Probabilistic Equivalence Verification Approach for Automatic Mathematical Solution Assessment

Minh Luan Nguyen **Siu Cheung Hui**
School of Computer Engineering
Nanyang Technological University
Singapore

Alvis C.M. Fong
School of Computing and Math Sciences
Auckland University of Technology
New Zealand

Abstract

Automatic mathematical solution assessment checks the equivalence of mathematical expressions in the user answer and standard solution. It is a challenging problem as the semantics of mathematical expressions are highly symbolic and equivalent mathematical expressions can be expressed in different forms. In this paper, we propose an effective Probabilistic Equivalence Verification (PEV) approach for automatic mathematical solution assessment. The proposed PEV approach is a randomized method based on the probabilistic numerical equivalence testing of two mathematical expressions. It can avoid false negative errors completely while guaranteeing a small probability of false positive errors to occur. The performance results have shown that the proposed PEV approach has outperformed other popular techniques in Computer Algebra Systems such as Maple and Mathematica.

1 Introduction

In an e-learning environment, it is important to assess the knowledge proficiency of learners in order to provide relevant feedback and suggestion to enhance their learning process [Guzman and Conejo, 2007]. Most of the current e-learning systems such as Coursera [2013] and Edventure [2013] have provided support for automatic assessment, but only with multiple choice questions as it is a great challenge to support other question formats. In addition, it also takes lots of effort to convert an open-ended math question into a multiple choice question [Hyman, 2012]. Recently, there has been a growing interest in intelligent tutoring systems for supporting mathematical learning such as geometry construction [Gulwani *et al.*, 2011], algebra problem generation [Singh *et al.*, 2012] and automatic solution assessment [Rasila *et al.*, 2007; 2010]. For mathematical solution assessment, it checks the equivalence of mathematical expressions in the user answer and standard solution. It is a challenging problem as the semantics of mathematical expressions are highly symbolic and equivalent mathematical expressions can be expressed in different forms.

Automatic mathematical solution assessment is different from other similarity-based solution assessment problems such as automatic essay grading [Hearst, 2000] and short text marking [Mohler *et al.*, 2011]. It is an exact verification problem which aims to verify whether the mathematical expressions in the user answer and the standard solution are equivalent or not, whereas similarity-based solution assessment aims to measure the similarity between the user answer and the standard solution according to a similarity threshold. The current techniques and Computer Algebra Systems such as Maple and Mathematica provide support for mathematical equivalence checking. However, these techniques have generated many false negative errors if the mathematical expressions are written in a different form from the standard solution.

In this paper, we propose a novel and effective Probabilistic Equivalence Verification (PEV) approach for automatic mathematical solution assessment. The proposed PEV approach is a randomized method based on probabilistic numerical equivalence testing of two mathematical expressions. It can avoid false negative errors completely while guaranteeing a small probability of false positive errors to occur. For the rest of this paper, we will discuss the related work, present the proposed PEV approach and its performance results, and give the conclusion.

2 Related Work

Currently, there are 3 main approaches for automatic mathematical solution assessment, namely computer algebraic approach, rule-based approach and structural approach. The computer algebraic approach is based on symbolic algebraic computation [Cohen, 2003; Von Zur Gathen and Gerhard, 2003] for automatic mathematical solution assessment. This technique is implemented in commercial Computer Algebra Systems such as Mathematica [2011] and Maple [2011]. The rule-based approach [Buchberger and Loos, 1982; Moses, 1971] is based on mathematical rules for equivalence assessment. Starting with a mathematical expression, this approach transforms it into another equivalent expression and compares it with the intended target expression for equivalence verification. The structural approach [Shatnawi and Youssef, 2007; Youssef and Shatnawi, 2006] is based on the tree representation of a mathematical expression. In this approach, two mathematical expressions are first represented as two math-

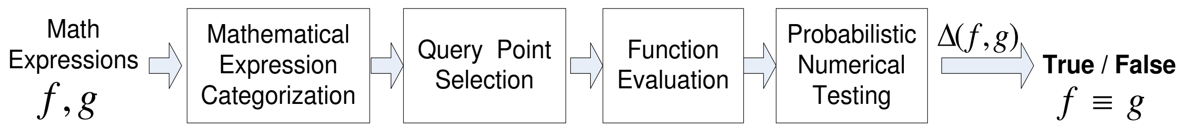


Figure 1: Probabilistic Equivalence Verification (PEV) Approach

ematical expression trees. Then, tree matching algorithm [Apostolico and Galil, 1997] is used to compare the two trees using dynamic programming [Zhang and Shasha, 1989] for equivalence verification.

