Analytica: A Theorem Prover for Mathematica

Edmund Clarke and Xudong Zhao, Carnegie Mellon University

Analytica is an automatic theorem prover written in Mathematica for theorems in elementary analysis. The goal of the project is to use a powerful symbolic computation system to prove theorems that are beyond the scope of previous automatic theorem provers. Analytica is also able to guarantee the correctness of certain steps that are made by the symbolic computation system and therefore prevent common errors like division by a symbolic expression that could be zero. We describe the structure of Analytica and explain the main techniques it uses to construct proofs. We illustrate Analytica's power with several examples including the basic properties of the stereographic projection and a series of three lemmas, each proved completely automatically, that lead to a proof of Weierstrass's example of a continuous nowhere-differentiable function.

Current automatic theorem provers, particularly those based on some variant of resolution, have concentrated on obtaining ever higher inference rates by using clever programming techniques, parallelism, etc. We believe that this approach is unlikely to lead to a useful system for actually doing mathematics. The main problem is the large amount of domain knowledge that is required for even the simplest proofs. In this paper, we describe an alternative approach that involves combining an automatic theorem prover with a symbolic computation system. The theorem prover, which we call Analytica, is able to exploit the mathematical knowledge that is built into this symbolic computation system [Clarke and Zhao 1992]. In addition, it can guarantee the correctness of certain steps that are made by the symbolic computation system and, therefore, prevent common errors like division by an expression that may be zero.

Analytica is written in the Mathematica programming language and runs in Mathematica's interactive environment. Since we wanted to generate proofs that were as similar as possible to proofs constructed by humans, we have used a variant of natural deduction similar to UT Prover developed by Bledsoe at the University of Texas [Bledsoe 1983]. In particular, quantifiers are handled by skolemization instead of explicit quantifier introduction and elimination rules. Although inequalities play a key role in all of analysis, Mathematica is only able to handle very simple numeric inequalities. We have developed a generalization of the SUP-INF method for linear inequalities [Bledsoe 1979] that is able to handle a large class of non-linear inequalities as well. Another important component of Analytica deals with expressions involving summation and product operators. At least eight pages of rules are devoted to the basic properties of these operators. We have also integrated Gosper's algorithm for hypergeometric sums with the other summation rules, since it can be used to find closed form representations for a wide class of summations that occur in practice.

There has been relatively little research on theorem proving in analysis. Bledsoe's work in this area is certainly the best known [Bledsoe 1983, 1984]. Analytica has been heavily influenced by his research. More recently, Farmer, Guttman, and Thayer [1990] at Mitre Corporation have developed an interactive theorem prover for analysis proofs that is based on a simple type theory. Neither of these uses a symbolic computation system for manipulating mathematical formulas, however. Suppes and Takahashi [1989] have combined a resolution theorem prover with the Reduce system, but their prover is only able to check very small steps and does not appear to have been able to handle very complicated proofs. London and Musser [1974] have also experimented with the use of Reduce for program verification.

Certainly, no other theorem prover uses techniques from symbolic computation as much as Analytica. By writing our theorem prover in Mathematica, we gain access to an extensive collection of mathematical and logical functions. Moreover, we are able to manipulate very complicated formulas by using functions like Simplify and Factor. The facility that Mathematica provides the user for specifying new rewrite rules is particularly valuable. We use this feature frequently in all parts of the prover for replacing subformulas by equiv-
alent forms. The ability to generate LaTeX code for mathematical formulas automatically is quite helpful for producing readable proofs. In addition, we have exploited several Mathematica packages for specialized tasks, such as the package for trigonometric simplification. Since it is easy to add additional Mathematica packages to Analytica, our theorem prover will become more powerful as use of Mathematica grows and the system improves.

Our paper is organized as follows: In the next section we give several simple examples that illustrate the power of our theorem prover and show how it uses various symbolic computation techniques provided by Mathematica. We then present an overview of the structure of Analytica and the major techniques that it uses in constructing proofs. The following sections contain more detailed descriptions of the techniques for simplification, our use of Gospers’s algorithm, the SUP-INF method, and the treatment of summations and inequalities. The paper concludes with a discussion of some extensions that we hope to add to Analytica in the near future. The Appendix contains more complicated examples: proofs of the basic properties of the stereographic projection and Weierstrass’s example of a continuous nowhere-differentiable function.

Several Simple Examples Proved by Analytica

In each example, the input for the prover is given first. The theorem and its proof are printed by the theorem prover. Mathematica automatically generates LaTeX commands to typeset formulas involving algebraic expressions.

```
Prove[Imp[and[r > 0, a < b], a < (a + b*r)/(1 + r) < b]];
```

**Theorem:**

\[ r > 0 \land a < b \implies a < \frac{a + br}{1 + r} < b \]

**Proof:**

\[ r > 0 \land a < b \implies a < \frac{a + br}{1 + r} < b \]

reduces to

\[ 0 < r \land a < b \implies a < \frac{a + br}{1 + r} < b \]

and split case 1.1:

\[ 0 < r \land a < b \implies a < \frac{a + br}{1 + r} \]

replace expression with its lower or upper bounds

\[ 0 < r \land a < b \implies a \leq a \]

reduces to

True

case 1.2:

\[ 0 < r \land a < b \implies a + br < b \]

replace expression with its lower or upper bounds

\[ 0 < r \land a < b \implies a \leq a \]

reduces to

True

```
Prove[\text{Induction}[n, \text{sum}[2^k / (1 + x^*(2^k)), \{k, 0, n\}] = 1/(x-1) + 2^n/(1-x^*(2^n)(n+1))];
```

**Theorem:**

\[ \sum_{k=0}^{n} \frac{2^k}{1 + x^{2^k}} = \frac{1}{x-1} + \frac{2^{n+1}}{1 - x^{2^{n+1}}} \]

**Proof:**

prove by induction on \( n \)
base case with \( n = 0 \)

\[ \frac{1}{1 + x} = \frac{1}{-1 + x} + \frac{2}{1 - x^2} \]

reduces to

True

induction step

\[ \sum_{k=0}^{n} \frac{2^k}{1 + x^{2^k}} = \frac{1}{-1 + x} + \frac{2^n}{1 - x^{2^n}} \]

\[ \implies \sum_{k=0}^{n+1} \frac{2^k}{1 + x^{2^k}} = \frac{1}{-1 + x} + \frac{1 - 2^{n+1}}{1 - x^{2^{n+1}}} \]

reduces to

\[ \sum_{k=0}^{n} \frac{2^k}{1 + x^{2^k}} = \frac{1}{-1 + x} + \frac{2^n}{1 - x^{2^n}} \]

substitute using equation

\[ \sum_{k=0}^{n} \frac{2^k}{1 + x^{2^k}} = \frac{1}{-1 + x} + \frac{2^n}{1 - x^{2^n}} \]

\[ \implies \frac{1}{-1 + x} + \frac{2^n}{1 - x^{2^n}} + \left( \sum_{k=0}^{n} \frac{2^k}{1 + x^{2^k}} \right) = \frac{1}{-1 + x} + \frac{1 - 2^{n+1}}{1 - x^{2^{n+1}}} \]

reduces to

True

```
(* Closed-form representation of the Fibonacci numbers *)
F[0] := 0;  F[1] := 1;
F[n_ + k_Integer] := F[n + k - 1] + F[n + k - 2] /; k >= 2;
Prove[Induction[n, 0, 2],
F[n] == ((1+sqrt[5])^n - (1-sqrt[5])^n) / (2^n sqrt[5]);
```

**Theorem:**

\[ F(n) = \frac{(1 + \sqrt{5})^n - (1 - \sqrt{5})^n}{2^n \sqrt{5}} \]

**Proof:**

prove by induction on \( n \)
base case with \( n = 0 \)

\[ True \]

base case with \( n = 1 \)

\[ 1 = \frac{1 + \sqrt{5} - (1 - \sqrt{5})}{2 \sqrt{5}} \]

reduces to

True
induction step

\[
F(n) = \frac{-(1 - \sqrt{5})^n + (1 + \sqrt{5})^n}{2^n \sqrt{5}}
\]

\[
\wedge F(1 + n) = \frac{-(1 - \sqrt{5})^{1+n} + (1 + \sqrt{5})^{1+n}}{2 \cdot 2^n \sqrt{5}}
\]

\[
\Rightarrow F(n) + F(1 + n) = \frac{-(1 - \sqrt{5})^{2+n} + (1 + \sqrt{5})^{2+n}}{4 \cdot 2^n \sqrt{5}}
\]

substitute using equation

\[
F(n) = \frac{-(1 - \sqrt{5})^n + (1 + \sqrt{5})^n}{2^n \sqrt{5}}
\]

\[
\wedge F(1 + n) = \frac{-(1 - \sqrt{5})^{1+n} + (1 + \sqrt{5})^{1+n}}{2 \cdot 2^n \sqrt{5}}
\]

\[
\Rightarrow \frac{-(1 - \sqrt{5})^n + (1 + \sqrt{5})^n}{2^n \sqrt{5}} + \frac{-(1 - \sqrt{5})^{1+n} + (1 + \sqrt{5})^{1+n}}{2 \cdot 2^n \sqrt{5}}
\]

\[
= \frac{-(1 - \sqrt{5})^{2+n} + (1 + \sqrt{5})^{2+n}}{4 \cdot 2^n \sqrt{5}}
\]

reduces to

\[
F(n) = \frac{-(1 - \sqrt{5})^n + (1 + \sqrt{5})^n}{2^n \sqrt{5}}
\]

\[
\wedge F(1 + n) = \frac{-(1 - \sqrt{5})^{1+n} + (1 + \sqrt{5})^{1+n}}{2 \cdot 2^n \sqrt{5}}
\]

\[
\Rightarrow \frac{(1 - \sqrt{5})^n(-3 + \sqrt{5}) + (1 + \sqrt{5})^n(3 + \sqrt{5})}{2}
\]

\[
= \frac{-(1 - \sqrt{5})^{2+n} + (1 + \sqrt{5})^{2+n}}{4}
\]

rewrite as

True

An Overview of Analytica

Analytica consists of four different phases: skolemization, simplification, inference, and rewriting. When a new formula is submitted to Analytica for proof, it is first skolemized to a quantifier-free form. Then it is simplified using a collection of algebraic and logical reduction rules. If the formula reduces to true, the current branch of the inference tree terminates with success. If not, the theorem prover checks to see if the formula matches the conclusion of some inference rule. If a match is found, Analytica will try to establish the hypothesis of the rule. If the hypothesis consists of a single formula, then it will try to prove that formula. If the hypothesis consists of a series of formulas, then Analytica will attempt to prove each of the formulas in sequential order. If no inference rule is applicable, then various rewrite rules are used attempting to convert the formula to a simpler, but equivalent form. If the rewriting phase is unsuccessful, the search terminates in failure; otherwise the three phases will repeat with the new formula. Backtracking will cause the entire inference tree to be searched before the proof of the original goal formula terminates with failure. As in the case of Bledsoe’s UT theorem prover [Bledsoe 1983], we have not attempted to use a complete system of inference rules even for the purely logical fragment of our logic. Consequently, there are some formulas which are valid in the set of axioms that our theorem proving system (implicitly) provides, but cannot be proved by using Analytica.

