Diplomarbeit

Deciding Polynomial-Exponential Problems

Melanie Achatz

Erstgutachter:	Prof. Dr. Volker Weispfenning
Zweitgutachter:	PD Dr. Thomas Sturm
Betreuer:	Scott McCallum, Ph.D., Macquarie University, Sydney
	Prof. Dr. Volker Weispfenning

29. September 2006

Prof. Dr. Volker Weispfenning
Lehrstuhl für Mathematik
Fakultät für Informatik und Mathematik
Universität Passau

Abstract

This thesis presents a decision procedure for certain polynomial-exponential problems for the real numbers. The inputs are restricted to prenex sentences where the quantified variables are of an extension of the elementary theory of the real closed field and only the outermost quantified variable occurs both in a polynomial and in an exponential function. The approach taken is based on S. McCallum and V. Weispfenning's paper "Deciding Polynomial-Transcendental Problems" [19].
The algorithm described here is implemented in the computer logic system Redlog.

Contents

Chapter 1

Introduction

- "Is there an algorithm answering the question of whether some sentence in a certain language is true or false?"
This is the question we ask on solving a decision problem. In 1948, Tarski proved the decidability of the first-order theory of the real numbers by his theorem on the real closed field [23]. Over the following years, many efforts were made to extend the first-order logic to a language including the real exponential function.
But it was not until 1996 that A. Macintyre and A.J. Wilkie succeeded in proving the decidability of the real exponential field [18]. However, their proof is based on the yet unproven Schanuel's conjecture.

Although V. Weispfenning considers only restricted classes of the first-order theory of the real closed field, his approach goes a new way by depending only on Lindemann's theorem as opposed to depending on a hypothesis [1, 19, 25, 26].
Based on this approach, the goal of this thesis is to outline a decision procedure for certain sentences involving mixed integral polynomials and the real exponential function. In this regard, the decision procedure in this thesis and its proof rely on S. McCallum and V. Weispfenning's paper "Deciding Polynomial-Transcendental Problems" [19]. Their approach is called *cylindrical analytic decomposition* and is in the spirit of cylindrical *algebraic* decomposition. Accordingly, their procedure consists of three basic steps: the *projection process*, the *cell decomposition of the real line*, and the *lifting process*.

In this thesis, the decision procedure reduces an input sentence to a polynomial-exponential problem in only one variable. Consequently, we consider in particular the univariate case of polynomial-exponential problems. For this

one dimensional case, we do not need to apply the projection process and the lifting process of the cylindrical analytic decomposition. But we have to describe in detail a cell decomposition of the real line which has specific properties regarding the exponential function. Hence, we are able to evaluate the truth value of the original input sentence.

The decision algorithm presented is implemented in the computer logic system REDLOG. Decision procedures have a wide range of applications in mathematics, computer science and industrial engineering. As the exponential function is of utmost importance, the decision algorithm described in this thesis gives new perspectives for applications in many areas.

The chapters of the thesis can be outlined as follows:
In *chapter 2*, we develop the formal framework and recall the most essential background material. In particular, we introduce exponential polynomials, and define the extension of the first-order theory of the real closed field. In addition, we consider the properties of the exponential function and exponential polynomials. In *chapter 3*, our goal is to outline an algorithm that decides univariate polynomial-exponential problems. First, we introduce the theoretical definitions needed for this algorithm, followed by the specification and a brief description. Before we give a full description of the procedure, we study its basic subalgorithm that isolates real roots of exponential polynomials. Furthermore, we discuss the data used in the algorithm, i.e. cell indices and sample points, and the evaluation of the sign of exponential polynomials. A description of the main algorithm for deciding polynomial-exponential problems is given in *chapter 4*. The Implementation is discussed in *chapter 5*. In the conclusion, *chapter 6*, we consider some generalizations of polynomial-exponential problems. In the *appendix*, detailed examples are given.

Chapter 2

Formal Framework and Background Material

The algorithm to decide polynomial-exponential problems takes sentences in prenex form as input. These sentences are created of an extension of the first order theory of the real closed field. Thus, in this chapter we introduce exponential polynomials, define the extension of the first order theory, and specify some properties of the exponential function and exponential polynomials.

2.1 Exponential Polynomials

In the following, exp denotes the exponential function $x \mapsto e^x$ defined for all $x \in \mathbb{R}$.

Definition 2.1 (exponential polynomial)
*Let p be an integral polynomial in x and y, i.e. $p \in \mathbb{Z}[x, y]$. We can view p as element of $\mathbb{Z}[x][y]$ which is the representation of p in y. Thus, we can write p in the form $\sum_{i=0}^{n} p_i(x)y^i$ with $p_i(x) \in \mathbb{Z}[x]$ and $p_n(x) \neq 0$. Putting $y = \exp(x)$ we obtain an **exponential polynomial** $p^*(x) := p(x, \exp(x))$.*

Remark 2.2
$p^*(x)$ is a real valued analytic function, defined on the whole real line (see [17]).

Theorem 2.3
Let $p(x, y)$ be an integral polynomial in x and y, and let $p^*(x) = p(x, \exp(x))$ be the exponential polynomial. Then the mapping $p(x, y) \to p^*(x)$ is an injective homomorphism of $\mathbb{Z}[x, y]$ into the ring of all real valued analytic functions defined on the whole real line.

Proof. Let $\mathcal{R}$ denotes the ring of all real valued analytic functions defined on the whole real line. Let $H : \mathbb{Z}[x, y] \to \mathcal{R}$ be the map defined by $H(p(x, y)) = p^*(x)$, with $p^*(x) = p(x, \exp(x))$.

First claim: H is a ring homomorphism.
Let $p = \sum_{i=0}^{n} p_i(x)y^i$, $q = \sum_{j=0}^{m} q_j(x)y^j$ with $n = \deg_y(p)$, $m = \deg_y(q)$ be two integral polynomials in x and y. Let $m \geq n$. Then:

$$\begin{aligned}
H(p(x,y)+q(x,y)) &= H(\sum_{i=0}^{n} p_i(x)y^i + \sum_{j=0}^{m} q_j(x)y^j) \\
&= H(p_0(x)+q_0(x)+(p_1(x)+q_1(x))y+...+ \\
&\quad (p_n(x)+q_n(x))y^n + q_{m+1}(x)y^{m+1}+...+ \\
&\quad q_m(x)y^m) \\
&= p_0(x)+q_0(x)+(p_1(x)+q_1(x))e^x+...+ \\
&\quad (p_n(x)+q_n(x))e^{nx}+q_{m+1}(x)e^{(m+1)x}+...+ \\
&\quad q_m(x)e^{mx} \\
&= \sum_{i=0}^{n} p_i(x)e^{ix} + \sum_{j=0}^{m} p_j(x)e^{jx} \\
&= p(x,\exp(x)) + q(x,\exp(x)) \\
&= p^*(x) + q^*(x) \\
&= H(p(x,y)) + H(q(x,y)). \\
H(p(x,y)\cdot q(x,y)) &= H((\sum_{i=0}^{n} p_i(x)y^i)\cdot(\sum_{j=0}^{m} q_j(x)y^j)) \\
&= H(\sum_{i=0}^{n+m}(\sum_{j=0}^{i} p_j(x)q_{i-j}(x))y^i) \\
&= \sum_{i=0}^{n+m}(\sum_{j=0}^{i} p_j(x)q_{i-j}(x))e^{ix} \\
&= (\sum_{i=0}^{n} p_i(x)e^{ix})\cdot(\sum_{j=0}^{m} q_j(x)e^{jx}) \\
&= p(x,\exp(x))\cdot q(x,\exp(x)) \\
&= p^*(x)\cdot q^*(x) \\
&= H(p(x,y))\cdot H(q(x,y)).
\end{aligned}$$

Therefore, H is a ring homomorphism.

Second claim: H is injective.
We have to show: $H(p(x,y)) = 0 \Rightarrow p(x,y) = 0$, i.e.

$$p(x,y) \neq 0 \Rightarrow H(p(x,y)) \neq 0.$$

Suppose $p(x,y) \neq 0$. By computing a real root bound for $p^*(x)$, it follows that there exists a real number x such that $p^*(x) \neq 0$. Therefore, H is injective. The computation of a real root bound for $p^*(x)$ is described and proved in section 3.3.2. □

Definition 2.4 (pseudodegree, derivation)
Let $p^*(x) = p(x, \exp(x))$ be an exponential polynomial. Then we define the following functions:

- *The **pseudodegree** of $p^*(x) \neq 0$ denoted as pdeg p^* to be the pair $(m,n) \in \mathbb{N}^2 = \mathbb{N} \times \mathbb{N}$, where $m = deg_y\ p(x,y)$ and $n = deg_x\ p(x,0)$, and $y = \exp(x)$. If $p(x,0) = 0$, we set $n = 0$.*
- *The **derivation** of $p^*(x)$ to be $(p^*)'(x) = p'(x,y)$, with $y = \exp(x)$, and $p'(x,y) = p_0'(x) + (p_1'(x) + p_1(x))y + (p_2'(x) + 2p_2(x))y^2 + ... + (p_n'(x) + np_n(x))y^n$. In particular, $(p^*)'(x) = (p')^*(x)$.*

Definition 2.5 (sylvestermatrix)
*Let $p = \sum_{i=0}^{n} p_i(x)y^i$, $q = \sum_{j=0}^{m} p_j(x)y^j$ with $p_i(x), q_j(x) \in \mathbb{Z}[x]$, and $p_n(x)$, $q_n(x) \neq 0$, be two integral polynomials represented in y. If $m, n \geq 1$ then the **Sylvester matrix** of p and q in y is defined as the following $n+m \times n+m$-matrix:*

$$Syl_y(p,q) = \begin{pmatrix} p_n & \cdots & p_0 & & & \\ & p_n & \cdots & p_0 & & \\ & & \ddots & & \ddots & \\ & & & p_n & \cdots & p_0 \\ q_m & \cdots & q_0 & & & \\ & q_m & \cdots & q_0 & & \\ & & \ddots & & \ddots & \\ & & & q_m & \cdots & q_0 \end{pmatrix}$$

Definition 2.6 (resultant)
*Let $p, q \in \mathbb{Z}[x,y]$ be two integral polynomials as defined in Definition 2.5. Then the **resultant** of p and q in y is the determinant of the Sylvester matrix of p and q in y:*

$$res_y(p,q) = det(Syl_y(p,q)).$$

Remark 2.7
Let $p, q \in \mathbb{Z}[x, y]$ be two integral polynomials as defined in Definition 2.5. If $p \neq 0$ and $q \neq 0$ then $res_y(p, q) = 0$ if and only if p and q have a common factor with positive degree in y.

Definition 2.8 (content, primitive part)
Let $p(x, y) = \sum_{i=0}^{n} p_i(x)y^i$ be a nonzero integral polynomial represented in y. We define the following functions:

- *The **content** of p with respect to y to be*

$$cont_y(p) = gcd(p_0, ..., p_n).$$

  We set $cont(0) = 0$.

- *If $p \neq 0$, the **primitive part** of p with respect to y to be*

$$pp_y(p) = \frac{p}{cont(p)}.$$

  We set $pp(0) = 0$.

Remark that a polynomial is primitive in case $cont(p) = 1$.

Definition 2.9 (squarefree part)
*Let $p(x, y) \in \mathbb{Z}[x, y]$. Then the **squarefree part** of p denoted as $sqfree(p)$ is the product of the distinct non-associated irreducible factors of $p(x, y)$.*
The following method is the basis of an efficient algorithm for computing $sqfree(p)$.

1. *Compute $c(x) = cont_y(p)$ and $q(x, y) = pp_y(p)$.*
2. *Compute $s_1(x) = sqfdec(c(x))$ by using a square-free decomposition $sqfdec$ for $c(x)$.*
3. *Compute $s_2(x) = sqfdec(q(x, y))$ by using by using a square-free decomposition $sqfdec$ for $q(x, y)$.*
4. *Finally we put $sqfree(p(x, y)) = s_1(x) \cdot s_2(x, y)$.*

Definition 2.10 (basis, squarefree basis)
*Let $\mathcal{A}$ be a set of integral polynomials over $\mathbb{Z}[x, y]$. Then a **basis** for $\mathcal{A}$ is a set $\mathcal{B}$ of primitive polynomials of positive degree over $\mathbb{Z}[x, y]$ satisfying the following three conditions:*

(a) If $B_1, B_2 \in \mathcal{B}$ and $B_1 \neq B_2$ then $gcd(B_1, B_2) = 1$.

(b) If $B \in \mathcal{B}$, then $B|A$ for some $A \in \mathcal{A}$.

(c) If $A \in \mathcal{A}$, there exist $B_1, ..., B_n \in \mathcal{B}$ and positive integers $e_1, ..., e_n$ such that

$$A \sim \prod_{i=1}^{n} B_i^{e_i} \text{ (with } n = 0 \text{ if } A \sim 1).$$

*A **squarefree basis** for $\mathcal{A}$ is a basis each of whose elements is squarefree. (For more details see [10]).*

2.2 Extension of the First-Order Theory of the Real Closed Field

The *extension of the first-order theory of the real closed field*, denoted as $\mathcal{T}_{\exp}$ defines the theory of the structure $\langle \overline{\mathbb{R}}, \exp \rangle$, where $\overline{\mathbb{R}} = \langle \mathbb{R}; +, -, \cdot, 0, 1, < \rangle$ denotes the real ordered field. $\mathcal{T}_{\exp}$ is expressed in the language $\mathcal{L}_{\exp} = (+, -, \cdot, \exp, <, 0, 1)$ of the structure $\langle \overline{\mathbb{R}}, \exp \rangle$, where we distinguish the binary functions addition, subtraction, and multiplication, the unary function exp, the binary order relation, and the real numbers 0 and 1.
Terms over this language are considered integral polynomials in the indeterminates $x_1, ..., x_n$ and another indeterminate y, where every occurrence of y is replaced by $\exp(x_1)$.

