TECHNISCHE
UNIVERSITÄT
DARMSTADT

Bachelor-Thesis

Data structures and efficient algorithms for power series in exact real arithmetic

Julian Bitterlich

May 2012

under the guidance of

Prof. Dr. Martin Ziegler

Contents

Abstract

On the basis of exact real arithmetic, we define a naming system for power series in 0. This renders evaluation of some point in $\mathbb{C}$, determing the maximum on a ball (other geometrical objects are also possible), addition, multiplication, differentiation and integration computable. The aim of this work is to give a background for a new datatype for analytic functions which can be implemented in iRRAM.

Chapter 1

Introduction

The following work is based on the model TTE (Type 2 Theory of Effectivity). TTE is a computational model which gives us a computability notion for exact complex analysis. For an introduction see [Wei00]. We will use the definitions of "notation, representation and naming systems" as in [Wei00]:

Definition 1.0.1.

1. A notation of a set X is a surjective mapping $\nu :\subseteq \Sigma^* \to X$.
2. A representation of a set X is a surjective mapping $\delta :\subseteq \Sigma^\omega \to X$.

A naming system is either a notation or a representation.

We started this work with the aim of finding a representation of analytic functions f, based on the power series expansion of f in different points. We realized that we could not achive a simple representation in a satisfactional way such that primitive functions like addition and multiplication are easily computable. So we restricted our approach to find a representation of power series in 0. For this we will use a suitable restriction of products of naming systems, which by [Wei00] are itselves naming systems. Therefore we introduce the basic naming systems for real, complex and sequences of complex numbers, so we can use these freely (clearly also $\mathbb{N}, \mathbb{Q}$) for our naming system.

Definition 1.0.2. The naming system δ for the real numbers is defined as follows: For all $w \in \Sigma^*$, we have that $\delta(w) = x \in \mathbb{R}$ iff w codes a sequence of intervals with rational endpoints (a_n, b_n), s.t. $x \in (a_n, b_n)$ for all n and $\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n = x$.

The naming system $\delta_{\mathbb{C}}$ for the complex numbers is given by the product $[\delta, \delta]$ because we can identify $\mathbb{C}$ with $\mathbb{R}^2$.

The naming system for $\mathbb{C}^{\mathbb{N}}$ is given by $[\delta_{\mathbb{C}}]^\omega$.

The upper discussion is the most abstract part of this paper. In fact it is just for the mathematical conscience. This work can also be viewed (and this is the real purpose of this paper) as a discussion for a datatype for the C++ package iRRAM (A documentation can be found at [Mue08]) which is based on the exact real arithmetic

and possesses the datatype REAL for (evidently) real numbers. This datatype REAL simulates the representation of real numbers of definition 1.0.1 by storing real numbers with a certain error bound, which can be viewed as a finite prefix of w. The show-stopper about iRRAM is, if one only use computable functions (of course they have to be implemented in iRRAM), one can neglect error bounds in any calculation, since iRRAM adjusts those internally. But the set of computable functions exclude = et cetera. Another thing to mention is that if we want to render a function f computable, we only have to find an algorithm which gives, for given x and $\varepsilon > 0$, an approximation of $f(x)$ up to an error ε. This is in accordance to the theory of exact real arithmetic and also (clearly) with iRRAM.

In chapter 2 we give some motivation, how we will choose our representation and eventually its definition. But we have to say that a big part of this representation is pure intuition.

In chapter 3 we give an algorithm for the evaluation. This was the part with the most workload and is also the main content of this work. For better understanding we have split the algorithm into small subtasks. In 3.3 we say some words to the complexity of the algorithm, but there will be no determination of the running time.

In chapter 4 we will give an ad-hoc definition of shapes and discuss an algorithm which determines the maximum of a given power series on a shape. This will be the only time we will use the strong results for analytic functions.

In chapter 5 we will give some algorithms for addition,multiplication,differentiation and integration. The concatenation of power series was to complicated and no results have been achieved.

Chapter 2

Datatype

We want to develop a naming system δ_p for power series in 0, s.t. we can compute the evaluation of this power series. In fact the naming system, we will use later, does not have this feature, but a weaker form of this, i.e. the domain of the evaluation algorithm for a given power series depends on the name of this power series.

Furthermore we want addition, multiplication, differentiation and integration to be computable in this naming system. And if procurable we want to avoid infinite data, since in implementations they become functions and usually lead to dependencies when we worke with them:

For example consider the naming system for the real numbers (Cauchy representation). Let $a, b \in \mathbb{R}$ be given, this means two procedures which gives us a cauchy representation of a resp. b. We want to work with $c = a + b$. If we want to know c up to a precision ε, we have to query the names of a and b for suitable approximations of a and b. This function nesting becomes more complex if we use longer terms. One can think of it, as each number has to know where it comes from.

In contrast therefore the standard naming system for rational numbers is a finite datatype. If we add two rationals we get a new fraction and it does not matter "where it came from".

2.1 Foregoing consideration

When developing the naming system for power series $f(x) = \sum_{i=0}^{\infty} a_n x^n$ we should also consider how to evaluate these later. For evaluation the use of partial sums seem appropriate. An evaluation algorithm can look like this:

Scheme of evaluation

```
1 \\Contract:
2 \\Input:  Name w of some power series f(x) = ∑_{i=0}^{∞} a_n x^n with
      radius of convergence ρ, z ∈ B_ρ(0) and some ε > 0.
3 \\Output: Some real y with |f(z) − y| ≤ ε.
4
```

5 `Determine with the help of` w `an` $M \in \mathbb{N}$ `s.t.` $\sum_{n=M}^{\infty} |a_n||z|^n \leq \varepsilon$

6 `Return` $\sum_{n=0}^{M-1} a_n z^n$

An immediate and reasonable approach is to take the sequence of coefficients a_n as one part of the naming system (this is some infinite data) and some bounding function F for the coefficients, for which we can compute an $M \in \mathbb{N}$ for a given z in the radius of convergence of f, s.t. $\sum_{n=M}^{\infty} F(n)|z|^n \leq \varepsilon$, as another. As a first guess we might take $F(n) = \frac{1}{\rho^n}$ with ρ being the radius of convergence of f (assuming it to be finite). But in general we have $|a_n| > \frac{1}{\rho^n}$ for infinitely many n. We could "repair" this in two ways: Use suitable subexponential functions s s.t. $|a_n| \leq \frac{s(n)}{\rho^n}$, but then we also have to somehow know the global behavior of s. In the other approach we use an $R \leq \rho$ and some constant A s.t. $|a_n| \leq \frac{A}{R^n}$. Then we know the global behavior of our bounding function, but we can only evaluate z with $|z| < R$, since for any $R < |z| < \rho$ and M we have $\sum_{n=M}^{\infty} \frac{A}{R^n}|z|^n = \infty$.

We will use the second method, to some extend combined with the first, where we restrict s to be polynomial of the form $A + Bn^\ell$. This form of s will also be usefull for differentiation.

2.2 Datatype

Definition 2.2.1. Let $f(x) = \sum_{i=0}^{\infty} a_n x^n$ be a power series. We say (R, c, A, B, ℓ, N) is a name of f if

- $R \in \mathbb{Q}^+ \setminus \{0\}$,
- $c \in \mathbb{R}^{\mathbb{N}}$ and $c(n) = a_n$,
- $A, B \in \mathbb{Q}^+; \ell, N \in \mathbb{N}$ and for all $n \in \mathbb{N}$ with $n \geq N$ it holds $|c(n)| \leq R^{-n}(A + Bn^\ell)$.

Remark 2.2.2. The given domains for A, B, R are rather suggestions than necessary. One could also use as domain for A and B the natural numbers. This would be convenient for calculating an upper bound for the logarithm of A and B which we could use for evaluation. And as domain for R we could use the real numbers $\mathbb{R}$.

In fact some algorithms, we will see, produce names (R, c, A, B, ℓ, N), s.t. A, B are irrational. We can easily fix this by taking suitable rational upper bounds instead of the calculated A, B.

We intentionally forbid negative A because otherwise easy estimations, we want to use, would be wrong.

The domain of the evaluation algorithm, we will introduce later, will depend on the name (R, c, A, B, ℓ, N) of the power series and is given by $B_R(0)$. From definition 2.2.1 follows that $R \leq \tau$, if we set τ as the radius of convergence of f. So one can easily show that for some power series there is no name for which we can evaluate the series on the whole area of convergence, even if we allow $R \in \mathbb{R}$.
For example, the power series $f = \sum_{n=0}^{\infty} e^{\sqrt{n}} x^n$ has radius of convergence $\rho = 1$ but $e^{\sqrt{n}}$

growths faster than any polynomial, so we cannot find A, B, N s.t. for $n \geq N$ it holds $e^{\sqrt{n}} \leq 1^{-n}(A + Bn^{\ell})$.
Another example is $g = \sum_{n=0}^{\infty} \frac{1}{n!} x^n$ with radius of convergence $\rho = \infty$, since for any given name (R, c, A, B, ℓ, N) we can only evaluate the series on a bounded ball.

Chapter 3

Evaluation

For the following section we fix a power series $f(x) = \sum_{i=0}^{\infty} a_n x^n$ with name (R, c, A, B, ℓ, N), $A \neq 0, B \neq 0, \ell \neq 0$ and $z \in B_R(0)$. Furthermore we will frequently use the notations

$$q := \frac{|z|}{R} < 1$$

and

$$D(M) := \sum_{n=M}^{\infty} Aq^n + Bn^{\ell}q^n.$$

In this chapter we want to design an algorithm for determing $f(z)$, we call this the evaluation algorithm.

As we mentioned above, we treat evaluation for the case $A \neq 0, B \neq 0, \ell \neq 0$. In the other cases, one can easily adapt the following algorithm. So we will not state that we treat the case $A \neq 0, B \neq 0, \ell \neq 0$ in the following listings and remarks.

Some general remarks towards the evaluation algorithm have already been made in remark 2.2.2. We want to add that the algorithm will diverge in iRRAM or will produce an error, if we want to evaluate f in a point in $B_R(0)^C$.

3.1 The evaluation algorithm

3.1.1 Outline of the evaluation algorithm

The general outline of our algorithm will be

Listing 3.1.1.: Outline of evaluation

```
1 \\Contract:
2 \\Input:  Name (R,c,A,B,ℓ,N) of some power series
      f(x) = ∑_{i=0}^{∞} a_n x^n , z ∈ B_R(0) and some ε > 0.
3 \\Output: Some real y with |f(z) − y| ≤ ε.
4
```

```
5   Get M from listing 3.1.2. \\Determines an M ∈ ℕ∩[0,N] s.t.
        D(M) ≤ ε; we call this "first step of evaluation"
6   Return Σ_{n=0}^{M-1} c(n)z^n.
```

This algorithm just computes an M s.t.

$$|f(z) - \sum_{n=0}^{M-1} c(n)z^n| = |\sum_{n=M}^{\infty} c(n)z^n| \leq \sum_{n=M}^{\infty} |c(n)||z|^n \leq \sum_{n=M}^{\infty} Aq^n + Bn^\ell R^{-n}|z|^n =: D(M) \leq \varepsilon$$

and returns

$$\sum_{n=0}^{M-1} c(n)z^n.$$

This shows that the algorithm is correct. So it remains to design the first step of evaluation.

Remark 3.1.1. The M determined in the upper algorithm does not depend on the actual form of z but on $|z|$. And for two values z, z' with $|z| \geq |z'|$ we have for any $M \in \mathbb{N}$

$$\sum_{n=M}^{\infty} A\left(\frac{|z|}{R}\right)^n + Bn^\ell \left(\frac{|z|}{R}\right)^n \leq \varepsilon \Rightarrow \sum_{n=M}^{\infty} A\left(\frac{|z'|}{R}\right)^n + Bn^\ell \left(\frac{|z'|}{R}\right)^n \leq \varepsilon.$$

So if one want to determine $f(z_i)$ for different z_i $(1 \leq i \leq n \in \mathbb{N})$, one only has to find a suitable M for $\max_i |z_i|$. After that one can take this M to evaluate the different points in the way like listing 3.1.1. does. This is of special interest when determing the maximum on shapes.

3.1.2 First step of evaluation

It would be optimal if listing 3.1.2. computes the smallest $M_{best} \in \mathbb{N} \cap [0, N]$ s.t. $D(M_{best}) \leq \varepsilon$. We could not develop a formula which gives us such an M_{best}. So our approach will be to approximate this M_{best} via binary search and determine whether $D(M) \leq \varepsilon$ or not for a given M. But since "$\leq$" is not computable we cannot check for a given M whether $D(M) \leq \varepsilon$, so we have to use multivalued decision which can give "wrong" results. Furthermore, instead of calculating $D(M)$, it might be better to use an easier calculable upper bound D_{up} of D. This means that for our binary search we use a predicate C with $C(M) \Rightarrow D(M) \leq \varepsilon$ rather than $C(M) \Leftrightarrow D(M) \leq \varepsilon$. So we have to be satisfied with results for M which are bigger than M_{best}.

