Analytica — A Theorem Prover for Mathematica

Edmund Clarke Xudong Zhao

September 1992
CMU-CS-92-117

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was sponsored in part by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597 and in part by National Science Foundation under Contract Number CCR-8722633.

Keywords: Theorem Prover, Symbolic computation, Mathemmatica, Analytica

Abstract

Analytica is an automatic theorem prover for theorems in elementary analysis. The prover is written in Mathematica language and runs in the Mathematica environment. 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. The theorem prover 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.

In this paper we describe the structure of Analytica and explain the main techniques that it uses to construct proofs. We have tried to make the paper as self-contained as possible so that it will be accessible to a wide audience of potential users. We illustrate the power of our theorem prover by several non-trivial examples including the basic properties of the stereographic projection and a series of three lemmas that lead to a proof of Weierstrass's example of a continuous nowhere differentiable function. Each of the lemmas in the latter example is proved completely automatically.

1. Introduction

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. 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 the interactive environment provided by this system [20]. 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 the one developed by Bledsoe [5] at the University of Texas. 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 [7] 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 [5, 6] is certainly the best known. Analytica has been heavily influenced by his research. More recently, Farmer, Guttman, and Thayer at Mitre Corporation [10] 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 [18] 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 [15] have also experimented with the use of Reduce for program verification.

Our paper is organized as follows: In Section 2, 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. In Section 3, we briefly describe the Mathematica programming language. We concentrate on the rewrite rule facility that is used extensively in our prover. Our intention is to provide enough of an introduction to the language so that a reader, who is unfamiliar with Mathematica, will still be able to understand most of the code that we use in examples. The reader who is already familiar with the language can skip this section and still understand the remainder of the paper. Section 4 contains an overview of the structure of Analytica and the major techniques that it uses in constructing proofs. Sections 5, 6, and 7 describe several of the most basic techniques in greater detail. The simplification phase is discussed in Section 5. Section 6 deals with summation and includes a short description of how we have integrated Gosper's algorithm into the prover. Section 7 discusses the *SUP-INF* method and how Analytica treats inequalities. The paper concludes in Section 8 with a discussion of some extensions that we hope to add to Analytica in the near future. Several more complicated examples, including the basic properties of the Stereographic Projection and Weierstrass's example of a continuous nowhere differentiable function, are given in the Appendix.

2. 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 :

$$\left(r > 0 \wedge a < b \Rightarrow a < \frac{a+br}{1+r} < b\right)$$

Proof :

$$r > 0 \wedge a < b \Longrightarrow a < \frac{a+br}{1+r} < b$$

reduces to

$$0 < r \wedge a < b \Longrightarrow a < \frac{a+br}{1+r} \wedge \frac{a+br}{1+r} < b$$

and split
case 1.1:

$$0 < r \wedge a < b \Longrightarrow a < \frac{a+br}{1+r}$$

replace expression with its lower or upper bounds

$$0 < r \wedge a < b \Longrightarrow a \leq a$$

reduces to

$$\mathit{True}$$

case 1.2:

$$0 < r \wedge a < b \Longrightarrow \frac{a+br}{1+r} < b$$

replace expression with its lower or upper bounds

$$0 < r \wedge a < b \Longrightarrow b \leq b$$

reduces to

$$\mathit{True}$$

□

```
Prove[imp[and[a!=0, x!=y, a x^2 + b x + c == 0, a y^2 + b y + c == 0], x + y == -b/a]]
```

Theorem :

$$\left(a \neq 0 \wedge x \neq y \wedge a x^2 + b x + c = 0 \wedge a y^2 + b y + c = 0 \Rightarrow x + y = -\frac{b}{a}\right)$$

Proof :

$$a \neq 0 \wedge x \neq y \wedge c + bx + ax^2 = 0 \wedge c + by + ay^2 = 0 \Longrightarrow x + y = -\frac{b}{a}$$

reduces to

$$c + bx + ax^2 = 0 \wedge c + by + ay^2 = 0 \Longrightarrow x = y \vee a = 0 \vee x + y = -\frac{b}{a}$$

rewrite as

$$c + bx + ax^2 = 0 \wedge c + by + ay^2 = 0 \Longrightarrow x - y = 0 \vee a = 0 \vee \frac{b + ax + ay}{a} = 0$$

reduces to

$$c + bx + ax^2 = 0 \wedge c + by + ay^2 = 0 \Longrightarrow x - y = 0 \vee a = 0 \vee b + a\,(x + y) = 0$$

solve linear equation

$$c = -\left(x\,(b + ax)\right) \wedge c = -\left(y\,(b + ay)\right) \Longrightarrow x - y = 0 \vee a = 0 \vee b + a\,(x + y) = 0$$

substitute using equation

$$-\left(x\,(b + ax)\right) = -\left(y\,(b + ay)\right) \Longrightarrow x - y = 0 \vee a = 0 \vee b + a\,(x + y) = 0$$

reduces to

$$x\,(b + ax) = y\,(b + ay) \Longrightarrow x - y = 0 \vee a = 0 \vee b + a\,(x + y) = 0$$

rewrite as

$$(x - y)\,(b + ax + ay) = 0 \Longrightarrow x - y = 0 \vee a = 0 \vee b + ax + ay = 0$$

reduces to

$$x - y = 0 \vee b + a\,(x + y) = 0 \Longrightarrow x - y = 0 \vee a = 0 \vee b + a\,(x + y) = 0$$

simplify formula using local context

$$\mathit{True}$$

$\square$

```
ProveByInduction[n, sum[2^k / (1 + x^(2^k)), {k, 0, n}] == 1/(x-1) + 2^(n+1)/(1-x^(2^(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

$$\textit{True}$$

induction step

$$\sum_{k=0}^{n} \frac{2^k}{1+x^{2^k}} = \frac{1}{-1+x} + \frac{2\cdot 2^n}{1-x^{2\cdot 2^n}} \Longrightarrow \sum_{k=0}^{1+n} \frac{2^k}{1+x^{2^k}} = \frac{1}{-1+x} + \frac{4\cdot 2^n}{1-x^{4\cdot 2^n}}$$

reduces to

$$\sum_{k=0}^{n} \frac{2^k}{1+x^{2^k}} = \frac{1}{-1+x} + \frac{2\cdot 2^n}{1-x^{2\cdot 2^n}} \Longrightarrow \frac{2\cdot 2^n}{1+x^{2\cdot 2^n}} + \left(\sum_{k=0}^{n} \frac{2^k}{1+x^{2^k}}\right) = \frac{1}{-1+x} + \frac{4\cdot 2^n}{1-x^{4\cdot 2^n}}$$

substitute using equation

$$\sum_{k=0}^{n} \frac{2^k}{1+x^{2^k}} = \frac{1}{-1+x} + \frac{2\cdot 2^n}{1-x^{2\cdot 2^n}} \Longrightarrow \frac{1}{-1+x} + \frac{2\cdot 2^n}{1-x^{2\cdot 2^n}} + \frac{2\cdot 2^n}{1+x^{2\cdot 2^n}} = \frac{1}{-1+x} + \frac{4\cdot 2^n}{1-x^{4\cdot 2^n}}$$

reduces to

$$\textit{True}$$

$\square$