Remark 2.11
The exponential function only occurs in the indeterminate x_1 and never in the indeterminates $x_2, ..., x_n$.

Examples 2.12 (terms of $\mathcal{T}_{\exp}$)*:*

1. $(1 + x) + xy + (x^2 - 1)y^2 + (x - 1)y^3$, *where* $y = \exp(x)$.
2. $(x_1 + x_2) + x_2^2 y$, *where* $y = \exp(x_1)$.

By an *atomic formula* we mean an equation, inequation or inequality of the form $\tau = 0$, $\tau \neq 0$, $\tau > 0$, etc. where τ is a term. Thus, for example, the following are atomic formulas:

Examples 2.13 (atomic formulas of $\mathcal{T}_{\exp}$)*:*

1. $(1 + x) + xy + (x^2 - 1)y^2 + (x - 1)y^3 > 0$, *where* $y = \exp(x)$.

2. $(x_1 + x_2) + x_2^2 y \neq x_1 y^2$, *where* $y = \exp(x_1)$.

To obtain *formulas* of the extension of the first-order theory of the real closed field, we first combine the atomic formulas with the boolean connectives. In addition, we quantify some or all of the variables in the resulting formulas over the real closed field by universal and existential quantifiers. In a formula φ an occurence of a variable x is said to be *bounded* if it is inside the scope of a quantifier (Qx), where $Q \in \{\exists, \forall\}$. Any occurence that is not bounded is said to be *free*. A *sentence* is a formula without free variables. A formula is called *quantifier-free* if it contains no quantifiers.

Example 2.14 (formula of $\mathcal{T}_{\exp}$)*:*

$$\varphi = (\exists x_1)[(x_1 + x_2) + x_2^2 y \neq x_1 y^2 \vee x_1 - x_2 = 0], \text{ where } y = \exp(x_1).$$

Note that x_1 is bounded in φ, but x_2 is free in φ.

Finally, a *prenex sentence* is a formula of the following form:

$$(Q_1 x_1)(Q_2 x_2)...(Q_n x_n)\psi(x_1, x_2, ...x_n)$$

where ψ is a quantifier-free formula of the extension of the first-order theory of the real closed field as described above, and the $(Q_i x_i)$ are quantifiers.

Examples 2.15 (prenex sentence)*:*

1. $(\forall x_1)(\exists x_2)[(x_1 + x_2) + x_2^2 y \neq x_1 y^2 \vee x_1 - x_2 = 0]$, *where* $y = \exp(x_1)$.

2. $(\exists x)[y - x - 1 = 0 \wedge x > 0]$, *where* $y = \exp(x)$.

2.3 Properties of the Exponential Function and Exponential Polynomials

In order to decide polynomial-exponential problems, the exponential function has to be an *admissible function*. Thus, in this section we introduce admissible functions and show that the exponential function belongs to this class. Moreover, we examine the main properties of exponential polynomials which we will need in the algorithm that isolates the real roots of exponential polynomials, and in the algorithm that decides univariate polynomial-exponential problems.

Definition 2.16 (algebraic, transcendental)
*A number α is said to be **algebraic** if it is a root of a polynomial $f(x) = a_0 + a_1x + ... + a_nx^n$ with rational coefficients.*
*A complex number α (which may be a real number) is said to be **transcendental** if it is not algebraic. Thus, a transcendental number is not a root of any polynomial with algebraic coefficients which is not identically zero.*

Considering the transcendence of the exponential function, F. Lindemann states the following:

Theorem 2.17 (Lindemann's theorem)
If z is a nonzero algebraic number, then e^z is transcendental.

Proof. See [22] □

The following terms, *strongly transcendental function*, and *admissible function*, are introduced by S. McCallum and V. Weispfenning in [19].

Definition 2.18 (strongly transcendental)
*A real or complex analytic function f(x) is called **strongly transcendental** (with exceptional points $\alpha_1, \alpha_2, ..., \alpha_k$) if for all real or complex numbers x excluding the α_i ($x \neq \alpha_i$) not both, x and $f(x)$ are algebraic.*

Corollary 2.19
The complex function e^z is strongly transcendental with exceptional point 0.

Proof. The strongly transcendence of the function is an immidiate consequence of Lindemann's theorem. □

Definition 2.20 (admissible)
*A strongly transcendental real valued function trans(x), with exceptional points $\alpha_1, ..., \alpha_n$, analytic on the whole real line is called an **admissible** function with exceptional points α_i if for every nonzero integral polynomial $p(x, t)$ where $t = trans(x)$, the associated real analytic function $p^*(x)$ has only finitely many zeros in the real line.*

Theorem 2.21
The exponential function $\exp(x)$ *is admissible with the exceptional point* 0.

Proof. By Corollary 2.19 and Remark 2.2, the exponential function $\exp(x)$ is strongly transcendental and a real valued analytic function defined on the whole real line. According to Hardy [14, page 17] a one-variable exponential function which does not vanish identically has only finitely many real

zeros. Hence, for every nonzero $p(x, \exp(x)) \in \mathbb{Z}[x, \exp(x)]$, the associated real analytic function i.e. the exponential polynomial $p^*(x)$ has only finitely many real zeros. □

The following proofs are taken from [19], but are applied here on the exponential function.

Lemma 2.22
Let $p(x, y)$ be an irreducible element of $\mathbb{Z}[x, y]$ of positive degree in y. Let $r(x) = res_y(p, p')$. Then $r(x) \neq 0$ unless $p(x, y) = \pm y$ or $p'(x, y) = 0$.

Proof. Suppose $r(x) = 0$. Then, by Remark 2.7, p and p' have a common factor with positive degree in y. Since p is irreducible it follows that $p|p'$. Hence $p(x, 0)|p'(x, 0)$. But $p'(x, 0)$ is the ordinary derivative (with respect to x) of $p(x, 0)$. Hence we must have $p'(x, 0) = 0$. Therefore y is a factor of $p'(x, y)$. Let $q = \frac{p'}{y}$. Since $p|yq$ and p is irreducible, either $p|y$ or $p|q$. In the former case $p = \pm y$. In the latter case, observe that the degree in y of q is less than that of p, hence $q = 0$. Therefore $p' = 0$ also. See Lemma 4.2 in [19, p.5]. □

Theorem 2.23
Let $p(x, y)$ be an irreducible element of $\mathbb{Z}[x, y]$, where $y = \exp(x)$. Then the only non-simple real zero of $p^*(x)$ is at the exceptional point of $\exp(x)$ i.e. at $x = 0$.

Proof. p^* has no simple zeros if p has degree 0 in y. So suppose that p has positive degree in y. Let $r(x) = res_y(p, p')$ and suppose that $r(x) = 0$. By Lemma , $p(x, y) = \pm y$ or $p'(x, y) = 0$. In the former case, $p^*(x) = \pm \exp(x)$. In the latter case, $p^*(x)$ is a nonzero constant. In both theses cases $p^*(x)$ has no real zeros, hence no non-simple real zeros. So henceforth suppose that $r(x) \neq 0$. Let α be a non-simple zero of $p^*(x)$ which is not an exceptional point of $\exp(x)$. Then $p^*(\alpha) = (p^*)'(\alpha) = 0$. Hence, with $\beta = \exp(\alpha)$, we have

$$p(\alpha, \beta) = p'(\alpha, \beta) = 0.$$

Therefore α is a root of $r(x) \neq 0$, hence algebraic. Now β is a root of the polynomial $p(\alpha, y)$, which is nonzero by the irreducibility of p. Hence β is also algebraic. By Lindemann's theorem (see Theorem 2.17) α must be an exceptional point of $\exp(x)$ (i.e. $x = 0$), contrary to assumption. See Theorem 4.3 in [19, p.6]. □

Theorem 2.24
Let $p(x, y)$ and $q(x, y)$ be relatively prime nonzero elements of $\mathbb{Z}[x, y]$, where

$y = \exp(x)$. *Then the only common real zero of* $p^*(x)$ *and* $q^*(x)$ *is at the exceptional point of* $\exp(x)$ *i.e. at* $x = 0$.

Proof. $p^*(x)$ and $q^*(x)$ have no common zeros if both p and q have degree 0 in y. So suppose that at least one of p and q has positive degree in y. Let α be a common zero of $p^*(x)$ and $q^*(x)$ which is not an exceptional point of $\exp(x)$. Then, with $\beta = \exp(\alpha)$, we have

$$p(\alpha, \beta) = q(\alpha, \beta) = 0.$$

Therefore α is a root of the resultant $res_y(p, q)$, which is a nonzero polynomial since p and q are assumed relatively prime. Hence β is also algebraic, since β is a root of the polynomials $p(\alpha, y)$ and $q(\alpha, y)$, at least one of which is nonzero by relative primality. But this contradicts Lindemann's theorem (see Theorem 2.17), which implies that not both α and β can be algebraic. See Theorem 4.4 in [19, p.6]. □

Corollary 2.25
Let $p(x, y) \in \mathbb{Z}[x, y]$ *be nonzero and squarefree, with* $y = \exp(x)$. *Then the only non-simple real root of* $p^*(x)$ *can be at the exceptional point of exp(x) i.e. at* $x = 0$.

Proof. Since $p(x, y)$ is squarefree it is a product of pairwise relatively prime irreducible elements of $\mathbb{Z}[x, y]$. The corollary follows immediately by application of the two preceding theorems. See Corollary 4.5 in [19, p.6]. □

Chapter 3

Deciding Univariate Polynomial-Exponential Problems

This chapter considers the one-variable case of the original polynomial-exponential problems. Our goal is to outline and describe an algorithm that decides such univariate polynomial-exponential problems.
The chapter is organized as follows:
First, we give the necessary theoretical facts on which the method is based on. Afterwards, we specify the algorithm UPEP, define univariate polynomial-exponential problems and present a first idea of the algorithm UPEP. In the following, we study precisely the subalgorithm ISOL that isolates real roots of exponential polynomials, since it covers the most difficult part of the algorithm UPEP. In addition, we introduce cell indices and sample points. In particular, we discuss the representation of sample points, and the sign-evaluation of exponential polynomials at these sample points. Finally, a full description of the algorithm UPEP is given.

3.1 Theoretical Preliminaries

For the following definitions, we only consider the one-dimensional case of the real numbers. For a more general definition see [2].

Definition 3.1 (region)
*A **region** is a nonempty connected subset of* $\mathbb{R}$.

Definition 3.2 (decomposition)
*A **decomposition** of* $\mathbb{R}$ *is a finite set of disjoint regions whose union is* $\mathbb{R}$.

Definition 3.3 (P^*-invariant)
Let P^* be an exponential polynomial, i.e. $P^* \in \mathbb{Z}[x, \exp(x)]$. Then P^* is invariant on $\mathbb{R}$, if one of the following three conditions holds:

1. *$P^*(s) > 0$ for all s in $\mathbb{R}$. ("P^* has positive sign on $\mathbb{R}$.")*
2. *$P^*(s) = 0$ for all s in $\mathbb{R}$. ("P^* has zero sign on $\mathbb{R}$.")*
3. *$P^*(s) < 0$ for all s in $\mathbb{R}$. ("P^* has negative sign on $\mathbb{R}$.")*

*Let $P^* = \{P_1^*, ..., P_n^*\}$ be a finite subset of $\mathbb{Z}[x, \exp(x)]$. Then $\mathbb{R}$ is P^*-**invariant** if each P_i^* is invariant on $\mathbb{R}$.*

Definition 3.4 (cell, sample point)
*Let D be a decomposition of $\mathbb{R}$. Then the elements of D are called **cells**. We distinguish between 0-cells and 1-cells, where points are 0-cells and open intervals are 1-cells.*
*For each cell of the decomposition D, we define an exact representation of a particular algebraic or transcendental number belonging to that cell. We call this a **sample point** for the cell.*

3.2 Description of the Algorithm and the Decision Problem

Specification 3.5 (algorithm to decide univariate polynomial-exponential problems)

$$v \leftarrow \mathrm{DUPEP}(\varphi)$$

Input: A sentence φ in $\mathcal{L}_{\exp}$ where the terms involve only x and $\exp(x)$.
Output: Truth value v of the input φ over the real numbers.

Remark 3.6
The inputs of the algorithm specified above are sentences of the form:

$$(Qx)\psi(x)$$

where ψ is a quantifier-free formula of $\mathcal{T}_{\exp}$ and (Qx) is a quantifier.

Definition 3.7
*An **univariate polynomial-exponential decision problem** is the problem to find an algorithm that takes inputs as in Specification 3.5 and outputs the truth value of such input formulas over the real numbers.*

In the following, we explain the basic ideas about how the algorithm DUPEP decides univariate polynomial-exponential problems: From the inputs described above, we extract a set P of nonzero integral polynomials in the indeterminates x and y, where $y = \exp(x)$. We denote P^* as the set of exponential polynomials relative to the polynomials in P.
Our goal is to outline a description of a P^*-invariant decomposition of the real line. This means that the decomposition is compatible with the roots of the exponential polynomials in P^*. Therefore, we have to describe an algorithm that isolates the real roots of an exponential polynomial.
From the decomposition of the real line, we obtain a list of cell indices and a list of sample points. In the end, the sample points are used to evaluate the original formula.

3.3 Isolating Real Roots of Exponential Polynomials

In the decision procedure for univariate polynomial-exponential problems, we need to isolate the real roots of a nonzero exponential polynomial. After the specification of the algorithm ISOL, we explain the computation of the two basic steps of the algorithm ISOL. This contains the computation of a real root bound for exponential polynomials and the subalgorithm REFINE. In the end, a full description of the algorithm ISOL shall be given.
Related algorithms for polynomial real root isolation can be found in [7, 9, 16].

3.3.1 Specification of the Algorithm ISOL

Specification 3.8 (algorithm to isolate real roots of exponential polynomials)

$$L \leftarrow \mathrm{ISOL}(p)$$

Input: $p(x, y) \in \mathbb{Z}[x, y]$, a nonzero squarefree polynomial, where $y = \exp(x)$.
Output: L, an isolation list for $p^*(x) = p(x, \exp(x))$.

