why they are true exactly.[22]

C. The problem of (hardware and software) errors. Because of the unsurveyability of CaPs like the 4CT (the length of the programs and computations) these CaPs also suffer from the further defect that one cannot be sure that no errors have occurred (See e.g. [7, 45]). It is well-known that checking the correctness of a program is a very hard problem. Besides, it is known that hardware errors do occur. When very long computer programs and computations are involved, there is thus a real chance of machine and human errors (See e.g. [25]).

These three problems are very real and the debates arising from them are still open-ended. As is clear from this summary, there are reasons to accept and reasons to reject the idea that CaPs do lead to a fundamental change of the notion of proof in mathematics, and thus have a fundamental impact on the foundations of mathematics.
Also in case of the Busy Beaver proofs these problems are real. In fact, one can say that, by definition, CaPs will always suffer from these problems. In the end, they are humanly impractical so some part of them must be unsurveyable

[22]The idea that there are mathematical truths that are true for no clear reason, that are in a certain way random, has been advocated by Gregory Chaitin, one of the founders of algorithmic information theory (AIT). He uses the definition of randomness from AIT in order to make his point.

by humans, and we thus also automatically get the problem of mathematical understanding. The problem of (software and hardware) errors also seems unavoidable, although there are some recent examples that avoid this problem to some extent.
Although a detailed philosophical analysis of these three problems on a macro level is necessary in order to come to terms with the impact of CaPs on the notion of proof, it is also interesting to see how these problems pop-up in different CaPs,[23] and, more importantly, how they are or can be dealt with locally. In the end, as these three problems are real, they need to be managed with in some or the other way on the more local level of finding, representing and communicating a proof. Tracing these "smaller" changes can at least help to clarify the changes CaPs bring about on the level of the practice itself.
So how how do these three problems occur in the context of the Busy Beaver proofs and how are they managed?[24] In a certain sense, these proofs are maybe not as unsurveyable as the original 4CT since the theoretical part of the proofs are relatively short.[25] However, the proofs remain unsurveyable. First of all, they involve a lot of human work of which hardly any of the details are published. For example, the 210 holdouts in the Kopp proof are all proven to be cases of infinite loops by hand. Clearly, if the proofs of these 210 holdouts would have been included in the published accounts of these proofs, tens of pages would need to be added, including the print-outs of the behavior of these 210 holdouts. Secondly, the proofs involve very long (unpublished computer) code. Finally, and this is an essential feature of CaPs, the proofs involve thousands if not millions of computations that result in thousands of different small proofs determining for each of the Turing machines whether or not they will halt. Taking these three aspects together, it is clear that Busy Beaver proofs are unsurveyable.[26] Since the problem of mathematical understanding and insight is very closely related to the problem of unsurveyability, it also occurs in case of the Busy Beaver proofs. The same goes for the problem of human and computer errors (problem **C**). So, how are these problems managed in the context of the Busy Beaver proofs discussed in the previous section?

I. Managing the problem of error. In the paper by Machlin (Kopp) and Stout it is noted that:

> The work in [22] is an independent confirmation of Brady's results, which is important since the sheer volume of human and computer work involved raises the possibility of error.

[23] And not just in the one case one usually considers

[24] I will mainly focus on the paper by Brady [12] and Machlin (Kopp) and Stout[32].

[25] The original proof of the 4CT involved not only millions of computations and computer code that initiated them but also a considerable amount of other "text". As is explained by Appel and Haken: "*This leaves the reader to face 50 pages containing text and diagrams, 85 pages filled with almost 2500 additional diagrams, and 400 microfiche pages that contain further diagrams and thousands of individual verifications of claims made in the 24 lemmas in the main sections of text.*" [2].

[26] Of course, the first two features are not exclusive for nor essential to CaPs.

Brady makes a similar comment, pointing at the necessity of independent verification:

> Proofs of "correctness" of the programs used are not practical. Independent verification is the only means we currently have at our disposal.