```
(* 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;
ProveByInduction[{n, 0, 2}, F[n] == ((1+Sqrt[5])^n - (1-Sqrt[5])^n) / (2^n Sqrt[5])];
```

Theorem :

$$F(n)=\frac{\left(1+\sqrt{5}\right)^n-\left(1-\sqrt{5}\right)^n}{2^n\sqrt{5}}$$

Proof :
prove by induction on n
base case with n = 0

$$\mathit{True}$$

base case with n = 1

$$1=\frac{1+\sqrt{5}-\left(1-\sqrt{5}\right)}{2\sqrt{5}}$$

reduces to

$$\mathit{True}$$

induction step

$$F(n)=\frac{-\left(1-\sqrt{5}\right)^n+\left(1+\sqrt{5}\right)^n}{2^n\sqrt{5}}\wedge F(1+n)=\frac{-\left(1-\sqrt{5}\right)^{1+n}+\left(1+\sqrt{5}\right)^{1+n}}{2\cdot 2^n\sqrt{5}}\Longrightarrow$$
$$F(n)+F(1+n)=\frac{-\left(1-\sqrt{5}\right)^{2+n}+\left(1+\sqrt{5}\right)^{2+n}}{4\cdot 2^n\sqrt{5}}$$

substitute using equation

$$F(n)=\frac{-\left(1-\sqrt{5}\right)^n+\left(1+\sqrt{5}\right)^n}{2^n\sqrt{5}}\wedge F(1+n)=\frac{-\left(1-\sqrt{5}\right)^{1+n}+\left(1+\sqrt{5}\right)^{1+n}}{2\cdot 2^n\sqrt{5}}\Longrightarrow$$
$$\frac{-\left(1-\sqrt{5}\right)^n+\left(1+\sqrt{5}\right)^n}{2^n\sqrt{5}}+\frac{-\left(1-\sqrt{5}\right)^{1+n}+\left(1+\sqrt{5}\right)^{1+n}}{2\cdot 2^n\sqrt{5}}=\frac{-\left(1-\sqrt{5}\right)^{2+n}+\left(1+\sqrt{5}\right)^{2+n}}{4\cdot 2^n\sqrt{5}}$$

reduces to

$$F(n)=\frac{-\left(1-\sqrt{5}\right)^n+\left(1+\sqrt{5}\right)^n}{2^n\sqrt{5}}\wedge F(1+n)=\frac{-\left(1-\sqrt{5}\right)^{1+n}+\left(1+\sqrt{5}\right)^{1+n}}{2\cdot 2^n\sqrt{5}}\Longrightarrow$$
$$\frac{\left(1-\sqrt{5}\right)^n\left(-3+\sqrt{5}\right)+\left(1+\sqrt{5}\right)^n\left(3+\sqrt{5}\right)}{2}=\frac{-\left(1-\sqrt{5}\right)^{2+n}+\left(1+\sqrt{5}\right)^{2+n}}{4}$$

rewrite as

$$\mathit{True}$$

□

3. Mathematica

Mathematica provides a powerful rule-based programming language. Analytica is written entirely in this language. Mathematica rules have the form "`Pattern op Body`", where the operation part `op` can be one of "`=, :=, ->, :>`". Normally, a rule is applicable to a class of expressions. The `pattern` part of the rule specifies the class of expressions. This part is constructed from an expression by replacing various parts with *blank patterns*. There are four kinds of blank patterns, "`x_, x_., x__, x___`". The first matches an arbitrary expression; the second indicates that the expression is optional; the third matches a sequence containing one or more expressions; and the last matches an expression sequence that may be empty. Names can be used to distinguish blank patterns. If a blank pattern appears more than once with the same name in a composite pattern, then each instance should match the same term. For example, consider the four patterns `f[x_, x_]`, `f[x_., x_.]`, `f[x__, x__]`, `f[x___, x___]`. The expression `f[a, a]` matches all four patterns, `f[a, b, a, b]` matches `f[x__, x__]` and `f[x___, x___]`, `f[ ]` matches `f[x_., x_.]` and `f[x___, x___]`, and `f[a, b]` does not match any pattern.

The `body` of the rule specifies the expression to which the left hand side should be rewritten. In a conditional rewrite rule, the body has the form "`exp /; cond`". In this case, the rule is applied only when the condition `cond` is satisfied. When a rule is applied, the variables appearing in the body that are names of blank patterns in the pattern part are replaced by the expressions that the corresponding blank patterns match.

The operation part of the rule determines when the body of the rule is evaluated. If the operation part is "`=`" or "`->`", the rule is an *eager rule*, and the body is evaluated as soon as the rule is introduced. If the operation part is "`:=`" or "`:>`", the rule is a *lazy rule* and the body is evaluated only when the rule is finally applied.

Rules may also differ in the way that they are applied. A rule with the operation part "`=`" or "`:=`" is a *global rule*. It is applied as soon as some expression occurs that matches its pattern part. A rule with operation part "`->`" or "`:>`" is a *local rule* and is not applied automatically. Local rules are explicitly applied by using "`/.`" or "`//.`". The expression "`e /. R`" causes the set of rules `R` to be applied to the expression `e` once; while "`e //. R`" causes the set of rules `R` to be applied to the expression `e` repeatedly until the expression fails to change. [20]

4. An overview of Analytica

Analytica consists of three different phases: simplification, inference, and rewriting. When a new formula is submitted to Analytica for proof, it is first 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*, 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.

4.1. Inference phase

The inference part of Analytica is similar to the *UT Prover* developed in the University of Texas at Austin by Bledsoe [5]. In particular, we use Skolemization to deal with the quantifiers that occur in the formula to be proved. We define the *position* of a quantifier within a formula as *positive* if it is in the scope of even number of negations, and *negative* otherwise. *Skolemization* consists of the following procedure: Replace $(\exists x.\Psi(x))$ at positive positions or $(\forall x.\Psi(x))$ at negative positions by $(\Psi(f(y_1, y_2, ..., y_n)))$ where $x, y_1, y_2, ..., y_n$ are all the free variables in $\Psi(x)$ and f 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 $\neg X'$ is the Skolemized form of $\neg X$ [11]. When a Skolemized formula is put in prefix form, all quantifiers are universal. 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[formula, pattern]` returns the list of all instances of `pattern` that appear in `formula`.

```
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 /. x->Var[UniqueName], positive];

Skolemize[all[x_, a_], negative] :=
    Skolemize[a /. x->Const[UniqueName, Instances[a, Var[_]]], negative];

Skolemize[some[x_, a_], positive] :=
    Skolemize[a /. x->Const[UniqueName, Instances[a, Var[_]]], positive];

Skolemize[some[x_, a_], negative] :=
    Skolemize[a /. x->Var[UniqueName], negative];

Skolemize[a_, _] := a;
```

For example, the Skolemized form of the formula

$$(\exists x.\forall y.P(x,y)) \rightarrow (\exists u.\forall v.Q(u,v))$$

is given by

$$P(x, y_0(x)) \rightarrow Q(u_0(), v)$$

where x, v are Skolem variables, and u_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.

Each formula to be proved is a sequent of the form $H \Longrightarrow C$, which is represented in Mathematica by `seq[H, C]`. The hypothesis part H is a conjunction of quantifier-free first-order formulas, while the conclusion part C is a disjunction of such formulas. A number of propositional simplification rules are used to reduce the complexity of sequents. For example, negation symbols are pushed inward using De Morgan's laws so that they only apply to atomic formulas. Analytica uses the following rules to construct a proof of the reduced sequent:

- 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.

  ```
  Imply[seq[h_, or[c1___, a_  ==  b_,  c2___]]] :=
      unify[a, b] /; unifiable[a, b];
  ```

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

  ```
  Imply[seq[h_, or[c1___, a_  <=  b_,  c2___]]] :=
      unify[a, b] /; unifiable[a, b];
  ```

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

```
  Imply[seq[h_, c_]] :=
      match[h, c] /; matchable[h, c];
```

- And-split: $H \to (A \wedge B)$ is equivalent to $(H \to A) \wedge ((H \wedge A) \to B)$.

```
  Imply[seq[h_, or[c1___, and[a_, b__], c2___]]] :=
      SequentialTry[seq[h, or[c1, a, c2]], seq[and[h, a], or[c1, and[b], c2]]];
```

- Cases: $(A \vee B) \to C$ is equivalent to $(A \to C) \wedge (B \to (C \vee A))$.

```
  Imply[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 to see if some part of the conclusion of the sequent matches the conclusion of a lemma.

```
  Imply[seq[h0_, or[c0___, c_, c1___]]] :=
      Imply[seq[h0, or[c0, hypothesis[ApplicableLemma[c]], c1]]] /;
          ApplicableLemma[c] =!= NIL;
```

Where SequentialTry is:

```
SequentialTry[s1_, s2_] := Imply[apply[Imply[s1], s]];
```

Backtracking is often necessary in the inference phase when there are multiple subgoals, because a substitution that satisfies one subgoal may not satisfy the other subgoals. 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 σ, then σ is applied to the other subgoals on the stack and Analytica attempts to prove them. If the other subgoals are not satisfiable under σ, then Analytica returns to the previous goal and tries to find another substitution that makes it true.

The changes that are necessary for the first three inference rules are shown below. The other three inference rules do not change.

- equations:

```
  Imply[seq[h_, or[c1___, a_ == b_, c2___]]] :=
      True /; unifiable[a, b] && TryOtherBranches[unify[a, b]];
```

- inequalities:

  ```
  Imply[seq[h_, or[c1___, a_ <= b_, c2___]]] :=
      True /; unifiable[a, b] && TryOtherBranches[unify[a, b]];
  ```

- match:

  ```
  Imply[seq[h_, c_]] :=
      True /; matchable[h, c] && TryOtherBranches[match[h, c]];
  ```

SequentialTry becomes:

```
SetAttributes[SequentialTry, {HoldAll}];

SequentialTry[s1_, s2_] :=
    (Push[s2]; Imply[s1]);
```

And TryOtherBranches:

```
TryOtherBranches[u_] :=
    (If[StackEmpty[], True, (ApplyToStack[u]; Imply[Pop[]])]);
```

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

$$A\sigma_1 = A\sigma_2 = A\sigma_3 = True$$

$$B\sigma_1 = false, B\sigma_2\sigma_4 = B\sigma_2\sigma_5 = B\sigma_3\sigma_6 = True$$

$$C\sigma_2\sigma_4 = C\sigma_2\sigma_5 = false, C\sigma_3\sigma_6\sigma_7 = True$$

$$\begin{tikzcd}
 & & (A \wedge B) \wedge C|\emptyset \arrow[d] & & \\
 & & A \wedge B|C \arrow[d] & & \\
 & & A|B, C \arrow[dll, "\sigma_1"'] \arrow[d, "\sigma_2"] \arrow[drr, "\sigma_3"] & & \\
\Box\,|B\sigma_1, C\sigma_1 \arrow[d] & & \Box\,|B\sigma_2, C\sigma_2 \arrow[d] & & \Box\,|B\sigma_3, C\sigma_3 \arrow[d] \\
B\sigma_1|C\sigma_1 \arrow[d] & & B\sigma_2|C\sigma_2 \arrow[dl, "\sigma_4"'] \arrow[dr, "\sigma_5"] & & B\sigma_3|C\sigma_3 \arrow[d, "\sigma_6"] \\
Fail & \Box\,|C\sigma_2\sigma_4 \arrow[d] & & \Box\,|C\sigma_2\sigma_5 \arrow[d] & \Box\,|C\sigma_3\sigma_6 \arrow[d] \\
 & C\sigma_2\sigma_4|\emptyset \arrow[d] & & C\sigma_2\sigma_5|\emptyset \arrow[d] & C\sigma_3\sigma_6|\emptyset \arrow[d, "\sigma_7"] \\
 & Fail & & Fail & \Box\,|\emptyset
\end{tikzcd}$$

For the examples in Section 2 and in the Appendix, only the success steps are shown. All proof steps that lead to dead ends have been deleted.