Definition 3.9 (isolation list)
*An **isolation list** for a real-valued function f(x) defined on the whole real line is a list $L = (I_1, I_2, ..., I_r)$, such that*

(a) $r \geq 0$ is the number of distinct real roots of f.

(b) each I_j is an interval (a_j, b_j), where a_j and b_j are binary rational numbers, that is, rational numbers whose denominators are non-negative powers of 2.

(c) each I_j contains a unique root of f.

(d) $a_1 < b_1 \leq a_2 < b_2 \leq ... \leq a_r < b_r$.

Remark 3.10 *Each I_j of the isolation list $L = (I_1, I_2, ..., I_r)$ is called an **isolating interval**.*

3.3.2 Real Root Bound for Exponential Polynomials

Definition 3.11 (real root bound)
*A **real root bound** for the polynomial $p \in \mathbb{R}[x]$ is a number $\gamma \in \mathbb{R}$ such that if α is any real root of p then $|\alpha| < \gamma$.*

Theorem 3.12 (cauchy bound)
*Let $p(x) = \sum_{j=0}^{d} a_j x^j$ with $a_d \neq 0$ and $d \geq 1$ be an integral polynomial, i.e. $p(x) \in \mathbb{Z}[x]$. Then a **cauchy bound** γ is a real root bound for p and is given by:*

$$\gamma = 1 + \frac{\max\{|a_0|, ..., |a_{n-1}|\}}{|a_n|}.$$

Proof. See [9, pp.259]. □

Lemma 3.13
Let $x > 0$. Let $p(\pm x) = \sum_{j=0}^{d} a_j (\pm x)^j$ with $a_d \neq 0$ and $d \geq 1$ be an integral polynomial, i.e. $p(\pm x) \in \mathbb{Z}[x]$. Let M be a cauchy bound for $p \pm 1$. Then for all values of x, with $x > M$:

$$|a_0 + a_1(\pm x) + ... + a_{d-1}(\pm x)^{d-1}| < |a_d| x^d - 1.$$

Proof. Let $x > 0$. Put $H = \max\{|a_0| + 1, |a_1|, ..., |a_{d-1}|\}$. Since $x > M = 1 + \frac{H}{|a_d|}$, we know that:

$$x > 1 \tag{1}$$

$$\frac{H}{|a_d|(x-1)} < 1 \tag{2}$$

Therefore:

$$\begin{aligned}
|a_0 + a_1(\pm x) + ... + a_{d-1}(\pm x)^{d-1}| &\leq |a_0| + |a_1||x| + ... + |a_{d-1}||x|^{d-1} \\
&\leq |a_0| + |a_1|x + ... + |a_{d-1}|x^{d-1} \\
&= 1 + |a_0| + |a_1|x + ... + |a_{d-1}|x^{d-1} - 1 \\
&\leq H(1 + x + ... + x^{d-1}) - 1 \\
&\overset{(1)}{<} \frac{Hx^d}{x-1} - 1 \\
&= \frac{H}{|a_d|(x-1)}|a_d|x^d - 1 \\
&\overset{(2)}{\leq} |a_d|x^d - 1.
\end{aligned}$$

□

Lemma 3.14
Let $x > 0$. Let $p(\pm x) = \sum_{j=0}^{d} a_j(\pm x)^j$ with $a_d \neq 0$ and $d \geq 1$ be an integral polynomial, i.e. $p(\pm x) \in \mathbb{Z}[x]$. Then we can find $S_1 > 0$ such that

$$|p(\pm x)| > 1 \quad \textit{for all } x > S_1.$$

Proof. Let $x > 0$. First we compute a cauchy bound M for $p \pm 1$. Take $x > M$, then:

$$\begin{aligned}
|p(\pm x)| &= |a_0 + a_1(\pm x) + ... + a_d(\pm x)^d| \\
&= |a_d(\pm x)^d - (-a_0 - a_1(\pm x) - ... - a_{d-1}(\pm x)^{d-1})| \\
&\geq |a_d|| \pm x|^d - |a_0 + a_1(\pm x) + ... + a_{d-1}(\pm x)^{d-1}| \\
&\overset{\text{Lemma 3.13}}{>} |a_d|x^d - |a_d|x^d + 1 \\
&= 1.
\end{aligned}$$

Putting $S_1 = M$, we obtain $S_1 > 0$ such that for $x > S_1$, we have $|p(\pm x)| > 1$.
□

Lemma 3.15
Let $x > 0$. For $0 \leq i \leq n$, let $p_i(\pm x) = \sum_{j=0}^{d_i} a_{ij}(\pm x)^j$ with $a_{id_i} \neq 0$ and $d_i \geq 1$ be integral polynomials, i.e. $p_i(\pm x) \in \mathbb{Z}[x]$. Then we can find $S_2 > 0$ and $k \in \mathbb{N}$ such that for all values of i in the range $v \leq i \leq w$, where $0 \leq v < w \leq n$, we have

$$|p_i(\pm x)| \leq \frac{x^k}{n} \quad \textit{for all } x > S_2.$$

Proof. Let $x > 0$. We first describe the construction of S_2 and k. We could use the following procedure:

1. For each $i = v, ..., w$ {First compute a cauchy bound M_i for p_i. If $M_i < 2n|a_{id_i}|$ then set $m_i \leftarrow 2n|a_{id_i}|$ else set $m_i \leftarrow M_i$}.
2. Set $S_2 \leftarrow \max\{m_v, m_{v+1}, ..., m_w\}$ and set $k \leftarrow 1+\max\{d_v, d_{v+1}, ..., d_w\}$.

By computing S_2 and k as described in the procedure above, we can now prove our assumption for each value of i in the range of $v \leq i < w$, where $0 \leq v < w \leq n$, and for each value of x with $x > S_2$:
Put $M_i = 1 + \frac{H_i}{|a_{id_i}|}$ (i.e. cauchy bound for p_i), where
$H_i = \max\{|a_{i0}|, |a_{i1}|, ..., |a_{i(d_i-1)}|\}$. We know that:

$$\begin{aligned} x > S_2 &\Rightarrow (x > M_i \text{ and } x > 2n|a_{id_i}|) && (1)\\ x > M_i &\Rightarrow x > 1 && (2)\\ x > M_i &\Rightarrow \frac{H_i}{|a_{id_i}|(x-1)} < 1 && (3) \end{aligned}$$

Therefore:

$$\begin{aligned} |p_i(\pm x)| &= |a_{i0} + a_{i1}(\pm x) + ... + a_{id_i}(\pm x)^{d_i}| \\ &\leq |a_{i0}| + |a_{i1}||\pm x| + ... + |a_{id_i}||\pm x|^{d_i} \\ &\leq |a_{i0}| + |a_{i1}|x + ... + |a_{id_i}|x^{d_i} \\ &\leq H_i(1 + x + ... + x^{d_i-1}) + |a_{id_i}|x^{d_i} \\ &\overset{(2)}{\leq} \frac{H_i x^{d_i}}{x-1} + |a_{id_i}|x^{d_i} \\ &= \frac{H_i|a_{id_i}|x^{d_i}}{|a_{id_i}|(x-1)} + |a_{id_i}|x^{d_i} \\ &\overset{(3)}{<} 2|a_{id_i}|x^{d_i} \\ &= 2n|a_{id_i}|\frac{x^{d_i}}{n} \\ &\overset{(1)}{<} \frac{x^{d_i+1}}{n}. \end{aligned}$$

Putting $k = 1 + \max\{d_v, d_{v+1}, ..., d_w\}$, we have $|p_i(x)| < \frac{x^k}{n}$ for all values of i, $v \leq i < w$, where $0 \leq v < w \leq n$, and all values of x with $x > S_2$. □

Lemma 3.16
Let $x > 0$. Let $k > 0$ be a natural number. Then we can find $S_3 > 0$ such that

$$x^k < \exp(\frac{x}{2}) \quad \textit{for all } x > S_3.$$

Proof. Let n be the integer next greater than k, i.e. $n = k+1$. If $x > 0$, $\exp(x) > \frac{x^n}{n!}$. Therefore:

$$\lim_{x\to\infty} \frac{x^k}{\exp\left(\frac{x}{2}\right)} = \lim_{x\to\infty} \frac{x^k 2^{k+1}(k+1)!}{x^{k+1}} = 0.$$

Hence, by numerical computation we can find S_3 such that for $x > S_3$, we have $x^k < \exp(\frac{x}{2})$. □

Theorem 3.17 (positive real root bound for exponential polynomials)
Let $x > 0$, and let $p^*(x) = p(x, \exp(x))$ be an exponential polynomial in the form $\sum_{i=0}^{n} p_i(x)\exp(x)^i$ with $p_i(x) \in \mathbb{Z}[x]$ and $p_n(x) \neq 0$. Then a ***positive real root bound*** *C for $p^*(x)$ can be obtained with the following procedure:*

1. *By applying Lemma 3.14 for $p_n(x)$, we find $C_1 > 0$ such that for all $x > C_1$, $|p_n(x)| > 1$.*
2. *By applying Lemma 3.15 for $p_i(x)$, for all i in the range $0 \leq i < n$ (i.e. set $v \leftarrow 0$ and $w \leftarrow n-1$), we find $C_2 > 0$ and $k \in \mathbb{N}$ such that for all i in the range $0 \leq i < n$ and for all $x > C_2$, $|p_i(x)| \leq \frac{x^k}{n}$.*
3. *By Lemma 3.16, we find $C_3 > 0$ such that for all $x > C_3$, $x^k < \exp(\frac{x}{2})$.*
4. *Set $C \leftarrow \max\{C_1, C_2, C_3\}$.*

Proof. Combining the computations of the procedure described above, we get for $x > C$,

$$\begin{aligned}
|\sum_{i=0}^{n-1} p_i(x)\exp(x)^i| &= |p_0(x) + p_1(x)e^x + ... + p_{n-1}(x)e^{(n-1)x}| \\
&\leq |p_0(x)| + |p_1(x)|e^x + ... + |p_{n-1}(x)|e^{(n-1)x} \\
&\overset{(2.)}{\leq} \frac{x^k}{n} + \frac{x^k}{n}e^x + ... + \frac{x^k}{n}e^{(n-1)x} \\
&= \frac{x^k}{n} \cdot (1 + e^x + ... + e^{(n-1)x}) \\
&\leq \frac{x^k}{n} \cdot (\underbrace{e^{(n-1)x} + e^{(n-1)x} + ... + e^{(n-1)x}}_{n \text{ terms}}) \\
&= \frac{x^k}{n} \cdot n \cdot e^{(n-1)x} \\
&\overset{(3.)}{<} e^{\frac{x}{2}} \cdot e^{(n-1)x} = e^{(n-\frac{1}{2})x} \\
&< e^{nx} \\
&\overset{(1.)}{<} |p_n(x)|e^{nx}.
\end{aligned}$$

Therefore, for $x > C$, we have $p^*(x) \neq 0$. Thus C is a positive real root bound for $p^*(x)$. □

Lemma 3.18
Let $y > 0$, and let $g^*(y) = g(y, \exp(y))$ be an exponential polynomial in the following form:

$$g(y, \exp(y)) = g_0(-y)e^{ny} + g_1(-y)e^{(n-1)y} + ... + g_n(-y).$$

Then a real root bound T for $g^*(y)$ can be obtained with the following procedure:

1. *By applying Lemma 3.14 for $g_0(-y)$, we find $T_1 > 0$ such that for all $y > T_1$, $|g_0(-y)| > 1$.*
2. *By applying Lemma 3.15 for $g_i(-y)$, for all i in the range $0 < i \leq n$ (i.e. set $v \leftarrow 1$ and $w \leftarrow n$), we find $T_2 > 0$ and $k \in \mathbb{N}$ such that for all i in the range $0 < i \leq n$ and for all $y > T_2$, $|g_i(-y)| \leq \frac{y^k}{n}$.*
3. *By Lemma 3.16, we find $T_3 > 0$ such that for all $y > T_3$, $y^k < \exp(\frac{y}{2})$.*
4. *Set $T \leftarrow \max\{T_1, T_2, T_3\}$.*

Proof. Combining the computations of the procedure described above, we get for $y > T$,

$$\begin{aligned}
|g(y, \exp(y))| &= |g_n(-y) + g_{n-1}(-y)e^y + ... + g_1(-y)e^{(n-1)y}| \\
&\leq |g_n(-y)| + |g_{n-1}(-y)|e^y + ... + |g_1(-y)|e^{(n-1)y} \\
&\overset{(2.)}{\leq} \frac{y^k}{n} + \frac{y^k}{n}e^y + ... + \frac{y^k}{n}e^{(n-1)y} \\
&= \frac{y^k}{n} \cdot (1 + e^y + ... + e^{(n-1)y}) \\
&\leq \frac{y^k}{n} \cdot (\underbrace{e^{(n-1)y} + e^{(n-1)y} + ... + e^{(n-1)y}}_{n \text{ terms}}) \\
&= y^k \cdot e^{(n-1)y} \\
&\overset{(3.)}{<} e^{\frac{y}{2}} \cdot e^{(n-1)y} = e^{(n-\frac{1}{2})y} \\
&< e^{ny} \\
&\overset{(1.)}{<} |p_0(-y)|e^{ny}.
\end{aligned}$$

Therefore, for $y > T$, we have $g^*(y) \neq 0$. Thus T is a real root bound for $g^*(y)$. □

Theorem 3.19 (negative real root bound for exponential polynomials)
Let $x < 0$, and let $p^*(x) = p(x, \exp(x))$ be an exponential polynomial in the form $\sum_{i=0}^{n} p_i(x)\exp(x)^i$ with $p_i(x) \in \mathbb{Z}[x]$ and $p_n(x) \neq 0$. Let $y > 0$, and let $g(y, \exp(y)) = p_0(-y)e^{ny} + p_1(-y)e^{(n-1)y} + ... + p_n(-y)$. By the previous Lemma, we compute a real root bound T for $g^*(y)$. Then $-T$ is a ***negative real root bound*** *for $p^*(x)$.*

Proof. Let $x < 0$. Consider $p(x, \exp(x)) = p_0(x)+p_1(x)e^x+...+p_n(x)e^{nx}$.
Put $y = -x$, i.e. $x = -y$. So $y > 0$.
Then $p(-y, \exp(-y)) = p_0(-y) + p_1(-y)e^{-y} + ... + p_n(-y)e^{-ny}$.
By multiplying p with e^{ny}, we obtain:

$$g(y, \exp(y)) := p_0(-y)e^{ny} + p_1(-y)e^{(n-1)y} + ... + p_n(-y).$$

By Lemma 3.18, we can find a positive real root bound $T > 0$ for $g(y, \exp(y))$, with $y > 0$.
We have $y > T$ and therefore:

$$g(y, \exp(y)) \neq 0 \Leftrightarrow e^{ny}(p^*(-y)) \neq 0.$$

So if $x < -T$, then $(-x) > T$ and therefore $p^*(x) \neq 0$.
Hence, -T is a negative real root bound for $p^*(x, \exp(x))$. □

3.3.3 The Subalgorithm REFINE

The following algorithm is the most important part of the algorithm ISOL. After termination, an interval is refined such that it contains no zeros of the exponential polynomial of the input. The algorithm is based on Rolle's theorem and the concept of modulus of continuity (see [4, 5]).