As is clear, Brady, Machlin (Kopp) and Stout are very much aware of the problem of (hardware, software and human) errors. Indeed, because of the length of the actual proof (the programming, the execution of the code in several stages (!) and the human work to prove the final holdouts) the chances that an error has occurred increase, and the result might thus be false. Furthermore, the unsurveyability of the proof also makes it impossible to check every detail of the proof and conclude that the proof is for 100% watertight. There are several ways to deal with these problems. The best solution I know of to avoid errors is the use of interactive theorem provers like HOL to attain formal proofs that have been checked completely by the computer.[27] The "method" mentioned by Brady, Machlin and Stout to reduce the chance that (human or machine) errors have occurred that of independent verification. As was explained in Sec. 2.3 the description of the process of the proof, can also be understood as a strategy to allow other mathematicians to check the correctness of the proof.
Although the method of independent verification can never lead to complete certainty, it is often one of the more efficient methods available. Even if one cannot exclude the possibility of errors in the Busy Beaver proofs, the fact that the cases $\Sigma(m), S(m), m < 5$ were verified, not only reduces the chances of errors but, maybe even more, says something about the way mathematicians think they should deal with one of the typical problems of CaPs.[28]

II. Managing unsurveyability. As is explained by Brady in the next-to-last quote of Sec. 2.2, "*keeping track of what resembled a large scientific experiment became a major task in itself.*" For this reason, Brady wrote more than 18 programs for, among other things, "*various housekeeping purposes*". In other words, it is clear that the length of the whole proof was really a problem which made it necessary to implement certain strategies to deal with the complexity of the proof.
The problem of unsurveyability is also a problem if one needs to write down and communicate the results one has found. One needs to find a good format to present the proof in, despite its unsurveyability. Contrary to the famous papers

[27] For example, the last version of the 4CT by Gonthier is such a proof. Also Thomas Hales has started the FlySpeck project in order to produce a completely formalized and computer-checked proof of the sphere packing problem in a reaction to the fact that after a team of referees had worked on the proof for 15 years they concluded that they were only 99% sure that it contains no errors. For a philosophical discussion of formal computer-checked proofs related to the problem of unsurveyability and error, see [4].

[28] It should be pointed out that they are not the only ones to emphasize the significance of independent verification. For example, in [25] Lam points out that: "*[the proof of the non-existence of the projective plane of order 10] is only an experimental result and it desperately needs an independent verification, or, better still, a theoretical explanation.*".

by Appel and Haken, or that by Hales, the published papers on Busy Beaver proofs considered here are relatively short despite their alleged unsurveyability. One of the reasons for this is that not that many details are provided in the respective published versions of the proofs.[29] This not only concerns the computer programs and the actual computations. For example, only the main types of infinite loops are described in detail, despite the different kinds of variants discovered for each of the types. Also, the details of the proofs of the holdouts that were proven by hand are not provided. To the sceptical reader, this can only add to the mistrust one must have in this kind of proofs, making them even less suitable for review than for example Thomas Hales' proof of the sphere packing problem. However, from another point of view, one could say that providing an explicitly short "summary" of a CaP, skipping a lot of the details, is just another way to deal with the unsurveyability and thus the impossibility of communicating every one of the details of the proof. By dealing with this problem in this way, the "persuasiveness", the argumentative power of these proofs, as they are communicated to the reader, functions differently. Even though not all the details are provided, the reader is given enough information to understand how the proof works. The papers describing the proofs provide a general descriptions of the programs, how they are used for the tree reduction, for the detection of infinite loops or the exploration of the behavior of the Turing machines studied, the order in which these programs were executed,...This way of communicating and writing down CaPs, a method which is in a certain way not new for mathematics,[30] is an important possibility for dealing with the inherent unsurveyability of these proofs. The unsurveyability of course remains, and the proof will not be represented in all its details. However, the problem is dealt with by developing a style, a method of representing these proofs that still allows them to be published in a reasonable amount of pages and be reviewed in a reasonable amount of time and still allows the reader to understand the main techniques of the proof. Of course, the reviewer will need a certain amount of trust if he/she does not want to go through the trouble of reconstructing every detail of the proof.[31] He/she has to trust not only the machine, but also, and maybe even more, the mathematician.

The question of what to include and what not to include in a paper describing a CaP is a very intricate one and needs further consideration. However, if CaPs are ever to become part of the common discourse of mathematics, published papers of proofs that need over 100 pages will have to be the exception rather than the rule, else, the communication of mathematical knowledge will become practically impossible.

[29] The technical note describing Brady's proof in more details and Kopp's PhD containing the proof are of course lengthier.

