Generic asymptotic expansions

by Joris van der Hoeven

LIX
École Polytechnique
F-91128, Palaiseau cedex
France

Email: vdhoeven@lix.polytechnique.fr
Web : http://lix.polytechnique.fr/~vdhoeven

November 20, 1997

Abstract

We give an expansion algorithm for germs of exp-log functions at infinity which is correct modulo Schanuel's conjecture. We also show how the algorithm can be made generic. More precisely, we reduce the expansion algorithm for exp-log functions depending on parameters to the problem of deciding whether a given system of exp-log equations and inequalities in several variables admits a solution.

Key words: Asymptotic expansion, exp-log function, algorithm, genericity.

1 Introduction

An exp-log function is a function built up from x and the rational numbers $\mathbb{Q}$ by the field operations, exponentiation and logarithm. In this paper, we shall only consider exp-log functions which are defined in a neigbourhood of infinity. Hardy has shown that, ultimately, such functions are either negative, zero or positive [11, 12]. In other words, the germs of exp-log functions at infinity form a totally ordered field. But how to decide whether a given exp-log function is asymptotically superior to another one in a neighbourhood of infinity? More generally, is it possible to compute an asymptotic expansion of a given exp-log function in a natural asymptotic scale? The main problem one encounters here is the problem of *indefinite cancelation*: consider a function like

$$f(x) = \frac{1}{1 - x^{-1}} - \frac{1}{1 - x^{-1} - e^{-x}}. \tag{1}$$

The naive algorithm to compute the expansion of f enters in an infinite loop, due to the cancelations $1 - 1 = 0, x^{-1} - x^{-1} = 0$, etc.

The first attempt to solve these problems was made by Geddes and Gonnet [8]. Shackell is the first to give an algorithm in [29] for computing the sign of an exp-log function at infinity, under the assumption that an oracle is given to decide

whether an exp-log function vanishes in a neighbourhood of infinity. His technique is based on so called nested expansions, by which one can find the order of growth of exp-log functions at infinity, but which do not allow to derive complete asymptotic expansions. This drawback is removed in [30], where Shackell gives a complete and natural asymptotic expansion algorithm. A weaker version of this algorithm, which only computes limits of exp-log functions was discovered independently in [10], and is currently incorporated in MAPLE V.3. The author generalized this limit computation algorithm and obtained variants of Shackell's algorithm in [14, 13]. An elegant synthesis of these algorithms appeared in [25].

However, several related problems were overlooked up till now. First, can we reduce the problem of deciding whether an exp-log function is zero at infinity to the corresponding problem for exp-log constants? Although several algorithms exist for deciding whether a given exp-log function f is locally zero in the neighbourhood of a point of analyticity [26, 6, 28, 20], no one considered the problem of deciding whether the germ of f at infinity is zero. Indeed, these problems are *not* equivalent: consider the function

$$f = \sqrt{x^2} - x.$$

In computer algebra, f is usually represented by $y - x$ in the ring $\mathbb{Q}[x, y]/(y^2 - x^2)$, whence $f \neq 0$. Performing a local zero test for f in a point $x < 0$, we also find that $f \neq 0$. However, f vanishes in a neighbourhood of infinity.

The second problem concerns the improvement of the dramatic complexity of the algorithm from [25]. Although this algorithm detects indefinite cancelations in (1), it does not detect the first N cancelations in the asymptotic expansion of

$$\frac{1}{1 - x^{-1}} - \frac{1}{1 - x^{-1} - x^{-N}}. \tag{2}$$

Hence, taking $N = \exp \cdots \exp 1$ very large, we observe that the complexity of the algorithm is worse than any iterated exponential, even on simple examples. Moreover, the algorithm can not be generalized to the case of *parameterized* exp-log functions (see also below), because it would not terminate on an example like (2), if N is a formal parameter.

In this article, we propose a new algorithm in order to deal with these problems and a generalization to the case of parameterized exp-log functions. In section 2, we first recall the basic expansion algorithm from [25], using the terminology from our thesis [16].

In section 3, we introduce the important concept of *Cartesian representations* in automatic asymptotics: we show that, given the germ at infinity of an exp-log function f, their exist infinitesimal elements $z_1, \cdots, z_k$ of a suitable asymptotic scale, such that f can be represented by a Laurent series in $z_1, \cdots, z_k$ (its Cartesian representation). This reduces the problem of computing with generalized power series (see (3) and (4) in section 2.1) to the analogue, more classical, problem for Laurent or power series in several variables. This point of view is beneficial for many reasons:

- We can efficiently detect cancelations of large numbers of terms like in (2), and obtain an improvement of the algorithm from section 2.
- We can reduce the problem of asymptotic zero tests for exp-log functions to the problem of zero tests for certain Laurent series in several variables (see section 4).
- Using a heuristic zero test for these Laurent series, we obtain a fast and reliable heuristic asymptotic zero test for exp-log functions.
- Several efficient algorithms can be used for the manipulation of Laurent series in several variables [18, 1, 2, 17].
- Cartesian representations are essentially needed for a further development of automatic asymptotics: see section 5 and [16].

Although we did not perform a detailed complexity analysis of our expansion algorithm which uses Cartesian representations, we have implemented a prototype of it in C++. This program takes a few seconds to compute several terms of the asymptotic expansion of a typical exp-log function, and, we presently do not have examples (like (2) for the previous algorithm) on which our implementation fails.

In section 4, we show how to decide whether an exp-log function is asymptotically zero, modulo an oracle for determining the signs of exp-log constants. In [24], Richardson has given a partial algorithm to decide whether an exp-log constant is zero: whenever the algorithm produces an answer, then this answer is correct, but termination is only guaranteed modulo Schanuel's conjecture. Moreover, if we can prove that the algorithm does not terminate on a given input, then we can construct a counterexample to Schanuel's conjecture from this input. In principle, Richardson's algorithm also yields a method to compute the sign of an exp-log constant: it suffices to perform a floating point evaluation at a sufficient precision. In practice, this method is intractable and a more efficient algorithm for sign computations was proposed in [15]. For completeness, we state Schanuel's conjecture:

Conjecture 1. (Schanuel) *If $\alpha_1, \cdots, \alpha_n$ are $\mathbb{Q}$-linearly independent complex numbers, then the transcendence degree of $\mathbb{Q}[\alpha_1, \cdots, \alpha_n, e^{\alpha_1}, \cdots, e^{\alpha_n}]$ over $\mathbb{Q}$ is at least n.*

In section 5, we show how to generalize our expansion algorithm to the case of germs of exp-log functions depending on parameters $\lambda_1, \cdots, \lambda_k$, using the technique of dynamic evaluation [5, 9]. This means that we consider exp-log functions built up from $\lambda_1, \cdots, \lambda_k$, x and $\mathbb{Q}$, by field operations, exponentiation and logarithm, where x tend to infinity. Letting x tend to infinity, one can see $\lambda_1, \cdots, \lambda_k$ as parameters. Our algorithm uses an oracle to test whether a given system of exp-log equations and inequalities admits a solution. We show that, given an exp-log function depending

on parameters, a finite number of cases can be distinguished, each for which we have a general solution to the expansion problem.

2 The basic algorithm

Let $\mathfrak{T}$ denote the field of germs at infinity of exp-log functions and $\mathfrak{C}$ the subfield of exp-log constants. Elements of $\mathfrak{T}$ can be represented by exp-log expressions — i.e. finite trees whose internal nodes are labeled by $+, -, \cdot, /, \exp$ or $\log$, and whose leafs are labeled by x or rational numbers. The set of exp-log expressions which can be evaluated in a neighbourhood of infinity is denoted by $\mathfrak{T}^{expr}$. We have a natural projection $f \mapsto \overline{f}$ from $\mathfrak{T}^{expr}$ onto $\mathfrak{T}$. In sections 2 and 3, we make the assumption that we have an oracle which decides whether a given exp-log expression in $\mathfrak{T}^{expr}$ is zero in a neighbourhood of infinity.