Definition 3.20 (modulus of continuity)
A ***modulus of continuity (moc)*** *for a real-valued function $f(x)$ defined on an interval [a,b] is a function $\delta : \mathbb{R}^+ \to \mathbb{R}^+$ such that*

$$\forall \epsilon > 0, \forall x, y \in [a, b] : |x - y| \leq \delta \Rightarrow |f(x) - f(y)| \leq \epsilon.$$

Theorem 3.21 (linear moc)
Let $f(x)$ be real valued and continuously differentiable on $I = [a, b]$. Let M be a positive number with $M \geq \max_{x \in I} |f'(x)|$. Then a ***linear moc*** *δ for f(x) on I can be obtained by putting $\delta(\epsilon) = \frac{\epsilon}{M}$.*

Proof. Let $\epsilon > 0$, let $x, y \in I$, and suppose that $|x - y| \leq \delta(\epsilon)$. By the mean value theorem, $f(x) - f(y) = f'(c)(x - y)$, for some c between x and y. Therefore

$$|f(x) - f(y)| = |f'(c)||x - y| \leq M|x - y| \leq M\delta(\epsilon) = \frac{M\epsilon}{M} = \epsilon.$$

This completes the proof. See Theorem 4.6 in [19, p.7]. □

Lemma 3.22
Let $f(x) = a_0 + a_1x + ... + a_dx^d$ be an integral polynomial, and let $[a, b]$ be a closed interval. Put $A = \max_{0\leq i\leq d} |a_i|$, and $c = \max\{|a|, |b|\}$, then a bound M for $|f(x)|$ on $[a, b]$ is

$$M = A \cdot (1 + c + c^2 + ... + c^d).$$

Proof. Put $A = \max_{0\leq i\leq d} |a_i|$, and $c = \max\{|a|, |b|\}$. Then,

$$\begin{aligned} |f(x)| &\leq |a_0| + |a_1||x| + |a_2||x|^2 + ... + |a_d||x|^d \\ &\leq A \cdot (1 + |x| + |x|^2 + ... + |x|^d) \\ &\leq A \cdot (1 + c + c^2 + ... + c^d). \end{aligned}$$

□

Theorem 3.23
Let $I = [a, b]$ be a closed interval. Let $p^*(x) = p_0(x) + p_1(x)e^x + ... + p_n(x)e^{nx}$ be an exponential polynomial, and

$$(p^*)'(x) = p_0(x) + (p_1'(x) + p_1(x))e^x + ... + (p_n'(x) + np_n(x))e^{nx}$$

be the derivative of $p^*(x)$. Then, a bound M for $|(p^*)'(x)|$ on $[a, b]$ can be obtained with the following procedure:

1. *By applying Lemma 3.22 for each term $|p_i'(x) + ip_i(x)|$ of $(p^*)'(x)$, we find bounds M_i for each these terms on $[a, b]$.*
2. *Set $M \leftarrow \max_{0\leq i\leq n}\{M_i\} \cdot (n + 1) \cdot e^{nb}$.*

Proof. Let $[a, b]$, $p^*(x)$, and $(p^*)'(x)$ as described above. Since the exponential function is always positive and increasing, it follows:

$$\begin{aligned} |(p^*)'(x)| &\leq |p_0'(x)| + |p_1'(x) + p_1(x)|e^x + ... + |p_n'(x) + np_n(x)|e^{nx} \\ &\leq |p_0'(x)| + |p_1'(x) + p_1(x)|e^b + ... + |p_n'(x) + np_n(x)|e^{nb}. \end{aligned}$$

Each of these terms $|p_i'(x) + ip_i(x)|$ is an integral polynomial, and $[a, b]$ is a closed interval. Therefore, we can find bounds M_i for each of these terms $|p_i'(x) + ip_i(x)|$ on $[a, b]$ by applying Lemma 3.22. Put $M_p = \max_{0 \le i \le n}\{M_i\}$. Therefore:

$$\begin{aligned} |(p^*)'(x)| &\le M_0 + M_1 e^b + ... + M_n e^{nb} \\ &\le M_p \cdot (1 + e^b + ... + e^{nb}) \\ &\le M_p \cdot (\underbrace{e^{nb} + e^{nb} + ... + e^{nb}}_{(n+1)\ terms}) \\ &= M_p \cdot (n+1) \cdot e^{nb}. \end{aligned}$$

□

Theorem 3.24 (Rolle's theorem)
If a function f is continuous on a closed interval [a,b] and differentiable on the open interval (a,b), and f(a) = f(b). Then there is a value c in the open interval (a,b) such that

$$f'(c) = 0.$$

Proof. See [6]. □

Remark 3.25
Rolle's theorem is true for exponential polynomials.

Remark 3.26
If p^* is a squarefree exponential polynomial and $\alpha_1', \alpha_2', ..., \alpha_k'$ are the real roots of its derivative $(p^*)'$ then each of the intervals $(-\infty, \alpha_1'), (\alpha_1', \alpha_2'), ...,$ $(\alpha_{k-1}', \alpha_k'), (\alpha_k', +\infty)$ contains at most one real root of p^*, which occurs if and only if $p^*(\alpha_i')p^*(\alpha_{i+1}') < 0$, for $i = 0, ..., k$.

Algorithm 3.27 (interval refinement)

$$\bar{I} \leftarrow \text{REFINE}(I, p^*, s^*)$$

Inputs: An interval $I = (a, b)$ such that (a, b) is an isolating interval for α', where $\alpha' \neq 0$, and $(p^*)'(\alpha') = 0$. $p^*, s^* \in \mathbb{Z}[x]$ are two nonzero squarefree exponential polynomials such that $s^*(x) = 0$ iff $(p^*)'(x) = 0$. p^*, and s^* are not identical and relatively prime.
Output: An interval $\bar{I} = (\bar{a}, \bar{b})$ such that $\bar{I}$ does not contain any root of $p^*(x)$.

1. **Moc.** Compute a linear moc δ for $p^*(x)$ on the (initial) interval $[a, b]$. [This is done by applying Theorem 3.21, and Theorem 3.23.]

2. **Bisection Process.** Repeatedly bisect $I = (a, b)$ by setting $m \leftarrow \frac{(a+b)}{2}$. **If** $s^*(m) = 0$ **then** {Continue by retaining the subinterval $(a + \frac{(b-a)}{4}, a + \frac{3(b-a)}{4})$ [the interval is centered at $m = \alpha'$]}
else if $s^*(a)s^*(m) < 0$ [retain the subinterval of I which contains α'] **then** {Set $b \leftarrow m$} **else** {Set $a \leftarrow m$}.
Set $\epsilon = \frac{\min\{|p^*(a)|,|p^*(b)|\}}{2}$.
Terminate the bisection process when $\epsilon > 0$ and [eventually] $b - a \leq \delta(\epsilon)$.

Remark 3.28
Take I, p^*, and s^* as defined above in the input of the algorithm REFINE. Then, by Theorem 2.24, p^*, and s^* have no common zeros in $I = (a, b)$.

Theorem 3.29 (termination)
The bisection process does terminate.

Proof. Suppose the bisection process does not terminate. Then the process would define infinite sequences of values for a and b and hence for ϵ. By Corollary 2.25, observe that $p^*(\alpha') \neq 0$, where α' denotes the root of the derivation $(p^*)'$ in the interval $[a, b]$. Throughout the bisection process $\epsilon = \frac{\min(|p^*(a)|,|p^*(b)|)}{2}$ and the sequences of values for a and b tend to the limit α'. Hence the sequence of values of ϵ tends to the limit $\frac{|p^*(\alpha')|}{2}$. By the above observation ϵ is positive and so it is eventually true that

$$\epsilon > 0. \tag{1}$$

Hence the sequences of values of $\delta(\epsilon)$ tends to the limit $\delta(\frac{|p^*(\alpha')|}{2}) > 0$ because $\epsilon \rightarrow \frac{|p^*(\alpha')|}{2}$ and $\delta : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is a linear function. Hence eventually

$$b - a \leq \delta(\epsilon). \tag{2}$$

(2) is true since $\delta(\frac{|p^*(\alpha')|}{2}) > 0$ and the sequences of values of a and b tend to the limit α'. But (1) and (2) contradict the assumption about non-termination and the theorem is proved. □

Theorem 3.30 (correctness)
Upon termination (say after n bisections), the interval $[a_n, b_n]$ contains no zero of $p^*(x)$.

Proof. The algorithm first computes a linear moc δ for $p^*(x)$ on the (initial) interval $[a, b]$ such that $\forall \epsilon > 0, \forall x, y \in [a, b] : |x - y| \leq \delta(\epsilon) \Rightarrow |p^*(x) - p^*(y)| \leq \epsilon$. Suppose after n bisections the process terminates. Hence

$\epsilon > 0$ and $b_n - a_n \leq \delta(\epsilon)$. We have $\alpha' \in [a_n, b_n]$, where α' denotes the root of the derivation $(p^*)'$ in the interval $[a, b]$. Hence $|a_n - \alpha'| \leq b_n - a_n$. We supposed above that: $b_n - a_n \leq \delta(\epsilon)$. Therefore $|a_n - \alpha'| \leq \delta(\epsilon)$. By the definition of moc,

$$|p^*(a_n) - p^*(\alpha')| \leq \epsilon. \tag{1}$$

But $\epsilon = \frac{\min(|p^*(a_n)|,|p^*(b_n)|)}{2}$ upon termination of the process. Therefore

$$0 < \epsilon \leq \frac{|p^*(a_n)|}{2} < |p^*(a_n)|. \tag{2}$$

By 1 and 2:

$$|p^*(\alpha') - p^*(a_n)| < |p^*(a_n)|.$$

In summary $\text{sign}(p^*(\alpha')) = \text{sign}(p^*(a_n)) \neq 0$. Since the input interval is an isolating interval for α', then by Remark 3.26, each interval between the zeros of the (pre-)derivation contains at most one zero of $p^*(x)$. Therefore $[a_n, \alpha']$ contains no zero of $p^*(x)$. Similarly we can proof that $[\alpha, b]$ contains no zero of $p^*(x)$. Combining two of our conclusions the theorem is proved. □

Sign-Evaluation of Exponential Polynomials at Rational Points

In the algorithm REFINE, we have to evaluate the sign of exponential polynomials at rational points. Therefore, we have to define bounds for the exponential function and the exponential polynomials.

Definition 3.31 (bounds for the exponential function)
The exponential function can be defined by a Taylor polynomial, i.e. the first (k+1)-summands, at $x_0 = 0$:

$$\exp(x) = \sum_{i=0}^{k} \frac{x^i}{i!} + R_k$$

where R_k denotes the remainder in Lagrange form:

$$|R_k| \leq \frac{|x|^{k+1}}{(k+1)!} \cdot \exp(\xi)$$

with $0 \leq \xi < |x|$.
Then, for positive x, the exponential function can be bounded as follows:

$$m(x) < \exp(x) < M(x)$$

where

$$m(x) = \sum_{i=0}^{k} \frac{x^i}{i!}$$

to be the ***lower bound*** *for* $\exp(x)$*, and*

$$M(x) = \sum_{i=0}^{k} \frac{x^i}{i!} + R_k$$

to be the ***upper bound*** *for* $\exp(x)$*. We can follow:*

$$\frac{1}{m(x)} > \exp(-x) > \frac{1}{M(x)}.$$

Note that the approximation error minimises for $k \to \infty$*.*

Definition 3.32 (bounds for exponential polynomials)
Let $p^*(x) = p(x, \exp(x))$ *be an exponential polynomial in the form* $\sum_{i=0}^{n} p_i(x) \exp(x)^i$*. Then, for all* $x \in \mathbb{Q}$*, an* ***upper bound*** B *for* $p^*(x)$ *can be obtained with the following procedure:*

1. *Initialize* $B \leftarrow 0$*.*
2. *By Definition 3.31, compute a lower bound* m*, and an upper bound* M *for* $\exp(x)$ *with accuracy* k*, where* k *denotes the degree of the Taylor polynomial.*
3. *For* $i = 0, ..., n$ *{If* $p_i(x) > 0$ *then set* $B \leftarrow B + p_i(x) \cdot M^i$ *else set* $B \leftarrow B + p_i(x) \cdot m^i$ *}.*

By switching m*, and* M *in the third step of the procedure described above, we obtain a* ***lower bound*** *for* $p^*(x)$*.*
Note that the accuracy of the bounds for $p^*(x)$ *depends on the accuracy of the computation of the bounds for* $\exp(x)$*, and therefore on the value* k*.*

