Analytica — An Experiment in Combining Theorem Proving and Symbolic Computation [*]

Andrej Bauer [†] andrej@cs.cmu.edu

Edmund Clarke [†] emc@cs.cmu.edu

Xudong Zhao [‡] xzhao@cs.cmu.edu

February 25, 1997

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 deduce correctness of certain simplification steps that would otherwise not be performed. We describe the structure of Analytica and explain the main techniques that it uses to construct proofs. Analytica has been able to prove several non-trivial theorems. In this paper, we show how it can prove a series of lemmas that lead to Bernstein approximation theorem.

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. For example, in certain situations the theorem prover can guarantee that an expression is not zero, and the symbolic computation system can then divide by it. This would not be possible without the support of a theorem prover.

Analytica is written in the Mathematica programming language and runs in the interactive environment provided by this system [22]. This programming language is based on term-rewriting. A short description of the key features of the language is given in section 3. Each step made by Analytica is the application of a rewriting rule. The rules can be divided into two sets. One set consists of rules that are built into Mathematica like simplification and factorization of polynomials. We call these rules called Mathematica rules. The other set consists of rules that we have added. These rules are called Analytica rules. For example, the inference phase of

[*]This research was sponsored in part by the National Science Foundation under grant no. CCR-8722633, by the Semiconductor Research Corporation under contract 92-DJ-294, and by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant F33615-93-1-1330.

[†]School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213

[‡]Mail Stop: JFT-102, Intel Corporation, 2111 N.E. 25th Ave., Hillsboro, OR 97124

Analytica contains a rule that rewrites the sequent $h \longrightarrow c \vee (a \wedge b)$ into two sequents $h \longrightarrow c \vee a$ and $h \longrightarrow c \vee b$. Another example is a rule in Analytica's simplification phase that rewrites the sequent $h \longrightarrow x \cdot y = 0$ into $h \longrightarrow x = 0 \vee y = 0$. Additional rules are given in section 5.

Since we wanted to generate proofs that were similar to proofs constructed by humans, we have used a variant of the sequent calculus [11, 12] in the inference phase of our theorem prover. However, 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 technique that is complete for linear inequalities and is able to handle a large class of non-linear inequalities as well. This technique is more closely related to the BOUNDER system developed at MIT [19] than to the traditional SUP-INF method of Bledsoe [6]. Another important component of Analytica deals with expressions involving summation and product operators. A large number 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.

Analytica is able to prove several non-trivial examples, such as the basic properties of the stereographic projection [9], and a series of three lemmas that lead to a proof of Weierstrass's example of a continuous nowhere differentiable function [9]. In the appendix we show how Analytica can prove four lemmas from which the Bernstein approximation theorem follows.

There has been relatively little research on theorem proving in analysis. Bledsoe's work in this area [4, 5] 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 [20] have combined a resolution theorem prover with the *Reduce* system. London and Musser [17] have experimented with the use of *Reduce* for program verification. More recently, Harisson and Théry [14] have combined *Maple* with the *HOL theorem prover*, and Ballarin, Homann and Calmet [2] have combined *Maple* and *Isabelle*. However, these systems do not appear to be able to handle very complicated proofs.

The paper is organized as follows: In section 2, we give two 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. section 4 contains an overview of the structure of Analytica and the major techniques that it uses in constructing proofs. sections 5 and 6 describe several of the most important techniques in greater detail. section 5 deals with summation and includes a short description of how we have integrated Gosper's algorithm into the prover. In section 6, we discuss how Analytica treats inequalities. In section 7, we state and prove Bernstein approximation theorem; the automated proof of the theorem is presented in the appendix. section 8 concludes with a discussion of improvements and extensions that we hope to add to Analytica in the future.

2 Simple Examples Proved by Analytica

In each example, the input for the prover is given first. The theorem and its proof are automatically generated by the theorem prover. Mathematica can generate LaTeX commands to typeset mathematical expressions.

2.1 The Sum of Two Roots of a Quadratic Equation

Note that in this example the input to the theorem prover contains division by a free variable a, which makes the right hand-side of the theorem undefined when $a = 0$. However, the division is "shielded" by a hypothesis $a \neq 0$. Analytica never introduces division unless it can prove that the divisor is non-zero. If the input already contains divisions, then all Analytica guarantees is that if the theorem makes sense, so will a proof. For a further discussion of this problem see section 8.

```
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 :

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

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

$$\mathit{True}$$

$\square$

2.2 Closed Form for a Summation