Listing 3.1.2.: First step of evaluation

```
1 \\Contract:
2 \\Input:  Name (R, c, A, B, ℓ, N) of some power series
      f(x) = Σ_{i=0}^{∞} a_n x^n, z ∈ B_R(0) and some ε > 0.
```

```
\\Output:  Some M ∈ [N,∞] ∩ ℕ with D(M) ≤ ε.
  Get M_up from listing 3.1.3. \\Determine an M_up ≥ N with
     D(M_up) ≤ ε
  Get M_low from listing 3.1.4. \\Determine an M_low ≥ N with
     D(M_low) ≥ ε or M_low = N.
  while(M_up ≠ M) do {
    M := ⌈(M_up + M_low)/2⌉
    if(C(M)){ \\this implies D(M) ≤ ε
      M_up := M
    }else{
      M_low := M
    }
  }
  Return M
```

3.1.3 How to calculate M_{up}

Listing 3.1.3.: How to calculate M_{up}

```
\\Contract:
\\Input:  Name (R, c, A, B, ℓ, N) of some power series
    f(x) = Σ_{i=0}^{∞} a_n x^n, z ∈ B_R(0) and some ε > 0.
\\Parameter: w ∈ ℕ\{0}
\\Output:  Some M_up ∈ [N,∞] ∩ ℕ with D(M_up) ≤ ε.
  M_up := ⌈max{N,
            ln(2) (−log_2(ε/2) − log_2(1−q) + log_2(A)) / (1−q),
            2ℓ/(1−q),
            (w ln^2(2) e)/(w ln(2) e − 1) (−log_2(ε/2) − log_2(1−q) + log_2(2B) + ℓ log_2(wℓ) − ℓ log_2(1−q)) / (1−q)}⌉
  Return M_up
```

Remark 3.1.2. In fact $\lceil\cdot\rceil$ is not computable, but if we use a multivalued version which either returns $\lceil\cdot\rceil$ or $\lceil\cdot\rceil + 1$, this becomes computable.

This algorithm works with any parameter $w \in \mathbb{N}\setminus\{0\}$. We want to dicuss which w would be the best one. Since w only occurs in

$$\frac{w\ln^2(2)e}{w\ln(2)e-1}\,\frac{-\log_2(\frac{\varepsilon}{2})-\log_2(1-q)+\log_2(2B)+\ell\log_2(w\ell)-\ell\log_2(1-q)}{1-q}$$

we want to minimise this expression. If we use the abbreviations

$$E = \frac{\ell}{1-q}$$
$$F = \frac{-\log_2(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(2B)}{1-q}$$
$$r = \frac{w\ln(2)e - 1}{w\ln(2)e}$$

and relax $r \in (0,1)$, then this task becomes to minimise the expression

$$\min_{r\in(0,1)} g(r) := \frac{\ln(2)}{r}(E\log_2\left(\frac{E}{\ln(2)e(1-r)}\right) + F).$$

E is positive. The minimum exists iff $e\log(2) \leq E2^{\frac{F}{E}}$. Using Lamberts W-Function [RMCJ96] we can give a formula for the point r_{min} where the minimum is attained

$$r_{min} = 1 + \left(W_{-1}(-\frac{ln(2)}{E2^{\frac{F}{E}}})\right)^{-1}.$$

Since $W_{-1}(x)$ goes to $-\infty$ when x goes to 0. r_{min} goes to 1 if E, F increase to ∞. But on the other hand for fixed E, F $g(r)$ goes to ∞ if r goes to 1, but plots of r suggest that this growth is rather slow. So values of $r = 0.8$ or $r = 0.9$ seem appropriate, which correspond to values for w in $\{2, \ldots, 6\}$ (Which is a purely heuristic reasoning). If $e\log(2) > E2^{\frac{F}{E}}$ than $g(r)$ goes to $-\infty$ if r goes to 0, but again plots suggest that in this cases the values $g(0.8), g(0.9)$ are also small. If one does not like this heuristic, one can make a database with precomputed w for given values of E and F.

3.1.4 How to calculate M_{low}

Listing 3.1.4.: How to calculate M_{low}

```
1  \\Contract:
2  \\Input:  Name (R, c, A, B, ℓ, N) of some power series
       f(x) = Σ_{i=0}^{∞} a_n x^n , z ∈ B_R(0) and some ε > 0.
3  \\Parameter: w ∈ ℕ\{0}
4  \\Output:  Some M_low ∈ [N, ∞] ∩ ℕ with M_low = N ∨ D(M_low) ≥ ε
5    if(q ≤_0.1 1/√2 + 0.1){
6      M_low := ⌊max{N,
7                    (ln(2)q)/(1−q) (− log(ε) − log_2(1 − q) + log_2(A))}⌋
8      Return M_low
9    }else{
10     M_low := ⌊max{N,
11                   2/(2+√2) (− log_2(ε)−log_2(1−q)+log_2(A))/(1−q),
```

```
12                  2/(2+√2) · (−log2(ε)−log2(1−q)+log2(B)+ℓ log2(ℓ)−1/log2(2+√2) ℓ log2(1−q))/(1−q) }⌋
13      Return M_low
14    }
```

Remark 3.1.3. Regarding max and min, it holds the same as in remark 3.1.2.

It is quite easy to construct a correct algorithm for the upper contract (use the projektion on N). We will not discuss in which regard our algorithm is better than a trivial one. In fact the proof of the correctness of the upper algorithm in section 3.2.2 does implicitly carry such a discussion.

3.1.5 How to compute C

For C we give two algorithms. The first one uses an explicit formula for $D(M)$, the second one uses a simplified estimation. We define the computable multivalued predicate

$$a \leq_\delta b := \begin{cases} true & \text{if } a \leq b \\ false & \text{if } a \geq b - \delta \end{cases}.$$

Listing 3.1.5.: First version of C

```
1 \\Contract: \\Input:  Name (R, c, A, B, ℓ, N) of some power
    series f(x) = Σ_{i=0}^∞ a_n x^n, z ∈ B_R(0), some ε > 0 and M ∈ ℕ.
2 \\Output: Boolean value B s.t. B ⇒ D(M) ≤ ε
3   D := A q^M/(1−q) + B q^M/(1−q) · ( M^ℓ + q Σ_{n=0}^{ℓ−1} (ℓ n) M^n/(1−q)^{ℓ−n} · Σ_{i=0}^{ℓ−k−1} ⟨ℓ−n; i⟩ q^i ) \\ for
        ⟨n; k⟩ see definition 3.2.9
4   Return D ≤_{ε/2} ε
```

Listing 3.1.6.: Second version of C

```
1 \\Contract:
2 \\Input:  Name (R, c, A, B, ℓ, N) of some power series
    f(x) = Σ_{i=0}^∞ a_n x^n, z ∈ B_R(0), some ε > 0 and M ∈ ℕ.
3 \\Output: Boolean value B s.t. B ⇒ D(M) ≤ ε
4   D := A q^M/(1−q) + B q^M/(1−q) · ( M^ℓ + qℓ!/(1−q)^ℓ Σ_{n=0}^{ℓ−1} ((1−q)M)^n/n! )
5   Return D ≤_{ε/2} ε
```

Remark 3.1.4. The local variable D in listing 3.1.5. actualy is the exact value of $D(M)$. In 3.1.5. D is just an upper bound on $D(M)$.

3.2 Correctness of evaluation algorithm

3.2.1 Correctness of listing 3.1.3.

We start with a lemma.

Lemma 3.2.1. *For $q \in (0,1), \ell \in \mathbb{N}$ and $M \geq \frac{2\ell}{1-q}$ it holds*

$$\sum_{n=M}^{\infty} q^n n^\ell \leq \frac{2q^M M^\ell}{1-q}.$$

Proof. The bound holds for $\ell = 0$. We assume that $\ell \geq 1$. Then

$$\begin{aligned}\sum_{n=M}^{\infty} q^n n^\ell &= \sum_{n=M}^{\infty} \frac{q^n - q^{n+1}}{1-q} n^\ell = \sum_{n=M}^{\infty} \frac{q^n}{1-q} n^\ell - \sum_{n=M}^{\infty} \frac{q^{n+1}}{1-q} n^\ell \\ &= \sum_{n=M}^{\infty} \left(\frac{q^n}{1-q} n^\ell - \frac{q^{n+1}}{1-q} (n+1)^\ell \right) - \sum_{n=M}^{\infty} \left(\frac{q^{n+1}}{1-q} n^\ell - \frac{q^{n+1}}{1-q} (n+1)^\ell \right) \\ &= \frac{q^M}{1-q} M^\ell + \frac{q}{1-q} \sum_{n=M}^{\infty} q^n ((n+1)^\ell - n^\ell).\end{aligned}$$

Now we want to develop a suitable bound for $(n+1)^\ell - n^\ell$. By the mean value theorem we have

$$(n+1)^\ell - n^\ell \leq \max\{\ell x^{\ell-1} : x \in [n, n+1]\}(n+1-n) = \ell(n+1)^{\ell-1}.$$

The term $(\frac{n+1}{n})^{\ell-1}$ is decreasing in n. Solving for n yields: For $n \geq \frac{\ell - \sqrt[\ell]{q}}{1 - \ell - \sqrt[\ell]{q}}$ (We set $\sqrt[0]{q} = 1$ and $\frac{x}{\infty} = 0$, then the results also hold for $\ell = 1$) we have

$$\left(\frac{n+1}{n} \right)^{\ell-1} \leq \frac{1}{q} \Leftrightarrow (n+1)^{\ell-1} \leq \frac{n^{\ell-1}}{q}.$$

Now we want to show that each $n \geq M$ fullfils the upper condition on n. The next fact is quite cumbersome to prove, but needs no insight: By differentiation we get that $\frac{1-q}{1 - \ell - \sqrt[\ell]{q}}$ is decreasing and with L'Hospital that $\lim_{q \to 1} \frac{1-q}{1 - \ell - \sqrt[\ell]{q}} = \ell - 1$ and hence $\frac{1-q}{1 - \ell - \sqrt[\ell]{q}} \leq \ell - 1$. So

$$M \geq \frac{2\ell}{1-q} \geq \frac{\ell - 1}{1-q} \geq \frac{1}{1 - \sqrt[\ell]{q}} \geq \frac{\ell - \sqrt[\ell]{q}}{1 - \sqrt[\ell]{q}}.$$

So we have

$$\begin{aligned}\sum_{n=M}^{\infty} q^n n^\ell &\leq \frac{q^M M^\ell}{1-q} + \frac{q}{1-q} \sum_{n=M}^{\infty} q^n \frac{\ell}{q} n^{\ell-1} \\ &\leq \frac{q^M M^\ell}{1-q} + \frac{\ell}{1-q} \sum_{n=M}^{\infty} q^n \frac{n}{M} n^{\ell-1} \\ &\leq \frac{q^M M^\ell}{1-q} + \frac{\ell}{(1-q)M} \sum_{n=M}^{\infty} q^n n^\ell.\end{aligned}$$

We substract $\frac{\ell}{(1-q)M}\sum_{n=M}^{\infty} q^n n^\ell$ and get

$$\sum_{n=M}^{\infty} q^n n^\ell (1 - \frac{\ell}{(1-q)M}) \leq \frac{q^M M^\ell}{1-q}.$$

And now by $M \geq \frac{2\ell}{1-q}$

$$\sum_{n=M}^{\infty} q^n n^\ell \frac{1}{2} \leq \frac{q^M M^\ell}{1-q}.$$

□

Another lemma:

Lemma 3.2.2. *Let $E > 0, F \in \mathbb{R}, w \in \mathbb{N}\setminus\{0\}$ and $x > 0$ with*

$$x \geq \frac{w\ln(2)e}{w\ln(2)e - 1}(E\log_2(wE) + F).$$

Then

$$x - E\log_2(x) - F \geq 0.$$

Proof. We will show a stronger result, that is: for any $C > 1$ and $r \in (0,1)$ and $x > 0$ with

$$x \geq \frac{1}{r}(E\log_C(\frac{E}{\ln(C)(1-r)e}) + F)$$

we have

$$x - E\log_C(x) - F \geq 0.$$

Now let $C = 2$ and $r = \frac{w\ln(2)e-1}{w\ln(2)e}$ and we have the assertion of the lemma. Now we proof the claim:

Define the function $f : \mathbb{R}^+ \to \mathbb{R}$ by

$$f(x) = x - E\log_C(x) - F.$$

We have $f'(x) = 1 - \frac{E}{\ln(C)}\frac{1}{x}$ and $f''(x) = \frac{E}{\ln(C)}\frac{1}{x^2}$, so f is convex. We define $y := \frac{E}{\ln(C)}\frac{1}{1-r} > 0$ and $h := \frac{1}{r}(E\log_C(\frac{E}{\ln(C)}\frac{1}{1-r}) + B - \frac{E}{\ln(C)}\frac{1}{1-r})$, then $y+h = \frac{1}{r}(E\log_C(\frac{E}{\ln(C)(1-r)e}) + F)$ which is the bound of the claim. So for any x of the claim we have $x > 0$ and $x \geq y+h$. Furthermore

$$f(x) = f(y + (x-y) \geq f(y) + f'(y)(x-y)$$

since f is convex and $x, y > 0$. So by $x - y \geq h$

$$f(x) \geq f(y) + f'(y)h = 0.$$

This proofs the claim. □

Remark 3.2.3. The upper lemma calculates just one step of the Newton method on the convex function $f(x) = x - E\log_2(x) - F$ starting at a point y at which f has slope $\frac{w\ln(2)e-1}{w\ln(2)e}$.