Lemma 3.33
Let $r \in \mathbb{Q}$ *be a rational number, and let* $p^*(x) = p(x, \exp(x))$ *be an exponential polynomial in the form* $\sum_{i=0}^{n} p_i(x) \exp(x)^i$*. Then*

$$p^*(r) = \sum_{i=0}^{n} p_i(r) \exp(r)^i = 0 \Leftrightarrow p_i(r) = 0 \text{ for all } i = 0, ..., n.$$

Proof. Let p^*, and r be as defined above. Then, it follows:

$$
\begin{aligned}
& p^*(r) = \sum_{i=0}^{n} p_i(r)e^{ir} = 0 \\
\Leftrightarrow \quad & \underbrace{\underbrace{e^r}_{transcendental\ by\ Theorem\ 2.17} \cdot (\sum_{i=1}^{n} p_i(r)e^{ir})}_{\Rightarrow\ transcendental} = \underbrace{-p_0(r)}_{algebraic} \\
\Leftrightarrow \quad & e^r \cdot (e^r \cdot (\sum_{i=2}^{n} p_i(r)e^{ir}) + p_1(r)) = -p_0(r) \\
& \dots \\
\Leftrightarrow \quad & \underbrace{e^r \cdot (e^r \cdot (\dots \cdot (e^r}_{n\ terms} \cdot (p_n(r)) + p_{n-1}(r)) + \dots) + p_1(r)) + p_0(r) = 0 \\
\Leftrightarrow \quad & p_i(r) = 0 \text{ for all } i = 0, \dots, n
\end{aligned}
$$

□

Remark 3.34 (sign evaluation of exponential polynomials at rational points)
Consider a rational point $r \in \mathbb{Q}$, and an exponential polynomial $p^*(x)$. Then, by Lemma 3.33, we determine if $p^*(r)$ is equal to zero. If $p^*(r) = 0$, it follows that the sign of $p^*(r)$ is equal to zero. In the case, $p^*(r) \neq 0$, we evaluate the sign of $p^*(r)$ as follows. By Definition 3.32, we compute bounds for $p^*(r)$ starting with a default accuracy k, with k denoting the degree of the Taylor polynomial which is used in the computation of the bounds for $\exp(r)$ *(see Definition 3.31). As long as the sign for the lower and upper bound of $p^*(r)$ is not equal, we compute new bounds for $p^*(r)$ by increasing the accuracy k.*

3.3.4 The Algorithm ISOL

Algorithm 3.35 (real root isolation of exponential polynomials)

$$L \leftarrow \mathrm{ISOL}(p)$$

Input and output: As stated in Specification 3.8.

1. **Basis.** Set $(m, n) \leftarrow \mathrm{pdeg}\ p^*(x)$. **If** $(m, n) = (0, 0)$ **then** {Set $L \leftarrow ()$. Return}.
2. **Recursion.** Compute $p'(x, y)$.
   If $n > 0$ **then** set $s(x, y) \leftarrow \mathrm{sqfree}(p'(x, y))$ **else** {Set $\hat{p}(x, y) \leftarrow \frac{p'(x,y)}{y}$. Set $s(x, y) \leftarrow \mathrm{sqfree}(\hat{p}(x, y)\}$. Set $L' \leftarrow \mathrm{ISOL}(s)$.

3. **Real root bound.** Set $\gamma_+ \leftarrow$ a binary rational positive real root bound for $p^*(x)$. Set $\gamma_- \leftarrow$ a binary rational negative real root bound for $p^*(x)$. [This is done by applying Theorem 3.17 and Theorem 3.19, respectively.]

4. **Prepare for induction step.** Let $L' = (I_1, I_2, ..., I_r)$, with $r > 0$ and $I_j = (a_j, b_j)$. [Let $\alpha'_0 = -\infty$ and $\alpha'_{r+1} = \infty$, and for $1 \leq j \leq r$, let α'_j denote the unique zero of $(p^*)'(x)$ in I_j.] **For** i=0,1,...,r, set $J_i = (b_i, a_{i+1})$, with $b_0 = \gamma_-$ and $a_{r+1} = \gamma_+$. Set $L \leftarrow ()$.

5. **Interval refinement. For** j=1,...,r {**If** {$0 \in I_j$ [By Corollary 2.25, $p^*(x)$ has no non-simple zero unless at the exceptional point of $\exp(x)$, which is at $x = 0$.] and $p^*(0) = 0$ [That is iff $p(0,1) = 0$, since $\exp(0) = 1$.]} **then** {Insert I_j into L [By Rolle's theorem (see Theorem 3.24), $x = 0$ is the only root of $p^*(x)$ in I_j, and therefore the interval has not to be refined.]}
else {Set $I_j \leftarrow$ REFINE(I_j, p^*, s^*))}.}.

6. **Completion of induction step. For** i=0,...,r {**If** $p^*(b_i)p^*(a_{i+1}) < 0$ **then** insert J_i into L}. [By step 5, $(\alpha'_i, \alpha'_{i+1})$ contains a zero of $p^*(x)$ iff J_i does, which occurs iff $p^*(b_i)p^*(a_{i+1}) < 0$.]

The algorithm ISOL is based on the derivative sequence of p^* and on recursion on the pseudodegree of p^*. Before we prove its validity, we consider the pseudodegree of p^*.
Recall from Definition 2.4 that *pdeg* p^* is the pair (m, n) of $\mathbb{N}^2 = \mathbb{N} \times \mathbb{N}$, where $m = deg_y\ p(x, y)$ and $n = deg_x\ p(x, 0)$, where $y = \exp(x)$. $\mathbb{N}^2$ has a total order $\leq$, which means that it is reflexive, transitive, antisymmetric and total. Here, we shall use the lexicographic order $\leq$ on $\mathbb{N}^2$.

Example 3.36 *For example, according to [3, p.168] a lexicographical order on* $\mathbb{N}^2$ *is defined as follows:*

$$(k, l) \leq (m, n) \text{ iff } (k < m) \text{ or } (k = m \text{ and } l < n) \text{ or } (k = m \text{ and } l = n).$$

Definition 3.37 (well-order)
A ***well-order*** *is a total order such that every non-empty subset S of* $\mathbb{N}^2$ *has at least one minimal element. This means that there is no infinite descending chain* $(m_1, n_1) > (m_2, n_2) > ...$ *in* $\mathbb{N}^2$.

Theorem 3.38 *The linear order* $\leq$ *on* $\mathbb{N}^2$ *is a well-order.*

Proof. Theorem 4.62 in [3] implies the theorem. □

In conclusion, we can now prove the validity of the algorithm ISOL using the principle of noetherian induction. For an explanation of noetherian induction see [3].

Theorem 3.39 (validity)
Let $P(m,n)$ be the statement "for every valid input p(x,y), with $y = \exp(x)$, and pdeg $p^* = (m,n)$, ISOL returns an isolation list L for $p^*(x) = p(x, \exp(x))$". Then for all $(m,n) \in \mathbb{N}^2$, $P(m,n)$ is true.

Proof. For the proof we use noetherian induction.
***Noetherian induction base**: Show that P(0,0) is true.*
If $(m,n) = (0,0)$, then the algorithm returns an empty list. Since for $(m,n) = (0,0)$, $p^*(x)$ is a nonzero constant, it follows that $p^*(x)$ has no roots. Therefore, the isolation list is an empty list.
***Noetherian induction hypothesis**: For all $(k,l) \in \mathbb{N}^2$ in the range $(k,l) < (m,n)$, $P(k,l)$ is true.*
***Noetherian induction step**: Let $(m,n) > (0,0)$. Show that the noetherian induction hypothesis implies that $P(m,n)$ is true.*
Let p be an integral polynomial in x and y. Write p in the form

$$p(x,y) = p_0(x) + p_1(x)y + p_2(x)y^2 + ... + p_n(x)y^n.$$

Claim 1: pdeg $s^*(x) <$ pdeg $p^*(x)$.
Let $(m,n) > (0,0)$, where $(m,n) =$ pdeg p^*. In step 2 of the algorithm, we have to consider two cases for the construction of $s(x,y)$: either $n > 0$ or $n = 0$.
In the former case $(n > 0)$, the derivative of p is

$$p'(x,y) = p_0'(x)+(p_1'(x)+p_1(x))y+(p_2'(x)+2p_2(x))y^2+...+(p_n'(x)+np_n(x))y^n.$$

Then, we have $\deg_y p'(x,y) = \deg_y p(x,y)$, and $\deg_x p'(x,0) < \deg_x p(x,0)$. So

$$\text{pdeg } (p')^* < (m,n).$$

Since $s(x,y)$ is a divisor of $p'(x,y)$, the degree of $s(x,y)$ in either x or y is less than or equal to the degree of $p'(x,y)$ in x or y, respectively. Hence

$$\text{pdeg } s^*(x) < \text{pdeg } (p')^*(x) < (m,n).$$

In the latter case $(n = 0)$, we have $p(x,y) = p_0(x) + p_1(x)y + ... + p_m(x)y^m$, where $p_0(x)$ is a constant. Then, the derivative of p is

$$p'(x,y) = (p_1'(x) + p_1(x))y + (p_2'(x) + 2p_2(x))y^2 + ... + (p_n'(x) + np_n(x))y^n.$$

Therefore, pdeg $(p')^*(x)$ = pdeg $p^*(x)$. But for this case in step 2 of the algorithm, we divide $p'(x,y)$ with y. Then, we have $\deg_y \frac{p'(x,y)}{y} = (\deg_y p'(x,y)) - 1$. Since $s(x,y)$ is a divisor of $\frac{p'(x,y)}{y}$, the degree of $s(x,y)$ in either x or y is less than or equal to the degree of $\frac{p'(x,y)}{y}$ in x or y, respectively. Hence

$$\text{pdeg } s^*(x) < \text{pdeg } p^*(x).$$

In both these cases we have pdeg s^* < pdeg p^* and therefore Claim 1 is proved.

Definition: Let $f(x,y)$ be an integral polynomial. Then $V(f(x,y))$ denotes the *real variety* of f, i.e. the set

$$V(f(x,y)) = \{(x,y) \in \mathbb{R}^2 | f(x,y) = 0\}$$

of all zeros of the polynomial f.

Claim 2: $V(s^*(x)) = V((p')^*(x))$.
We consider again the two cases of $s(x,y)$ in step 2 of the algorithm:
If $n > 0$, then

$$s(x,y) = \text{sqfree}(p'(x,y)).$$

By Definition 2.9, we know that the squarefree part of a polynomial is the product of the distinct non-associated irreducible factors of $p(x,y)$. Therefore, the real variety of a polynomial is equal to the real variety of its squarefree part. Putting $y = \exp(x)$, it follows that $V(s^*(x)) = V((p')^*(x))$.
If $n = 0$, then

$$s(x,y) = \text{sqfree}(\frac{p'(x,y)}{y}).$$

We know that $p(x,y) = p_0(x) + p_1(x)y + ... + p_m(x)y^m$, where $p_0(x)$ is a nonzero constant. Then,

$$\begin{aligned} p'(x,y) &= (p_1'(x) + p_1(x))y + (p_2'(x) + 2p_2(x))y^2 + ... + \\ &\quad (p_m'(x) + mp_m(x))y^m \\ &= y \cdot ((p_1'(x) + p_1(x)) + (p_2'(x) + 2p_2(x))y + ... + \\ &\quad (p_m'(x) + mp_m(x))y^{m-1}) \end{aligned}$$

Therefore, $p'(x,y) = 0$ if and only if $y = 0$ or $(p_1'(x) + p_1(x)) + (p_2'(x) + 2p_2(x))y + ... + (p_m'(x) + mp_m(x))y^{m-1} = 0$. Putting $y = \exp(x)$, we obtain the exponential polynomial $(p')^*(x) = p'(x, \exp(x))$. Since $y = \exp(x)$ has no roots on the whole real line, we have $p'(x,y) = 0$ if and only $\frac{p'(x,y)}{y} = 0$.

By the above remark about the real variety of the squarefree part of a polynomial, it follows that $V(s^*(x)) = V((p'(x))^*)$.
Hence, in both these cases we have $V(s^*(x)) = V((p'(x))^*)$. This proves Claim 2.

By Claim 1, and by noetherian induction hypothesis, L' is an isolation list for $s^*(x) = s(x, y)$, with $y = \exp(x)$. Hence, Claim 2 implies that L' is also an isolation list for $(p'(x))^*$.

Let $L' = (I_1, I_2, ...I_r)$, and let J_i, for $0 \leq i \leq r$, be as described in step 4 of the algorithm. We know that:

- By Remark 3.26, each interval $[\alpha'_i, \alpha'_{i+1}]$ contains at most one zero of $p^*(x)$. Hence each complementary interval $J_i = (b_i, a_{i+1})$, a subinterval of $[\alpha'_i, \alpha'_{i+1}]$, contains at most one such zero.

- By Corollary 2.25, $p^*(\alpha'_i) = 0$ if and only if α'_i is an exceptional point of $\exp(x)$, which is if and only if $\alpha'_i = 0$.

Therefore, we have to examine two cases: either α_i is an exceptional point or not. If α'_i is an exceptional point of $\exp(x)$, then we have to verify if $p^*(\alpha'_i) = 0$, which is if and only if $p^*(0) = p(0, 1) = 0$. This is done by step 5 of the algorithm. Otherwise, we have to ensure that after suitable refinements of the intervals I_j, for each $j = 0, 1, ..., r$, no $[a_j, b_j]$ contains a zero of $p^*(x)$. This is done by applying algorithm REFINE. Its termination, and correctness is stated by Theorem 3.29, and Theorem 3.30, respectively.
By step 5, $(\alpha'_i, \alpha'_{i+1})$ contains a zero of $p^*(x)$ if and only if J_i does, which occurs if and only if $p^*(b_i)p^*(a_{i+1}) < 0$. Hence, step 6 of the algorithm completes the isolation list L for p^*, and therefore the noetherian induction step is proved.
We can conclude that for all $(m, n) \in \mathbb{N}^2$, $P(m, n)$ is true. □

3.4 Cell Indices and Sample Points

In the previous section, we have explained how we can obtain the real zeros of exponential polynomials. These zeros will determine a decomposition of the real line. Assuming that we have n zeros, then the decomposition splits into n 0-cells (the zeros) and $n + 1$ 1-cells (the open intervals between the zeros).