```
Prove[imp[and[integer[n], 0<=n, m!=1], sum[2^k/(1+m^(2^k)), {k, 0, n}] ==
                            1/(m-1) + 2^(n+1)/(1-m^(2^(n+1)))]];
```

Theorem :

$$(integer(n) \wedge 0 \leq n \wedge m \neq 1 \Rightarrow \sum_{k=0}^{n} \frac{2^k}{1+m^{2^k}} = \frac{1}{m-1} + \frac{2^{n+1}}{1-m^{2^{n+1}}})$$

Proof :

$$integer(n) \wedge 0 \leq n \wedge m \neq 1 \Longrightarrow \sum_{k=0}^{n} \frac{2^k}{1+m^{2^k}} = \frac{1}{-1+m} + \frac{2 \cdot 2^n}{1-m^{2 \cdot 2^n}}$$

reduces to

$$integer(n) \wedge 0 \leq n \Longrightarrow m = 1 \vee \sum_{k=0}^{n} \frac{2^k}{1+m^{2^k}} = \frac{1}{-1+m} + \frac{2 \cdot 2^n}{1-m^{2 \cdot 2^n}}$$

prove

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

use induction on n
base case with n = 0

$$m = 1 \vee \frac{1}{1+m} = \frac{1}{-1+m} + \frac{2}{1-m^2}$$

reduces to

$$\mathit{True}$$

induction step

$$integer(n) \wedge 0 \leq n \wedge \sum_{k=0}^{n} \frac{2^k}{1+m^{2^k}} = \frac{1}{-1+m} + \frac{2 \cdot 2^n}{1-m^{2 \cdot 2^n}} \Longrightarrow$$
$$m = 1 \vee \sum_{k=0}^{1+n} \frac{2^k}{1+m^{2^k}} = \frac{1}{-1+m} + \frac{4 \cdot 2^n}{1-m^{4 \cdot 2^n}}$$

calculate summations

$$integer(n) \wedge 0 \leq n \wedge \sum_{k=0}^{n} \frac{2^k}{1+m^{2^k}} = \frac{1}{-1+m} + \frac{2 \cdot 2^n}{1-m^{2 \cdot 2^n}} \Longrightarrow$$
$$m = 1 \vee \frac{2 \cdot 2^n}{1+m^{2 \cdot 2^n}} + \left(\sum_{k=0}^{n} \frac{2^k}{1+m^{2^k}}\right) = \frac{1}{-1+m} + \frac{4 \cdot 2^n}{1-m^{4 \cdot 2^n}}$$

substitute using equation

$$integer(n) \wedge 0 \leq n \wedge \sum_{k=0}^{n} \frac{2^k}{1+m^{2^k}} = \frac{1}{-1+m} + \frac{2 \cdot 2^n}{1-m^{2 \cdot 2^n}} \Longrightarrow$$
$$m = 1 \vee \frac{2 \cdot 2^n}{1+m^{2 \cdot 2^n}} + \frac{1}{-1+m} + \frac{2 \cdot 2^n}{1-m^{2 \cdot 2^n}} = \frac{1}{-1+m} + \frac{4 \cdot 2^n}{1-m^{4 \cdot 2^n}}$$

reduces to

$$\mathit{True}$$

□

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 after the rule has been 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 list of rules `R` to be applied to the expression `e` once; while "`e //. R`" causes the list of rules `R` to be applied to the expression `e` repeatedly until the expression fails to change. [22]

4 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 another equivalent form. If the rewriting phase is unsuccessful, the search terminates in failure; otherwise the simplification, inference and rewriting 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.

4.1 Skolemization phase

In Analytica (as in Bledsoe's *UT Prover* [4]), 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 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. We call $\neg$*skolemize*$(\neg f)$ the *negatively skolemized form* of f . A formula is valid if and only 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. For example, the skolemized form of the formula

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

is given by

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

while its negatively skolemized form is

$$P(x_0(), y) \to Q(u, v_0(u)).$$

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.

4.2 Simplification phase

Simplification is the key phase of Analytica. A formula is simplified with respect to its *proof context*. Intuitively, the proof context consists of the formulas that may be assumed true when the formula is encountered in the proof. The formula that results from simplifying f under context C is denoted by Simplify(f, C). In order for the simplification procedure to be sound, Simplify(f, C) must always satisfy the condition

$$C \vdash \text{Simplify}(f, C) \leftrightarrow f.$$

When a sequent $\Gamma \vdash \Delta$ enters the simplification phase, the disjunction Δ is simplified in the context Γ augmented by the *given* properties C_0 of the variables and constants. In other words, the simplification phase consists of computing Simplify$(\Delta, \Gamma \wedge C_0)$.

The simplification procedure for *composite formulas* is given by the following rules:

1. Simplify$(\neg f, C) = \neg$Simplify(f, C)
2. Simplify$(f_1 \wedge f_2, C) =$ Simplify$(f_1, C \wedge f_2) \wedge$ Simplify$(f_2, C \wedge f_1)$
3. Simplify$(f_1 \vee f_2, C) =$ Simplify$(f_1, C \wedge \neg f_2) \vee$ Simplify$(f_2, C \vee \neg f_1)$
4. Simplify$(f_1 \to f_2, C) =$ Simplify$(f_1, C \wedge \neg f_2) \to$ Simplify$(f_2, C \wedge f_1)$

The soundness of these rules can be easily established by structural induction. For example, if the soundness condition holds for f_1 and f_2, it will also hold for $f_1 \wedge f_2$, etc.

A large number of rules are provided for simplifying *atomic formulas*. They consist of simplification rules for inequalities, equations, summations and products, limits, and frequently used functions, such as `Abs`, `Min`, and `Max`. Simplification of inequalities is described in section 6, and simplifications of summations and products is discussed in section 5.

The following example illustrates how the context information is used to simplifying formulas:

Theorem :

$$(0 < a < b \Rightarrow b^3 - a^3 > (b - a)^3)$$

Proof :

$$0 < a < b \Longrightarrow -a^3 + b^3 > (-a + b)^3$$

`reduces to`

$$0 < a \wedge a - b < 0 \Longrightarrow 3a\,(a-b)\,b < 0$$

`reduces to`

$$\begin{aligned} 0 < a \wedge a - b < 0 \quad &\Longrightarrow \quad (0 < b \wedge -a + b < 0 \vee a - b < 0 \wedge b < 0) \wedge a < 0 \vee \\ & \quad\; 0 < a \wedge (0 < b \wedge a - b < 0 \vee -a + b < 0 \wedge b < 0) \end{aligned}$$

`simplify formula using context information`

$$0 < a \wedge a - b < 0 \Longrightarrow 0 < b$$

`replace expression with its lower or upper bounds`

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

`reduces to`

$$\mathit{True}$$

□

4.3 Inference phase

The inference phase is based on a *sequent calculus* [12]. 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. In order to accomplish this, f is decomposed into a set of *sequents* using rules of the sequent calculus. Each sequent has the form $\Gamma \vdash \Delta$, where Γ and Δ are initially sets of subformulas of f. The formula f will be proved, if substitution can be found that makes all of the sequents valid. A sequent $\Gamma \vdash \Delta$ is valid if it is impossible to make all of the elements of Γ true and all of the elements of Δ false.

In Analytica, the funcition $\mathit{FindSubstitution}(f)$ is used to determine the appropriate substitution for f. If f is not provable, $\mathit{FindSubstitution}(f)$ will return *Fail*. *FindSubstitution* has rules corresponding to each of the rules of the sequent calculus except those concerning quantifiers. The two rules for implication are given as examples:

1. Implification on the left:
   $\mathit{FindSubstitution}(\Gamma, A \rightarrow B, \Delta \vdash \Lambda) = \sigma_1\sigma_2$ where
   $\sigma_1 = \mathit{FindSubstitution}(\Gamma, \Delta \vdash A, \Lambda)$, and
   $\sigma_2 = \mathit{FindSubstitution}(\Gamma\sigma_1, B\sigma_1, \Delta\sigma_1 \vdash \Lambda\sigma_1)$.
2. Implication on the right:
   $\mathit{FindSubstitution}(\Gamma \vdash \Delta, A \rightarrow B, \Lambda) = \mathit{FindSubstitution}(\Gamma, A \vdash \Delta, B, \Lambda)$

Rules are also needed for atomic formulas. The three below are typical.

1. Equation: $\mathit{FindSubstitution}(\Gamma \vdash \Delta, a = b, \Lambda) = \sigma$ where $a\sigma = b\sigma$.
2. Inequality: $\mathit{FindSubstitution}(\Gamma, a < b, \Delta \vdash \Lambda) = \sigma$ where $a\sigma = b\sigma$.
3. Matching: $\mathit{FindSubstitution}(\Gamma, A, \Delta \vdash \Lambda, B, \Theta) = \sigma$ where $A\sigma = B\sigma$.

Backtracking is often necessary in the inference phase when there are multiple subgoals, because a substitution that makes one subgoal valid may not make another subgoal valid. When this happens it is necessary to find another substitution for the first subgoal. In order to restart the inference phase at the correct point, a stack is added to the procedure described above. When a rule is applied that may generate several subgoals, one subgoal is selected as the current goal and the others are saved on the stack. If some substitution σ makes the current subgoal valid, then σ is applied to the other subgoals on the stack and Analytica attempts to prove them. If the other subgoals are not valid under σ, then Analytica returns to the previous goal and tries to find another substitution that makes it valid.

Special tactics are included in the inference phase for handling inequalities and constructing inductive proofs. The inequalities tactics are described in section 6.

The induction tactic selects a suitable induction scheme for the formula to be proved, and attempts to establish the basis and induction steps. A typical induction scheme is

$$f(n_0) \land \forall n(n \geq n_0 \land f(n) \rightarrow f(n+1)) \Rightarrow \forall n(n \geq n_0 \rightarrow f(n))$$

The induction tactics identifies an induction variable n and determines a base value for it. In order to find a suitable induction variable for formula f, we list all variables that appear in f and select those that have type integer. To reduce the search space, we would like to make sure that our choice of the induction variable is a good one, i.e., the induction hypothesis should be useful for proving the induction conclusion. This will be more likely if the terms that appear in the induction conclusion also appear in the induction hypothesis or in the current context. Hence, we arrive at the following heuristic for selecting the induction variable: use n as the induction variable to prove $f(n)$ provided that after we simplify $f(n+1)$, it contains only terms that appear in $f(n)$, or in the current context, or are polynomials in n.

Once the induction variable n has been selected, a base value for that variable must be found in order to start the induction. In Analytica, a suitable base value may be determined by calculating the set of lower bounds of n as described in section 6 and choosing the simplest element of this set. If the basis case fails for this value, Analytica will choose another base value and try again until the basis is proven or no other choice is available. In the former case, the induction step is tried; otherwise the induction scheme fails and Analytica will try other techniques like those in the rewriting phase. This strategy is used in the constructing the induction proof for the second example in section 2.

4.4 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. For example,

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

   substitute using equation

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

2. Rewrite a trigonometric expression to an equivalent form.
   Given that a is an odd integer, k, m, n are integers, $m \leq n$,

$$-\cos(\pi a^n x) + (-1)^k \cos(\pi a^{-m+n}(a^m x - k)) = 0$$

   rewrite trigonometric expressions

True

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

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

`rewrite as`

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

4. Solve linear equations.

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

`solve linear equation`

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

5. Replace a user-defined function by its definition. In the example below the user-defined function S is expanded.

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

`expand definition`

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

5 Summation

Summations play an important role in symbolic computation. Nevertheless, Mathematica's ability to handle summations is very limited. A summation with range from n_1 to n_2, where n_1 and n_2 are integer constants and $n_1 \leq n_2$, is explicitly expanded into a sum with n_2-n_1+1 terms. However, a summation with a symbolic range will not be simplified. Consequently, we have introduced a number of rules for dealing with summations. Although most of these rules are based on simple identities, Analytica is able to handle a variety of summations. A few rules for summation are listed below. There are three groups of rules. For illustration, we show how some of them are coded in Mathematica. The function `FreeQ[`*a*`,` *b*`]` returns `True` if expression b does not appear in expression a.

1. Rules that reduce the number of summations occuring in an expression:

$$\sum_{n=n_1}^{n_2} c \longrightarrow c\cdot(n_2-n_1+1) \quad n \text{ does not occur in } c$$