In this section we recall the classical asymptotic expansion algorithm for exp-log functions at infinity from [25] (see also [30, 13]). We use the terminology from [VdH 97a], which is better suited for generalizations of the algorithm to the multivariate case [16]. For a detailed example of the algorithm at work, we refer to [25].

2.1 Grid-based series

Let us first recall some basic concepts. An *effective asymptotic basis* is an ordered finite set $\{b_1, \cdots, b_n\}$ of positive infinitesimal exp-log expressions in $\mathfrak{T}^{expr}$, such that $\log b_i = o(\log b_{i+1})$ for $1 \leqslant i \leqslant n-1$. For instance, the set $B = \{\log^{-1} x, x^{-1}, e^{-x^2}\}$ is an effective asymptotic basis. An effective asymptotic basis B generates an *effective asymptotic scale*, namely the set S_B of all products $b_1^{\alpha_1} \cdots b_n^{\alpha_n}$ of powers of the b_i, with the α_i in $\mathfrak{C}$. Elements of S_B are also called *monomials*.

Given an effective asymptotic basis B, let $\mathfrak{G}_B^{expr}$ denote the set of expressions which are built up from $\mathfrak{C}, S_B, +, -, \cdot, /$ and the operations $\varepsilon \mapsto \exp \varepsilon$ resp. $\varepsilon \mapsto \log(1+\varepsilon)$ for infinitesimal ε. We observe that each exp-log expression $f \in \mathfrak{G}_B^{expr}$ has a series expansion of the form

$$f = \sum_{(\alpha_1, \cdots, \alpha_n) \in \mathfrak{C}^n} f_{\alpha_1, \cdots, \alpha_n} b_1^{\alpha_1} \cdots b_n^{\alpha_n}. \tag{3}$$

Alternatively, we can expand f as a series in b_n with coefficients in $\mathfrak{G}^{expr}_{\{b_1, \cdots, b_{n-1}\}}$. These coefficients can recursively be expanded in $b_{n-1}, \cdots, b_1$:

$$\begin{aligned} f &= \textstyle\sum_{\alpha_n \in \mathfrak{C}} f_{\alpha_n} b_n^{\alpha_n} \\ &\vdots \\ f_{\alpha_n, \cdots, \alpha_2} &= \textstyle\sum_{\alpha_1 \in \mathfrak{C}} f_{\alpha_n, \cdots, \alpha_1} b_1^{\alpha_1}. \end{aligned} \tag{4}$$

The exp-log expressions of the form $f_{\alpha_n, \cdots, \alpha_i}$ are called *iterated coefficients* of f. In particular, the iterated coefficients of the form $f_{\alpha_n, \cdots, \alpha_1}$ are exp-log constants.

The above expansions of f have an important property [16]: the support of f as a series in b_n (resp. $b_1, \cdots, b_n$) is included in a set of the form $\lambda_1 \mathbb{N} + \cdots + \lambda_p \mathbb{N} + \nu$ — we say that f is a *grid-based series*. Here the λ_i and ν are constants in $\mathfrak{C}$ (resp. vectors in $\mathfrak{C}^n$). From this property, it follows that the support of f is well-ordered (in the case of vector supports, $\mathfrak{C}^n$ is ordered lexicographically; see also the ordering $\preccurlyeq_B$ on S_B in section 3.3). In particular, if f is non-zero, then the expansion of f admits a first term, which we call the *dominant term* of f. The corresponding monomial in S_B and its coefficient are called the *dominant monomial* and *dominant coefficient* of f respectively.

Another important property of the expansion of f in b_n and the expansions of its iterated coefficients is that they can be computed automatically. By this we mean that for each integer i, we can compute the first i terms of the expansion of f and so can we for its iterated coefficients. In particular, we can compute the sign of f, test whether f is infinitesimal, test whether $f \asymp 1$ (i.e. $f = O(1)$ and $1 = O(f)$), etc.

For the computation of the expansions of f in b_n, we use the usual Taylor series formulas. In the case of division $1/f$, we compute the first term $f_\mu b_n^\mu$ of f and then use the formula $1/f = (1/f_\mu) b_n^{-\mu} (1/(1+\varepsilon))$, where $\varepsilon = (f/f_\mu b_n^\mu) - 1$. The only problem when applying these formulas is that we have to avoid indefinite cancelation. Now we note that indefinite cancelation only occurs if after having computed the first i terms of the expansion, f is actually equal to the sum of these terms. But this can be tested using the oracle, and we stop the expansion in this case.

2.2 Automatic expansions of exp-log expressions

The asymptotic expansion algorithm takes an exp-log expression $f \in \mathfrak{T}^{expr}$ on input, computes a suitable effective asymptotic basis B and rewrites f into an element of $\mathfrak{G}_B^{expr}$. The main idea of the algorithm lies in the idea to impose some suitable conditions on B: we say that a linearly ordered set $B = \{b_1, \cdots, b_n\}$ is an *effective normal basis* if

NB1. B is an effective asymptotic basis.

NB2. $b_1 = \log_l^{-1} x$ for some $l \in \mathbb{N}$, where $\log_l x \stackrel{\text{def}}{=} \log^{l \text{ times}} \cdots \log x$.

NB3. For all $i > 1$ there exists an $i^* < i$ with $\log b_i \in \mathfrak{G}_{\{b_1, \cdots, b_{i^*}\}}^{expr}$ and $\log\log b_i^{-1} \asymp \log b_{i^*}$.

Such a basis is constructed gradually during the algorithm — i.e. B is a global variable in which we insert new elements during the execution of the algorithm, while maintaining the property that B is an effective normal basis. We also say that B is a *dynamic effective normal basis*. We initialize B with $B := \{x^{-1}\}$. Let us now explicitly give the algorithm, using a PASCAL-like notation:

Algorithm `expand`(f).
INPUT: An exp-log expression $f \in \mathfrak{T}^{expr}$.
OUTPUT: A grid-based series φ in $\mathfrak{G}_B^{expr}$ with $\overline{\varphi} = \overline{f}$.

case $f \in \mathbb{Q}$: **return** f

case $f = x$: **return** $(x^{-1})^{-1}$

case $f = g \top h$, $\top \in \{+, -, \cdot, /\}$:
  if $\top = /$ **and** $\overline{h} = 0$ **then error** "division by zero"
  return `expand`$(g) \top$ `expand`(h)

case $f = \log g$:
  $g :=$ `expand`(g)
  • Denote $B = \{b_1 = \log_l^{-1} x, b_2, \cdots, b_n\}$.
  if $g \leqslant 0$ **then error** "invalid logarithm"
  • Rewrite $g = c b_1^{\alpha_1} \cdots b_n^{\alpha_n}(1 + \varepsilon)$, with infinitesimal ε in $\mathfrak{G}_B^{expr}$.
  if $\alpha_1 \neq 0$ **then** $B := B \cup \{\log_{l+1}^{-1} x\}$
  return $\log c + \alpha_1 \log b_1 + \cdots + \alpha_n \log b_n + \log(1 + \varepsilon)$

case $f = e^g$:
  $g :=$ `expand`(g)
  • Denote $B = \{b_1, \cdots, b_n\}$.
  if $g = O(1)$ **then return** $e^c e^{g-c}$, where $c := g_{0,\overset{n \text{ times}}{\cdots},0}$
  if $\exists 1 < i \leqslant n \quad g \asymp \log b_i$ **then**
    $\alpha := \lim(g / \log b_i)$
    return b_i^{α} `expand`$(e^{g - \alpha \log b_i})$
  • Let i^* be such that $\log |g| \asymp \log b_{i^*}$.
  $g^+ := g_{0,\overset{n-i^* \text{ times}}{\cdots},0}$
  $g^- := g - g^+$
  $B := B \cup \{e^{-|g^+|}\}$
  return $(e^{-|g^+|})^{-\mathbf{sign}\, g^+} e^{g^-}$

Let us comment the algorithm. The first three cases do not need explanation. In the case $f = \log g$, the fact that B is an effective normal basis is used at the end: $\alpha_1 \log b_1 + \cdots + \alpha_n \log b_n$ is indeed an expression in $\mathfrak{G}_B^{expr}$. The expansion of the exponential of a bounded series g is done by a straightforward Taylor series expansion. If g is unbounded, then we test whether g is asymptotic to the logarithm of an element in B — i.e. we test whether $\alpha := \lim(g / \log b_i)$ is a non-zero finite number for some i. If this is so, then $f = b_i^{\alpha} e^{g - \alpha \log b_i}$ and $e^{g - \alpha \log b_i}$ is expanded recursively. We remark that no infinite loops can arise from this, because successive values of g in such a loop would be asymptotic to the logarithms of smaller and smaller elements of B, while B remains unchanged. Finally, if g is not asymptotic to the logarithm of an element in B, then B has to be extended with an element having

the order of growth of f. The decomposition $g = g^+ + g^-$ is computed in order to ensure that B remains an effective normal basis.