3.4.1 Cell Indices

Each cell of the decomposition has to be labeled with an index. These *cell indices* are defined as follows: the index of the leftmost 1-cell (the 1-cell with left endpoint $-\infty$) is **(1)**, the index of the 0-cell (if any) immediately to its right is **(2)**, then the index of the 1-cell to the right of that 0-cell (if any) is **(3)**, etc.

3.4.2 Sample Points

Recall from Definition 3.4 that a sample point for a cell is an exact representation of a particular algebraic or non-algebraic (i.e. transcendental) number belonging to that cell.
The following definition is taken from [8].

Definition 3.40 (representation of algebraic numbers)
An algebraic number α is represented by its minimal polynomial $M(x)$ and by an isolating interval $(r, s]$ such that $r < \alpha \leq s$, $r, s \in \mathbb{Q}$ and $\alpha \notin \mathbb{Q}$ or by the point interval $[r, -r]$ if $\alpha = r$. The isolating interval indicates which root of $M(x)$ α is.

In the decomposition we have 0-cells and 1-cells. For each of them we must define sample points.
As *sample points for the 1-cells* we use appropriately chosen rational endpoints from the isolating intervals obtained as described in section 3.3.4.
As *sample points for the 0-cells* we have to consider two cases: either the 0-cell is an algebraic number or not. If the 0-cell c is an algebraic number we use the representation of algebraic numbers as described in Definition 3.40. Otherwise, we represent c by the element $p(x, y) \in \mathbb{Z}[x, y]$, where $y = \exp(x)$, for which $p^*(c) = 0$ and an isolating interval for c.

3.5 Sign-Evaluation of Sample Points

The following discussion about the sign-evaluation of sample points is an important step of the algorithm that decides univariate polynomial-exponential problems. The task is to evaluate whether an exponential polynomial has a positive, zero or negative sign at a particular sample point. As described in the previous section, sample points for the 1-cells are rational, and sample points for the 0-cells are either algebraic or non-algebraic (i.e. transcendental).
The evaluation of exponential polynomials at **rational points** is described

in Remark 3.34.
In the **non-algebraic** case, we evaluate the sign of an exponential polynomial at a non-algebraic number.
Let $q^*(x) = \sum_{i=0}^{m} q_i(x)e^{ix}$ be the exponential polynomial, and let r be the given non-algebraic number. As described in section 3.4.2, r is represented by a polynomial $p(x, y) \in \mathbb{Z}[x, y]$, where $y = \exp(x)$, for which $p^*(r) = 0$, and an isolating interval $I = (a, b)$ for r.
In order to evaluate the sign, we have to distinguish between the following cases: In the first case, $q(x, y)$ and $p(x, y)$ are identical, and it follows immediately that the sign of $q^*(r)$ is equal to zero. In the second case, $q(x, y)$ and $p(x, y)$ are relatively prime, and then Theorem 2.24 implies that $q^*(r) \neq 0$. Therefore, we can determine the sign of $q^*(r)$ by using the algorithm REFINE. First, we refine I about r such that, after refinement, I contains no zero of $q^*(x)$. Then, by evaluating q^* at the left or right border, we can determine the sign of $q^*(r)$.
In the **algebraic** case, we have to evaluate the sign of exponential polynomials at an algebraic number. First, we determine whether the exponential polynomial at this sample point is equal to zero. This can be done by applying a similar procedure as the one in the rational case (see Lemma 3.33). If the exponential polynomial at the sample point is equal to zero, it follows immediately that the sign is also equal to zero. In the other case, we determine the sign by using the algorithm REFINE in analogy to the second case in the sign-evaluation of non-algebraic numbers.

3.6 The Algorithm DUPEP

Algorithm 3.41 (deciding univariate polynomial-exponential problems)

$$\upsilon \leftarrow \mathrm{DUPEP}(\varphi)$$

Input and output: As stated in Specification 3.5.

1. **Extraction.** Let φ be of the following form

   $$(Qx)\psi(x)$$

   where ψ is a quantifier-free formula of $\mathcal{T}_{\exp}$ and (Qx) is a quantifier. Then extract a list $P := \{p_1(x, y), p_2(x, y), ..., p_n(x, y)\}$ from $\psi(x)$, where $p_i(x, y) \in \mathbb{Z}[x, y]$, and $y = \exp(x)$.

2. **Contents and primitive parts.** Compute the set $cont_y(P)$ of contents (w.r.t. y) of the elements of P and the set $pp_y(P)$ of primitive parts (w.r.t. y) of elements of P of positive degree in y. [With $y = \exp(x)$.]

3. **Squarefree bases.** Compute squarefree bases K and Q of $cont_y(P)$ and $pp_y(P)$, respectively.

4. **Root isolation.** Apply algorithm ISOL to each polynomial $q(x, y)$ in Q and to each polynomial $c(x)$ in K, individually.

5. **Isolation list.** By their relative primality, for any pair of distinct elements p and q of $K \cup Q$, p^* and q^* have common real zeros only at $x = 0$ (see Corollary 2.25). Hence by refining the isolating intervals for the zeros of all the $p^*(x) \in K \cup Q^*$ we obtain an isolation list for the product $\hat{q}(x)$ of all $p^*(x)$.

6. **Sample points.** Use the isolation list for $\hat{q}(x)$ to construct sample points for all of the cells of the decomposition of the real line determined by the zeros of $\hat{q}(x)$. [For the representation of the sample points see section 3.4.2.]

7. **Evaluation.** Use the sample points to decide the original function $(Qx)\psi(x)$. [This can be done as follows: Compute a squarefree decomposition of each polynomial in the formula φ. By evaluating the sign of all the squarefree factors at the sample points as described in section 3.5, the sign of each polynomial in φ can be determined. Then the truth value of the original formula φ can be decided.]

Theorem 3.42 (validity)
The Algorithm 3.41 decides a prenex sentence φ in $\mathcal{L}_{\exp}$, where the terms involve only x and exp(x), over the real numbers.

Proof. Since the algorithm ISOL for the root isolation terminates by Theorem 3.39, the termination of the algorithm DUPEP follows immediately. The correcteness is proved as follows. First, from the quantifier-free part ψ of the input formula φ a list

$$P := \{p_1(x, y), p_2(x, y), ..., p_n(x, y)\}$$

with $p_i(x, y) \in \mathbb{Z}[x, y]$, and $y = \exp(x)$ is extracted. Now, we want to construct a cell-decomposition of the real line determined by the zeros of the product of all $p_i^*(x) \in P^*$.

By Definition 2.9, we know that the squarefree part of a polynomial is the product of the distinct non-associated irreducible factors of the polynomial. Therefore, the real variety of a polynomial is equal to the real variety of its squarefree part. It follows that we have only to compute the roots of the squarefree part of $p_i(x, y)$.
Further, we know by Definition 2.9 that the squarefree part is the product of a squarefree decomposition of its content (w.r.t. y) and a squarefree decomposition of its primitive part (w.r.t. y). Therefore, we have only to compute the roots of polynomials of the squarefree decomposition of the content and the polynomials of the squarefree decomposition of the primitive part.
This is now used in the steps of the algorithm. First, we compute squarefree bases K and Q of $cont_y(P)$, and $pp_y(P)$, respectively. Then, by applying Algorithm 3.35, we isolate the roots of each polynomial in Q and K, individually. Theorem 3.39 states the validity of Algorithm 3.35.
By Corollary 2.25, we know that $p^*, q^* \in K \cup Q^*$ have common zeros only at $x = 0$. Hence, by refining the isolation intervals for the zeros of all the $p^*(x) \in K \cup Q^*$ we obtain an isolation list for the product $\hat{q}(x)$ of all $p^*(x)$. Now, the isolation list for $\hat{q}(x)$ is used to construct a cell-decomposition of the real line.
As the decomposition is P^*-invariante, the sample points for the cells can be used to decide the original sentence φ. □

Chapter 4

Deciding Polynomial-Exponential Problems

In chapter 3, we have described an algorithm that decides univariate polynomial-exponential problems. In the following, our goal is to outline an extention to this procedure, i.e. an algorithm that decides polynomial-exponential problems in general.
In the first section, we specify the algorithm, state the problem and give a first idea of how the algorithm works. The first phase of our algorithm involves a quantifier elimination by cylindrical algebraic decomposition which we describe in section 4.2. In the end, we give a full description of our algorithm for deciding polynomial-exponential problems.

4.1 Description of the Algorithm and the Decision Problem

Specification 4.1 (algorithm to decide polynomial-exponential problems)

$$v \leftarrow \mathrm{DPEP}(\varphi)$$

Input: A prenex sentence φ in $\mathcal{L}_{\exp}$.
Output: Truth value v of the input φ over the real numbers.

Definition 4.2
*A **polynomial-exponential decision problem** is the problem to find an algorithm that decides whether the input as defined in Specification 4.1 is true or false over the real numbers.*

In order to get an idea about how the algorithm DPEP decides polynomial-exponential problems, we give a brief description of the two basic phases of the algorithm.

(1) **First Phase.** We apply a quantifier-elimination algorithm for formulas of the form:

$$(Q_2x_2)(Q_3x_3)...(Q_nx_n)\psi(x_1, x_2, ..., x_n)$$

where ψ is a quantifier-free formula of $\mathcal{T}_{\exp}$ and (Q_ix_i) are quantifiers. We know that ψ involves polynomials in x_i and $\exp(x_1)$. As the variable x_1 is not quantified, the quantifier elimination algorithm can proceed without any precautions concerning the exponential function. The output is a quantifier-free formula $\psi_1(x_1)$.

(2) **Second Phase.** We combine the quantifier corresponding to the variable x_1 of the original polynomial-exponential problem with the output of the QE algorithm $\psi_1(x_1)$. Thus we obtain an univariate polynomial-exponential decision problem. Therefore, we have to decide prenex sentences of the form:

$$(Q_1x_1)\psi_1(x_1)$$

where $\psi_1(x)$ is a quantifier-free formula of $\mathcal{T}_{\exp}$ and (Q_1x_1) is a quantifier.

4.2 Quantifier Elimination by Cylindrical Algebraic Decomposition

As we have seen in the previous section, we apply a *quantifier elimination by cylindrical algebraic decomposition* (QE by CAD) in the first phase of the main algorithm DPEP. In this section, we shortly revise the main steps of this approach. For a more detailed explanation, we refer to [2, 10, 21].

A *CAD algorithm* is an algorithm which solves the following problem: The input is a set $\mathcal{A}$ of integral polynomials in r variables. Then the output is a cylindrical algebraic decomposition (CAD) of $\mathbb{R}^r$ into cells of which each input polynomial is sign-invariant.
This algorithm consists of three phases: the projection phase, the base phase, and the extension phase.
In the **projection phase**, a projection operator is successively used to define

sets of polynomials in $r-1$, $r-2$,...,1 variables. In particular, the projection of a set of polynomials in r variables is a set of polynomials in $r-1$ variables. Since the zeros of the polynomials form the boundaries of the cells, and the cells have to be arranged cylindrically, the projection operator must ensure the *delineability* of the roots (see [10]).
In the **base phase**, we have to describe a CAD of $\mathbb{R}^1$. By the last projection, all polynomials are univariate. Using the zeros of these univariate polynomials, we obtain a CAD of the one-dimensional space in which the cells are all sign-invariant.
In the **extension phase**, we successively extend the CAD of $\mathbb{R}^1$ to a CAD of $\mathbb{R}^2$, CAD of $\mathbb{R}^2$ to CAD of $\mathbb{R}^3$,...,CAD of $\mathbb{R}^{r-1}$ to CAD of $\mathbb{R}^r$. For example, for an extension of a CAD of $\mathbb{R}^{r-1}$ to a CAD of $\mathbb{R}^r$, we construct a stack over each cell in the CAD of $\mathbb{R}^{r-1}$, using the set of r-variate polynomials which were produced by the projection. The properties of the projection operation, and the underlying theory, ensure a CAD of $\mathbb{R}^r$ into cells of which each input polynomial is sign-invariant.

In the next point, we use the CAD algorithm for a quantifier elimination. A *QE algorithm by CAD* proceeds as follows: The input is a prenex formula. First of all, the QE algorithm applies the CAD algorithm to the set of polynomials occuring in this input formula. In the next step, the output of the CAD algorithm is used to determine which cells of the CAD satisfy the unquantified part of the input formula. Hence, the output of the QE is a quantifier-free formula satisfied only by the true cells.

In the following, we assume that we have an algorithm QECAD that eliminates the quantifiers using CAD.

Example 4.3 *Suppose that we have the following prenex formula:*

$$(\exists u)(\exists v)[x = u \cdot v \wedge y = u^2 \wedge z = v^2].$$

Then, we want to find a quantifier-free formula for this prenex formula. This is done by applying the algorithm QECAD. We obtain the following quantifier-free formula:

$$x^2 - yz = 0 \wedge y \geq 0 \wedge z \geq 0.$$

4.3 The Algorithm DPEP

Algorithm 4.4 (deciding polynomial-exponential problems)

$$\upsilon \leftarrow \mathrm{DPEP}(\varphi)$$

Input and output: As stated in Specification 4.1.

1. **Preparation.** Let the input sentence φ be of the following form

$$(Q_1x_1)(Q_2x_2)...(Q_nx_n)\psi(x_1, x_2, ..., x_n)$$

where ψ is a quantifier-free formula of $\mathcal{T}_{\exp}$ and (Q_ix_i) are quantifiers. [The exponential function occurs only in x_1.] Then eliminate (Q_1x_1) from φ by setting

$$\varphi' \leftarrow (Q_2x_2)(Q_3x_3)...(Q_nx_n)\psi(x_1, x_2, ..., x_n).$$

[Then x_1 is the only free variable in φ'.]