```
sum[c_, {n_, n1_, n2_}] :> c (n2-n1+1)/; FreeQ[c,n]
```

$$a\cdot\sum_{n=n_1}^{n_2} f_1(n)+b\cdot\sum_{n=n_1}^{n_2} f_2(n)\longrightarrow\sum_{n=n_1}^{n_2}(af_1(n)+bf_2(n))$$

```
a_. sum[f1_, {n_, n1_, n2_}] + b_. sum[f2_, {n_, n1_, n2_}] :>
sum[a f1 + b f2, {n, n1, n2}]
```

$$a\cdot\sum_{n=n_1}^{n_2} f(n)+a\cdot\sum_{n=n_2+1}^{n_3} f(n)\longrightarrow a\cdot\sum_{n=n_1}^{n_3} f(n)$$

```
a_. sum[f_, {n_, n1_, n2_}] + a_. sum[f_, {n_, m_, n3_}] :>
a sum[f, {n, n1, n3}] /; simplify[m - n2 == 1]
```

$$a \cdot \sum_{n=n_1}^{n_3} f(n) - a \cdot \sum_{n=n_2}^{n_3} f(n) \longrightarrow a \cdot \sum_{n=n_2+1}^{n_3} f(n)$$

```
a_. sum[f_, {n_, n1_, n3_}] + b_. sum[f_, {n_, n2_, n3_}] :>
a sum[f, {n, n2+1, n3}] /; simplify[a + b == 0]
```

$$a \cdot \sum_{n=n_1}^{n_2} f(n) - a \cdot \sum_{n=n_3}^{n_2} f(n) \longrightarrow a \cdot \sum_{n=n_1}^{n_3-1}$$

```
a_. sum[f_, {n_, n1_, n2_}] + b_. sum[f_, {n_, n3_, n2_}] :>
a sum[f, {n, n1, n3-1}] /; simplify[a + b == 0]
```

2. Rules that simplify summands. The function `simpler` below is a heuristics that decides whether one expression is simpler than another.

$$\sum_{n=n_1}^{n_2} cf(n) = c \cdot \sum_{n=n_1}^{n_2} f(n) \quad \text{where } n \text{ does not occur in } c$$

```
sum[c_ f_, {n_, n1_, n2_}] :> c sum[f, {n, n1. n2}] /; FreeQ[a, v],
```

$$\sum_{n=n_1}^{n_2} f(n+1) = \sum_{n=n_1+1}^{n_2+1} f(n)$$

```
sum[f_, {n_, n1_, n2_}] :>
sum[f/.(n->n-1), {n, n1+1, n2+1}] /; simpler[f/.(n->n-1), f],
```

$$\sum_{n=n_1}^{n_2} f(n-1) = \sum_{n=n_1-1}^{n_2-1} f(n)$$

```
sum[f_, {n_, n1_, n2_}] :>
sum[f/.(n->n+1), {n, n1-1, n2-1}] /; simpler[f/.(n->n+1),f]
```

3. Rules that simplify summation bounds. Note that the lower bound of a summation can be bigger than the higher bound. We interpret these so that the equation $\sum_a^b + \sum_{b+1}^c = \sum_a^c$ always holds.

$$\sum_{n=n_1}^{n_2} f(n) = -\sum_{n=n_2+1}^{n_1-1} f(n) \quad \text{if } n_1 > n_2$$