3 Cartesian representations

In practice it is not always efficient to perform the expansions of elements in $\mathfrak{G}_B$ by applying the classical formulas for Taylor series expansions in a direct way. Consider for example the expression

$$f(x) = \frac{1}{1 - x^{-1}} - \frac{1}{1 - x^{-1}} + x^{-N},$$

or, alternatively,

$$f(x) = \frac{1}{1 - x^{-1}} - \frac{1}{1 - x^{-1} - x^{-N}},$$

where N is very large (say $N = 10^{10^{100}}$) and x tends to infinity. Determining the first term of this series using a straightforward expansion would need a time proportional to N. The point here is that, in order to detect the cancelation $1/(1-x^{-1}) - 1/(1 - x^{-1}) = 0$, we need to represent f as a Laurent series in two variables, namely x^{-1} and x^{-N}. This is possible by the fact that f is a grid-based series in x^{-1}. In this section we show that any exp-log expression f can be represented in such a way and how to exploit this in order to improve the algorithm `expand` from section 2.

3.1 Cartesian representations

A Laurent series u in several variables $z_1, \cdots, z_k$ is a series in $z_1, \cdots, z_k$ whose support is included in $(\mathbb{N} + p_1) \times \cdots \times (\mathbb{N} + p_k)$ for certain $p_1, \cdots, p_k \in \mathbb{Z}$. We say that u is infinitesimal if its support is included in $\mathbb{N}^k \backslash (0, \cdots, 0)$. The α-th coefficient of u in z_i is denoted by $[z_i^\alpha]u$. We abbreviate $[z_{i_1}^{\alpha_1}] \cdots [z_{i_j}^{\alpha_j}]u$ by $[z_{i_1}^{\alpha_1} \cdots z_{i_j}^{\alpha_j}]u$. We notice that $z_1, \cdots, z_k$ should be interpreted as variables which tend to zero.[1]

Let B be an effective asymptotic basis and let $Z = \{z_1, \cdots, z_k\}$ be a finite set of infinitesimal monomials in S_B. We denote by $\mathfrak{L}_L^{expr}$ the set of expressions built up from $\mathfrak{C}, z_1, z_1^{-1}, \cdots, z_k$ and z_k^{-1} by $+, -, \cdot$ and the operations $\varepsilon \mapsto e^\varepsilon, \varepsilon \mapsto \log(1 + \varepsilon)$ and $\varepsilon \mapsto 1/(1 + \varepsilon)$ for infinitesimal ε. Given such a Laurent series $u \in \mathfrak{L}_Z^{expr}$, its expansion

$$u = \sum_{\alpha = p_i}^{\infty} ([z_i^\alpha]u)\, z_i^\alpha$$

in any of the z_i can be computed automatically. Moreover, the coefficients $[z_i^\alpha]u$ of such an expansion are also expressions in $\mathfrak{L}_Z^{expr}$, so that they can recursively be

[1] The fact that x tends to infinity and $z_1, \cdots, z_k$ to zero might confuse the reader. This apparently illogical choice stems from the potentially different asymptotic behaviours of an exp-log function $f(x)$, if x tends to zero from below or from above.

expanded — we say that u is an *automatic Laurent series*. In what follows, we will only consider automatic Laurent series which are in $\mathfrak{L}_Z^{expr}$ for some Z.

Remark. We notice that many efficient expansion algorithms for formal Laurent series in $\mathfrak{L}_Z^{expr}$ can be used, such as Karatsuba's or FFT multiplication [18] and Brent and Kung's or the author's algorithms for exponentiation and logarithm [1, 2, 17]. We also remark that we systematically store all coefficients of all expansions we compute, in order to perform these computations only once (i.e. we use a MAPLE-like remember option).

We denote by $\overline{u}$ the germ at infinity of the exp-log function represented by a Laurent series u in $\mathfrak{L}_Z^{expr}$. We call u a *Cartesian representation* of $\overline{u}$. Let an expression $f \in \mathfrak{G}_B^{expr}$ be given. The aim of the rest of this section is to compute a Cartesian representation $u \in \mathfrak{L}_Z^{expr}$ of $\overline{f}$ for some suitable subset Z of S_B. Furthermore, we will show how to compute the expansion of f from the knowledge of u only. Clearly, this will enable us to replace all computations with elements in $\mathfrak{G}_B^{expr}$ by computations with Cartesian representations in `expand`.

Warning. One should carefully distinguish Cartesian representations from the germs at infinity they represent. For instance, if $B = \{x^{-1}, e^{-x}\}$, $z_1 = x^{-1}$ and $z_2 = e^{-x}$, then $\overline{z_1^{-1} z_2}$ is infinitesimal, while $z_1^{-1} z_2$ is not. In cases where confusion might arise, we therefore distinguish u from $\overline{u}$ by means of the upper bar. Moreover, we will use the prefix "C-" to emphasize that we are referring to properties of Cartesian representations. For instance, infinitesimal Cartesian representations will be called C-infinitesimal.

3.2 Restrictions of Cartesian representations

Let $Z = \{z_1, \cdots, z_k\}$ and let $S_Z = \{z_1^{\alpha_1} \cdots z_k^{\alpha_k} | \alpha_1, \cdots, \alpha_k \in \mathbb{Z}\}$ be the set of monomials in $z_1, \cdots, z_k$. We have a natural partial ordering on S_Z:

$$z_1^{\alpha_1} \cdots z_k^{\alpha_k} \leqslant_Z z_1^{\beta_1} \cdots z_k^{\beta_k} \Leftrightarrow \alpha_1 \leqslant \beta_1 \ \wedge \ \cdots \ \wedge \ \alpha_k \leqslant \beta_k.$$

Let u be a Laurent series in $z_1, \cdots, z_k$ and let Φ be a subset of S_Z. We denote by

$$[u|\Phi] = \sum_{z_1^{\alpha_1} \cdots z_k^{\alpha_k} \in \Phi} ([z_1^{\alpha_1} \cdots z_k^{\alpha_k}]u)\, z_1^{\alpha_1} \cdots z_k^{\alpha_k}$$

the *restriction* of u w.r.t. Φ. For singletons $\Phi = \{\varphi\}$ we also write $[u|\varphi] = [u|\{\varphi\}]$. We finally define $(\Phi) = \{\psi \in S_Z | \exists \varphi \in \Phi \ \ \varphi \leqslant_Z \psi\}$ to be the final segment generated by Φ. Here we recall that a *final segment* of S_Z is a subset $F \subseteq S_Z$ such that $\varphi \in F \ \wedge \ \varphi \leqslant_Z \psi \Rightarrow \psi \in F$ for all $\varphi, \psi \in S_Z$.

Proposition 1. *Let u be a Laurent series in $\mathfrak{L}_Z^{expr}$. There exists an algorithm to compute the restriction $[u|(\Phi)]$ of u w.r.t. any final segment (Φ) for finite Φ.*

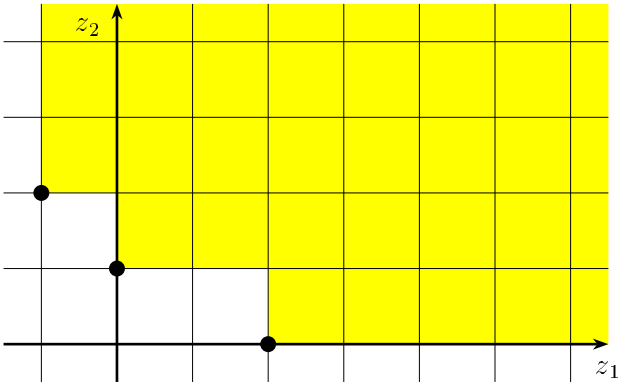

Figure 1: Dominant monomials for $u = \frac{1}{z_1(1-z_1-z_2)} - z_1^{-1} - 1 - z_1 - 3z_2 - z_1^{-1}z_2$.