With the next corollary we can finally proof the correctness of 3.1.3.

Corollary 3.2.4. *Let $0 < q < 1$, $\varepsilon > 0$ and $A, B \in \mathbb{R}^+ \setminus \{0\}$ and $\ell, w \in \mathbb{N} \setminus \{0\}$. Then for $M \in \mathbb{N}$ holds*

$$M \geq \max\{\frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(A)}{-\log_2(q)}, \frac{2\ell}{1-q},$$
$$\frac{w\ln(2)e}{w\ln(2)e-1}\frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(2B) + \ell\log_2(w\ell) - \ell\log_2(-\log_2(q))}{-\log_2(q)}\}$$
$$\Rightarrow \sum_{n=M}^{\infty} Aq^n + Bn^\ell q^n \leq \varepsilon.$$

Proof. We fix some M fullfiling the upper condition. Since $M \geq \frac{2\ell}{1-q}$ we can apply lemma 3.2.1 and the formula for geometric series

$$\sum_{n=M}^{\infty} Aq^n + Bn^\ell q^n \leq A\frac{q^M}{1-q} + B\frac{2q^M M^\ell}{1-q}.$$

So it suffices to show that both addends of the upper inequality are less or equal $\frac{\varepsilon}{2}$.

We start our analysis with the first addend. Since $q \neq 0$ we get by easy transformation

$$\frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(A)}{-\log_2(q)} \leq M \Rightarrow A\frac{q^M}{1-q} \leq \frac{\varepsilon}{2}.$$

And since M fullfils the upper condition, the first addend is less or equal $\frac{\varepsilon}{2}$.

Now let us look at the second term. Some transformations yield

$$0 \leq M - \underbrace{\frac{\ell}{-\log(q)}}_{=:E}\log_2(M) - \underbrace{\frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(2B)}{-\log_2(q)}}_{=:F} \Rightarrow 2M_{up}\frac{q^M M^\ell}{1-q}2 \leq \frac{\varepsilon}{2}.$$

And by lemma 3.2.2 with E and F indicated as above, we have

$$M \geq \frac{w\ln(2)e}{w\ln(2)e-1}\left(\frac{\ell}{-\log(q)}\log_2\left(w\frac{\ell}{-\log(q)}\right) + \frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(2B)}{-\log_2(q)}\right)$$
$$\Rightarrow 0 \leq M - \frac{\ell}{-\log(q)}\log_2(M) - \frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(2B)}{-\log_2(q)}.$$

And since M fullfils the upper condition (after some transformation the upper condition is the same as the last term of the maximum of the corollary) also the second addend is less or equal $\frac{\varepsilon}{2}$. This shows $\sum_{n=M}^{\infty} Aq^n + Bn^\ell q^n \leq \varepsilon$. □

One could ask why the formula given by the upper corollary is not used. The reason is, that the upper formula cannot be applied if $q = 0$. One could make a case distinction, but this had to be multivalued and so we need a formula which yields correct results for a neighbourhood of 0.

In fact we will do this in a disguised form. We will use suitable upper bounds for the different terms occuring it the maximum of the upper corollary, which are optimal for $q \to 1$. So to speak we will use a continuous case distinction.

Proposition 3.2.5. *Listing 3.1.3. is correct.*

Proof. If $q = 0$ then $z = 0$ and so for each $M \geq 1$, $\sum_{n=M}^{\infty} a_n z^n = 0 \leq \varepsilon$. By definition of M_{up} we have $M_{up} \geq \frac{2\ell}{1-q} \geq 1$.

So we assume that $q \neq 0$. Since $M_{up} \geq N$ it holds $\sum_{n=M_{up}}^{\infty} \leq D(M)$. So we want to apply corollary 3.2.4 on M_{up}. Therefore we have to check

$$M_{up} \geq \frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(A)}{-\log_2(q)},$$
$$M_{up} \geq \frac{2\ell}{1-q},$$
$$M_{up} \geq \frac{w\ln(2)e}{w\ln(2)e-1}\frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(B) + \ell\log_2(w\ell) - \ell\log_2(-\log_2(q))}{-\log_2(q)}.$$

$M_{up} \geq \frac{2\ell}{1-q}$ is clear by the definition of M_{up}. By definition of M_{up} we have $M_{up} \geq \ln(2)\frac{-\log(\frac{\varepsilon}{2})-\log_2(1-q)+\log_2(A)}{1-q}$. So it suffices to show

$$\frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(A)}{-\log_2(q)} \leq \ln(2)\frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(A)}{1-q}.$$

This is obviously equivalent to

$$\frac{1}{-\log_2(q)} \leq \frac{\ln(2)}{1-q}.$$

We proof this by showing that the function $g(x) := -\log_2(x) - \frac{1-x}{\ln(2)}$ is not-negative on $(0,1)$. The derivative of g tells us that g is decreasing in $(0,1)$ and hence it suffices to check $g(1) \geq 0$, which is true.

It remains to show that

$$M_{up} \geq \frac{w\ln(2)e}{w\ln(2)e-1}\frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(B) + \ell\log_2(w\ell) - \ell\log_2(-\log_2(q))}{-\log_2(q)}.$$

Which will be shown in the same matter as above by

$$\frac{w\ln(2)e}{w\ln(2)e-1}\ln(2)\frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(B) + \ell\log_2(w\ell) - \ell\log_2(1-q)}{1-q}$$
$$\geq \frac{w\ln(2)e}{w\ln(2)e-1}\frac{-\log(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(B) + \ell\log_2(w\ell) - \ell\log_2(-\log_2(q))}{-\log_2(q)}.$$

Therefore it suffices to show that

$$\frac{1}{-\log_2(q)} \leq \frac{\ln(2)}{1-q} \text{ and}$$
$$-\log_2(-\log_2(q)) \leq -\log_2(1-q).$$

The first inequality was already shown and the second inequality can be shown in the same way. □

3.2.2 Correctness of listing 3.1.4.

We start with a lemma.

Lemma 3.2.6. *For $M \in \mathbb{N}$ and $q \in (0,1)$ it holds*

$$\frac{q^M M^\ell}{1-q} \leq \sum_{n=M}^{\infty} q^n n^\ell.$$

Proof.

$$\sum_{n=M}^{\infty} q^n n^\ell = \sum_{n=0}^{\infty} q^{M+n}(M+n)^\ell = \sum_{n=0}^{\infty} q^{M+n} \sum_{k=0}^{\ell} \binom{\ell}{k} M^k n^{\ell-k}$$
$$= q^M \sum_{k=0}^{\ell} M^k \binom{\ell}{k} \sum_{n=0}^{\infty} q^n n^{\ell-k} \geq q^M M^\ell \sum_{n=0}^{\infty} q^n = \frac{q^M M^\ell}{1-q}.$$

□

Now we derive a suitable criterion on M s.t. $D(M) \geq \varepsilon$.

Lemma 3.2.7. *Let $0 < q < 1$,$\varepsilon > 0$,$A, B, \ell \in \mathbb{R}^+ \setminus \{0\}$ and $\ell, w \in \mathbb{N} \setminus \{0\}$. Then for $M \in \mathbb{N}$ holds*

$$M \leq \frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(A)}{-\log_2(q)} \Rightarrow \sum_{n=M}^{\infty} Aq^n + Bn^\ell q^n \geq \varepsilon \tag{3.1}$$

and

$$\frac{\ell}{-\log_2(q)} \geq \frac{2^w}{w} \wedge M \leq \frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(B) + \ell \log_2(\frac{w\ell}{-\log_2(q)})}{-\log_2(q)} \tag{3.2}$$
$$\Rightarrow \sum_{n=M}^{\infty} Aq^n + Bn^\ell q^n \geq \varepsilon.$$

Proof. One easily checks that the premise of (3.1) implies $\sum_{n=M}^{\infty} Aq^n \geq \varepsilon$ and the premise of (3.2) implies $\sum_{n=M}^{\infty} Bq^n n^\ell \geq \varepsilon$. From this follows the lemma. □

Proposition 3.2.8. *Listing 3.1.4. is correct.*

Proof. We recall what listing 3.1.4. does.

If $q \leq \frac{1}{\sqrt{2}}$, it returns M_{low} as the maximum of the two terms

$$t_1 := N$$
$$t_2 := \frac{\ln(2)q}{1-q}\left(-\log(\varepsilon) - \log_2(1-q) + \log_2(A)\right).$$

If $M_{low} = t_1$, then we have not found a lower bound bigger then N (maybe there is none or we were just to inexact), so this is the bad case.

If $M_{low} = t_2$, then we have the following inequality

$$M_{up} = \frac{\ln(2)q}{1-q}\left(-\log(\varepsilon) - \log_2(1-q) + \log_2(A)\right) \leq \frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(A)}{-\log_2(q)}$$

since $\frac{\ln(2)x}{1-x} \leq \frac{1}{-\log_2(x)}$ for all $x \in (0,1)$. And so by 3.1 $D(M_{low}) \geq \varepsilon$.

If $q \geq \frac{1}{\sqrt{2}}$, it returns M_{low} as the maximum of the three terms (In fact in the intervall $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} + 0.1]$ it may just return the M_{low} from above. This is due to the exact real computation)

$$t_1 := N$$
$$t_2 := \frac{2}{2+\sqrt{2}} \frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(A)}{1-q}$$
$$t_3 := \frac{2}{2+\sqrt{2}} \frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(B) + \ell\log_2(\ell) - \frac{1}{\log_2(2+\sqrt{2})}\ell\log_2(1-q)}{1-q}$$

If $M_{low} = t_1$ then, as above, we did not find anything better.

If $M_{low} = t_2$, then we have the following inequality

$$M_{low} = \frac{2}{2+\sqrt{2}} \frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(A)}{1-q} \leq \frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(A)}{-\log_2(q)}$$

since $\frac{2}{2+\sqrt{2}} \frac{1}{1-x} \leq \frac{1}{-\log_2(x)}$ for all $x \in (\frac{1}{\sqrt{2}}, 1)$. And so by 3.1 $D(M_{up}) \geq \varepsilon$.

If $M_{low} = t_3$, then we have the following inequality

$$\begin{aligned} M_{low} &= \frac{2}{2+\sqrt{2}} \frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(B) + \ell\log_2(\ell) - \frac{1}{\log_2(2+\sqrt{2})}\ell\log_2(1-q)}{1-q} \\ &\leq \frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(B) + \ell\log_2(\ell) - \ell\log_2(-\log_2(q))}{-\log_2(q)} \end{aligned}$$

since $\frac{2}{2+\sqrt{2}} \frac{1}{1-x} \leq \frac{1}{-\log_2(x)}$, $\frac{1}{\log_2(2+\sqrt{2})} - \log_2(1-x) \leq -\log_2(-\log_2(x))$ for all $x \in [\frac{1}{\sqrt{2}}, 1)$. And since $\frac{\ell}{-\log(q)} \geq 2$ ($q \geq \frac{1}{\sqrt{2}}$) we have by 3.2 with $w = 1$ that $D(M_{up}) \geq \varepsilon$. □

3.2.3 Correctness of listing 3.1.5. and listing 3.1.6.

We will shortly introduce the eulerian numbers, as we need them for an exact formula for $D(M)$.