4.2. Rewrite phase

Five rewriting tactics are used in Analytica:

1. When the left hand side of an equation in the hypothesis appears in the sequent, it is replaced by the right hand side of the equation.

$$\sum_{k=0}^{n} \frac{2^k}{1+x^{2^k}} = \frac{1}{-1+x} + \frac{2 \cdot 2^n}{1-x^{2 \cdot 2^n}} \Longrightarrow \frac{2 \cdot 2^n}{1+x^{2 \cdot 2^n}} + \left(\sum_{k=0}^{n} \frac{2^k}{1+x^{2^k}}\right) = \frac{1}{-1+x} + \frac{4 \cdot 2^n}{1-x^{4 \cdot 2^n}}$$

```
substitute using equation
```

$$\sum_{k=0}^{n} \frac{2^k}{1+x^{2^k}} = \frac{1}{-1+x} + \frac{2\cdot 2^n}{1-x^{2\cdot 2^n}} \Longrightarrow \frac{1}{-1+x} + \frac{2\cdot 2^n}{1-x^{2\cdot 2^n}} + \frac{2\cdot 2^n}{1+x^{2\cdot 2^n}} = \frac{1}{-1+x} + \frac{4\cdot 2^n}{1-x^{4\cdot 2^n}}$$

2. Rewrite a trigonometric expression to an equivalent form.

   Given that a is an odd integer, $\alpha = round(a^m x)$,

   $m \le n \Longrightarrow -\cos(\pi a^n x)+(-1)^\alpha \left(1+\cos(\pi a^{-m+n}(a^m x-\alpha))\right)+\cos(\pi a^{-m+n}(1+\alpha)) = 0$

   `rewrite trigonometric expressions`

   $$True$$

3. Move all terms in equations or inequalities to left hand side and factor the expression.

   $$\frac{(-1+x_3)^2\,(-1+{y_2}^2+{y_3}^2)}{(-1+y_3)^2} = -1+{x_3}^2+\frac{(-1+x_3)^2{y_2}^2}{(-1+y_3)^2} \Longrightarrow x_3-y_3=0$$

   `rewrite as`

   $$\frac{2\,(-1+x_3)\,(x_3-y_3)}{-1+y_3} = 0 \Longrightarrow x_3-y_3=0$$

4. Solve linear equations.

   $$c+bx+ax^2=0 \wedge c+by+ay^2=0 \Longrightarrow x-y=0 \vee a=0 \vee b+a\,(x+y)=0$$

   `solve linear equation`

   $$c=-\left(x\,(b+ax)\right) \wedge c=-\left(y\,(b+ay)\right) \Longrightarrow x-y=0 \vee a=0 \vee b+a\,(x+y)=0$$

5. Replace a user defined function by its definition.

   $$0 < \pi a^m b^m + (1-ab)\,\mathrm{Abs}(S(m))$$

   `open definition`

   $$0 < \pi a^m b^m + (1-ab)\,\mathrm{Abs}\Big(\sum_{n=0}^{-1+m} \frac{b^n\left(-\cos(\pi a^n x)+\cos(\pi a^n\,(x+h))\right)}{h}\Big)$$

5. Simplification

The simplification phase of Analytica uses relatively uncomplicated techniques, but is nevertheless very powerful. Since the number of rules is quite large, this section contains only a few typical examples of such rules. In addition to the rules given below, special simplification rules are included in Analytica in order to handle those functions that are often used, like sum, product, limit, Abs, Min, Max, etc.

5.1. Rules for reducing expressions involving equations and inequalities.

```
InequalityRules = {
```

- Write inequalities in standard form.

```
(a_ != b_) :> not[a == b],

(a_ > b_) :>  (b < a),

(a_ >= b_) :> (b <= a),
```

- Simplify inequalities involving 0 and a product.

```
(x_ a_ < 0) :> or[and[0 < x, a < 0], and[0 < a, x < 0]],

(x_ a_ <=  0) :> or[and[0 <= x, a <= 0], and[0 <= a, x <= 0]],

(0 < x_ a_) :> or[and[0 < x, 0 < a], and[a < 0, x < 0]],

(0 <= x_ a_) :> or[and[0 <= x, 0 <= a], and[a <= 0, x <= 0]],
```

- Remove a common additive term from both sides of an inequality.

```
(x_. + a_ < y_. + a_) :> (x < y),

(x_. + a_ <= y_. + a_) :> (x <= y),
```

- Remove a common factor from both sides of an inequality.

```
(x_. a_ < y_. a_) :> or[and[0 < a, x < y], and[a < 0, y < x]],

(x_. a_  <=  y_. a_) :> or[and[0 <= a, x <= y], and[a <= 0, y <= x]],
```

- Simplify inequalities involving a power.

```
(0 <= a_^n_) :> True /; simplify[0<=a],

(0 < a_^n_) :> True /; simplify[0<a],

(a_^n_ <= 0) :> False /; simplify[0<a],

(a_^n_ < 0) :> False/; simplify[0<=a],

(0 <= a_^n_) :> or[0<=a, Even[n]]/;integer[n],
     (a_^n_ <= 0) :> or[a==0, and[a<0, not[Even[n]]]]/;integer[n],

     (0 < a_^n_) :> or[0<a, and[a<0, Even[n]]]/;integer[n],

     (a_^n_ < 0) :> and[a<0, not[Even[n]]]/;integer[n],
     };
```

```
EquationRules = {
```

- Simplify a product that is 0.

```
     (a_ b_ == 0) :> or[a == 0, b == 0],
```

- Remove a common additive term from both sides of an equation .

```
     (x_. + a_ == y_. + a_) :> (x == y),
```

- Remove a common factor.

```
     (x_. a_  ==  y_. a_) :> or[a == 0, x == y],

     (x_. a_^n1_. == y_. a_^n2_.) :>
             or[a^n1 == 0, x == y a^(n2-n1)],
```

- Simplify an equation that involves a power.

```
     (a_^n_ == 0) :> and[a==0, n>0]
     };
```

5.2. Decide inequalities by using upper and lower bound information.

If a has a negative upper bound, then $a < 0$ is true, while $a > 0$ and $a = 0$ are both false. The function `Lower(Upper)` gives a set of lower(upper) bounds for its argument and will be discussed in Section 7. `ContainsPositive[s]` is True when the set `s` contains a positive element, and `ContainsNonNegative[s]` is True when `s` contains a non-negative element.

```
RulesForRelations = {

f1_ <= f2_ :> False /; ContainsPositive[Lower[f1-f2]],
f1_ <= f2_ :> True /; ContainsNonNegative[Lower[f2-f1]],

f1_ < f2_ :> False /; ContainsNonNegative[Lower[f1-f2]],

f1_ < f2_ :> True /; ContainsPositive[Lower[f2-f1]],

f1_ == f2_ :> False /;
        (ContainsPositive[Lower[f1-f2]] || ContainsPositive[Lower[f2-f1]])

};
```

5.3. Simplify conjunctions and disjunctions that involve inequalities

In $A \wedge B$, A can be simplified assuming B is True.
In $A \vee B$, A can be simplified assuming B is false.

```
RulesUsingContext = {

or[a___, b_ < c_, d___] :>
        or[b<c, or[a, d] /. {c<b -> c!=b, b<c -> False,
                            c<=b -> True, b<=c -> c==b}];

or[a___, b_ <= c_, d___] :>
        or[b<=c, or[a, d] /. {c<b -> True, b<c -> False, c<=b -> True,
                              b<=c -> False, c==b -> False, b==c -> False}],

or[a___, b_ == c_, d___] :>
        or[b==c, or[a, d] /. {c==b -> False, b==c -> False}],

and[a___, b_ < c_, d___] :>
        and[b<c, and[a, d] /. {b<c -> True, c<b -> False, b<=c -> True,
                              c<=b -> False, c==b -> False, b==c -> False}],

and[a___, b_ <= c_, d___] :>
        and[b<=c, and[a, d] /. {b<c -> b!=c, c<b -> False,
                                b<=c -> True, c<=b -> b==c}],

and[a___, b_ == c_, d___] :>
        and[b==c, and[a, d] /. {b==c -> True, c==b -> True, b<c -> False,
                                c<b -> False, b<=c -> True, c<=b -> True}]
}
```

5.4. Examples illustrating the simplification phase

The following examples shows the combined power of the techniques we discussed above.

Let's assume that $a > 1, b > 0, ab > 1 + \frac{3}{2}\pi$.

The first example illustrates the simplification of inequalities and summations.

$$-\left(\pi b^m\right) + \frac{(-1+ab)\,\mathrm{Abs}(\pi)\mathrm{Abs}(-1+x-round(x))\left(\sum_{n=0}^{-1+m} b^n \mathrm{Abs}(a)^n\right)}{\mathrm{Abs}(a)^m\,(1-x+round(x))} < 0$$

`reduces to`

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

`simplify summations`

$$-\frac{a^m b^m + \frac{-1+a^m b^m}{-1+ab} - \frac{ab(-1+a^m b^m)}{-1+ab}}{a^m} < 0$$

`reduces to`

$$True$$

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

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

`simplify limits`

$$-\infty < 0$$

`reduces to`

$$True$$

6. 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 in example proofs. A few of the rules for summation are listed below. Analogous rules for products are also included in Analytica.

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.

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

6.1. Rules for simplifying summations

```
SumSimplifyRules := {
```

- Calculate the summation directly when each term has the same value.

```
sum[n_?NumberQ, {v_, min_, max_}] :> n (max-min+1),
```

- Factor a constant from each term of the summation.

```
sum[a_ f_, {v_, n__}] :> a sum[f, {v, n}] /; FreeQ[a, v],
```

- Merge summations with same range.

```
n1_. sum[a_, range_] + n2_. sum[b_, range_] :> sum[(n1 a + n2 b), range],
```

- Append two summations with the same term and adjacent ranges.

```
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.

```
a_. sum[f_, {v_, n1_, n0_}] + b_. sum[f_, {v_, n2_, n0_}] :>
    a sum[f, {v, n1, n2-1}] /; simplify[a + b == 0],

a_. sum[f_, {v_, n0_, n1_}] + b_. sum[f_, {v_, n0_, n2_}] :>
    a sum[f, {v, n2+1, n1}] /; simplify[a + b == 0],
```

- Change the range when the lower bound exceeds the upper bound.

```
sum[f_, {v_, min_, max_}] :>
    -sum[f, {v, max+1, min-1}] /; simplify[min > max]

};
```

6.2. Rules for rewriting summations

```
SumRewriteRules = {
```

- Calculate the sum of a geometric series $\sum_{k=n}^{m} a^k = \frac{a^{m+1}-a^n}{a-1}$.

```
  sum[a_, {v_, min_, max_}] :>
      (a^((max + 1)/v) - a^(min/v))/(a^(1/v) - 1) /; FreeQ[a^(1/v), v]],
```

- Calculate the sum of a binomial series $\sum_{k=0}^{n} \binom{n}{k} a^k = (1+a)^n$.