Proof. Let Φ_α denote the subset of Φ of monomials whose exponents in z_k are inferior or equal to α and let λ be the smallest integer with $\Phi = \Phi_\lambda$. Let Ψ_α denote the set of monomials ψ in $z_1, \cdots, z_{k-1}$, such that ψz_k^β is in Φ_α, for some $\beta \leqslant \alpha$. Now expand u up to order λ in z_k, say $u = u_{p_k} z_k^{p_k} + \cdots + u_{\lambda-1} z_k^{\lambda-1} + \tilde{u}$. Then we have

$$[u|\Phi] = \sum_{i=p_k}^{\lambda-1} [u_j|(\Psi_\alpha)]\, z_k^\alpha + [\tilde{u}|(z_k^\lambda \Psi_\lambda)].$$

The right hand side of this equation is evaluated by expanding each of the terms in $z_{k-1}, \cdots, z_1$ using the same method. □

3.3 Intermediate dominant monomials

Let $u \in \mathfrak{L}_Z^{expr}$ be a Cartesian representation. A set of *intermediate dominant monomials* of u is a finite subset G of S_Z, such that $\overline{u} = \overline{[u|(G)]}$, and such that the dominant monomial of f is equal to $\overline{\varphi}$, for each minimal monomial φ in G for $\leqslant_Z$. Most of the time, but not always, G is unique and we say that G is the set of intermediate dominant monomials.

In figure 1, we have represented the dominant monomials of $u = z_1^{-1}(1 - z_1 - z_2)^{-1} - z_1^{-1} - 1 - z_1 - 3z_2 - z_1^{-1}z_2$. If $\overline{z_1} \neq \overline{z_2}$ and $\overline{z_1^2} \neq \overline{z_2}$, then $\{z_1^{-1}z_2^2, z_2, z_1^2\}$ is also the set of intermediate dominant monomials of u. If $\overline{z_1} = \overline{z_2}$, then $\{z_1^{-1}z_2^3, z_2^2, z_1z_2, z_1^2\}$ is the set of intermediate dominant monomials of u, because of the cancelation $z_1^{-1}z_2^2 - z_2 = 0$. Similarly, if $\overline{z_1^2} = \overline{z_2}$, then $\{z_1^{-1}z_2^2, z_1z_2, z_1^3\}$ is the set of intermediate dominant monomials of u.

In order to compute intermediate dominant monomials, we first need to introduce some more orderings. First, we have a total ordering $\leqslant_B$ on S_B, which is analogous

to $\leqslant_Z$ on S_Z:

$$b_1^{\alpha_1}\cdots b_n^{\alpha_n} \leqslant_B b_1^{\beta_1}\cdots b_n^{\beta_n} \Leftrightarrow b_1^{\beta_1}\cdots b_n^{\beta_n} = O(b_1^{\alpha_1}\cdots b_n^{\alpha_n}).$$

Via the natural (not necessarily injective) mapping $\nu : S_Z \to S_B$, the ordering $\leqslant_B$ induces a quasi-ordering $\preccurlyeq_B$ on S_Z: $\varphi \preccurlyeq_B \psi \Leftrightarrow \nu(\varphi) \leqslant_B \nu(\psi)$ for all $\varphi, \psi \in S_Z$. The reader should not confuse this quasi-ordering with $\leqslant_Z$, nor with the usual asymptotic ordering on germs of exp-log functions (which is actually opposite to $\preccurlyeq_B$ on S_Z). Now consider the following algorithm:

Algorithm `idm`(u).
INPUT: A Cartesian representation $u \in \mathfrak{L}_Z^{expr}$ with $\overline{u} \neq 0$.
OUTPUT: A set of intermediate dominant monomials for u.

• Let $z_i^{p_i}$ be the dominant monomial of u in z_i, for $1 \leqslant i \leqslant k$.
$G := \{z_1^{p_1}\cdots z_k^{p_k}\}$
while true
  $M := \{\varphi \in G | \forall \psi \in G \quad \varphi \preccurlyeq_B \psi\}$
  if $\overline{\sum_{\varphi\in M} u_\varphi} \neq 0$ **then return** G
  • Denote $G\backslash M = \{\varphi_1, \cdots, \varphi_q\}$, with $\varphi_1 \preccurlyeq_B \cdots \preccurlyeq_B \varphi_q$.
  if $\exists 0 \leqslant i \leqslant q \quad \overline{u - [u|(\varphi_i, \cdots, \varphi_q)]} = 0$
    then $G := \{\varphi_i, \cdots, \varphi_q\}$ (with i chosen minimal)
    else $G := (G\backslash M) \cup M\{z_1, \cdots, z_k\}$
  • Eliminate non minimal elements from G.

Remark. We recall the existence of an oracle to decide whether a given exp-log expression in $\mathfrak{T}^{expr}$ is zero in a neighbourhood of infinity. Hence, the test $\exists 0 \leqslant i \leqslant q \quad \overline{u - [u|(\varphi_i, \cdots, \varphi_q)]} = 0$ is indeed effective, by proposition 1.

Proposition 2. *The algorithm* `idm` *is correct and terminates.*

Proof. Let $G_1, G_2, \cdots$ be the successive values of G at the beginning of the main loop. By induction, we observe that $\overline{u} = \overline{[u|(G_j)]}$ for all j. This proves the correctness of `idm`. Suppose that the algorithm does not terminate. Let $F = \bigcap_{j\geqslant 1}(G_j)$. We have $\overline{u} = \overline{u_F}$. By Dickson's lemma, F is finitely generated, say by Γ. There are only a finite number of monomials $\varphi \geqslant_Z z_1^{p_1}\cdots z_k^{p_k}$ with $\varphi <_Z \psi$ for some $\psi \in \Gamma$. For sufficiently large j, none of these monomials belongs to G_j. We have $\Gamma \subseteq G_j$, since $\Gamma \subseteq (G_j)$. There do not exist $\varphi \in G_j\backslash\Gamma$ and $\psi \in \Gamma$, with $\psi <_B \varphi$: indeed, such a φ would belong to $G_{j'}$ for all $j' \geqslant j$, although $\varphi \notin F = (\Gamma)$. We deduce that $\Gamma = \{\varphi_i, \cdots, \varphi_q\}$ at the j-th iteration of the main loop for some q. But this means that $G_{j+1} = \Gamma$ and $\Gamma \not\subseteq (G_{j+2})$. This contradiction proves the termination of `idm`. □

Remark. We observe that the dominant term τ_f of f is given by $\tau_f = \sum_{\varphi\in M} u_\varphi\varphi$ at the end of the algorithm. More terms of the expansion of f can be obtained by rerunning the algorithm recursively on $u - \tau_f$.

Example 1. Let us apply `idm` to $u = (1-z_1-z_2)^{-1} - (1-z_1)^{-1} - z_2 - 2z_1z_2$, with $\overline{z_1} = x^{-1}$ and $\overline{z_2} = x^{-2}$. We start with $G = \{1\}$ and find that $[z_1^0 z_2^0]u = 0$. Since $\overline{u-0} \neq 0$, we set $G := \{z_1, z_2\}$ and iterate. We now obtain $M = \{z_1\}$ and again $[z_1^1 z_2^0]u = 0$. This time, the test $\overline{u - [u|(z_2)]} = 0$ is positive, whence z_1 is eliminated from G and we iterate again. Since $[z_1^0 z_2^1]u = 0$ and $\overline{u-0} \neq 0$, we next set $G := \{z_1z_2, z_2^2\}$ and iterate. This yields $M = \{z_1z_2\}$ and $[z_1^1 z_2^1]u = 0$, so one more iteration with $G := \{z_1^2z_2, z_2^2\}$ is needed in order to find the final intermediate set of dominant monomials $\{z_1^2z_2, z_2^2\}$.