Definition 3.2.9. [JMHM08, (2.114),(2.115) and (2.120)] $\left\langle {n \atop k} \right\rangle$ is the eulerian number, which is defined for every $n, k \in \mathbb{N}$ with $0 \leq k < n$ or $k = n = 0$. Its recursive definition is given by:

$$\left\langle {n \atop 0} \right\rangle = 1$$

$$\left\langle {n \atop n-1} \right\rangle = 1$$

$$\left\langle {n \atop k} \right\rangle = (k+1) \left\langle {n-1 \atop k} \right\rangle - (n-k) \left\langle {n-1 \atop k-1} \right\rangle$$

Lemma 3.2.10. *For $q \in \mathbb{R}$ and $\ell, N \geq 0$ holds*

$$\sum_{n=N}^{\infty} n^{\ell} q^n = \frac{q^N}{1-q} \left(N^{\ell} + q \sum_{k=0}^{\ell-1} N^k \binom{\ell}{k} \frac{1}{(1-q)^{\ell-k}} \sum_{i=0}^{\ell-k-1} \left\langle {\ell-k \atop i} \right\rangle q^i \right).$$

Proof.

$$\begin{aligned}
\sum_{n=N}^{\infty} q^n n^{\ell} =& \sum_{n=0}^{\infty} q^{N+n} (M+n)^{\ell} = \sum_{n=0}^{\infty} q^{N+n} \sum_{k=0}^{\ell} \binom{\ell}{k} N^k n^{\ell-k} \\
=& q^N \sum_{k=0}^{\ell} N^k \binom{\ell}{k} \sum_{n=0}^{\infty} q^n n^{\ell-k} \\
=& q^N \sum_{k=0}^{\ell} N^k \binom{\ell}{k} \left(\delta_{\ell-k,0} + \sum_{n=1}^{\infty} q^n n^{\ell-k} \right) \\
\overset{\text{[JMHM08, (2.126)]}}{=}& q^N \sum_{k=0}^{\ell} N^k \binom{\ell}{k} \left(\delta_{\ell-k,0} + \frac{q}{(1-q)^{\ell-k+1}} \sum_{i} \left\langle {\ell-k \atop i} \right\rangle q^i \right)
\end{aligned}$$

□

Proposition 3.2.11. *Listing 3.1.5. and listing 3.1.6. are correct.*

Proof. Since lemma 3.2.10 the variable D in listing 3.1.5. is equal to $D(M)$. So $D \leq_{\frac{\varepsilon}{2}} \varepsilon$ can only hold if $D(M) \leq \varepsilon$.

For listing 3.1.6. we use the the following estimation

$$\begin{aligned}\sum_{n=N}^{\infty} n^\ell q^n &\overset{\text{lem } 3.2.10}{=} \frac{q^N}{1-q}\left(N^\ell + q\sum_{k=0}^{\ell-1} N^k \binom{\ell}{k} \frac{1}{(1-q)^{\ell-k}} \sum_{i=0}^{\ell-k-1} \left\langle {\ell-k \atop i} \right\rangle q^i\right)\\ &\overset{q\in(0,1)}{\leq} \frac{q^N}{1-q}\left(N^\ell + q\sum_{k=0}^{\ell-1} N^k \binom{\ell}{k} \frac{1}{(1-q)^{\ell-k}} \sum_{i=0}^{\ell-k-1} \left\langle {\ell-k \atop i} \right\rangle\right)\\ &\overset{\text{[JMHM08, (2.117)]}}{=} \frac{q^N}{1-q}\left(N^\ell + q\sum_{k=0}^{\ell-1} N^k \binom{\ell}{k} \frac{1}{(1-q)^{\ell-k}} (\ell-k)!\right)\end{aligned}$$

So the variable D in listing 3.1.6. is greater equal $D(M)$ and by the same argumentation as above listing 3.1.6. is correct. □

3.3 Towards the complexity of the evaluation algorithm

Before all else, we want to say that we did not manage to determine the complexity of the algorithm. So we can only argue in a limited sense for the efficiency of the algorithm. Furthermore we cannot even argue that our general concept of the evaluation algorithm is efficient.

The concept of the evaluation algorithm is based on the idea of finding a suitable partial sum $\sum_{n=0}^{M-1} a_n z^n$ approximating $f(z)$. The following two aspects seem to be most important for the complexity of the algorithm:

A1 The computation for M should be efficient, which is evident if the overall algorithm should be fast.

A2 M should be as small as possible, so we need less values of c which has unknown complexity.

For the first aspect, as previously mentioned, we have no answer. For the second aspect we want to give an heuristic argument why our algorithm determines a rather small M. But before this, we want to emphasise that even if we could answer this question to full extend the result had limited consequences.

3.3.1 Towards the generall concept of the evaluation algorithm

How good behaves the algorithm with respect to the two upper aspects. In fact we do not know. But even if we had an algorithm which uses the minimal M_{min}, satisfying $D(M_{min}) \leq \varepsilon$, i.e. an optimal result with respect to the second aspect of the upper list, one can construct a different algorithm which is better.

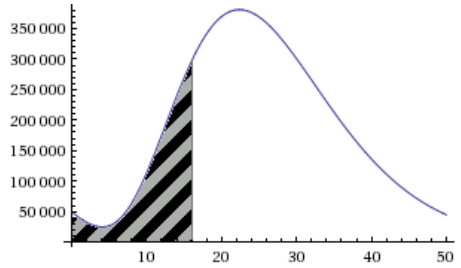
Remark 3.3.1. The upper algorithm is not optimal with respect to number of values of c are needed.

We will show the remark soon, but first we want to mention that this is the first efficiency gap we have encountered. There might be more sophisticated algorithms which work utterly different and are more efficient as our approach and this early design desicion hampers our algorithm to be such effective.

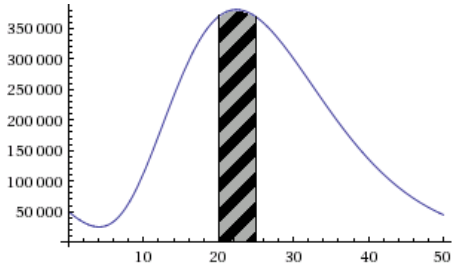
Here is an example which is related to the upper remark. The following algorithm is more economic with respect to the needed values of c.

Possible outline of evaluation

1 \\Contract:
2 \\Input: Name (R, c, A, B, ℓ, N) of some power series $f(x) = \sum_{i=0}^{\infty} a_n x^n$, $z \in B_R(0)$ and some $\varepsilon > 0$.
3 \\Output: Some real y with $|f(z) - y| \leq \varepsilon$.
4 Determine a set X with least cardinality in $\{X \subseteq [N, \infty] : \sum_{n \in X^c} Aq^n + Bn^\ell q^n \leq \varepsilon\}$.
5 Return $\sum_{n=0}^{N-1} c(n)z^n + \sum_{n \in X} c(n)z^n$.



(a) Area summed up by Listing 3.1.1. in the example



(b) Area summed up by the possible outline of evaluation in the example

Figure 3.1: Listing 3.1.1. is not optimal in the usage of c (made with wolframalpha.com and Gimp)

We show this for an example. Consider the holomorphic function $f(z) = \sum_{i=0}^{\infty}(50000 + 10x^5)$. We want to evaluate at $z = 0.8$ with error $\varepsilon = 7.78 \cdot 10^6$ (sic!) (one could scale the whole example to get a small ε). We use the datatype $(1, c, 50000, 10, 5, 0)$. Listing 3.1.1. has to compute at least $\sum_{n=0}^{16} c(n)z^n$ since the minimal M satisfying $D(M) \leq \varepsilon$ is 17, so in the best case we needed 17 values of c. The upper listing would just compute $\sum_{n \in X} c(n)z^n$ with $X = \{20, 21, 22, 23, 24, 25\}$ and hence uses 6 values of c. See figure 3.1.

This example is rather unnatural and the difference of values of c is rather small, but this was only a simple example (We do not know whether the latter algorithm does better in not pathological examples).

3.3.2 How good is the output of the first step of evaluation

Since the first step of evaluation entails a binary search one could think that the output M is the least M, s.t. $D(M) \leq \varepsilon$ holds. But, as said in the beginning of section 3.1.2, the decision predicate C has not the property $C(M) \Leftrightarrow D(M) \leq \varepsilon$. So we will use the output M_{up} of listing 3.1.3. as base for our discussion. In order to make concrete results we assume that $-\log_2(\varepsilon) + \log_2(A) \geq 1$ and $-\log_2(\varepsilon) + \log_2(B) \geq 1$. This restriction seems quite reasonable, since, in general, ε is small and A, B are big. M_{up} is basically the maximum of the four terms:

$$
\begin{aligned}
t_1 &:= N \\
t_2 &:= \ln(2)\frac{-\log_2(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(A)}{1-q} \\
t_3 &:= \frac{2\ell}{1-q} \\
t_4 &:= \frac{w\ln^2(2)e}{w\ln(2)e-1}\frac{-\log_2(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(2B) + \ell\log_2(w\ell) - \ell\log_2(1-q)}{1-q}
\end{aligned}
$$

If $q \leq \frac{1}{\sqrt{2}}$, then

$$
\begin{aligned}
t_2 &\leq 2.4(-\log_2(\varepsilon) + \log_2(A)) + 6.6 \\
t_3 &\leq 6.8\ell \\
t_4 &\leq \frac{w\ln(2)e}{w\ln(2)e-1}\left(2.4(-\log_2(\varepsilon) + \log_2(B) + \ell(\log_2(w\ell) + 1.8) + 9.0\right).
\end{aligned}
$$

So we may make a huge relative error for finding an optimal M_{up} (and, in general, this can be arbitrarily big), but the value of M_{up} is a nice formula in A, B, ℓ, ε.

So now we will base the discussion on the case that $q \geq \frac{1}{\sqrt{2}}$. We do a case distinction.

$M_{up} = t_1$ Then we cannot do any better, since the estimation of the a_n starts from N and before this a_n can go beserk.

$M_{up} = t_2$ We have

$$
\begin{aligned}
M_{up} &= \ln(2)\frac{-\log_2(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(A)}{1-q} \\
&\leq \ln(2)(1+\frac{1}{\sqrt{2}})\frac{\log_2(\varepsilon) - \log_2(1-q) + \log_2(A) + 1}{-\log_2(q)} \\
&\leq \underbrace{1.2\frac{\log_2(\varepsilon) - \log_2(1-q) + \log_2(A) + 1}{-\log_2(q)}}_{:=U}.
\end{aligned}
$$

Here we used that

$$
\forall x \in [\frac{1}{\sqrt{2}}, 1), \quad \frac{1}{1-x} \leq (1+\frac{1}{\sqrt{2}})\frac{1}{-\log(x)}.
$$

We also have

$$M \leq \underbrace{\frac{\log_2(\varepsilon) - \log_2(1-q) + \log_2(A)}{-\log_2(q)}}_{:=L} \Rightarrow A\sum_{n=M}^{\infty} q^n \geq \varepsilon \Rightarrow D(M) \geq \varepsilon.$$

The fraction $\frac{U}{L}$ gives some upper bound for the relative error we have made with M_{up}

$$\frac{U}{L} = 1.2(1 + \frac{1}{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(A)}).$$

Since the nominator of the upper formula is greater or equal 1, we see that the relative error is less or equal 2.4.

$M_{up} = t_4$ We have

$$\begin{aligned} M_{up} &= \frac{w\ln(2)^2 e}{w\ln(2)e - 1} \frac{-\log_2(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(2B) + \ell\log_2(w\ell) - \ell\log_2(1-q)}{1-q} \\ &\leq \frac{w\ln(2)^2 e}{w\ln(2)e - 1}(1 + \frac{1}{\sqrt{2}}) \\ &\frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(B) + \ell\log_2(w\ell) - \ell\log_2(2+\sqrt{2})\log_2(-\log_2(q)) + 2}{-\log_2(q)} =: U \end{aligned}$$

Here we used that

$$\forall x \in [\frac{1}{\sqrt{2}}, 1), \quad \frac{1}{1-x} \leq (1 + \frac{1}{\sqrt{2}})\frac{1}{-\log(x)} \text{ and } -\log_2(1-x) \leq -\log_2(2+\sqrt{2})\log_2(\log_2(x)).$$

We also have by (3.1) if $\frac{\ell}{-\log_2(q)} \geq \frac{2^w}{w}$

$$M \leq \underbrace{\frac{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(B) + \ell\log_2(\frac{w\ell}{-\log_2(q)})}{-\log_2(q)}}_{=:L} \Rightarrow D(M) \leq \varepsilon$$

The fraction $\frac{U}{L}$ gives some upper bound for the relative error we have made with M_{up}

$$\begin{aligned} \frac{U}{L} =& \frac{w\ln(2)^2 e}{w\ln(2)e - 1}(1 + \frac{1}{\sqrt{2}}) \left(1 + \frac{-\ell(\log_2(2+\sqrt{2}) - 1)\log_2(-\log_2(q)) + 2}{-\log_2(\varepsilon) - \log_2(1-q) + \log_2(B) + \ell\log_2(\frac{w\ell}{-\log_2(q)})}\right) \\ &\leq 9.5 \end{aligned}$$

$M_{up} = t_3$ Since $q \geq \frac{1}{\sqrt{2}}$ we get $t_4 \geq \ln(2)\frac{\ell\log(1-\frac{1}{\sqrt{2}})}{1-q} \geq 1.3\frac{\ell}{1-q}$. So t_3 is approximately 1.5-times of t_4. So the relative error is smaller than 15.

