Conditional Computing
(what proof checking needs from computer algebra)

Freek Wiedijk
Nijmegen University
<freek@cs.kun.nl>

Abstract

We show that current computer algebra systems are not suitable for use in proof checking, because they don't answer the right kind of question. By analyzing the calculations from a sample formalization, we find that calculations as they occur in proof checking make use of results of previous calculations (*conditions*). We list a number of such conditional calculation problems and argue that because of these conditions, current computer algebra packages like Maple and Mathematica are not able to solve them.

1 Introduction

The discipline of *proof checking* is about encoding mathematics inside a computer in such detail that the computer can check its correctness. Such a 'computer proof' can be guaranteed to be free of mistakes, and because it contains all the details of the mathematics there will be no room for ambiguity. However, coding all the details of a mathematical development inside a computer tends to get very tedious because all the steps have to be entered in full detail. Hence, automation is needed. One particular area where one would like to have computer assistance is the formalization of calculations.

Mathematics consists of two quite different kinds of activity. On the one hand there is *calculating,* and on the other hand there's *reasoning* (defining notions and proving propositions.) The experience with proof assistants is that it's much less work to formalize reasoning than it is to formalize calculation.

The field of *computer algebra* is about making the computer do symbolic calculations. So the natural thing would be to have a computer algebra system fill in the details of the calculations. However, this turns out to be more difficult than one would expect. We propose an explanation why this is the case (with a number of examples.)

There already are a number of reasons (which are reasonably well accepted) why the current generation of computer algebra packages aren't very suitable to be interfaced to proof checkers/theorem provers:

- Computer algebra systems (like Mathematica and Maple) are semantically very sloppy [1]. It is easy to derive nonsensical equations from them. Even when just trying to use them to get *answers,* one is occasionally bitten by this.

- Computer algebra systems don't know much about logic, so for problems that need *reasoning* (most problems) they're not adequate.

- Computer algebra systems just produce answers and no proofs. Hence, they can't be used to find out *why* a certain answer is what it is.

- Computer algebra systems are focused on *evaluation* instead of on *verfication.* So, it is not so easy to use them to verify equations that are given beforehand.

However, we here try to make a different point:

- Computer algebra systems are not designed to calculate when there are given equations (*conditions*) that have to be used to be able to get to the answer.

Note that this point is not about logic (the conditions are just a list of equalities and inequalities, without logical structure) and also that it is not about the difference between evaluation and verification (unconditional verification of equations is not very useful to solve problems involving conditions.)

To give a characteristic example, suppose that (in a proof of the irrationality of $\sqrt{2}$) we are given that:

$$\sqrt{p} = \frac{m}{n}$$

and that we need to establish that:

$$m \cdot m = p \cdot n^2.$$

Now a calculation that proves this is:

$$m \cdot m = \left(\frac{m}{n}\right)^2 \cdot n^2 = (\sqrt{p})^2 \cdot n^2 = p \cdot n^2.$$

We could do the first and the third steps of this calculation in a computer algebra system. However, the way this calculation is structured – to make it possible to apply the given equation – can *not* be found by a computer algebra system. This is not satisfactory, as the idea of interfacing to a computer algebra system is that we do not want to have to think about calculations like this at all.

The point of this note is that this situation is typical, and that it is the primary reason why it is difficult to use computer algebra systems from proof checkers and theorem provers.

2 A Small Mizar Article

Consider the following proof (taken from [5], line numbers added for reference), which is a Mizar[1] version of the classic proof of the irrationality of $\sqrt{2}$.

```
 reserve m, m1, n, n1, p for Nat;
 reserve i, i1 for Integer;

theorem T1:
 p is prime implies √p is irrational
proof
 assume H1: p is prime;
 then H2: p>1 by INT_2:def 5;
 then H3: p>0 by AXIOMS:22;
 assume √p is rational;
 then consider i, n such that H4: n<>0 & √p=i/n &
   for i1, n1 st n1<>0 & √p=i1/n1 holds n≤n1
  by RAT_1:25;
 H5: i=√p·n by REAL_2:73,H4;
```