Below, we show how the various phases of Analytica are used in conjunction with Mathematica to prove a simple theorem.

Prove \[ \text{imp}(\text{and}(a \neq 0, x \neq y), a \neq 0 \Rightarrow x + y = -b/a) \]

Theorem:

\[ a \neq 0 \wedge x \neq y \Rightarrow a x^2 + b x + c = 0 \Rightarrow x + y = -b/a \]

Proof: The skolemization phase does not change the sequent since it is quantifier free.

\[ a \neq 0 \wedge x \neq y \Rightarrow c + b x + a x^2 = 0 \Rightarrow a \neq 0 \wedge a \neq 0 \Rightarrow x + y = -b/a \]

The inference phase moves two negated atomic formulas from the left side of the sequent to the right side.

\[ c + b x + a x^2 = 0 \wedge c + b + a y^2 = 0 \Rightarrow x + y = -b/a \]

The rewriting phase causes \(-b/a\) to be moved from the right side of the last equation to the left side, and Mathematica is used to find the common denominator of the resulting expression.

\[ c + b x + a x^2 = 0 \wedge c + b + a y^2 = 0 \Rightarrow x + y = -b/a \]

The simplification phase drops the denominator since it is known to be different from 0.

\[ c + b x + a x^2 = 0 \wedge c + b + a y^2 = 0 \Rightarrow x + y = -b/a \]

The rewrite phase solves the two linear equations in the hypothesis for c. The variable c is selected since it has the simplest coefficient.

\[ c = \frac{-(b + a x)}{a} \wedge c = \frac{-(b + a y)}{a} \Rightarrow x + y = 0 \wedge a = 0 \Rightarrow x + y = 0 \]

The rewriting phase eliminates c by substitution.

\[ -(x + a y) = -(y + a x) \Rightarrow x + y = 0 \wedge a = 0 \Rightarrow x + y = 0 \]

The simplification phase drops the minus sign from both sides of the first equation.

\[ x + a y = y + a x \Rightarrow x + y = 0 \wedge a = 0 \Rightarrow x + y = 0 \]

The rewriting phase moves the right side of the first equation to the left side, and the resulting expression is factored by Mathematica.

\[ (x + y)(b + a x + a y) = 0 \Rightarrow x + y = 0 \wedge a = 0 \Rightarrow x + y = 0 \]

The simplification phase replaces an equation xy = 0 by x = 0 \iff y = 0.

\[ x + y = 0 \Rightarrow b + a (x + y) = 0 \Rightarrow x + y = 0 \wedge a = 0 \Rightarrow x + y = 0 \]

The simplification phase determines that the formula is true since each atomic formula on the left side of the sequent also appears on the right side.

True
Skolemization Phase

In Analytica (as in Bledsoe's UT Prover), we use skolemization to deal with the quantifiers that occur in the formula to be proved. Initially, quantified variables are standardized so that each has a unique name. We define the position of a quantifier within a formula as positive if it is in the scope of an even number of negations, and negative otherwise. Skolemization consists of the following procedure: Replace (∃x. Ψ(x)) at positive positions or (∀x. Ψ(x)) at negative positions by (Ψ(y, y_1, ..., y_n)) where x, y_1, y_2, ..., y_n are all the free variables in Ψ(x) and y is a new function symbol, called a skolem function. The original formula is satisfiable if and only if its skolemized form is satisfiable. Thus, X is valid if and only if X is valid where X is the skolemized form of -X [Fitting 1990]. We call -skolemize(¬f) the negatively skolemized form of f. A formula is valid if and only if its negatively skolemized form is valid. When a negatively skolemized formula is put in prefix form, all quantifiers are existential. These quantifiers are implicitly represented by marking the corresponding quantified variables. The marked variables introduced by this process are called skolem variables. The resulting formula will be quantifier-free.

This procedure can be easily expressed in Mathematica. Instances[form, pattern] returns the list of all instances of pattern that appear in form.

Skolemize[and[a_, b_], position_] := and[Skolemize[a, position, Skolemize[and[b, position]]];
Skolemize[or[a_, b_], position_] := or[Skolemize[a, position, Skolemize[or[b, position]]];
Skolemize[imp[a_, b_], position_] := imp[Skolemize[a, -position], Skolemize[b, position]];
Skolemize[seq[a_, b_], position_] := seq[Skolemize[a, -position], Skolemize[b, position]];
Skolemize[not[a_], position_] := not[Skolemize[a, -position]];
Skolemize[all[x_, a_], positive] := Skolemize[a, -> Var[UniqueName], positive];
Skolemize[all[x_, a_], negative] := Skolemize[a, -> Const[UniqueName, Instances[a, Var[_]]], negative];
Skolemize[some[i_, a_], positive] := Skolemize[a, -> Const[UniqueName, Instances[a, Var[_]]], positive];
Skolemize[some[i_, a_], negative] := Skolemize[a, -> Var[UniqueName], negative];
Skolemize[a_,_] := a;

For example, the skolemized form of the formula

(∃x. ∀y. P(x, y)) → (∃u. ∀v. Q(u, v))

is given by

P(x, y_0(x)) → Q(u_0), v),

while its negatively skolemized form is

P(x_0(y), y) → Q(u, v_0(u))

where x, y, u, and v are skolem variables, and u_0, v_0, x_0, y_0 are skolem functions. Although formulas are represented internally in skolemized form without quantifiers, quantifiers are added when a formula is displayed so that proofs will be easier to read.

Inference Phase

The inference phase is based on the sequent calculus [Gallier 1986]. We selected this approach because we wanted our proofs to be readable. Suppose that f is the formula that we want to prove. In this phase we attempt to find an instantiation for the skolem variables that makes f a valid ground formula. To accomplish this, f is decomposed into a set of sequents using rules of the sequent calculus. Each sequent has the form seq[h, c], where h is initially a conjunction and c is a disjunction of such subformulas of f. The formula f will be proved if a substitution can be found that makes all of the sequents valid. A sequent seq[h, c] is valid if it is impossible to make all of the conjuncts of h true and all of the disjuncts of c false. Analytica uses the following inference rules:

- Equation: Check if there is an equation in the conclusion that is satisfiable. The boolean procedure unifiable[a, b] determines if there is an instantiation for skolem variables that makes the terms a and b identical; unify[a, b] gives the most general such instantiation.

  Implies[seq[h, c], or[cl___, a == b, c2___]] := unify[a, b]; unifiable[a, b];

- Inequality: Check if there is an inequality in the conclusion that is satisfiable.

  Implies[seq[h, or[cl___, a <= b, c2___]], unify[a, b]; unifiable[a, b];

- Match: Check if a disjunct of the conclusion matches a conjunct of the hypothesis.

  Implies[seq[h, c], match[h, c]]; matchable[h, c];

- And-split: H → (A → B) is equivalent to (H → A) → B).

  Implies[seq[h, or[cl___, and[a, b_], c2___]], sequentialTry[seq[h, or[cl, a, c2]], seq[and[h, a], or[cl, and[b], c2]]];

  SequentialTry[s1_, s2_] := Implies[apply[Implies[s1, s2]];]

- Cases: (A → B) → C is equivalent to (A → C) → (B → (C → A)).

  Implies[seq[and[h1___, or[a, b_], h2___], c], sequentialTry[seq[and[h1, a, h2], c], seq[and[h1, or[b], h2], or[c, a]]];
Back-chain: Check if some part of the conclusion of the sequent matches the conclusion of a lemma.

\[
\text{Impl}[\text{seq} [h_0, \text{or} [c_0, \ldots, c_n, c_{1 \ldots}]]] := \\
\text{Impl}[\text{seq} [h_0, \text{or} [c_0, \text{hypothesis} [\text{ApplicableLemma} [c], c_{1}]]] /; \\
\text{ApplicableLemma} [c] \neq \text{NIL};
\]

Backtracking is often necessary in the inference phase when there are multiple subgoals, because a substitution that satisfies one subgoal may not satisfy the others. When this happens it is necessary to find another substitution for the first subgoal. The inference rules given above must be modified in this case since the inference phase for a particular subgoal terminates as soon as a satisfying substitution is found.

In order to restart the inference phase at the correct point, a stack must be added to the procedure described above. When a rule is applied that may generate several subgoals (e.g., And-split or Cases), one subgoal is selected as the current goal and the others are saved on the stack. If the current goal is satisfied by some substitution \( \sigma \), then \( \sigma \) is applied to the other subgoals on the stack and Analytica attempts to prove them. If the other subgoals are not satisfied under \( \sigma \), then Analytica returns to the previous goal and tries to find another substitution that makes it true.

