Article Submitted to Journal of Symbolic Computation

Automatic Derivation of the Irrationality of e

MICHAEL BEESON*

Department of Mathematics and Computer Science
San Jose State University

Abstract

As part of a project on automatic generation of proofs involving both logic and computation, we have automatically generated a proof of the irrationality of e. The proof involves inequalities, bounds on infinite series, type distinctions (between real numbers and natural numbers), a subproof by mathematical induction, and significant mathematical steps, including correct simplification of expressions involving factorials and summing an infinite geometrical series. Metavariables are instantiated by inference rules embodying mathematical knowledge, rather than only by unification. The proof is generated completely automatically, without any interactive component.

1. Introduction

The irrationality of e can be stated as $q > 0 \rightarrow |p/q - e| > 0$, where p and q are variables over natural numbers. This theorem was first proved by Euler in 1737. It has now been proved by a machine (or perhaps I should say, by a program). To prove this theorem automatically, the program must start with a slightly different form of the goal: $q > 0 \rightarrow \exists C(|p/q - e| \geq C/q! > 0)$. Without the existential variable C, the program cannot get started. Without the denominator $q!$ it also cannot succeed. But given this form of the theorem, it does proceed completely automatically to find a proof. (The denominator $q!$ gives it the "hint" to multipy by $q!$.) The proof it finds is similar to the usual proof (see for example (13), Chapter I) but employs a slightly different estimate to bound a certain infinite series. A certain inequality involving factorials is needed in the course of the proof; the program finds a proof of this inequality by mathematical induction. The inductive proof requires some not-quite-straightforward algebraic

*Research partially supported by NSF grant number CCR-9528913.

manipulations to make use of the induction hypothesis; the program also finds these steps automatically. The heart of the proof involves expressing a certain quantity (in this case expressed by an infinite series) as a sum of two parts, one of which is an an integer and the other of which can be shown to be between 0 and 1. It follows that the quantity is not zero, and indeed can be estimated from below. This principle is a variation on the basic principle that an integer known to be positive must be at least 1.

The instantiation that the program finds for the variable C involves the denominator q. A human mathematician can "refine the estimates" to find a constant B independent of p and q ($B = 1/2$ will do) such that $|p/q - e| > B/(q+1)!$. This is beyond the present capacity of the program. (It also is not presented in books that present the proof of the irrationality of e.)

This paper explains the rules of inference used by the prover, and then explain how the prover constructs this particular proof. An appendix contains the actual printout of the proof as produced by the program. The reader wishing more information about the program can consult (1), where the first results with this theorem-proving program are described. Those results were automatically-generated epsilon-delta proofs of continuity of specific functions. It was because of the application to epsilon-delta proofs that the program was named *Weierstrass*, after one of the pioneers of the epsilon-delta method. However, the name seems less appropriate now, as an expanded version of the prover demonstrates its capabilities in number theory as well as analysis.

2. Context of this Research

Mathematics consists of logic and computation, interwoven in tapestries of proofs. "Logic" is represented by the manipulation of phrases (or symbols) such as *for all x*, *there exists an x*, *implies*, etc. "Computation" refers to chains of formulas progressing towards an "answer", such as one makes when evaluating an integral or solving an equation. Typically computational steps move "forwards" (from the known facts further facts are derived) and logical steps move "backwards" (from the goal towards the hypothesis, as in *it would suffice to prove*. The mixture of logic and computation gives mathematics a rich structure that has not yet been captured, either in the formal systems of logic, or in computer programs. The computer program that produced the proof reported on here demonstrates an approach to this problem. The general features of this approach have been described in (1). The research involves two computer programs, *Mathpert* and *Weierstrass*. The former has been reported on elsewhere in detail (3; 4; 5): it contains implementations of over two thousand mathematical operations, together with logical apparatus to keep track of assumptions that may be required or generated by those operations. *Mathpert* (as in "Math Expert") uses these operations to provide a computerized environment for learning algebra, trigonometry, and calculus.

The second program, *Weierstrass*, is built upon a simple backwards-Gentzen

theorem prover, described in (6) (but re-implemented in C). To this logical backbone has been added a set of control structures, or if you like, implementations of special inference rules, to facilitate the proper control of logical and computational steps. These control structures operate at the top level of *Weierstrass*, but the computational steps themselves can use, in principle, anything that has been implemented in *Mathpert*, which is all of high-school algebra, trigonometry, and one-variable calculus including limits, differentiation, and (textbook-style) integration, as well as a good many techniques for rewriting inequalities, and a few advanced algorithms, such as the Coste-Roy algorithm (9), based on Sturm's theorem, for determining whether polynomials have roots in given intervals. The implementations of these operations in *Mathpert* are logically correct, so that they can be used in *Weierstrass* without the risk of inconsistency that would accompany the similar use of *Mathematica*, *Maple*, or *Macsyma*.† Simplification in *Mathpert* interacts with the list of current assumptions; assumptions can be used in simplification, and simplification can generate new assumptions. A more detailed description of *Weierstrass* can be found in (1).