```
  sum[a_ Binomial[n_, k_], {k_, n0_, n_}] :>
      (1 + a^(1/k))^n - sum[a Binomial[n, k], {k, 0, n0-1}] /;
      FreeQ[a^(1/k), k],
```

- Split off additional terms in order to make the range of summation simpler.

```
  sum[a_, {v_, n1_, n2_ + n_Integer}] :>
      sum[a, {v, n1, n2}] + sum[a, {v, n2 + 1, n2 + n}],
```

- Decrease the index of summation $\sum_{k=n_1}^{n_2} f(k+1) = \sum_{k=n_1+1}^{n_2+1} f(k)$.

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

- Increase the index of summation $\sum_{k=n_1}^{n_2} f(k-1) = \sum_{k=n_1-1}^{n_2-1} f(k)$.

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

6.3. A summation example

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

```
simplify summations
```

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

```
reduces to
```

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

```
simplify summations
```

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

```
reduces to
```

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

```
simplify summations
```

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

```
reduces to
```

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

```
simplify summations
```

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

```
reduces to
```

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

6.4. Gosper's Algorithm

In many examples, it would be helpful if we could obtain a closed form representation for some summation. *Gosper's algorithm* is able to compute such a representation for a large class of summations. Consequently, we have also integrated this method into our theorem prover. A function g 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 $g(n) = \sum_{k=1}^{n} a_k + g(0)$ [14]. The following example illustrates how Gosper's algorithm is used in Analytica:

Theorem :

$$(|x| > 1 \Rightarrow \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k^2+(2x^2+1)\,k+x^2\,(x^2+1)}\right) < \frac{1}{2})$$

Proof :

$$|x| > 1 \Longrightarrow \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k^2+x^2\,(1+x^2)+k\,(1+2x^2)}\right) < \frac{1}{2}$$

`reduces to`

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

`calculate summation with Gosper's Algorithm`

$$1-|x| < 0 \Longrightarrow -\frac{1}{2}+\lim_{n\to\infty}\left(\frac{1}{2+x^2}+\frac{1}{(1+x^2)\,(2+x^2)}-\frac{1}{1+n+x^2}\right) < 0$$

`simplify limits`

$$1-|x| < 0 \Longrightarrow -\frac{1}{2}+\frac{1}{2+x^2}+\frac{1}{(1+x^2)\,(2+x^2)} < 0$$

`reduces to`

$$1-|x| < 0 \Longrightarrow \frac{1-x^2}{2+2x^2} < 0$$

`reduces to`

$$1-|x| < 0 \Longrightarrow 1-x^2 < 0$$

`replace expression with its lower or upper bounds`

$$\mathit{True}$$

□

7. Inequalities

Inequalities play a key role in all areas of analysis. Since Mathematica does not provide any facility for handling inequalities, we have built several techniques into Analytica for reasoning about them. As mentioned in Section 5, some inequality formulas can be reduced in the simplification phase. However, not all valid inequality formulas can be reduced to true in that way. For example, $(a \leq 0 \wedge b \leq a) \rightarrow b \leq 0$ can not be proved by the technique used in simplification phase alone. Other more powerful techniques for deciding satisfiability of inequality formulas must be used in addition.

7.1. SUP-INF method

Bledsoe's *SUP-INF* method [7] 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 subformulas. 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

$$F = (e_1 < 0 \wedge e_2 < 0 \wedge \ldots \wedge e_n < 0 \wedge e_{n+1} \leq 0 \wedge \ldots \wedge e_{n+m} \leq 0),$$

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

$$\bigwedge_i x_1 < A_i \wedge \bigwedge_i B_i < x_1 \wedge \bigwedge_i C_i < 0 \wedge \bigwedge_i x_1 \leq A'_i \wedge \bigwedge_i B'_i \leq x_1 \wedge \bigwedge_i C'_i \leq 0$$

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

$$\bigwedge_{i,j} B_i < A_j \wedge \bigwedge_{i,j} B_i < A'_j \wedge \bigwedge_{i,j} B'_i < A_j \wedge \bigwedge_{i,j} B'_i \leq A'_j \wedge \bigwedge_i C_i < 0 \wedge \bigwedge_i C'_i \leq 0$$

Thus, we have reduced the original problem with k variables to another 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 [17] this method is shown to be complete for all quantifier free linear inequality formulas. The method does not apply directly to nonlinear inequalities, however.

7.2. 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 the case of 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 $c \in \mathit{Upper}(a)$ such that $c \leq b$ is true or that there is some $c \in \mathit{Lower}(b)$ such that $a \leq c$ is true.

In order to handle strict inequalities, we introduce a new function symbol S. $S_U(a)$ is an upper bound of b if a is a strict upper bound of b. Likewise, S_L is used for strict lower bounds. Thus, $S_L(a) \leq b$ and $a \leq S_U(b)$ are both equivalent to $a < b$, $S_U(a)+b$ and $a+S_U(b)$ are equivalent to $S_U(a+b)$, etc. This convention permits the same code to be used for both strict inequalities and nonstrict inequalities.

7.3. Calculating upper or lower bounds for expressions

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 ($|sin(x)| \leq 1$ for all x, etc.).

- code for obtaining bounds from local context information.

```
Given[a_ < b_] := AddUpperBound[a-b, SU[0]];

Given[a_ <= b_] := AddUpperBound[a-b, 0];

AddUpperBound[a_ + b_, c_] :=
    (AddUpperBound[a, c-b]; AddUpperBound[b, c-a]);

AddUpperBound[c_?NumberQ a_, b_] :=
    If[c>0, AddUpperBound[a, b/c], AddLowerBound[a, b/c]];

AddUpperBound[a_, b_] := PrependTo[GivenUpperBoundOf[a], b];

AddLowerBound[a_ + b_, c_] :=
    (AddLowerBound[a, c-b]; AddLowerBound[b, c-a]);

AddLowerBound[c_?NumberQ a_, b_] := AddUpperBound[-c a, -b];

AddLowerBound[a_, b_] := PrependTo[GivenLowerBoundOf[a], b];
```

When proving $(a \leq b) \vee c$, `Given[not[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 $(a \leq 0 \wedge b \leq a) \rightarrow b \leq 0$. When our technique is applied to the right hand side of the formula, the local context is $(a \leq 0 \wedge b \leq a)$ which gives an upper bound of a to b. Thus, the formula can be reduced to $(a \leq 0 \wedge b \leq a) \rightarrow a \leq 0$ after b is replaced by its upper bound. The resulting expression can easily be reduced to True.

- bounds obtained from the monotonicity of some function.

  If f is a monotonically increasing function, and a' is an upper(lower) bound of a, $f(a')$ is an upper(lower) bound of $f(a)$; if f is a monotonically decreasing function and a' is an upper(lower) bound of a, $f(a')$ is a lower(upper) bound of $f(a)$.

  For example:

$$\{a + b' | b' \in \mathit{Upper}(b)\} \subseteq \mathit{Upper}(a + b)$$

$$\{ca' | a' \in \mathit{Upper}(a)\} \subseteq \mathit{Lower}(ca), \mathit{if}\, c \leq 0$$

$$\{a^{b'} | b' \in \mathit{Upper}(b)\} \subseteq \mathit{Upper}(a^b), \mathit{if}\, a \geq 1$$

- bounds obtained from some known bound on the value of a function.

  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:

$$1 \in \mathit{Upper}(\sin(x))$$

$$-1 \in \mathit{Lower}(\cos(x))$$

$$0 \in \mathit{Lower}(|f|)$$

$$x + \frac{1}{2} \in \mathit{Upper}(\mathit{round}(x))$$

$$x - \frac{1}{2} \in \mathit{Lower}(\mathit{round}(x))$$

7.4. Examples to illustrate inequality proofs

1.

$$0 < r \wedge a < b \Longrightarrow a < \frac{a + br}{1 + r}$$

`replace expression with its lower or upper bounds`

$$0 < r \wedge a < b \Longrightarrow a \leq a$$

`reduces to`

$$\mathit{True}$$

2. given that $b > 0$,

$$2b^m - \frac{3\left(\sum_{n=m}^{\infty} b^n \left(1 + \cos(\pi a^{-m+n}(a^m - round(a^m)))\right)\right)}{1 - (a^m - round(a^m))} \leq 0$$

`replace expression with its lower or upper bounds`

$$2b^m - \frac{3b^m\left(1 + \cos(\pi(a^m - round(a^m)))\right)}{1 - (a^m - round(a^m))} \leq 0$$

`reduces to`

$$2 - \frac{3\left(1 + \cos(\pi(a^m - round(a^m)))\right)}{1 - (a^m - round(a^m))} \leq 0$$

`replace expression with its lower or upper bounds`

$$-2\cos(\pi(a^m - round(a^m))) \leq 0$$

`reduces to`

$$0 \leq \cos(\pi(a^m - round(a^m)))$$

which is True.

3. given that $0 < b < 1$,

$$\exists f_0[-f_0 + b^n \mathrm{Abs}(\cos(\pi a^n x)) \leq 0 \wedge IsConstant(f_0, x) \wedge Convergent(\sum_{n=0}^{\infty} f_0)]$$

`and split`
`case 1.1:`

$$\exists f_0[-f_0 + b^n \mathrm{Abs}(\cos(\pi a^n x)) \leq 0]$$

`replace expression with its lower or upper bounds`

$$\exists f_0[-f_0 + b^n \leq 0]$$

`inequality`

$$\{f_0 \rightarrow b^n\}$$

`case 1.2:`

$$Convergent(\sum_{n=0}^{\infty} b^n)$$

`simplify summations`

$$True$$

8. Conclusion

In a related project that we plan to describe in a forthcoming paper, we have managed to prove all of the theorems and examples in Chapter 2 of Ramanujan's Collected Works[4] completely automatically. The techniques that we use are similar to those described in this paper. We believe that the examples that 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 [8], but relatively little of this research seems applicable to inductive proofs in analysis. Perhaps the research that is most relevant is the *rippling* technique developed by Bundy [9].

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 [16]. 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 [13] and HOL [12] 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 last paragraph. We intend to experiment with combining ideas from this paper with Andrews' theorem prover for higher order logic [2] in the near future.

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 correct results in most cases but do not lead to correct results all the time. For example, it will always simplify 0^x to 0. This may lead to a problem in certain circumstances:

```
In[1] := f[x_] := Sum[a[k] x^k, {k, -1, n}]

In[2] := f[x]
Out[2] = Sum[x^k*a[k], {k, -1, n}]

In[3] := f[0]
Out[3] = 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

[1] L.V.Ahlfors, *Complex Analysis*, McGrew-Hill Book Company, 1966, pp18-20.