3.4 On the computation of Cartesian representations

Lemma 1. *There exists an algorithm, which given a Cartesian representation $u \in \mathfrak{L}_Z^{expr}$ of an infinitesimal germ $\overline{u}$ at infinity computes $Z' = \{z'_1, \cdots, z'_{k'}\}$ and a C-infinitesimal Cartesian representation $u' \in \mathfrak{L}_{Z'}^{expr}$ for $\overline{u}$.*

Proof. We first compute a set of intermediate dominant monomials $G = \{\varphi_1, \cdots, \varphi_m\}$ for u. If all monomials in G are strictly superior to 1, then we can take $Z' = Z$ and $u' = [u|(1)]$. More generally, we can write $u = v_1 + \cdots + v_m$, with $v_i = [u|(\varphi_1, \cdots, \varphi_i)] - [u|(\varphi_1, \cdots, \varphi_{i-1})]$, for $1 \leqslant i \leqslant m$. Putting $v_i = \varphi_i h_i$, each h_i belongs to $\mathfrak{L}_Z^{expr}$ and its support is included in $\mathbb{N}^k$. Now $u' = \varphi_1 h_1 + \cdots + \varphi_m h_m \in \mathfrak{L}_{Z'}^{expr}$, with $Z' = \{z_1, \cdots, z_k, \varphi_1, \cdots, \varphi_m\}$ satisfies the requirements of the lemma. □

Remark. Actually, we can take $k' \leqslant k$, as will easily follow from lemma 3 below.

Example 2. Assume that u' is the Laurent series from figure 1. We can take $\varphi_1 = z_1^{-1}z_2^2, \varphi_2 = z_2$ and $\varphi_3 = z_1^2$. Then we get $v_1 = z_1^{-1}z_2^2(1/(1-z_2))$, $v_2 = z_2((1+z_1)/(1-z_2))$ and $v_3 = z_1^2(1/(1-z_1-z_2))$. We observe that $\overline{u}$ is infinitesimal if $\overline{z_1^{-1}z_2^2}$ is. In that case, an expression like e^u can be expanded in z_1, z_2 and $z_1^{-1}z_2^2$, by using the identity $e^u = \exp(v_1 + v_2 + v_3)$.

Theorem 1. *Let B be an effective normal basis. Then there exists an algorithm which given an expression f in $\mathfrak{G}_B^{expr}$ computes a finite set Z of infinitesimals in S_B, and a Cartesian representation $u \in \mathfrak{L}_Z^{expr}$ for $\overline{f}$.*

Proof. Constants are by definition Cartesian representations of themselves. If $f \in S_B \backslash \{1\}$, then either $f \in \mathfrak{L}_{\{f\}}^{expr}$ or $f \in \mathfrak{L}_{\{1/f\}}^{expr}$. Now assume that u' and u'' are Laurent series in $\mathfrak{L}_{Z'}^{expr}$ and $\mathfrak{L}_{Z''}^{expr}$ respectively, with $Z' = \{z'_1, \cdots, z'_{k'}\}$ and $Z'' = \{z''_1, \cdots, z''_{k''}\}$. Then $u'+u''$, $u'-u''$ and $u'u''$ are Cartesian representations for $\overline{u'}+\overline{u''}$, $\overline{u'} - \overline{u''}$ resp. $\overline{u'u''}$ in $\mathfrak{L}_{Z' \cup Z''}^{expr}$. If $\overline{u'}$ is infinitesimal, then we may assume without loss of generality that u' is C-infinitesimal by lemma 1. Hence, we have straightforward Cartesian representations for $1/(1+\overline{u'}), \log(1+\overline{u'})$ and $\exp \overline{u'}$ in $\mathfrak{L}_{Z'}^{expr}$. □

3.5 Asymptotic expansions via Cartesian representations

Having computed a Cartesian representation u for $\overline{f}$ by theorem 1, we would like to take advantage of u to compute the asymptotic expansion of f.

Lemma 2. *There exists an algorithm, which given a Laurent series u in $\mathfrak{L}_Z^{expr}$ with $\overline{u} \neq 0$ computes $Z' = \{z'_1, \cdots, z'_{k'}\}$ and $u' \in \mathfrak{L}_{Z'}^{expr}$ with $\overline{u'} = \overline{u}$, such that u' has only one dominant monomial.*

Proof. Let $\{\varphi_1, \cdots, \varphi_m\}$ be a set of intermediate dominant monomials for u. Let $c = u_{\varphi_1} + \cdots + u_{\varphi_m}$ and $\varepsilon = (u - u_{\varphi_1}\varphi_1 - \cdots - u_{\varphi_m}\varphi_m)/\varphi_1$. By lemma 1 we can compute a C-infinitesimal Cartesian representation $\varepsilon' \in \mathfrak{L}_{Z'}^{expr}$ for $\overline{\varepsilon}$, where $Z' = \{z'_1, \cdots, z'_{k'}\}$. Now we take $u' = (c + \varepsilon')\varphi_1$. □

Modulo this lemma, we may assume without loss of generality, that u has a unique dominant monomial. The following proposition gives us the first term of the expansion of f w.r.t. b_n:

Proposition 3. *Let $f \in \mathfrak{G}_B^{expr}$ and let $u \in \mathfrak{L}_Z^{expr}$ be a Cartesian representation of $\overline{f}$ with a unique dominant monomial $z_1^{\mu_1} \cdots z_k^{\mu_k}$ and $Z \subseteq S_B$. Let $z_1, \cdots, z_l$ those elements among $z_1, \cdots, z_k$ which depend on b_n, say $z_i = z'_i b_n^{\alpha_i}$ for $1 \leqslant i \leqslant l$, with z'_i free from b_n and $\alpha_i > 0$. Then the dominant exponent of f w.r.t. b_n equals*

$$\mu_f = \mu_1\alpha_1 + \cdots + \mu_l\alpha_l$$

and

$$([z_1^{\mu_1} \cdots z_l^{\mu_l}]u) z_1'^{\mu_1} \cdots z_l'^{\mu_l}$$

is a Cartesian representation for $\overline{[b_n^{\mu_f}]f}$. □

Clearly, this proposition enables us to extract the first term of the expansion of f w.r.t. b_n. More terms can (for instance) be obtained by subtracting the first term from f and iterating the process. Similarly, we can iterate the process on the coefficients of this expansion in order to obtain the iterated coefficients of f. In particular, this yields an algorithm to compute the iterated coefficients $g_{0,\overset{n-i \text{ times}}{\cdots},0}$ of g involved in the exponential case $f = e^g$ in `expand`.

4 An asymptotic zero test for exp-log functions

In this section we no longer assume that we have an oracle for deciding whether an exp-log function is zero in a neighbourhood of infinity. Instead, we assume that we dispose of an oracle which can decide whether an exp-log constant is zero. Such an oracle is in fact an algorithm under the assumption that Schanuel's conjecture holds (see the introduction). Now a zero test for Laurent series in $\mathfrak{L}_Z^{expr}$, which depends on the oracle, is given in [20] (see also [22, 21, 16]).

4.1 Elimination of Cartesian coordinates

Given a Cartesian representation $u \in \mathfrak{L}_Z^{expr}$ of a germ $f \in \mathfrak{T}$, we have $f = 0$ as soon as $u = 0$. Usually, we also have $f = 0 \Rightarrow u = 0$, and in this case the zero test in $\mathfrak{L}_Z^{expr}$ can be used as an asymptotic zero test for exp-log functions. However, in exceptional cases such as $u = z_1^3 - z_2^2$ with $\overline{z_1} = x^{-2}$ and $\overline{z_2} = x^{-3}$, this does not hold. Nevertheless, we may eliminate one of the Cartesian coordinates z_1 and z_2 here, by setting $z_1' = x^{-1}$, whence $z_1 = {z_1'}^2$ and $z_2 = {z_1'}^3$. Rewriting u w.r.t. z_1' yields a Cartesian representation which is identical to zero. In this section, we prove that something similar holds in general.