The driving idea of this research program is that finding mathematical proofs requires expert knowledge of hundreds of special inference rules. An inference rule typically encapsulates knowledge of how to prove theorems of a certain form or in a certain context. For example, in (1) we gave special inference rules that are used for bounding given quantities in terms of other quantities. These rules enabled *Weierstrass* to automatically generate epsilon-delta proofs of the continuity of certain specific functions, but they are not special-purpose rules–bounding or "estimating" expressions is a fundamental activity in analysis. Now, in this paper we offer further evidence for this view of the nature of mathematics: we add to *Weierstrass* a few simple inference rules concerning estimates for infinite series, and the principle that a positive quantity which is an integer must be at least one. The underlying logic already possessed the ability to distinguish variables of type **integer** and **real**. We also add mathematical induction and another new inference rule to aid in controlling the transitivity law for inequalities. These few inference rules, together with the elementary mathematics available from *Mathpert*, enable the general logical mechanism of *Weierstrass* to find a proof of the irrationality of e, which is a non-trivial theorem, usually presented in an upper-division number theory course.

There are at present some fifteen mathematical inference rules in *Weierstrass*, in addition to the logical (Gentzen) rules and mathematical induction. It is noteworthy that all mathematical knowledge needed for the proof of the irrationality of e is embedded in these rules and in the simplification laws implemented in *Mathpert*. That is, the input file for this proof contains only the goal; no axioms special to this proof need to be provided. This strategy of embodying math-

†To assert that the implementations are logically correct is not to say that this logical correctness has been proved. What we mean is simply that any failure of logical correctness would be a programming error, which could be fixed without altering the program design.

ematical knowledge in inference rules is the key to our success: the inference rules can use mathematical knowledge to instantiate metavariables, rather than relying on unification. Unification (even the more sophisticated versions of it) is a very primitive way to instantiate a metavariable. It is hopelessly inadequate for dealing with inequalities: try proving $\exists x(0 < x < 1)$ by a unification-based algorithm.

Related work includes *Analytica* (8), which is a theorem-prover linked to the computational facilities of *Mathematica*, but deals only with quantifier-free proofs. This means also metavariable-free proofs; even though most of the work on the irrationality of e is quantifier-free, the use of metavariables and control mechanisms for instantiating them is crucial, so it will not be possible to reproduce this work in *Analytica*. The prover Nqthm (7) has proved some impressive theorems of number theory, including the law of quadratic reciprocity, but like *Analytica*, works best with free-variable proofs, and in particular, lacks the rules of inference introduced here, so it cannot prove the irrationality of e.[‡]

Some work on the verification in HOL of proofs involving computation is presented in (11), but this does not involve proof generation. Otter (12) can do some kinds of computation using rewrite rules and AC unification, but these are nowhere near adequate for the kinds of simplification needed in this proof. The strength of Otter lies in its proof-search capabilities. The rules of inference used here are sophisticated enough that there is almost no searching involved in the proof. Each step is either simplification (forward reasoning) or is an inference step dictated by the form of the goal, which turns out to succeed.

A scheme for regarding computation as defining an equivalence relation on formulas and defining deduction on the equivalence classes is set forth in (10). The prover discussed in this paper does not fit into this framework, since computations can both use the assumptions and generate new ones.

3. Nature of the Proofs Produced by *Weierstrass*

To avoid confusion, some discussion of nature and purpose of the proofs generated by *Weierstrass* is necessary. *Weierstrass* produces (internally) a proof-object, which can be displayed or saved in more than one form. The intention is to produce a proof that can be read and checked for correctness by a human mathematician; the standard to be met is "peer review", just as for journal publication. The proof object produced does not represent a completely formal proof in a specified formal system of the type traditionally studied by logicians. It fails to meet this standard because some of the steps are calculations carried out by algorithms, for example, factoring polynomials or "simplifying". One could, of course, easily specify a formal system allowing such steps, and this is in essence what the code for *Weierstrass* does. But the steps labeled "simplify" are hard to

[‡]Presumably one could *add* these rules to Nqthm. We are not speaking here of some limitation on all possible extensions of Nqthm.

translate into traditional logical systems, because what seems to a human mathematician a very simple "simplification" can be hard to prove in a traditional logical system. This is part of the reason that no previous computer program has produced such proofs: most such programs insist on producing formal proofs. If a machine to produce humanly-readable proofs, simplification steps *must* be allowed. Then one has three choices: (1) Settle for computer-generated proofs whose correctness is judged by human beings, as in Weierstrass. (2) Supply, once and for all, a formal proof of the *result* of each of the (1800 or so) possible simplification steps, e.g. a proof that $x^2 + 2xy + y^2 = (x + y)^2$, and instantiate these proofs for each simplification actually used. This would turn Weierstrass' proofs into formal ones. (3) Demand in addition that all algorithms involved have their correctness proved within the system. This is a higher standard, which *Weierstrass* cannot meet, and would not be met even by (2). Human mathematicians do not meet standards (2) and (3) either, so it seems reasonable to demand of a machine intended as a prototype "mathematician's assistant" that it should meet the standards required for journal publication, instead of a higher standard.

4. Inference rules used in *Weierstrass*

In this section we describe the new rules, and some variations on old rules, that have been added to *Weierstrass* since the publication of (1). The discussion will make it apparent that these rules are general in nature, rather than *ad hoc*, and that they embody mathematical ideas and techniques that are intuitive and can be applied to mathematical arguments in different areas. That is, they are not special to the example of the irrationality of e. Of course, there are probably several hundred such rules encapsulating the mathematical knowledge of a beginning graduate student, and the selection of these particular rules was motivated by the experiment of seeing what was required for the irrationality of e.