[2] P.B.Andrews, *On Connections and Higher-Order Logic*, Journal of Automated Reasoning 5, 1989, pp257-291.

[3] R.G.Bartle, *The Elements of Real Analysis*, John Wiley & Sons, Inc., 1964, pp180-183.

[4] B.C.Berndt, *Ramanujan's Notebooks, Part I*, Springer-Verlag, 1985, pp 25-43.

[5] W.W. Bledsoe, *The UT Natural Deduction Prover*, Technique Report ATP-17B, Mathematical Dept., University of Texas at Austin, 1983.

[6] W.W.Bledsoe, *Some Automatic Proofs in Analysis*, Contemporary Mathematics, Vol.29, 1984.

[7] W.W.Bledsoe, P.Bruell, and R.Shostak, *A Prover for General Inequalities*, Technique Report ATP-40A, Mathematical Dept., University of Texas at Austin, 1979.

[8] R.S.Boyer and J.S.Moore, *A Computational Logic*, Academic Press, 1979.

[9] A.Bundy, F.van Harmelen, J.Hesketh, and A.Smaill, *Experiments with Proof Plans for Induction*, Technique Report, Department of Artificial Intelligence, University of Edinburgh, 1988.

[10] W.M.Farmer, J.D.Guttman and F.J.Thayer, *IMPS: AN Interactive Mathematical Proof System*, Technique Report, The MITRE Corporation, 1990.

[11] M.Fitting, *First-order Logic and Automated Theorem Proving*, Springer-Verlag, 1990.

[12] M.Gorden, *HOL: A Machine Oriented Formulation of Higher Order Logic*, Technique Report, Computer Laboratory, University of Cambridge, 1985.

[13] M.Gorden, R.Milner and C.Wadsworth, *Edinburgh LCF: A Mechanised logic of computation*, Lecture Notes in Computer Science Number 78, Springer-Verlag, 1979.

[14] R.W.Gosper, *Indefinite Hypergeometric sums in MACSYMA*, Proc. MACSYMA Users Conference, Berkeley CA, 1977, pp237-252.

[15] R.L.London and D.R.Musser, *The Application of a Symbolic Mathematical System to Program Verification*, Technique Report, USC Information Science Institute.

[16] A.Quaife, *Automated Deduction in von Neumann-Bernays-Gödel Set Theory*, Technique Report, Dept. of Mathematics, Univ. of California at Berkeley, 1989(submitted to the Journal of Automated Reasoning).

[17] R.E.Shostak, *On the SUP-INF Method for Proving Presburger Formulas*, Journal of ACM, vol. 24, No. 4, October 1977, pp.529-543.

[18] P.Suppes and S.Takahashi, *An Interactive Calculus Theorem-prover for Continuity Properties*, Journal of Symbolic Computation, No.7, 1989, pp 573-590.

[19] E.C.Titchmarsh, *The Theory of Functions*, Oxford University Press, 1932, pp351-353.

[20] S. Wolfram. *Mathematica: A System for Doing Mathematics by Computer*, Wolfram Research Inc., 1988.

APPENDIX. More complicated examples proved by Analytica

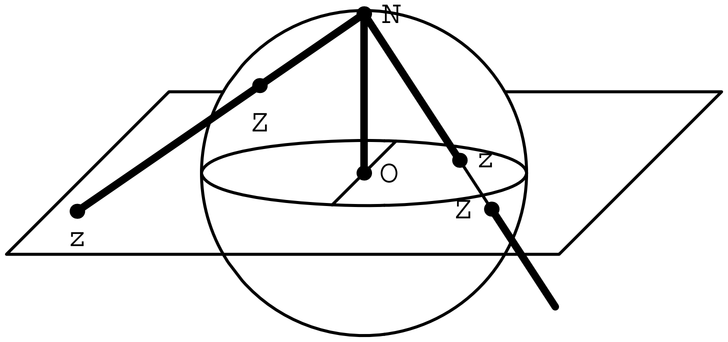
1. Stereographic projection

Statement of the problem:

Consider the function that maps each point (x_1, y_2, z_1) in 3-space to the complex plane C:

$$sp(x_1, x_2, x_3) = \frac{x_1 + ix_2}{1 - x_3}$$

We will use Analytica to prove that this mapping is bijection between the unit sphere S whose equation is $x_1^2 + x_2^2 + x_3^2 = 1$ and the complex plane. We will also prove that this mapping is a projection, i.e. 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 all collinear[1].



outline of the proof:

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

$$\begin{aligned} & {x_1}^2 + {x_2}^2 + {x_3}^2 = 1 \wedge {y_1}^2 + {y_2}^2 + {y_3}^2 = 1 \wedge sp(x_1, x_2, x_3) = sp(y_1, y_2, y_3) \\ \Rightarrow \quad & x_3 = y_3 \wedge x_1 = y_1 \wedge x_2 = y. \end{aligned}$$

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

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

hence,

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

Using the fact that $(x_1, x_2, x_3), (y_1, y_2, y_3)$ 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, $x_1 = y_1, x_2 = y_2$.

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

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

It is sufficient to have

$$x_1 = (1-x_3)z_1, x_2 = (1-x_3)z_2, x_3 = \frac{z_1^2+z_2^2-1}{z_1^2+z_2^2+1}$$

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

$$\begin{vmatrix} 0 & 0 & 1 \\ x_1 & x_2 & 1 \\ z_1 & z_2 & 1 \end{vmatrix} = 0 \bigwedge \begin{vmatrix} 0 & 1 & 1 \\ x_1 & x_3 & 1 \\ z_1 & 0 & 1 \end{vmatrix} = 0 \bigwedge \begin{vmatrix} 0 & 1 & 1 \\ x_2 & x_3 & 1 \\ z_2 & 0 & 1 \end{vmatrix} = 0$$

This follows directly from the definition.

Input for the problem:

```
(* the stereographic projection *)

sp[x1_, x2_, x3_] := over[(x1 + i x2), (1-x3)];

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

unit[x1_, x2_, x3_] := (x1^2 + x2^2 + x3^2 == 1);

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

Prove[imp[and[unit[x1, x2, x3], unit[y1, y2, y3], sp[x1, x2, x3] == sp[y1, y2, y3]],
    and[x3 == y3, x1==y1, x2==y2]]];

(* stereographic projection is an onto mapping *)

Prove[some[{x1, x2, x3}, and[sp[x1, x2, x3] == z1 + z2 i, unit[x1,x2,x3]]]];

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

collinear[{x1_, x2_, x3_}, {y1_, y2_, y3_}, {z1_, z2_, z3_}] :=
        and[Det[{{x1, x2, 1}, {y1, y2, 1}, {z1, z2, 1}}] == 0,
            Det[{{x1, x3, 1}, {y1, y3, 1}, {z1, z3, 1}}] == 0,
            Det[{{x2, x3, 1}, {y2, y3, 1}, {z2, z3, 1}}] == 0];

Prove[imp[z == sp[x1, x2, x3],
        collinear[{Rpart[z], Ipart[z], 0}, {x1, x2, x3}, {0,0,1}]]];
```

Theorems proved :

$$(unit(x_1,x_2,x_3) \wedge unit(y_1,y_2,y_3) \wedge sp(x_1,x_2,x_3) = sp(y_1,y_2,y_3) \Rightarrow x_3 = y_3 \wedge x_1 = y_1 \wedge x_2 = y_2)$$
$$\exists\{x_1,x_2,x_3\}[sp(x_1,x_2,x_3) = z_1 + z_2 i \wedge unit(x_1,x_2,x_3)]$$
$$(z = sp(x_1,x_2,x_3) \Rightarrow collinear(\{R(z), I(z), 0\}, \{x_1,x_2,x_3\}, \{0,0,1\}))$$

Theorem :

$$(unit(x_1,x_2,x_3) \wedge unit(y_1,y_2,y_3) \wedge sp(x_1,x_2,x_3) = sp(y_1,y_2,y_3) \Rightarrow x_3 = y_3 \wedge x_1 = y_1 \wedge x_2 = y_2)$$

Proof :

$$x_1{}^2 + x_2{}^2 + x_3{}^2 = 1 \land y_1{}^2 + y_2{}^2 + y_3{}^2 = 1 \land \frac{x_1}{1-x_3} = \frac{y_1}{1-y_3} \land \frac{x_2}{1-x_3} = \frac{y_2}{1-y_3} \Longrightarrow$$
$$x_3 = y_3 \land x_1 = y_1 \land x_2 = y_2$$

reduces to

$$x_1{}^2 + x_2{}^2 + x_3{}^2 = 1 \land y_1{}^2 + y_2{}^2 + y_3{}^2 = 1 \land \frac{x_1}{-1+x_3} = \frac{y_1}{-1+y_3} \land \frac{x_2}{-1+x_3} = \frac{y_2}{-1+y_3} \Longrightarrow$$
$$x_3 = y_3 \land x_1 = y_1 \land x_2 = y_2$$

and split
case 1.1:

$$x_1{}^2 + x_2{}^2 + x_3{}^2 = 1 \land y_1{}^2 + y_2{}^2 + y_3{}^2 = 1 \land \frac{x_1}{-1+x_3} = \frac{y_1}{-1+y_3} \land \frac{x_2}{-1+x_3} = \frac{y_2}{-1+y_3} \Longrightarrow x_3 = y_3$$