Lemma 3. *Let B be an effective normal basis and let $z_1, \cdots, z_{k+1}$ be infinitesimals in S_B. Assume that $z_{k+1} = z_1^{\alpha_1} \cdots z_k^{\alpha_k}$ with $\alpha_1, \cdots, \alpha_k \in \mathbb{Z}$. There is an algorithm which computes $z_1', \cdots, z_k' \in S_B$ and a matrix $M = (\beta_{i,j})$ with $i \in \{1, \cdots, k+1\}$, $j \in \{1, \cdots, k\}$ and coefficients in $\mathbb{N}$, such that*

$$z_i = {z_1'}^{\beta_{i,1}} \cdots {z_k'}^{\beta_{i,k}}$$

for all $1 \leqslant i \leqslant k+1$.

Proof. Let us describe a recursive method to compute such z_j' and $\beta_{i,j}$. Since one of the α_i must be strictly positive, we may assume without loss of generality that $\alpha_k > 0$ by permuting variables. Now $z_1^{\alpha_1} \cdots z_{k-1}^{\alpha_{k-1}}$ — as an element of S_B — is either infinitesimal, equal to 1, or infinitely large.

In the first case, we recursively compute $z_1', \cdots, z_{k-1}' \in S_B$ and $\gamma_{i,j} \in \mathbb{N}$ for $i \in \{1, \cdots, k\}$ and $j \in \{1, \cdots, k\}$, such that

$$z_i = {z_1'}^{\gamma_{i,1}} \cdots {z_{k-1}'}^{\gamma_{i,k-1}}$$

for all $1 \leqslant i \leqslant k-1$ and

$$z_1^{\alpha_1} \cdots z_{k-1}^{\alpha_{k-1}} = {z_1'}^{\gamma_{k,1}} \cdots {z_{k-1}'}^{\gamma_{k,k-1}}.$$

Now we can take $z_k' = z_k$ and

$$M = \begin{pmatrix} \gamma_{1,1} & \cdots & \gamma_{1,k-1} & 0 \\ \vdots & & \vdots & \vdots \\ \gamma_{k-1,1} & \cdots & \gamma_{k-1,k-1} & 0 \\ 0 & \cdots & 0 & 1 \\ \gamma_{k,1} & \cdots & \gamma_{k,k-1} & \alpha_k \end{pmatrix}.$$

The second case is trivial, since $z_{k+1} = z_k^{\alpha_k}$.

In the last case, we recursively compute $z'_1, \cdots, z'_{k-1} \in S_B$ and $\gamma_{i,j} \in \mathbb{N}$ for $i \in \{1, \cdots, k\}$ and $j \in \{1, \cdots, k\}$, such that

$$z_i = {z'_1}^{\alpha_k \gamma_{i,1}} \cdots {z'_{k-1}}^{\alpha_k \gamma_{i,k-1}}$$

for all $1 \leqslant i \leqslant k-1$ and

$$z_1^{\alpha_1} \cdots z_{k-1}^{\alpha_{k-1}} = {z'_1}^{-\alpha_k \gamma_{k,1}} \cdots {z'_{k-1}}^{-\alpha_k \gamma_{k,k-1}}.$$

Now take $z'_k = z_k {z'_1}^{\alpha_1} \cdots {z'_{k-1}}^{\alpha_{k-1}}$ and

$$M = \begin{pmatrix} \alpha_n \gamma_{1,1} & \cdots & \alpha_n \gamma_{1,k-1} & 0 \\ \vdots & & \vdots & \vdots \\ \alpha_n \gamma_{k-1,1} & \cdots & \alpha_n \gamma_{k-1,k-1} & 0 \\ \gamma_{k,1} & \cdots & \gamma_{k,k-1} & 1 \\ 0 & \cdots & 0 & \alpha_k \end{pmatrix}.$$

□

The following is an easy consequence of the lemma:

Lemma 4. *Let B be an effective normal basis and let $z_1, \cdots, z_{k+1}$ be infinitesimals in S_B. Assume that $z_1^{\alpha_1} \cdots z_k^{\alpha_k} = 1$, for $\alpha_1, \cdots, \alpha_k \in \mathbb{Z}$ not all zero. There is an algorithm which computes $z'_1, \cdots, z'_{k-1} \in S_B$ and a matrix $M = (\beta_{i,j})$ with $i \in \{1, \cdots, k\}$, $j \in \{1, \cdots, k-1\}$ and coefficients in $\mathbb{N}$, such that*

$$z_i = {z'_1}^{\beta_{i,1}} \cdots {z'_{k-1}}^{\beta_{i,k-1}}$$

for all $1 \leqslant i \leqslant k$. □

4.2 The algorithm

Theorem 2. *Assuming Schanuel's conjecture, there exists an algorithm which given an exp-log expression $f \in \mathfrak{T}^{expr}$*

(a) computes an effective normal basis B for f.
(b) computes an asymptotic expansion for f w.r.t. B at any order.
(c) determines the sign of f.
(d) determines whether f is infinitesimal.

Proof. In view of what precedes, we only have to show how to decide whether $\overline{u} = 0$ for a given $u \in \mathfrak{L}_Z^{expr}$, with the notations from the previous section. To do this, we slightly modify `idm`:

Algorithm `zero_test`(u).
INPUT: A Cartesian representation $u \in \mathfrak{L}_Z^{expr}$ for some Z.
OUTPUT: Result of the test $\overline{u} = 0$.

if $u = 0$ **then return true**
• Let $z_i^{p_i}$ be the dominant monomial of u in z_i, for $1 \leqslant i \leqslant k$.
$G := \{z_1^{p_1} \cdots z_k^{p_k}\}$
while true
  $M := \{\varphi \in G | \forall \psi \in G \quad \varphi \preccurlyeq_B \psi\}$
  if $|M| > 1$ **then return** `zero_test`(`simplify`(u, M))
  if $\sum_{\varphi \in M} u_\varphi \neq 0$ **then return false**
  • Denote $G \backslash M = \{\varphi_1, \cdots, \varphi_q\}$, with $\varphi_1 \preccurlyeq_B \cdots \preccurlyeq_B \varphi_q$.
  if $\exists 0 \leqslant i \leqslant q \quad u - [u|(\varphi_i, \cdots, \varphi_q)] = 0$ **then** $G := \{\varphi_i, \cdots, \varphi_q\}$
  else $G := (G \backslash M) \cup M\{z_1, \cdots, z_k\}$
  • Eliminate non minimal elements from G.

Let us comment this algorithm. All zero tests we perform are zero tests for Laurent series. If the cardinal $|M|$ of M never exceeds 1, then the usual termination proof of `idm` remains valid and we are done. In the other case, the function `simplify` is invoked, which undertakes the following action:

Step 1. Determine a non trivial relation of the form $z_1^{\alpha_1} \cdots z_k^{\alpha_k} = 1$ in S_B, with $\alpha_1, \cdots, \alpha_k \in \mathbb{Z}$.

Step 2. Apply lemma 4 to find infinitesimals $z'_1, \cdots, z'_{k-1} \in S_B$ and positive integers $\beta_{i,j}$ with $z_i = {z'_1}^{\beta_{i,1}} \cdots {z'_{k-1}}^{\beta_{i,k-1}}$ for each i.

Step 3. Return u after having replaced each z_i by ${z'_1}^{\beta_{i,1}} \cdots {z'_{k-1}}^{\beta_{i,k-1}}$.

The recursive call of `zero_test` terminates, since $Z' = \{z'_1, \cdots, z'_{k-1}\}$ has one element less than Z. □

Remark. A heuristic zero test for Laurent series u in $\mathfrak{L}_Z^{expr}$ often suffices for practical purposes: we perform a floating point evaluation of u in a random point $(\zeta_1, \cdots, \zeta_k)$ with reasonably small ζ_i. Instead of rewriting u in the above algorithm, whenever we find a dependency $z_1^{i_1} \cdots z_k^{i_k} = 1$, we use these dependencies to impose a posteriori additional conditions on the ζ_i.