Due to the nature of symbolic computation [Yap, 2000; Cohen, 2003], all these approaches can avoid false positive errors completely. However, these approaches may generate false negative error if the provided correct answer is written in a different form from the given standard solution.

3 Problem Statement

In most mathematics questions, mathematical expressions are the most common form of the required answers. The mathematical equivalence verification problem is specified as follows. Given 2 mathematical functions $f(x_1, x_2, .., x_n)$ and $g(x_1, x_2, .., x_n)$, mathematical equivalence verification aims to verify the equivalence of the two mathematical functions f and g. Mathematically, two functions f and g are said to be *equivalent* ($f \equiv g$), if and only if they have the same value at every point in the domain $\mathcal{D}$ (e.g., $\mathbb{Z}^n, \mathbb{R}^n$, etc.). Note that $f \equiv g$ iff $h = f - g \equiv 0$. Therefore, mathematical equivalence verification can be considered as equivalent to check whether function h is equivalent to zero, i.e., $h(x_1, .., x_n) \equiv 0, \forall x_1, .., x_n \in \mathcal{D}$. Table 1 shows some examples of mathematical answer and solution expressions.

Table 1: Examples

	Example	
	Answer	Solution
Constant	$\frac{2}{\sqrt{5}}$	$\frac{2\sqrt{5}}{5}$
Polynomial	$(x+1)^2(1-x)$	$(1+x)(1-x^2)$
Exponential + log	$e^{2+\ln x}$	$e^2 x$
Trigonometric	$\frac{\sin x + \cos x}{\sqrt{2}}$	$\sin(x + \frac{\pi}{4})$
Roots	$\sqrt{x^2 - 6x + 9}$	$\lvert x - 3 \rvert$
Fractional	$\frac{(x^2+2)(x+1)}{x^3}$	$\frac{x^3+x^2+2x+2}{x^3}$
Mixed	$x \sin x + e^{\ln x}$	$x(\sin x + 1)$

4 Probabilistic Equivalence Verification

Probabilistic or randomized algorithms [Alon and Spencer, 2008; Motwani and Raghavan, 1996] have been studied quite extensively for the last 30 years. As compared with purely deterministic algorithms, probabilistic algorithms are generally faster, simpler, and easier to implement. In particular, probabilistic algorithms are the most effective and efficient algorithms for a class of semantic or equivalence verification problems such as fingerprinting [Rabin, 1981], DNA screening [Ngo and Du, 2000], primality testing [Agrawal *et al.*, 2004] and boolean expression equivalence testing [Blum *et al.*, 1980]. In this paper, we propose a novel Probabilistic Equivalence Verification (PEV) approach for equivalence verification of two mathematical expressions. As a mathematical expression is analogous to a programming function, we can verify the equivalence of two mathematical expressions by borrowing the idea of testing methods in software engineering [Beizer, 2002]. More specifically, the proposed PEV approach is based on probabilistic numerical testing method [Alon and Spencer, 2008] and algebraic properties of mathematical expressions. Figure 1 shows the proposed PEV approach, which consists of four main steps: Mathematical Expression Categorization, Query Point Selection, Function Evaluation and Probabilistic Numerical Testing.

Before discussing these steps in details, we need to define the following definition.

Let D be a set in a finite field $\mathbb{F}$ and R be the real number field. In the mathematical verification problem, we aim to determine probabilistically how *close* a test function $f : \mathbb{F} \to \mathbb{R}$ is to a target function g according to a well-defined measure. We introduce the concept of distance to formalize the notion of "closeness".

Definition 1 (Distance). Let f be the test function and g be the target function. The closeness of the two functions f and g is measured in terms of the *Hamming* distance over D as follows:

$$\Delta(f,g) = \Pr_{x \in D}[f(x) \neq g(x)] = \frac{|\{x \in D | f(x) \neq g(x)\}|}{|D|}$$

The distance function is needed to determine the minimal number of testing points that guarantees the likelihood of false positive errors. Although it is difficult to derive the distance $\Delta(f,g)$ explicitly, we can estimate the Hamming distance $\Delta(f,g)$ of f and g by sampling random points of $x \in D$ such that $f(x) \neq g(x)$.

Algorithm 1 presents the high level description of the PEV approach. It repeatedly evaluates the value of f at random points $r_1, r_2, .., r_n$ of the corresponding multivariate variable $x = x_1, x_2, .., x_n$, which are sampled uniformly from a sufficiently large range. If the evaluation result is equal to 0, it returns true. Otherwise, it returns false. Different from CASs such as Maple and Mathematica, PEV can avoid false negative errors completely while guaranteeing small possibility for false positive errors to occur. As such, the proposed PEV algorithm is effective for verifying the equivalence of 2 mathematical expressions which are equivalent but may be expressed in different forms.