[30] How often does one not find something written like "*The details of the proof are left to the reader*"? because it is considered that the techniques needed for the proof are known.

[31] The reviewing of CaPs is indeed a difficult problem that has become very apparent in the context of Hales' proof of the sphere packing problem. In this respect it is interesting to read the statement by the editors of the *Annals of Mathematics* concerning CaPs (http://annals.math.princeton.edu/EditorsStatement.html).

III. Managing the problem of understanding. A problem that is very closely related to the unsurveyability of CaPs is the problem of mathematical insight and understanding, the question of the explanatory power of CaPs. The same question must be posed in the context of the Busy Beaver proofs. Indeed, in how far does e.g. Kopp's CaP provide an understanding of the fact that of $\Sigma(4) = 13$? From a certain point of view, the answer to this question is that the Busy Beaver proofs do not really provide an understanding of the facts proven. Since large portions of the proof are "generated" by the computer without any human mathematician having surveyed this part of the proof we can never fully understand why $\Sigma(4) = 13$. This problem seems to become even worse if one has not "found" the proof but has merely read the descriptions of the proofs by Radó, Lin, Brady and Kopp. However, to conclude on this basis that these Busy Beaver proofs do not provide any insight or understanding whatsoever, that, to put it in Halmos' words, we learned nothing at all, seems a case of throwing the baby out with the bath water.

Clearly, because of the problem of unsurveyablity, it is indeed impossible to understand every detail of the proof. However, this does not exclude understanding on a more general level, on the level of the structure of the proof and the feasibility of the methods used. In order to understand this, we need to return to the actual proofs.

As was explained in Sec. 2.2 the Busy Beaver proofs discussed proceed in two main stages. The first stage concerns the reduction of the number of Turing machines to be considered by implementing certain theoretical considerations, normalization being the main technique. Clearly, these theoretical methods are explained and one can understand why they work. Although this first set of reductions is achieved by the computer, one cannot say that, given a specific machine that has been eliminated, one cannot understand why it has been eliminated. Furthermore, although one does not have every one of the details in order to understand why only 603,712 and not 603,711 holdouts remain in case of Kopp's tree normalization program, one can understand how the program works and why so many machines are eliminated.

In the second stage of the proof the idea is to differentiate between halting and non-halting machines by using several different heuristic programs. First, a program is used that runs each of the holdouts for some hundred of steps and eliminates those that have halted. Then the proof goes on with the remaining machines, trying to prove that each one of them is a non-halting machine through the detection of infinite loops. Now, in each of the proofs discussed, the remaining machines "happened to be" cases of non-halting machines. Why? Because, through a complex interaction between man and the computer, it was found that all of these holdouts are cases of one of the different types of infinite loops. Clearly, this does not provide a good understanding. However, one can easily understand the general idea of the second stage of the proof. One can understand why all the remaining holdouts must be non-halting machines: during a process of exploration several different types of patterns were found. It can be proven that if a given Turing machine shows this patterns in its behavior than it will never halt. Now, for each of the holdouts it was proven – some

by hand, most through the use of heuristic infinite-loop detection programs – that they are each cases of one of these types of patterns. Hence it follows that they cannot halt. Of course, we do not have the details to understand why *exactly* 417 (and not say 320) of the 4-state machines, and only those, were identified as counters by Kopp's counter-detection program, nor do we have a complete understanding why some machine x results in, say, a Christmas tree. However, we do know what a counter is, we do know why a counter must be a non-halting machine, we do know why the counter-detection program eliminates some machines and others not (even though these might still be counters),...In other words, although we do not have all the details and we do not have a kind of complete and detailed explanation or understanding of the Busy Beaver results, one cannot neglect that the papers on he Busy Beaver proofs do convey an understanding of *how* the proof works and *why* it works on the global level. Furthermore, the proofs also give an insight on another level: they say something about the possible behavior one can expect from very simple programs, which are on the borderline of undecidability. This observation can be used in other settings. In fact, this is done in the paper by Machlin and Stout: the techniques and observations of the Busy Beaver proof are also used in the context of determining halting probabilities.
It is true that the Busy Beaver proofs do not give the kind of insight one gets from the several proofs of for instance the Pythagorean theorem. However, to call them oracles which do not provide any insight or understanding whatsoever is equally wrong.