[1]There is no good reference for the current version of the Mizar system, which is a very nice proof checker by Andrzej Trybulec from Poland. For instance, it doesn't have a reference manual: only [3] comes close, and it is rather hard to understand. In [6] we give a brief impression of Mizar. The website for the Mizar project is <http://www.mizar.org/>.

```
    √p ≥ 0 & n ≥ 0 by SQUARE_1:87,NAT_1:18,H3;
15  then i ≥ 0 by REAL_2:121,H5;
    then reconsider m = i as Nat by INT_1:16;
    H6: m^2 = (√p·n)^2 by H5
     .= (√p)^2·n^2 by SQUARE_1:68
     .= p·n^2 by SQUARE_1:88,H3;
20  then p|m^2 by NAT_1:def 3;
    then p|m·m by SQUARE_1:58;
    then p|m by NAT_LAT:95,H1;
    then consider m1 such that H7: m=p·m1 by NAT_1:def 3;
    n^2 = m^2/p by REAL_2:72,H3,H6
25   .= (p·m1)^2/p by H7
     .= p^2·m1^2/p by SQUARE_1:68
     .= p·p·m1^2/p by SQUARE_1:58
     .= p·(p·m1^2)/p by AXIOMS:16
     .= p·m1^2 by REAL_2:62,H3;
30  then p|n^2 by NAT_1:def 3;
    then p|n·n by SQUARE_1:58;
    then p|n by NAT_LAT:95,H1;
    then consider n1 such that H8: n=p·n1 by NAT_1:def 3;
    n1<>0 by H4,H8;
35  then H9: n1>0 by NAT_1:19;
    then m1/n1 = (p·m1)/(p·n1) by REAL_2:55,H3
     .= m/n by H7,H8
     .= √p by H4;
    then H10: n ≤ n1 by H4,H9;
40  p·n1>1·n1 by REAL_2:199,H2,H9;
    then n>n1 by H8;
    hence contradiction by H10;
end;
```

In normal language this proof becomes something like this:

> We are going to prove that if p is a prime number then $\sqrt{p}$ is irrational (lines 3–5). For suppose that it is not, then $\sqrt{p}$ is rational (line 9) and can be written as a fraction m/n (line 10). Moreover, this fraction can be chosen to be simplified, so with minimal n (line 11).
>
> Now from $m^2 = p \cdot n^2$ we get that m^2 is divisible by p (lines 17–20) and because p is prime, this implies that m is divisible by p (lines 21–22), and so we can write $m = p \cdot m'$ (line 23). Then $n^2 = p \cdot m'^2$ (lines 24–29) and so for the same reasons we also can write $n = p \cdot n'$ (lines 30–33). But then $m'/n' = m/n$ (lines 36–38) and because $n' < n$ we get a contradiction with the fact that m/n was simplified (lines 39–42).
>
> So this means that the assumption that $\sqrt{p}$ was rational can't be true, and therefore that $\sqrt{p}$ is irrational.

The Mizar proof itself is almost completely straight-forward. The only unexpected element is that the divisibility conditions have to be expressed for natural numbers, as `NAT_LAT:95`, the relevant lemma in the Mizar library:

```
      for i st i is prime holds
for m,n holds  i | m · n implies (i|m or i|n);
```

is (unnecessarily) restricted to natural numbers. This explains the 'cast' to a natural number (an object of type `Nat`) in line 14.

3 Taking Out the Calculations

Now this Mizar proof contains a number of 'calculations' that are rather low-level. For instance the deduction of $n^2 = p \cdot m'^2$ (which uses the '`.=`' construction) in lines 22–27:

$$n^2 = \frac{m^2}{p} = \frac{(p \cdot m')^2}{p} = \frac{p^2 \cdot m'^2}{p} = \frac{(p \cdot p) \cdot m'^2}{p} = \frac{p \cdot (p \cdot m'^2)}{p} = p \cdot m'^2$$

seems quite involved. Apart from these calculations, the proof is almost exactly on the level of the informal English language proof.