Algorithm 1: Probabilistic_Equivalence_Verification

```
Input: $f(x), g(x)$ - test and target functions
Output: True if $f \equiv g$, False if $f \not\equiv g$
Process:
1   Identify the expression category of $f$ and $g$;
2   Select a set of trial points $V \subset \mathcal{D}$ randomly such
    that $|V|$ is sufficiently large;
3   Select a variable domain $x \in V \subset \mathcal{D}$ and a
    function target value range $\mathbb{T}$ for evaluating $f$
    and $g$;
4   repeat
5       Select a random point $x$ from $V$ uniformly
        and independently;
6       Test $f(x) \neq g(x)$ and Update $\Delta(f, g)$;
    until $|V|$;
7   if $\Delta(f, g) \geq 1 - \epsilon$ then return False;
8   else return True;
```

4.1 Mathematical Expression Categorization

It categorizes a given pair of mathematical expressions based on the Latex representation format. The category will be used for selecting an appropriate algorithm for equivalence verification. Each mathematical expression is categorized according to the following 5 expression categories based on the increasing order of complexity for equivalence verification: Constant (CO), Univariate Polynomial (UP), Multivariate Polynomial (MP), Univariate General Function (UF) and Multivariate General Function (MF). Before discussing the details, we need to distinguish the different expression categories. As the constant, univariate polynomial and univariate general function are straightforward, we only discuss the multivariate polynomial and multivariate general function.

Definition 2 (Multivariate Polynomial). Let $\mathbb{F}$ be a finite field and $|\mathbb{F}| = q$ and $d < q$. A function $f : \mathbb{F}^m \leftarrow \mathbb{R}$ is said to be a multivariate polynomial with degree d if it can be expressed as follows:

$$f(x_1, x_2, ..., x_m) = \sum_{i_1+...+i_m \leq d} a_{a_1...i_m} x^{i_1} ... x^{i_m}, x_i \in \mathbb{F}^m$$

The degree of a particular monomial in a multivariate polynomial f is defined as the sum of the exponent of each variable in that monomial. The degree of f, denoted as $deg(f)$, is defined as the maximum degree of all the monomials in f. For example, the degree of the monomial $x_1^6 x_2 x_3^5$ is 12, and the degree of $f = x_1^3 x_2^2 + x_1^6 x_2 x_3^5$ is $deg(f) = 12$.

Multivariate general function f is basically a multivariate polynomial with fractional functions or transcendental functions applied on a variable x such as $\frac{x^3-1}{x+1}$, $\sin x$, $\sqrt{x}$ and e^x.

The category of a pair of expressions is categorized according to the highest order of the two. Table 2 gives some examples of answers and solutions for the different mathematical expression categories.

In this research, mathematical expressions are represented in the Latex format. For example, the expression $(a + b)^2$ has its Latex format of (a+b)^2. Similarly, the expression $e^{2+\ln x}$ has its Latex format of e^{2+\ln x}. To categorize a mathematical expression, content features of the Latex format of the mathematical expression is used. There are four main types of content features: constant, univariate variable, multivariate variable and elementary function. Table 3 shows some examples of the content features of answers and solutions. Based on the extracted content features, we can categorize a mathematical expression according to the decision table given in Table 4.

Table 2: Examples of Mathematical Expression Categories

Category	Example	
	Answer	Solution
CO	$\frac{2}{\sqrt{5}}$	$\frac{2\sqrt{5}}{5}$
UP	$(x+1)^2(1-x)$	$(1+x)(1-x^2)$
MP	$x^4y^2 - 3x^2y - 10$	$(x^2y-2)(x^2y-5)$
UF	$e^{2+\ln x}$	e^2x
MF	$\sqrt{x^2-6x+9y^2}$	$\|x-3y\|$

Table 3: Examples of Category Features

Content Feature	Example
Constant	2, $\sqrt{5}$, $\frac{3}{10}$, $\sin \pi/6$,...
Univariate Variable	{a}, {b}, {c}, {x}, {y}, {z},...
Multivariate Variable	{a, b, c}, {x, y}, {p, q, r, s},...
Elementary Function	\ln, \frac, \sqrt, a^, \sin, \cot, ...