rewrite as

$$-1 + x_1{}^2 + x_2{}^2 + x_3{}^2 = 0 \land -1 + y_1{}^2 + y_2{}^2 + y_3{}^2 = 0 \land \frac{-x_1 + y_1 - x_3 y_1 + x_1 y_3}{(-1+x_3)(-1+y_3)} = 0 \land$$
$$\frac{-x_2 + y_2 - x_3 y_2 + x_2 y_3}{(-1+x_3)(-1+y_3)} = 0 \Longrightarrow$$
$$x_3 - y_3 = 0$$

reduces to

$$-1 + x_1{}^2 + x_2{}^2 + x_3{}^2 = 0 \land -1 + y_1{}^2 + y_2{}^2 + y_3{}^2 = 0 \land (1-x_3)\,y_1 + x_1\,(-1+y_3) = 0 \land$$
$$(1-x_3)\,y_2 + x_2\,(-1+y_3) = 0 \Longrightarrow$$
$$x_3 - y_3 = 0$$

solve linear equation

$$x_1{}^2 = -\left(-1 + x_2{}^2 + x_3{}^2\right) \land y_1{}^2 = -\left(-1 + y_2{}^2 + y_3{}^2\right) \land x_1 = \frac{(-1+x_3)\,y_1}{-1+y_3} \land x_2 = \frac{(-1+x_3)\,y_2}{-1+y_3} \Longrightarrow$$
$$x_3 - y_3 = 0$$

substitute using equation

$$-\frac{(-1+x_3)^2\left(-1 + y_2{}^2 + y_3{}^2\right)}{(-1+y_3)^2} = -\left(-1 + x_3{}^2 + \frac{(-1+x_3)^2 y_2{}^2}{(-1+y_3)^2}\right) \Longrightarrow x_3 - y_3 = 0$$

reduces to

$$\frac{(-1+x_3)^2\left(-1 + y_2{}^2 + y_3{}^2\right)}{(-1+y_3)^2} = -1 + x_3{}^2 + \frac{(-1+x_3)^2 y_2{}^2}{(-1+y_3)^2} \Longrightarrow x_3 - y_3 = 0$$

rewrite as

$$\frac{2\,(-1+x_3)\,(x_3 - y_3)}{-1+y_3} = 0 \Longrightarrow x_3 - y_3 = 0$$

reduces to

$$\mathit{True}$$

case 1.2:

$$x_1{}^2 + x_2{}^2 + x_3{}^2 = 1 \wedge y_1{}^2 + y_2{}^2 + y_3{}^2 = 1 \wedge \frac{x_1}{-1+x_3} = \frac{y_1}{-1+y_3} \wedge \frac{x_2}{-1+x_3} = \frac{y_2}{-1+y_3} \wedge x_3 = y_3 \Longrightarrow x_1 = y_1 \wedge x_2 = y_2$$

and split
case 1.2.1:

$$x_1{}^2 + x_2{}^2 + x_3{}^2 = 1 \wedge y_1{}^2 + y_2{}^2 + y_3{}^2 = 1 \wedge \frac{x_1}{-1+x_3} = \frac{y_1}{-1+y_3} \wedge \frac{x_2}{-1+x_3} = \frac{y_2}{-1+y_3} \wedge x_3 = y_3 \Longrightarrow x_1 = y_1$$

substitute using equation

$$x_1{}^2 + x_2{}^2 + y_3{}^2 = 1 \wedge y_1{}^2 + y_2{}^2 + y_3{}^2 = 1 \wedge \frac{x_1}{-1+y_3} = \frac{y_1}{-1+y_3} \wedge \frac{x_2}{-1+y_3} = \frac{y_2}{-1+y_3} \Longrightarrow x_1 = y_1$$

reduces to

$$\mathit{True}$$

case 1.2.2:

$$x_1{}^2 + x_2{}^2 + x_3{}^2 = 1 \wedge y_1{}^2 + y_2{}^2 + y_3{}^2 = 1 \wedge \frac{x_1}{-1+x_3} = \frac{y_1}{-1+y_3} \wedge \frac{x_2}{-1+x_3} = \frac{y_2}{-1+y_3} \wedge x_3 = y_3 \wedge x_1 = y_1 \Longrightarrow x_2 = y_2$$

substitute using equation

$$x_2{}^2 + y_1{}^2 + y_3{}^2 = 1 \wedge y_1{}^2 + y_2{}^2 + y_3{}^2 = 1 \wedge \frac{x_2}{-1+y_3} = \frac{y_2}{-1+y_3} \Longrightarrow x_2 = y_2$$

reduces to

$$\mathit{True}$$

□

Theorem :

$$\exists\{x_1, x_2, x_3\}[sp(x_1, x_2, x_3) = z_1 + z_2 i \wedge unit(x_1, x_2, x_3)]$$

Proof :

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

check denominator

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

reduces to

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

and split
case 1.1:

$$\forall x_3 [1 - x_3 = 0] \Longrightarrow \mathit{False}$$

rewrite as

$$\forall x_3[-(-1+x_3)=0] \Longrightarrow \mathit{False}$$

reduces to

$$\forall x_3[-1+x_3=0] \Longrightarrow \mathit{False}$$

solve linear equation

$$\forall x_3[x_3=1] \Longrightarrow \mathit{False}$$

add restriction:

$$x_3 \neq 1$$

case 1.2:

$$\exists x_1 \exists x_2 \exists x_3[1-x_3=0 \vee -\frac{x_1}{-1+x_3}=z_1 \wedge -\frac{x_2}{-1+x_3}=z_2 \wedge {x_1}^2+{x_2}^2+{x_3}^2=1]$$

reduces to

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

and split
case 1.2.1:

$$\exists x_1 \exists x_3[-\frac{x_1}{-1+x_3}=z_1]$$

rewrite as

$$\exists x_1 \exists x_3[-\frac{x_1-z_1+x_3 z_1}{-1+x_3}=0]$$

reduces to

$$\exists x_1 \exists x_3[x_1+(-1+x_3)\,z_1=0]$$

equation

$$\{x_1 \rightarrow -((-1+x_3)\,z_1)\}$$

case 1.2.2:

$$\exists x_2 \exists x_3[-\frac{x_2}{-1+x_3}=z_2 \wedge {x_2}^2+{x_3}^2+(-1+x_3)^2{z_1}^2=1]$$

and split
case 1.2.2.1:

$$\exists x_2 \exists x_3[-\frac{x_2}{-1+x_3}=z_2]$$

rewrite as

$$\exists x_2 \exists x_3[-\frac{x_2-z_2+x_3 z_2}{-1+x_3}=0]$$

reduces to

$$\exists x_2 \exists x_3[x_2+(-1+x_3)\,z_2=0]$$

equation

$$\{x_2 \to -((-1+x_3)\,z_2)\}$$

case 1.2.2.2:

$$\exists x_3[{x_3}^2 + (-1+x_3)^2{z_1}^2 + (-1+x_3)^2{z_2}^2 = 1]$$

reduces to

$$\exists x_3[{x_3}^2 + (-1+x_3)^2\left({z_1}^2 + {z_2}^2\right) = 1]$$

rewrite as

$$\exists x_3[(-1+x_3)\left(1 + x_3 - {z_1}^2 + x_3{z_1}^2 - {z_2}^2 + x_3{z_2}^2\right) = 0]$$

reduces to

$$\exists x_3[1 + x_3 + (-1+x_3)\left({z_1}^2 + {z_2}^2\right) = 0]$$

solve linear equation

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

equation

$$\{x_3 \to \frac{-1+{z_1}^2+{z_2}^2}{1+{z_1}^2+{z_2}^2}\}$$

check restrict on instantiation

$$\frac{-1+{z_1}^2+{z_2}^2}{1+{z_1}^2+{z_2}^2} = 1 \Longrightarrow \mathit{False}$$

reduces to

$$\mathit{True}$$

□

Theorem :

$$(z = sp(x_1, x_2, x_3) \Rightarrow collinear(\{R(z), I(z), 0\}, \{x_1, x_2, x_3\}, \{0, 0, 1\}))$$

Proof :

$$z = \frac{x_1}{1-x_3} + \frac{x_2}{1-x_3}i \Longrightarrow -(x_1 I(z)) + x_2 R(z) = 0 \wedge x_1 - R(z) + x_3 R(z) = 0 \wedge x_2 - I(z) + x_3 I(z) = 0$$

reduces to

$$z = \frac{x_1}{1-x_3} + \frac{x_2}{1-x_3}i \Longrightarrow -(x_1 I(z)) + x_2 R(z) = 0 \wedge x_1 + (-1+x_3)\,R(z) = 0 \wedge x_2 + (-1+x_3)\,I(z) = 0$$

reduces to

$$z = -\frac{x_1}{-1+x_3} - \frac{x_2 i}{-1+x_3} \Longrightarrow$$
$$-(x_1 I(z)) + x_2 R(z) = 0 \wedge x_1 + (-1+x_3)\,R(z) = 0 \wedge x_2 + (-1+x_3)\,I(z) = 0$$

and split
case 1.1:

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

substitute using equation

$$\mathit{True}$$

case 1.2:

$$z = -\frac{x_1}{-1 + x_3} - \frac{x_2 i}{-1 + x_3} \wedge -(x_1 I(z)) + x_2 R(z) = 0 \Longrightarrow$$
$$x_1 + (-1 + x_3)\, R(z) = 0 \wedge x_2 + (-1 + x_3)\, I(z) = 0$$

and split
case 1.2.1:

$$z = -\frac{x_1}{-1 + x_3} - \frac{x_2 i}{-1 + x_3} \wedge -(x_1 I(z)) + x_2 R(z) = 0 \Longrightarrow x_1 + (-1 + x_3)\, R(z) = 0$$

substitute using equation

$$\mathit{True}$$

case 1.2.2:

$$z = -\frac{x_1}{-1 + x_3} - \frac{x_2 i}{-1 + x_3} \wedge -(x_1 I(z)) + x_2 R(z) = 0 \wedge x_1 + (-1 + x_3)\, R(z) = 0 \Longrightarrow$$
$$x_2 + (-1 + x_3)\, I(z) = 0$$

substitute using equation

$$\mathit{True}$$

□

2. An Everywhere Continuous but Nowhere Differentiable Function

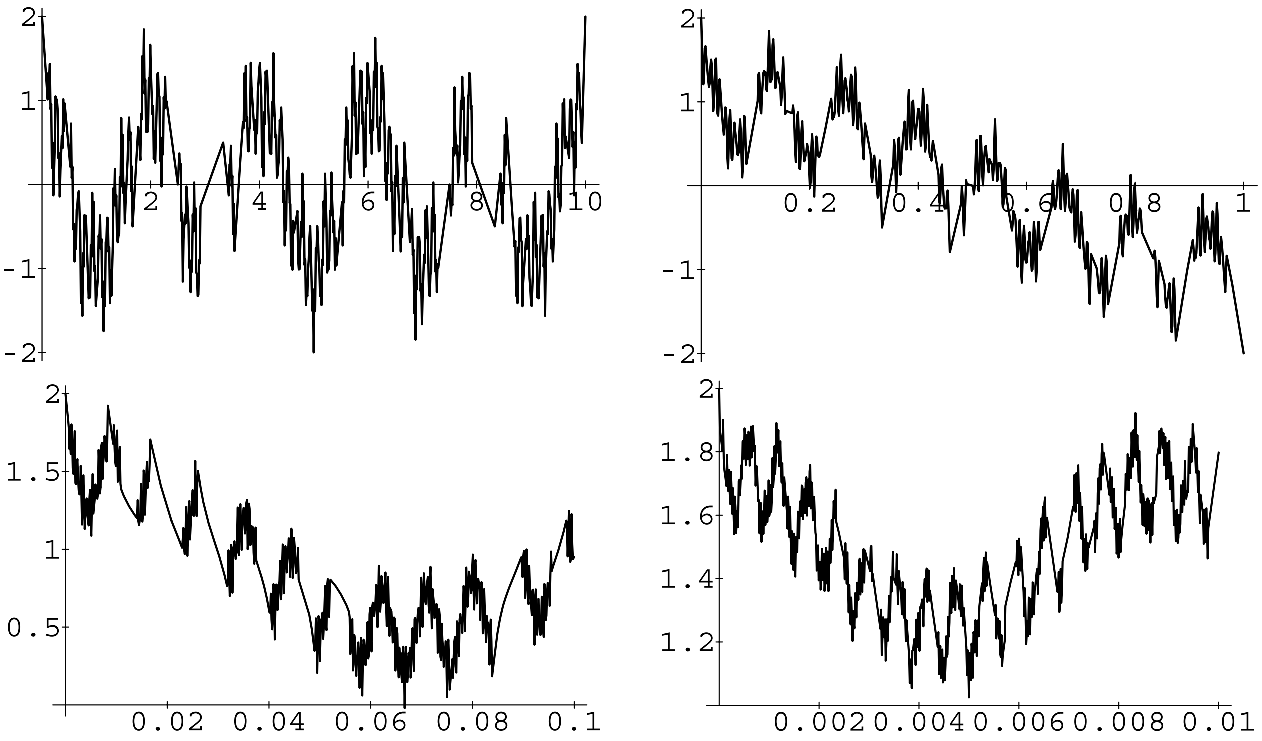
Statement of the problem:

Weierstrass's non-differentiable function is defined be the series

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

where $0 < b < 1$, and a is a odd positive integer. When $ab > 1 + \frac{3}{2}\pi$, the derivative of the function does not exist for any value of x.

It is interesting to plot the behavior of this function using Mathematica. Titchmarsh [19] described the function as having "infinitely many infinitesimal crinkles". Let $f(x) = \sum_{n=0}^{10} \cos(15^n \pi x) 0.5^n$. The digrams plots $f(x)$ over ranges [0,10], [0, 1], [0, 0.1], and [0, 0.01] respectively.



Outline of the proof:

If $0 < b < 1$, $\sum_{n=0}^{\infty} b^n$ is convergent. Hence the series $\sum_{n=0}^{\infty} b^n \cos(a^n \pi x)$ 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

$$\begin{aligned}
& \frac{f(x+h) - f(x)}{h} \\
= & \sum_{n=0}^{\infty} b^n \frac{\cos\{a^n \pi (x+h)\} - \cos(a^n \pi x)}{h} \\
= & \sum_{n=0}^{m-1} b^n \frac{\cos\{a^n \pi (x+h)\} - \cos(a^n \pi x)}{h} + \sum_{n=m}^{\infty} b^n \frac{\cos\{a^n \pi (x+h)\} - \cos(a^n \pi x)}{h} \\
= & S_m + R_m
\end{aligned}$$

Because $\forall xy. \mid \cos(x) - \cos(y) \mid \leq \mid x - y \mid$,

$$\mid S_m \mid \leq \sum_{n=0}^{m-1} b^n \frac{\mid \cos\{a^n \pi(x+h)\} - \cos(a^n \pi x) \mid}{\mid h \mid} \leq \sum_{n=0}^{m-1} \pi a^n b^n = \pi \frac{a^m b^m - 1}{ab-1} < \pi \frac{a^m b^m}{ab-1}$$

Take α_m as the nearest integer to $a^m x$, i.e. $-\frac{1}{2} \leq a^m x - \alpha_m < \frac{1}{2}$. Let $\xi_m = a^m x - \alpha_m$ and $h = \frac{1-\xi_m}{a^m}$.

$$0 < h \leq \frac{3}{2a^m}.$$

When $n \geq m$,

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

Also

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

Hence

$$R_m = \frac{(-1)^{\alpha_m+1}}{h} \sum_{n=m}^{\infty} b^n \{1 + \cos(a^{n-m}\pi\xi_m)\}.$$

$$\mid R_m \mid = \frac{1}{h} \sum_{n=m}^{\infty} b^n \{1 + \cos(a^{n-m}\pi\xi_m)\} \geq \frac{b^m\{1 + \cos(\pi\xi_m)\}}{h} \geq \frac{2}{3} a^m b^m$$

So

$$\mid \frac{f(x+h) - f(x)}{h} \mid \geq \mid R_m \mid - \mid S_m \mid > \left(\frac{2}{3} - \frac{\pi}{ab-1}\right) a^m b^m$$

If $ab > 1 + \frac{3}{2}\pi$, 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_] := sum[b^n Cos[a^n Pi x], {n, 0, 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 b > 1];
Given[b<1];
Given[n>0];
Given[m>0];
Given[a>1];
```

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

(* several auxiliary functions *)

AddDefinition[S[m_] == sum[b^n (Cos[a^n Pi (x+h)] - Cos[a^n Pi x]) / h, {n, 0, m-1}]];

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

AddDefinition[diff[h0_, f0_, x_] == (f0[x+h0] - f0[x]) / h0];

(* a given value for h *)

h = (1 - xi[m]) / a^m;

(* continuity of the function *)

Prove[Continuous[f[x], {x, x0}]];

(* several lemmas needed *)

ProveAndSave[diff[h,f,x]==R[m]+S[m]];

ProveAndSave[Abs[R[m]]>=2/3 a^m b^m];

ProveAndSave[Abs[S[m]] < Pi a^m b^m / (a b - 1)];

(* h tends to 0 *)

Prove[all[epsilon, imp[epsilon>0, limit[Abs[h], {m, infinity}]<epsilon]]];

(* the derivative tends to infinity *)

Prove[all[M, limit[Abs[diff[h,f,x]], {m,infinity}]>M]];
```

Given properties:

$$0 < \pi$$
$$0 < b$$
$$0 < m$$
$$0 < n$$
$$1 - a < 0$$
$$-1 + b < 0$$
$$1 - ab < 0$$
$$1 + \frac{3\pi}{2} - ab < 0$$

Definitions used in proof:

$$S(m) = \frac{a^m \left(\sum_{n=0}^{-1+m} b^n \left(-\cos(\pi a^n x) + \cos(\pi a^{-m+n}\,(1+\alpha))\right)\right)}{1 - \xi(m)}$$

$$R(m) = -\frac{(-1)^{\alpha} a^m \left(\sum_{n=m}^{\infty} b^n \left(1 + \cos(\pi a^{-m+n} \xi(m))\right)\right)}{1 - \xi(m)}$$

$$\mathit{diff}(h, f, x) = \frac{-f(x) + f(h + x)}{h}$$

Abbreviations used in proof:

$$h \leftrightarrow \frac{1 - a^m x + \mathit{round}(a^m x)}{a^m}$$

$$\xi(m) \leftrightarrow a^m x - \mathit{round}(a^m x)$$

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

$$\alpha \leftrightarrow \mathit{round}(a^m x)$$

Theorems proved :

$$\mathit{Continuous}(f(x), \{x, x_0\})$$

$$\mathit{diff}(h, f, x) = R(m) + S(m)$$

$$\mathrm{Abs}(R(m)) \geq \frac{2a^m b^m}{3}$$

$$\mathrm{Abs}(S(m)) < \frac{\pi a^m b^m}{ab - 1}$$

$$\forall \epsilon [\left(\epsilon > 0 \Rightarrow \lim_{m \to \infty} (\mathrm{Abs}(h)) < \epsilon\right)]$$

$$\forall M [\lim_{m \to \infty} (\mathrm{Abs}(\mathit{diff}(h, f, x))) > M]$$

Theorem :

$$\mathit{Continuous}(f(x), \{x, x_0\})$$

Proof :

$$\mathit{Continuous}(f(x), \{x, x_0\})$$

matching lemma

$$\forall f \forall n \forall x \forall x_0 \forall \mathit{min} [$$

$$\mathit{Continuous}(f, \{x, x_0\}) \wedge \mathit{UniformlyConvergent}(\sum_{n=\mathit{min}}^{\infty} f, \{x, x_0 - \sigma, x_0 + \sigma\})$$

$$\Rightarrow \mathit{Continuous}(\sum_{n=\mathit{min}}^{\infty} f, \{x, x_0\})]$$

with

$$\{f \to b^n \cos(\pi a^n x), n \to n, x \to x, x_0 \to x_0, \mathit{min} \to 0\}$$