2. **QE by CAD.** If φ' is not quantifier-free then find a quantifier-free formula ψ_1 with x_1 as the only free variable in ψ_1 such that

$$\mathbb{R} \models \varphi' \longleftrightarrow \psi_1.$$

[This can be done by applying a QE by CAD.]

3. **UPEP.** Combine the quantifier (Q_1x_1) of the input sentence φ with the quantifier-free formula ψ_1. This is done by setting

$$\varphi'' \leftarrow (Q_1x_1)\psi_1(x_1).$$

[φ'' is an univariate polynomial-exponential problem.]
Set

$$\upsilon \leftarrow \text{UPEP}(\varphi'').$$

[Decide φ'' by applying Algorithm 3.41.]

Theorem 4.5 (validity)
The Algorithm 4.4 decides a prenex sentence φ in $\mathcal{L}_{\exp}$ over the real numbers.

Proof. The termination is clear. The correctness follows by putting the results of the previous section and the previous chapter together. Remember that the exponential function only occurs in the variable x_1. Let φ be of the form

$$(Q_1x_1)(Q_2x_2)...(Q_nx_n)\psi(x_1, x_2, ..., x_n)$$

where ψ is a quantifier-free formula of $\mathcal{T}_{\exp}$ and (Q_ix_i) are quantifiers.
First of all, we compute a formula

$$\varphi' = (Q_2x_2)(Q_3x_3)...(Q_nx_n)\psi(x_1, x_2, ..., x_n)$$

by eliminating the quantifier (Q_1x_1) of the input formula φ. Note that x_1 is the only free variable in φ'. By an algorithm that eliminates the quantifiers by CAD, we obtain a quantifier-free formula ψ_1. Since no new free variables were introduce, the only free variable which has remained in ψ_1 is x_1. In the next step of the algorithm, φ'' is obtained by putting the quantifier (Q_1x_1) and the quantifier-free formula ψ_1 together. Then $varphi''$ is of the following form

$$(Q_1x_1)\psi_1(x_1).$$

By Definition 3.7, φ'' is an univariate polynomial-exponential problem. Since the Algorithm 3.41 decides the truth-value of φ'', we obtain the truth value of the original input sentence. Therefore, the proof is complete. □

Chapter 5

Implementation

5.1 REDUCE and REDLOG

REDUCE is a computer algebra system which is used in a wide variety of areas. It was developed by Anthony C. Hearn and its first release was published in 1968.
Basically, REDUCE has three modes. The first one is called *algebraic*, in which the user can interact with the system [15]. The second mode is called *symbolic*, and consists of a syntactic variant of Lisp called RLISP [20]. Finally, this Lisp dialect is translated to *Standard Lisp* [13].
The current release is the version 3.8 of April 15, 2004.

The computer logic system REDLOG forms an extention to the computer algebra system REDUCE, deriving its name from the term REDuce LOGic system. This system implements symbolic algorithms on first-order formulas over a fixed language and theory. For the thesis at hand, it is sufficient to focus on the language of ordered rings and the theory of real closed fields. For more information see [11, 12].

5.2 Approximation of the Bounds for the Exponential Function

In the decision procedure, we have to compute lower and upper bounds for the exponential function. As described in Definition 3.31, for positive x, we use the following bounds for $\exp(x)$:

$$\sum_{i=0}^{k} \frac{x^i}{i!} < \exp(x) < \sum_{i=0}^{k} \frac{x^i}{i!} + R_k$$

with $R_k \leq \frac{x^{k+1}}{(k+1)!}\exp(\xi)$, and where $0 \leq \xi < x$.
The approach for the computation of the first (k+1)-summands of the Taylor polynomial is taken from [24], and uses the concept of repeated squaring. We use the following equation:

$$e^x = e^{s\ln 2 + r} = 2^s e^r$$

where $x = s\ln 2 + r$.
Therefore, we only have to approximate the exponential function on the interval $I = [0, \ln 2]$ by the Taylor serie. For the approximation for all $x \in I$, we use the first k summands of the Taylor serie. The accuracy k is 20 by default, but can be controlled by the user.

Algorithm 5.1 (exponential function approximation)

$$\text{expApprox}(x, k)$$

Input: Integers x, and k (i.e. the accuracy of exp).
Output: A rational number that is $\exp(x)$.

If $x < \ln 2$ **then** $\sum_{i=0}^{k} \frac{x^i}{i!}$
else $2^s \cdot$(expApprox(r)) **where** $(s, r) =$ findsr x.

Algorithm 5.2 (find s, and r for algorithm expApprox)

$$\text{findsr}(x)$$

Input: Integer x.
Output: Integers s,and r such that $x = s\ln 2 + r$.

Return findsr_helper(x,0).

where findsr_helper(x,s)
If $x \leq \ln(2)$ **then** (s, x)
else findsr_helper($x - \ln(2)$,$s + 1$).

For the upper bound, we have to compute the remainder R_k. We approximate as follows:

$$R_k \leq \frac{x^{k+1}}{(k+1)!} 3^{\lceil x \rceil}.$$

For negative x, we use the equation that we defined in Definition 3.31.

5.3 Documentation

The user can start the decision procedure with the following command:

`rldpep` φ [*accuracy_of_exp*] Function

Deciding polynomial-exponential problems. The variable φ is a prenex sentence in $\mathcal{L}_{\text{exp}}$ as described in Section 2.2. The *accuracy_of_exp* is 20 by default. This function returns the truth-value of φ, if the method succeeds. Otherwise, the user has to restart the decision procedure with an increased value for the *accuracy_of_exp*.

`rldpepverbose` Switch

Deciding exponential-polynomial problems verbose. By default this switch is on. It protocols the computations of the main steps of the decision procedure.

`rldpepiverbose` Switch

Deciding exponential-polynomials isol verbose. By default this switch is off. In addition to the rldpepverbose, it protocols the computations of the steps of the real root isolation of polynomials, i.e. the steps of the algorithm ISOL.

Chapter 6

Conclusion

In this thesis, we have defined a method for deciding polynomial-exponential problems for the real numbers. The approach for the decision procedure is based on S. McCallum and V. Weispfenning's paper "Deciding Polynomial-Transcendental Problems" [19].
The developed decision procedure uses only elementary analysis, quantifier-elimination by CAD, and Lindemann's theorem on the transcendence of $\exp(x)$ for real algebraic x.
The method described here is implemented in the computer logic system REDLOG. As known, the algorithm is called with prenex sentences of the extension of the first-order theory of the real closed field where terms are considered integral polynomials in $x_1, ..., x_n$, and $\exp(x_1)$. Furthermore, we can use the procedure to decide prenex sentences where terms are polynomials in $x_1, ..., x_n$, and $\cosh(x_1)$, $\sinh(x_1)$, and $\tanh(x_1)$. By using the following equations:

$$\begin{aligned}
\cosh(x) &= \frac{e^x + e^{-x}}{2} = \frac{e^x + \frac{1}{e^x}}{2} \\
\sinh(x) &= \frac{e^x - e^{-x}}{2} = \frac{e^x - \frac{1}{e^x}}{2} \\
\tanh(x) &= \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}
\end{aligned}$$

we can transform the atomic formulas such that the terms are again polynomials in $x_1, ..., x_n$, and $\exp(x_1)$, i.e. terms of $\mathcal{T}_{\exp}$. For example, we compare the hyperbolic cosine $\cosh(x)$ with polynomials $p(x)$. By substituting $\cosh(x)$

for the term $\frac{e^x+\frac{1}{e^x}}{2}$, follows:

$$\begin{aligned} & (\exists x)[\cosh(x) < p(x)] \\ \Leftrightarrow\ & (\exists x)[\exp(x) + \exp(-x) < 2 \cdot p(x)] \\ \Leftrightarrow\ & (\exists x)[(\exp(x))^2 + 1 < 2 \cdot p(x) \cdot \exp(x)] \\ \Leftrightarrow\ & (\exists x)[y^2 + 1 < 2 \cdot p(x) \cdot y], \text{ where } y = \exp(x_1). \end{aligned}$$

This shows that the decision procedure also works for problems where terms are polynomials in hyperbolic functions. More examples are given in the Appendix A.4.
In order to show another generalization of polynomial-exponential problems, we consider prenex sentences where terms are polynomials in x, and $\gamma(x)$, with $\gamma(x) = \exp(-x^2)$ (the Gauss curve).
By introducing a new variable z, we are able to build prenex sentences with terms that are polynomials in z, x, and $\exp(z)$, and where we added the equation $z + x^2 = 0$. Note that the equivalence between the input sentence and the transformed sentence is valid in $\mathcal{T}_{\exp}$. In the transformed sentence the exponential function occurs only with argument z. With the equation $z + x^2 = 0$, it is assured that $\exp(z) = \exp(-x^2)$ defines the Gauss curve. Examples are given in the Appendix A.5.

Appendix A

Examples

All examples are tested on a Pentium 4 (2 Ghz, 128 MB Heap Size).

A.1 Complete Example

In the following, we reconstruct the inner functionality of the algorithm **DPEP** for a particular example. We consider the following polynomial-exponential problem:

$$\varphi := (\exists x_1)(\exists x_1)[y - x_2^2 = 0 \wedge x_1 - x_2 = 0], \text{ where } y = \exp(x_1).$$

DPEP is called with input φ. First, we set

$$\varphi' := (\exists x_1)[y - x_2^2 = 0 \wedge x_1 - x_2 = 0], \text{ where } y = \exp(x_1).$$

By applying a QE by CAD for φ', we obtain the following quantifier-free formula

$$\psi_1 := [y - x_1^2 = 0 \wedge y \geq 0], \text{ where } y = \exp(x_1).$$

In the next step of the algorithm, we combine $(\exists x_1)$ with ψ_1, and therefore, we have to decide the following univariate polynomial-exponential problem:

$$\varphi'' := (\exists x_1)\psi_1(x_1).$$

This is done by applying algorithm **DUPEP** with input φ''. We follow the steps of the algorithm. Let

$$P := \{y - x_1^2, y\}, \text{ where } y = \exp(x_1).$$

Then, we compute from P the following sets:

$$\begin{aligned} cont_y(P) &= \{1, 1\} \\ pp_y(P) &= \{y - x^2, y\}, \text{ where } y = \exp(x_1). \end{aligned}$$

We obtain the following squarefree bases for $cont_y(P)$, and $pp_y(P)$ which are denoted with K and Q, respectively:

$$K = \{1\}, Q = \{y - x^2, y\}, \text{ where } y = \exp(x_1).$$

We now have to isolate the roots of each polynomial in K and Q. That means we successively call algorithm **ISOL** with each polynomial. Since the polynomial in K is a constant, its pseudo-degree is the pair $(0, 0)$. Therefore, the isolation list is empty.
Next, ISOL is called with $p(x_1, y) = y - x_1^2$, where $y = \exp(x_1)$. See Example A.1 for a detailed description of ISOL(p), with $p(x, y) = y - x^2$, and $y = \exp(x)$.
After this, ISOL is called with $p(x_1, y) = y$, where $y = \exp(x_1)$. ISOL returns an empty isolation list, since p has no roots. Resumed, we have the following isolation list:

$$\begin{aligned} L(1) &= () \\ L(\exp(x_1) - x_1^2) &= (-3, 17) \\ L(\exp(x_1)) &= () \end{aligned}$$

We now refine the isolating intervals for the zeros of $1, \exp(x_1) - x_1^2, \exp(x_1) \in K \cup Q^*$ in such a way that we obtain an isolation list for their product $\hat{q}(x_1) = (\exp(x_1))^2 - x_1^2 \exp(x_1)$. Since $L(1)$, and $L(\exp(x_1))$ are empty, $L(\hat{q}(x_1)) = L(\exp(x_1) - x_1^2) = (-3, 17)$.
We now construct representations for the sample points of the cells of the decomposition of the real line determined by the zeros of $\hat{q}(x_1)$. We have one 0-cell, and two 1-cells. Each 1-cell will have a rational sample point. We may take the 1-cell sample points to be -3, 17.
The 0-cell is not an algebraic number. Therefore, its representation consists of $p(x_1, y) = \exp(x_1) - x_1^2$, and the isolating interval $(-3, 17)$.
Finally, we use these three sample points to decide the original formula φ''. The result is *true* which is also the answer for φ. The answer is obtained after 110 ms.

A.2 Properties of the Exponential Function

1. We know that for all values of x, $\exp(x) > 0$. This shows the following sentence

   $$(\exists x_1)[y \leq 0], \text{ where } y = \exp(x_1).$$

   After 0 ms we get the answer *false*.

ISOL(p) , where $p(x,y) = y - x^2$, and $y = \exp(x)$
- pdeg $p^*(x) = (1,2)$;
- $p'(x,y) = y - 2x$;
- $s(x,y) = y - 2x$;
- **ISOL(p)** , where $p(x,y) = y - 2x$, and $y = \exp(x)$
  - pdeg $p^*(x) = (1,1)$;
  - $p'(x,y) = y - 2$;
  - $s(x,y) = y - 2$;
  - **ISOL(p)** , where $p(x,y) = y - 2$, and $y = \exp(x)$
    - pdeg $p^*(x) = (1,0)$;
    - $p'(x,y) = y$;
    - $s(x,y) = 1$;
    - **ISOL(p)** , where $p(x,y) = 1$, and $y = \exp(x)$
      - pdeg $p^*(x) = (0,0)$;
    - **RETURN** $L = ()$;
    - $L' = ()$;
    - $\gamma_+ = 4$;
    - $\gamma_- = -3$;
    - /* Since L' is empty, no interval has to be refined. */
  - **RETURN** $L = ((-3,4))$;
  - $L' = ((-3,4))$;
  - $\gamma_+ = 9$;
  - $\gamma_- = -3$;
  - /* Interval $(-3,4)$ has to be refined. */
  - **REFINE(I,p^*,s^*)** , where $I = (-3,4)$, $p^*(x) = \exp(x) - 2x$, and $s^*(x) = \exp(x) - 2$
    - /* First, the linear moc δ is computed as a rational number. */
    - /* To refine the interval $(-3,4)$, the bisection process has to be called eight times. */
  - **RETURN** $\bar{I} = (\frac{85}{128}, \frac{177}{256})$;
  - /* p^* has no roots in the refined interval. */
- **RETURN** $L = ()$;
- $L' = ()$;
- $\gamma_+ = 17$;
- $\gamma_- = -3$;
- /* Since L' is empty, no interval has to be refined. */

RETURN $L = ((-3,17))$;

Example A.1: ISOL(p)

2. The input sentence is

$$(\forall x_1)[y \geq 1 + x_1], \text{ where } y = \exp(x_1).$$

Afte 20 ms we get the answer *true*.