The following changes are necessary for the first three inference rules. The other three inference rules do not change.

\[
\text{SetAttributes}[\text{SequentialTry}, \{\text{HoldAll}\}];
\]

\[
\text{SequentialTry}[s_1, s_2] := (\text{Push}[s_2]; \text{Impl}[s_1]);
\]

\[
\text{TryOtherBranches}[u_\_] :=
\]
\[
(\text{If}[\text{StackEmpty}[\_], \text{True}, (\text{ApplyToStack}[u]; \text{Impl}[\text{Pop}[\_]])]);
\]

\[
\text{Impl}[\text{seq} [h_\_, \text{or} [c_1, \ldots, a_\_ \Rightarrow b_\_, c_2]]] :=
\]
\[
\text{True} /; \text{unifiable} [a, b] \& \& \text{TryOtherBranches} [\text{unify} [a, b]];
\]

\[
\text{Impl}[\text{seq} [h_\_, \text{or}[c_1, \ldots, a_\_ \Rightarrow b_\_, c_2]]] :=
\]
\[
\text{True} /; \text{unifiable} [a, b] \& \& \text{TryOtherBranches} [\text{match} [h, c]];
\]

The following example illustrates the order in which the proof tree is searched. Each node is a pair \(A[B] \), where \(A \) is the current goal and \(B \) is a list of other subgoals that are on the stack. Assume that

\[
A_{\sigma_1} = A_{\sigma_2} = A_{\sigma_3} = \text{True}
\]

\[
B_{\sigma_1} = \text{false}, B_{\sigma_2} \sigma_4 = B_{\sigma_2} \sigma_5 = B_{\sigma_1} \sigma_6 = \text{True}
\]

\[
C_{\sigma_2} \sigma_4 = C_{\sigma_2} \sigma_5 = \text{false}, C_{\sigma_1} \sigma_6 \sigma_7 = \text{True}
\]

In the examples given above and in the Appendix, only the success steps are shown. All proof steps that lead to dead ends have been deleted.

**Rewrite Phase**

Five rewriting tactics are used in Analytica:

1. **When the left side of an equation in the hypothesis appears in the sequent, replace it by the right side of the equation.**

\[
\sum_{k=0}^{n} \frac{2^k}{1 + x^{2^k}} = \frac{1}{1 + x} + \frac{2 \cdot 2^n}{1 - x^{2^n}}
\]

\[
\Rightarrow \frac{2 \cdot 2^n}{1 + x^{2^n}} + \left( \sum_{k=0}^{n} \frac{2^k}{1 + x^{2^k}} \right) = \frac{1}{1 + x} + \frac{2^n}{1 - x^{2^n}}
\]

**substitute using equation**

\[
\sum_{k=0}^{n} \frac{2^n}{1 + x^{2^n}} = \frac{1}{1 + x} + \frac{2 \cdot 2^n}{1 - x^{2^n}}
\]

\[
\Rightarrow \frac{1}{1 + x} + \frac{2^n}{1 - x^{2^n}} + \frac{2 \cdot 2^n}{1 - x^{2^n}} = \frac{1}{1 + x} + \frac{1 \cdot 2^n}{1 - x^{2^n}}
\]

2. **Rewrite a trigonometric expression to an equivalent form.**

For example, given that \(a \) is an odd integer, and \( \alpha = \text{round}(a^{m} x) \),

\[
m \leq n \Rightarrow \cos(a^{m} x) + \cos(a^{m-n}(1 + n)) + (-1)^{n}(1 + \cos(a^{m-n}(a^{m} x - \alpha))) = 0
\]

**rewrite trigonometric expressions**

\[
\text{True}
\]

3. **Move all terms in equations or inequalities to left side and factor the expression.**
\[
\frac{(-1 + x_3) \left( -1 + 2x_2 + y_1^2 \right)}{\left( -1 + y_3 \right)^2} = -1 + x_3^2 + \frac{(-1 + x_3)^2 y_1^2}{\left( -1 + y_3 \right)^2} \\
\Rightarrow x_3 - y_3 = 0
\]

rewrite as

\[
\frac{2(1 - x_1)(x_3 - y_3)}{1 + y_1} = 0 \Rightarrow x_3 - y_3 = 0
\]

4. Solve linear equations.
\[
e + bx + ax^2 = 0 \land c + by + ay^2 = 0 \\
\Rightarrow x - y = 0 \lor a = 0 \lor b + a(x + y) = 0
\]
solve linear equation
\[
e = - (x(b + ax)) \land c = - (y(b + ay)) \\
\Rightarrow x - y = 0 \lor a = 0 \lor b + a(x + y) = 0
\]

5. Replace a user defined function by its definition.
\[
0 < \pi a^m b^n + (1 - ab) \text{Abs}(S(m))
\]
on-open definition
\[
0 < \pi a^m b^n + (1 - ab) \text{Abs}\left( \sum_{n=0}^{\infty} \frac{y^n}{n!} \frac{1}{\pi} \cos(\mu a^n x) + \cos(\pi a^n (x + h)/n) \right)
\]

\[
f_1 \land f_2 \Leftrightarrow \text{False} /; \text{ContainsNonNegative}([f_1 f_2]),
\]
\[
f_1 \land f_2 \Rightarrow \text{True} /; \text{ContainsPositive}([f_1 f_2]),
\]
\[
f_1 \Rightarrow f_2 \Rightarrow \text{False} /;
\]
\[
\text{ContainsPositive}([f_1 f_2]) \\
|| \text{ContainsPositive}([f_2 f_1])
\]

In a conjunction \( A \land B \), \( A \) can be simplified assuming \( B \) is true, while in a disjunction \( A \lor B \), \( A \) can be simplified assuming \( B \) is false. Conjunctions and disjunctions that involve equations or inequalities are therefore simplified by rules such as these:

\[
\text{or}([a \land b \land c \land d], \text{or}([a \land c], [a \land d]), [b \land c] /; [c \land b] \Rightarrow \text{True}, [c \land b] \Rightarrow \text{False}, [c \land b] \Rightarrow \text{False}, [c \land b] \Rightarrow \text{True}, [c \land b] \Rightarrow \text{True})
\]

The following examples show the combined power of the simplification techniques discussed above. Let's assume that \( a > 1 \), \( b > 0 \), \( ab > 1 + 3\pi/2 \).

The first example illustrates the simplification of inequalities and summations.

\[
(1 + ab) \text{Abs}(x) \text{Abs}(1 + x - \text{round}(x)) \left( \sum_{n=0}^{\infty} \frac{(-1)^{n+1} a^n b^n}{n!} \text{Abs}(a^n) \right)
\]

\[
\Rightarrow (x^n)^m < 0
\]

reduces to

\[
-a^m b^n + \sum_{n=0}^{\infty} (-1)^{n+1} b^n a^n = ab \left( \sum_{n=1}^{\infty} (-1)^{n+1} b^n \right) < 0
\]

simplify summations

\[
-a^m b^n + \sum_{n=1}^{\infty} (-1)^{n+1} b^n - ab \left( \sum_{n=0}^{\infty} (-1)^{n+1} b^n \right) < 0
\]

reduces to

\[
\text{True}
\]

The second example illustrates the simplification of an expression involving a limit.

\[
\forall n \left[ M - \lim_{n \to \infty} \left( -\frac{a^m b^n (2 + 3\pi - 2ab)}{3 (-1 + ab)} \right) < 0 \right]
\]

simplify limits

\[
-\infty < 0
\]

reduces to

\[
\text{True}
\]

**Summation**

Summations and products play an important role in symbolic computation, so we have introduced a number of special rules for dealing with them. Although most of the rules in this section are based on simple identities, Analytica is able to handle a fairly large range of summations and products. A few of the rules for summation are listed below.
The rules are partitioned into two sets, `SumSimplifyRules` and `SumRewriteRules`. The first is used to rewrite summations to simpler forms, while the second is used to rewrite summations to equivalent but not necessarily simpler forms. `SumRewriteRules` is only applied when it actually simplifies the summation. The function `simpler` is a heuristic that compares the complexity of two summation expressions. Currently, the most important parameter of this function is the relative number of summations in its arguments.

```plaintext
SimplifySummation[f_] :=
  If[simpler[///.SumRewriteRules, f],
    ///.SumRewriteRules, f///.SumSimplifyRules];
```

The rules for simplifying summations are the following:

- Calculate the summation directly when each term has the same value.
  ```plaintext
  sum[n_?NumberQ, {v_, min_, max_}] := n (max-min+1),
  ```

- Factor a constant from each term of the summation.
  ```plaintext
  sum[a_, f_, {v_, n_}] := a sum[f, {v, n}]; FreeQ[a, v],
  ```

- Merge summations with same range.
  ```plaintext
  n1_ sum[a_, range1] + n2_ sum[b_, range2] :=
  sum[(n1 a + n2 b), range],
  ```

- Append two summations with the same term and adjacent ranges.
  ```plaintext
  a_ sum[f_, {v_, n1_, n2_}] + a_ sum[f_, {v_, n3_, n4_}] :=
  a sum[f, {v, n1, n4}]; n3 - n2 == 1,
  ```

- Subtract the first or last part from a summation.
  ```plaintext
  a_ sum[f_, {v_, n1_, n0_}] + b_ sum[f_, {v_, n2_, n1_}] :=
  a sum[f, {v, n1, n2}] + b; simplify[a + b == 0],
  ```

- Change the range when the lower bound exceeds the upper bound.
  ```plaintext
  sum[f_, {v_, min_, max_}] :=
  -sum[f, {v, max+1, min+1}]; simplify[min > max]
  ```

The following rules are used to rewrite summations:

- Calculate the sum of a geometric or binomial series.
  ```plaintext
  sum[a_, {v_, min_, max_}] :=
  (a'((max+1)/v) - a'(min/v))/a'(1/v) - 1; FreeQ[a', 1/v],
  ```

- `Sum[a Binomial[n, k], {k, 0, n}] :=
  (1 + a'(1/k)) - n sum[a Binomial[n, k], {k, 0, n-1}] + FreeQ[a'(1/k), k],
  ```

- Split off additional terms in order to make the range of summation simpler.
  ```plaintext
  sum[a_, {v_, n1_, n2_ + n_Integer}] :=
  sum[a, {v, n1, n2}] + sum[a, {v, n2 + 1, n2 + n}],
  ```

- Increase or decrease the index of summation.
  ```plaintext
  sum[t_, {v_, min_, max_}] :=
  sum[t /. (v -> v+1), {v, min-1, max-1}] /;
  simpler[t /. (v -> v+1), t]
  ```