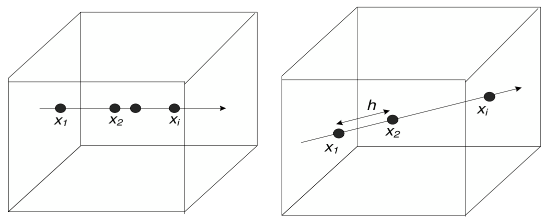
Table 4: Content Features for Math Expression Categories

Category	Content Feature			
	Cons	Uni Var	Mul Var	Elem Func
CO	✔	✘	✘	✘
UP	✔	✔	✘	✘
MP	✔	✘	✔	✘
UF	✔	✔	✘	✔
MF	✔	✘	✔	✔

4.2 Query Point Selection

It aims to select random points according to a geometric structure such as axis or line, and determine the query point size for efficient equivalence verification.

Borrowing the strategies [Ibarra and Moran, 1983] used in software program testing, we use the following two strategies to select query points: axis-based selection and line-based selection. The axis-based selection strategy is used for univariate function verification, whereas the line-based selection strategy is used for multivariate function verification. The axis-based selection strategy chooses random points along the axis of a variable. The line-based selection strategy chooses random points along a line $l = \{x_1 + th\}$, where $t = 1, 2, .., k$, which passes through x_1 with a constant slope h: $x_1 \in \mathbb{F}^m, h \in \mathbb{F}^m$. Figure 2 gives an illustration of the two strategies.



(a) Axis-based Selection (b) Line-based Selection

Figure 2: Query Point Selection Strategies

In our proposed probabilistic approach, the query point size, i.e., the number of distinguishing query points, is very important as it determines the correctness of the equivalence verification results. In Section 5, our theoretical investigation will show that the query point size can be determined based on the degree of f and g. In short, the query point size $|V|$ is determined by $|V| = \max\{deg(f), deg(g)\}$.

Gathen et al. [2003] showed that any elementary function can be expressed in terms of a quotient of two polynomials. Hence, the above discussion is also applicable for general functions even though their degrees are not defined explicitly.

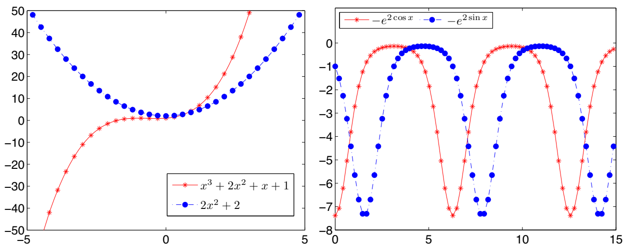
4.3 Function Evaluation

In this research, we assume that the functions $f(x)$ and $g(x)$ can be evaluated accurately by using a computer algebra system. Thus, this assumption simplifies the computational implementation issues to obtain fixed point accuracy. However, in some situations, it is possible that the functions f and g may have high degree terms such as x^d or consist of exponents such as a^x. Hence, evaluating these terms would result in infinite values due to large values of either d or x. Thus, it will generate false negative results and degrade the performance.

To overcome this, we also observe that it is sufficient to map the final result $f(x)$ to $f(x) \mod p, p \geq |V|$, where p is a prime number and V is the size of random sampling points which will be discussed in the next section. Often, the sampling size $|V|$ is small and it is easy to find a prime number $p \geq |V|$. As such, we can avoid the issue on computing large values. For efficiency, we apply the Multiply-And-Square algorithm [Yap, 2000] to compute the value of $f(x) \mod p$ when $f(x)$ contains high degree terms or exponents.

4.4 Probabilistic Numerical Testing

It aims to estimate the distance $\Delta(f, g)$ between two mathematical expressions f and g based on the point selection strategy and function evaluation discussed in the previous steps. Specifically, it repeatedly selects a random point $x \in V$ in the query point set uniformly and independently, and verifies whether $f(x) \neq g(x)$ or not. Then, it estimates the distance $\Delta(f, g)$ based on the fraction of number of successful points such that $f(x) \neq g(x)$ and the query point size. Lastly, the equivalence of two mathematical expressions f and g is determined based on whether $\Delta(f, g) \geq 1 - \epsilon$. Note that the query point size depends on both of the mathematical expression category and the desired error bound ϵ, i.e., $|V| = \frac{d+2}{\epsilon}$ for univariate polynomials. In other words, the query point size determines the trade-off between the correctness of verification result and computational time.



(a) Univariate Polynomial (b) Univariate General Function

Figure 3: Examples of Equivalence Verification

5 Theoretical Analysis

This section aims to study the correctness of the PEV approach. The correctness property ensures that the algorithm will return *True* with a high probability if the two expressions have the same value on sufficiently large number of random points of x. In this case, the correctness property determines the false negative error. To guarantee the correctness property, we need to prove that the Hamming distance $\Delta(f, g)$ is small. In additon, the correctness property also guarantees that the algorithm will return *False* with a high probability if the two expressions do not agree on sufficiently large number of random points of x. The correctness property determines the false positive error. To guarantee the correctness property, we need to prove that the Hamming distance $\Delta(f, g)$ is high. We aim to design an effective and efficient equivalence verification algorithm, which can minimize the false positive error, false negative error and query size. Depending on the mathematical category of the two expressions $f(x)$ and $g(x)$, we have developed different appropriate verification algorithms accordingly. These algorithms differ on the number of query points, the strategy to select query points and the domain values of query points.