3.4 The relation between A and B

We want to discuss which relation between A and B is the best for the evaluation algorithm. We base our discussion on the step for finding the M_{up}, to be more precisely we discuss, listing 3.1.3.. In this algorithm M_{up} is basically the maximum of the two terms.

$$t_1 = \ln(2)\frac{-\log_2(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(A)}{1-q}$$
$$t_2 = \frac{w\ln(2)e}{w\ln(2)e-1}\ln(2)\frac{-\log_2(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(2B) + \ell\log_2(w\ell) - \ell\log_2(1-q)}{1-q}.$$

We have

$$\begin{aligned}
&A \geq 2B\cdot\ell^{w\ell}\\
\Rightarrow &\log_2(A) \geq \log_2(2B) + \ell\log_2(w\ell) - \log_2(1-q)\\
\Rightarrow &\ln(2)\frac{-\log_2(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(A)}{1-q}\\
\geq &\frac{w\ln(2)e}{w\ln(2)e-1}\ln(2)\frac{-\log_2(\frac{\varepsilon}{2}) - \log_2(1-q) + \log_2(2B) + \ell\log_2(w\ell) - \ell\log_2(1-q)}{1-q}\\
\Rightarrow &t_1 \geq t_2.
\end{aligned}$$

So we think that it is most suitable if one can choose A and B, s.t. $A \approx 2B\cdot\ell^{w\ell}$. This may be of special interest for the functionals. When implementing these functionals, one has a certain degree of freedom for the choise of A, B in the output (R, c, A, B, ℓ, N). But there may also be a problem with this, if in some procedure the values of A and B get mixed, for example in multiplication, we might get higher overall results. Therefore a simple example: Assume we have the choise of two representations of a power series. The first has $A_1 = 10$,$B_1 = 10$ and $\ell_1 = 4$. The other one has $A_2 = 512$,$B_2 = 1$ and $\ell_2 = 4$. Of the point of view from the upper discussion, the latter representation is superior. Now we want to apply some functional and the result A_r, B_r both consist of the sum of A and B of the input. So we would get for the first representation $A_{r,1} = 20$,$B_{r,1} = 20$ and for the second $A_{r,2} = 513$,$B_{r,2} = 513$. Then clearly the first representation is superior.

So we try to develope methods which do not have this mixing behaviour. But nevertheless we did not manage to create methods which produce small B by costs of big A in a controlled way, s.t. $A \approx 2B\cdot\ell^{w\ell}$.

Chapter 4

Maximum on shapes

For the following chapter we fix a power series $f(x) = \sum_{i=0}^{\infty} a_n x^n$ with name (R, c, A, B, ℓ, N).

We use the following ad-hoc definition

Definition 4.0.1. A class $\mathcal{S}$ of closed subsets of $\mathbb{C}$ is called a class of shapes if there is a naming system $\delta_{\mathcal{S}}$ for $\mathcal{S}$, s.t. for each $w \in \mathrm{dom}(\delta_{\mathcal{S}})$ (we set $S := \delta_{\mathcal{S}}(w)$) and $R \in \mathbb{R}$ with $S \subseteq B_R(0)$

- we can compute a $\tau \in \mathbb{R}$, s.t.

$$S \subseteq \overline{B}_\tau \subseteq B_R(0).$$

  Then $\tau < R$.

- for each $\varepsilon > 0$ we can compute finitely many points $a_i \in S, 1 \leq i \leq n$ s.t.

$$\partial S \subseteq \bigcup_{i=1}^{n} B_\varepsilon(a_i)$$

An element of $\mathcal{S}$ is called $\mathcal{S}$-shape.

Remark 4.0.2. We want to argue that for each closed set $S \subseteq \mathbb{C}$,$R \in \mathbb{R}$ with $S \subseteq B_R(0)$ and $\varepsilon > 0$ there exists τ and $a_1, \ldots, a_n$ as in definition 4.0.1, but clearly they do not have to be computable. As S is bounded ($S \subseteq B_R(0)$) and closed, it is compact, so it has an element z with maximal absolut value. Since $S \subseteq B_R(0)$, $|z| < R$ there is a τ that fullfills the upper condition.

The $B_\varepsilon(a_1), \ldots, B_\varepsilon(a_n)$ covering ∂S exists since S is compact. Hence S can be covered by finitely balls of the upper form and so can ∂S.

In this chapter we want to discuss how to implement a procedure which determines the maximum of f on closed $\mathcal{S}$-shapes.

4.1 Maximum on general shapes

Lef $\mathcal{S}$ be a class of shapes. Let $S \in \mathcal{S}$ with $S \subseteq B_R(0)$. Since analytic functions obtain its maximum on closed sets on the boundary, we only have to consider the boundary of S to find its maximum. For determing the maximum of f on ∂S up to an error of ε, we want to cover ∂S by balls of suitable radii, s.t. f varies only ε in this balls. For this we first want to determine a lipschitz constant for f on S. For this we will use the next lemma.

Lemma 4.1.1. *Lef $f(x) = \sum_{i=0}^{\infty} a_n x^n$ be a power series with name (R, c, A, B, ℓ, N). Let $\mathbb{R} \ni \tau < R$. Then for all $a, b \in \overline{B}_\tau(0)$*

$$|f(a) - f(b)| \leq L|a - b|$$

with

$$L = \sum_{n=1}^{N-1} n|a_n|\tau^{n-1} + \frac{1}{\tau}\left(\frac{A\left(\frac{\tau}{R}\right)^N}{1-\frac{\tau}{R}}\left(N + \frac{\frac{\tau}{R}}{1-\frac{\tau}{R}}\right) + \frac{B\left(\frac{\tau}{R}\right)^N}{1-\frac{\tau}{R}}\left(N^{\ell+1} + \frac{\tau}{R}\sum_{k=0}^{\ell}\frac{N^k(\ell+1)!}{\left(1-\frac{\tau}{R}\right)^{\ell+1-k}k!}\right)\right).$$

Proof. From analysis we know that for all points $a, b \in \overline{B}_\tau(0)$ we have

$$|f(a) - f(b)| \leq |a - b| \max_{z \in \overline{B}_\tau(0)} |f'(z)|.$$

f' is given by $f' = \sum_{n=0}^{\infty} a_{n+1}(n+1)x^n$. For any point $z \in \overline{B}_\tau(0)$ we have

$$\begin{aligned}
|f'(z)| &= \left|\sum_{n=0}^{\infty} a_{n+1}(n+1)z^n\right| \leq \sum_{n=0}^{N-2}(n+1)|a_{n+1}|\tau^n + \sum_{n=N-1}^{\infty}(n+1)|a_{n+1}|\tau^n \\
&= \sum_{n=0}^{N-2}(n+1)|a_{n+1}|\tau^n + \frac{1}{\tau}\sum_{n=N}^{\infty} n|a_n|\tau^n \\
&\leq \sum_{n=0}^{N-2}(n+1)|a_{n+1}|\tau^n + \frac{1}{\tau}\sum_{n=N}^{\infty} nA\left(\frac{\tau}{R}\right)^n + n^{\ell+1}B\left(\frac{\tau}{R}\right)^n \\
&\overset{\text{lem 3.2.10}}{=} \sum_{n=0}^{N-2}(n+1)|a_{n+1}|\tau^n \\
&\quad + \frac{1}{\tau}\left(\frac{A\left(\frac{\tau}{R}\right)^N}{1-\frac{\tau}{R}}\left(N + \frac{\frac{\tau}{R}}{1-\frac{\tau}{R}}\right) + \frac{B\left(\frac{\tau}{R}\right)^N}{1-\frac{\tau}{R}}\left(N^{\ell+1} + \frac{\tau}{R}\sum_{k=0}^{\ell}\frac{N^k\binom{\ell+1}{k}}{\left(1-\frac{\tau}{R}\right)^{\ell+1-k}}\sum_{i=0}^{\ell-k}\left\langle{\ell+1-k \atop i}\right\rangle\left(\frac{\tau}{R}\right)^i\right)\right) \\
&\leq \sum_{n=1}^{N-1} n|a_n|\tau^{n-1} \\
&\quad + \frac{1}{\tau}\left(\frac{A\left(\frac{\tau}{R}\right)^N}{1-\frac{\tau}{R}}\left(N + \frac{\frac{\tau}{R}}{1-\frac{\tau}{R}}\right) + \frac{B\left(\frac{\tau}{R}\right)^N}{1-\frac{\tau}{R}}\left(N^{\ell+1} + \frac{\tau}{R}\sum_{k=0}^{\ell}\frac{N^k(\ell+1)!}{\left(1-\frac{\tau}{R}\right)^{\ell+1-k}k!}\right)\right)
\end{aligned}$$

The last estimation is the same as in the proof of the correctness of listing 3.1.6. For $\left\langle {n \atop k} \right\rangle$ see definition 3.2.9. □

So the general procedure for finding the maximum of an $\mathcal{S}$-shape can be given by

Listing 4.1.1.: General algorithm for finding the maximum

```
\\Contract:
\\Input:  Datatype (R,c,A,B,ℓ,N) of some power series f,
    name w of a S-shape S with S ⊆ B_R(0) and a ε > 0
\\Output: real y s.t. |y − max_{z∈S} |f(z)|| ≤ ε
  Determine a τ ∈ ℝ s.t. τ < R and S ⊆ B̄_τ(0) \\ depends on
     the class S
  Calculate L as in lemma 4.1.1 applied to (R,c,A,B,ℓ,N) and
     τ
  Cover ∂S by balls B̄_{ε/L}(a_1),...,B̄_{ε/L}(a_n) with a_1,...,a_n ∈ S \\
     depends on the class S
  Return max{|f(a_1)|,...,|f(a_2)|}
```

4.2 Maximum on balls

We want to describe an algorithm which finds the maximum of f on balls $\overline{B}_r(z)$ with $r \in \mathbb{R}^+$ and $z \in \mathbb{C}$. We say that the two parameters r and z describe the ball (so the tupel (r, b) gives rise to a naming system for the closed balls as in definition 4.0.1). We describe how to perform the abstract steps given in listing 4.1.1. and then present the whole algorithm.

Calculating τ is easy. We just add $|z|$ and b, so $\tau = |z| + b$. By triangle inequality we get $\overline{B}_r(z) \subseteq \overline{B}_\tau(0)$. We also see that $\overline{B}_r(z) \subseteq \overline{B}_R(0) \Leftrightarrow |z| + b < R$.

The next step is to determine the balls $\overline{B}_{\frac{\varepsilon}{L}}(a_1), \ldots, \overline{B}_{\frac{\varepsilon}{L}}(a_n)$. Here we try to minimise the number n of balls needed to cover $\partial\overline{B}_r(z)$. This means we have to place the points a_i, such that the greatest possilbe angle ϕ of $\partial\overline{B}_r(z)$ is covered by $\overline{B}_{\frac{\varepsilon}{L}}(a_i)$. If $r \leq \frac{\varepsilon}{L}$, then we just have to use one ball, that is $B_{\frac{\varepsilon}{2}}(z)$. If $r > \frac{\varepsilon}{L}$,without going into great detail, the best placement can be obtained by placing a_i as in figure 4.1.