4 Discussion

What exactly is the impact of the computer on mathematics? Has the computer really changed mathematical knowledge and the way we attain it, and, if yes, how? As was explained throughout the paper this is an intricate question which has had no definite answer. A lot depends on ones own epistemological and ontological position. Questions like: "Are CaP's fundamentally different from the usual proofs of mathematics?" or "Do CaP's show that mathematical knowledge can be a posteriori?" are fundamental issues, issues that, in my opinion, cannot be answered satisfactorily on the basis of one micro-analysis. However, whatever ones answer might be to these fundamental issues, there is one thing which cannot be denied on the basis of this and other case studies. Computers are changing the way we are doing mathematics. They are changing "mathematical practice".
As is clear from the case-analysis provided here, CaP's share some very typical features, problems and techniques. First of all, the aspect of man-computer interaction should not be underestimated. This is something completely new in mathematics, the fact that a proof is the result from a *collaboration* between a human and a non-human. It is the first time in history that a non-human is *actively* involved in the process of the proof. Although this non-human is mostly regarded as a mere quantitative help, it has led to fundamental quali-

tative changes, witness the fact that the mere increase in speed and memory added by the computer has made it possible to prove theorems that could not be or have not yet been proven without it. Besides this, the use of the computer almost naturally introduces the use of experimental methods in the process of finding a proof. These methods involve both the computer as well as the human, they arise within the complex interaction between man and machine.
A consequence of the involvement of the computer is that the representation of CaP's, the way they are communicated, is different from that of usual proofs. A proof-as-communicated is no longer a sequence of lines of symbols that represent the proof found by the mathematician. It also includes a description of what the non-human has done (the computational process), what kind of results it communicates back (the output) and how the human has communicated his/her questions to the non-human (the code). Furthermore, unlike more traditional proofs, there is a strong emphasis on the process of finding a proof in the published papers of the CaP's discussed in this paper, making explicit mention of the several experimental methods used, the complexity of this process,...As was explained in Sec. 2.3 one of the reasons for doing so is to deal on a local level with the problem of error and to allow for the reader-mathematician, even if he/she has not all the details, to follow the proof and verify it. Indeed, it is a typical property of CaP's that they all suffer from the same fundamentally philosophical problems of unsurveyability, understanding and error. These problems seem unavoidable, but are dealt with on the local level. I.e. specific strategies are implemented both in the process of finding the proof as well as in the process of finding a good form for the the proof in order to "manage" these problems.
The least one can thus conclude is that the process of finding a CaP, the method of writing it down as well as the communication of a CaP is different from usual proofs, at least, from a practical point of view. If one furthermore accepts that the way a proof is attained, its written-down form and the way it is communicated, is a determining feature of what a proof "really" is, or better, means, then this alone changes (the meaning of) proof.
It is thus clear that, practically speaking, CaP's do have an impact on mathematical practice. However, as long as CaP's, and, more generally, computer-assisted mathematics are the exception rather than the rule, the impact of the computer on mathematical practice remains rather limited. They only have a local impact, not a global one. The fact that proofs-made-flesh are different from the usual proofs of mathematics, the fact that one needs to deal with the fundamental problems of unsurveyability, understanding and error, the fact that experimental methods are needed and all the consequences these features and problems have – all these facts and effects can be discarded on the basis of the marginal role CaP's play. The impact remains limited to a few cases and one can simply classify these cases as some of those exceptional "behemoths" mathematics sometimes gives rise to. However, the question is whether this will indeed remain the case. It is only since the last 20 to 30 years that the computer and thus computer-time has become widely available to every mathematician not only physically, in the sense that every mathematician now has his/her per-

sonal computer, but also intellectually, since more and more mathematicians are getting used to using computers. Furthermore, the memory and computation speed is exponentially larger as compared to that of the early computers.[32] This wide availability of the computer and the increase in speed and memory, mere quantitative facts, could result and are in fact already resulting in more and more computer-aided mathematical results.

If the usage of computers in mathematics is becoming more general, one cannot but conclude that they have an important impact on mathematics-as-practiced on the global level. A consequence of this is that, as the present case-analysis shows, the mathematicians will have to develop new methods and they will have to deal with the problems computer-assisted mathematics gives rise to. New results will be found as a consequence.