4.1. Types

Weierstrass accepts typed variables, as in $\forall x : \mathbf{N}(x \geq 0)$. It also accepts type "judgements" such as $x : \mathbf{N}$ as atomic formulae. The latter is necessary to support the "variable type" of Feferman's applicative theories (which are not used in the present example), and the former is very useful for efficient theorem-proving. The two are connected by means of an inference rule that permits the inference of $n : A$ (from no premises) when n is a variable of type A. Type information for the atomic types (in this case, just $\mathbf{N}$, $\mathbf{Z}$, and $\mathbf{R}$) is immediately accessible in the (implemented) variable itself. Type information for compound types (such as $\mathbf{N} \to \mathbf{N}$ or the types of Feferman's theories) is kept in a separate table.

An important feature of this prover is that simplification is applied to propositions as well as to mathematical expressions. Propositions expressing type judge-

ments are no exception. For example, the type expression $q!/n! : \mathbf{Z}$ (which expresses that $q!/n!$ is an integer) reduces to $n \leq q$. Another example of a type expression that occurs in the proof of irrationality of e is $p(q-1)! : \mathbf{Z}$. This will reduce to **true** if $p : Z$ and $(q-1)! : \mathbf{Z}$ reduce to **true**. The former will reduce to **true** since p is a variable of type $\mathbf{N}$. The latter will reduce to **true** if $(q-1)! : \mathbf{Z}$ does, but this reduces to $1 \leq q$. However, since the assumptions can also be used in simplification, if $0 < q$ is in the assumption list, *Weierstrass* is able to use the type information that q is of type $\mathbf{Z}$ to reduce $1 \leq q$ to **true**. Thus $p(q-1)! : \mathbf{Z}$ can be reduced to **true** by simplification alone. No logical inference is required. When debugging this proof, this simplification failed at first, because until then, I had forgotten to put the hypothesis $q > 0$ in the statement of the theorem.

Reducing type expressions involving series: $\sum_n^q q!/n!$ is an integer, and this also can be deduced without inference rules, by simplification alone. To deduce this, we simplify the expression $q!/n! : Z$; but to simplify this to **true**, we need the assumption $n \leq q$. In previous publications the technique has been described by which *Mathpert* makes temporary assumptions while simplifying an expression such as an indexed sum or definite integral, so that such assumptions are indeed available when required. This technique is called the "binders" technique, since it applies when expressions involving variable-binding operators are simplified.

Subtypes (type embedding) are supported by *Weierstrass*. For example, one can infer $a : \mathbf{Z}$ from $a : \mathbf{N}$. The system of atomic types includes $\mathbf{N}$, $\mathbf{Z}$, $\mathbf{R}$, $\mathbf{Q}$, and $\mathbf{C}$, although only $\mathbf{N}$, $\mathbf{Z}$, and $\mathbf{R}$ are involved in the irrationality of e. More complicated types, such as function types or comprehension types, are not used in this example.

4.2. Simplification of factorials and infinite series

Certain simplifications needed in the proof were not provided by calls to *Mathpert* code, and new simplification rules were introduced in *Weierstrass* to meet these needs. For example, the laws of the factorial function are not considered elementary enough that *Mathpert* should use them when a student chooses "simplify". I have in mind here such laws as $(n+1)! = (n+1)n!$ and the corresponding "factorial cancellation laws", such as $(n+1)!/(n+1) = n!$ and $(n+1)!/n! = (n+1)$. These laws need to be applied to fractions that contain other factors in numerator and denominator, in an arbitrary order (as with many simpler rules used in *Mathpert*). But for pedagogical considerations they could just as well have been included with other more "algebraic" laws. An example of their use is in verifying the inequality $2n! \leq (n+1)!$, which simplifies to $2 \leq n+1$ and then to $1 \leq n$.

Another "simplification" needed in the proof is "recognizing" and summing an infinite geometric series. Again, there are only pedagogical reasons why this step is not performed by the "simplify" command of *Mathpert*, and so had to be added specially to *Weierstrass*.

On the other hand, it is essential in this example that e be expanded into an

infinite series. Clearly we do not want to expand e into a series every time we see e in a problem; it is done only as a last resort when we cannot get anywhere by other means.

4.3. Lower bounds on infinite series

The only rule of this kind presently implemented in *Weierstrass* is this: To prove a series is positive, prove its general term is positive. Specifically, this rule takes the following form:

$$\frac{c \leq k, \Gamma \Rightarrow 0 < a_k}{\Gamma \Rightarrow 0 < \sum_{k=c}^{\infty} a_k}$$

There is a similar rule for finite series, and a similar rule for $\leq$ in place of strict inequality. Of course one could also formulate a version of the rule in which the hypothesis requires only that the general term be nonnegative and at least one term is positive, but this is not required in the example of the irrationality of e.

This rule could have been implemented just as well as a simplification rule (applying to the inequality in the conclusion) as an inference rule. For that matter, it could have been included as an axiom and treated as special to this proof; in that case the logical mechanisms would have handled this type of inference.