```
sum[f_, {n_, n1_, n2_}] :> -sum[f, {n, n2+1, n1-1}] /; simplify[n1 > n2]
```

$$\sum_{n=n_1}^{n_2+N} f(n) = \left(\sum_{n=n_1}^{n_2} f(n)\right) + f(n_2+1) + \ldots + f(n_2+N)$$

$$\sum_{n=n_1}^{n_2-N} f(n) = \left(\sum_{n=n_1}^{n_2} f(n)\right) - f(n_2) - \ldots - f(n_2-N+1)$$

```
sum[f_, {n_, n1_, n2_ + n_Integer}] :>
sum[f, {n, n1, n2}] + sum[f, {n, n2 + 1, n2 + n}],
```

5.1 A summation example

The following example comes from a lemma used in the proof of the existence of a continuous, nowhere differentiable function given by Weierstrass [9, 21].

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

`simplify summations`

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

`simplify summations`

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

`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`

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

This can be simplified to `True` by trigonometric rules.

5.2 Gosper's Algorithm

We integrated *Gosper's algorithm* [16] for finding closed forms of indefinite hypergeometric summations into the theorem prover. A function $g(n)$ is *hypergeometric* if $g(n+1)/g(n)$ is a rational function of n. Gosper's algorithm computes a hypergeometric closed form of an indefinite summation $\sum g(n)$, where $g(n)$ is hypergeometric, if such a closed form exists. This covers a relatively large class of summations that appear in practice.

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`

$$\mathit{True}$$

□

6 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.

6.1 Simplification of inequalities

There are many rules that simplify atomic formulas involving inequalities. Here are four examples:

1. Simplify$(0 \leq a^n, C) = \mathit{True}$ if Simplify$(0 \leq a, C) = \mathit{True}$
2. Simplify$(0 < a^n, C) = \mathit{True}$ if Simplify$(0 < a, C) = \mathit{True}$
3. Simplify$(a^n \leq 0, C) = \mathit{False}$ if Simplify$(0 < a, C) = \mathit{True}$
4. Simplify$(a^n < 0, C) = \mathit{False}$ if Simplify$(0 < a, C) = \mathit{True}$

There are also rules that use upper and lower bound information to simplify inequalities. 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 6.3. The set of lower(upper) bounds is calculated in the current context.

1. Simplify$(f_1 \leq f_2, C) = \mathit{False}$ if $\exists x[x \in \mathit{Lower}(f_1 - f_2, C) \wedge x > 0]$.
2. Simplify$(f_1 \leq f_2, C) = \mathit{True}$ if $\exists x[x \in \mathit{Lower}(f_2 - f_1, C) \wedge x \geq 0]$.
3. Simplify$(f_1 < f_2, C) = \mathit{True}$ if $\exists x[x \in \mathit{Lower}(f_2 - f_1, C) \wedge x > 0]$.
4. Simplify$(f_1 < f_2, C) = \mathit{False}$ if $\exists x[x \in \mathit{Lower}(f_1 - f_2, C) \wedge x \geq 0]$.

6.2 Proof strategy for Inequalities

Although many inequality formulas can be handled in the simplification phase, some valid inequality formulas cannot be reduced to true in this phase. For example, $(a \leq 0 \wedge b \leq a) \rightarrow b \leq 0$ cannot 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. If the inequality $a \leq b$ is not directly provable using the techniques in the simplification phase, then

Analytica will try to find a term c, such that $a \leq c$ and $c \leq b$ are both provable in the current context. In order to find such a term c, we compute a set of upper bounds for a and a set of lower bounds for b by using information provided by the current context. The sets computed are denoted by $Upper(a)$ and $Lower(b)$, respectively. A term x will be in $Upper(a)$ only if $a \leq x$ is true in the current context. Likewise, x will be in $Lower(b)$ only if $x \leq b$ is true in the current context. To prove $a \leq b$, Analytica automatically computes the sets $Lower(b)$ and $Upper(a)$. Then it attempts 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 deal with strict inequalities, we introduce a new symbol S such that both $S_L(a) \leq b$ and $a \leq S_U(b)$ are equivalent to $a < b$. Hence, $S_U(x) \in Upper(a)$ only if $a < x$ is true in the current context, and $S_L(x) \in Lower(a)$ only if $x < a$ is true in the current context. $S_U(a) + b = S_U(a + b)$ because $c \leq S_U(a + b)$ iff $c < a + b$ iff $c - b < a$ iff $c - b \leq S_U(a)$ iff $c \leq S_U(a) + b$. Similarly, $S_L(a) + b = S_L(a + b)$, $-S_L(a) = S_U(-a)$ and $-S_U(a) = S_L(-a)$, etc.

When Analytica attempts to prove $a < b$, it first computes $Upper(a)$ and $Lower(b)$. All terms in $Upper(a)$ and $Lower(b)$ are linear combinations of S_U and S_L terms. The above rules are used to rewrite the terms in $Upper(a)$. Each of the resulting terms either has the form $S_U(c)$ or has the form c where c does not contain S_U or S_L. If the term has the form $S_U(c)$, it is sufficient to prove $c \leq b$. Otherwise it is sufficient to prove $c < b$. Similarly, terms in $Lower(b)$ can be rewritten into either the form $S_L(c)$ or the form c where c does not contain S_U or S_L. If the term is rewritten into $S_L(c)$, it is sufficient to prove $a \leq c$. Otherwise it is sufficient to prove $a < c$. This convention permits both strict inequalities and nonstrict inequalities to be handled by the same method.

The technique is complete for linear inequalities, and it can also prove many of the nonlinear inequalities that arise in practice. The technique is not complete for nonlinear inequalities, however.

6.3 Calculating upper and lower bounds for expressions

There are three main ways of obtaining upper and lower bounds for expressions.

1. Obtain bounds from context information:
   Upper and lower bounds for an expression are calculated in the current context. For example, when proving $(a \leq b) \vee c$, the upper bounds of a and the lower bounds of b are calculated under the context of $\neg c$. In general, If $a \leq b$ is a conjunct of the current context, we have
   $$a \in Lower(b), \quad b \in Upper(a),$$
   and if $a < b$ is a conjunct of the current context, we have
   $$S_L(a) \in Lower(b), \quad S_U(b) \in Upper(a).$$

2. Obtain bounds 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:
   $$\{cx | x \in Upper(a)\} \subseteq Lower(ca), \text{ if } \ c \leq 0$$

3. Use 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:
   $$x + \frac{1}{2} \in Upper(round(x))$$
   $$x - \frac{1}{2} \in Lower(round(x))$$

6.4 An example to illustrate inequality proofs

The following example also comes from the proof of the Weierstrass theorem mentioned earlier. Assume 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`

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

The last inequality will be reduced to True in the rewriting phase by using the tactic for trigonometric identities.

7 The Bernstein Approximation Theorem

In this section we show how a non-trivial theorem from real analysis can be proved with Analytica. The proof needs to be broken down into several lemmas. First we state the theorem and provide a manual proof. Then we look at how Analytica proves pieces of the same theorem.

Theorem: Let $I = [0, 1]$ be the closed unit interval and f a real continuous function on I. Define the n-th Bernstein polynomial for f as

$$B_n(x, f) = \sum_{k=0}^{n} \binom{n}{k} f(k/n)\, x^k\, (1-x)^{n-k}.$$

On the interval I, the sequence of Bernstein polynomials for f converges uniformly to f.

Proof: It follows from the binomial theorem that

$$f(x) = \sum_{k=0}^{n} \binom{n}{k} f(x)\, x^k (1-x)^{n-k},$$

hence

$$|f(x) - B_n(x, f)| \leq \sum_{k=0}^{n} \binom{n}{k} |f(x) - f(k/n)|\, x^k (1-x)^{n-k}. \tag{1}$$