It seems natural to look what happens when one isolates these calculations in a number of 'calculation lemmas'. The statements of these lemmas then turn out to be (where `m`, `m1`, `n`, `n1` and `p` are declared to be natural numbers and `i` is delared integer):

```
C1: n<>0 & √p=i/n implies i≥0;
C2: n<>0 & √p=m/n implies m·m=p·n^2;
C3: p>1 & m·m=p·n^2 & m=p·m1 implies n·n=p·m1^2;
C4: n<>0 & n=p·n1 implies n1>0;
C5: n<>0 & m=p·m1 & n=p·n1 & √p=m/n implies √p=m1/n1;
C6: p>1 & n1>0 & n=p·n1 implies n>n1;
```

With these calculations supplied, the proof of `T1` becomes (where the × signs mark the lines where the lemmas are applied):

```
theorem T1:
 p is prime implies √p is irrational
proof
 assume H1: p is prime;
 then H2: p>1 by INT_2:def 5;
 assume √p is rational;
 then consider i, n such that H4: n<>0 & √p=i/n &
   for i1, n1 st n1<>0 & √p=i1/n1 holds n≤n1
  by RAT_1:25;
×  i≥0 by C1,H4;
 then reconsider m = i as Nat by INT_1:16;
×  H6: m·m=p·n^2 by C2,H4;
 then p|m·m by NAT_1:def 3;
 then p|m by NAT_LAT:95,H1;
 then consider m1 such that H7: m=p·m1 by NAT_1:def 3;
×  n·n=p·m1^2 by C3,H2,H6,H7;
 then p|n·n by NAT_1:def 3;
 then p|n by NAT_LAT:95,H1;
 then consider n1 such that H8: n=p·n1 by NAT_1:def 3;
×  H9: n1>0 by C4,H4,H8;
×  √p=m1/n1 by C5,H4,H7,H8;
 then H10: n≤n1 by H4,H9;
×  n>n1 by C6,H2,H8,H9;
 hence contradiction by H10;
end;
```

which is shorter than the original, and much more on the level of abstraction of the natural language version.

The statements of the calculation lemmas are almost exactly determined by the way they are used. If one omits all quantified conjuncts from the premisses of the

lines that use them, one gets something close to the conditions from the statement of the lemma. For instance, the line in which lemma `C3` is used:

```
... n·n=p·m1^2 by C3,H2,H6,H7; ...
```

refers to:

```
H2: p>1;
H6: m·m=p·n^2;
H7: m=p·m1;
```

so the statement for that line would be:

```
p>1 & m·m=p·n^2 & m=p·m1 implies n·n=p·m1^2
```

which happens to be exactly the statement of lemma `C3`.

The variables in these calculation lemmas are typed, but this is only used for deducing that a variable `n` that has type `Nat` satisfies `n≥0`. Apart from this the type of the variables is not relevant, and the lemmas could have been quantified over variables of type `Real` just as well.

4 Current Computer Algebra

There's no easy way to verify the truth of calculation lemmas like this with current computer algebra systems (like Macsyma, Maple [2] or Mathematica [7]) because those systems don't have a way to enter and use the conditions that have to be used in the calculations.

(The Mathematica system can have conditions on parameters of integrals with `Assumptions`, and it can check for conditions when applying rewrite rules with `Condition`, but those mechanisms are not suitable for the general use that we are writing about here.)

The basic mechanism most general purpose computer algebra packages use to manipulate expressions is *rewriting.* The rewrite system that it applies can be modified by the user, but it does not depend on the expression that is being evaluated. When one presents 'condition' equations for solving a problem, those equations somehow have to be combined with the rewrite system. There seems no easy way to do that, as both sides of these equations can be quite intricate, and in order to be able to use them a lot of formula rearrangement might be necessary. The manipulations necessary for that strongly depend on the shape of the condition, and are not in a form that is easily implemented by a rewrite system.