4.4. Upper bounds on infinite series

To bound a series from above, use the comparison test to estimate the series from above by a series whose sum is known. The simplest form of this rule would take the following form:

$$\frac{c \leq k, \Gamma \Rightarrow |a_k| \leq b_k}{\Gamma \Rightarrow \sum_{k=c}^{\infty} a_k \leq \sum_{k=c}^{\infty} b_k}$$

However, this simple form of the rule would be useless. We want to use this principle, for example, to prove that $\sum_{k=c}^{\infty} a_n < 1$, by finding a suitable comparison series whose sum is less than 1. Therefore this rule must be used in combination with the transitivity of equality. Control of transitivity is a recurring theme in automated deduction, and here is one contribution to it:

$$\frac{c \leq k, \Gamma \Rightarrow |a_k| \leq b_k \qquad \Gamma \Rightarrow \sum_{k=c}^{\infty} b_k < d}{\Gamma \Rightarrow \sum_{k=c}^{\infty} a_k \leq d}$$

This rule is much more interesting, since the question arises as to how, when trying to apply this rule in reverse, the comparison series b_k is to be determined.

There is an even more basic question: how can this rule be properly expressed? In the first version of the rule, a_k and b_k are just some terms of the language containing the variable k. In the second form, however, if b is to be determined, we would appear to need a variable b, and that variable would have to be of type $\mathbf{N} \to \mathbf{R}$. The language of *Weierstrass* would support such a formulation, but that would leave it to unification to find a series with a known upper bound. Even if higher-order unification were implemented in *Weierstrass* it would not be helpful unless there were a list of axioms giving upper bounds on known series.

Instead, this rule is included specifically for geometric series $\sum_{k=c}^{\infty} b_k$. A geometric series is constructed with ratio $1/2$ and first term $b_c = a_c$. In the future, this can be extended to other types of known-summable series. This seems to correspond fairly well to our intuitive idea of a mathematician's "bag of tricks" for bounding a series from above.

A difficulty arises when we want to prove a strict inequality. Of course the rule works if we replace $\leq$ by $<$ in both premise and conclusion; but since in the cases of interest, we are using a comparison series with the same first term, we will not be able to establish a strict inequality in the premise. The rule is of course not correct with $<$ in the conclusion and $\leq$ in the premise, but it is correct if we add the additional premise that the second term of the series satisfies a strict inequality. This form of the rule has been put into the program.

4.5. Controlling transitivity

It is a constant struggle in automated deduction to permit all desired uses of transitivity (of equality or inequality) while preventing infinite regress or combinatorial explosion. Here is another rule along these lines. We call it the "transitivecancel" rule. (In this rule, Av means A multiplied by v.)

$$\frac{\Gamma, A \leq B \Rightarrow C = Av \qquad v : \mathbf{N} \qquad \Gamma, A \leq B \Rightarrow Bv \leq D}{\Gamma, A \leq B \Rightarrow C \leq D}$$

In the implementation of this rule, unification is not sufficient to find v. Instead, algebraic cancellation is applied to C/A for the various A occurring in inequalities in the antecedent, until a cancellation is found producing a term $v = C/A$ which is "obviously" a nonnegative integer (for example, a variable of type $\mathbf{N}$, or a number, or a product of integers). Without the restriction that v should be an integer, the rule is of course still valid (if v is nonnegative), but it led to fruitless lines of attack. As implemented, then, only one new subgoal is generated, since the first two hypotheses will be verified by simplification.

This rule is very useful in proofs by induction, where $A \leq B$ is the induction hypothesis $\phi(n)$, and $C \leq D$ is the induction goal $\phi(n+1)$. If the induction variable n occurs linearly in an exponent, then C/A may cancel, producing a quotient without n in the exponent, so $Bv \leq D$ may be simple to prove. Such a situation occurs in the proof of the irrationality of e, when proving an estimate on the terms of an infinite series by induction. A simple example of its use would be in the proof by induction of $2^n \leq (n+1)!$. Here we try to derive

$2^n \leq (n+1)! \Rightarrow 2^{n+1} \leq (n+1+1)!$. Cancelling $2^{n+1}/2^n$ we find $v = 2$, so the new goal is $2(n+1)! \leq (n+1+1)!$. Using factorial simplification as discussed above this reduces to $2 \leq (n+1+1)$, which is easily verified. Indeed *Weierstrass* can prove $2^n \leq (n+1)!$ by induction in just this way.

4.6. Mathematical induction

The principle of mathematical induction is easily added to almost any theorem-prover. The delicate issues are the selection of an induction variable and the decision to try mathematical induction. *Weierstrass* has rather crude rules for both of these, but they are sufficient for the estimates needed in the course of the proof of the irrationality of e. The rule of induction in *Weierstrass* is as follows:

$$\frac{\Gamma \Rightarrow \phi(c), \qquad c \leq n, \phi(n), \Gamma \Rightarrow \phi(n+1)}{c \leq n, \Gamma \Rightarrow \phi(n)}$$

where n does not occur in Γ or in c. That is, the basis case is $n = c$ since $c \leq n$ is in the assumption list. Here n is a variable of type $\mathbf{N}$ or $\mathbf{Z}$.