  ```plaintext
  sum[t_, {v_, min_, max_}] :=
  sum[t /. (v -> v-1), {v, min+1, max+1}] /;
  simpler[t /. (v -> v-1), t],
  ```

Gosper's Algorithm

In many examples, it would be helpful if we could obtain a closed form representation for a summation. Gosper's algorithm is able to compute such a representation for a large class of summations. Consequently, we have integrated this method into our theorem prover. A function is said to be a hypergeometric function if \( g(n+1)/g(n) \) is a rational function of \( n \). Gosper's algorithm is able to find a closed form for the series \( \sum_{k=1}^{n} a_k \) when there is a hypergeometric function that satisfies \( g(n) = \sum_{k=1}^{n} a_k \) + \( g(0) \) [Gosper 1977]. The following example illustrates how Gosper's algorithm is used in Analytica:

**Theorem:**

\[
|x| > 1 \implies \lim_{n \to \infty} \frac{\sum_{k=1}^{n} k^2 + (2x^2 + 1)k + x^2 (x^2 + 1)}{n} < \frac{1}{2}
\]

**Proof:**

\[
|x| > 1 \implies \lim_{n \to \infty} \frac{\sum_{k=1}^{n} k^2 + x^2 (1 + x^2) + k (1 + 2x^2)}{n} < \frac{1}{2}
\]

reduces to

\[
1 - |x| < 0 \implies \frac{1}{2} + \lim_{n \to \infty} \left( \sum_{k=1}^{n} \frac{1}{k + x^2 (1 + k + x^2)} \right) < 0
\]

calculate summation with Gosper's Algorithm

\[
1 - |x| < 0 \implies
\]

\[
\frac{1}{2} + \frac{1}{2 + x^2} + \frac{1}{(1 + x^2) (2 + x^2)} - \frac{1}{1 + n + x^2} < 0
\]

simplify limits

\[
1 - |x| < 0 \implies \frac{1}{2} + \frac{1}{2 + x^2} + \frac{1}{(1 + x^2) (2 + x^2)} < 0
\]

reduces to

\[
1 - |x| < 0 \implies \frac{1}{2 + x^2} < 0
\]

reduces to

\[
1 - |x| < 0 \implies 1 - x^2 < 0
\]

replace expression with its lower or upper bounds

True
Inequalities and the SUP-INF Method

Inequalities play a key role in all areas of analysis. Since Mathematica provides only limited facilities for handling inequalities, we have built several techniques into Analytica for this purpose. As mentioned above, some inequalities can be reduced in the simplification phase. However, there are valid inequality formulas, such as \((a \leq 0 \land b \leq a) \rightarrow b \leq 0\), that can not be proved by the simplification techniques alone. Other more powerful techniques are needed.

Bledsoe's SUP-INF method [Bledsoe 1979] is a decision procedure for quantifier-free linear inequality formulas. It decides whether such a formula is satisfiable or not. The procedure for a formula \(F\) is carried out in two stages. In the first stage, \(F\) is reduced to a disjunction of linear inequalities. Each disjunct is a conjunction of linear inequalities. Clearly, \(F\) is satisfiable if and only if one of the disjuncts is satisfiable. The second stage checks if any disjunct is satisfiable.

The first stage is quite simple, since it is only necessary to rewrite the formula into disjunctive normal form. Hence, we will concentrate on the second stage. In this stage, we must decide if an individual conjunction of linear inequalities is satisfiable. Suppose we want to eliminate the variable \(x_1\) from the formula

\[
F = (e_1 < 0 \land e_2 < 0 \land \ldots \land e_n < 0 \land e_{n+1} \leq 0 \land e_{n+2} \leq 0) \]

where each \(e_i\) is a linear expression of variables \(x_1, \ldots, x_n\). Since each of the \(e_i\) is linear in \(x_1\), \(F\) can be rewritten as:

\[
\bigwedge_i x_i < A_i \land \bigwedge_i B_i < x_i \land \bigwedge_i C_i < 0
\]

\[
\land \bigwedge_i x_i \leq A'_i \land \bigwedge_i B'_i \leq x_i \land \bigwedge_i C'_i \leq 0
\]

where each of the \(A_i, B_i, C_i, A'_i, B'_i, C'_i\) is a linear expression in the remaining variables \(x_2, \ldots, x_n\). It is easy to show that \(F\) is satisfiable if and only if the following formula is satisfiable:

\[
\bigwedge_{i,j} B_i < A_j \land \bigwedge_{i,j} B_i < A'_j \land \bigwedge_{i,j} B'_i < A_j
\]

\[
\land \bigwedge_{i,j} B'_i < A'_j \land \bigwedge_{i,j} C_i < 0 \land \bigwedge_{i,j} C'_i \leq 0
\]

Thus, we have reduced the original problem with \(k\) variables to a problem with \(k - 1\) variables. By continuing in this manner, we either get a contradiction or the empty formula. The original formula is unsatisfiable in the former case and satisfiable in the latter. In [Hushak 1977] this method is shown to be complete for all quantifier-free linear inequality formulas. The method does not apply directly to nonlinear inequalities, however.

Extending the SUP-INF Method

In this section we show how to generalize the SUP-INF method to certain nonlinear inequalities. We first consider the case in which all of the inequalities have the form \(a \leq b\); we will consider strict inequalities later. The basic idea behind the SUP-INF method can be stated as follows: If we want to prove \(a \leq c\), and we know from the context of the proof that \(b \leq c\), then it is sufficient to prove that \(a \leq b\). This principle can be applied to nonlinear inequalities as well, although completeness can no longer be guaranteed. We provide a method of calculating upper and lower bounds for expressions. By using these bounds it is possible to handle many of the nonlinear inequalities that arise in practice. The sets computed for the expression \(a\) are denoted by \(Upper(a)\) and \(Lower(a)\), respectively. To prove \(a \leq b\), it is sufficient to prove that there is some \(c \in Upper(a)\) such that \(c \leq b\) is true or that there is some \(c \in Lower(b)\) such that \(a \leq c\) is true.

In order to handle strict inequalities, we introduce a new function symbol \(S_i(a)\). \(S_i(a)\) is an upper bound of \(a\) if \(a\) is a strict upper bound of \(b\). Likewise, \(S_i\) is used for strict lower bounds. Thus, \(S_i(a)\) is less than or equal to \(a\) and \(S_i(b)\) is greater than or equal to \(b\). This convention permits the same code to be used for both strict and nonstrict inequalities.

There are three main ways of obtaining upper and lower bounds for expressions. The first is to use information provided by the local context of the formula (for example, information provided by an inequality in some hypothesis is used in the conclusion). The second method exploits the monotonicity of certain functions (an upper bound for a sum can be obtained by replacing one of the arguments by an upper bound for it). The third uses some bound on the value of a function (such as \(|\sin(x)| < 1\) for all \(x\)).

Among the rules for obtaining bounds from local context information are the following:

- \(Given[a < b] := AddUpperBound[a-b, SU[0]]\);
- \(Given[a = b] := AddUpperBound[a-b, 0]\);
- \(AddUpperBound[a + b, c, d] := \{AddUpperBound[a, c-b]; AddUpperBound[b, c-a]\}\);
- \(AddUpperBound[c ? Number?, a, b, c, d] := \{\}
  \quad AddUpperBound[a, b/c, \ldots, a, b/c];
  \quad AddUpperBound[a ? Number?, b/c, c, a, b/c] := \{AddUpperBound[c, ? Number?, a, b/c, c, a, b/c];
  \quad AddUpperBound[c, ? Number?, a, b, c, a, b] := \{AddUpperBound[c, ? Number?, a, b, c, a, b];
  \quad AddUpperBound[a, b, c, a, b, c, a, b] := \{AddUpperBound[a, b, c, a, b, c, a, b];