Here, we provide some theoretical analysis on the correctness of our approach. The basic idea of our proof is that if f is not equivalent to g, then the existent probability of x such that $f(x) = g(x)$ is small. Figure 3 gives 2 examples of equivalence verification for a pair of expressions with univariate polynomials and another pair of expressions with univariate general functions. It can be seen that when the complexity of the expressions increases, the probability of false positive errors to occur also increases. It also explains why the query point size for equivalence verification of higher complexity mathematical categories is larger.

Theorem 1 (Univariate Polynomial). Let d be the maximum degree of the univariate test polynomial $g(x) : \mathbb{V} \to \mathbb{R}$ and the univariate target polynomial $g(x) : \mathbb{V} \to \mathbb{R}$, where $V \subset \mathbb{Z}$ is a set of at least $d + 2$ integer points of x. If there exists a function $f(x)$ satisfying that $f(x_i) = g(x_i), x_i \in V$, $\forall i = 1, .., d{+}2$, then $f(x) \equiv g(x)$. In other words, the two polynomials $f(x)$ and $g(x)$ are equivalent, i.e., $\Delta(f, g) = 0$. In addition, if $f(x) \not\equiv g(x)$, then $f(x)$ agrees with $g(x)$ at at most $d+1$ points. Hence, we have the probability of false positive error:

$$\Pr[f(x)=g(x)] = \frac{|\{x \in V | f(x)=g(x)\}|}{|V|} \leq \frac{d+1}{|V|} = \epsilon$$

By definition, we have $\Delta(f,g) \geq 1-\epsilon$. Therefore, it means that $f(x)$ is not close to $g(x)$ with a high probability. Moreover, the minimal query size of V such that the correctness property is guaranteed is $d+2$.

Proof: This proof is straightforward due to the well-known Lagrange Interpolation Polynomial Theorem [Hardy and Wright, 1980], which states that two univariate polynomials $g(x)$ and $f(x)$ with degree at most d can coincide at most $d+1$ points of x. ∎

Theorem 2 (Multivariate Polynomial). Let d be the maximum degree of the multivariate test polynomial $f(x) : \mathbb{V} \to \mathbb{R}$ and the multivariate target polynomial $g(x) : \mathbb{V} \to \mathbb{R}$, where $V \subset \mathbb{Z}$ is a set of at least d multivariate integer points of $x \in \mathbb{Z}^m$. If $f(x) \not\equiv g(x)$, then the probability that the PEV algorithm has false positive error is bounded by

$$\Pr[f(x)=g(x)] \leq \frac{d}{|V|} = \epsilon$$

By definition, we have $\Delta(f,g) \geq 1-\epsilon$. Therefore, it means that $f(x)$ is not close to $g(x)$ with a high probability. In addition, the minimal query size of V such that the correctness property is guaranteed is $d+1$.

Proof: It is straightforward that the PEV algorithm has no false negative error as it is a trial-and-error method. The false positive error bound can be proved using mathematical induction on the number of variables n of the polynomial. Let $P(x) = f(x) - g(x)$ be a multivariate polynomial. It is equivalent to prove that $\Pr[P(r_1, r_2, .., r_n) = 0] \leq \frac{d}{|V|}$.

For the case $n=1$, if $P \not\equiv 0$, then P is a univariate variable polynomial with degree d. Thus, it has at most d roots. Hence, $\Pr[P(r_1, r_2, .., r_n) = 0] \leq \frac{d}{|V|}$.

For the inductive step, we can solve the multivariate case by fixing $n-1$ variables $x_2, .., x_n$ to $r_2, .., r_n$ and apply the result from the univariate case. Assume that $P \not\equiv 0$, we compute $\Pr[P(r_1, .., r_n) = 0]$. First, let k be the highest power of x_1 in P. We can rewrite P as

$$P(x_1, x_2, .., x_n) = x_1^k A(x_2, .., x_n) + B(x_1, x_2, .., x_n)$$

for some polynomials A and B. Let $\mathcal{X}_1$ be the case that $P(r_1, .., r_n)$ is evaluated to 0, and $\mathcal{X}_2$ be the case that $A(r_2, .., r_n)$ is evaluated to 0. Using Bayes rule, we can rewrite the probability that $P(r_1, .., r_n) = 0$ as

$$\begin{aligned}\Pr[P(r)=0] &= \Pr[\mathcal{X}_1] \\ &= \Pr[\mathcal{X}_1|\mathcal{X}_2]\Pr[\mathcal{X}_2] + \Pr[\mathcal{X}_1|\neg\mathcal{X}_2]\Pr[\neg\mathcal{X}_2] \\ &\leq \Pr[\mathcal{X}_2] + \Pr[\mathcal{X}_1|\neg\mathcal{X}_2] \qquad (1)\end{aligned}$$