This formulation of the rule implies an answer to the "delicate issues" mentioned above. Namely, induction on n will not be tried unless there is an inequality of the form $c \leq n$ in the assumption list, with n of type $\mathbf{N}$ or $\mathbf{Z}$, and in that case, the first such n will be selected as the induction variable.

The usual formulation without $c \leq n$, namely

$$\frac{\Gamma \Rightarrow \phi(0) \qquad \phi(n), \Gamma \Rightarrow \phi(n+1)}{\Gamma \Rightarrow \phi(n)}$$

is used in *Weierstrass* when there is exactly one variable of type $\mathbf{N}$ or $\mathbf{Z}$ in ϕ. However, this plays no role in the proof of irrationality of e.

4.7. The Zbound rule

The logical, as opposed to mathematical, heart of the proof is the interplay between the type $\mathbf{Z}$ and the type $\mathbf{R}$. The simplest principle connecting these two types is the principle that if $n : \mathbf{Z}$ and $0 < n$ then $1 \leq n$. That is, there are no integers between 0 and 1. A more quantitative formulation of this principle is the following rule, $\dfrac{m : \mathbf{Z} \qquad 0 < \alpha \qquad \alpha < 1}{\min(\alpha, 1-\alpha) \leq |m + \alpha|}$ which we call the "Zbound rule".

We have omitted to write the antecedent Γ since it is the same in both the hypotheses and conclusion.

This rule is only useful once the quantity we are trying to bound below has been written in the form $m + \alpha$ with $m : \mathbf{Z}$. For this purpose there is a special simplification rule which tries to write its input in that form. In particular, this rule will break an infinite series into a finite sum and an infinite "tail" if it can be calculated (by simplification) that the terms up to a certain point are integers. In the application to the irrationality of e, the series has the form $\sum_{n=0}^{\infty} q!/n!$,

so the natural place to break the series is at $n = q$, since for $n \leq q$, we have $q!/n! : \mathbf{Z}$.

Of interest for the future are various refinements of this rule. If one wants the program to be able to "refine the estimates" as mentioned in the introduction, one may wish to consider rules such as

$$\frac{m : \mathbf{Z} \qquad 0 < A \leq \alpha \qquad \alpha \leq B < 1}{\min(A, 1 - B) \leq |m + \alpha|}$$

Here, in the application of the rule, A and B would usually be new metavariables, with certain variables "forbidden" to their values, so that we would be searching e.g. for bounds independent of q. This rule has been implemented, but there are difficulties to be overcome before such a rule can be made to work properly, and no such rule is currently used.

5. Points of interest in the proof of irrationality of e

In this section we go through the proof step-by-step, bringing out the points about the proof that are of special interest for automated deduction, and showing how the inference rules given above are used to find the proof.

The starting point is

$$\forall p : \mathbf{N} \forall q : \mathbf{N} (q > 0 \rightarrow \exists C (|e - p/q| \geq C/q! \wedge C > 0).$$

The logical apparatus strips off the quantifiers and assumes $q > 0$ (puts it in the antecedent). The existential variable C becomes a metavariable which must eventually be instantiated. Putting the clause $C > 0$ last saves us from a fruitless attempt to prove the inequality with $C = q$, since $C > 0$ will unify with the hypothesis $q > 0$. We then start to verify $|e - p/q| \geq C/q!$. Here is where the "hint" given by the $q!$ term is used: the inequality simplifies by clearing the denominator. The prover is able to deduce the side condition that $q! > 0$ since this inequality reduces to **true**. The $q!$ is then multiplied into the absolute value (for reasons discussed below), yielding the goal

$$|q!e - pq!/q| \geq C.$$

Factorial simplification is then used to reduce $q!/q$ to $(q - 1)!$. This is not as trivial as it seems, since this simplification requires $q > 0$, so the assumptions must be used. Next the prover expands e in an infinite series–but only after trying everything else! (Since, as discussed above, expanding e in a series *should* be a last resort.) The form of this expression, however, is such that nothing else in the repertoire generates a noticeable false start. (One has to wonder, though, if that would still be true if *Weierstrass* contained a couple of hundred widely-varied mathematical inference rules.) This leads to

$$|q! \sum_{n=0}^{\infty} 1/n! - p(q - 1)!| \geq C.$$

Now it is a problem to get the $q!$ multiplied into the series, since generally simplification will pull constants *out* of series, rather than push them in. This is therefore not performed by the "simplify" operation of *Mathpert*. Perhaps it should be, on the grounds that it is a good idea to multiply constants involving factorial into series whose general term contains factorial; but at present, it is not done. The same applies to the step mentioned above, of multiplying the $q!$ term into the absolute value. Both these steps are performed under the direct control of *Weierstrass* rather than by *Mathpert*'s simplification. The general principle here seems to be that if you cannot get your theorem proved by simplifying as usual, you probably should expand things that were contracted, multiply in things that were factored out, etc. At any rate that is what *Weierstrass* does; there is no fine-tuning about "multiply factorials into series that contain factorials". It is simply, if nothing else works, push constants into absolute values and series instead of factoring them out. We arrive at

$$|\sum_{n=0}^{\infty} q!/n! - p(q-1)!| \geq C.$$

Now the prover starts to prepare for the eventual use of the Zbound rule, by breaking the sum into two parts, separating off the initial terms that it can see are integers:

$$|\sum_{n=0}^{q} q!/n! + \sum_{n=q+1}^{\infty} q!/n! - p(q-1)!| \geq C.$$

Since simplification is able to simplify the type expression that says the finite sum is of type **Z** to **true**, the Zbound rule is now applied, writing this inequality in the form $|m+\alpha| > C$, where $m = \sum_{n=0}^{q} q!/n! - p(q-1)!$ and $\alpha = \sum_{n=q+1}^{\infty} q!/n!$. This unifies the metavariable C with $\min(\alpha, 1-\alpha)$. (Of course, the prover does not introduce a new letter α and a definition as we do here for convenience and legibility.) It remains to verify $0 < \alpha$ and $\alpha < 1$. $0 < \alpha$ is easily verified, since the general term of the series is positive. The hard part is $\alpha < 1$.

The prover attacks $\alpha < 1$ using the comparison test. The proof presented in (13) does not use a geometric series, but rather the series for e. *Weierstrass* uses a geometric series as discussed above. This seems to be an improvement over the original proof! Namely, using the series for e causes two problems: First, it is necessary to factor out the first term $q!/(q+1)! = 1/(q+1)$, and nothing in the program will cause that to happen; and it seems that only an *ad hoc* rule could force it to happen. Second, and more important, the estimate that comes out is $e/(q+1)$, and it is not even true that this is bounded by 1 for all q. Instead, one has to separately prove $e < 3$ and introduce a case split, treating the cases $q = 1$ and $q = 2$ separately. All this is left implicit in (13), but nothing in *Weierstrass* would be able to make this case split. In other words, if we added the series for e to the comparison test inference rule to be tried before the geometric series, and

also forced it to factor out the first term following (13), it would fail to prove $e/(q+1) < 1$ and hence fail to find the estimate in (13).

Not to worry: the geometric series estimate does succeed. The geometric comparison series with ratio 2 and first term (when $n = q+1$) matching that of α has general term $2^{-n+q+1}/(q+1)$. This generates the subgoal

$$q+1 \leq n \Rightarrow \frac{q!}{n!} \leq \frac{2^{-n+q+1}}{q+1}.$$

After simplification this inequality becomes

$$2^{n-q-1}(q+1)! \leq n!$$

The prover then proves this by induction. Since the assumption $q+1 \leq n$ is in the antecedent, the induction variable is selected as n, and the basis case is $n = q+1$. The basis case then simplifies to **true**. The induction step yields to the "transitivecancel" rule discussed above. Namely, the induction step's goal is

$$2^{n+1-q-1}(q+1)! \leq (n+1)!$$

Regarding this as the $C \leq D$ in the conclusion of the transitivecancel rule, and the induction hypothesis as the $A \leq B$ term in that rule, we have $C/A = v = 2$. The new goal (generated by the transitivecancel rule) is $2n! \leq (n+1)!$, which reduces using factorial simplification to $2 \leq n+1$, hence to $1 \leq n$, which follows immediately from $q+1 \leq n$.

Since we want to use the comparison test to establish a strict inequality, we must also verify that strict inequality holds in the comparison for at least one term; *Weierstrass* chooses the second term, where $n = q+2$, and the inequality to be proved is

$$\frac{q!}{(q+2)!} < \frac{2^{-(q+2)+q+1}}{q+1}$$

which simplifies to *true*, completing the comparison test. Note that factorial simplication, not just polynomial algebra, is needed here.

This completes the verification of the hypotheses of the Zbound rule. At this point the metavariable C is unified with $\min(\alpha, 1-\alpha)$, and the prover moves on to the last goal, namely $0 < C$. Originally I had added the inference rule

$$\frac{X < A \qquad X < B}{X < \min(A, B)}$$

But this turned out to be superfluous since *Mathpert* already simplifies $X < \min(A, B)$ to $X < A \wedge X < B$. Here is another case where the line between calculation and inference is blurred.

The final interesting point is that we have already verified both of these goals. Yet originally, there was nothing in *Weierstrass* to enable it to re-use those deductions. It would simply repeat them! The general problem here is, how do we recognize a lemma when we see one? What subproofs should be accorded the

status of lemmas, and recorded, and referred to when the same goal comes up again later? For the present, *Weierstrass* accords that status to the hypotheses of the Zbound rule, on the grounds that almost always we will be wanting to prove that the bound produced is positive. The prover keeps a list of lemmas; it records sequents in this list on command, and the first thing it does when trying to prove a goal is to compare the goal to the list of lemmas. At present, only the subgoals generated by the Zbound rule are recorded as lemmas. This mechanism permits the proof to complete in a few lines after the Zbound rule inference completes, rather than nearly doubling its length.

References

[1] Beeson, M.: Automatic Generation of epsilon-delta Proofs of Continuity, in: Calment, J., and Plaza, J. (eds.) *Artificial Intelligence and Symbolic Computation*, Lecture Notes in Artificial Intelligence **1476**, pp. 67–83, Springer-Verlag, Berlin/Heidelberg/New York (1998).