When proving \((a \leq b)\) \(c\), \(Given[Net[c]]\) is processed before the upper bounds of \(a\) and the lower bounds of \(b\) are computed.

Consider the example at the beginning of this section, \((a \leq 0 \land b \leq a) \rightarrow b \leq 0\). When our technique is applied to the right side of the formula, the local context is \((a \leq 0 \land b \leq a)\), which gives an upper bound of \(a\) to \(b\). Thus, the formula can be reduced to \((a \leq 0 \land b \leq a) \rightarrow a \leq 0\) after \(b\) is replaced by its upper bound. The resulting expression can easily be reduced to True.

If \(f\) is a monotonically increasing function, and \(a\) is an upper (lower) bound of \(a\), \(a \rightarrow b\) is an upper (lower) bound of \(f(a)\); if \(f\) is a monotonically decreasing function and \(a\) is an upper (lower) bound of \(a\), \(f(a) \rightarrow b\) is a lower (upper) bound of \(f(a)\). For example:

- \{a + b' | b' \in Upper(b)\} \subseteq Upper(a + b)
- \{ca' | a' \in Upper(a)\} \subseteq Lower(c \times a), \quad \text{if} \quad c \leq 0
- \{a' | b' \in Upper(b)\} \subseteq Upper(ca'), \quad \text{if} \quad a \geq 1
If \( f \) is bounded, i.e. for all \( x \), \( f(x) \leq M \), or \( f(x) \geq M \), \( M \) is an upper bound for \( f(x) \) and \( M \) a lower bound for \( f(x) \). For example:

\[
1 \in \text{Upper}(\sin(x)), \quad -1 \in \text{Lower}(\cos(x)), \quad 0 \in \text{Lower}(|f|)
\]

\[
x + \frac{1}{2} \in \text{Upper}(\text{round}(x)), \quad x - \frac{1}{2} \in \text{Lower}(\text{round}(x))
\]

Here are two simple examples of inequality proofs:

\[
0 < r \wedge a < b \implies a < \frac{a + br}{1 + r}
\]

replace expression with its lower or upper bounds

\[0 < r \wedge a < b \implies a \leq a\]

reduces to

\[True\]

Given that \( 0 < b < 1 \),

\[
\exists \beta\{-f_0 + b^n \text{Abs}(\cos(xa^n z)) \leq 0\}
\]

\[\wedge \text{IsConstant}(f_0, z) \wedge \text{Convergent}(\sum_{a=0}^{\infty} f_0)\]

and split

case 1.1:

\[
\exists \beta\{-f_0 + b^n \text{Abs}(\cos(xa^n z)) \leq 0\}
\]

replace expression with its lower or upper bounds

\[
\exists \beta\{-f_0 + b^n \leq 0\}
\]

inequality

\[
\{f_0 \to b^n\}
\]

case 1.2:

\[
\text{Convergent}(\sum_{a=0}^{\infty} b^n)
\]

simplify summations

\[True\]

**Conclusion**

In a related project that we plan to describe in a forthcoming paper, we have managed to prove completely automatically all of the theorems and examples in Chapter 2 of Ramanujan’s Collected Works [Berndt 1985]. The techniques we use are similar to those described in this paper. We believe that the examples we have been able to prove provide convincing justification for combining powerful symbolic computation techniques with theorem provers.

Nevertheless, there are many ways to improve Analytica. (Indeed, there are so many different extensions to try that it is hard to decide which to try first.) One direction is to add powerful algorithmic techniques for simplifying particular classes of formulas (like extensions of Gosper’s algorithm for summations). The difficulty with adding such techniques is that a proof obtained in this manner may be virtually impossible for a human to follow.

Another direction is to strengthen the ability of Analytica to do inductive proofs. The technique that Analytica currently uses for generating induction schemes is quite simple. More research is needed on the generation of complex induction schemes and the identification of sufficiently general hypotheses for inductive proofs. There has been a fair amount of research on this problem in the context of structural induction proofs of program correctness [Boyer and Moore 1979], but relatively little of this research seems applicable to inductive proofs in analysis. Perhaps the research that is most relevant is the ripples technique developed by Bundy [Bundy et al 1988].

Most proofs in modern analysis are based on set theory and many use topological concepts. Clearly, the extension of Analytica to handle such proofs is critical. Although theorem proving in set theory has been an important problem for a long time, there is no generally accepted technique for constructing such proofs. The most successful work on set theory so far is probably that of Quaife [1989]. His work, however, uses a theorem prover based on hyper-resolution and may not produce proofs that are very readable.

Better methods for managing hypotheses and previously proved lemmas and theorems are also needed. Techniques developed for proof-checking systems like LCF [Gordon et al 1979] and HOL [Gordon 1985] may be adequate in the short run, but some type of higher-order unification or matching will probably be necessary in the majority of cases. In general, deciding when to use an hypothesis or previous result is a very difficult problem. Every student of elementary calculus learns the mean value theorem by heart, but giving a good set of rules for determining when to apply this theorem in order to obtain a simpler bound on some complicated expression is not easy.

Certainly, some type of higher-order logic would be more appropriate for analysis than the first order logic we currently use. The ability to state higher-order lemmas would be an additional advantage of basing the prover on a higher-order logic and might help solve the problem described in the next paragraph. We intend to experiment with combining ideas from this paper with Andrews’ theorem prover for higher-order logic [Andrews 1989].

Perhaps, the most serious problem in building a theorem prover like Analytica is the soundness of the underlying symbolic computation system. Mathematica has some rules that lead to incorrect results in some cases. For example, Mathematica will always simplify \( 0^0 \) to 0, which may lead to a problem in certain circumstances:

\[
\text{ln[1]}\rightarrow \text{f[x_]}::=\text{Sum}[a\{k\} \cdot x^k, \{k, -1, n\}]
\]

\[
\text{ln[2]}\rightarrow \text{f[x]}
\]

\[
\text{Out[2]}\rightarrow \text{Sum}[x^k \cdot a\{k\}, \{k, -1, n\}]
\]

\[
\text{ln[3]}\rightarrow \text{f[0]}
\]

\[
\text{Out[3]}\rightarrow \text{Sum}[0, \{k, -1, n\}]
\]

We believe the solution to the soundness problem is to develop the theorem prover and the symbolic computation system together so that each simplification step can be rigorously justified.
References


Appendix

Here are two more complicated examples proved by Analytica.

**Stereographic Projection**

Consider the function that maps each point $(x_1, x_2, x_3)$ in 3-space to the complex plane $C$:

$$sp(x_1, x_2, x_3) = \frac{x_1 + i x_2}{1 - x_3}$$

We will use Analytica to prove that this mapping is a bijection between the unit sphere $S$ (given by $x_1^2 + x_2^2 + x_3^2 = 1$) and the complex plane. We will also prove that it is a projection: If $sp(x_1, x_2, x_3) = a + bi$, then the north pole $(0, 0, 1)$, $(x_1, x_2, x_3)$ and $(a, b, 0)$ are collinear [Ahlfors 1966].

![Stereographic Projection Diagram](image)

Outline of the proof:

1. $sp$ is a one-to-one mapping from $S$ to $C$.

   $$x_1^2 + x_2^2 + x_3^2 = 1 \land y_1^2 + y_2^2 + y_3^2 = 1$$