back chaining

$$\mathit{UniformlyConvergent}(f(x), \{x, -\sigma + x_0, \sigma + x_0\})$$

matching lemma

$$\forall f \forall n \forall x \forall c_1 \forall c_2 \forall min[$$
$$\exists f_0[(-x + c_1 < 0 \wedge -c_2 + x < 0 \Rightarrow -f_0 + \mathrm{Abs}(f) \le 0) \wedge \mathit{IsConstant}(f_0, x) \wedge \mathit{Convergent}(\sum_{n=min}^{\infty} f_0)]$$
$$\Rightarrow \mathit{UniformlyConvergent}(\sum_{n=min}^{\infty} f, \{x, c_1, c_2\})]$$

with

$$\{c_2 \to \sigma + x_0, min \to 0, f \to b^n \cos(\pi a^n x), n \to n, x \to x, c_1 \to -\sigma + x_0\}$$

back chaining

$$\exists f_0[(-\sigma - x + x_0 < 0 \wedge x - (\sigma + x_0) < 0 \Rightarrow -f_0 + \mathrm{Abs}(b)^n \mathrm{Abs}(\cos(\pi a^n x)) \le 0) \wedge \mathit{IsConstant}(f_0, x) \wedge \mathit{Convergent}(\sum_{n=0}^{\infty} f_0)]$$

reduces to

$$\exists f_0[(-\sigma - x + x_0 < 0 \wedge -\sigma + x - x_0 < 0 \Rightarrow -f_0 + b^n \mathrm{Abs}(\cos(\pi a^n x)) \le 0) \wedge \mathit{IsConstant}(f_0, x) \wedge \mathit{Convergent}(\sum_{n=0}^{\infty} f_0)]$$

and split
case 1.1:

$$-\sigma - x + x_0 < 0 \wedge -\sigma + x - x_0 < 0 \Longrightarrow \exists f_0[-f_0 + b^n \mathrm{Abs}(\cos(\pi a^n x)) \le 0]$$

replace expression with its lower or upper bounds

$$-\sigma - x + x_0 < 0 \wedge -\sigma + x - x_0 < 0 \Longrightarrow \exists f_0[-f_0 + b^n \le 0]$$

inequality

$$\{f_0 \to b^n\}$$

case 1.2:

$$\mathit{Convergent}(\sum_{n=0}^{\infty} b^n)$$

simplify summations

$$\mathit{True}$$

□

Theorem :

$$\mathit{diff}(h, f, x) = R(m) + S(m)$$

Proof :

$$\mathit{diff}(h, f, x) = R(m) + S(m)$$

rewrite as

$$-\left(R(m)+S(m)-\mathit{diff}(h,f,x)\right)=0$$

reduces to

$$R(m)+S(m)-\mathit{diff}(h,f,x)=0$$

open definition

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

reduces to

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

simplify summations

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

reduces to

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

simplify summations

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

reduces to

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

simplify summations

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

reduces to

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

simplify summations

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

reduces to

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

matching lemma

$$\forall k\forall low\forall up\forall f[\left((-k+low\leq 0\wedge k-up\leq 0\Rightarrow f=0)\Rightarrow\sum_{k=low}^{up} f=0\right)]$$

with

$$\{k\to n,\, low\to m,\, up\to\infty,$$
$$f\to b^n\left(-\cos(\pi a^n x)+(-1)^{\alpha}\left(1+\cos(\pi a^{-m+n}\xi(m))\right)+\cos(\pi a^{-m+n}(1+\alpha))\right)\}$$

back chaining

$$m-n\leq 0\wedge-\infty\leq 0\Longrightarrow b^n\left(-\cos(\pi a^n x)+(-1)^{\alpha}\left(1+\cos(\pi a^{-m+n}\xi(m))\right)+\cos(\pi a^{-m+n}(1+\alpha))\right)=0$$

reduces to

$$m-n\leq 0\Longrightarrow-\cos(\pi a^n x)+(-1)^{\alpha}\left(1+\cos(\pi a^{-m+n}\xi(m))\right)+\cos(\pi a^{-m+n}(1+\alpha))=0$$

rewrite trigonometric expressions

$$\mathit{True}$$

□

Theorem :

$$\mathrm{Abs}(R(m))\geq\frac{2a^m b^m}{3}$$

Proof :

$$\mathrm{Abs}(R(m))\geq\frac{2a^m b^m}{3}$$

reduces to

$$\frac{2a^m b^m}{3}-\mathrm{Abs}(R(m))\leq 0$$

rewrite as

$$\frac{2a^m b^m-3\mathrm{Abs}(R(m))}{3}\leq 0$$

reduces to

$$2a^m b^m - 3\mathrm{Abs}(R(m)) \leq 0$$

open definition

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

reduces to

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

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

$$\forall x[\frac{-\pi}{2} - x \leq 0 \wedge \frac{-\pi}{2} + x \leq 0 \Rightarrow 0 \leq \cos(x)]$$

with

$$\{x \rightarrow \pi\xi(m)\}$$

back chaining

$$\frac{-\pi}{2} - \pi\xi(m) \leq 0 \wedge \frac{-\pi}{2} + \pi\xi(m) \leq 0$$

reduces to

$$\mathit{True}$$

$\square$

Theorem :

$$\mathrm{Abs}(S(m)) < \frac{\pi a^m b^m}{ab-1}$$

Proof :

$$\mathrm{Abs}(S(m)) < \frac{\pi a^m b^m}{-1+ab}$$

reduces to

$$-\frac{\pi a^m b^m}{-1+ab} + \mathrm{Abs}(S(m)) < 0$$

rewrite as

$$-\frac{\pi a^m b^m + \mathrm{Abs}(S(m)) - ab\mathrm{Abs}(S(m))}{-1+ab} < 0$$

reduces to

$$-\left(\pi a^m b^m\right) + \left(-1+ab\right)\mathrm{Abs}(S(m)) < 0$$

open definition

$$-\left(\pi a^m b^m\right) + \frac{\left(-1+ab\right)\mathrm{Abs}(a)^m \mathrm{Abs}(\sum_{n=0}^{-1+m} b^n \left(-\cos(\pi a^n x) + \cos(\pi a^{-m+n}\left(1+\alpha\right))\right))}{\mathrm{Abs}(1-\xi(m))} < 0$$

reduces to

$$-\left(\pi b^m\right) + \frac{\left(-1+ab\right)\mathrm{Abs}(\sum_{n=0}^{-1+m} b^n \left(-\cos(\pi a^n x) + \cos(\pi a^{-m+n}\left(1+\alpha\right))\right))}{1-\xi(m)} < 0$$

replace expression with its lower or upper bounds

$$\frac{\left(-1+ab\right)\mathrm{Abs}(\sum_{n=0}^{-1+m} b^n \left(-\cos(\pi a^n x) + \cos(\pi a^{-m+n}\left(1+\alpha\right))\right))}{1-\xi(m)} \leq 0$$

reduces to

$$-\left(\pi b^m\right) + \frac{\left(-1+ab\right)\left(\sum_{n=0}^{-1+m} \mathrm{Abs}(b)^n \mathrm{Abs}(-\cos(\pi a^n x) + \cos(\pi a^{-m+n}\left(1+\alpha\right)))\right)}{1-\xi(m)} < 0$$

reduces to

$$-\left(\pi b^m\right) + \frac{\left(-1+ab\right)\left(\sum_{n=0}^{-1+m} b^n \mathrm{Abs}(\cos(\pi a^n x) - \cos(\pi a^{-m+n}\left(1+\alpha\right)))\right)}{1-\xi(m)} < 0$$

replace expression with its lower or upper bounds

$$-\left(\pi b^m\right) + \frac{\left(-1+ab\right)\mathrm{Abs}(\pi)\mathrm{Abs}(-1+\xi(m))\left(\sum_{n=0}^{-1+m} b^n \mathrm{Abs}(a)^n\right)}{\mathrm{Abs}(a)^m\left(1-\xi(m)\right)} < 0$$

reduces to

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

simplify summations

$$-\frac{a^m b^m + \frac{-1+a^m b^m}{-1+ab} - \frac{ab\left(-1+a^m b^m\right)}{-1+ab}}{a^m} < 0$$

reduces to

$$\mathit{True}$$

$\square$

Theorem :

$$\forall\epsilon[\left(\epsilon > 0 \Rightarrow \lim_{m\to\infty}\left(\mathrm{Abs}(h)\right) < \epsilon\right)]$$

Proof :

$$\forall \epsilon [\epsilon > 0 \Longrightarrow \lim_{m \to \infty} \left(\frac{\mathrm{Abs}(1 - \xi(m))}{\mathrm{Abs}(a)^m} \right) < \epsilon]$$

reduces to

$$\forall \epsilon [0 < \epsilon \Longrightarrow -\epsilon + \lim_{m \to \infty} (h) < 0]$$

replace expression with its lower or upper bounds

$$\forall \epsilon [0 < \epsilon \Longrightarrow -\epsilon + \lim_{m \to \infty} \left(\frac{3}{2a^m} \right) < 0]$$

simplify limits

$$\forall \epsilon [0 < \epsilon \Longrightarrow -\epsilon < 0]$$

reduces to

True

$\square$

Theorem :

$$\forall M [\lim_{m \to \infty} (\mathrm{Abs}(\mathit{diff}(h, f, x))) > M]$$

Proof :

$$\forall M [\lim_{m \to \infty} (\mathrm{Abs}(\mathit{diff}(h, f, x))) > M]$$

reduces to

$$\forall M [M - \lim_{m \to \infty} (\mathrm{Abs}(\mathit{diff}(h, f, x))) < 0]$$

substitute using equation

$$\forall M [M - \lim_{m \to \infty} (\mathrm{Abs}(R(m) + S(m))) < 0]$$

replace expression with its lower or upper bounds

$$\forall M [M - \lim_{m \to \infty} (\mathrm{Abs}(R(m)) - \mathrm{Abs}(S(m))) < 0]$$

replace expression with its lower or upper bounds

$$\forall M [M - \lim_{m \to \infty} \left(-\frac{\pi a^m b^m}{-1 + ab} + \mathrm{Abs}(R(m)) \right) < 0]$$

replace expression with its lower or upper bounds

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

simplify limits

$$-\infty < 0$$

reduces to

True

$\square$