5 Generic expansions

Often, one is lead to expand functions which depend on a finite number of real parameters. For instance, consider the problem of expanding $e^{e^{\lambda x}}$: if $\lambda < 0$, then $e^{e^{\lambda x}} = 1 + e^{\lambda x} + e^{2\lambda x}/2 + \cdots$. If $\lambda > 0$, then $e^{e^{\lambda x}}$ forms its own expansion. The same situation is encountered when solving differential equations, due to the presence of initial conditions. In this section we show that the expansion algorithm for exp-log functions can be made generic, by using a technique called dynamic evaluation [5, 9]. This technique is the computer algebra version of constraint logical programming. Of course, the resolution techniques used here can not be the same as in logical programming.

5.1 Notations and terminology

Let C be a subset of $\mathbb{R}$ and Λ a set of formal parameters. We denote by $C\langle\Lambda\rangle$ the set of *exp-log expressions* over C in Λ — i.e. those expressions built up from C, Λ by $+, -, \cdot, /, \exp$ and $\log$. The *domain* of such an exp-log expression f is the subset $\operatorname{dom} f$ of $\mathbb{R}^\Lambda$, consisting of those substitutions $\varphi : \Lambda \to \mathbb{R}$, such that $\varphi(f)$ is naturally defined. Let us now consider *systems* $\Sigma = (\Sigma_e, \Sigma_i)$ of exp-log equalities $\Sigma_e \subseteq C\langle\Lambda\rangle$ and inequalities $\Sigma_i \subseteq C\langle\Lambda\rangle$. We will refer to such systems as *exp-log systems* over C in Λ. The *domain* of such a system Σ is defined by $\operatorname{dom}\Sigma = \bigcap_{f\in\Sigma_e\cup\Sigma_i} \operatorname{dom} f$. We say that a substitution $\varphi : \Lambda \to \mathbb{R}$ in $\operatorname{dom}\Sigma$ is a *solution* to Σ, if $\varphi(f) = 0$, for each $f \in \Sigma_e$, and $\varphi(f) > 0$, for each $f \in \Sigma_i$.

Let R be a subset or *region* of $\mathbb{R}^\Lambda$. An exp-log expression f over C in $x, \lambda_1, \cdots, \lambda_p$ is said to be defined at infinity relative to R, if for each $P \in \mathbb{R}^\Lambda$, the substitutions $\lambda_1 \mapsto P(\lambda_1), \cdots, \lambda_p \mapsto P(\lambda_p)$ in f determine a germ of an exp-log function at infinity. In a similar way, we define effective asymptotic scales, asymptotic expansions, effective normal bases, etc. relative to R. Assume that we are given a partition

$$\mathbb{R}^\Lambda = R_1 \amalg \cdots \amalg R_r \tag{5}$$

of $\mathbb{R}^\Lambda$ and an exp-log expression f over C in $x, \lambda_1, \cdots, \lambda_p$, which is either defined or undefined at infinity relative to R_i, for each $1 \leqslant i \leqslant r$. Then by *generic asymptotic expansion* of f relative to the partition (5), we mean a set of asymptotic expansions of f relative to each region where f is defined at infinity. In a similar way, we define generic effective asymptotic scales, normal bases, etc. relative to (5).

5.2 The generic expansion theorems

In this section, we show how to compute generic asymptotic expansions of parameterized exp-log functions. We first give an algorithm which may yield *virtual asymptotic expansions*, i.e. expansions which are valid on empty regions R_i in the partition (5). Modulo an algorithm to test the consistency of exp-log systems over the rationals, such empty regions may be eliminated.

Theorem 3. (Generic expansion theorem, weak form) *Let $\Lambda = \{\lambda_1, \cdots, \lambda_p\}$ be a finite set of parameters. There exists an algorithm which takes an exp-log expression in $x, \lambda_1, \cdots, \lambda_p$ over $\mathbb{Q}$ on input and computes*

(a) A partition $\mathbb{R}^\Lambda = R_1 \amalg \cdots \amalg R_r$ of $\mathbb{R}^\Lambda$, which we denote by P;
(b) A generic effective normal basis B relative to P;
(c) An algorithm which computes the generic asymptotic expansion of f w.r.t. B relative to P at any order.

Each (possibly empty) region R_i is represented as the solution set to a system Σ_i of exp-log equalities and inequalities.

Proof. We use the strategy of dynamic evaluation. If we want the algorithms from sections 2 and 3 to work without any modifications, when we allow the exp-log expressions to depend on parameters $\lambda_1, \cdots, \lambda_p$, then we need to implement the exp-log field operations and the equality and inequality tests in $\mathbb{Q}\langle\Lambda\rangle$. The exp-log operations can be implemented in a straightforward way, since $\mathbb{Q}\langle\Lambda\rangle$ consists of expressions. Actually, we represent these expressions as polynomials in a dynamic polynomial ring $\mathbb{Q}[\lambda_1, \cdots, \lambda_q]$, where $q \geqslant p$ may increase during the algorithm. Each time we need the exponential, the logarithm, or the inverse of an element, we represent it by a new formal parameter λ_i.

In order to describe the equality and inequality tests, we adopt a *parallel computational model*, which proves to be more convenient than the usual sequential model. We introduce a new global variable Σ, which is initialized by $(\emptyset, \emptyset)$ and represents the system of exp-log equalities and inequalities which are assumed to be verified at each moment during the execution. These exp-log equalities and inequalities are actually polynomial equalities and inequalities in $\mathbb{Q}[\lambda_1, \cdots, \lambda_q]$. Now suppose that we want to compute the sign of $f \in \mathbb{Q}[\lambda_1, \cdots, \lambda_q] \subseteq \mathbb{Q}\langle\Lambda\rangle$ in a given environment. Then we divide the current process up into three subprocesses, in which we respectively assign $\Sigma := (\Sigma_e \cup \{f\}, \Sigma_i)$, $\Sigma := (\Sigma_e, \Sigma_i \cup \{f\})$ and $\Sigma := (\Sigma_e, \Sigma_i \cup \{-f\})$. A process is eliminated, whenever the new system Σ is *algebraically* inconsistent; this can be tested for instance by using cylindrical decompositions [3]. Each process leads to an effective normal basis B and an expansion algorithm for f w.r.t. B relative to the solution set of Σ. Processes in which an error occurs correspond to regions relative to which f is undefined at infinity.

We notice that the parallel computation process can be represented by a ternary tree, which is called the *computation tree*: each internal node of this tree is labeled by an exp-log expression $c \in \mathbb{Q}\langle\Lambda\rangle$ and its outgoing edges by the constraints $c < 0, c = 0$ and $c > 0$. Each leaf of the tree is labeled by an effective normal basis B and an expansion algorithm for f w.r.t. B relative to the common solution set to the constraints on the path from the root to the leaf. In order to prove the termination of our algorithm, we have to show that the computation tree is finite. Because of Königs lemma [19], it suffices to prove that the computation tree admits no infinite branches.

Suppose on the contrary that there exists a non terminating process. Careful observation shows that the non termination comes from the loop in the algorithm `idm` or `zero_test`. Now u is a Laurent series with coefficients in $\mathbb{Q}[\lambda_1, \cdots, \lambda_q]$ in these algorithms, for *fixed* q. Let $c_1, c_2, \cdots$ denote the successive values of $\Sigma_{\varphi \in M} u_\varphi$ during the loop. Since $\mathbb{Q}[\lambda_1, \cdots, \lambda_q]$ is Noetherian, the chain of ideals $(c_1), (c_1, c_2), \cdots$ is stationary. In particular, $c_i = 0$ follows from Σ, for sufficiently large i, and the usual termination argument is used to obtain a contradiction. □

Remark. In order to apply the algorithms from section 4, we still have to verify that the zero test for Laurent series remains valid if we allow them to depend on a

finite number of parameters. This is not hard to show and we refer to [16] for details. We remark that some new but finite branching may occur during the execution of such a generic zero test.

Remark. By default, we take $(\emptyset, \emptyset)$ for the initial value of Σ. Clearly, we can make additional hypotheses on the parameters, by taking other initial values.