   $$\land sp(x_1, x_2, x_3) = sp(y_1, y_2, y_3)$$

   $$\Rightarrow x_3 = y_3 \land x_1 = y_1 \land x_2 = y_2$$

   Since $sp(x_1, x_2, x_3) = sp(y_1, y_2, y_3)$,

   $$\frac{x_1}{1 - x_3} = \frac{y_1}{1 - y_3} \land \frac{x_2}{1 - x_3} = \frac{y_2}{1 - y_3}$$

   Hence,

   $$\frac{x_1^2}{(1 - x_3)^2} = \frac{y_1^2}{(1 - y_3)^2}$$

   Using the fact that $(x_1, x_2, x_3)$ and $(y_1, y_2, y_3)$ are on $S$, we get

   $$\frac{1 - x_3^2}{(1 - x_3)^2} = \frac{1 - y_3^2}{(1 - y_3)^2}$$

   which leads to $x_3 = y_3$. Similarly, $x_1 = y_1$ and $x_2 = y_2$. 


2. sp is an onto mapping from S to C.

\[ \exists \{x_1, x_2, x_3\} [sp(x_1, x_2, x_3) = z_1 + z_2 i \land z_1^2 + z_2^2 + z_3^2 = 0] \]

It is sufficient to have

\[ z_1 = (1 - x_3) z_2, \quad z_2 = (1 - x_3) z_2, \quad z_3 = \frac{z_1^2 + z_2^2 - 1}{z_1^2 + z_2^2 + 1} \]

3. If \( sp(x_1, x_2, x_3) = z_1 + z_2 i \), then \((0,0,1), (x_1, x_2, x_3), (z_1, z_2, 0)\) are on a straight line.

\[
\begin{array}{c|c|c}
0 & 0 & 1 \\
\hline
x_1 & x_2 & 1 \\
\hline
z_1 & z_2 & 1 \\
\end{array}
\]

This follows directly from the definition.

Input for the problem:

(* the stereographic projection *)

\( sp(x_1, x_2, x_3) := \overline{sp(x_1 + i x_2, (1 - x_3));} \)

(* the predicate that decides if a point is on the unit sphere *)

\( \text{unit}(x_1, x_2, x_3) := (x_1^2 + x_2^2 + x_3^2 = 1); \)

(* stereographic projection is a one-to-one mapping *)

\( \text{Prove}(\text{imp}[\text{and}(\text{unit}(x_1, x_2, x_3), \text{unit}(y_1, y_2, y_3)]

\quad \text{sp}(x_1, x_2, x_3) = \text{sp}(y_1, y_2, y_3),

\quad \text{and}(x_3 = y_3, x_1 = y_1, x_2 = y_2)]; \)

(* stereographic projection is an onto mapping *)

\( \text{Prove}(\text{some}[[x_1, x_2, x_3],

\quad \text{and}(\text{sp}(x_1, x_2, x_3) = z_1 + z_2 i, \text{unit}(x_1, x_2, x_3)]; \)

(* The north pole \((0, 0, 1)\), the original point, and the projection are on a straight line *)

\( \text{collinear}([[x_1, x_2, x_3],

\quad \{y_1, y_2, y_3\}, [x_1, x_2, x_3]);

\quad \text{and}(\text{Det}([[x_1, x_2, 1], [y_1, y_2, 1], [x_1, x_2, 1]])) = 0,

\quad \text{Det}([[x_1, x_2, 1], [y_1, y_2, 1], [x_1, x_2, 1]])) = 0,

\quad \text{Det}([[x_1, x_2, 1], [y_1, y_2, 1], [x_2, x_3, 1]])) = 0]; \)

\( \text{Prove}(\text{imp}[z = \text{sp}(x_1, x_2, x_3),

\quad \text{collinear}([[\text{part}(z), \text{part}(z), 0],

\quad [x_1, x_2, x_3], (0,0,1)]]); \)

Theorems proved:

\( (\text{unit}(x_1, x_2, x_3) \land \text{unit}(y_1, y_2, y_3) \land \text{sp}(x_1, x_2, x_3) = \text{sp}(y_1, y_2, y_3),

\quad \text{implies} \quad x_3 = y_3 \land x_1 = y_1 \land x_2 = y_2 ); \)

\( \exists \{x_1, x_2, x_3\} [sp(x_1, x_2, x_3) = z_1 + z_2 i \land \text{unit}(x_1, x_2, x_3)]; \)

\( z = \text{sp}(x_1, x_2, x_3) \Rightarrow \text{collinear}([[R(z), I(z), 0], [x_1, x_2, x_3], (0,0,1)]); \)

Theorem:

\( (\text{unit}(x_1, x_2, x_3) \land \text{unit}(y_1, y_2, y_3) \land \text{sp}(x_1, x_2, x_3) = \text{sp}(y_1, y_2, y_3),

\quad \text{implies} \quad x_3 = y_3 \land x_1 = y_1 \land x_2 = y_2 ); \)

Proof:

\( x_1^2 + x_2^2 + x_3^2 = 1 \land y_1^2 + y_2^2 + y_3^2 = 1 \land \frac{x_1}{1 - x_3} = \frac{y_1}{1 - y_3}; \)

\( \frac{x_2}{1 - x_3} = \frac{y_2}{1 - y_3} \Rightarrow x_3 = y_3 \land x_1 = y_1 \land x_2 = y_2 \)

reduces to

\( x_1^2 + x_2^2 + x_3^2 = 1 \land y_1^2 + y_2^2 + y_3^2 = 1 \land \frac{x_1}{1 - x_3} = \frac{y_1}{1 - y_3}; \)

\( \frac{x_2}{1 - x_3} = \frac{y_2}{1 - y_3} \Rightarrow x_3 = y_3 \land x_1 = y_1 \land x_2 = y_2 \)

and split

- case 1.1:

\( x_1^2 + x_2^2 + x_3^2 = 1 \land y_1^2 + y_2^2 + y_3^2 = 1 \land \frac{x_1}{1 - x_3} = \frac{y_1}{1 - y_3}; \)

\( \frac{x_2}{1 - x_3} = \frac{y_2}{1 - y_3} \Rightarrow x_3 = y_3 \)

rewrite as

\( -1 + x_1^2 + x_2^2 + x_3^2 = 0 \land -1 + y_1^2 + y_2^2 + y_3^2 = 0 \)

\( \land \frac{x_1 + y_1 - x_3 y_1 + x_3 y_1}{(-1 + x_3)(-1 + y_3)} = 0 \land \frac{-x_2 + y_2 - x_3 y_2 + x_3 y_2}{(-1 + x_3)(-1 + y_3)} = 0 \)

\( \Rightarrow x_3 - y_3 = 0 \)

solves linear equation

\( x_1^2 = -(-1 + x_2^2 + x_3^2) \land y_1^2 = -(-1 + y_2^2 + y_3^2) \)

\( \land x_1 = \frac{-1 + x_1}{y_1} \land x_2 = \frac{-1 + x_2}{y_2} \Rightarrow x_1 - y_3 = 0 \)

substitute using equation

\( \frac{(-1 + x_1^2)(-1 + y_2^2 + y_3^2)}{(-1 + y_3^2)} = \frac{(-1 + x_2^2)(-1 + y_2^2 + y_3^2)}{(-1 + y_3^2)} \)

\( \Rightarrow x_3 - y_3 = 0 \)

reduces to

\( \frac{(-1 + x_1^2)(-1 + y_2^2 + y_3^2)}{(-1 + y_3^2)} = \frac{(-1 + x_2^2)(-1 + y_2^2 + y_3^2)}{(-1 + y_3^2)} \)

rewritten as

\( \frac{2(-1 + x_2)(x_3 - y_3)}{-1 + y_3} = 0 \Rightarrow x_3 - y_3 = 0 \)

reduces to

\( \text{True} \)

- case 1.2:

\( x_1^2 + x_2^2 + x_3^2 = 1 \land y_1^2 + y_2^2 + y_3^2 = 1 \land \frac{x_1}{1 - x_3} = \frac{y_1}{1 - y_3}; \)

\( \frac{x_2}{1 - x_3} = \frac{y_2}{1 - y_3} \land x_3 = y_3 \Rightarrow x_1 = y_1 \land x_2 = y_2 \)

and split

- case 1.2.1:

\( x_1^2 + x_2^2 + x_3^2 = 1 \land y_1^2 + y_2^2 + y_3^2 = 1 \land \frac{x_1}{1 - x_3} = \frac{y_1}{1 - y_3}; \)

\( \frac{x_2}{1 - x_3} = \frac{y_2}{1 - y_3} \Rightarrow x_1 = y_1 \)

substitute using equation

\( x_1^2 + x_2^2 + y_3^2 = 1 \land y_1^2 + y_2^2 + y_3^2 = 1 \land \frac{x_1}{1 - x_3} = \frac{y_1}{1 - y_3}; \)

\( \frac{x_2}{1 - x_3} = \frac{y_2}{1 - y_3} \Rightarrow x_1 = y_1 \)

reduces to

\( \text{True} \)
case 1.2.2:

\[ x_1^2 + x_2^2 + x_3^2 = 1 \land y_1^2 + y_2^2 + y_3^2 = 1 \land \frac{x_1}{1 + x_3} = \frac{y_1}{1 + y_3} \]

\[ \land \frac{x_2}{1 + x_3} = \frac{y_2}{1 + y_3} \land x_1 = y_1 \land x_2 = y_2 \Rightarrow x_2 = y_2 \]

substitute using equation

\[ x_2^2 + y_2^2 + z_2^2 = 1 \land y_2^2 + y_3^2 + z_2^2 = 1 \land \frac{x_2}{1 + y_3} = \frac{y_2}{1 + y_3} \]

\[ \Rightarrow x_2 = y_2 \]

reduces to

\[ True \]

Theorem:

\[ \exists (x_1, x_2, x_3) [sp(x_1, x_2, x_3) = z_1 + z_2i \land \text{unit}(x_1, x_2, x_3)] \]

Proof:

\[ \exists x_1 \exists x_2 \exists x_3 [\frac{x_1}{1 - x_1} = z_1 \land \frac{x_2}{1 - x_2} = z_2 \land x_1^2 + x_2^2 + x_3^2 = 1] \]

check denominator

\[ \exists x_1 \exists x_2 \exists x_3 [1 - x_3 \neq 0 \land \frac{x_1}{1 - x_3} = z_1 \land \frac{x_2}{1 - x_3} = z_2 \land x_1^2 + x_2^2 + x_3^2 = 1] \]

and split case 1.1:

\[ \forall x_3 [1 - x_3 = 0] \Rightarrow False \]

rewrite as

\[ \forall x_3 [(-1 + x_3) = 0] \Rightarrow False \]

reduces to

\[ \forall x_3 [-1 + x_3 = 0] \Rightarrow False \]

solve linear equation

\[ \forall x_3 [x_3 = 1] \Rightarrow False \]

add restriction:

\[ x_3 \neq 1 \]

case 1.2:

\[ \exists x_1 \exists x_2 \exists x_3 [1 - x_3 = 0 \lor \frac{x_1}{1 - x_3} = z_1 \land \frac{x_2}{1 - x_3} = z_2 \land x_1^2 + x_2^2 + x_3^2 = 1] \]

and split case 1.2.1:

\[ \exists x_1 \exists x_2 \exists x_3 [\frac{-x_1}{1 - x_1} = z_1 \land \frac{-x_2}{1 - x_2} = z_2 \land x_1^2 + x_2^2 + x_3^2 = 1] \]

rewrite as

\[ \exists x_1 \exists x_2 \exists x_3 [\frac{-x_1 - x_2 + x_1 x_2}{1 - x_3} = 0] \]

reduces to

\[ \exists x_1 \exists x_2 [x_1 + (-1 + x_2) x_1 = 0] \]

equation

\[ \{ x_1 \rightarrow ((-1 + x_2) x_1) \}

and split case 1.2.2:

\[ \exists x_2 \exists x_3 [\frac{-x_2}{1 - x_3} = z_2 \land x_2^2 + x_3^2 + (-1 + x_3)^2 = 1] \]