[2] Beeson, M.: Unification in lambda-calculus with if-then-else, in: Kirchner and Kirchner (eds.): *Automated Deduction–CADE-15, Proceedings of the 15th International Conference on Automated Deduction, Lindau, Germany, July 1988*, pp. 103–118. Lecture Notes in Artificial Intelligence **1421**, Springer-Verlag, Berlin/Heidelberg/New York (1998).

[3] Beeson, M.: Logic and computation in *Mathpert*: an expert system for learning mathematics, in: Kaltofen, E., and Watt, S. M. (eds.), *Computers and Mathematics*, pp. 202–214, Springer-Verlag, Berlin/Heidelberg/New York (1989).

[4] Beeson, M.: Design principles of Mathpert: software to support education in algebra and calculus, in: Kajler, N. (ed.) *Computer-Human Interaction in Symbolic Computation*, pp. 89-115, Springer-Verlag, Wien (1996).

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

[6] Beeson, M.: Some applications of Gentzen's proof theory in automated deduction, in: Shroeder-Heister, P., *Extensions of Logic Programming*, Lecture Notes in Computer Science **475**, pp. 101–156, Springer-Verlag, Berlin/Heidelberg/New York (1991).

[7] Boyer, R., and Moore, J.: *A Computational Logic Handbook*, Academic Press, San Diego (1988).

[8] Clarke, E., and Zhao, X.: Analytica: a theorem prover in Mathematica, in: Kapur, D. (ed.), *Automated Deduction: CADE-11 - Proc. of the 11th*

International Conference on Automated Deduction, pp. 761–765, Springer-Verlag, Berlin/Heidelberg (1992).

[9] Coste, M., and Roy, M. F.: Thom's lemma, the coding of real algebraic numbers, and the computation of the topology of semi-algebraic sets, in: Arnon, D. S., and Buchberger, B., *Algorithms in Real Algebraic Geometry*, Academic Press, London (1988).

[10] Dowek, G., Hardin, Th., and Kirchner, C., Theorem proving modulo, Rapport de Recherche 3400, INRIA (1998).

[11] Harrison, J., and Thery, L.: Extending the HOL theorem prover with a computer algebra system to reason about the reals, in *Higher Order Logic Theorem Proving and its Applications: 6th International Workshop, HUG '93*, pp. 174–184, Lecture Notes in Computer Science **780**, Springer-Verlag, Berlin/Heidelberg/New York (1993).

[12] McCune, W.: Otter 2.0, in: Stickel, M. E. (ed.), *10th International Conference on Automated Deduction* pp. 663–664, Springer-Verlag, Berlin/Heidelberg/New York (1990).

[13] Siegel, C. L., *Transcendental Numbers*, Annals of Mathematics Studies **16**, Princeton University Press, Princeton, New Jersey (1949).

A. Appendix: Verbatim copy of the output

Weierstrass produces an internal proof object, which can be viewed and saved in either "trace view" or "proof tree view". There is an option to save the trace view as a TEX file. Here is the trace-view file, as produced and typeset by Weierstrass:

The goal is

$$\forall p : N, \forall q : N, \exists C, \left(0 < q \to \left|e - \frac{p}{q}\right| \geq \frac{C}{q!}, 0 < C\right)$$

Trying

$$\forall q : N, \exists C, \left(0 < q \to \left|e - \frac{p}{q}\right| \geq \frac{C}{q!}, 0 < C\right)$$

Trying

$$\exists C, \left(0 < q \to \left|e - \frac{p}{q}\right| \geq \frac{C}{q!}, 0 < C\right)$$

Trying

$$0 < q \to \left|e - \frac{p}{q}\right| \geq \frac{\min\left(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\right)}{q!}, 0 < \min\left(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\right)$$

Assuming $0 < q$
Trying

$$\left|e - \frac{p}{q}\right| \geq \frac{\min\left(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\right)}{q!}, 0 < \min\Big(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\Big)$$

Trying

$$\left|e - \frac{p}{q}\right| \geq \frac{\min\left(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\right)}{q!}$$

Simplifying, it would suffice to prove

$$\min\Big(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\Big) \leq \left|e - \frac{p}{q}\right| q!$$

Since $0 \leq q!$
it would suffice to prove

$$\min\Big(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\Big) \leq \left|\Big(e - \frac{p}{q}\Big) q!\right|$$

By the distributive law
it would suffice to prove

$$\min\Big(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\Big) \leq \left|eq! - \frac{pq!}{q}\right|$$

Using

$$\frac{pq!}{q} = p(q-1)!$$

it would suffice to prove

$$\min\Big(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\Big) \leq |eq! - p(q-1)!|$$

Using the series definition of e
it would suffice to prove

$$\min\Big(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\Big) \leq \left|\Big(\sum_{k=0}^{\infty} \frac{1}{k!}\Big) q! - p(q-1)!\right|$$

Simplifying, it would suffice to prove

$$\min\Big(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\Big) \leq \left|q! \sum_{k=0}^{\infty} \frac{1}{k!} - p(q-1)!\right|$$

Pushing a constant into the infinite series,
it would suffice to prove

$$\min\Big(\sum_{k=q+1}^{\infty}\frac{q!}{k!}, 1-\sum_{k=q+1}^{\infty}\frac{q!}{k!}\Big) \le \Big|\sum_{k=0}^{\infty}\frac{q!}{k!} - p(q-1)!\Big|$$