Is this where the impact of the computer stops? It is my view that as changes on the micro-level of the practice become more and more wide-spread, fundamental changes on the macro level and thus also changes in the foundations and philosophy of mathematics, become unavoidable. The way mathematics is perceived and understood changes. Questions concerning the certainty of mathematical knowledge, the fallible character of mathematics,...will only gain in importance, as more and more mathematicians are confronted with the consequences of computer-assisted mathematics. The fact that one is no longer able to check a proof completely, the fact that part of the proof is done by a non-human, the necessity of using experimental methods, the fact that one is confronted with very long proofs,...are real problems that not only change mathematics-as-practice but also the standard of what a proof is/should be and thus, ultimately, the standard of what mathematical knowledge is/should be.

References

[1] Kenneth Appel and Wolfgang Haken, *Every planar map is four colorable. part i. discharging.*, Illonois Journal of Mathematics **21** (1977a), 429–490.

[2] ———, *The four color proof suffices*, Mathematical Intelligencer **8** (1986), no. 1, 10–20.

[3] Kenneth Appel, Wolfgang Haken, and John Koch, *Every planar map is four colorable. part ii. reducibility*, Illonois Journal of Mathematics **21** (1977b), 491–567.

[4] Konstantine Arkoudas and Selmer Bringsjord, *Computer, justification and mathematical knowledge*, Minds and Machines **17** (2007), no. 4, 185–202.

[5] Alan Baker, *Experimental mathematics*, Erkenntnis **68** (2008), no. 3, 331–344.

[32] ENIAC for example did not even store a megabyte and it could handle "only" 1000 computations in one second.

[6] Jean-Paul Van Bendegem, *The Collatz-conjecture. A case study in mathematical problem solving*, Logic and Logical Philosophy **14** (2005), no. 1, 7–23.

[7] Frank F. Bonsall, *A down-to-earth view of mathematics*, The American Mathematical Monthly **89** (1981), no. 1, 8–15.

[8] Jonathan Borwein, *Implications of experimental mathematics for the philosophy of mathematics*, Proof and other Dilemmas: Mathematics and Philosophy (B. Gold and R. Simons, eds.), Spectrum, Mathematical Association of America, 2008, pp. 33–60.

[9] Jonathan Borwein and Davis Bailey, *Mathematics by Experiment. Plausible reasoning in the 21st century*, AK Peters, Wellesley, MA, 2003.

[10] ———, *Experimentation in Mathematics: Computational paths to discovery*, AK Peters, Wellesley, MA, 2004.

[11] Allen H. Brady, *The conjectured highest scoring machines for Radó's $\sigma(k)$ for the value $k = 4$*, IEEE Transactions on Electronic Computers **EC-15** (1966), no. 5, 802–803.

[12] ———, *The determination of the value of Radó's noncomputable function σ for four-state Turing machines*, Mathematics of Computation **40** (1983), no. 162, 647–665.

[13] Tyler Burge, *Computer proof, apriori knowledge, and Other Minds. the sixth Philosophical Perspectives Lecture*, Nous, Supplement: Philosophical Perspectives, 12, Language, Mind and Ontology **32** (1998), 1–37.

[14] Andreea S. Calude, *The journey of the four colour theorem through time*, The New Zealand Mathematics Magazine **38** (2001), no. 3, 27–35.

[15] Larry Thiel Clement W. H. Lam and Stanley Swiercz, *The non-existence of finite projective planes of order 10*, Canadian Journal of Mathematics **41** (1989), 1117–1123.

[16] Michael Detlefsen and Mark Luker, *The Four-Color Theorem and Mathematical Proof*, The Journal of Philosophy **77** (1980), no. 12, 803–820.

[17] Shalosh B. Ekhad, *A very short proof of Dixon's theorem*, Journal of combinatorial theory, Series A **54** (1990), 141–142.

[18] Georges Gonthier, *A computer-checked proof of the Four Color Theorem*, Tech. report, 2004, Unpublished, Available at: research.microsoft.com/en-us/um/people/gonthier/4colproof.pdf.

[19] Thomas C. Hales, *A proof of the kepler conjecture*, Annals of Mathematics **162** (2005), no. 3, 1063–1183.