Consider the probability of $\mathcal{X}_2$ (or $A(r_2, .., r_n) = 0$), the polynomial A has degree $d-k$ with $n-1$ variables. So, by inductive hypothesis on the number of variables, we obtain

$$\Pr[\mathcal{X}_2] = \Pr[A(r_2, .., r_n) = 0] \leq \frac{d-k}{|V|} \qquad (2)$$

Similarly, if $\neg\mathcal{X}_2$ (or $A(r_2, .., r_n) \neq 0$), P is a univariate polynomial degree k. By inductive hypothesis, we have

$$\Pr[\mathcal{X}_1|\neg\mathcal{X}_2] = \Pr[P(r_1, ., r_n) = 0 | A(r_2, ., r_n) \neq 0] \leq \frac{k}{|V|} \qquad (3)$$

Finally, we can substitute the results from Equations (1) and (2) into Equation (3) to complete the inductive step. ∎

Theorem 3 (General Function). Let $d = \max\{2^{deg(f)}, 2^{deg(g)}\} + 1$ be the maximum degree of the test function $f(x) : \mathbb{V} \to \mathbb{R}$ and the target function $g(x) : \mathbb{V} \to \mathbb{R}$, where $V \subset \mathbb{Z}$ is a set of at least d multivariate integer points $x \in \mathbb{Z}$ or $x \in \mathbb{Z}^m$. If $f(x) \not\equiv g(x)$, then the probability that the PEV algorithm has false positive error is bounded by

$$\Pr[f(x)=g(x)] \leq \frac{d}{|V|} = \epsilon$$

By definition, we have $\Delta(f,g) \geq 1-\epsilon$. Therefore, it means that $f(x)$ is not close to $g(x)$ with a high probability. In addition, the minimal query size of V such that the correctness property is guaranteed is $\max\{2^{deg(f)}, 2^{deg(g)}\} + 1$.

Proof: Gathen et al. [Von Zur Gathen and Gerhard, 2003] showed that every mathematical expression $f(x)$ can be represented as a quotient of two polynomials, i.e., $f(x) = \frac{f_1(x)}{f_2(x)}$ for some polynomials $f_1(x)$ and $f_2(x)$. Let $f(x)$ and $g(x)$ be represented in terms of quotients of polynomials, i.e., $f(x) = \frac{f_1(x)}{f_2(x)}$ and $g(x) = \frac{g_1(x)}{g_2(x)}$. By equating the denominators of the 2 quotients, the problem becomes verifying the equivalence of two polynomials, which is similar to the previous proof. ∎

6 Performance Evaluation

To evaluate the proposed PEV approach, we have devised 100 answer-solution pairs with 50 correct (positive) answers, and 50 incorrect (negative) answers for 100 test questions based on the undergraduate mathematics course contents. The test questions are general questions selected according to some standard mathematical functions. The 100 test cases are representative, completely different and well-selected to cover all possible function types and different levels of complexity for equivalent verification. Table 5 shows some examples of the pairs of answers and solutions for the test questions. As we are unable to obtain the codes for the existing techniques, we only evaluate the performance of PEV and compare it with Maple and Mathematica whose performance outperformed other techniques for mathematical expression equivalence verification. In addition, we also used SimPy [2013], which is a Python library for symbolic mathematics and computer algebra system, for processing Latex represented mathematical expressions.

We use the precision measure to evaluate the performance of the proposed PEV algorithm. Precision is defined as the

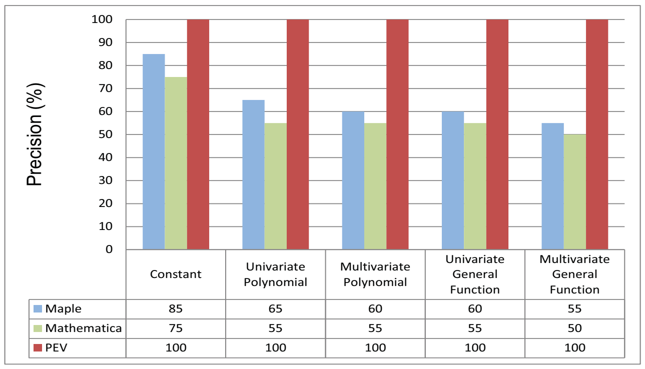
Table 5: Test Dataset for Equivalent Answers