Since

$$k \le q \to \frac{q!}{k!} : Z$$

we can separate the initial integer-valued terms;
it would suffice to prove

$$\min\Big(\sum_{k=q+1}^{\infty}\frac{q!}{k!}, 1-\sum_{k=q+1}^{\infty}\frac{q!}{k!}\Big) \le \Big|\sum_{k=0}^{q}\frac{q!}{k!} + \sum_{k=q+1}^{\infty}\frac{q!}{k!} - p(q-1)!\Big|$$

Simplifying, it would suffice to prove

$$\min\Big(\sum_{k=q+1}^{\infty}\frac{q!}{k!}, 1-\sum_{k=q+1}^{\infty}\frac{q!}{k!}\Big) \le \Big|-p(q-1)! + \sum_{k=q+1}^{\infty}\frac{q!}{k!} + \sum_{k=0}^{q}\frac{q!}{k!}\Big|$$

Since

$$-p(q-1)! + \sum_{k=0}^{q}\frac{q!}{k!}$$

is an integer
it suffices to prove

$$0 < \sum_{k=q+1}^{\infty}\frac{q!}{k!}$$

and

$$\sum_{k=q+1}^{\infty}\frac{q!}{k!} < 1$$

Trying

$$0 < \sum_{k=q+1}^{\infty}\frac{q!}{k!}$$

It would suffice to prove the general term is positive.
Trying

$$\forall k, \Big(q+1 \le k \to 0 < \frac{q!}{k!}\Big)$$

Trying

$$q+1 \le k \to 0 < \frac{q!}{k!}$$

Assuming $q+1 \leq k$
Trying

$$0 < \frac{q!}{k!}$$

Hey, that simplifies to true!
Success
Discharging
Success
Success
Success
Success: that completes the proof of

$$0 < \sum_{k=q+1}^{\infty} \frac{q!}{k!}$$

Recording this result as Lemma 1
Trying

$$\sum_{k=q+1}^{\infty} \frac{q!}{k!} < 1$$

Try comparison with an easier series.
Trying

$$\forall k : Z, \left(q+1 \leq k \rightarrow \left|\frac{q!}{k!}\right| \leq \frac{q!}{(q+1)!}\left(\frac{1}{2}\right)^{k-q-1}\right)$$

Trying

$$q+1 \leq k \rightarrow \left|\frac{q!}{k!}\right| \leq \frac{q!}{(q+1)!}\left(\frac{1}{2}\right)^{k-q-1}$$

Assuming $q+1 \leq k$
Trying

$$\left|\frac{q!}{k!}\right| \leq \frac{q!}{(q+1)!}\left(\frac{1}{2}\right)^{k-q-1}$$

Simplifying, it would suffice to prove

$$2^{k-q-1}(q+1)! \leq k!$$

Trying mathematical induction on k
The basis case is when $k = q+1$
Hey, that simplifies to true!
That completes the basis case.
Now for the induction step.

Assume the induction hypothesis

$$2^{k-q-1}(q+1)! \leq k!$$

We must prove

$$2^{k+1-q-1}(q+1)! \leq (k+1)!$$

Simplifying, it would suffice to prove

$$2^{k-q}(q+1)! \leq (k+1)!$$

In view of

$$2^{k-q}(q+1)! = 2^{k-q-1}(q+1)!2$$

and the assumption

$$2^{k-q-1}(q+1)! \leq k!$$

it would suffice to prove $2k! \leq (k+1)!$
Hey, that simplifies to true!
Success
That completes the induction step.
Induction completed successfully.
Success
Discharging
Success
Success
Trying

$$\sum_{k=q+1}^{\infty} \frac{q!}{(q+1)!}\left(\frac{1}{2}\right)^{k-q-1} \leq 1$$

Summing the geometric series,
it would suffice to prove

$$\frac{\frac{q!}{(q+1)!}\left(\frac{1}{2}\right)^{q+1-q-1}}{1-\frac{1}{2}} \leq 1$$

Hey, that simplifies to true!
Success
We must prove strict inequality for at least one term.
We try the term with $k = q+1+1$
Trying

$$\left|\frac{q!}{(q+1+1)!}\right| < \frac{q!}{(q+1)!}\left(\frac{1}{2}\right)^{q+1+1-q-1}$$

Hey, that simplifies to true!
Success of comparison test
Success: that completes the proof of

$$\sum_{k=q+1}^{\infty} \frac{q!}{k!} < 1$$

Recording this result as Lemma 2
Success
Success
Trying

$$0 < \min\Big(\sum_{k=q+1}^{\infty} \frac{q!}{k!}, 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}\Big)$$

Simplifying, it would suffice to prove

$$0 < \sum_{k=q+1}^{\infty} \frac{q!}{k!}, 0 < 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}$$

Trying

$$0 < \sum_{k=q+1}^{\infty} \frac{q!}{k!}$$

This has been proved in Lemma 1
Success
Trying

$$0 < 1 - \sum_{k=q+1}^{\infty} \frac{q!}{k!}$$

Simplifying, it would suffice to prove

$$\sum_{k=q+1}^{\infty} \frac{q!}{k!} < 1$$

This has been proved in Lemma 2
Success
Success
Success
Discharging
Success
Success
Success
Success. That completes the proof.