Thereby d is chosen so that a_i is on the line between the two intersections of $\partial\overline{B}_r(z)$ and $\partial\overline{B}_{\frac{\varepsilon}{L}}(a_i)$. Some easy calculation yields: The optimal distance is $d = \sqrt{r^2 - \left(\frac{\varepsilon}{L}\right)^2}$ and the covered angle of $\phi = 2\arcsin\left(\frac{\varepsilon}{Lr}\right)$. Since arcsin is not so easy to calculate, we want to work with some easier term. We just

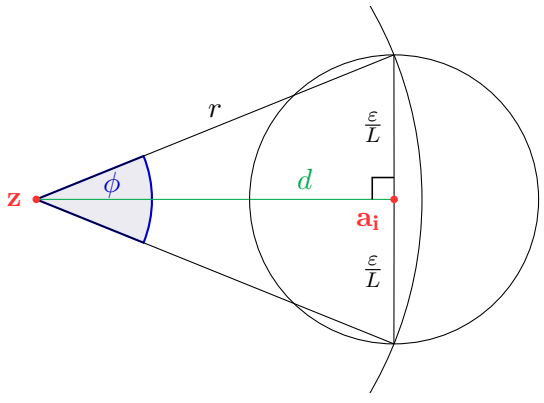


Figure 4.1: Sketch for ϕ and d

use the estimation $x \leq \arcsin(x)$ for all $0 \leq x \leq 1$. So $\phi \geq 2\frac{\varepsilon}{Lr}$ and we can cover $\partial\overline{B}_r(z)$ with $\lceil\frac{\pi Lr}{\varepsilon}\rceil$ many points. So we arrange the points $a_1, \ldots, a_{\lceil\frac{\pi Lr}{\varepsilon}\rceil}$ with distant d to z, s.t. the angle between $\overline{a_i, z}$ and $\overline{a_{i+1}, z}$ is $2\frac{\varepsilon}{Lr}$. So the balls $\overline{B}_{\frac{\varepsilon}{L}}(z + de^{1\cdot 2\frac{\varepsilon}{Lr}}), \ldots, \overline{B}_{\frac{\varepsilon}{L}}(z + de^{\lceil\frac{\pi Lr}{\varepsilon}\rceil\cdot 2\frac{\varepsilon}{Lr}})$ do the job. So it remains to get rid of the case distinction between $r > \frac{\varepsilon}{L}$ and $r \leq \frac{\varepsilon}{L}$. We somehow want to incorporate the case in which just one ball is sufficient in the case where we have to use several balls. Therefore we will adapt the formula for d: $d = \sqrt{\max\{r^2 - \left(\frac{\varepsilon}{L}\right)^2, 0\}}$, so if $r \leq \frac{\varepsilon}{L}$ all the points $\overline{B}_{\frac{\varepsilon}{L}}(z+de^{1\cdot 2\frac{\varepsilon}{Lr}}), \ldots, \overline{B}_{\frac{\varepsilon}{L}}(z+de^{\lceil\frac{\pi Lr}{\varepsilon}\rceil\cdot 2\frac{\varepsilon}{Lr}})$ will be placed on z. This will be good enough for us. Now we can state the whole algorithm (use instead of $\lceil\cdot\rceil$ some computable upper bound as in the end of remark 3.1.2).

Listing 4.2.1.: Algorithm for finding the maximum on a ball

```
\\Contract:
\\Input:  Name (R, c, A, B, ℓ, N) of some power series f, the
    description of a closed ball B̄_r(z), i.e. (r, z) with
    |z| + r < R and a ε > 0
\\Output: real y s.t. |y − max_{z∈B_r(z)} f(z)| ≤ ε
  τ := |z| + r
  Calculate L as in lemma 4.1.1 applied to (R, c, A, B, ℓ, N) and
      τ
  d := √(max{r² − (ε/2)², 0})
  Return max{z + de^{1·2ε/(Lr)}, ..., z + de^{⌈πLr/ε⌉·2ε/(Lr)}}
```

Chapter 5

Computing operators

5.1 Addition

We are given a power series $f = \sum a_n x^n$ with name $(R_f, c_f, A_f, B_f, \ell_f, N_f)$ and a power series $g = \sum b_n x^n$ with name $(R_g, c_g, A_g, B_g, \ell_g, N_g)$ and want to determine a name $(R_h, c_h, A_h, B_h, \ell_h, N_h)$ for the power series $h = f + g$.

Listing 5.1.1.: Summation algorithm

\\Contract:
\\Input: Name $(R_f, c_f, A_f, B_f, \ell_f, N_f)$ of some power series f, name $(R_g, c_g, A_g, B_g, \ell_g, N_g)$ of some power series g s.t. $\ell_f \geq \ell_g \geq 1$.
\\Parameter: $K \in \mathbb{N}$
\\Output: Some name $(R_h, c_h, A_h, B_h, \ell_h, N_h)$ of the power series $h := f + g$
$R_h := \min\{R_f, R_g\}$
$c_h := c_f + c_g$
$N_h := \max\{N_g, N_f\}$
$\ell_h := \ell_f$
$A'_f := \left(\frac{R_h}{R_f}\right)^{N_h} A_f$, $A'_g := \left(\frac{R_h}{R_g}\right)^{N_h} A_g$, $B'_f := \left(\frac{R_h}{R_f}\right)^{N_h} B_f$, $B'_g := \left(\frac{R_h}{R_g}\right)^{N_h} B_g$
$A_h := \begin{cases} \max\left\{0, A'_f + A'_g + N_h^{\ell_g} B'_g \left(1 - \frac{\ell_g}{\ell_f}\left(\frac{N_h}{K}\right)^{\ell_f - \ell_g}\right)\right\} & \texttt{if } K \leq N_h \\ A'_f + A'_g + K^{\ell_g} B'_g \left(1 - \frac{\ell_g}{\ell_f}\right) & \texttt{if } K \geq N_h \end{cases}$
$B_h := B'_f + B'_g \frac{\ell_g}{\ell_f K^{\ell_f - \ell_g}}$
Return $(R_h, c_h, A_h, B_h, \ell_h, N_h)$

Proposition 5.1.1. *Listing 5.1.1. is correct.*

Proof. It is clear, that $c_h = c_f + c_g$ is the appropriate function for the coefficients. It is left to check that for all $n \geq N_h$ $|c_h(n)| \leq R_h^{-n}(A_h + n^{\ell_h} B_h)$. So fix some $n \geq N_h$. Since

$N_f, N_g \leq N_h$ we have

$$|c_h(n)| \leq |c_f(n)| + |c_g(n)| \leq R_f^{-n}(A_f + n^{\ell_f} B_f) + R_g^{-n}(A_g + n^{\ell_g} B_g).$$

So it suffices to show that

$$R_f^{-n}(A_f + n^{\ell_f} B_f) + R_g^{-n}(A_g + n^{\ell_g} B_g) \leq R_h^{-n}(A_h + n^{\ell_h} B_h).$$

We treat the case $K \geq N_h$, so by definition of A_h, B_h, n^h we have to show

$$\begin{aligned}
&R_f^{-n}(A_f + n^{\ell_f} B_f) + R_g^{-n}(A_g + n^{\ell_g} B_g) \\
&\leq R_h^{-n} \left(\left(\frac{R_h}{R_f} \right)^{N_h} A_f + \left(\frac{R_h}{R_g} \right)^{N_h} A_g + K^{\ell_g} \left(\frac{R_h}{R_g} \right)^{N_h} B_g \left(1 - \frac{\ell_g}{\ell_f} \right) \right) \\
&+ R_h^{-n} \left(n^{\ell_f} \left(\left(\frac{R_h}{R_f} \right)^{N_h} B_f + \left(\frac{R_h}{R_g} \right)^{N_h} B_g \frac{\ell_g}{\ell_f K^{\ell_f - \ell_g}} \right) \right).
\end{aligned}$$

Since $R_h^{-n} \left(\frac{R_h}{R_f} \right)^{N_h} \geq R_f^{-n}$ and $R_h^{-n} \left(\frac{R_h}{R_g} \right)^{N_h} \geq R_g^{-n}$ the following inequality implies the upper inequality.

$$\begin{aligned}
&R_f^{-n}(A_f + n^{\ell_f} B_f) + R_g^{-n}(A_g + n^{\ell_g} B_g) \\
&\leq R_f^{-n} A_f + R_g^{-n} A_g + K^{\ell_g} R_g^{-n} B_g \left(1 - \frac{\ell_g}{\ell_f} \right) + n^{\ell_f} \left(R_f^{-n} B_f + R_g^{-n} B_g \frac{\ell_g}{\ell_f K^{\ell_f - \ell_g}} \right) \\
&\Leftrightarrow n^{\ell_g} - n^{\ell_f} \frac{\ell_g}{\ell_f K^{\ell_f - \ell_g}} \leq K^{\ell_g} \left(1 - \frac{\ell_g}{\ell_f} \right)
\end{aligned}$$

The polynomial $p(x) = x^{\ell_g} - x^{\ell_f} \frac{\ell_g}{\ell_f K^{\ell_f - \ell_g}}$ has its maximum at K. So it suffices to check the upper inequality for $n = K$ and one easily sees that this is true.

For the case $K \leq N_h$ we procede in the same way as above, but set

$$A_h = \left(\frac{R_h}{R_f} \right)^{N_h} A_f + \left(\frac{R_h}{R_g} \right)^{N_h} A_g + N_h^{\ell_g} \left(\frac{R_h}{R_g} \right)^{N_h} B_g \left(1 - \frac{\ell_g}{\ell_f} \left(\frac{N_h}{K} \right)^{\ell_f - \ell_g} \right).$$

Which makes $R_h^{-n}(A_h + n^{\ell_h} B_h)$ only smaller. So we get the inequality

$$n^{\ell_g} - n^{\ell_f} \frac{\ell_g}{\ell_f K^{\ell_f - \ell_g}} \leq N_h^{\ell_g} \left(1 - \frac{\ell_g}{\ell_f} \left(\frac{N_h}{K} \right)^{\ell_f - \ell_g} \right).$$

But the polynomial p from above is decreasing on $[K, \infty]$ so it suffices to check the upper inequality for $n = N_h$ and one easily sees that this is true. □

Remark 5.1.2. If $\ell_g = 0$ we can assume w.l.o.g $B_g = 0$ and one easily finds a suitable algorithm.

If one chooses K suitable, one can get the following formulas for A_h, B_h

$$\begin{aligned}
A_h &= A'_f + A'_g \\
B_h &= B'_f + B'_g \frac{1}{N_h^{\ell_f - \ell_g}}
\end{aligned}$$

5.2 Multiplication

We are given a power series $f = \sum a_n x^n$ with name $(R_f, c_f, A_f, B_f, \ell_f, N_f)$ and a power series $g = \sum b_n x^n$ with name $(R_g, c_g, A_g, B_g, \ell_g, N_g)$ and want to determine a name $(R_h, c_h, A_h, B_h, \ell_h, N_h)$ for the power series $\sum d_n x^n = h := f \cdot g$.

Listing 5.2.1.: Multiplication algorithm

\\Contract:
\\Input: Name $(R_f, c_f, A_f, B_f, \ell_f, N_f)$ of some power series f, name $(R_g, c_g, A_g, B_g, \ell_g, N_g)$ of some power series g s.t. $\ell_f \geq \ell_g$.
\\Parameter: $K \in \mathbb{N}$
\\Output: Some name $(R_h, c_h, A_h, B_h, \ell_h, N_h)$ of the power series $h := f \cdot g$
$R_h := \min\{R_f, R_g\}$
$c_h(n) := \sum_{k=0}^{\infty} c_f(n) \cdot c_g(n-k)$
$N_h := N_g + N_f$
$\ell_h := \ell_g + \ell_f + 1$
$A'_f := \left(\frac{R_h}{R_f}\right)^{N_h} A_f$, $A'_g := \left(\frac{R_h}{R_g}\right)^{N_h} A_g$, $B'_f := \left(\frac{R_h}{R_f}\right)^{N_h} B_f$, $B'_g := \left(\frac{R_h}{R_g}\right)^{N_h} B_g$
$A_h := \max\Big\{0, A'_f \sum_{k=0}^{N_g-1} |b_k| R_f^k + A'_g \sum_{k=0}^{N_f-1} |a_k| R_g^k$
$\qquad -A'_f A'_g (N_f + N_g - 1) - A'_g B'_f \sum_{k=1}^{N_f-1} k^{\ell_f} - A'_f B'_g \sum_{k=1}^{N_g-1} k^{\ell_g}\Big\}$
$B_h := \frac{B'_f \sum_{k=0}^{N_g-1} |b_k| R_f^k}{N_h^{\ell_g+1}} + \frac{B'_g \sum_{k=0}^{N_f-1} |a_k| R_g^k}{N_h^{\ell_f+1}} + \frac{A'_f A'_g}{N_h^{\ell_f+\ell_g}} + \frac{A'_g B'_f}{N_h^{\ell_g}} + \frac{A'_f B'_g}{N_h^{\ell_f}} + \frac{\ell_f^{\ell_f} \ell_g^{\ell_g}}{(\ell_f+\ell_g)^{\ell_f+\ell_g}}$
Return $(R_h, c_h, A_h, B_h, \ell_h, N_h)$

Remark 5.2.1. In fact the definition of A_h and B_h are in general not in $\mathbb{Q}$, but we can also use any upper bound for those.