[20] Paul R. Halmos, *Has progress in Mathematics slowed down?*, The American Mathematical Monthly **97** (1990), no. 7, 561–588.

[21] Bart Van Kerkhove and Jean Paul van Bendegem, *Pi on Earth, or Mathematics in the Real World*, Erkenntnis **68** (2008), no. 3, 421–435.

[22] Rona J. Kopp, *The busy beaver problem*, Mathematical sciences, State University of New York at Binghamton, 1981, Girl name of Rona Machlin.

[23] Israel Krakowski, *The four color problem reconsidered*, Philosophical Studies **38** (1980), 91–96.

[24] Imre Lakatos, *Proofs and refutations*, Cambridge University Press, Cambridge, 1976.

[25] Clement W. H. Lam, *The Search for a Finite Projective Plane of Order 10*, The American Mathematical Monthly **98** (1991), no. 4, 305–318.

[26] Derrick H. Lehmer, *Some high-speed logic*, Experimental Arithmetic, High Speed Computing and Mathematics, Proceedings of Symposia in Applied Mathematics, vol. 15, 1963, pp. 141–376.

[27] _____, *Mechanized mathematics*, Bulletin of the American Mathematical Society **72** (1966), no. 5, 739–750.

[28] _____, *Computer technology applied to the theory of numbers*, Studies in Number Theory (W.J. Leveque, ed.), Studies in Mathematics, vol. 6, 1969, pp. 117–151.

[29] Derrick H. Lehmer, Emma Lehmer, W.H. Mills, and John L. Selfridge, *Machine proof of a theorem on cubic residues*, Mathematics of computation **16** (1962), no. 80, 407–415.

[30] Margarita Levin, *On Tymocsko's argument for mathematical empiricism*, Philosophical Studies **39** (1981), 79–86.

[31] Shen Lin and Tibor Radó, *Computer studies of Turing machine problems*, Journal of the ACM **12** (1965), no. 2, 196–212.

[32] Rona Machlin and Quentin F. Stout, *The complex behaviour of simple machines*, Physica D **42** (1990), 85–98.

[33] Donald MacKenzie, *Slaying the kraken: The sociohistory of mathematical proof*, Social Studies of Science **29** (1999), no. 1, 7–60.

[34] William McCune, *Solution of the Robbins problem*, Journal of Automated Reasoning **19** (1997), no. 3, 263–276.

[35] Pascal Michel, *Small Turing Machines and generalized busy beaver competition*, Theoretical Computer Science **326** (2004), no. 1–3, 45–56.

[36] Liesbeth De Mol, *Study of fractals derived from IFS-fractals through metric procedures*, Fractals **13** (2005), no. 3, 237–244.

[37] George Polya, *Mathematics and plausible reasoning, vol. 1: Induction and Analogy in Mathematics*, Princeton university Press, Princeton, 1954.

[38] Tibor Radó, *On non-computable functions*, The Bell System Technical Journal **41** (1962), no. 3, 877–884.

[39] ———, *On a simple source for non-computable functions*, Mathematical Theory of Automata (Jeremy Fox, ed.), Microwave Research Institute Symposia Series, vol. XII, Polytechnic Press, 1963, pp. 75–81.

[40] Neil Robertson, Daniel P. Sanders, Paul Seymour, and Robin Thomas, *The four-color theorem*, Journal of combinatorial theory, series B **70** (1997), no. 1, 2–44.

[41] Steve Seiden, *A manifesto for the computational method*, Theoretical Computer Science **282** (1998), no. 2, 381–395.

[42] Stuart G. Shanker, *The Appel-Haken solution of the the four-color problem*, Ludwig Wittgenstein: Critical Assessments (Stuart G. Shanker, ed.), Croom Helm, London, 1986, pp. 395–412.

[43] Edward R. Swart, *The Philosophical Implications of the Four Color Problem*, The American Mathematical Monthly **87** (1980), 697–707.

[44] Paul Teller, *Computer proof*, Journal of Philosophy **77** (1980), no. 12, 797–803.

[45] Thomas Tymoczko, *The four-color problem and its philosophical significance*, The Journal of Philosophy **76** (1979), no. 2, 57–83.

[46] Doron Zeilberger, *Theorem for a price: Tomorrows semi-rigorous mathematical culture*, Notices of the American Mathematical Society **40** (1993), 978–981.