rewrite as

\[ \exists x_2 \exists x_3 [\frac{-x_2 x_3 + x_2}{1 - x_3} = 0] \]

reduces to

\[ \exists x_2 \exists x_3 [x_2 + (-1 + x_3) x_2 = 0] \]

equation

\[ \{ x_2 \rightarrow ((-1 + x_3) x_2) \}

Theorem:

\[ z = sp(x_1, x_2, x_3) \Rightarrow \text{rollinear} \{ R(z), I(z), 0 \}, \{ x_1, x_2, x_3 \}, \{ 0, 0, 1 \} \]

Proof:

\[ z = \frac{x_1}{1 - x_1} + \frac{x_2}{1 - x_3} \Rightarrow (x_1 I(z)) + x_2 R(z) = 0 \land \]

\[ x_1 - R(z) + x_2 R(z) = 0 \land x_2 - I(z) + x_2 I(z) = 0 \]

reduces to

\[ z = \frac{x_1}{1 - x_1} + \frac{x_2}{1 - x_3} \Rightarrow (x_1 I(z)) + x_2 R(z) = 0 \land \]

\[ x_1 + (-1 + x_3) R(z) = 0 \land x_2 + (-1 + x_3) I(z) = 0 \]

and split case 1.1:

\[ z = \frac{-x_1}{-1 + x_3} - \frac{x_2}{1 - x_3} \Rightarrow (x_1 I(z)) + x_2 R(z) = 0 \land \]

\[ x_1 + (-1 + x_3) R(z) = 0 \land x_2 + (-1 + x_3) I(z) = 0 \]

substitute using equation

\[ True \]

case 1.2:

\[ z = \frac{-x_1}{-1 + x_3} - \frac{x_2}{1 - x_3} \land (x_1 I(z)) + x_2 R(z) = 0 \land \]

\[ x_1 + (-1 + x_3) R(z) = 0 \land x_2 + (-1 + x_3) I(z) = 0 \]
and split case 1.2.1:

\[ z = \frac{x_1}{-1 + x_3} - \frac{x_2 i}{-1 + x_3} - (x_1 I(z)) + x_3 R(z) = 0 \]

\[ \implies x_1 + (-1 + x_3) I(z) = 0 \]

substitute using equation \( \text{True} \)

\[ z = -\frac{x_1}{-1 + x_3} - \frac{x_2 i}{-1 + x_3} - (x_1 I(z)) + x_3 R(z) = 0 \]

\( \land x_1 + (-1 + x_3) I(z) = 0 \implies x_2 + (-1 + x_3) I(z) = 0 \)

substitute using equation \( \text{True} \)

\[ z = -\frac{x_1}{-1 + x_3} - \frac{x_2 i}{-1 + x_3} - (x_1 I(z)) + x_3 R(z) = 0 \]

A nowhere-Differentiable Function
Weierstrass's non-differentiable function is defined by the series

\[ f(x) = \sum_{n=0}^{\infty} b^n \cos(a^n \pi x) \]

where \( a \) is a odd positive integer and \( 0 < b < 1 \). When \( ab > 1 \) and \( \frac{a}{b} < 1 \), the derivative of the function does not exist for any value of \( x \). Titchmarsh [1932] described the function as having "infinitely many infinitesimal crinkles." Here are plots of

\[ f(x) = \sum_{n=0}^{10} \cos(15^n \pi x)(0.5)^n \]

over the intervals \([0, 10], [0, 1], [0, 0.1], \) and \([0, 0.01] \).

Outline of the proof:
If \( 0 < b < 1 \), then \( \sum_{n=0}^{\infty} b^n \) is convergent. Hence \( \sum_{n=0}^{\infty} b^n \cos(a^n \pi x) \) is uniformly convergent in any interval, which means \( f(x) \) is everywhere continuous.

The derivative of \( f \) at \( x \) is the limit as \( h \) approaches 0 of

\[ \frac{f(x + h) - f(x)}{h} = \sum_{n=0}^{\infty} b^n \cos(a^n \pi(x + h)) - \cos(a^n \pi x) \]

\[ = \sum_{n=0}^{\infty} b^n \cos(a^n \pi(x + h)) - \cos(a^n \pi x) \]

Because \( \forall x, y. |\cos(x) - \cos(y)| \leq |x - y| \),

\[ |S_m| \leq \sum_{n=0}^{\infty} b^n \cos(a^n \pi(x + h)) - \cos(a^n \pi x) \]

\[ = \sum_{n=0}^{\infty} b^n \cos(a^n \pi(x + h)) - \cos(a^n \pi x) \]

\[ = \sum_{n=0}^{\infty} b^n \cos(a^n \pi(x + h)) - \cos(a^n \pi x) \]

\[ \leq \sum_{n=0}^{\infty} \frac{a^n}{b^n} \frac{1}{n+1} \leq \frac{a^n}{b^n} \frac{1}{n+1} \]

Take \( \alpha_m \) to be the nearest integer to \( a^m \pi \), i.e., \(-1/2 < a^m \pi - \alpha_m < 1/2 \). Let \( \xi_m = a^m \pi - \alpha_m - h \). Let \( h = (1 - \xi_m)/a^m \). Then \( 0 < h < 3/2a^m \). When \( n \geq m \),

\[ \cos(a^n \pi(x + h)) = \cos(a^n \pi(x + h) + 1) = (-1)^{a^m \pi + 1} \]

Also

\[ \cos(a^n \pi x) = \cos(a^n \pi(x + \xi_m)) = (-1)^{a^m \pi + 1} \cos(a^n \pi \xi_m) \]

Hence

\[ R_m = \frac{(-1)^{a^m \pi + 1}}{h} \sum_{n=m}^{\infty} b^n \left(1 + \cos(a^n \pi \xi_m)\right) \]

\[ |R_m| = \frac{1}{h} \sum_{n=m}^{\infty} b^n \left(1 + \cos(a^n \pi \xi_m)\right) \geq \frac{b^n(1 + \cos(a^n \pi \xi_m))}{h} \geq \frac{2}{3} \frac{1}{a^m} \frac{b^n}{h} \]

So

\[ \left| \frac{f(x + h) - f(x)}{h} \right| \leq |R_m| - |S_m| \sim \frac{2}{3} \frac{1}{a^m} \frac{b^n}{h} \]

If \( ab > 1 + \frac{2}{3}\pi \), when \( m \to \infty \), \( h \) tends to 0 and the above formula tends to infinity. So \( f(x) \) cannot be finite.

Input of the problem:

(* the definition of the nondifferentiable function *)

\[ f[x_] := \text{sum}[b^n \text{Cos}[a^n \text{Pi} x], (n, 0, \text{infinity})]; \]

(* a, n, m are all integers, while a is odd *)

integer[a] := True;
integer[n] := True;
integer[m] := True;
Odd[a] := True;

(* some given properties *)

Given[a > 1];
Given[b<1];
Given[a>0];
Given[a<1];
Given[b>0];
Given[b>0];
Given[Pi>0];
Given[a > 1 + 3/2 Pi];

(* several auxiliary functions *)

AddDefinition[diff[Pi, 0, 1]] := \text{sum}[b^n \text{Cos}[a^n \text{Pi} \xi[m]] - \text{Cos}[a^n \text{Pi} \xi[m]] / h, (n, 0, m-1)];

AddDefinition[h[n]] := (-1)^{\text{round}(a^n \pi)} \text{Cos}[a^n \text{Pi} x] + 1 \text{sum}[b^n (1 + \text{Cos}[a^n \text{Pi} \xi[m]]) / h, (n, a, \text{infinity})];

AddDefinition[diff[Pi, 0, 1]] := \text{diff}[Pi, 0, 1]/h + 1; (* a given value for h *)

h = (1 - xi[n]) / a^m;
Given properties:

\[ 0 < \pi \quad 0 < b \quad 0 < m \quad 0 < n \quad 1 - a < 0 \]
\[ -1 + t < 0 \quad 1 - ab < 0 \quad 1 + \frac{3\pi}{2} - ab < 0 \]

Definitions used in proof:

\[ S(m) = \frac{a^m}{1 - \xi(m)} \left( \sum_{n=-m}^{m} b^n \left( -\cos(\pi a^n x) + \cos(\pi a^{-n-m}(1 + \alpha)) \right) \right) \]
\[ R(m) = \frac{(-1)^m}{1 - \xi(m)} \left( \sum_{n=-m}^{m} b^n (1 + \cos(\pi a^{-n-m}\xi(m))) \right) \]
\[ \text{diff}(h, f, x) = \frac{-f(x) + f(h + x)}{h} \]

Abbreviations used in proof:

\[ a = 1 - a^m x + \text{round}(a^m x) \]
\[ \xi(m) = a^m x - \text{round}(a^m x) \]
\[ f(x) = \sum_{n=-m}^{m} b^n \cos(\pi a^n x) \]
\[ \alpha = \text{round}(a^m x) \]

Theorems proved:

\[ \text{Continuous}(f(x), [x, x_0]) \]
\[ \text{diff}(h, f, x) = R(m) + S(m) \]
\[ \text{Abs}(R(m)) \geq 2a^m b^m \]
\[ \text{Abs}(S(m)) < \pi a^m b^m \]
\[ \forall \left( e > 0 \implies \lim_{m \to \infty} \text{Abs}(h) < e \right) \]
\[ \forall M \left( \lim_{m \to \infty} \text{Abs} \left( \text{diff}(h, f, x) \right) > M \right) \]

Theorem:

\[ \text{Continuous}(f(x), [x, x_0]) \]

Proof:

\[ \text{Continuous}(f(x), [x, x_0]) \]

matching lemma

\[ \forall \forall a \forall x \forall \delta \exists \text{min} \left( \text{Continuous}(f, [x, x_0]) \right) \]
\[ \land \text{UniformlyConvergent} \left( \sum_{n=-m}^{m} f, [x, x_0 - \alpha, x_0 + \alpha] \right) \]
\[ \implies \text{Continuous} \left( \sum_{n=-m}^{m} f, [x, x_0] \right) \]

with

\[ \{ f = b^n \cos(\pi a^n x), n < n, x < x, x_0 \to x_0, \min \to 0 \} \]

back chaining

\[ \text{UniformlyConvergent}(f(x), [x, x_0 - \alpha, x_0 + \alpha]) \]

matching lemma

\[ \forall \forall \forall a \forall x \forall \delta \exists \delta \left( f(x) < 0 \land f(x) < 0 \implies f(x) + \text{Abs}(f(x)) \leq 0 \right) \]
\[ \land \text{IsConstant}(f_0, x) \land \text{Convergent} \left( \sum_{n=-m}^{m} f_0 \right) \]
\[ \implies \text{UniformlyConvergent} \left( \sum_{n=-m}^{m} f(x, x_0, x_1, x_2) \right) \]

with

\[ \{ c_2 = \alpha + x_0, \min \to 0 \}
\[ f = b^n \cos(\pi a^n x), n < n, x < x, x_1 = -\alpha + x_1 \}

back chaining

\[ \exists \exists \exists f(x) < 0 \land f(x) < 0 \implies f(x) + \text{Abs}(f(x)) \leq 0 \]