Lemma 5.2.2. *Let $\sum_{i=0}^{\ell} \alpha_i n^i$ be a polynomial in n with positive coefficients, $N \in \mathbb{N}$ and $\alpha := \sum_{i=0}^{\ell} \frac{\alpha_i}{N^{\ell-i}}$, then for all $n \geq N$*

$$\alpha n^\ell \geq \sum_{i=0}^{\ell} \alpha_i n^i \tag{5.1}$$

Proof. Fix some $n \in \mathbb{N}$ with $n \geq N$. Consider the difference of the two polynomials:

$$\begin{aligned} \alpha n^\ell - \sum_{i=0}^{\ell} \alpha_i n^i &= \sum_{i=0}^{\ell-1} n^\ell \frac{\alpha_i}{N^{\ell-i}} - n^i \alpha_i \\ &= \sum_{i=0}^{\ell-1} n^i \alpha_i \Big(\Big(\frac{n}{N}\Big)^{\ell-i} - 1\Big) \geq 0 \end{aligned}$$

□

Proposition 5.2.3. *Listing 5.2.1. is correct.*

Proof. From analysis it is known that

$$d_n = \sum_{k=0}^{n} a_k b_{n-k}.$$

So the formula for c_h is correct. It is left to check that for all $n \geq N_h$ $|c_h(n)| \leq R_h^{-n}(A_h + n^{\ell_h} B_h)$. Rather then showing it directly we want to develope this estimation. We adopt the names of the programm for A'_f, A'_g, B'_f, B'_g i.e.

$$A'_f := \left(\frac{R_h}{R_f}\right)^{N_h} A_f, A'_g := \left(\frac{R_h}{R_g}\right)^{N_h} A_g, B'_f := \left(\frac{R_h}{R_f}\right)^{N_h} B_f, B'_g := \left(\frac{R_h}{R_g}\right)^{N_h} B_g$$

So fix an $n \geq N_f + N_g$. We have

$$\begin{aligned}
|d_n| = \left|\sum_{k=0}^{n} a_k b_{n-k}\right| &\leq \sum_{k=0}^{N_f-1} |a_k||b_{n-k}| + \sum_{k=N_f}^{n-N_b} |a_k||b_{n-k}| + \sum_{k=n-(N_g-1)}^{n} |a_k||b_{n-k}| \\
&= \underbrace{\sum_{k=0}^{N_f-1} |a_k||b_{n-k}|}_{I_g} + \underbrace{\sum_{k=0}^{N_g-1} |b_k||a_{n-k}|}_{I_f} + \underbrace{\sum_{k=N_f}^{n-N_g} |a_k||b_{n-k}|}_{II}
\end{aligned}$$

We discuss the terms I_f, I_g, II seperately. One observes that I_f and I_g are of similar form, so we discuss I_f only and I_g is treated analogously.

$$\begin{aligned}
I_f = \sum_{K=0}^{N_g-1} |b_k||a_{n-k}| &\leq \sum_{k=0}^{N_g-1} |b_k|(A_f + B_f(n-k)^{\ell_f}) R_f^{-(n-k)} \\
&= (R_f)^{-n} \left(A_f \sum_{k=0}^{N_g-1} |b_k| R_f^k + B_f \sum_{k=0}^{N_g-1} |b_k| R_f^k (n-k)^{\ell_f} \right) \\
&\leq (R_f)^{-n} \left(A_f \sum_{k=0}^{N_g-1} |b_k| R_f^k + B_f \sum_{k=0}^{N_g-1} |b_k| R_f^k n^{\ell_f} \right) \\
&\leq (R_h)^{-n} \left(\frac{R_h}{R_f}\right)^{N_h} \left(A_f \sum_{k=0}^{N_g-1} |b_k| R_f^k + B_f \sum_{k=0}^{N_g-1} |b_k| R_f^k n^{\ell_f} \right) \\
&= (R_h)^{-n} \left(A'_f \sum_{k=0}^{N_g-1} |b_k| R_f^k + B'_f \sum_{k=0}^{N_g-1} |b_k| R_f^k n^{\ell_f} \right)
\end{aligned}$$

We get an analogously estimation for I_g:

$$I_g \leq (R_h)^{-n} \left(A'_g \sum_{k=0}^{N_f-1} |a_k| R_g^k + B'_g \sum_{k=0}^{N_f-1} |a_k| R_g^k n^{\ell_g} \right)$$

Now we discuss II:

$$
\begin{aligned}
II &= \sum_{k=N_f}^{n-N_g} |a_k||b_{n-k}| \\
&\leq \sum_{k=N_f}^{n-N_g} (R_f)^{-k}(A_f + B_f k^{\ell_f})(R_g)^{-(n-k)}(A_g + B_g(n-k)^{\ell_g}) \\
&\leq \sum_{k=N_f}^{n-N_g} (R_h)^{-k}\left(\frac{R_h}{R_f}\right)^{N_h}(A_f + B_f k^{\ell_f})(R_h)^{-(n-k)}\left(\frac{R_h}{R_g}\right)^{N_h}(A_g + B_g(n-k)^{\ell_g}) \\
&= (R_h)^{-n}\sum_{k=N_f}^{n-N_g}(A'_f + B'_f k^{\ell_f})(A'_g + B'_g(n-k)^{\ell_g}) \\
&\leq (R_h)^{-n}\left(A'_f A'_g\left(\sum_{k=N_f}^{n-N_g} 1\right) + A'_g B'_f\left(\sum_{k=N_f}^{n-N_g} k^{\ell_f}\right) + B'_g A'_f\left(\sum_{k=N_g}^{n-N_f} k^{\ell_g}\right) + B'_f B'_g\left(\sum_{k=N_f}^{n-N_g} k^{\ell_f}(n-k)^{\ell_g}\right)\right)
\end{aligned}
$$

We want to develope for each addend in the last term a polynomial estimation. The first one can be calculated exact as

$$
A'_f A'_g\left(\sum_{k=N_f}^{n-N_g} 1\right) = A'_f A'_g(-N_f - N_g + 1) + A'_f A'_g n.
$$

Now we turn our attention to the next term and use some rather crude estimation

$$
\begin{aligned}
A'_g B'_f\left(\sum_{k=N_f}^{n-N_g} k^{\ell_f}\right) &= A'_g B'_f\left(\sum_{k=1}^{n-N_g} k^{\ell_f} - \sum_{k=1}^{N_f-1} k^{\ell_f}\right) \\
&\leq A'_g B'_f\left(\sum_{k=1}^{n} n^{\ell_f} - \sum_{k=1}^{N_f-1} k^{\ell_f}\right) = A'_g B'_f n^{\ell_f+1} - A'_g B'_f \sum_{k=1}^{N_f-1} k^{\ell_f}
\end{aligned}
$$

We want to argue that at least the exponent $\ell_f + 1$ is optimal in the estimation. For all m large enough we have

$$
A'_g B'_f\left(\sum_{k=1}^{m-N_g} k^{\ell_f} - \sum_{k=1}^{N_f-1} k^{\ell_f}\right) \geq A'_g B'_f\left(\sum_{k=\frac{m}{2}}^{m-N_g}\left(\frac{m}{2}\right)^{\ell_f} - \sum_{k=1}^{N_f-1} k^{\ell_f}\right) = A'_g B'_f\left(\frac{m}{2}\right)^{\ell_f} + O(m^{\ell_f}).
$$

This shows the remark. A similiar treatment can be done with the third term and obtain:

$$
A'_f B'_g\left(\sum_{k=N_g}^{n-N_f} k^{\ell_g}\right) \leq A'_f B'_g n^{\ell_g+1} - A'_f B'_g \sum_{k=1}^{N_g-1} k^{\ell_g}
$$

So the last term remains. We have $k^{\ell_f}(n-k)^{\ell_g}$ is biggest for $k = n\frac{\ell_f}{\ell_f+\ell_g}$. So we have

$$\begin{aligned}\sum_{k=N_f}^{n-N_g} k^{\ell_f}(n-k)^{\ell_g} &\leq \sum_{k=1}^{n} k^{\ell_f}(n-k)^{\ell_g} \\ &\leq \sum_{k=1}^{n} \left(\frac{\ell_f}{\ell_f+\ell_g}n\right)^{\ell_f}\left(\frac{\ell_g}{\ell_f+\ell_g}n\right)^{\ell_g} \\ &\leq \frac{\ell_f^{\ell_f}\ell_g^{\ell_g}}{(\ell_f+\ell_g)^{\ell_f+\ell_g}} n^{\ell_f+\ell_g+1}\end{aligned}$$

Sadly the exponent in the upper estimation is not optimal since

$$\sum_{k=1}^{n} k^{\ell_a}(n-k)^{\ell_b} = n^{\ell_a+\ell_b}\sum_{k=1}^{n}\left(\frac{k}{n}\right)^{\ell_a}\left(1-\frac{k}{n}\right)^{\ell_b} \overset{\text{for large } n}{\approx} n^{\ell_a+\ell_b}\int_0^1 x^{\ell_a}(1-x)^{\ell_b}.$$

So at least here is some optimisation possible. Now we have estimated all three terms I_f, I_g, II if we collect all the constant terms (except the R_h^{-n}) we get a restriction for A_h:

$$A_h \geq A'_f \sum_{k=0}^{N_g-1} |b_k| R_f^k + A'_g \sum_{k=0}^{N_f-1} |a_k| R_g^k - A'_f A'_g (N_f+N_g-1) - A'_g B'_f \sum_{k=1}^{N_f-1} k^{\ell_f} - A'_f B'_g \sum_{k=1}^{N_g-1} k^{\ell_g}.$$

Which is obviously true by definition of A_h.

Now we gather all the terms which depend on n and get an constraint for $A_h n^{\ell_f+\ell_g+1}$:

$$\begin{aligned}B_h n^{\ell_f+\ell_g+1} \geq\ & n^{\ell_f}\left(B'_f \sum_{k=0}^{N_g-1}|b_k|R_f^k\right) + n^{\ell_g}\left(B'_g\sum_{k=0}^{N_f-1}|a_k|R_g^k\right) + n\left(A'_f A'_g\right) \\ & + n^{\ell_f+1}\left(A'_g B'_f\right) + n^{\ell_g+1}\left(A'_f B'_g\right) + n^{\ell_f+\ell_g+1}\left(\frac{\ell_f^{\ell_f}\ell_g^{\ell_g}}{(\ell_f+\ell_g)^{\ell_f+\ell_g}}\right)\end{aligned}$$

We can apply on the latter polynomial lemma 5.2.2 and get the formula for B_h:

$$\begin{aligned}B_h n^{\ell_f+\ell_g+1} = n^{\ell_f+\ell_g+1}\Bigg(&\frac{B'_f\sum_{k=0}^{N_g-1}|b_k|R_f^k}{N_h^{\ell_g+1}} + \frac{B'_g\sum_{k=0}^{N_f-1}|a_k|R_g^k}{N_h^{\ell_f+1}} + \frac{A'_f A'_g}{N_h^{\ell_f+\ell_g}} + \frac{A'_g B'_f}{N_h^{\ell_g}} + \frac{A'_f B'_g}{N_h^{\ell_f}} \\ &+ \frac{\ell_f^{\ell_f}\ell_g^{\ell_g}}{(\ell_f+\ell_g)^{\ell_f+\ell_g}}\Bigg)\end{aligned}$$

□

5.3 Differentiation

We are given a power series $f = \sum a_n x^n$ with name $(R_f, c_f, A_f, B_f, \ell_f, N_f)$ and want to determine a name $(R_h, c_h, A_h, B_h, \ell_h, N_h)$ for the power series $h = f'$. Therefore we have two methods. The first method is a straightforward calculation, but has the drawback that B_h consists of a mixing of A_f and B_f.