Remark. It is in place to comment the parallel computational model we use. To our knowledge, no computer algebra systems support parallel constructs yet. Nevertheless, parallelism can be simulated, by replacing the arguments to functions (and the return values) by lists in which each item corresponds to a parallel environment plus the corresponding values of the arguments. Another way to simulate parallelism is to rerun the program several times, by choosing each time another branch of the computation tree. This strategy has the advantage that no code has to be rewritten; the price to be paid is that the same computations are often performed several times.

If the constraints in Σ can be tested for their exp-log consistency, then we may eliminate the empty regions R_i in the partition (5):

Theorem 4. (Generic expansion theorem, strong form) *Let $\Lambda = \{\lambda_1, \cdots, \lambda_p\}$ be a finite set of parameters. Assume that we have an oracle which decides whether a given system of exp-log equalities and inequalities over $\mathbb{Q}$ in any finite set of parameters admits a solution. Then there exists an algorithm which takes an exp-log expression in $x, \lambda_1, \cdots, \lambda_p$ over $\mathbb{Q}$ on input and computes*

(a) A partition $\mathbb{R}^\Lambda = R_1 \amalg \cdots \amalg R_r$ of $\mathbb{R}^\Lambda$, which we denote by P;
(b) A generic effective normal basis B relative to P;
(c) An algorithm which computes the generic asymptotic expansion of f w.r.t. B relative to P at any order.

Each (non empty) region R_i is represented as the solution set to a system Σ_i of exp-log equalities and inequalities. □

Example 3. Let us consider the expansion of the exp-log function

$$f(x) = e^{1/x + e^{\lambda x}} - e^{1/x},$$

depending on one formal parameter λ. The expansions of $1/x$ and λx are straightforward. For the expansion of $e^{\lambda x}$, one needs to compute the sign of λx and thus of λ. This leads to a branching into three processes, corresponding to the cases $\lambda < 0$, $\lambda = 0$ and $\lambda > 0$. The first case leads to the expansion

$$f = b_2 + b_1 b_2 + \frac{1}{2} b_1^2 b_2 + \cdots + \frac{1}{2} b_2^2 + \frac{1}{2} b_1 b_2^2 + \cdots,$$

with effective normal basis $B = \{b_1 = x^{-1}, b_2 = e^{\lambda x}\}$. The second case leads to the expansion

$$f = (e-1) + (e-1)b_1 + \frac{e-1}{2}b_1^2 + \cdots ,$$

where $B = \{b_1 = x^{-1}\}$. Finally, the case $\lambda > 0$ leads to the expansion

$$f = b_3 + b_1 b_3 + \frac{1}{2}b_1^2 b_3 + \cdots - 1 - b_1 - \cdots ,$$

with $B = \{b_1 = x^{-1}, b_2 = e^{-\lambda x}, b_3 = e^{-e^{\lambda x}}\}$.

6 References

References

[1] R.P. Brent and H.T. Kung. $o((n \log n)^{3/2})$ algorithms for composition and reversion of power series. In J.F. Traub, editor, *Analytic Computational Complexity*, 1975. Proc. of a symposium on analytic computational complexity held by Carnegie-Mellon University.

[2] R.P. Brent and H.T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25:581–595, 1978.

[3] G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proc. 2-nd conf. on automata theory and formal languages*, volume 33 of *Lect. Notes in Comp. Science*, pages 134–183. Springer, 1975.

[4] B.I. Dahn and P. Göring. Notes on exponential-logarithmic terms. *Fundamenta Mathematicae*, 127:45–50, 1986.

[5] J. Della Dora, C. Dicrescenzo, and D. Duval. A new method for computing in algebraic number fields. In G. Goos and J. Hartmanis, editors, *Eurocal'85 (2)*, volume 174 of *Lect. Notes in Comp. Science*, pages 321–326. Springer, 1985.

[6] J. Denef and L. Lipshitz. Decision problems for differential equations. *The Journ. of Symb. Logic*, 55(3):941–950, 1989.

[7] J. Écalle. *Introduction aux fonctions analysables et preuve constructive de la conjecture de Dulac*. Hermann, collection: Actualités mathématiques, 1992.

[8] K.O. Geddes and G.H. Gonnet. A new algorithm for computing symbolic limits using hierarchical series. In *Proc. ISSAC '88*, volume 358 of *Lect. Notes in Comp. Science*, pages 490–495. Springer, 1988.

[9] T. Gomez-Diaz. *Quelques applications de l'évaluation dynamique.* PhD thesis, Univ. of Limoges, France, 1994.

[10] G.H. Gonnet and D. Gruntz. Limit computation in computer algebra. Technical Report 187, ETH, Zürich, 1992.

[11] G.H. Hardy. *Orders of infinity.* Cambridge Univ. Press, 1910.

[12] G.H. Hardy. Properties of logarithmico-exponential functions. *Proceedings of the London Mathematical Society*, 10(2):54–90, 1911.

[13] J. van der Hoeven. General algorithms in asymptotics I: Gonnet and Gruntz' algorithm. Technical Report LIX/RR/94/10, LIX, École polytechnique, France, 1994.

[14] J. van der Hoeven. Outils effectifs en asymptotique et applications. Technical Report LIX/RR/94/09, LIX, École polytechnique, France, 1994.

[15] J. van der Hoeven. Automatic numerical expansions. In J.-C. Bajard, D. Michelucci, J.-M. Moreau, and J.-M. Müller, editors, *Proc. of the con,ference "Real numbers and computers", Saint-Étienne, France*, pages 261–274, 1995.

[16] J. van der Hoeven. *Automatic asymptotics.* PhD thesis, École polytechnique, France, 1997.

[17] J. van der Hoeven. Lazy multiplication of formal power series. In W. W. Küchlin, editor, *Proc. ISSAC '97*, pages 17–20, Maui, Hawaii, July 1997.

[18] D.E. Knuth. *The art of computer programming*, volume 2: Seminumerical algorithms. Addison-Wesley, 2-nd edition, 1981.

[19] D. König. *Theorie der endlichen und unendlichen Graphen.* Chelsea publ. Comp., New York, 1950.

[20] A. Péladan-Germa. Testing identities of series defined by algebraic partial differential equations. In G. Cohen, M. Giusti, and T. Mora, editors, *Proc. of AAECC-11*, volume 948 of *Lect. Notes in Comp. Science*, pages 393–407, Paris, 1995. Springer.

[21] A. Péladan-Germa. *Tests effectifs de nullité dans des extensions d'anneaux différentiels.* PhD thesis, Gage, École Polytechnique, Palaiseau, France, 1997.

[22] A. Péladan-Germa and J. van der Hoeven. A local buchberger algorithm. *Accepted for publication in the CRAS*, 1996.

[23] D. Richardson. The elementary constant problem. In *Proc. ISSAC '92*, pages 108–116, 1992.

[24] D. Richardson. A simplified method for recognizing zero among elementary constants. In *Proc. ISSAC '95*, pages 104–109, 1995. See also "How to recognize zero", accepted for publication in JSC.

[25] D. Richardson, B. Salvy, J. Shackell, and J. van der Hoeven. Expansions of exp-log functions. In Y.N. Lakhsman, editor, *Proc. ISSAC '96*, pages 309–313, Zürich, Switzerland, July 1996.

[26] R.H. Risch. Algebraic properties of elementary functions in analysis. *Amer. Journ. of Math.*, 4(101):743–759, 1975.

[27] B. Salvy. *Asymptotique automatique et fonctions génératrices.* PhD thesis, École Polytechnique, France, 1991.

[28] J. Shackell. A differential-equations approach to functional equivalence. In *Proc. ISSAC '89*, pages 7–10, Portland, Oregon, A.C.M., New York, 1989.

[29] J. Shackell. Growth estimates for exp-log functions. *Journal of symbolic computation*, 10:611–632, 1990.

[30] J. Shackell. Limits of Liouvillian functions. *Proc. of the London Math. Soc.*, 72:124–156, 1996. Appeared in 1991 as a technical report at the Univ. of Kent, Canterbury.