Since f is continuous on a closed interval, it is uniformly continuous and bounded. Let $M = \sup\{|f(x)|\ ;\ x \in I\}$.

For every $\epsilon > 0$, there is $\delta(\epsilon) > 0$ such that $|x - y| < \delta(\epsilon)$ implies $|f(x) - f(y)| < \epsilon$ for all $x, y \in I$.

Let $\epsilon > 0$ be arbitrary. Choose n larger than $\delta(\epsilon/2)^{-4}$ and $(M/\epsilon)^2$. We show that the sum on the right side of (1) is less than ϵ. In order to do this, split the sum into two parts:

- *Near* is the sum of those terms in (1) for which $|x - k/n| < n^{-1/4}$, i.e.

$$Near = \sum_{\substack{0 \le k \le n \\ |x-k/n|<n^{-1/4}}} \binom{n}{k} |f(x) - f(k/n)|\, x^k (1-x)^{n-k}$$

- *Far* is the sum of those terms in (1) for which $|x - k/n| \ge n^{-1/4}$, i.e.

$$Far = \sum_{\substack{0 \le k \le n \\ |x-k/n|\ge n^{-1/4}}} \binom{n}{k} |f(x) - f(k/n)|\, x^k (1-x)^{n-k}$$

We bound *Near* and *Far* above by $\epsilon/2$. Then the result follows.

For showing that $Near \le \epsilon/2$, note that since the sum involves those terms for which $|x - k/n| < n^{-1/4} \le \delta(\epsilon/2)$, we have $|f(x) - f(k/n)| \le \epsilon/2$, and hence

$$Near \le \frac{\epsilon}{2} \sum_{\substack{0 \le k \le n \\ |x-k/n|<n^{-1/4}}} \binom{n}{k} x^k (1-x)^{n-k} \le \frac{\epsilon}{2} \sum_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k} = \frac{\epsilon}{2}$$

To show that $Far \le \epsilon/2$, bound $|f(x) - f(y)| \le 2M$ and note that in *Far* we have $\sqrt{n}\,(x - k/n)^2 \ge 1$. Hence, in *Far*, the inequality

$$|f(x) - f(y)| \le 2M\sqrt{n}(x - k/n)^2$$

is valid. We also need the identity

$$\sum_{k=0}^{n} \binom{n}{k} \left(x - \frac{k}{n}\right)^2 x^k (1-x)^{n-k} = \frac{x(1-x)}{n}. \qquad (2)$$

Now we can bound *Far*:

$$\begin{aligned} Far &\le 2M\sqrt{n} \sum_{\substack{0 \le k \le n \\ |x-k/n|<n^{-1/4}}} (x - \frac{k}{n})^2 \binom{n}{k} x^k (1-x)^{n-k} \\ &\le 2M\sqrt{n} \sum_{k=0}^{n} (x - \frac{k}{n})^2 \binom{n}{k} x^k (1-x)^{n-k} \\ &= 2M\sqrt{n}\, \frac{x(1-x)}{n} \le \frac{M}{2\sqrt{n}} \le \epsilon/2 \end{aligned}$$

This completes the proof.
□

We help Analytica by specifying the following facts as "given":

$$\begin{aligned} 0 &< \epsilon \\ 0 &< m \\ 0 < x &\wedge x < 1 \\ m &\le \delta(\epsilon/2) \\ m^2 &\le \frac{\epsilon}{M} \end{aligned}$$

Here m is the quantity $n^{-1/4}$ from the above proof. We only considered the cases that $x \neq 0$ and $x \neq 1$ in the proof. Since $B(n,0) \equiv f(0)$ and $B(n,1) \equiv f(1)$, the other two cases are trivial.

We also provide the definitions of $B(n,x)$, the sums *Near* and *Far*, which are called $near(x,n)$ and $far(x,n)$, and the intermediate sum $far1(x,n)$ that appears in the proof of $Far \leq \frac{\epsilon}{2}$. The proof is broken down into five lemmas:

1. $near(x,n) \leq \epsilon/2$
2. $far1(x,n) \leq \epsilon/2$
3. $far(x,n) \leq far1(x,n)$
4. $|f(x) - B(n,x)| \leq far(x,n) + near(x,n)$
5. $|f(x) - B(n,x)| \leq \epsilon$

It is clear that Analytica has no magical powers. All of the really difficult guesswork, namely the definition of far and $near$ and the connection between ϵ and δ, has to be specified by a human. Still, this is not an easy theorem and the amount of work that can be done by Analytica automatically is impressive.

In the appendix we provide the complete input and machine generated output for the proof of the Bernstein's theorem. Below we follow Analytica's proof of the first lemma, commenting on what Analytica is doing.

Theorem :

$$near(x,n) \leq \frac{\epsilon}{2}$$

Proof :

(Skolemization does not change the sequent since it is quantifier free.)

$$near(x,n) \leq \frac{\epsilon}{2}$$

(Simplification causes all terms in the inequality to be moved to the left hand side.)

$$\frac{-\epsilon}{2} + near(x,n) \leq 0$$

(Rewriting finds the common denominator 2 *for the sum.)*

$$\frac{-\epsilon + 2\,near(x,n)}{2} \leq 0$$

(Simplification removes the common denominator 2*.)*

$$-\epsilon + 2\,near(x,n) \leq 0$$

(Rewriting expands the definition of the function near*.)*

$$-\epsilon + 2\,(1-x)^n \sum_{0 \leq k \leq n}^{\left|\frac{k}{n} - x\right| < m} \frac{x^k \left| -f(\frac{k}{n}) + f(x) \right| \binom{n}{k}}{(1-x)^k} \leq 0$$

(Simplification moves all terms in the summation condition to left hand side.)

$$-\epsilon + 2\,(1-x)^n \sum_{0 \leq k \leq n}^{-m + \left|\frac{k}{n} - x\right| < 0} \frac{x^k \left| f(\frac{k}{n}) - f(x) \right| \binom{n}{k}}{(1-x)^k} \leq 0$$

(Inequality reasoning in the inference phase determines that $\left|\frac{k}{n} - x\right| < m < \delta_f(\frac{\epsilon}{2})$. *Since* f *is a uniformly continuous function, it follows that* $\left|f(\frac{k}{n}) - f(x)\right| \leq \frac{\epsilon}{2}$.*)*

$$\epsilon \left(-1 + (1-x)^n \sum_{0 \leq k \leq n}^{-m+\left|\frac{k}{n}-x\right|<0} \frac{x^k \binom{n}{k}}{(1-x)^k} \right) \leq 0$$

(Simplification removes ϵ *from the inequality since* ϵ *is greater than* 0.*)*

$$-1 + (1-x)^n \sum_{0 \leq k \leq n}^{-m+\left|\frac{k}{n}-x\right|<0} \frac{x^k \binom{n}{k}}{(1-x)^k} \leq 0$$

(Since each term of the summation is non-negative, adding more terms makes the summation bigger. The inequality reasoning tactic in the inference phase is used in this step.)

$$-1 + (1-x)^n \left(\sum_{k=0}^{n} \frac{x^k \binom{n}{k}}{(1-x)^k} \right) \leq 0$$

(The summation tactic in the rewriting phase finds a closed form for the summation.)

$$-1 + \frac{(-1)^n (1-x)^n}{(-1+x)^n} \leq 0$$

(Simplification reduces the left hand side to 0.*)*

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[3] 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. One direction is to add powerful algorithmic techniques for simplifying particular classes of formulas (like extensions of Gosper's algorithm for summations). The problem with adding such techniques is that a proof obtained in this manner may be difficult 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 [7, 8], but more work should be done in the context of inductive proofs in analysis.

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 [18]. His work, however, uses a theorem prover based on hyper-resolution and may not produce proofs that are readable.

Better methods for managing hypotheses and previously proved lemmas and theorems are also needed. Techniques developed for proof checking systems like LCF [15] and HOL [13] may be adequate in the short run. In general, deciding when to use a hypothesis or previous result is a very difficult problem. Every student of elementary calculus learns the mean value theorem by heart, but it is not easy to give a good set of rules for determining when to apply this theorem in order to obtain a simpler bound on some complicated expression. For some work in this area see, e.g., Baker-Plummer [1].

Perhaps, the most serious problem in building a theorem prover like Analytica is the soundness of the underlying symbolic computation system. Mathematica, as well as Macsyma, Reduce, Maple, etc., has rules that are not always correct, and some of them are simply wrong. There is no easy way for a theorem prover to check correctnes of everything that the symbolic computation system is doing, because some steps are very involved. We did not address this issue in a satisfactory manner in Analytica, and it is not clear how it could be done on an existing symbolic computation system. We believe the solution to the soundness problem is to develop a theorem prover and a symbolic computation system together so that each simplification step can be rigorously justified.

Appendices

A Input To Analytica