#	Example		
	Answer	Solution	
1	$\frac{2}{\sqrt{5}}$	$\frac{2\sqrt{5}}{5}$	CO
2	$2^{10}3^5$	$2^5 6^5$	CO
3	$\frac{\log_2 25}{\log_2 5}$	2	CO
4	$\cot^2 36^\circ \cot^2 72^\circ$	$1/5$	CO
5	$2x^{10}+5x^5+3$	$(2x^5+3)(x^5+1)$	UP
6	$(x+1)^2(1-x)$	$(1+x)(1-x^2)$	UP
7	$25x^4-16$	$(5x^2-4)(5x^2+4)$	UP
8	$x^6-x^4-2x^3+2x^2$	$x^2(x-1)^2(x^2+2x+2)$	UP
9	$3x+10xy-5y-6x^2$	$(3x-5y)(1-2x)$	MP
10	$x^4y^2-3x^2y-10$	$(x^2y+2)(x^2y-5)$	MP
11	$e^{2+\ln x}$	e^2x	UF
12	$\frac{\sin x+\cos x}{\sqrt{2}}$	$\sin(x+\frac{\pi}{4})$	UF
13	$\sqrt{x^2-6x+9}$	$\|x-3\|$	UF
14	$\frac{(x^2+2)(x+1)}{x^3}$	$\frac{x^3+x^2+2x+2}{x^3}$	UF
15	$x\sin x+e^{\ln x}$	$x(\sin x+1)$	UF
16	$x^{\sqrt{\log_x y}}$	$y^{\sqrt{\log_y x}}$	MF
17	$\frac{x^2y}{xy^2+3y}$	$\frac{x^2}{xy+3}$	MF
18	$\frac{9\pi x^2}{y-\sqrt{3}}$	$\frac{9\pi x^2(y+\sqrt{3})}{y^2-3}$	MF
19	$\ln(xy)-\ln y$	$\ln x$	MF
20	$\log_x y^2+\log_{x^2} y^4$	$\log_x y$	MF

Table 6: Performance Results based on Error Types

	False Negative		False Positive	
	(#)	(%)	(#)	(%)
Maple	35	70	0	0.0
Mathematica	42	84	0	0.0
PEV	0	0.0	0	0.0

percentage of total correct verifications from all verifications. In addition, we also measure the false negative errors and false positive errors. Note that the performance evaluation is based on the final verification result of all steps in the PEV approach. Figure 4 shows the performance results based on precision. It shows that the proposed PEV algorithm has consistently outperformed Maple and Mathematica. PEV can achieve 100% performance in precision while Maple and Mathematica can only achieve about 65% and 58% respectively. This is because PEV can avoid false negative errors and it can also reduce false positive errors to as small as possible by using many trials. Theoretically, there may occur false positive errors. However, PEV can guarantee the possibility of the false positive errors to be closed to zero by sampling a sufficiently large number of points such that the two expressions will disagree on some of these points if they are not equivalent. Table 6 gives the performance results based on the false negative error and false positive error for Maple, Mathematica and PEV. As shown from the table, both commercial CAS systems produce false negative errors with 70% in Maple and 84% in Mathematica. This has shown the effectiveness of the proposed PEV algorithm in verifying the equivalence of two mathematical expressions numerically rather than symbolically.



	Constant	Univariate Polynomial	Multivariate Polynomial	Univariate General Function	Multivariate General Function
🟦 Maple	85	65	60	60	55
🟩 Mathematica	75	55	55	55	50
🟥 PEV	100	100	100	100	100

Figure 4: Performance Results based on Precision

7 Conclusion

In this paper, we have proposed a novel approach for automatic mathematical solution assessment in Web-based mathematics testing. The proposed approach is an effective Probabilistic Equivalence Verification algorithm for mathematical solution assessment. In this paper, we have discussed and analyzed the correctness of our approach for most of the typical mathematical function types. In addition, we have also analyzed the efficiency of the proposed approach by determining the minimal number of testing points such that the small likelihood of false positive errors is guaranteed. The performance results have shown that the proposed PEV approach is very effective for mathematical answer verification and assessment. For future work, we intend to investigate Web-based testing techniques for other subjects such as physics and chemistry which contain mathematical and chemical expressions as the required answer type.

References

[Agrawal *et al.*, 2004] M. Agrawal, N. Kayal, and N. Saxena. Primes is in p. *Annals of mathematics*, 160(2):781–793, 2004.