\[ \land \text{IsConstant}(f_0, x) \land \text{Convergent} \left( \sum_{n=-m}^{m} f_0 \right) \]

reduces to

\[ \exists \exists \exists f(x) < 0 \land f(x) < 0 \implies f(x) + \text{Abs}(f(x)) \leq 0 \]
\[ \land \text{IsConstant}(f_0, x) \land \text{Convergent} \left( \sum_{n=-m}^{m} f_0 \right) \]

and split

case 1.1:

\[ \{ -\alpha - x + x_0 < 0 \land -\alpha - x - x_0 < 0 \}
\[ \implies \exists \exists f_0 + b^n \text{Abs}(\cos(\pi a^n x)) \leq 0 \]

replace expression with its lower or upper bounds

\[ -\alpha - x + x_0 < 0 \land -\alpha - x + x_0 < 0 \implies \exists \exists f_0 + b^n \text{Abs}(\cos(\pi a^n x)) \leq 0 \]

inequality

\[ \{ f_0 = b^n \}

case 1.2:

\[ \text{Convergent} \left( \sum_{n=0}^{m} b^n \right) \]

simplify summations

\[ \text{True} \]

Theorem:

\[ \text{diff}(h, f, x) = R(m) + S(m) \]

Proof:

\[ \text{diff}(h, f, x) = R(m) + S(m) \]

rewrite as

\[ -(R(m) + S(m)) - \text{diff}(h, f, x) = 0 \]

reduces to

\[ R(m) + S(m) - \text{diff}(h, f, x) = 0 \]

open definition

\[ \frac{(-1)^m b^n \left( \sum_{n=m}^{m} b^n \left( -\cos(\pi a^n x) + \cos(\pi a^{-n-m}\xi(m)) \right) \right) \left( \sum_{n=-m}^{m} \left( 1 + \cos(\pi a^{-n-m}(1 + \alpha)) \right) \right) \}}{1 - \xi(m)} \]
\[ \frac{a^m \left( \sum_{n=-m}^{m} b^n \left( -\cos(\pi a^n x) + \cos(\pi a^{-n-m}(1 + \alpha)) \right) \right) \left( \sum_{n=-m}^{m} \left( 1 + \cos(\pi a^{-n-m}(1 + \alpha)) \right) \right) \}}{1 - \xi(m)} \]
reduces to

\[ f(x) - (-1)^n \left( \sum_{n=0}^{\infty} b^n (1 + \cos(\pi a^{-m+n} \xi(m))) \right) \]

\[ - \left( \sum_{n=0}^{\infty} b^n \cos(\pi a^{-m+n} (1 + \alpha)) \right) \]

simplify summations

\[ - \left( (-1)^n \left( \sum_{n=0}^{\infty} b^n (1 + \cos(\pi a^{-m+n} \xi(m))) \right) \right) \]

\[ + \sum_{n=0}^{\infty} b^n (\cos(\pi a^n x) - \cos(\pi a^{-m+n} (1 + \alpha))) \]

\[ + \sum_{n=0}^{\infty} b^n \cos(\pi a^n x) - b^n \cos(\pi a^{-m+n} (1 + \alpha)) = 0 \]

reduces to

\[ - \left( (-1)^n \left( \sum_{n=0}^{\infty} b^n (1 + \cos(\pi a^{-m+n} \xi(m))) \right) \right) \]

\[ + \sum_{n=0}^{\infty} b^n (\cos(\pi a^n x) - \cos(\pi a^{-m+n} (1 + \alpha))) \]

\[ + \sum_{n=0}^{\infty} b^n \cos(\pi a^n x) - b^n \cos(\pi a^{-m+n} (1 + \alpha)) = 0 \]

simplify summations

\[ - \left( (-1)^n \left( \sum_{n=0}^{\infty} b^n (1 + \cos(\pi a^{-m+n} \xi(m))) \right) \right) \]

\[ - \left( \sum_{n=0}^{\infty} b^n \cos(\pi a^n x) - \cos(\pi a^{-m+n} (1 + \alpha)) \right) \]

0

reduces to

\[ - \left( (-1)^n \left( \sum_{n=0}^{\infty} b^n (1 + \cos(\pi a^{-m+n} \xi(m))) \right) \right) \]

\[ + \sum_{n=0}^{\infty} b^n (\cos(\pi a^n x) - \cos(\pi a^{-m+n} (1 + \alpha))) \]

\[ + \sum_{n=0}^{\infty} b^n \cos(\pi a^n x) - b^n \cos(\pi a^{-m+n} (1 + \alpha)) = 0 \]

simplify summations

\[ - \left( (-1)^n \left( \sum_{n=0}^{\infty} b^n (1 + \cos(\pi a^{-m+n} \xi(m))) \right) \right) \]

\[ - \left( \sum_{n=0}^{\infty} b^n \cos(\pi a^n x) - \cos(\pi a^{-m+n} (1 + \alpha)) \right) \]

0

reduces to

\[ - \left( (-1)^n \left( \sum_{n=0}^{\infty} b^n (1 + \cos(\pi a^{-m+n} \xi(m))) \right) \right) \]

\[ + \sum_{n=0}^{\infty} b^n (\cos(\pi a^n x) - \cos(\pi a^{-m+n} (1 + \alpha))) \]

\[ - \sum_{n=0}^{\infty} b^n \cos(\pi a^n x) + \cos(\pi a^{-m+n} (1 + \alpha)) = 0 \]

simplify summations

\[ \sum_{n=0}^{\infty} \left( (-1)^n b^n (1 + \cos(\pi a^{-m+n} \xi(m))) \right) \]

\[ - b^n (\cos(\pi a^n x) - \cos(\pi a^{-m+n} (1 + \alpha))) = 0 \]

reduces to

\[ \sum_{n=0}^{\infty} \left( b^n (\cos(\pi a^n x) + (-1)^n (1 + \cos(\pi a^{-m+n} \xi(m))) \right) \]

\[ - \sum_{n=0}^{\infty} b^n (\cos(\pi a^n x) + (-1)^n (1 + \cos(\pi a^{-m+n} \xi(m))) \]

\[ + \cos(\pi a^{-m+n} (1 + \alpha)) = 0 \]

matching lemma

∀x \left[ -\frac{\pi}{2} - x \leq 0 \land -\frac{\pi}{2} + x \leq 0 \Rightarrow 0 \leq \cos(x) \right]

with

\{ k - n.\text{low} - m.\text{up} \to -\infty \Rightarrow b^n (\cos(\pi a^n x) + (-1)^n (1 + \cos(\pi a^{-m+n} \xi(m))) + \cos(\pi a^{-m+n} (1 + \alpha))) \}

hack chaining

\[ m - n \leq 0 \land -\infty \leq 0 \Rightarrow b^n (\cos(\pi a^n x) + (-1)^n (1 + \cos(\pi a^{-m+n} \xi(m))) + \cos(\pi a^{-m+n} (1 + \alpha))) = 0 \]

reduces to

\[ m - n \leq 0 \Rightarrow -\cos(\pi a^n x) + (-1)^n (1 + \cos(\pi a^{-m+n} \xi(m))) + \cos(\pi a^{-m+n} (1 + \alpha)) = 0 \]

rewrite trigonometric expressions

\[ \text{True} \]

Theorem:

\[ \text{Abs}(R(m)) \geq \frac{2a^m b^n}{3} \]

Proof:

\[ \text{Abs}(R(m)) \geq \frac{2a^m b^n}{3} \]

reduces to

\[ \frac{2a^m b^n}{3} - \text{Abs}(R(m)) \leq 0 \]

rewrite as

\[ \frac{2a^m b^n}{3} - 3\text{Abs}(R(m)) \leq 0 \]

reduces to

\[ 2a^m b^n - 3\text{Abs}(R(m)) \leq 0 \]

open definition

\[ 2a^m b^n - \frac{3\text{Abs}(R(m))}{\text{Abs}(1 - \xi(m))} \leq 0 \]

reduces to

\[ 2b^m - \frac{3(\sum_{n=0}^{\infty} b^n (1 + \cos(\pi a^{-m+n} \xi(m))))}{1 - \xi(m)} \leq 0 \]

replace expression with its lower or upper bounds

\[ 2b^m - \frac{3b^m (1 + \cos(\pi \xi(m)))}{1 - \xi(m)} \leq 0 \]

reduces to

\[ 2 - \frac{3(1 + \cos(\pi \xi(m)))}{1 - \xi(m)} \leq 0 \]

replace expression with its lower or upper bounds

\[ -2\cos(\pi \xi(m)) \leq 0 \]

reduces to

\[ 0 \leq \cos(\pi \xi(m)) \]

matching lemma

∀x \left[ -\frac{\pi}{2} \leq 0 \land -\frac{\pi}{2} + x \leq 0 \Rightarrow 0 \leq \cos(x) \right]

with

\{ x - \pi \xi(m) \}

hack chaining

\[ -\frac{\pi}{2} - \pi \xi(m) \leq 0 \land -\frac{\pi}{2} + \pi \xi(m) \leq 0 \]
Theorem:
\[ \text{Abs}(S(m)) < \frac{\pi a^m b^m}{1 + ab} \]

Proof:
\[ \text{Abs}(S(m)) < \frac{\pi a^m b^m}{1 + ab} \]

reduces to
\[ -\frac{\pi a^m b^m}{1 + ab} < 0 \]

rewrite as
\[ -\pi a^m b^m + \text{Abs}(S(m)) - ab\text{Abs}(S(m)) < 0 \]

reduces to
\[ -\pi a^m b^m + (-1 + ab)\text{Abs}(S(m)) < 0 \]

open definition
\[ -\pi a^m b^m + \left[ \frac{(-1 + ab)\text{Abs}(a)^m}{\text{Abs}(1 - \xi(m))} \right] \]
\[ \cdot \text{Abs}\left( \sum_{n=0}^{1+m} b^n \left( -\cos(\pi a^n x) + \cos(\pi a^{-n+m} (1 + n)) \right) \right) < 0 \]

reduces to
\[ -\pi b^m + \left[ \frac{(-1 + ab)}{(1 - \xi(m))} \right] \]
\[ \cdot \text{Abs}\left( \sum_{n=0}^{1+m} b^n \left( -\cos(\pi a^n x) + \cos(\pi a^{-n+m} (1 + n)) \right) \right) < 0 \]

replace expression with its lower or upper bounds
\[ \frac{(-1+ab)}{1-\xi(m)} \text{Abs}\left( \sum_{n=0}^{1+m} b^n \left( -\cos(\pi a^n x) + \cos(\pi a^{-n+m} (1 + n)) \right) \right) \leq 0 \]

reduces to
\[ -\pi b^m + \left[ \frac{(-1 + ab)}{(1 - \xi(m))} \right] \]
\[ \cdot \text{Abs}\left( \sum_{n=0}^{1+m} b^n \text{Abs}\left( -\cos(\pi a^n x) + \cos(\pi a^{-n+m} (1 + n)) \right) \right) < 0 \]

reduces to
\[ -\pi b^m + \left[ \frac{(-1 + ab)}{1 - \xi(m)} \right] \]
\[ \cdot \text{Abs}\left( \sum_{n=0}^{1+m} b^n \text{Abs}\left( \cos(\pi a^n x) - \cos(\pi a^{-n+m} (1 + n)) \right) \right) < 0 \]

replace expression with its lower or upper bounds
\[ -\pi b^m + \left[ \frac{(-1 + ab)}{1 - \xi(m)} \right] \]
\[ \cdot \text{Abs}\left( \sum_{n=0}^{1+m} b^n \text{Abs}\left( \cos(\pi a^n x) - \cos(\pi a^{-n+m} (1 + n)) \right) \right) < 0 \]

reduce limits
\[ \infty < 0 \]

reduces to
\[ \text{True} \]