3. We know that for all $x < 1$, $\exp(x) \leq \frac{1}{1-x}$. Therefore, we formulate the following input sentence

$$(\forall x_1)[(1 - x_1) \cdot y \leq 1 \vee x_1 \geq 1], \text{ where } y = \exp(x_1).$$

After 60 ms we get the answer *true*.

A.3 Exponential Systems

1. The input sentence is

$$\begin{aligned}(\exists x_1)(\exists x_2)(\exists x_3)[(x_1 - 1) \cdot (x_1 - 2) = 0 \wedge \\ 2x_1 + x_2 - x_3 = 0 \wedge \\ y^2 \cdot x_2 - x_1 \cdot x_3 = 0],\end{aligned}$$

where $y = \exp(x_1)$. After 410 ms we get the answer *true*.

2. The input sentence is

$$\begin{aligned}(\exists x_1)(\exists x_2)(\exists x_3)[2x_1 - x_2 + x_3 + y^2 = 0 \wedge \\ 3x_2 - x_3 = 0 \wedge \\ 2x_1 + x_2 + 3x_3 + y = 0],\end{aligned}$$

where $y = \exp(x_1)$. After 190 ms we get the answer *true*.

3. The input sentence is

$$(\exists x)[y^2 - x \cdot y - 1 = 0 \wedge y^2 + y - 2x > 0],$$

where $y = \exp(x)$. After 160 ms we get the answer *true*.

A.4 Examples with Hyperbolic Functions

1. We compare the hyperbolic cosine $\cosh(x)$ with the polynomial $p(x) = x^3 - 4 \cdot x$. The prenex sentence is

$$(\forall x)[x > 7 \Rightarrow \cosh(x) > p(x)].$$

First, we have to transform the formula such that it is of $\mathcal{T}_{\exp}$. By substituting $\cosh(x)$ for the term $\frac{e^x+\frac{1}{e^x}}{2}$, follows:

$$\begin{aligned} & (\forall x)[x > 7 \Rightarrow \cosh(x) > p(x)] \\ \Leftrightarrow\ & (\forall x)[x \leq 7 \vee \exp(x) + \exp(-x) > 2 \cdot (x^3 - 4 \cdot x)] \\ \Leftrightarrow\ & (\forall x)[x \leq 7 \vee (\exp(x))^2 + 1 > (2 \cdot x^3 - 8 \cdot x) \cdot \exp(x)]. \end{aligned}$$

Then the input sentence is

$$(\forall x)[x \leq 7 \vee y^2 + 1 > (2 \cdot x^3 - 8 \cdot x) \cdot y],$$

where $y = \exp(x)$. After 16620 ms and with an accuracy of 50, we get the answer *true*.

2. We compare the hyperbolic sine $\sinh(x)$ with the polynomial $p(x) = x^3$. The prenex sentence is

$$(\forall x)[x > 2 \Rightarrow \sinh(x) < p(x)].$$

In analogy to the previous example, we have to transform the formula such that it is of $\mathcal{T}_{\exp}$. By substituting $\sinh(x)$ for the term $\frac{e^x-\frac{1}{e^x}}{2}$, follows:

$$\begin{aligned} & (\forall x)[x > 2 \Rightarrow \sinh(x) < p(x)] \\ \Leftrightarrow\ & (\forall x)[x \leq 2 \vee \sinh(x) < x^3] \\ \Leftrightarrow\ & (\forall x)[x \leq 2 \vee \exp(x) - \exp(-x) < 2 \cdot x^3] \\ \Leftrightarrow\ & (\forall x)[x \leq 2 \vee (\exp(x))^2 - 1 < 2 \cdot x^3 \cdot \exp(x)]. \end{aligned}$$

Then the input sentence is

$$(\forall x)[x \leq 2 \vee y^2 - 1 < 2 \cdot x^3 \cdot y],$$

where $y = \exp(x)$. After 20510 ms and with an accuracy of 55, we get the answer *false*.

3. We compare the hyperbolic tangent $\tanh(x)$ with the polynomial $p(x) = -\frac{1}{x} + 1$. The prenex sentence is

$$(\forall x)[x > 0 \Rightarrow \tanh(x) > p(x)].$$

In analogy to the previous examples, we have to transform the formula such that it is of $\mathcal{T}_{\exp}$. By substituting $\tanh(x)$ for the term $\frac{e^{2x}-1}{e^{2x}+1}$,

follows:

$$\begin{aligned}
& (\forall x)[x > 0 \Rightarrow \tanh(x) > p(x)] \\
\Leftrightarrow\ & (\forall x)[x \leq 0 \vee \tanh(x) > -\frac{1}{x} + 1] \\
\Leftrightarrow\ & (\forall x)[x \leq 0 \vee ((\exp(x))^2 - 1) \cdot x > (-1 + x) \cdot ((\exp(x))^2 + 1)] \\
\Leftrightarrow\ & (\forall x)[x \leq 0 \vee x \cdot (\exp(x))^2 - x > (x - 1) \cdot (\exp(x))^2 + x - 1].
\end{aligned}$$

Then the input sentence is

$$(\forall x)[x \leq 0 \vee x \cdot y^2 - x > (x - 1) \cdot y^2 + x - 1],$$

where $y = \exp(x)$. After 70 ms we get the answer *true*.

A.5 Examples with the Gauss Curve

In the following examples, we build prenex sentences where terms are considered polynomials in x, and $\gamma(x)$, where $\gamma(x) = \exp(-x^2)$ is the Gauss Curve. In order to decide these sentences with the developed procedure, we have to transformed them as explained in chapter 6.

1. We compare the Gauss curve with the polynomial $p(x) = \frac{x^2}{4} + \frac{1}{4 \cdot x^2}$. The prenex sentence is

$$\begin{aligned}
& (\exists x)[\gamma(x) = p(x)] \\
\Leftrightarrow\ & (\exists x)[\exp(-x^2) = p(x)] \\
\Leftrightarrow\ & (\exists z)(\exists x)[\exp(z) = p(x) \wedge z + x^2 = 0] \\
\Leftrightarrow\ & (\exists z)(\exists x)[\exp(z) = \frac{x^2}{4} + \frac{1}{4 \cdot x^2} \wedge z + x^2 = 0] \\
\Leftrightarrow\ & (\exists z)(\exists x)[4 \cdot x^2 \cdot \exp(z) = x^2 + 1 \wedge z + x^2 = 0].
\end{aligned}$$

   Then we decide the following prenex sentence

$$(\exists z)(\exists x)[4 \cdot x^2 \cdot y = x^2 + 1 \wedge z + x^2 = 0],$$

   where $y = \exp(z)$. After 320 ms we get the answer *false*.

2. We compare the Gauss curve with the polynomial $p(x) = -\frac{8}{10} \cdot x^2 + 1$. The prenex sentence is

$$\begin{aligned}
& (\exists x)[\gamma(x) = p(x) \wedge x \neq 0] \\
\Leftrightarrow\ & (\exists x)[\exp(-x^2) = p(x) \wedge x \neq 0] \\
\Leftrightarrow\ & (\exists z)(\exists x)[\exp(z) = p(x) \wedge x \neq 0 \wedge z + x^2 = 0] \\
\Leftrightarrow\ & (\exists z)(\exists x)[\exp(z) = -\frac{8}{10} \cdot x^2 + 1 \wedge x \neq 0 \wedge z + x^2 = 0] \\
\Leftrightarrow\ & (\exists z)(\exists x)[10 \cdot \exp(z) = -8 \cdot x^2 + 10 \wedge x \neq 0 \wedge z + x^2 = 0].
\end{aligned}$$

Then we decide the following prenex sentence

$$(\exists z)(\exists x)[10 \cdot y = -8 \cdot x^2 + 10 \wedge x \neq 0 \wedge z + x^2 = 0],$$

where $y = \exp(z)$. After 270 ms we get the answer *true*.

Bibliography

[1] H. Anai and V. Weispfenning. *Deciding linear-trigonometric problems*. In C. Traverso, editors, *ISSAC'2000*. ACM-Press, 2000, pages 14-22.

[2] D.S. Arnon, G.E. Collins, and S. McCallum. Cylindical algebraic decomposition I: Basic Algorithm. In B.B. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Sympolic Computation. Springer, Wien, New York, 1998, pages 136-151.

[3] T. Becker, H. Kredel and V. Weispfenning. *Gröbner Bases, a Computational Approach to Commutative Algebra*, volume 141 of *Graduate Texts in Mathematics*. Springer, New York, corrected second printing edition, 1998.

[4] E. Bishop. *Foundations of Constructive Analysis*. McGraw-Hill, New York, 1967.

[5] E. Bishop and D. Bridges. *Constructive Analysis*. Grundlehren der math. Wissenschaften. Springer-Verlag, 1985.

[6] J.C. Burkill. *A first course in Mathematical Analysis*. Cambridge University Press, 1964, pp.73.

[7] G.E. Collins, and R. Loos. Polynomial real root isolation by differentiation. *Symposium on Symbolic and Algebraic Manipulation*. Yorktown Heights, New York, United States, 1976, pages 15-25.

[8] G.E. Collins and R. Loos. Computing in algebraic extensions. In *Computer Algebra: Symbolic and Algebraic Manipulation*, Bruno Buchberger, George E. Collins, Rüdiger Loos, and Rudolf Albrecht, editors. Springer-Verlag, Wien, New York, second edition, 1982, pages 173-188.

[9] G.E. Collins and R. Loos. Real zeros of polynomials. In *Computer Algebra: Symbolic and Algebraic Manipulation*, Bruno Buchberger, George E. Collins, Rüdiger Loos, and Rudolf Albrecht, editors. Springer-Verlag, Wien, New York, second edition, 1982, pages 83-94.

[10] G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Sympolic Computation. Springer, Wien, New York, 1998, pages 85-121.

[11] A. Dolzmann and T. Sturm. *Redlog: Computer algebra meets computer logic*. ACM SIGSAM Bulletin, 31(2):2–9, 1997.

[12] A. Dolzmann and T. Sturm. Redlog User Manual. FMI, Universität Passau, D-94030 Passau, Germany, Edition 3.0, April 2004.

[13] M.L. Griss, C. Griss, A.C. Hearn, and J. Marti. *Standard Lisp report*. SIGSAM Bulletin, 14(1):23-41, February 1980.

[14] G.H. Hardy. *Orders of Infinity*. Cambridge University Press, Cambridge, 1910.

[15] A.C. Hearn. *REDUCE User's Manual for Version 3.8*. RAND, Santa Monica, CA, February 2004.

[16] J.R. Johnson. Algorithms for polynomial real root isolation. In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Sympolic Computation. Springer, Wien, New York, 1998, pages 269-299.

[17] S. Kratz and H. Parks. *A Primer of Real Analytic Functions*. Basler Lehrbücher. Birkhauser, 1992.

[18] A. Macintyre and A.J. Wilkie. On the decidability of the real exponential field. In *Kreiseliana: About and around Georg Kreisel*, A.K. Peters, 1996, pages 441-467.

[19] S. McCallum and V. Weispfenning. *Deciding Polynomial-Transcendental Problems*. 2006.

[20] H. Melenk. *REDUCE Symbolic Mode Primer*. Konrad-Zuse-Zentrum für Informationstechnik, Berlin.

[21] A. Seidl. *Cylindrical Decomposition Under Application-Oriented Paradigms*. Doctoral Dissertation, Fakultät für Mathematik und Informatik, Universität Passau, 2006, to appear.

[22] A.B. Shidlovskii. *Transcendental Numbers*. Walter de Gryter, Berlin, New York, 1989, p.51.

[23] A. Tarski. A decision algorithm for elementary algebra and geometry. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*, B.F. Caviness and J.R. Johnson, Eds., Texts and Monographs in Symbolic Computation. Springer, Wien, New York, 1998, pages 24-84.

[24] M. Triska, *Approximation von Standardfunktionen*, November 2004, `http://stud4.tuwien.ac.at/~e0225855/various/compnumerik.pdf`

[25] V. Weispfenning. *Deciding linear-exponential problems*. Poster presentation at ISSAC'99, Vancouver, July 1999.

[26] V. Weispfenning. *Deciding linear-transcendental problems*. In V.G. Ganzha, E.W. Mayr, and E.V. Vorozhtshov, editors, *Computer Algebra in Scientific Computation - CASC 2000*, Springer, 2000, pages 423-438.

Acknowledgement

I would like to thank both of my supervisors for their support in preparing and writing my thesis. First of all, I want to express my gratitude to Scott McCallum, Ph.D. of Macquarie University, Sydney who took the effort to supervise and help me all through my stay in Sydney. Likewise, I am very much indebted to Prof. Dr. Volker Weispfenning of Universität Passau who made it possible for me to write my thesis in Sydney. He was always there to answer my questions.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich diese Diplomarbeit selbständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle wörtlich oder sinngemäß übernommenen Ausführungen wurden als solche gekennzeichnet. Weiterhin erkläre ich, dass ich diese Arbeit in gleicher oder ähnlicher Form nicht bereits einer anderen Prüfungsbehörde vorgelegt habe.

Passau, den 29. September 2006

. . . . . .
(Melanie Achatz)