```
(* Bernstein Approximation Theorem *)

integer[n] = True;
integer[k] = True;

sum1[f_] := sum[f,  {k,0,n,Abs[k/n-x]<n^(-1/4)}];
sum2[f_] := sum[f,  {k,0,n,n^(-1/4)<=Abs[k/n-x]}];

(* the bernstein polynomial for "f" *)
AddDefinition[B[n_,x_] ==
   sum[f[k/n] Binomial[n, k] x^k (1-x)^(n-k), {k, 0, n}]];

(* Several auxiliary functions *)
AddDefinition[near[x_, n_] ==
  sum1[Abs[f[x]-f[k/n]] Binomial[n,k] x^k (1-x)^(n-k)]];
AddDefinition[far[x_, n_] ==
  sum2[Abs[f[x]-f[k/n]] Binomial[n,k] x^k (1-x)^(n-k)]];
AddDefinition[far1[x_, n_] ==
  2 M Sqrt[n] sum2[(x-k/n)^2 Binomial[n,k] x^k (1-x)^(n-k)]];

(* f is a continuous function on [0, 1] and therefore bounded *)
ContinuousFunction[f] := True;
Domain[f] := ClosedInterval[0, 1];
M=Bound[f];

(* n >= max(delta[f][epsilon/2]^(-4), (M/epsilon)^2) *)
n = m^(-4);
Given[m<= delta[f][epsilon/2]];
Given[m^2 <= epsilon/M];

Given[m>0, epsilon>0, x>0, x<1];

(* Lemmas *)
ProveAndSave[near[x, n]<=epsilon/2];
ProveAndSave[far1[x, n]<=epsilon/2];
ProveAndSave[far[x, n]<=far1[x, n]];
ProveAndSave[Abs[f[x]-B[n,x]]<=far[x, n]+near[x, n]];