Listing 5.3.1.: First differentiation algorithm

```
1  \\Contract:
2  \\Input:  Name (R_f, c_f, A_f, B_f, ℓ_f, N_f) of some power series f
3  \\Output: Some name (R_h, c_h, A_h, B_h, ℓ_h, N_h) of the power series
       h := f'
4     R_h := R_f
5     c_h(n) := (n + 1)c_f(n + 1)
6     N_h := max{0, N_f − 1}
7     if(ℓ_f = 0 ∨ B_f = 0){
8        ℓ_h = 1
9        A_h := (A_f + B_f)/R_h
10       B_h := A_f/R_h
11    }
12    if(N_h = 0 ∧ ℓ ≠ 0 ∧ B_g ≠ 0)){
13       ℓ_h := ℓ_f + 1
14       A_h := A_f/R_h + B_f/R_h
15       B_h := (B_f 2^{ℓ_h} + A_f − B_f)/R_h
16    }
17    if((N_h ≠ 0 ∧ ℓ ≠ 0 ∧ B_g ≠ 0)){
18       ℓ_h := ℓ_f + 1
19       A_h := A_f/R_h + B_f/R_h
20       B_h := (B_f(N_h+1)^{ℓ_h} + A_f N_h − B_f)/(R_h N_h^{ℓ_h})
21    }
22    Return (R_h, c_h, A_h, B_h, ℓ_h, N_h)
```

The next program does not have this mixing behaviour (except for $\ell = 0 \wedge B = 0$ but then the problem discussed in section 3.4 does not occur), but is harder to compute.

Listing 5.3.2.: Second differentiation algorithm

```
1  \\Contract:
2  \\Input:  Name (R_f, c_f, A_f, B_f, ℓ_f, N_f) of some power series f
3  \\Output: Some name (R_h, c_h, A_h, B_h, ℓ_h, N_h) of the power series
       h := f'
```

4 $R_h := R_f$

5 $c_h(n) := (n+1)c_f(n+1)$

6 $N_h := \max\{0, N_f - 1\}$

7 `if (`$\ell_f = 0 \vee B_f = 0$`) {`

8 $\quad \ell_h = 1$

9 $\quad A_h := \frac{A_f + B_f}{R_h}$

10 $\quad B_h := \frac{A_f}{R_h}$

11 `}`

12 `if (`$N_h = 0 \wedge \ell \neq 0 \wedge B_g \neq 0$`) {`

13 $\quad \ell_h := \ell_f + 1$

14 $\quad A_h := \frac{A_f}{R_f} + \frac{B_f}{R_f} + \frac{A_f}{R_f}\sqrt[\ell_h - 1]{\frac{A}{\ell_h B}}(1 - \frac{1}{\ell_h})$

15 $\quad B_h := \frac{B}{R} 2^{\ell_h}$

16 `}`

17 `if (`$(N_h \neq 0 \wedge \ell \neq 0 \wedge B_g \neq 0)$`) {`

18 $\quad \ell_h := \ell_f + 1$

19 $$A_h := \begin{cases} \frac{A_f}{R_f} + \frac{B_f}{R_f} + \frac{A_f}{R_f}\sqrt[\ell_h - 1]{\frac{AN_h^{\ell_h}}{\ell_h B}}(1 - \frac{1}{\ell_h}) & \texttt{if } 1 \leq \frac{A_f N_h}{B_f \ell_h} \\ \frac{A_f(N_h+1)}{R_h} & \texttt{else} \end{cases}$$

20 $\quad B_h := \frac{B_f}{R_h}\left(\frac{N_h+1}{N_h}\right)^{\ell_h}$

21 `}`

22 `Return` $(R_h, c_h, A_h, B_h, \ell_h, N_h)$

Before we proof the correctness of these algorithms we state a technical lemma.

Lemma 5.3.1. *Let $\ell, N \in \mathbb{N}$ with $N \geq 1$. Then for all $n \in N$ with $n \geq N$*

$$\left(\frac{N+1}{N}\right)^{\ell} n^{\ell} - (n+1)^{\ell} \geq \left(\frac{n}{N}\right)^{\ell} - 1 \geq 0$$

Proof. Since $n \geq N$ we have $\left(\frac{n}{N}\right)^{j} - 1 \geq 0$

$$\begin{aligned} \left(\frac{N+1}{N}\right)^{\ell} n^{\ell} - (n+1)^{\ell} &= \sum_{i=0}^{\ell} \binom{\ell}{i} \left(\frac{n^{\ell}}{N^{\ell-i}} - n^{i}\right) \\ &= \sum_{i=0}^{\ell} \binom{\ell}{i} n^{i} \left(\left(\frac{n}{N}\right)^{\ell-i} - 1\right) \geq \binom{\ell}{0} n^{0} \left(\left(\frac{n}{N}\right)^{\ell-0} - 1\right) \\ &= \left(\frac{n}{N}\right)^{\ell} - 1 \end{aligned}$$

□

Proposition 5.3.2. *Listing 5.3.1. and 5.3.2. are correct.*

Proof. We have

$$f' = \sum_{n=0}^{\infty} a_{n+1}(n+1)x^n$$

So the formula for c_h is correct in both listings. Now for the estimation of the coefficients. For all $n \geq N_h$ holds $n+1 \geq N_f$ and so:

$$|c_h(n)| = |a_{n+1}(n+1)| \leq \frac{1}{R_f^{n+1}}(A_f + B_f(n+1)^{\ell_f})(n+1) = \frac{1}{R_f^n}\left(\frac{B_f}{R_f}(n+1)^{\ell_f+1} + \frac{A_f}{R_f}n + \frac{A_f}{R_f}\right)$$
$$= \frac{1}{R_h^n}\left(\frac{B_f}{R_h}(n+1)^{\ell_h} + \frac{A_f}{R_h}n + \frac{A_f}{R_h}\right)$$

We first show that listing 5.3.1. is correct. Therefore we verify the representations generated in the different cases.

Let $\ell_f = 0 \vee B_f = 0$ then for all $n \geq N_h$

$$\frac{1}{R_h^n}\left(\frac{B_f}{R_h}(n+1)^{\ell_h} + \frac{A_f}{R_h}n + \frac{A_f}{R_h}\right) = \frac{1}{R_h^n}\left(\frac{A_f}{R_h}n + \frac{A_f + B_f}{R_h}\right) = \frac{1}{R_h^n}\left(B_h n^{\ell_h} + A_h\right)$$

Now let $(N_h \neq 0 \wedge \ell \neq 0 \wedge B_g \neq 0)$ then for all $n \geq N_h$

$$\begin{aligned}\frac{1}{R_h^n}\left(\frac{B_f}{R_h}(n+1)^{\ell_h} + \frac{A_f}{R_h}n + \frac{A_f}{R_h}\right) &\leq \frac{1}{R_h^n}\left(\frac{B_f}{R_h}\left(\frac{(N_h+1)^{\ell_h}-1}{N_h^{\ell_h}}n^{\ell_h} + 1\right) + \frac{A_f}{R_h}n + \frac{A_f}{R_h}\right)\\ &= \frac{1}{R_h^n}\left(\frac{B_f}{R_h}\frac{(N_h+1)^{\ell_h}-1}{N_h^{\ell_h}}n^{\ell_h} + \frac{A_f}{R_h}n + \frac{A_f}{R_h} + \frac{B_f}{R_h}\right)\\ &\leq \frac{1}{R_h^n}\left(\left(\frac{B_f}{R_h}\frac{(N_h+1)^{\ell_h}-1}{N_h^{\ell_h}} + \frac{A_f}{R_h N_h^{\ell_h-1}}\right)n^{\ell_h} + \frac{A_f}{R_h} + \frac{B_f}{R_h}\right)\\ &= \frac{1}{R_h^n}\left(B_h n^{\ell_h} + A_h\right)\end{aligned}$$

Now let $(N_h = 0 \wedge \ell \neq 0 \wedge B_g \neq 0)$. We reduce this case to the previous one. Take A_h, B_h such as $N_h = 1$. Then we know that

$$\frac{1}{R_h^n}\left(\frac{B_f}{R_h}(n+1)^{\ell_h} + \frac{A_f}{R_h}n + \frac{A_f}{R_h}\right) \leq \frac{1}{R_h^n}\left(B_h n^{\ell_h} + A_h\right)$$

for all $n \geq 1$. One easily checks that the upper inequality also holds for $n = 0$.

Now we show that listing 5.3.2. is correct. Therefore we verify the representations generated in the different cases.

Let $\ell_f = 0 \vee B_f = 0$ then we can apply the same reasoning above.

Now let $(N_h \neq 0 \wedge \ell \neq 0 \wedge B_g \neq 0)$ then for all $n \geq N_h$

$$\begin{aligned}\frac{1}{R_h^n}\left(\frac{B_f}{R_h}(n+1)^{\ell_h} + \frac{A_f}{R_h}n + \frac{A_f}{R_h}\right) &\leq \frac{1}{R_h^n}\left(\frac{B_f}{R_h}\left(\frac{(N_h+1)^{\ell_h}-1}{N_h^{\ell_h}}n^{\ell_h}+1\right) + \frac{A_f}{R_h}n + \frac{A_f}{R_h}\right)\\ &= \frac{1}{R_h^n}\left(\frac{B_f}{R_h}\frac{(N_h+1)^{\ell_h}}{N_h^{\ell_h}}n^{\ell_h} + \left(\frac{A_f}{R_h}n - \frac{B_f}{R_h N_h^{\ell_h}}n^{\ell_h}\right) + \frac{A_f}{R_h} + \frac{B_f}{R_h}\right)\end{aligned}$$

The expression

$$\left(\frac{A_f}{R_h}n - \frac{B_f}{R_h N_h^{\ell_h}}n^{\ell_h}\right)$$

attains its maximum at $n_{max} := N_h \sqrt[\ell_h - 1]{\frac{A_f N_h}{B_f \ell_h}}$. So by either pluging in n_{max} if $N_h \leq N_h \sqrt[\ell_h - 1]{\frac{A_f N_h}{B_f \ell_h}}$ $(\Leftrightarrow 1 \leq \frac{A_f N_h}{B_f \ell_h})$ or N_h in the other case, we get the formula of the listing.

Now let $(N_h = 0 \wedge \ell \neq 0 \wedge B_g \neq 0)$. We reduce this case to the previous one. Take A_h, B_h such as $N_h = 1$. Then we know that

$$\frac{1}{R_h^n}\left(\frac{B_f}{R_h}(n+1)^{\ell_h} + \frac{A_f}{R_h}n + \frac{A_f}{R_h}\right) \leq \frac{1}{R_h^n}\left(B_h n^{\ell_h} + A_h\right)$$

for all $n \geq 1$. One easily checks that the upper inequality also holds for $n = 0$. □

5.4 Integration

We are given a power series $f = \sum a_n x^n$ with name $(R_f, c_f, A_f, B_f, \ell_f, N_f)$ and want to determine a name $(R_h, c_h, A_h, B_h, \ell_h, N_h)$ for a power series h, s.t. $h' = f$ this means to find an integral for f. Since the integral is uniquely except for a constant we norm h, s.t. $h(0) = 0$.

Listing 5.4.1.: Integration

1 \\Contract:
2 \\Input: Name $(R_f, c_f, A_f, B_f, \ell_f, N_f)$ of some power series f
3 \\Output: Some name $(R_h, c_h, A_h, B_h, \ell_h, N_h)$ of the power series h s.t. $h' = f$ and $h(0) = 0$
4 $R_h := R_f$
5 $c_h(n) := \begin{cases} \frac{c_f(n-1)}{n} & \text{if } n \geq 1 \\ 0 & \text{if } n = 0 \end{cases}$
6 $N_h := N_f + 1$
7 $\ell_h := \ell_f - 1$
8 $A_h := \frac{A_f R_h}{N_h}$
9 $B_h := B_f R_h$
10 Return $(R_h, c_h, A_h, B_h, \ell_h, N_h)$

Proposition 5.4.1. *Listing 5.4.1. is correct.*

We omit this straight forward proof.

Chapter 6

Conclusion

So we have created a fundament for a new datatype for iRRAM. If this can be used efficiently in practice, should be tested in practice. For instance the discussion of the efficiency of the first step of evaluation was based purely on the algorithm for finding M_{up}. This is comprehensible since we cannot make predictions for the binary search which is in fact involved since it depends on the representation. So in the worst case the binary search will not find a lower value for M_{up}, altough there exist one. But it may also be in other cases very beneficial.

The algorithm for multiplication is not very good developed. But there are things, one is not able to improve. So it seems that multiplication is theoretical doable, but not feasible.

Bibliography

[JMHM08] Jeffrey L. Hirst John M. Harris and Michael J. Mossinghoff. *Combinatorics and Graph Theory*. Springer, New York, second springer printing edition, 2008.

[Mue08] Norbert Mueller. The irram. Website, 2008. Available online at http://irram.uni-trier.de/irram.ps visited on Mai 13th 2012.

[RMCJ96] G. H. Gonnet R. M. Corless, D. E. G. Hare and D. J. Jeffrey. On the lambert w function. Technical report, Department of Applied Mathematics University of Western Ontario, Institut fuer Wissenschaftliches Rechnen ETH Zuerich, Symbolic Computation Group University of Waterloo, Department of Computer Science Stanford University, 1996.

[Wei00] Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, first springer printing edition, 2000.