5 Some More Examples

If we are sloppy about the 'domain conditions' of partial functions like division and square root, and omit the premisses which are only present for that, and furthermore if we replace typing information like $n : \mathrm{N}$ by inequalities like $n \geq 0$, then the calculation lemmas from section 3 become:

$$n \geq 0\ ,\ \sqrt{p} = i/n \ \vdash\ i \geq 0$$

$$\sqrt{p} = m/n \ \vdash\ m \cdot m = p \cdot n^2$$

$$p > 1\ ,\ m \cdot m = p \cdot n^2\ ,\ m = p \cdot m' \ \vdash\ n \cdot n = p \cdot m'^2$$

$$n' \geq 0\ ,\ n \neq 0\ ,\ n = p \cdot n' \ \vdash\ n' > 0$$

$$m = p \cdot m'\ ,\ n = p \cdot n'\ ,\ \sqrt{p} = m/n \ \vdash\ \sqrt{p} = m'/n'$$

$$p > 1\ ,\ n' > 0\ ,\ n = p \cdot n' \ \vdash\ n > n'$$

These are the kind of questions that we would like to see solved automatically. More examples in this format (also from [5]) are:

$$x = \sqrt{2} \ \vdash\ (x^x)^x = 2$$

$$k + 1 \leq n \ \vdash\ \binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$$

$$n - k = 0 \ \vdash\ \binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$$

$$k > n \ \vdash\ \binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$$

$$\binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k} \ \vdash\ \binom{n}{k+1} n^{-(k+1)} = \frac{1}{k+1} \cdot \binom{n}{k} n^{-k} \cdot \frac{n-k}{n}$$

$$\frac{k}{n} \geq 0 \ \vdash\ \left|\frac{n-k}{n} - 1\right| = \frac{k}{n}$$

$$\lim_{n\to\infty} \binom{n}{k} n^{-k} = \frac{1}{k!} \ \vdash\ \lim_{n\to\infty} \left(\frac{1}{k+1} \cdot \binom{n}{k} n^{-k} \cdot \frac{n-k}{n}\right) = \frac{1}{(k+1)!}$$

$$x = i/n\ ,\ n = m + 1 \ \vdash\ n! \cdot x = i \cdot m!$$

$$x = \frac{1}{n+1}\ ,\ \frac{n!}{(n+k+1)!} \leq x^{k+1} \ \vdash\ \frac{n!}{(n+(k+1)+1)!} \leq x^{(k+1)+1}$$

$$n \geq 2\ ,\ x = \frac{1}{n+1} \ \vdash\ \frac{x}{1-x} < 1$$

Some of these examples might seem silly, but they actually appear naturally in the proofs and all are quite a bit of work when doing them in Mizar.

Note that some conditions in these examples aren't informative. For instance, the relation:

$$\binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$$

is provable without conditions too (but because $\binom{n}{k}$ is defined with a case split on $k \leq n$, it would involve doing logic, and the point of this note is that the inability of computer algebra systems to do logic is *not* the real problem.) Also note that conditions can be derivable, like:

$$\lim_{n\to\infty} \binom{n}{k} n^{-k} = \frac{1}{k!}$$

However, because the conditions from these example calculations are ones that were used in the Mizar proof, we feel that an automated system also should be allowed to refer to them. And really, as the examples become more involved, it might be unrealistic to require a solution without using superfluous conditions. For instance:

$$\lim_{n\to\infty} \left(\frac{1}{k+1} \cdot \binom{n}{k} n^{-k} \cdot \frac{n-k}{n}\right) = \frac{1}{(k+1)!}$$

is true without the condition, but maybe too difficult for an automated system to solve all by itself. (Mathematica 3.0 isn't able to compute the left hand side of this equality and complains about 'essential singularities' of the Γ function.)

6 Concluding Remarks

The balance between user intervention and automation in a proof checker/theorem prover is a delicate one. If there's not enough automation present then using the system becomes very tedious, while if there's too much automation then one is not able to control the system well enough. Ideally one would like the 'conscious' parts of a proof to be entered by the user of the system (so that the user can keep control of the proof), while the 'unconscious' parts of the proof are handled automatically. We feel that the kind of calculational inferences that we describe here are of the second kind, and should be handled by an automated computer algebra system.

What this note doesn't present, and what we are looking for, is an approach for automatically doing the conditional calculations as presented in this paper. Once such an approach has been found, interfacing computer algebra systems and theorem proving systems will become quite interesting.