[Alon and Spencer, 2008] N. Alon and J. H. Spencer. *The probabilistic method*, volume 73. Wiley-Interscience, 2008.

[Apostolico and Galil, 1997] A. Apostolico and Z. Galil. *Pattern matching algorithms*. Oxford University Press, USA, 1997.

[Beizer, 2002] B. Beizer. *Software testing techniques*. Dreamtech Press, 2002.

[Blum *et al.*, 1980] M. Blum, A. K. Chandra, and M. N. Wegman. Equivalence of free boolean graphs can be decided probabilistically in polynomial time. *Information Processing Letters*, 10(2):80–82, 1980.

[Buchberger and Loos, 1982] B. Buchberger and R. Loos. Algebraic simplification. *Computer Algebra-Symbolic and Algebraic Computation*, pages 11–43, 1982.

[Cohen, 2003] J.S. Cohen. *Computer algebra and symbolic computation: Mathematical methods*. Universities Press, 2003.

[Coursera, 2013] Coursera. https://www.coursera.org/. 2013.

[Edventure, 2013] Edventure. Edventure, https://edventure.ntu.edu.sg. 2013.

[Gulwani *et al.*, 2011] S. Gulwani, V.A. Korthikanti, and A. Tiwari. Synthesizing geometry constructions. In *ACM SIGPLAN Notices*, volume 46, pages 50–61. ACM, 2011.

[Guzman and Conejo, 2007] E. Guzman and R. Conejo. Improving student performance using self-assessment tests. *IEEE Intelligent Systems*, 22(4):46–52, 2007.

[Hardy and Wright, 1980] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*. Oxford University Press, USA, 1980.

[Hearst, 2000] M. A. Hearst. The debate on automated essay grading. *IEEE Intelligent Systems and their Applications*, 15(5):22–37, 2000.

[Hyman, 2012] Paul Hyman. In the year of disruptive education. *Commun. ACM*, 55(12):20–22, December 2012.

[Ibarra and Moran, 1983] O.H. Ibarra and S. Moran. Probabilistic algorithms for deciding equivalence of straight-line programs. *Journal of the ACM (JACM)*, 30(1):217–228, 1983.

[Mapple, 2011] Mapple. Version 15, http://www.maplesoft.com/. 2011.

[Mathematica, 2011] Mathematica. http://www.wolfram.com/mathematica/. 2011.

[Mohler *et al.*, 2011] M. Mohler, R. Bunescu, and R. Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of Association for Computational Linguistics*, pages 752–762, 2011.

[Moses, 1971] J. Moses. Algebraic simplification a guide for the perplexed. In *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, pages 282–304, 1971.

[Motwani and Raghavan, 1996] R. Motwani and P. Raghavan. Randomized algorithms. *ACM Computing Surveys (CSUR)*, 28(1):37, 1996.

[Ngo and Du, 2000] H. Q. Ngo and D. Z. Du. A survey on combinatorial group testing algorithms with applications to dna library screening. *Discrete mathematical problems with medical applications*, 55:171–182, 2000.

[Rabin, 1981] M. O. Rabin. *Fingerprinting by random polynomials*. Center for Research in Computing Techn., Aiken Computation Laboratory, University, 1981.

[Rasila *et al.*, 2007] A. Rasila, M. Harjula, and K. Zenger. Automatic assessment of mathematics exercises: Experiences and future prospects. *ReflekTori Symposium of Engineering Education*, pages 70–80, 2007.

[Rasila *et al.*, 2010] A. Rasila, L. Havola, H. Majander, and J. Malinen. Automatic assessment in engineering mathematics: evaluation of the impact. *ReflekTori Symposium of Engineering Education*, pages 37–45, 2010.

[Shatnawi and Youssef, 2007] M. Shatnawi and A. Youssef. Equivalence detection using parse-tree normalization for math search. In *2nd International Conference on Digital Information Management, ICDIM*, volume 2, pages 643–648, 2007.

[Sim, 2013] Simpy, http://sympy.org/en/index.html, 2013.

[Singh *et al.*, 2012] R. Singh, S. Gulwani, and S. Rajamani. Automatically generating algebra problems. *AAAI*, 2012.

[Von Zur Gathen and Gerhard, 2003] J. Von Zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 2003.

[Yap, 2000] C. K. Yap. *Fundamental problems of algorithmic algebra*, volume 54. Oxford University Press Oxford, 2000.

[Youssef and Shatnawi, 2006] A. Youssef and M. Shatnawi. Math search with equivalence detection using parse-tree normalization. In *The 4th International Conference on Computer Science and Information Technology*, 2006.

[Zhang and Shasha, 1989] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989.