(* Theorem *)
Prove[Abs[f[x]-B[n,x]] <= epsilon];
```

B A Proof of the Bernstein Approximation Theorem

This proof was machine generated from the input above. We only inserted a couple of line breaks because some of the lines were too long to fit on a page.

Definitions used in proof:

$$B(n,x) = (1-x)^n \left(\sum_{k=0}^{n} \frac{x^k \binom{n}{k} f(\frac{k}{n})}{(1-x)^k} \right)$$

$$near(x,n) = (1-x)^n \sum_{0\le k\le n}^{\left|\frac{k}{n}-x\right|<m} \frac{x^k \left|-f(\frac{k}{n}) + f(x)\right| \binom{n}{k}}{(1-x)^k}$$

$$far(x,n) = (1-x)^n \sum_{0\le k\le n}^{m\le\left|\frac{k}{n}-x\right|} \frac{x^k \left|-f(\frac{k}{n}) + f(x)\right| \binom{n}{k}}{(1-x)^k}$$

$$far1(x,n) = \frac{2\,(1-x)^n\, M \sum_{0\le k\le n}^{m\le\left|\frac{k}{n}-x\right|} \frac{x^k \left(-(\frac{k}{n})+x\right)^2 \binom{n}{k}}{(1-x)^k}}{m^2}$$

Theorems proved :

$$near(x,n) \leq \frac{\epsilon}{2}$$
$$far1(x,n) \leq \frac{\epsilon}{2}$$
$$far(x,n) \leq far1(x,n)$$
$$|f(x) - B(n,x)| \leq far(x,n) + near(x,n)$$
$$|f(x) - B(n,x)| \leq \epsilon$$

Theorem :

$$near(x,n) \leq \frac{\epsilon}{2}$$

Proof :

$$near(x,n) \leq \frac{\epsilon}{2}$$

reduces to

$$\frac{-\epsilon}{2} + near(x,n) \leq 0$$

rewrite as

$$\frac{-\epsilon + 2\, near(x,n)}{2} \leq 0$$

reduces to

$$-\epsilon + 2\, near(x,n) \leq 0$$

open definition

$$-\epsilon + 2\,(1-x)^n \sum_{0\leq k\leq n}^{\left|\frac{k}{n}-x\right|<m} \frac{x^k \left|-f(\frac{k}{n}) + f(x)\right| \binom{n}{k}}{(1-x)^k} \leq 0$$

reduces to

$$-\epsilon + 2\,(1-x)^n \sum_{0\leq k\leq n}^{-m+\left|\frac{k}{n}-x\right|<0} \frac{x^k \left|f(\frac{k}{n}) - f(x)\right| \binom{n}{k}}{(1-x)^k} \leq 0$$

replace expression with its lower or upper bounds

$$\epsilon \left(-1 + (1-x)^n \sum_{0\leq k\leq n}^{-m+\left|\frac{k}{n}-x\right|<0} \frac{x^k \binom{n}{k}}{(1-x)^k}\right) \leq 0$$

reduces to

$$-1 + (1-x)^n \sum_{0\leq k\leq n}^{-m+\left|\frac{k}{n}-x\right|<0} \frac{x^k \binom{n}{k}}{(1-x)^k} \leq 0$$

replace expression with its lower or upper bounds

$$-1 + (1-x)^n \left(\sum_{k=0}^{n} \frac{x^k \binom{n}{k}}{(1-x)^k}\right) \leq 0$$

calculate summations

$$-1 + \frac{(-1)^n\,(1-x)^n}{(-1+x)^n} \le 0$$

reduces to

$$\mathit{True}$$

□

Theorem :

$$\mathit{far1}\,(x,n) \le \frac{\epsilon}{2}$$

Proof :

$$\mathit{far1}\,(x,n) \le \frac{\epsilon}{2}$$

reduces to

$$\frac{-\epsilon}{2} + \mathit{far1}\,(x,n) \le 0$$

rewrite as

$$\frac{-\epsilon + 2\,\mathit{far1}\,(x,n)}{2} \le 0$$

reduces to

$$-\epsilon + 2\,\mathit{far1}\,(x,n) \le 0$$

open definition

$$-\epsilon + \frac{4\,(1-x)^n\,M\,\sum_{0\le k\le n}^{m\le\left|\frac{k}{n}-x\right|} \frac{x^k\,\left(-\left(\frac{k}{n}\right)+x\right)^2\,\binom{n}{k}}{(1-x)^k}}{m^2} \le 0$$

reduces to

$$-\epsilon + \frac{4\,(1-x)^n\,M\,\sum_{0\le k\le n}^{m-\left|\frac{k}{n}-x\right|\le 0} \frac{\left(\frac{k}{n}-x\right)^2\,x^k\,\binom{n}{k}}{(1-x)^k}}{m^2} \le 0$$

replace expression with its lower or upper bounds

$$-\epsilon + \frac{4\,(1-x)^n\,M\,\left(\sum_{k=0}^{n} \frac{\left(\frac{k}{n}-x\right)^2\,x^k\,\binom{n}{k}}{(1-x)^k}\right)}{m^2} \le 0$$

calculate summations

$$-\epsilon + 4\,m^2\,(1-x)\,x\,M \le 0$$

replace expression with its lower or upper bounds

$$-\left(\epsilon\,(-1+2\,x)^2\right) \le 0$$

reduces to

$$\mathit{True}$$

□

Theorem :

$$\mathit{far}(x,n) \le \mathit{far1}(x,n)$$

Proof :

$$\mathit{far}(x,n) \le \mathit{far1}(x,n)$$

reduces to

$$\mathit{far}(x,n) - \mathit{far1}(x,n) \le 0$$

open definition

$$\frac{-2\,(1-x)^n\,M\,\sum_{0\le k\le n}^{m\le\left|\frac{k}{n}-x\right|}\frac{x^k\,\left(-(\frac{k}{n})+x\right)^2\,\binom{n}{k}}{(1-x)^k}}{m^2} + (1-x)^n\sum_{0\le k\le n}^{m\le\left|\frac{k}{n}-x\right|}\frac{x^k\,\left|-f(\frac{k}{n})+f(x)\right|\,\binom{n}{k}}{(1-x)^k} \le 0$$

reduces to

$$\frac{-2\,M\,\sum_{0\le k\le n}^{m-\left|\frac{k}{n}-x\right|\le 0}\frac{\left(\frac{k}{n}-x\right)^2\,x^k\,\binom{n}{k}}{(1-x)^k}}{m^2} + \sum_{0\le k\le n}^{m-\left|\frac{k}{n}-x\right|\le 0}\frac{x^k\,\left|f(\frac{k}{n})-f(x)\right|\,\binom{n}{k}}{(1-x)^k} \le 0$$

simplify summations

$$\sum_{0\le k\le n}^{m-\left|\frac{k}{n}-x\right|\le 0}\left(\frac{x^k\,\left|f(\frac{k}{n})-f(x)\right|\,\binom{n}{k}}{(1-x)^k} - \frac{2\left(\frac{k}{n}-x\right)^2\,x^k\,\binom{n}{k}\,M}{m^2\,(1-x)^k}\right) \le 0$$

reduces to

$$\sum_{0\le k\le n}^{m-\left|\frac{k}{n}-x\right|\le 0}\frac{x^k\,\binom{n}{k}\,\left(\left|f(\frac{k}{n})-f(x)\right| - \frac{2\left(\frac{k}{n}-x\right)^2 M}{m^2}\right)}{(1-x)^k} \le 0$$

matching lemma

$$\forall k\forall \mathit{low}\forall \mathit{up}\forall f\forall \mathit{cond}[$$

$$((-k+\mathit{low}\le 0 \wedge k-\mathit{up}\le 0 \wedge \mathit{cond} \Rightarrow f\le 0) \Rightarrow \sum_{\mathit{low}\le k\le \mathit{up}}^{\mathit{cond}} f \le 0)]$$

back chaining

$$-k\le 0 \wedge k-n\le 0 \wedge m-\left|\frac{k}{n}-x\right|\le 0 \Longrightarrow \frac{x^k\,\binom{n}{k}\,\left(\left|f(\frac{k}{n})-f(x)\right| - \frac{2\left(\frac{k}{n}-x\right)^2 M}{m^2}\right)}{(1-x)^k} \le 0$$

reduces to

$$0\le k \wedge k-n\le 0 \wedge m-\left|\frac{k}{n}-x\right|\le 0 \Longrightarrow$$

$$-\left|f(\frac{k}{n})-f(x)\right| + \frac{2\left(\frac{k}{n}-x\right)^2 M}{m^2} \le 0 \wedge (k<0 \vee -k+n<0) \vee$$

$$\left|f(\frac{k}{n})-f(x)\right| - \frac{2\left(\frac{k}{n}-x\right)^2 M}{m^2} \le 0$$

simplify formula using local context

$$0 \le k \wedge k - n \le 0 \wedge m - \left|\frac{k}{n} - x\right| \le 0 \Longrightarrow \left|f(\frac{k}{n}) - f(x)\right| - \frac{2\left(\frac{k}{n} - x\right)^2 M}{m^2} \le 0$$