There is a difference between having a system decide whether an inference is true, and having a system present a chain of calculation steps that show *why* the inference is true. We feel that the first problem is much harder than the second: if an algorithm can be found that determines whether a conditional calculation problem is solvable, distilling an algorithm from it that gives an explicit solution should be relatively easy.

Acknowledgements Thanks to Henk Barendregt, Wieb Bosma and Herman Geuvers for valuable comments on this note.

References

[1] M. Beeson. Using nonstandard analysis to ensure the correctness of symbolic computations. *International Journal of Foundations of Computer Science*, 6(3):299–338, 1995.

[2] M. Monagan, K. Geddes, K. Heal, G. Labahn, and S. Vorkoetter. *Maple V Programming Guide for Release 5*. Springer-Verlag, Berlin/Heidelberg, 1997.

[3] M. Muzalewski. *An Outline of PC Mizar*. Fondation Philippe le Hodey, Brussels, 1993. URL: `<http://www.cs.kun.nl/~freek/mizar/mizarmanual.ps.gz>`.

[4] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley/Los Angeles, 1951.

[5] F. Wiedijk. Irrationality of e. *Journal of Formalized Mathematics*, 11, 1999. MML Identifier: `IRRAT_1`.

[6] F. Wiedijk. Mizar: An impression. Unpublished,
URL: `<http://www.cs.kun.nl/~freek/mizar/mizarintro.ps.gz>`, 1999.

[7] S. Wolfram. *The Mathematica book*. Cambridge University Press, Cambridge, 1996.

A Mathematica Syntax

To show that the problems from section 5 involve expressions in the domain of computer algebra, here are the same examples in the syntax of the Mathematica system [7]:

```
In[1]:= Implies[n>=0 && Sqrt[p]==i/n, i>=0]
In[2]:= Implies[Sqrt[p]==m/n, m*m==p*n^2]
In[3]:= Implies[p>1 && m*m==p*n^2 && m==p*m1, n*n==p*m1^
        2]
In[4]:= Implies[n1>=0 && n!=0 && n==p*n1, n1>0]
In[5]:= Implies[m==p*m1 && n==p*n1 && Sqrt[p]==m/n, Sqrt
        [p]==m1/n1]
In[6]:= Implies[p>1 && n1>0 && n==p*n1, n>n1]
In[7]:= Implies[x==Sqrt[2], (x^x)^x==2]
In[8]:= Implies[k+1<=n, Binomial[n,k+1]==(n-k)/(k+1)*
        Binomial[n,k]]
In[9]:= Implies[n-k==0, Binomial[n,k+1]==(n-k)/(k+1)*
        Binomial[n,k]]
In[10]:= Implies[k>n, Binomial[n,k+1]==(n-k)/(k+1)*
         Binomial[n,k]]
In[11]:= Implies[Binomial[n,k+1]==(n-k)/(k+1)*Binomial[n,
         k], Binomial[n,k+1]*n^(-(k+1))==1/(k+1)*(
         Binomial[n,k]*n^(-k))*(n-k)/n]
In[12]:= Implies[k/n>=0, Abs[(n-k)/n-1]==k/n]
In[13]:= Implies[Limit[Binomial[n,k]*n^(-k),n->Infinity]=
         =1/k!, Limit[1/(k+1)*(Binomial[n,k]*n^(-k))*(n-k
         )/n,n->Infinity]==1/(k+1)!]
In[14]:= Implies[x==i/n && n==m+1, n!*x==i*m!]
In[15]:= Implies[x==1/(n+1) && n!/(n+k+1)!<=x^(k+1), n!/(
         n+(k+1)+1)!<=x^(k+1+1)]
In[16]:= Implies[n>=2 && x==1/(n+1), x/(1-x)<1]
```

Note that these problems don't restrict themselves to the basic set of the comparison relations `Equal`, `Unequal`, `Less`, `LessEqual`, `Greater` and `GreaterEqual` together with the basic arithmetical operations `Plus`, `Subtract`, `Times` and `Divide` (which is a pity, as Tarski has shown in [4] that the first order theory of these operations on the real numbers is decidable.) Even the first examples refer to the `Sqrt` operation, while the others also refer to `Power`, `Factorial`, `Binomial`, `Abs`, `Limit` and `Infinity`.