replace expression with its lower or upper bounds

$$0 \le k \wedge k - n \le 0 \wedge m - \left|\frac{k}{n} - x\right| \le 0 \Longrightarrow \left|f(\frac{k}{n}) - f(x)\right| - 2\,M \le 0$$

replace expression with its lower or upper bounds

$$0 \le k \wedge k - n \le 0 \wedge m - \left|\frac{k}{n} - x\right| \le 0 \Longrightarrow \left|f(\frac{k}{n})\right| + |f(x)| - 2\,M \le 0$$

replace expression with its lower or upper bounds

$$0 \le k \wedge k - n \le 0 \wedge m - \left|\frac{k}{n} - x\right| \le 0 \Longrightarrow \left|f(\frac{k}{n})\right| - M \le 0$$

reduces to

$$\mathit{True}$$

□

Theorem :

$$|f(x) - B(n, x)| \le \mathit{far}(x, n) + \mathit{near}(x, n)$$

Proof :

$$|-B(n, x) + f(x)| \le \mathit{far}(x, n) + \mathit{near}(x, n)$$

reduces to

$$|B(n, x) - f(x)| - \mathit{far}(x, n) - \mathit{near}(x, n) \le 0$$

open definition

$$\left|-f(x) + (1-x)^n \left(\sum_{k=0}^{n} \frac{x^k \binom{n}{k} f(\frac{k}{n})}{(1-x)^k}\right)\right| - \left((1-x)^n \sum_{0\le k\le n}^{\left|\frac{k}{n}-x\right|<m} \frac{x^k \left|-f(\frac{k}{n}) + f(x)\right| \binom{n}{k}}{(1-x)^k}\right)$$
$$- \left((1-x)^n \sum_{0\le k\le n}^{m\le\left|\frac{k}{n}-x\right|} \frac{x^k \left|-f(\frac{k}{n}) + f(x)\right| \binom{n}{k}}{(1-x)^k}\right) \le 0$$

reduces to

$$\left|f(x) - (1-x)^n \left(\sum_{k=0}^{n} \frac{x^k \binom{n}{k} f(\frac{k}{n})}{(1-x)^k}\right)\right| -$$
$$(1-x)^n \left(\sum_{0\le k\le n}^{-m+\left|\frac{k}{n}-x\right|<0} \frac{x^k \left|f(\frac{k}{n}) - f(x)\right| \binom{n}{k}}{(1-x)^k} + \sum_{0\le k\le n}^{m-\left|\frac{k}{n}-x\right|\le 0} \frac{x^k \left|f(\frac{k}{n}) - f(x)\right| \binom{n}{k}}{(1-x)^k}\right) \le 0$$

simplify summations

$$\left|f(x) - (1-x)^n \left(\sum_{k=0}^{n} \frac{x^k \binom{n}{k} f(\frac{k}{n})}{(1-x)^k}\right)\right| - \left((1-x)^n \left(\sum_{k=0}^{n} \frac{x^k \left|f(\frac{k}{n}) - f(x)\right| \binom{n}{k}}{(1-x)^k}\right)\right) \leq 0$$

replace expression with its lower or upper bounds

$$\left|f(x) - (1-x)^n \left(\sum_{k=0}^{n} \frac{x^k \binom{n}{k} f(\frac{k}{n})}{(1-x)^k}\right)\right| - \left((1-x)^n \left|\sum_{k=0}^{n} \frac{x^k \binom{n}{k} \left(f(\frac{k}{n}) - f(x)\right)}{(1-x)^k}\right|\right) \leq 0$$

calculate summations

$$-(1-x)^n \left|-\frac{(-1)^n f(x)}{(-1+x)^n} + \sum_{k=0}^{n} \frac{x^k \binom{n}{k} f(\frac{k}{n})}{(1-x)^k}\right| + \left|f(x) - (1-x)^n \left(\sum_{k=0}^{n} \frac{x^k \binom{n}{k} f(\frac{k}{n})}{(1-x)^k}\right)\right| \leq 0$$

reduces to

$$\textit{True}$$

□

References

[1] D. Baker-Plummer. *Gazing: An Approach to the Problem of Definition and Lemma Use. Journal of Automated Reasoning*, 8:311-344, 1992.

[2] C. Ballarin, K. Homann, J. Calmet. *Theorems and Algorithms: An Interface between Isabelle and Maple.* In *Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation* July, 1995, Montreal, Canada

[3] B. C. Berndt. *Ramanujan's Notebooks, Part I.* Springer-Verlag, 1985.

[4] W. W. Bledsoe. The UT natural deduction prover. Technical Report ATP-17B, Mathematical Dept., University of Texas at Austin, 1983.

[5] W. W. Bledsoe. Some automatic proofs in analysis. In *Automated Theorem Proving: After 25 Years.* American Mathematical Society, 1984.

[6] W. W. Bledsoe, P. Bruell, and R. Shostak. A prover for general inequalities. Technical Report ATP-40A, Mathematical Dept., University of Texas at Austin, 1979.

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

[8] A. Bundy, F. van Harmelen, J. Hesketh, and A. Smaill. Experiments with proof plans for induction. Technical report, Department of Artificial Intelligence, University of Edinburgh, 1988.

[9] E. M. Clarke and X. Zhao. Analytica: A theorem prover for Mathematica. *The Journal of Mathematica*, 3(1), 1993.

[10] W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: An interactive mathematical proof system. Technical report, The MITRE Corporation, 1990.

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

[12] J. H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving.* Harper & Row, 1986.

[13] M. Gordon. HOL: A machine oriented formulation of higher order logic. Technical report, Computer Laboratory, University of Cambridge, 1985.

[14] J. Harrison, L. Théry. *Extending HOL Theorem Prover with a Computer Algebra System to Reason About the Reals.* In *Proceedings of Higher Order Logic Theorem Proving and its Applications.* Editors: J.J. Joyce, C. Seger Lecture Notes in Computer Science 780, 1994.

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

[16] R. W. Gosper. Indefinite hypergeometric sums in MACSYMA. In *Proceedings of the MACSYMA Users Conference*, pages 237–252, 1977.

[17] R. L. London and D. R. Musser. The application of a symbolic mathematical system to program verification. Technical report, Information Science Institute, University of Southern California, 1975.

[18] A. Quaife. Automated deduction in von Neumann-Bernays-Godel set theory. Technical report, Dept. of Mathematics, Univ. of California at Berkeley, 1989.

[19] E. Sacks. Hierarchical inequality reasoning. Technical report, MIT Laboratory for Computer Science, 1987.

[20] P. Suppes and S. Takahashi. An interactive calculus theorem-prover for continuity properties. *Journal of Symbolic Computation*, 7:573–590, 1989.

[21] E. C. Titchmarsh. *The Theory of Functions.* Oxford University Press, 1932.

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